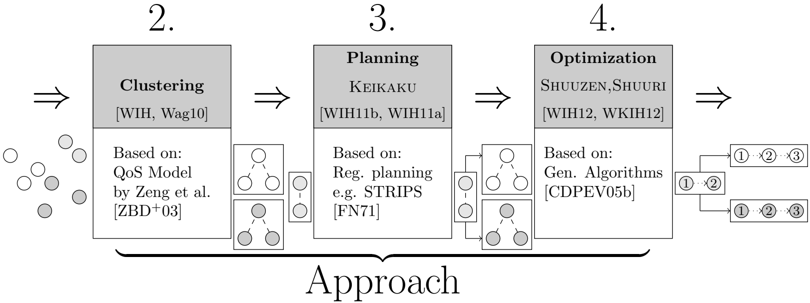
サービスコンピューティング研究会

Robust Workflows by Applying Functional Clustering on Multi-Objective Service Composition
(多目的のサービス合成における，機能クラスタリングの適用によるロバストなワークフローの構築)

Florian Wagner (NII)
Honiden Laboratory

平成25年3月15日

1. Introduction

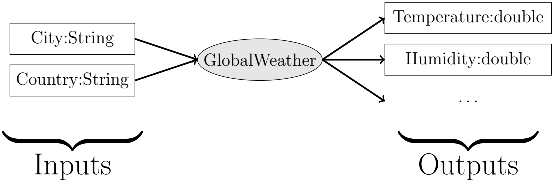


5. Summary

Service-Oriented Computing

- Services **encapsulate** business logic
- **Loosely**-coupled, **flexible** components
- Interface description documents:
  1. **Functional** IOPE interface
     - ⇒ **I**nputs, **O**utputs, **P**reconditions, **E**ffects
     - ⇒ Semantics: associate concepts to IOPE
  2. **Non-functional** Service-Level Agreement
     - ⇒ Contains Quality-of-Service (QoS)
     - ⇒ Specified by the provider
     - ⇒ Example: price, response time, ...

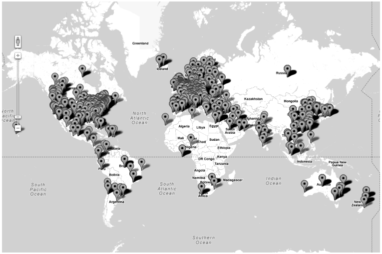
GlobalWeather service[1]:



[1] `http://www.webservicex.net/globalweather.asmx`

Services in Practice

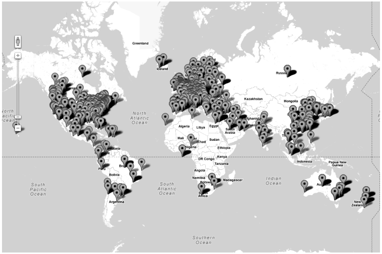
- Web services: **20,000** [ZZL10] to **30,000** [Tec12]:



- Successfully applied in **many companies**, such as eBay, Amazon [DPPS⁺08, ZDN12], IBM, DreamWorks, HP [ZDN12], Winterthur, Deutsche Post [KBS04]
  - **Credit Suisse** [Mur11]: "All applications on the Swiss Platform offer and/or consume services"
    - ⇒ 1000 services, 400Mio. calls per month.
    - ⇒ Research challenges: existence of 1000s of services, fault-tolerant design, varying service interfaces
  - **Twitter API** invoked 15 billion times a day, **Google** and **Facebook** 5 billion [LG11]

Services in Practice

- Web services: **20,000** [ZZL10] to **30,000** [Tec12]:

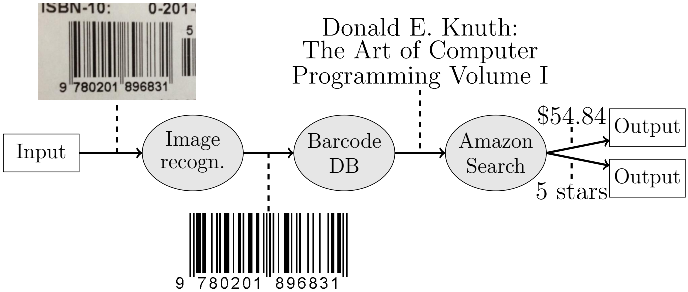


- Successfully applied in **many companies**, such as eBay, Amazon [DPPS+08, ZDN12], IBM, DreamWorks, HP [ZDN12], Winterthur, Deutsche Post [KBS04]
  - **Credit Suisse** [Mur11]: "All applications on the Swiss Platform offer and/or consume services"
    - ⇒ 1000 services, 400Mio. calls per month.
    - ⇒ Research challenges: existence of 1000s of services, fault-tolerant design, varying service interfaces
  - **Twitter API** invoked 15 billion times a day, **Google** and **Facebook** 5 billion [LG11]

Service Composition

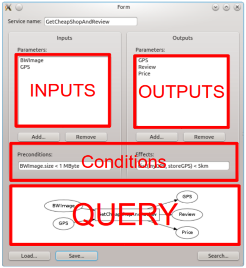
- **Key benefit** of services: generate new software
  - Services are arranged in workflows, described with BPEL
  - Executed with BPEL engines → no additional code necessary (in theory)

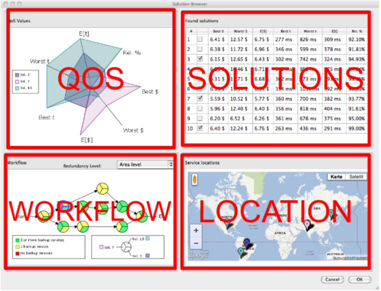
⇒ **Goal**: Automatic service composition



Scenario: Mobile Service Store

- **User** of the system: service broker
- Combines **existing services** with his/her **own** services
- **Interesting services**: database access (IP to GPS), changing data (weather, stocks), data-intensive (genome alignment)





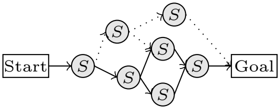
Steps:

1. Definition of the **parameters** and **QoS preferences**
2. System **proposes** set of **solutions**
3. User **compares** the **solutions**, **picks** one
4. Composite service is **registered** at service directory

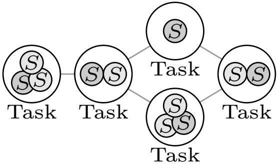
Service Composition Approaches

Two main approaches:

1. **Planning** [WPS+03, KG06, LKS08]
   - Starts from scratch
   - Applies AI planning tool
   - **Drawbacks:**
     - ⇒ **Scalability** issues: $\mathcal{O}(S^W)$ (**C1**)
     - ⇒ Insufficient coverage of **QoS aspects** (Challenge **C2**)



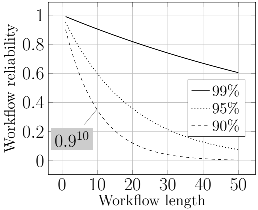
2. **Selection** [ZBD+03, CDPEV05a, YZL07]
   - Refines workflow templates
   - Faster, QoS-aware: T^W, $T \ll S$
   - **Drawbacks**:
     - ⇒ No flexibility ⇒ **template required** (**C3**)
     - ⇒ Simplified modeling by Zeng [ZBD+03]: Services in task are assumed to be equal (**C4**)



Service Composition Approaches

Both fail to achieve an insufficient reliability:

- With **growing** workflow **length**, service **crashes** become more likely
- Most related approaches rely on **ad-hoc replanning** during runtime:
  - Works **only** if **suitable backup** services exist
  - Might cause **additional costs** for un-doing certain actions
  - No predictability:
    - ⇒ Backup services have **worse QoS**
    - ⇒ **Response time** and, potentially, **price** of the failed service(s) increase the costs
- Constitutes **challenge C5**

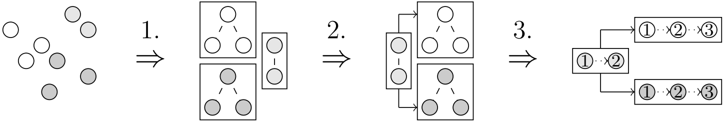


$$Rel(WF) = \prod_{S \in WF} Rel(S)$$

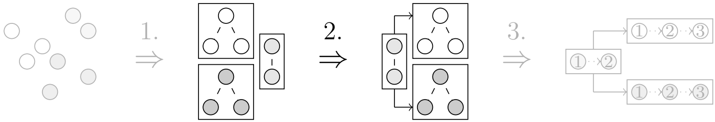
$$\Rightarrow 0.9^{10} \approx 0.35$$

Research Theme

- **Observation**: for a certain purpose (e.g. book hotel room), multiple services exist
  - Developed **independently**
  - **Functionally similar** but not equivalent
  - **Naïve integration** of planning with selection **infeasible**
    - ⇒ Requires identical functional interfaces
    - ⇒ Planning has to consider QoS
- **Our proposal**: integrate planning with selection by
  1. **Clustering** the existing services
  2. **Planning** to create templates, not workflows
  3. **Selection** to refine templates to workflows

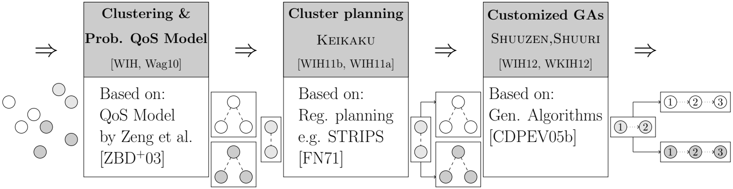


Research Theme



- **Problems**:
  - Planning with clusters is "**fuzzy**": **which** clusters can be **combined**?
  - How to compute **templates** that contain services **with "promising"** QoS?
- **Advantages**:
  - ⇒ Addresses challenges **C2** and **C3**
    - Planning (2.) generates functional template ⇒ flexible (C3)
    - Selection (3.) optimizes the QoS ⇒ complex QoS (C2)
  - ⇒ **Clustering** is **basis for** tackling the **other challenges**
  - ⇒ **Encodes domain knowledge**
    - ⇒ Used in planning & selection

Approach



Holistic Approach

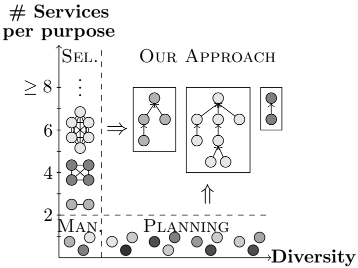
- **Issues** in:
  - **Planning**:
    - **C1** Scalability
    - **C2** QoS Aspects
  - **Selection**:
    - **C3** Flexibility
    - **C4** Functional Diversity
  - **Both**:
    - **C5** Reliability

- **Addressed** by:
  - Combining planning and selection, we address **C2** and **C3**
  - In the following, we focus on:
    - **C1** Scalability
    - **C4** Functional Diversity
    - **C5** Reliability

Assumptions

Assumptions

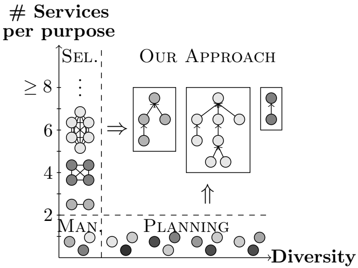
1. **Semantic annotations**
   - Interfaces **annotated**
   - Otherwise no **planning**
2. **Functionally related**
   - We know **which** services can be **combined**
   - Otherwise no **clustering**



3. **QoS known** at **any time**
   - QoS are **claimed by** the **provider**
     - ⇒ Violations → **penalty mechanisms**
     - ⇒ Alternative: (continuous) **monitoring** or **prediction** applied
   - **Input-independent QoS**: backup slides (page 106)
4. **Large number** of **services**
   - Otherwise **scalability not an issue**
   - **Possibilities** for optimization phase **limited**

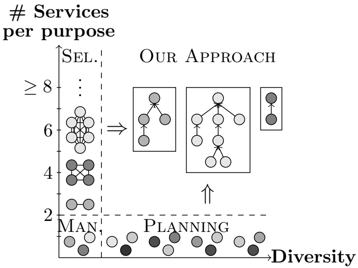
Assumptions

1. **Semantic annotations**
   - Interfaces **annotated**
   - Otherwise no **planning**
2. **Functionally related**
   - We know **which** services can be **combined**
   - Otherwise no **clustering**



3. **QoS known** at **any time**
   - QoS are **claimed by** the **provider**
     - ⇒ Violations → **penalty mechanisms**
     - ⇒ Alternative: (continuous) **monitoring** or **prediction** applied
   - **Input-independent QoS**: backup slides (page 106)
4. **Large number** of **services**
   - Otherwise **scalability not an issue**
   - **Possibilities** for optimization phase **limited**

Assumptions



❶ **Semantic annotations**

- Interfaces **annotated**
- Otherwise no **planning**

❷ **Functionally related**

- We know **which** services can be **combined**
- Otherwise no **clustering**

❸ **QoS known** at **any time**

- QoS are **claimed by** the **provider**
  - ⇒ Violations → **penalty mechanisms**
  - ⇒ Alternative: (continuous) **monitoring** or **prediction** applied
- **Input-independent QoS**: backup slides (page 106)

❹ **Large number** of **services**

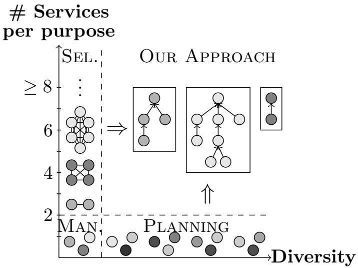
- Otherwise **scalability not an issue**
- **Possibilities** for optimization phase **limited**

Assumptions

Assumptions

❶ **Semantic annotations**
- Interfaces **annotated**
- Otherwise no **planning**

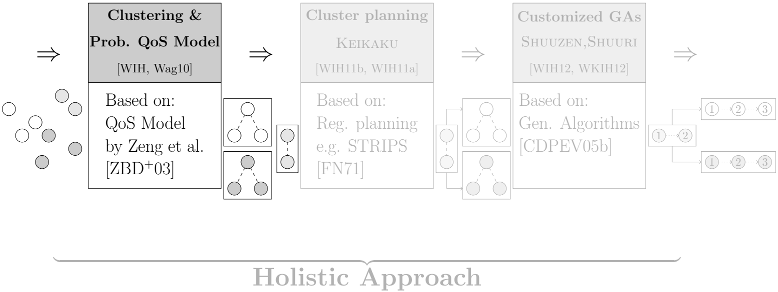
❷ **Functionally related**
- We know **which** services can be **combined**
- Otherwise no **clustering**



❸ **QoS known** at **any time**
- QoS are **claimed by** the **provider**
  - ⇒ Violations → **penalty mechanisms**
  - ⇒ Alternative: (continuous) **monitoring** or **prediction** applied
- **Input-independent QoS**: backup slides (page 106)

❹ **Large number** of **services**
- Otherwise **scalability not an issue**
- **Possibilities** for optimization phase **limited**

2. Service Clustering



Comparison with Related Work

- **Clustering** has been applied to **service discovery** [MPG+08]
  - **Different** service **comparison**, not applicable to composition
  - **No** additional **cached information** or **QoS model** based on the clustering
- In **service composition**, only **QoS-based** clustering algorithms
  - Not applicable to planning

Novelty

- Cluster Representatives
- SEO / Backup Services
- Caching of Service Parameters
- Probabilistic QoS Model

Based On

- Semantic Matchmaking
- QoS Model by [ZBD+03]

Service Clustering - Algorithm

- Services are **compared** with each other:
  - **Exact match**: same node

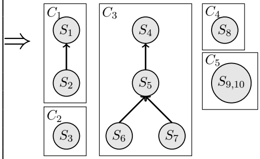
$$S \equiv S' \Leftrightarrow I \equiv I' \wedge O \equiv O' \wedge P \Leftrightarrow P \wedge E \Leftrightarrow E$$

  - **Plugin match**: edge between the nodes (weaker input and / or stronger output)

$$S \sqsubseteq S' \Leftrightarrow I \sqsupseteq I' \wedge O \sqsubseteq O' \wedge P' \Rightarrow P \wedge E \Rightarrow E'$$

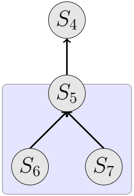
- Results in a **directed-acyclic graph**
- Connected components become the **clusters**
- **Root nodes** become **representatives**
- Takes around 6 sec. for 10,000 services

		Name	Inputs	Outputs
	S_1	BWImgToBarc.	BWImage	Barcode
Data-int.	S_2	ImageToBarcode	Image	Barcode
	S_3	GetProduct	Image	PID
	S_4	EUBarcodeDB	EANBC	PID
DB acc.	S_5	BarcodeToPInfo	Barcode	PID
	S_6	GetD9Info	Barcode	PID_{D9}
	S_7	Prod.Info.	Barcode	PID_{D14}
	S_8	GetReview	PID	Review
Dyn. data	S_9	GetCheapShop	GPS,PID	GPS, Price
	S_{10}	FindLocalShop	GPS,PID	GPS, Price



Service Clustering - Execution Orders

- **Observation**: **services** can be **replaced** with **services** from the **same node** and its **child nodes** (= subcluster):



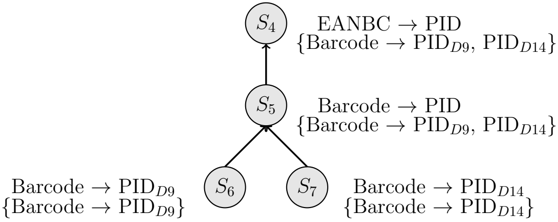
= Subcluster of S_5

- Introduce **service execution orders** (SEO)
  - Determines **which service** is executed
    - ⇒ Arrange services in fronts, then QoS aggregation
  - In case a service **crashes**, the **next service** in line is chosen, e.g. SEO for S_5:

  - Addresses challenge **C5**

Functional Parameter Caching

- Apart from backup services, the structure can be used as **background knowledge** in **planning** & **optimization**
- Helps to **avoid** unnecessary computations (challenge **C1**)



- Nodes **aggregate** parameter **types** of their **subcluster**
- **Super**types in **inputs** and **sub**types in **outputs** are omitted:

Inputs: $\{EANBC, Barcode\}$ $\quad EANBC \sqsubseteq Barcode \Rightarrow \quad \{Barcode\}$

Outputs: $\{PID, PID_{14}\}$ $\quad PID \sqsupseteq PID_{14} \Rightarrow \quad \{PID_{14}\}$

Service Clustering - QoS Model

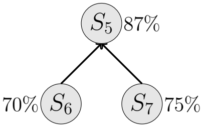
- **Consequences** of backup services: **probabilistic QoS**
- **Goal**: predict the values as closely as possible

  ⇒ Probabilistic QoS model:

  – Reliability of a node:

$$N^{rel} = 1 - \prod_{S \in cluster(N)} (1 - S^{rel})$$

Example:



$$N^{rel} = 1 - \Big(\underbrace{(1-0.87)}_{S_5 \text{ crashes}} \cdot \underbrace{(1-0.7)}_{S_6 \text{ crashes}} \cdot \underbrace{(1-0.75)}_{S_7 \text{ crashes}} \Big) \approx 99\%$$

- **Advantage**: build *reliable* systems with *low-cost* services

Service Clustering - QoS Model

- Introduce for each QoS attribute **three values**:
  1. **Best case**: first service executed successfully
  2. **Average case**:

$$\mathbb{E}[N^{price}] = \underbrace{S_1^{price}}_{\text{Price in case 1}} \cdot \underbrace{S_1^{rel}}_{\text{Probability of case 1}} + \\ + \underbrace{\left(S_1^{price} + S_2^{price}\right)}_{\text{Price in case 2}} \cdot \underbrace{\left((1 - S_1^{rel}) \cdot S_2^{rel}\right)}_{\text{Probability of case 2}} + \\ + \dots$$

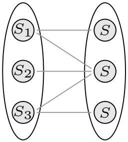
  3. **Worst case**: General idea: all services except for the last one fail [WKIH12]
     - ⇒ Too pessimistic in reality
     - ⇒ **Solution**: Apply Tchebysheff's inequality [WIH]

⇒ In the end, **7 objectives**:

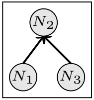
$$\left\{\left(p_{best}, \mathbb{E}[p], p_{worst}\right), \left(t_{best}, \mathbb{E}[t], t_{worst}\right), rel\right\}$$

Service Clustering - Extensions

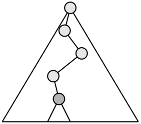
- Clustering **without type concepts** [WKIH12]:
  - No parameter annotations ⇒ clustering **still applicable**
  - Need relation *compatible*:



⇒



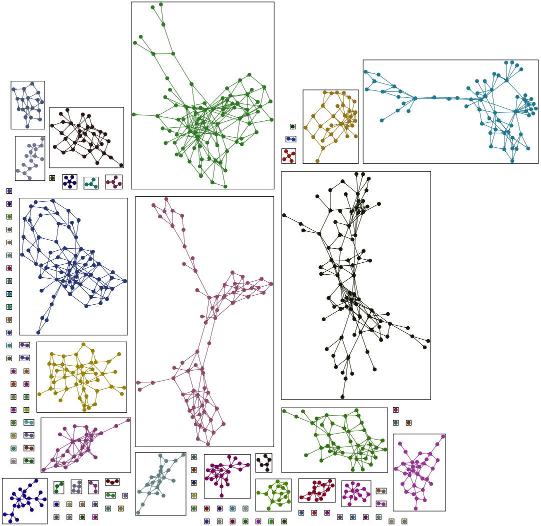
  - Applicable to scenarios with **given workflow template**
    ⇒ "Pure" service selection
- **Dynamic service environment**:
  - Efficient **insertion** of new services described in [MPG⁺08]
  - **Remove** service → **virtual** service
  - **Services or QoS change**: update all **parent nodes**:

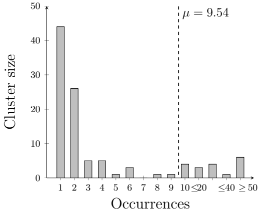


- **Virtual services** [WKIH12] . . . (backup slides)
- **Physical location** [WIH] . . . (backup slides)

Evaluation

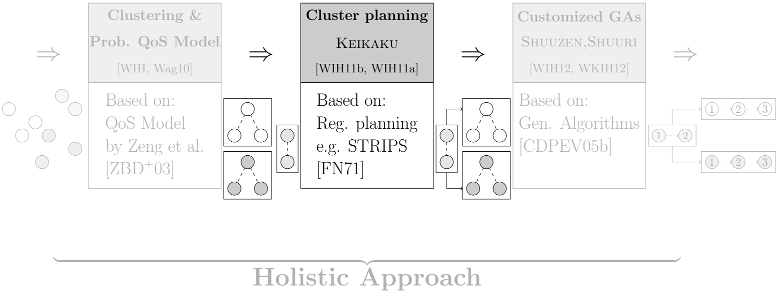
Clustering the OWLS-TC testset[2] (≈ 1,000 services):





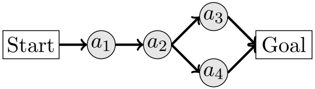
[2] http://www.semwebcentral.org/projects/owls-tc

3. Service Planning



Planning Algorithm

- **Planning**:
  - Given an **initial state** and **goal state**, plus a set of actions
  - Compose actions to establish a path between these states:



- **Service planning**:
  - Services and query are **translated into PDDL**, AI planner such as SHOP2 [WPS+03] or Xplan [KG06] are applied
  - Multiple QoS + constraints → no admissible heuristic
    - ⇒ **Scalability issues**: In each step, S possibilities → solution space is $\mathcal{O}(S^W)$, W unknown (**Challenge C1**)
    - ⇒ **Insufficient coverage** of QoS aspects (**Challenge C2**)

Novelty
- AI Planning on Cluster Level
- QoS-aware Template Gen.

Based On
- AI Regression Planning

Planning Algorithm - QoS Aspects

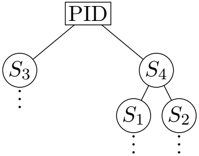
Challenge QoS Aspects (C2):

- **Multiple QoS** must be optimized and **constraints** must be met
- Domain-independent planner **mostly neglect QoS**
- Recently, **hybrid algorithms** have been proposed, such as QSynth [JZH+10]:
  - Won the WS-Challenge in 2009
  - Employs **simplified QoS model** and **ignores constraints**
    - ⇒ Used to evaluate our approach in [WIH11b] (next slides)

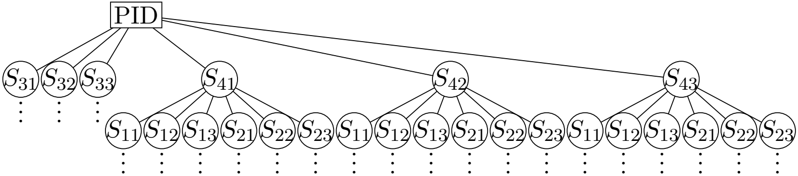
Planning Algorithm - Scalability

Challenge Scalability (C1):

- Many **functionally similar** but not equivalent services exist
  - ⇒ Search tree grows exponentially:



⇓ adding 2 alternatives per service ⇓

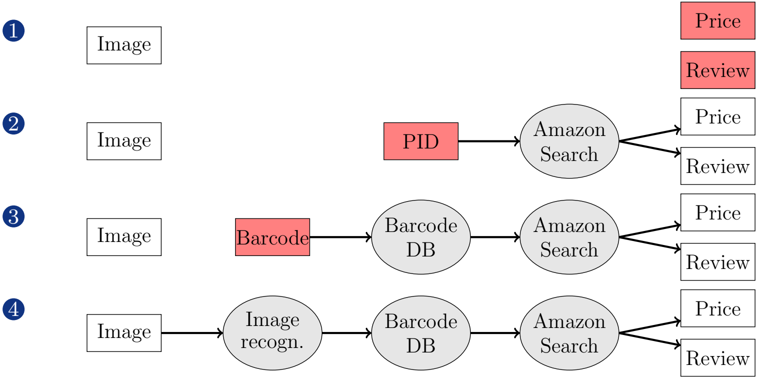


- **Not addressed** by related work in SOC community

Planning Algorithm - Scalability

Algorithm: Regression planning [GNT04] with services

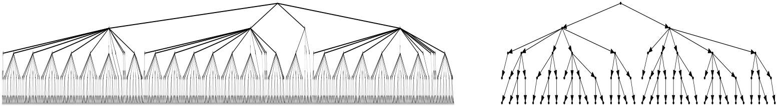
- Start with given goals
- Adding candidate services:



Problem when clusters are used instead of services:

1. When is a cluster **applicable**?
2. How does **adding** a cluster **modify** the set of **open goals**?

Planning Algorithm - Scalability



Service planning Cluster planning

- **Proposal**: **cluster planner** KEIKAKU:
  - Operates on **cluster level** instead of service level
  - Selects "promising" clusters
  - QoS are **optimized in the next stage**
    - ⇒ Multiple QoS and constraints can be considered
  - Consider **only representatives** in the clusters
  - **If** aggregated parameters in representatives **don't match**
    - ⇒ **Omit** entire cluster
    - ⇒ Avoids **unnecessary computations** with similar services ⇒ addresses scalability (Challenge **C1**)
  - **Else**:
    - ⇒ Determine **set of matching services** in the cluster (reverse lookup)
    - ⇒ **Weakest** input constitutes new goal

Example

Current plan:

GPS

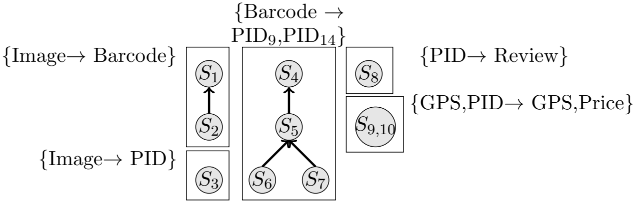
Image

GPS

Price

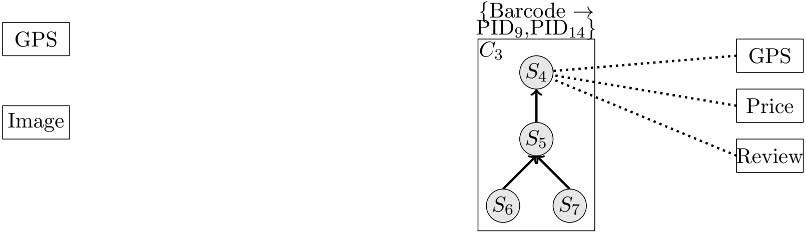
Review

Candidate clusters:

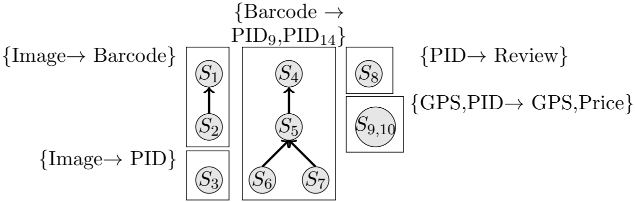


Example

Current plan:

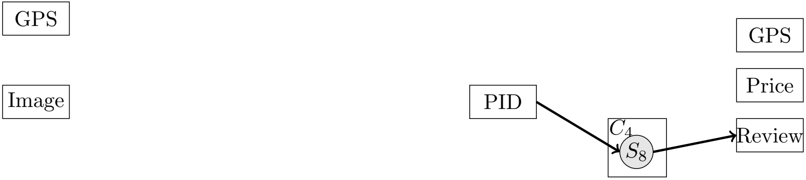


Candidate clusters:

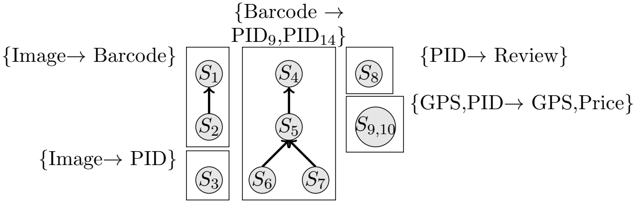


Example

Current plan:

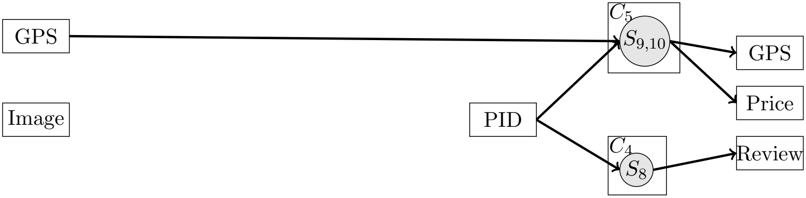


Candidate clusters:

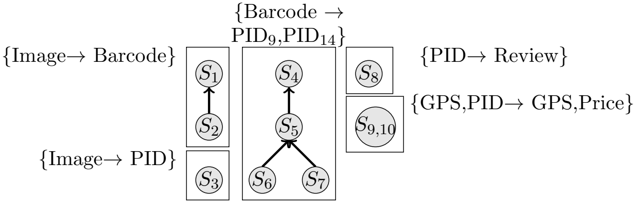


Example

Current plan:

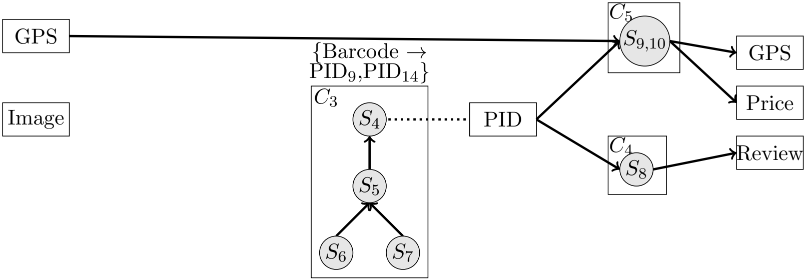


Candidate clusters:

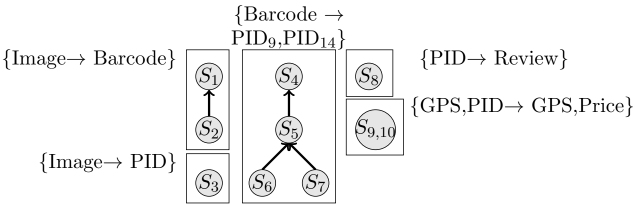


Example

Current plan:

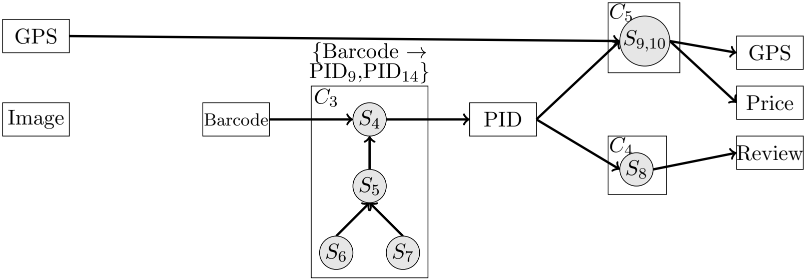


Candidate clusters:

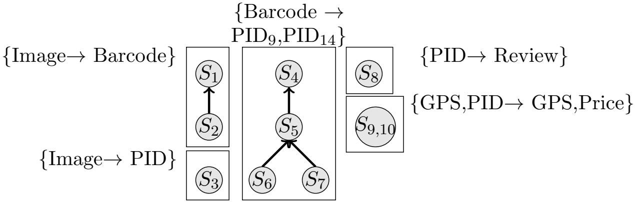


Example

Current plan:

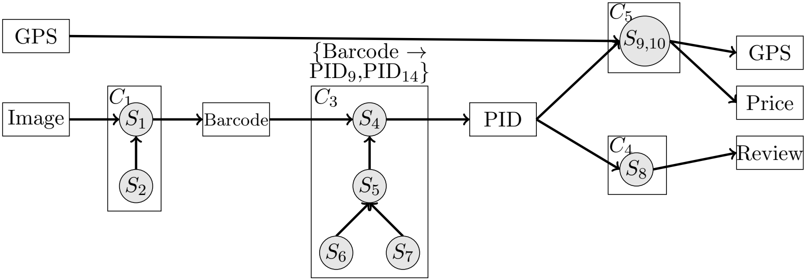


Candidate clusters:

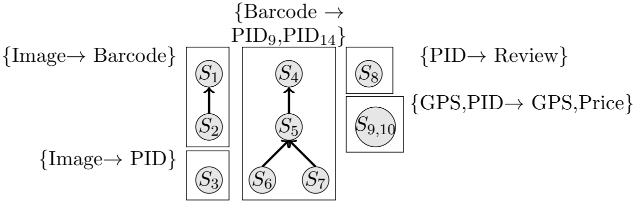


Example

Current plan:

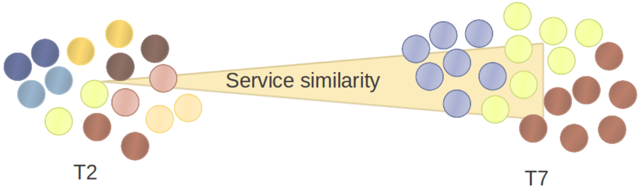


Candidate clusters:



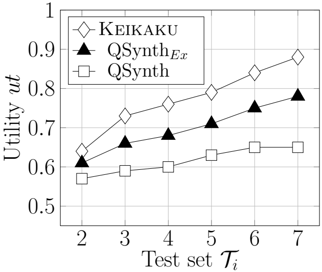
Evaluations - QoS Aspects (C2)

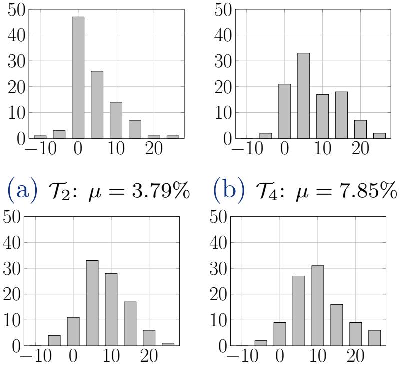
- Used **different test sets** called $T_2, T_3, \ldots, T_7$ containing 1,000 random services
- In every test set T_i for each service, i **similar services** are generated:



- Helps to examine **in which scenario** the KEIKAKU algorithm can be applied (narrow domain, open directory, . . .)
- Compared with **QSynth** [JZH+10], winner of the WS-Challenge 2009
- Added **extension of QSynth** that can handle backup services

Evaluations - QoS Aspects (C2)





(a) $\mathcal{T}_2$: $\mu = 3.79\%$ (b) $\mathcal{T}_4$: $\mu = 7.85\%$

(c) $\mathcal{T}_5$: $\mu = 8.60\%$ (d) $\mathcal{T}_7$: $\mu = 10.1\%$

Figure: ut(Keikaku) - ut(QSynth)

- Services are chosen based on a **simple heuristic**

⇒ **No** real **optimization** phase

- Clearly **outperforms** QSynth, especially when **many services per purpose exist (≥ 3)**

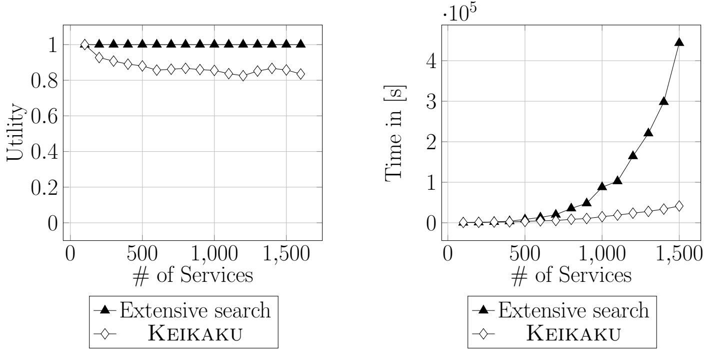
Evaluations - Scalability (C1)

- In the next evaluation, used the **test set generator** from the **WS-challenge**[3]
- Generated **100 services**
- Modified it to **generate test sets similar** to T_2 to T_7
- Compared with an **exhaustive search planner**
- Clusters were refined with a **simple hill-climbing algorithm**

⇒ Applying a GA might improve the results

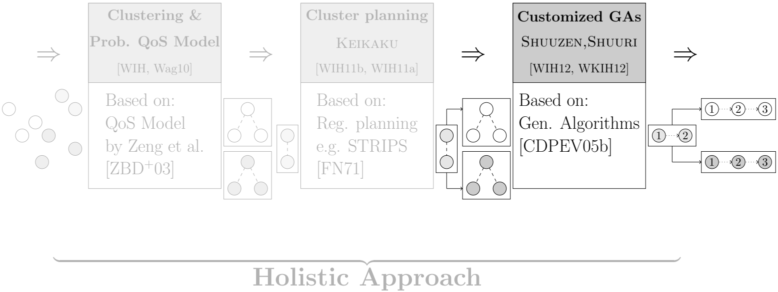
[3] http://ws-challenge.georgetown.edu/wsc10/

Evaluations - Scalability (C1)



- Utility is **near-optimal**
- **Runtime** of the extensive search is **exponential**
- Keikaku planner:
  leverages the similarity of the services

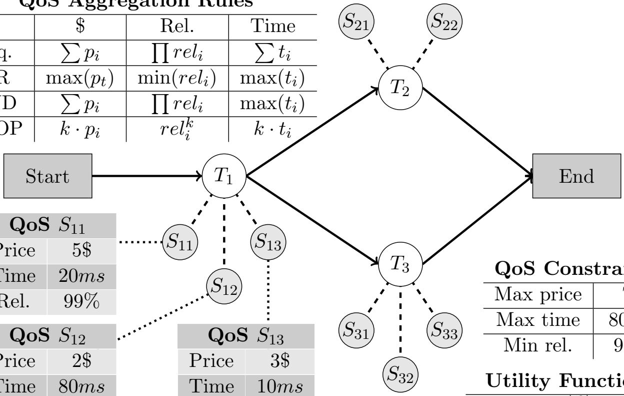
4. Workflow QoS Optimization



QoS-aware Service Selection [ZBD+03]

QoS Aggregation Rules

	\$	Rel.	Time
Seq.	$\sum p_i$	$\prod rel_i$	$\sum t_i$
OR	$\max(p_t)$	$\min(rel_i)$	$\max(t_i)$
AND	$\sum p_i$	$\prod rel_i$	$\max(t_i)$
LOOP	$k \cdot p_i$	rel_i^k	$k \cdot t_i$



QoS S_{11}	
Price	5\$
Time	$20ms$
Rel.	99%

QoS S_{12}	
Price	2\$
Time	$80ms$
Rel.	97%

QoS S_{13}	
Price	3\$
Time	$10ms$
Rel.	95%

QoS Constraints

Max price	7\$
Max time	$80ms$
Min rel.	90%

Utility Function

$$\mu(\mathcal{Q}) = \sum_{i=1}^{|\mathcal{Q}|} w_i \cdot \mathcal{Q}_i$$

- Goals:
  - **Utility** function is maximized
  - **All constraints** are met
- Very **active research field** in the past decade, mostly published on WWW, ICWS, ICSOC, and GECCO

Related Optimization Problems (more on backup slides!)

1. **Multiconstrained Optimal Path Problem**
   - **Problem**: exponential non-dominated paths possible
   - **Heuristics** only of **little help** [YZL07]
2. **Task Scheduling Problem**
   - **Problem**: most TSP algorithms apply **activity list representation**
   - **Only few** algorithms are **efficient**, still not competitive [JMG05]
3. **Multidimension multichoice 0-1 Knapsack Problem**
   - Applied by most related work, covers all aspects
   - **NP-hard** problem, search space: SPT^{WF}, scalability issues (**C1**) ⇒ **heuristics**
   - Both, MMKP and selection problem **tackled by**:
     - ⇒ Integer / Dynamic Programming [ZBD+03, HJHL09]
     - ⇒ Hill-climber [KIH11]
     - ⇒ **Genetic Algorithms** [CDPEV05b]
     - ⇒ MOO Meta-heuristics [WCSO08]
     - ⇒ ...

Open Research Problems

Open research problems:

1. **Flexibility (C3)**: Workflow **templates required**, **unusable** if requirements **change**
   - **Process template generator** described in [LGG+10]: instead of generating templates from scratch, this generator **retrieves templates** from **past execution logs**
   - We employ **planning**, **no** past execution **logs required**
2. **Functionally div. services (C4)**: Related approaches assume **large sets** of **equivalent services** exist [Str10]
   - Instead: **sparse solution space**
   - Uninformed meta-heuristics can get **stuck in local optima** (next slide)

   ⇒ Insufficient utility / performance

   ⇒ Addressed by **customized GA**
3. **Reliability (C5)**: Addressed by **ad-hoc replanning** [LZZ09], neglects **impact** on QoS
   - Alternative: select **multiple services** per task
   - **Increases** number of input **variables**

   ⇒ Insufficient utility / performance

   ⇒ Addressed by **prob. QoS model**

Open Research Problems

Open research problems:

1. **Flexibility (C3)**: Workflow **templates required**, **unusable** if requirements **change**
   - **Process template generator** described in [LGG+10]: instead of generating templates from scratch, this generator **retrieves templates** from **past execution logs**
   - We employ **planning**, **no** past execution **logs required**
2. **Functionally div. services (C4)**: Related approaches assume **large sets** of **equivalent services** exist [Str10]
   - Instead: **sparse solution space**
   - Uninformed meta-heuristics can get **stuck in local optima** (next slide)

   ⇒ Insufficient utility / performance

   ⇒ Addressed by **customized GA**
3. **Reliability (C5)**: Addressed by **ad-hoc replanning** [LZZ09], neglects **impact** on QoS
   - Alternative: select **multiple services** per task
   - **Increases** number of input **variables**

   ⇒ Insufficient utility / performance

   ⇒ Addressed by **prob. QoS model**

Open Research Problems

Open research problems:

1. **Flexibility (C3)**: Workflow **templates required**, **unusable** if requirements **change**
   - **Process template generator** described in [LGG+10]: instead of generating templates from scratch, this generator **retrieves templates** from **past execution logs**
   - We employ **planning**, **no** past execution **logs required**
2. **Functionally div. services (C4)**: Related approaches assume **large sets** of **equivalent services** exist [Str10]
   - Instead: **sparse solution space**
   - Uninformed meta-heuristics can get **stuck in local optima** (next slide)

   ⇒ Insufficient utility / performance

   ⇒ Addressed by **customized GA**
3. **Reliability (C5)**: Addressed by **ad-hoc replanning** [LZZ09], neglects **impact** on QoS
   - Alternative: select **multiple services** per task
   - **Increases** number of input **variables**

   ⇒ Insufficient utility / performance

   ⇒ Addressed by **prob. QoS model**

Functionally Diverse Services (C4)

- **Related work** assumes services are **functionally equivalent**

  ⇒ Services **developed independently**

  ⇓

  Functionally **heterogeneous**

  ⇓

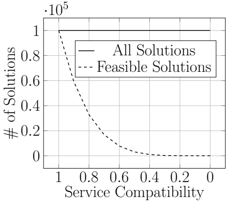
  Certain links **invalid**

- Consequences:
  - Local optima **more likely**, but still exponential search space
  - Meta-heuristics w/o domain knowledge **explore** search space **randomly**

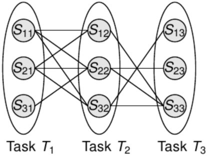
    ⇒ Slow convergence / low utility
  - Simple solution in [LM11]: just **modified** the **fitness function** (compared in eval.)
  - Solution space: $SPT^{WF} \Rightarrow SPT^{WF} \cdot p^{WF}$

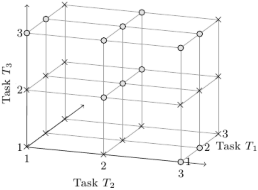
    Example: SPT = 10, WF = 50, p=0.5

$$10^{50} \Rightarrow 10^{50} \cdot 0.5^{50} \approx 10^{35}$$



SPT = 10, WF = 5





Including Functionally Diverse Services:

- **Tackled by:** Integrate **domain knowledge** (=service clustering) into existing meta-heuristic
  - **Customize** existing single-objective (SOO) and multi-objective optimization (MOO) **genetic algorithms** (GA)
    - ⇒ MOO-GA has **best performance** in the extended selection problem out of 15 algorithms
    - ⇒ **Easily customizable**
    - ⇒ Addresses Challenge **C4** (low utility in the context of functionally diverse services)
  - Propose **novel genome encoding** to cover SEOs
    - ⇒ Addresses Challenge **C5** (low reliability)

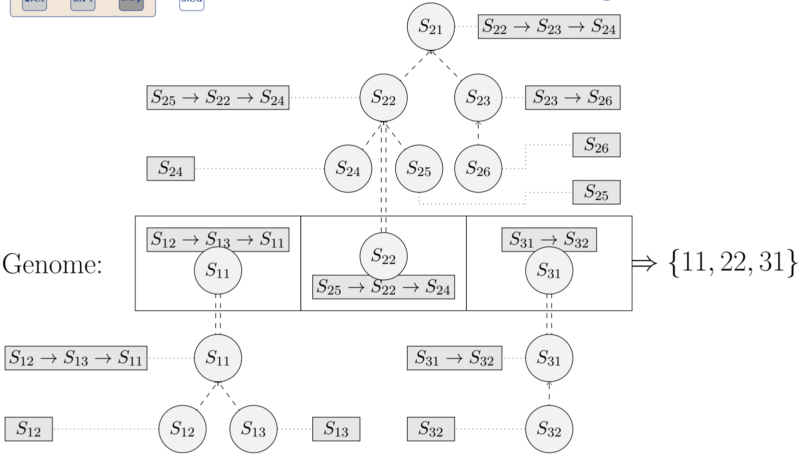
Novelty

- Genome Encoding of SEOs
- Customized GA Operators

Based On

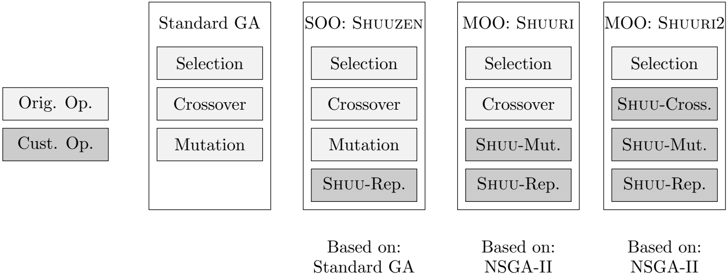
- GA and their application to service selection

Encoding of a Genome



- **Each cell** encodes a **SEO**, cached in the cluster nodes.
  - ⇒ **One decision** variable for up to 3 services
  - ⇒ No increase in the number of input variables
  - ⇒ Preserves utility / performance, fault-tolerant workflow possible

Customized GA operators

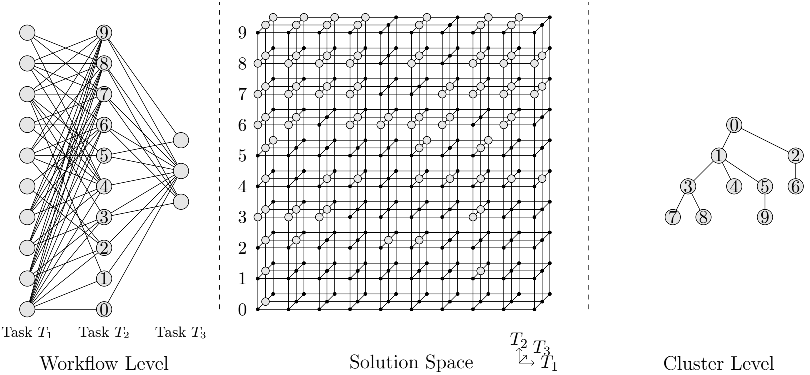


- **Customized operators** leverage service clustering:
  - SHUU-Repair: Find **functionally valid** solution (**C4**)

    infeasible solution → feasible solution
  - SHUU-Mutate: Explore **feasible** solution space (**C4**)

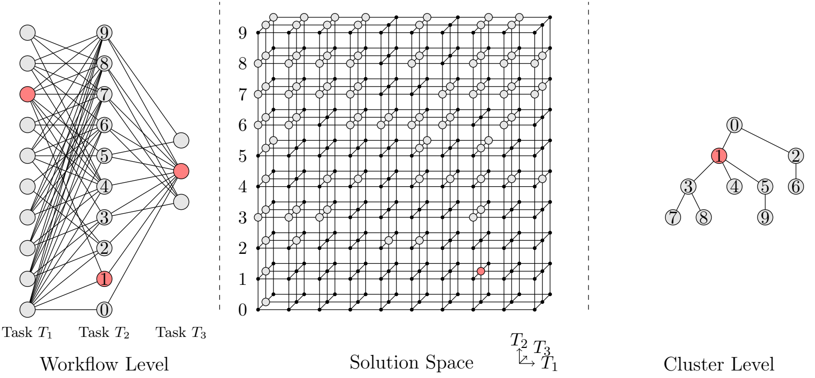
    feasible solution → feasible solution
  - SHUU-Crossover: Max. **distrib.** of backup services (**C5**)
- **Remark**: This presentation covers SHUURI/SHUURI2

Optimization Phase



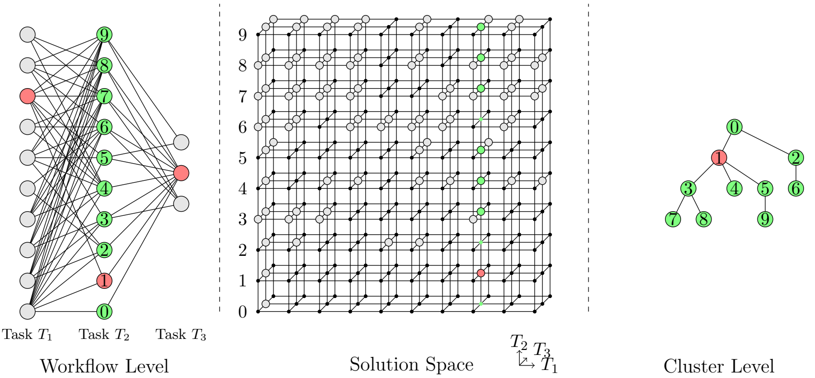
- **Left**: For each task, select one service
- **Middle**: Visualizes the search space, one point = path
- **Right**: Clustering of task T_2, new view

Custom ops. - 1. Mutate operator



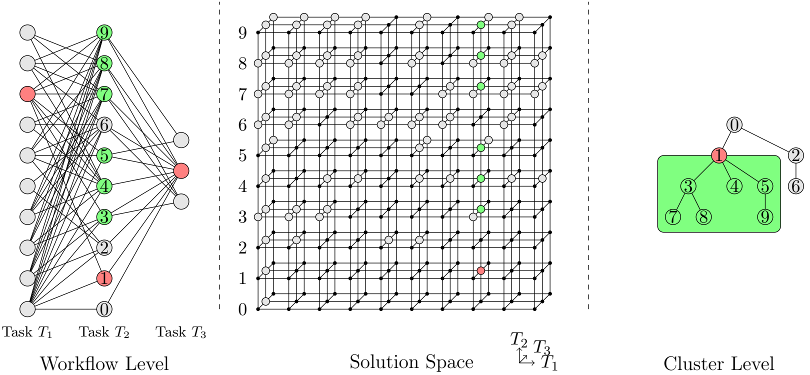
- Given solution: $\{7, \mathbf{1}, 1\}$
- In each generation, **mutate operator** is applied
  - ⇒ Explore **new solutions**

Custom ops. - 1. Mutate operator



- **Uninformed** mutate operator picks task T_2
- Selects random service from T_2
  - 3 of 9 possibilities (33%) **invalid!**
  - Given $p = 50\%, WF = 3$, $1 - 0.5^3 \approx 97\%$ offspring invalid

Custom ops. - 1. Mutate operator

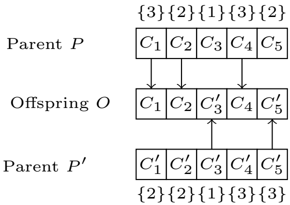


- Shuu-Mutate: given a feasible solution
  - ⇒ With probability P_{mut} **only** pick **nodes** from **subcluster**
  - ⇒ Explores **feasible solution subspace** efficiently

Custom ops. - 2. Crossover operator

- SHUU-Crossover: Modified **uniform crossover** operator
- Genomes are **annotated** with number of **independent service locations**
- Compare the annotations of both **parent cells**:
  - If one parent **has more**: 75% pick this node
  - Else, pick one with 50%

⇒ Favors cluster **nodes** with **distributed services**



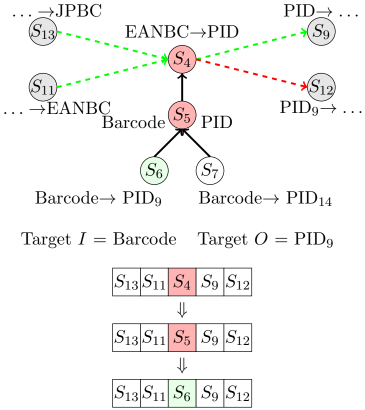
Custom ops. - 3. Repair operator

- SHUU-Repair: Applied with **probability** P_{rep}
  - Leverage **domain knowledge**
  - By exp.: 33% best trade-off
- Compute **target** inputs and outputs by:

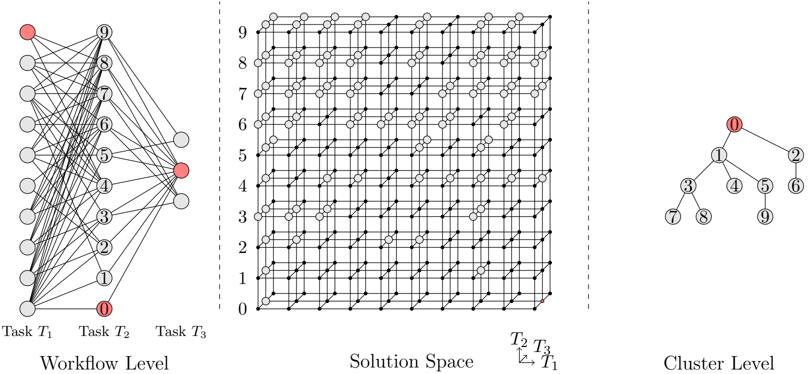
$$\text{Target } I : C \in \mathcal{O} \ . \ \forall I \ . \ I \sqsubseteq C$$

$$\text{Target } O : C \in \mathcal{O} \ . \ \forall O \ . \ C \sqsubseteq O$$

- Intuition: **Invalid** solutions "**pushed**" to feasible solutions, uses **cached parameters**
- Applicable for **SOO** and **MOO**
- **No similarities** with repair extensions of GAs [CB98]

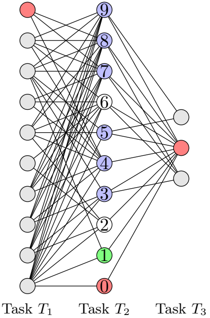


Custom ops. - 3. Repair operator

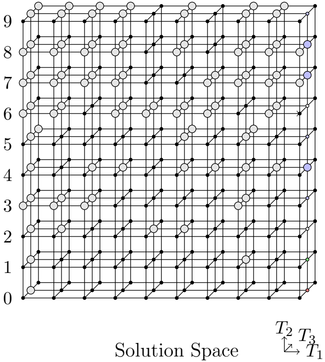


⇒ Start: Invalid solution $\{9, \mathbf{0}, 1\}$ selected

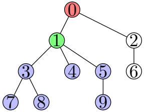
Custom ops. - 3. Repair operator



Workflow Level



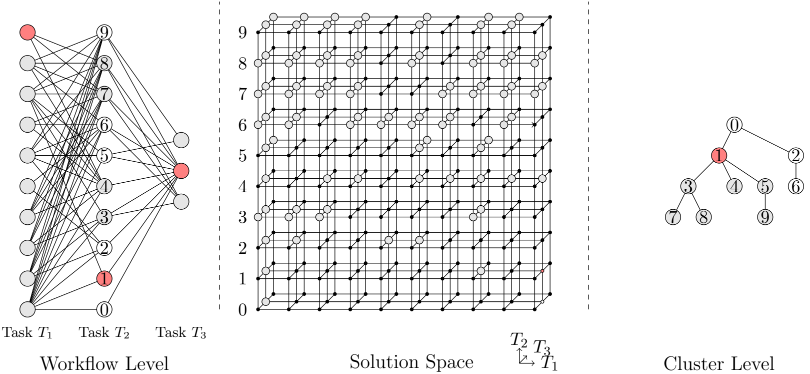
Solution Space



Cluster Level

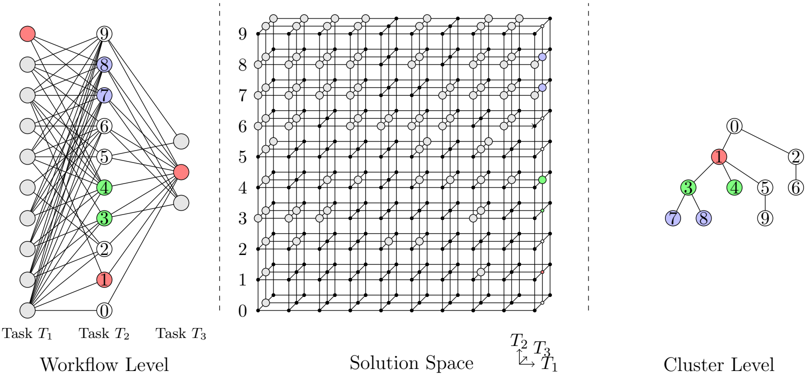
⇒ Subcluster **2** pruned ⇒ clustering = search tree

Custom ops. - 3. Repair operator



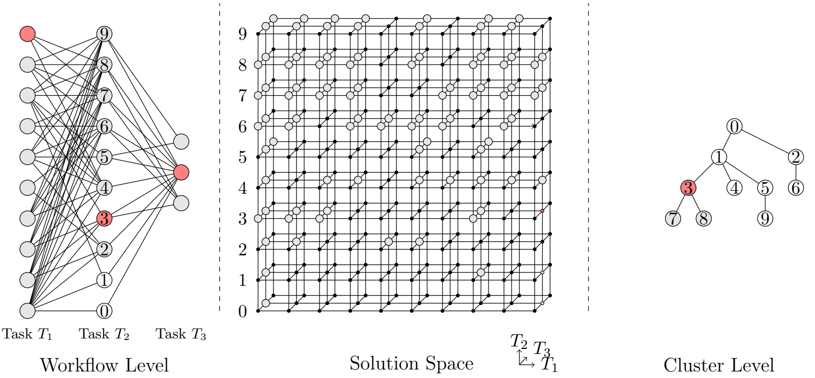
⇒ Descend to **1**, still invalid

Custom ops. - 3. Repair operator



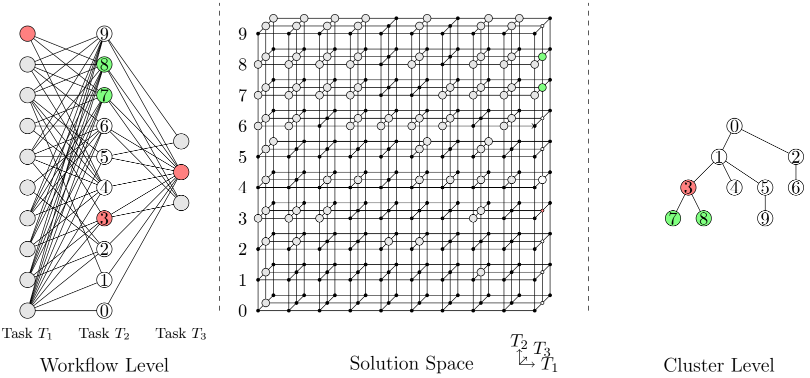
⇒ According to clustering, only **3** and **4** are valid

Custom ops. - 3. Repair operator



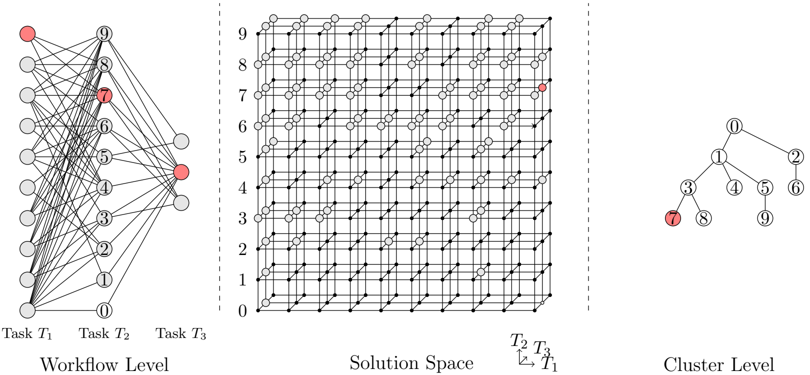
⇒ Randomly select **3**

Custom ops. - 3. Repair operator



⇒ Both **7** and **8** are valid

Custom ops. - 3. Repair operator



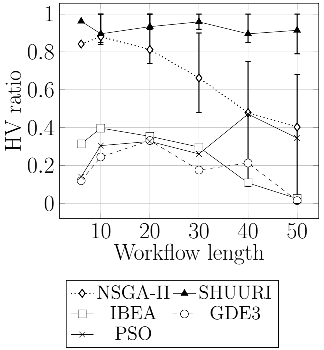
⇒ Repaired genome by replacing **0** with **7**

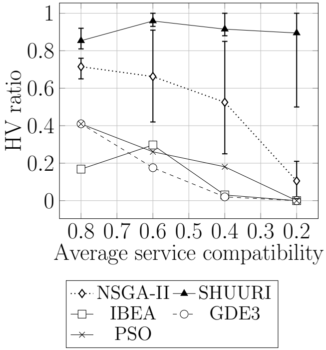
Evaluations - MOO - Settings

- Used the **JMetal** framework[4]
- **Extended** the **NSGA-II** algorithm ⇒ **SHUURI**
- **Compared** it with **15 MOO** algorithms (top-5 in the next slides)
- Generated 20 services for each task, associated with types from the SUMO ontology[5]
- **QoS randomly** generated, except for the price
  - In [WIH] we've used the **QWS dataset**[6] (backup slides)
  - Service **reliability from real data**
  - Moreover, implemented a **workflow simulator**
- Each test case was evaluated 100 times, max. runtime 5000msec

[4] http://jmetal.sourceforge.net/
[5] http://www.ontologyportal.org/
[6] http://www.uoguelph.ca/~qmahmoud/qws/index.html

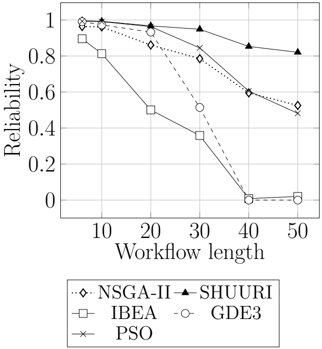
Evaluations - MOO (C4) - Results

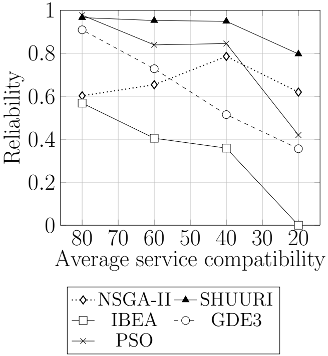




- Bounds show 90% of evaluation results
- Hypervolume (HV) ratio computed by **merging the fronts** of all algorithms
- With **increasing problem size** (workflow length, low compatibility) SHUURI **outperforms** other algorithms

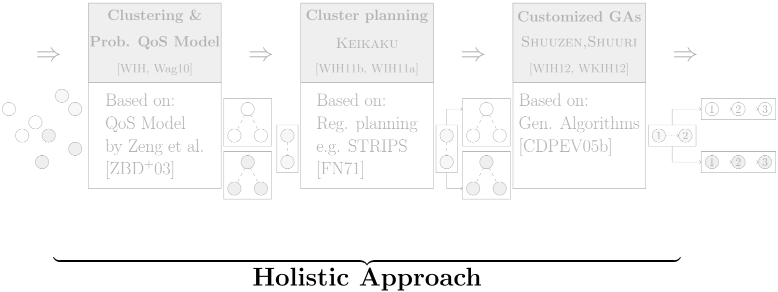
Evaluations - MOO (C5) - Results





- **Same setting** as before, comparing the **reliability**
- **Backup slides**: using workflow simulator and simulated hosts ⇒ physical location

Holistic Approach



Start movie of the prototype

5. Summary

Central Contributions 1/2 (repeat)

Issues in existing approaches:

Planning

C1 **Scalability**

- ⇒ Many **similar services**
- ⇒ **Branching factor** in search tree very **large** ⇒ exponential search space

C2 **Complex QoS Aspects**

- ⇒ Optimize **multiple QoS**, meet hard QoS **constraints**
- ⇒ No admissible heuristic, NP-hard

Selection

C3 **Flexibility**

- ⇒ User requirements **might change**
- ⇒ **Re-computation** of workflow might be **necessary**

C4 **Functional Diversity**

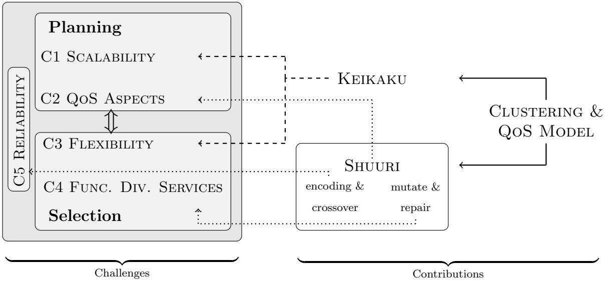
- ⇒ **Sparse** solution space → domain-independent heuristics **get stuck** in local optima
- ⇒ Low utility / slow convergence

Both

C5 **Reliability**

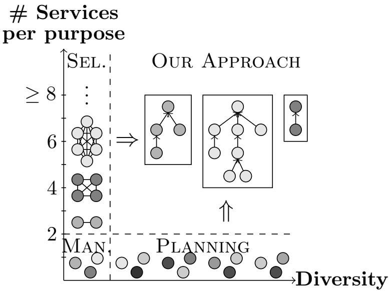
- ⇒ Selecting additional **backup services** for each task **increases number** of input **variables**
- ⇒ **Impact** of service crashes **on QoS unclear**

Central Contributions 2/2 (repeat)



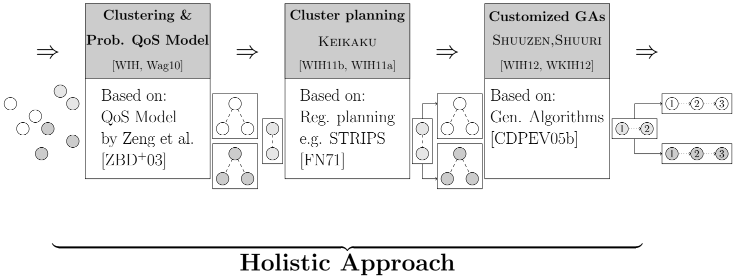
- **Extended QoS model and Clustering**
  - ⇒ Basis for Keikaku and Shuuri, encodes domain knowledge
  - ⇒ Estimates **QoS** of service **crashes**
- **Scalable cluster planer**: Keikaku
  - ⇒ **Avoids unnecessary comparisons** of services efficiently
  - ⇒ Computes workflow templates with "promising" QoS
- **Customized GA**: Shuuri
  - ⇒ **Encoding** and **crossover** : efficiently encodes **multiple services** with only **one variable**, maximizes **distribution** of backup services
  - ⇒ **Mutate** and **repair** : fast exploration of feasible solution space, **higher utility**, **faster convergence**

Assumptions (repeat)



- **Requirements**:
  1. **Semantic** annotations → Keikaku
  2. **Func.** related services → clustering, Keikaku, Shuuri
  3. **QoS** → Shuuri
  4. **Large number** of services → Shuuri
- **Evaluation results**: better results with ...
  - ... **growing** number of **services** per purpose
  - ... **increasing** degree of **diversity**

Applicability to other domains



- Functional **clustering** leverages background **knowledge** on the **service functionalities**
- **Characteristics**: functionally related entities, arranged in **hierarchy**
- Related fields:
  - Software components: theoretically applicable; however, usually only one entity per functionality

  ⇒ **Modified planning problem**

  ⇒ **Extended service selection problem** (based on MMKP)

Thank you very much for your kind attention!

P. C. Chu and J. E. Beasley, *A Genetic Algorithm for the Multidimensional Knapsack Problem*, Journal of Heuristics **4** (1998), no. 1, 63–86.

G. Canfora, M. Di Penta, R. Esposito, and M. Villani, *QoS-Aware Replanning of Composite Web Services*, In Proceedings of the IEEE International Conference on Web Services ICWS) (Washington, DC, USA), IEEE Computer Society, 2005, pp. 121–129.

Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani, *An Approach for QoS-aware Service Composition Based on Genetic Algorithms*, In Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (GECCO) (New York, NY, USA), ACM, 2005, pp. 1069–1075.

I. Di Pietro, F. Pagliarecci, L. Spalazzi, A. Marconi, and M. Pistore, *Semantic Web Service Selection at the Process-Level: The eBay/Amazon/PayPal Case Study*, In Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, vol. 1, Dec. 2008, pp. 605 –611.

Richard E. Fikes and Nils J. Nilsson, *STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving*, In Proceedings of the 2nd International Joint Conference on Artificial intelligence (IJCAI) (San Francisco, CA, USA), Morgan Kaufmann Publishers Inc., 1971, pp. 608–620.

Malik Ghallab, Dana Nau, and Paolo Traverso, *Automated Planning: Theory & Practice*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

Zhenqiu Huang, Wei Jiang, Songlin Hu, and Zhiyong Liu, *Effective Pruning Algorithm for QoS-Aware Service Composition*, In Proceedings of the 2009 IEEE Conference on Commerce and Enterprise Computing (CEC) (Washington, DC, USA), IEEE Computer Society, 2009, pp. 519–522.

Michael C. Jaeger, Gero Mühl, and Sebastian Golze, *QoS-Aware Composition of Web Services: An Evaluation of Selection Algorithms*, In Proceedings of the Confederated International Conference on the Move to Meaningful Internet Systems (OTM) - Volume I (Berlin, Heidelberg), Springer-Verlag, 2005, pp. 646–661.

Wei Jiang, Charles Zhang, Zhenqiu Huang, Mingwen Chen, Songlin Hu, and Zhiyong Liu, *QSynth: A Tool for QoS-aware Automatic Service Composition*, In Proceedings of the 2010 IEEE International Conference on Web Services (ICWS) (Washington, DC, USA), IEEE Computer Society, 2010, pp. 42–49.

Dirk Krafzig, Karl Banke, and Dirk Slama, *Enterprise SOA: Service-Oriented Architecture Best Practices (The Coad Series)*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

Matthias Klusch and Andreas Gerber, *Evaluation of Service Composition Planning with OWLS-XPlan*, In Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (Washington, DC, USA), WI-IATW '06, IEEE Computer Society, 2006, pp. 117–120.

Adrian Klein, Fuyuki Ishikawa, and Shinichi Honiden, *Efficient Heuristic Approach with Improved Time Complexity for QoS-aware Service Composition*, In Proceedings of the IEEE International Conference on Web Services (ICWS), Washington D.C., USA, 2011.

Markus Lanthaler and Christian Gutl, *Aligning Web Services with the Semantic Web to Create a Global Read-Write Graph of Data*, In Proceedings of the IEEE Ninth European Conference on Web Services (ECOWS) (Washington, DC, USA), IEEE Computer Society, 2011, pp. 15–22.

Freddy Lécué, Yosu Gorronogoitia, Rafael Gonzalez, Mateusz Radzimski, and Matteo Villa, *SOA4All: An Innovative Integrated Approach to Services Composition*, In Proceedings of the 2010 IEEE International Conference on Web Services (ICWS) (Washington, DC, USA), IEEE Computer Society, 2010, pp. 58–67.

Naiwen Lin, Ugur Kuter, and Evren Sirin, *Web Service Composition with User Preferences*, In Proceedings of the 5th European Semantic Web Conference on the Semantic Web (ESWC): Research and Applications (Berlin, Heidelberg), ESWC'08, Springer-Verlag, 2008, pp. 629–643.

Freddy Lécué and Nikolay Mehandjiev, *Seeking Quality of Web Service Composition in a Semantic Dimension*, IEEE Transactions on Knowledge and Data Engineering **23** (2011), no. 6, 942–959.

Kwei-Jay Lin, Jing Zhang, and Yanlong Zhai, *An Efficient Approach for Service Process Reconfiguration in SOA with End-to-End QoS Constraints*, In Proceedings of the 2009 IEEE Conference on Commerce and Enterprise Computing (CEC) (Washington, DC, USA), IEEE Computer Society, 2009, pp. 146–153.

Sonia Ben Mokhtar, Davy Preuveneers, Nikolaos Georgantas, Valérie Issarny, and Yolande Berbers, *EASY: Efficient semAntic Service discoverY in pervasive Computing Environments with QoS and context support*, Journal of System Software **81** (2008), no. 5, 785–808.

Stephan Murer, *13 Years of SOA at Credit Suisse: Lessons Learned-Remaining Challenges*, In Proceedings of the European Conference on Web Services (ECWS) **0** (2011), 12.

A. Strunk, *QoS-Aware Service Composition: A Survey*, In Proceedings of the 2010 Eighth IEEE European Conference on Web Services (ECOWS) (Washington, DC, USA), IEEE Computer Society, 2010, pp. 67–74.

Seekda Technologie, *The Web Service Search Engine Seekda*, `http://webservices.seekda.com/`, July 2012.

Florian Wagner, *Efficient, Failure-Resilient Semantic Web Service Planning*, In Proceedings of the IEEE International Conference on Service Oriented Computing (ICSOC) (Paul P. Maglio, Mathias Weske, Jian Yang, and Marcelo Fantinato, eds.), Lecture Notes in Computer Science, vol. 6470, 2010, pp. 686–689.

Hiroshi Wada, Paskorn Champrasert, Junichi Suzuki, and Katsuya Oba, *Multiobjective Optimization of SLA-Aware Service Composition*, In Proceedings of the IEEE Congress on Services (SERVICES) - Part I (Washington, DC, USA), IEEE Computer Society, 2008, pp. 368–375.

Florian Wagner, Fuyuki Ishikawa, and Shinichi Honiden, *A Location-aware Approach for Robust Service Compositions (under review at the IEEE Transactions on Service Computing).*

Florian Wagner, Fuyuki Ishikawa, and Shinichi Honiden, *Achieving Constraint Compliance in QoS-aware Service Planning*, In Proceedings of the 2nd Intl. Joint Agent Workshop and Symposium (iJAWS), 2011.

Florian Wagner, Fuyuki Ishikawa, and Shinichi Honiden, *QoS-Aware Automatic Service Composition by Applying Functional Clustering*, In Proceedings of the 2011 IEEE International Conference on Web Services (ICWS) (Washington, DC, USA), IEEE Computer Society, 2011, pp. 89–96.

Florian Wagner, Fuyuki Ishikawa, and Shinichi Honiden, *Applying QoS-Aware Service Selection on Functionally Diverse Services*, International Conference on Service Oriented Computing (ICSOC) 2011 NFPSLAM-SOC Workshop (George Pallis, Mohamed Jmaiel, Anis Charfi, Sven Graupner, Yücel Karabulut, Sam Guinea, Florian Rosenberg, Quan Z. Sheng, Cesare Pautasso, and Sonia Ben Mokhtar, eds.), Lecture Notes in Computer Science, vol. 7221, Springer, 2012, pp. 100–113.

Florian Wagner, Benjamin Klöpper, Fuyuki Ishikawa, and Shinichi Honiden, *Towards Robust Service Compositions in the Context of Functionally Diverse Services*, In Proceedings of the 21st International Conference on World Wide Web (WWW) (New York, NY, USA), ACM, 2012, pp. 969–978.

Dan Wu, Bijan Parsia, Evren Sirin, James A. Hendler, and Dana S. Nau, *Automating DAML-S Web Services Composition Using SHOP2*, In Proceedings of the International Semantic Web Conference (ISWC), 2003, pp. 195–210.

Tao Yu, Yue Zhang, and Kwei-Jay Lin, *Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints*, ACM Transactions on the Web **1** (2007), no. 1.

Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Z. Sheng, *Quality Driven Web Services Composition*, In Proceedings of the 12th International Conference on World Wide Web (WWW) (New York, NY, USA), ACM, 2003, pp. 411–421.

ZDNet, *Ten companies where soa made a difference in 2006*, `http://www.zdnet.com/blog/service-Oriented/ten-companies-where-soa-made-a-difference-in-2006/781`, August 2012.

Zibin Zheng, Yilei Zhang, and Michael R. Lyu, *Distributed QoS Evaluation for Real-World Web Services*, In Proceedings of the International Conference on Web Services (ICWS), 2010, pp. 83–90.