2016.1.27 SEMINAR

WANG ZIYUAN BELONGING TO HONIDEN LAB.

A METHOD OF IMPROVING QOS: EXPLORATIONS OF THE POSSIBILITY OF FUNCTION COMBINATIONS IN SERVICE COMPOSITIONS

BACKGROUND

MOTIVATION

- There is a growing need for web service providers to develop customised and flexible web services as quick as they can.
- One way to satisfy this demand is to utilise service compositions, which provide a method of consolidating several services to a richer service

BACKGROUND

WHAT IS SERVICE

- A service S is a reusable system that provides functions which are documented in a service description.

  e.g. : internet settlement, house information search engine

- S is defined by IOPEQ = (S.I, S.O, S.P, S.E, S.Q), where
  - S.I, S.O = required inputs and outputs
  - S.P, S.E = preconditions and effects
  - S.Q = QoS(quality-of-service)
- However, in my research, S is defined by IOQ = (S.I, S.O, S.Q)

BACKGROUND

WHAT IS QOS

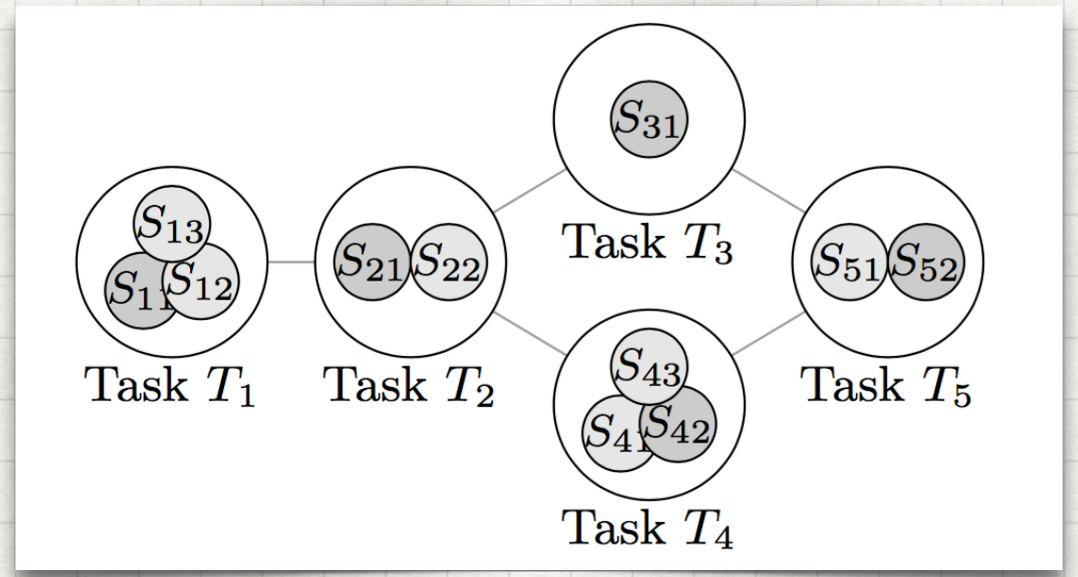
- QoS(quality-of-service):
  - Generic name of the criteria which express the quality of service Response time,Price,Reliability, ...and so on
  - In my research, $S.Q = \{q_1, q_2 , \ldots, q_n\}$

    where, q_i is normalized between 0 and 1, with 0 being the worst and 1 being the best.

BACKGROUND

WHAT IS WORKFLOW

- workflow: sequence of two or more linked services
- workflow template: sequence of two or more linked services task instead of actual services
- task: has abstract function (task.I, task.O). contains a set of services which meet the task function
- workflow ’s function: (first_service.I, last_service.O)



[1]Towards robust service compositions in the context of functionally diverse services
F Wagner, B Klöpper, F Ishikawa, S Honiden
Proceedings of the 21st international conference on World Wide Web, 969-978

BACKGROUND

WHAT IS FUNCTIONAL REQUIREMENTS

- Functional requirements f:
  - for a workflow template W

    f = (i,o) , where i ⊂ W.fisrt_task.I, o ⊂ W.last_task.O

  - In my research,

    f = (a set of W.fisrt_task.services, a set of W.last_task.services)

BACKGROUND

WHAT IS QOS OPTIMIZATION

- 3 steps for QoS Optimization[1]:
  - step1: calculate the QoS vector Q of the workflow

    computed on the basis of the types of QoS attributes and the control flow of the workflow.

  - step2: calculated value **σ**

    the components of the obtained QoS vector Q are aggregated into a single value **σ** by applying a weighted sum.

  - step3: algorithms optimize **σ** and try to meet the QoS constraints.

[1]Towards robust service compositions in the context of functionally diverse services
F Wagner, B Klöpper, F Ishikawa, S Honiden
Proceedings of the 21st international conference on World Wide Web, 969-978

BACKGROUND

WHAT IS SERVICE COMPOSITION

- service composition[1]: a system to solve composition problem
- existing composition problem definition:
  - input: a workflow template, a number of services, function requirements
  - output: QoS best workflow and its QoS
- existing algorithm of service composition: GA, ACO,...

[1]Towards robust service compositions in the context of functionally diverse services
F Wagner, B Klöpper, F Ishikawa, S Honiden
Proceedings of the 21st international conference on World Wide Web, 969-978

PROBLEM

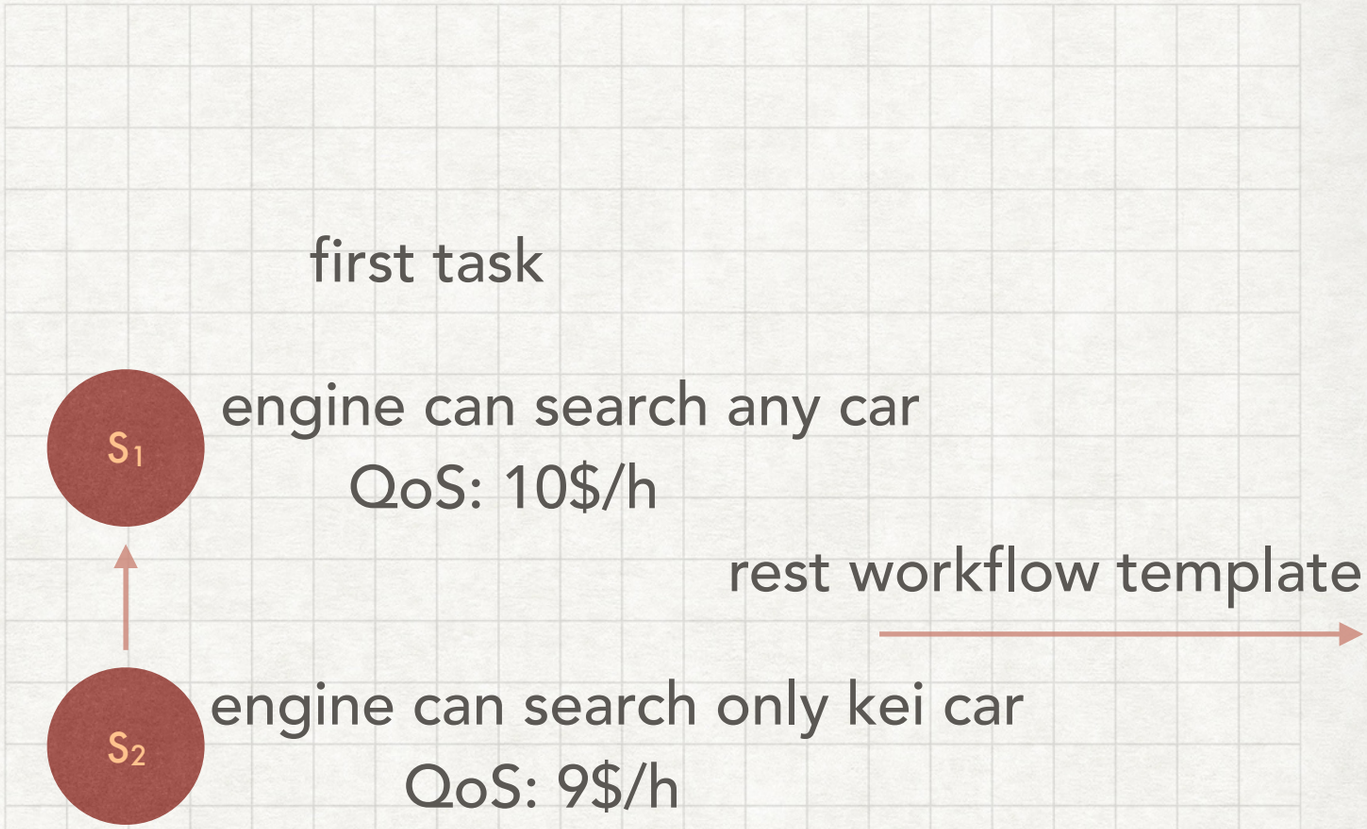
BEST WORKFLOW IS LIMITED

- Reason:
  - existing composition problem definition utilised fixed functional requirements as input
- Result:
  - the search space in that case is limited, which would possibly not include service compositions with better QoS whose functions are slightly different from the required one

PROBLEM

BEST WORKFLOW IS LIMITED

- web service providers want to create a service which use S_1 or S_2 as first service of workflow.
- service purpose: help user to rental a car for shopping
- if providers insist “can search any car” as input part of functional requirements, he will not find the best solution !



APPROACH

EXTENDED COMPOSITION PROBLEM

- put forward the concept of Variation:

  to a functional requirement = (i, o),

  where i = a set of W.fisrt_task.services,

  o = a set of W.last_task.services

  adjust (i, o) into (i', o'), where i' and o' is stronger or weaker than i and o' for obtaining a Variation function (i, o)

APPROACH

EXTENDED COMPOSITION PROBLEM

- the concept of Strong and Weak:
  - to a service s ∈ fisrt_task, Strong(s) : {s' ∈ fisrt_task, s.I ⊊ s'.I}
  - to a service s ∈ fisrt_task, Weak(s) : {s' ∈ fisrt_task, s'.I ⊊ s.I and s' adjoin s}. Too weak service is not expected
  - to a service s ∈ last_task, Strong(s) : {s' ∈ last_task, s.I ⊊ s'.I}
  - to a service s ∈ last_task, Weak(s) : {s' ∈ last_task, s'.I ⊊ s.I and s' adjoin s}. Too weak service is not expected
  - to a set of service S ⊂ first_task, Strong(S) : Union(Strong(every s ∈ S))
  - to a set of service S ⊂ first_task, Weak(S) : Union(Weak(every s ∈ S))
  - to a set of service S ⊂ last_task, Strong(S) : Union(Strong(every s ∈ S))
  - to a set of service S ⊂ last_task, Weak(S) : Union(Weak(every s ∈ S))

APPROACH

EXTENDED COMPOSITION PROBLEM

to a functional requirement FQ = (i, o),

where i = a set of W.fisrt_task.services,

o = a set of W.last_task.services

There are 8 kind of basic Variation: $FQ_{(s,s)}$, $FQ_{(s,f)}$, $FQ_{(s,w)}$, $FQ_{(f,s)}$, $FQ_{(f,w)}$, $FQ_{(w,s)}$, $FQ_{(w,f)}$, $FQ_{(w,w)}$,

where s = Strong(i) or Strong(o), f = fixed(i) or fixed(o), w = Weak(i) or Weak(o). The former element apply i, latter element apply o

APPROACH

EXTENDED COMPOSITION PROBLEM

- There are 3 degrees of Variation:
- $FQ_{compatible} = FQ \cup FQ_{(s,s)} \cup FQ_{(s,f)} \cup FQ_{(f,s)}$, where compatible means functional compatible
- $FQ_{trade\text{-}off} = FQ \cup FQ_{(s,s)} \cup FQ_{(s,f)} \cup FQ_{(f,s)} \cup FQ_{(w,s)} \cup FQ_{(s,w)}$, where trade-off means functional trade-off
- $FQ_{compromise} = FQ \cup FQ_{(s,s)} \cup FQ_{(s,f)} \cup FQ_{(f,s)} \cup FQ_{(w,s)} \cup FQ_{(s,w)} \cup FQ_{(f,w)} \cup FQ_{(w,f)} \cup FQ_{(w,w)}$, where compromise means functional compromise

APPROACH

EXTENDED COMPOSITION PROBLEM

- extended composition problem definition:
  - input: a workflow template, a number of services, function requirement, the degree of Variation
  - procedure: According to the degree of Variation, calculate new function requirement, then the new system take each basic Variation in it as input
  - output: each basic Variation’s QoS best workflow and its QoS, then generate a Scheme as selection advice to user.

APPROACH

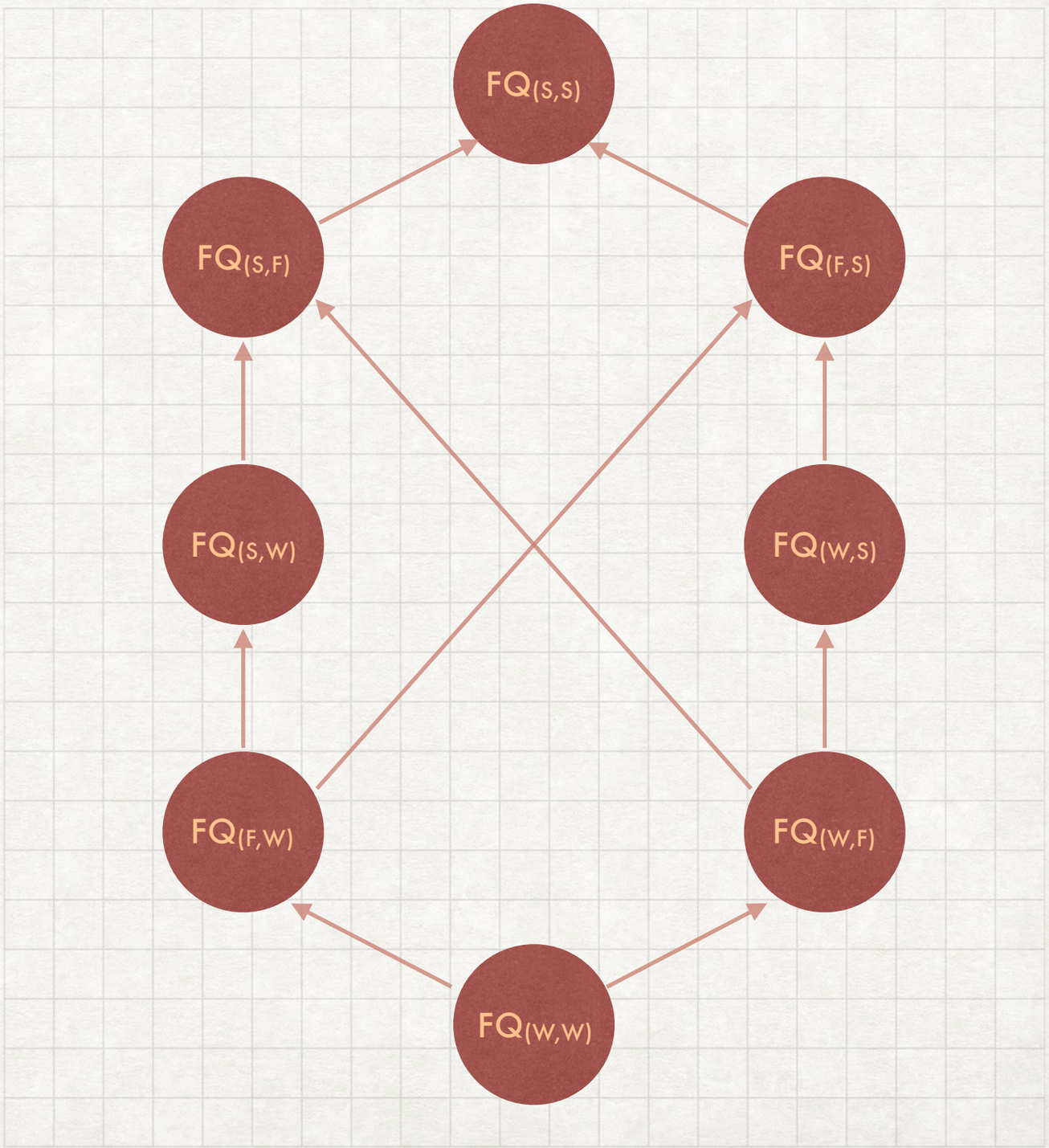
EXTENDED COMPOSITION PROBLEM

- Scheme: some local QoS best workflows which belong to different basic Variation
- How to generate Scheme:
  - step1: calculate each basic Variation's QoS best workflow and its QoS as each node
  - step2: refer to graph in the right and Scheme algorithm below

    for each node:

      if this node's QoS is greater than every node it can get to:

        add it to Scheme



APPROACH

ALGORITHM FOR EXTENDED COMPOSITION PROBLEM

- Full search algorithm: using dfs algorithm, but slow
- Skyline algorithm: apply function selection at every search step
- def selection(service_list):
  - ans = {}
  - Next = Union(every service in service_list.next)
  - while Next != {}:
    - s = QoS best in service_list and s.next != {}
    - add s to ans
    - service_list = service_list - s
    - Next = Next - s.next
    - for service in service_list:
      - service.next = service.next - s.next
- return ans

APPROACH

ALGORITHM FOR EXTENDED COMPOSITION PROBLEM

- Naive Variation algorithm:
  - for basic_variation in degree of variation:
    - Skyline(basic_variation)
      - put the local best QoS into temp_ans
- Scheme(temp_ans)

APPROACH

ALGORITHM FOR EXTENDED COMPOSITION PROBLEM

- Developed Variation algorithm:
  - divide basic_variation in degree of variation into some groups by whether containing the same input
  - for each group:
    - Skyline(group.input)
    - put the local best QoS into temp_ans
- Scheme(temp_ans)

APPROACH

ALGORITHM FOR EXTENDED COMPOSITION PROBLEM

- $FQ_{compatible} = [FQ, FQ_{(f,s)}], [FQ_{(s,f)}, FQ_{(s,s)}]$
- $FQ_{trade\text{-}off} = [FQ, FQ_{(f,s)}], [FQ_{(s,f)}, FQ_{(s,s)}, FQ_{(s,w)}], [FQ_{(w,s)}]$
- $FQ_{compromise} = [FQ_{(s,f)}, FQ_{(s,s)}, FQ_{(s,w)}],[FQ, FQ_{(f,s)}, FQ_{(f,w)}],[FQ_{(w,f)}, FQ_{(w,s)}, FQ_{(w,w)}]$

IMPLEMENTATION

EXTENDED COMPOSITION PROBLEM

- Evaluation of Proposal 1: How QoS can be improved by allowing Variation in functional requirements?
- calculate 1000 times random service composition problem
- Scheme improvement degree: Σ element's improvement of Scheme

Performance of three Variation degree in 828 valid cases

	Improved scheme numbers	average improvement degree(vs fixed functional requirements)
$FQ_{compatible}$	453	5.089129%
$FQ_{trade\text{-}off}$	485	8.827387%
$FQ_{compromise}$	566	11.80829%

IMPLEMENTATION

ALGORITHM FOR EXTENDED COMPOSITION PROBLEM

- Skyline vs Full search:

- step numbers: search from a service to a next service called one step

Comparison of the search space and time of Full search and Skyline

	step numbers	runtime(second)
Full search	2999026	16.729
Skyline	1974.1	0.5128
Full search/ Skyline	1519.1864647181	32.622854914196 6

IMPLEMENTATION

ALGORITHM FOR EXTENDED COMPOSITION PROBLEM

- From the user perspective,

  maybe $FQ_{compromise}$ is best, if he don't mind some functional compromise. $FQ_{compatible}$ is the next best

Comparison of the search space and time of different Variation

	step numbers	runtime(second)
Fixed functional requiremnet	2113	0.569703
Naive Variation algorithm $FQ_{compatible}$	6550	2.017894
Naive Variation algorithm $FQ_{trade\text{-}off}$	8837	2.642321
Naive Variation algorithm $FQ_{compromise}$	13062	3.992750
Developed Variation algorithm $FQ_{compatible}$	4100	0.992360
Developed Variation algorithm $FQ_{trade\text{-}off}$	5456	1.231265
Developed Variation algorithm $FQ_{compromise}$	5867	1.263835

DISCUSSION

LIMITATION

- the condition functional compromise can not tolerate
- the example in page 10:
  - web service providers want to create a service, which service purpose: help user to rental a car for climbing mountains

DISCUSSION

FUTURE WORK

- concern QoS constraint
- research Skyline and Variation algorithm is different scabilitiy, such as change task numbers and services numbers in each task