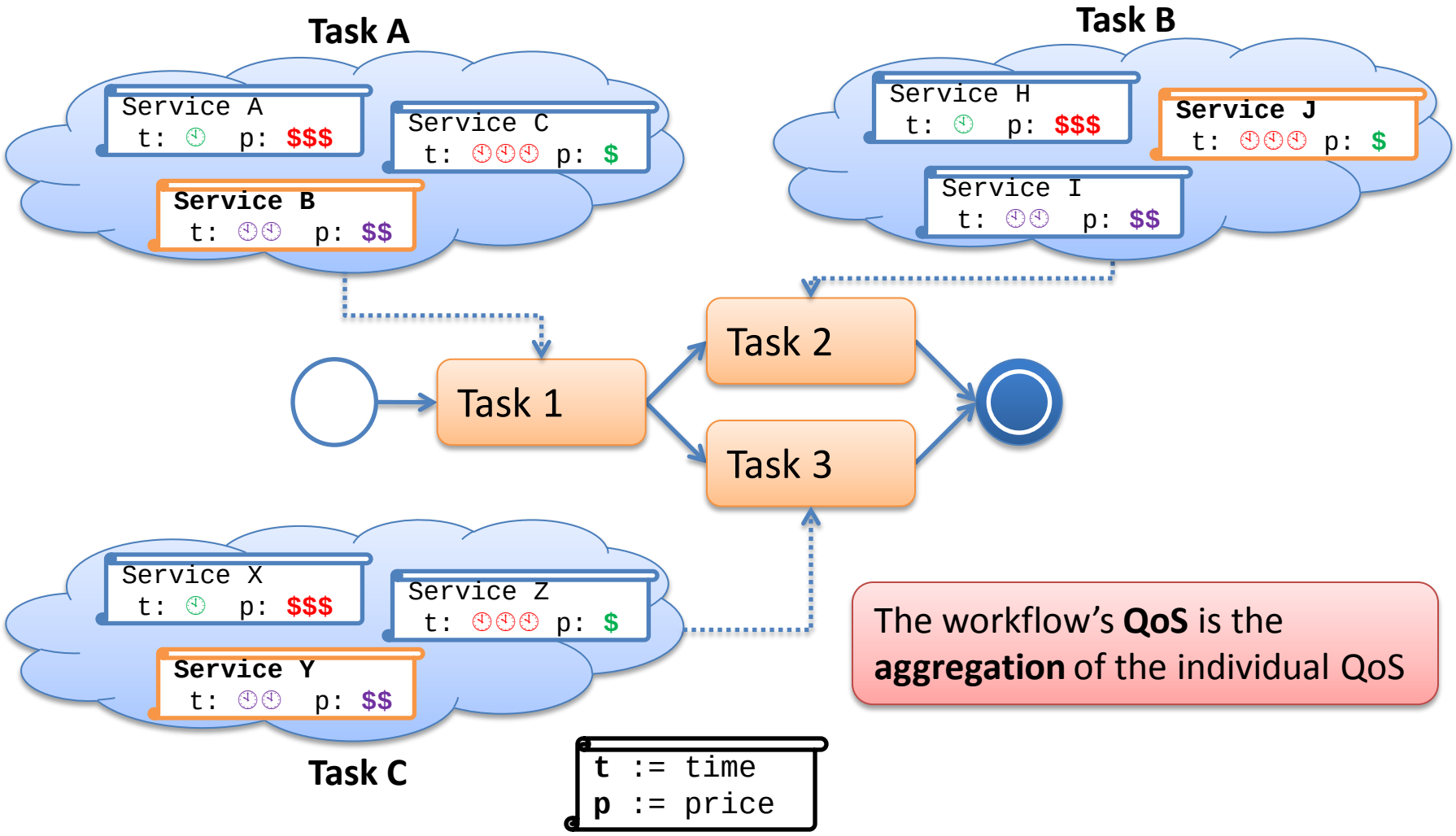
Efficient Heuristic Approach
with
Improved Time Complexity
for
QoS-aware Service Composition

Adrian Klein
The University of Tokyo
2nd year PhD student
www.**adrianobits.de**

Outline

1. **Introduction**
   - QoS-aware Service Composition
   - Optimization Problem
2. Challenges
   - Computation Time
   - Solution Quality
3. Approach
   - Initial Bias
   - HC* Algorithm
4. Evaluation
5. Conclusion

QoS-aware Service Composition



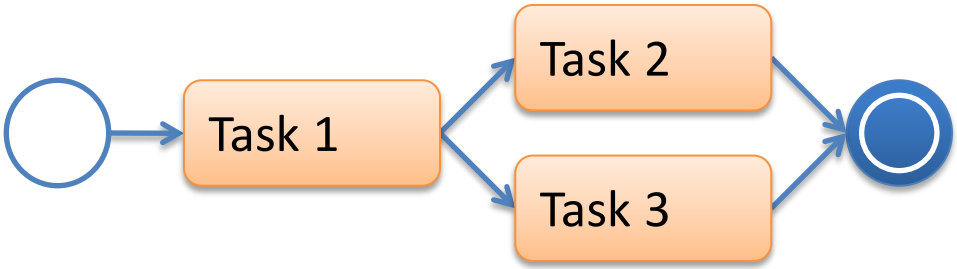
Optimization Problem

Minimize:

weighted sum of QoS

Fulfill:

constraints on QoS



E.g: `MIN{0.8 t + 0.2 p}`

E.g: `(t≤50ms AND p≤5$)`

NP-hard problem…

Use **Heuristic** Approach

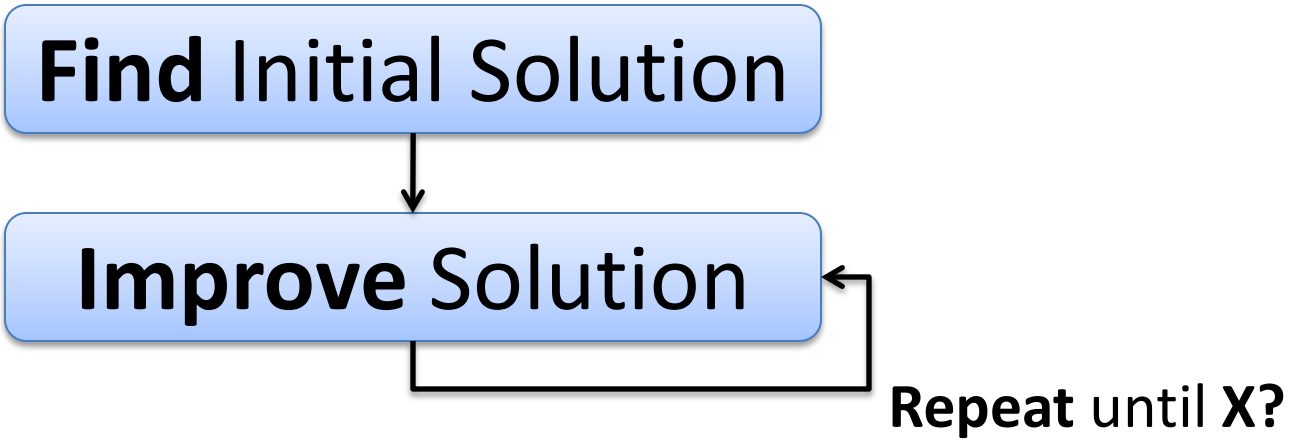
Outline

1. Introduction
   - QoS-aware Service Composition
   - Optimization Problem
2. **Challenges**
   - Computation Time
   - Solution Quality
3. Approach
   - Initial Bias
   - HC* Algorithm
4. Evaluation
5. Conclusion

Overview:
Finding a **Solution**

Once

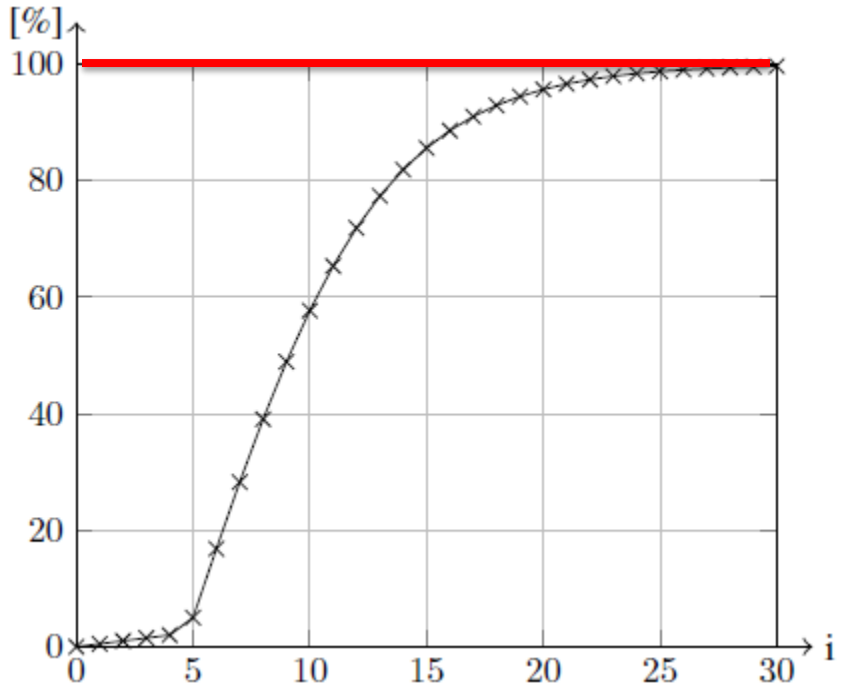
Iteratively



Computation Time:
Iterating till **Near-Optimality**

1. Reasonable quality needs very ***few*** iterations
2. **(Near-)optimality** requires ***many*** iterations

Solution Quality

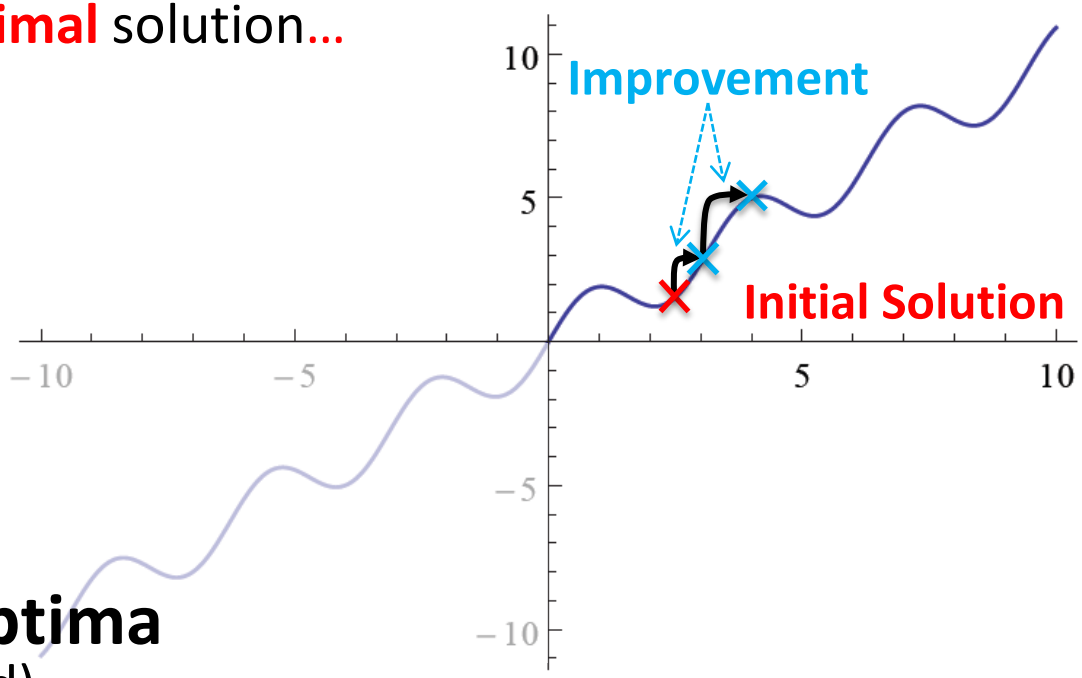


Optimum

Iterations

Solution Quality:
Reaching **Near-Optimality**

Getting stuck with a **sub-optimal** solution…



Remedies

1. Overcome **Local Optima**
   (may not always succeed)
2. Find **Better Initial Solution**
   (better = closer to optimal solution)

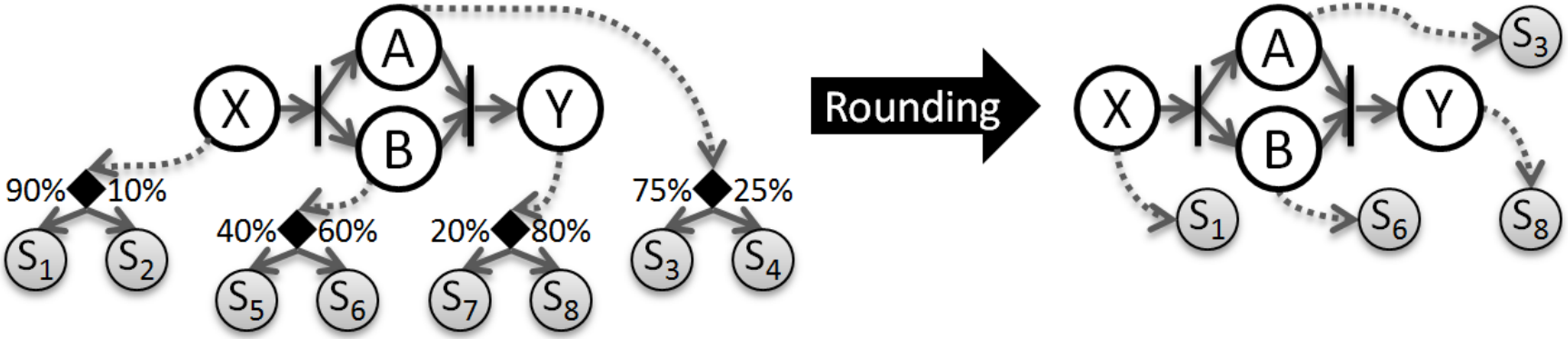
Outline

1. Introduction
   - QoS-aware Service Composition
   - Optimization Problem
2. Challenges
   - Computation Time
   - Solution Quality
3. **Approach**
   - Initial LP Solution
   - HC* Algorithm
4. Evaluation
5. Conclusion

Overview

Approach	Benefits
1. Find **Initial Solution** by **Linear Programming**	→ **Reach** *Near-Optimality*! → **Reach** it *fast*!
2. Custom **Hill-Climber** w/ *low* **Time Complexity**	→ *Still* **reach** it**!** → **Reach** it **even** *faster*!

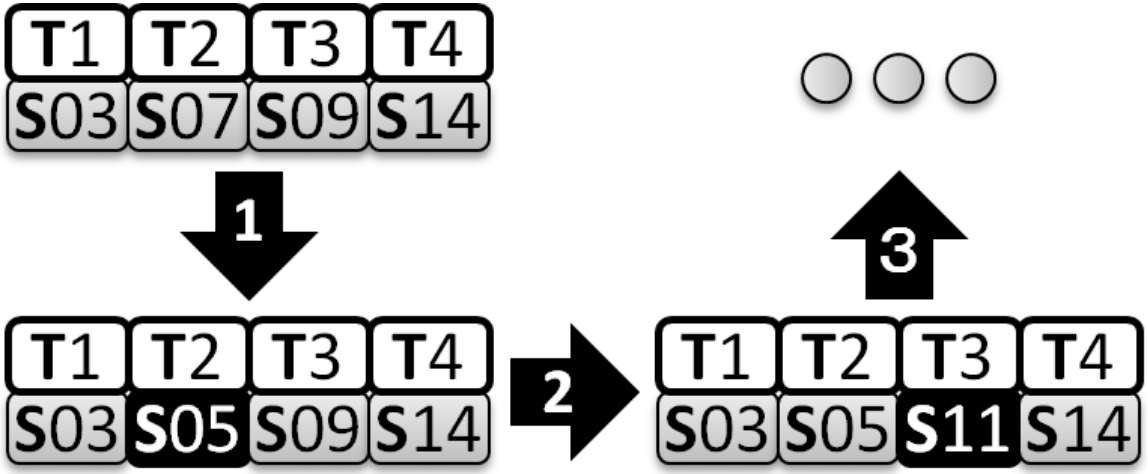
Initial **LP** Solution



LP = Linear Programming
(not Integer Programming)

Standard Hill-Climbing HC

1. Start with an **Initial Solution**
2. In each **Iteration**
   - Pick one task
   - Try all services available for that task
   - Choose the service which maximizes overall utility



Our **Utility** Function

$$u(x) = (\sum_{i=1}^{n} w_i \tilde{q}_i) - penalty(x)$$

$$penalty(x) = \sum_{i=1}^{n} (isExceeded(i) \cdot exceed(i)^2)$$

$$isExceeded(i) = \begin{cases} 1 & \text{if constraint on } \tilde{q}_i \text{ is violated} \\ 0 & \text{otherwise} \end{cases}$$

$$exceed(i) = (1 + w_i)(1 + exceedAmount(\tilde{q}_i))$$

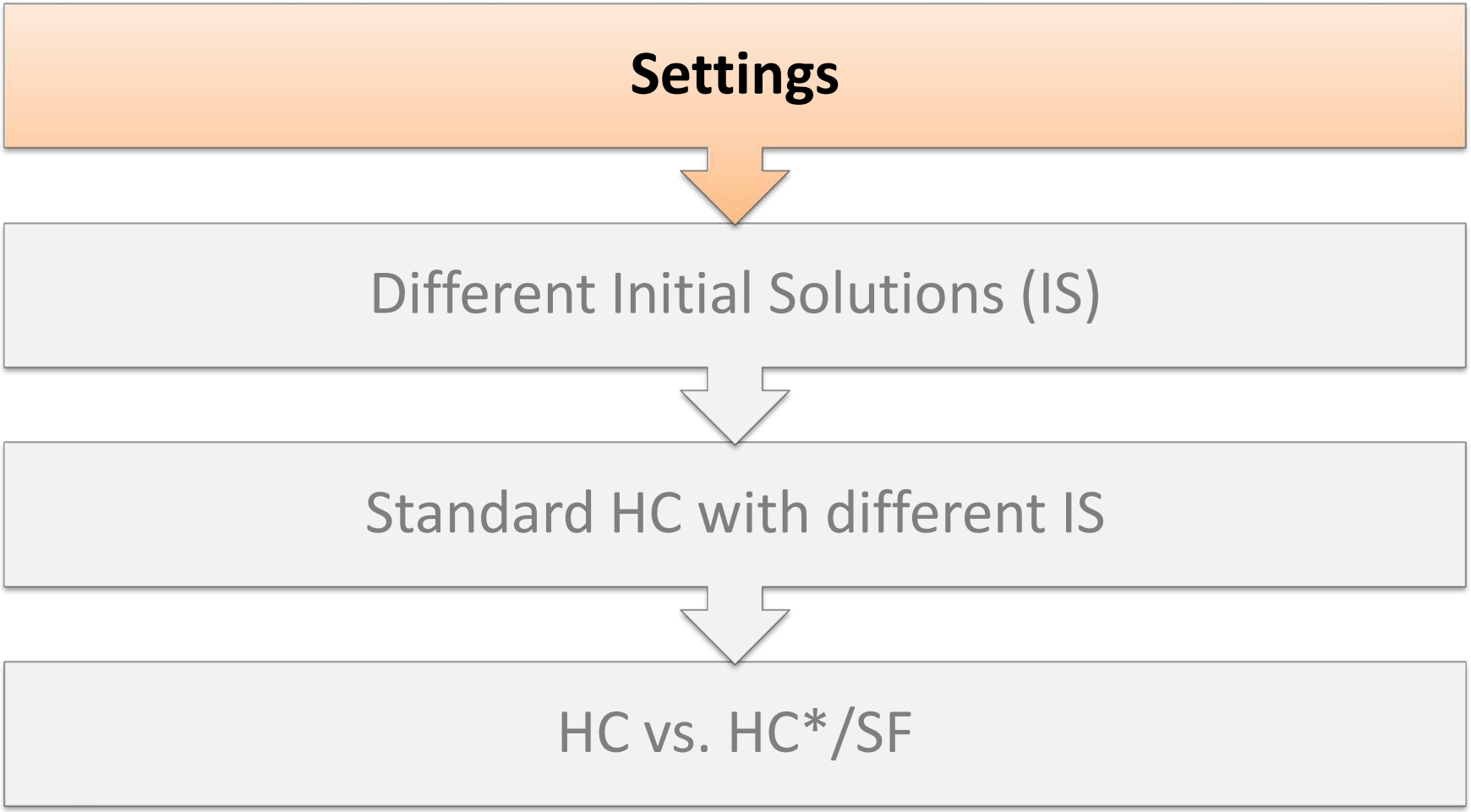
Our modified **Algorithm HC*/L**

```
1  s := initial candidate solution
2  i := 0
3  while i < √L do
4  |   N := random sub ⊆ neighbourhoud of s
5  |           with |sub| ≤ √L
6  |   s' := random n ∈ N with max u(n)
7  |   if u(s') ≤ u(s) then
8  |   |   return s
9  |   end
10 |   s := s'
11 |   i++
12 end
13 return s
```

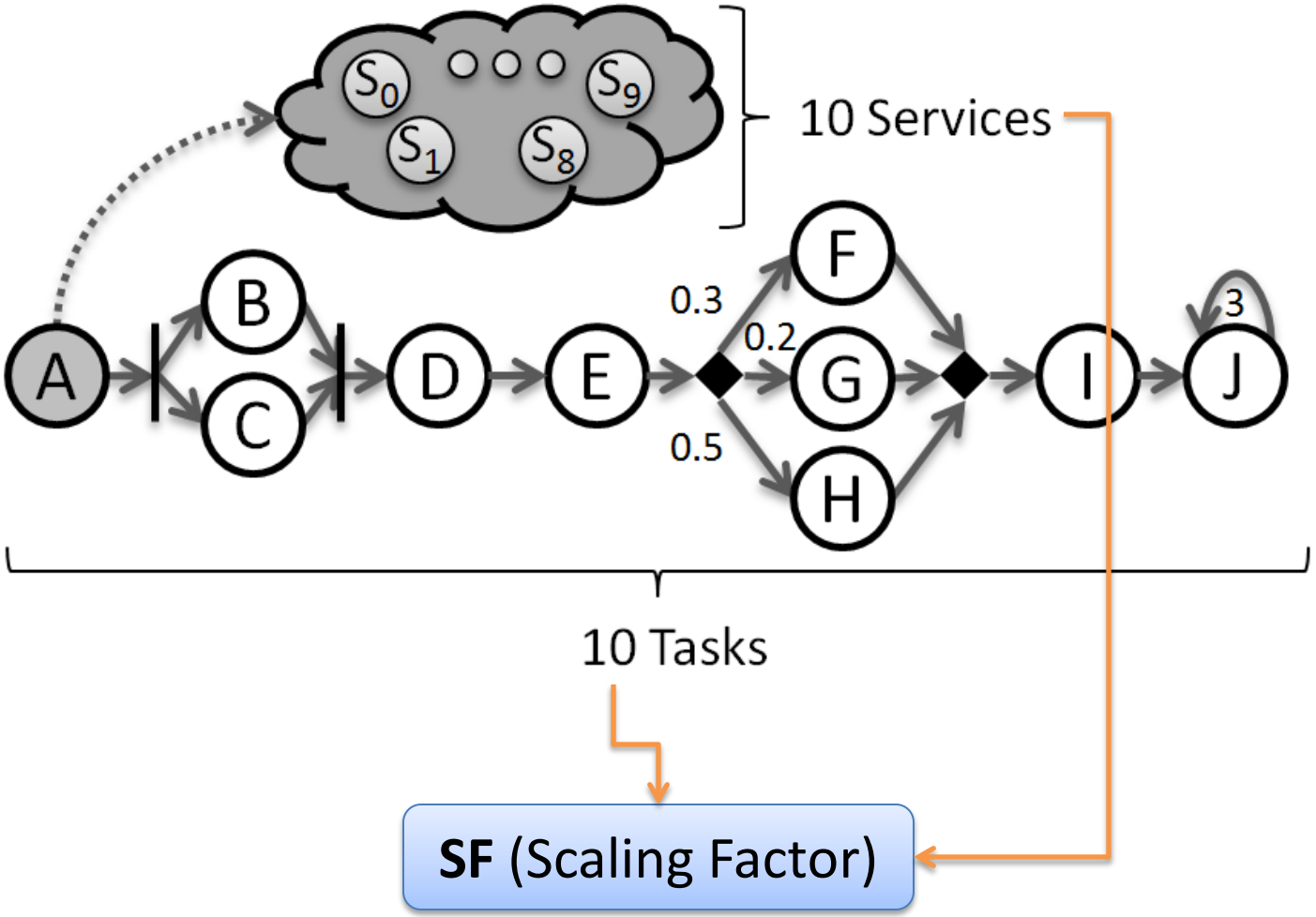
Outline

1. Introduction
   - QoS-aware Service Composition
   - Optimization Problem
2. Challenges
   - Computation Time
   - Solution Quality
3. Approach
   - Initial Bias
   - HC* Algorithm
4. **Evaluation**
5. Conclusion

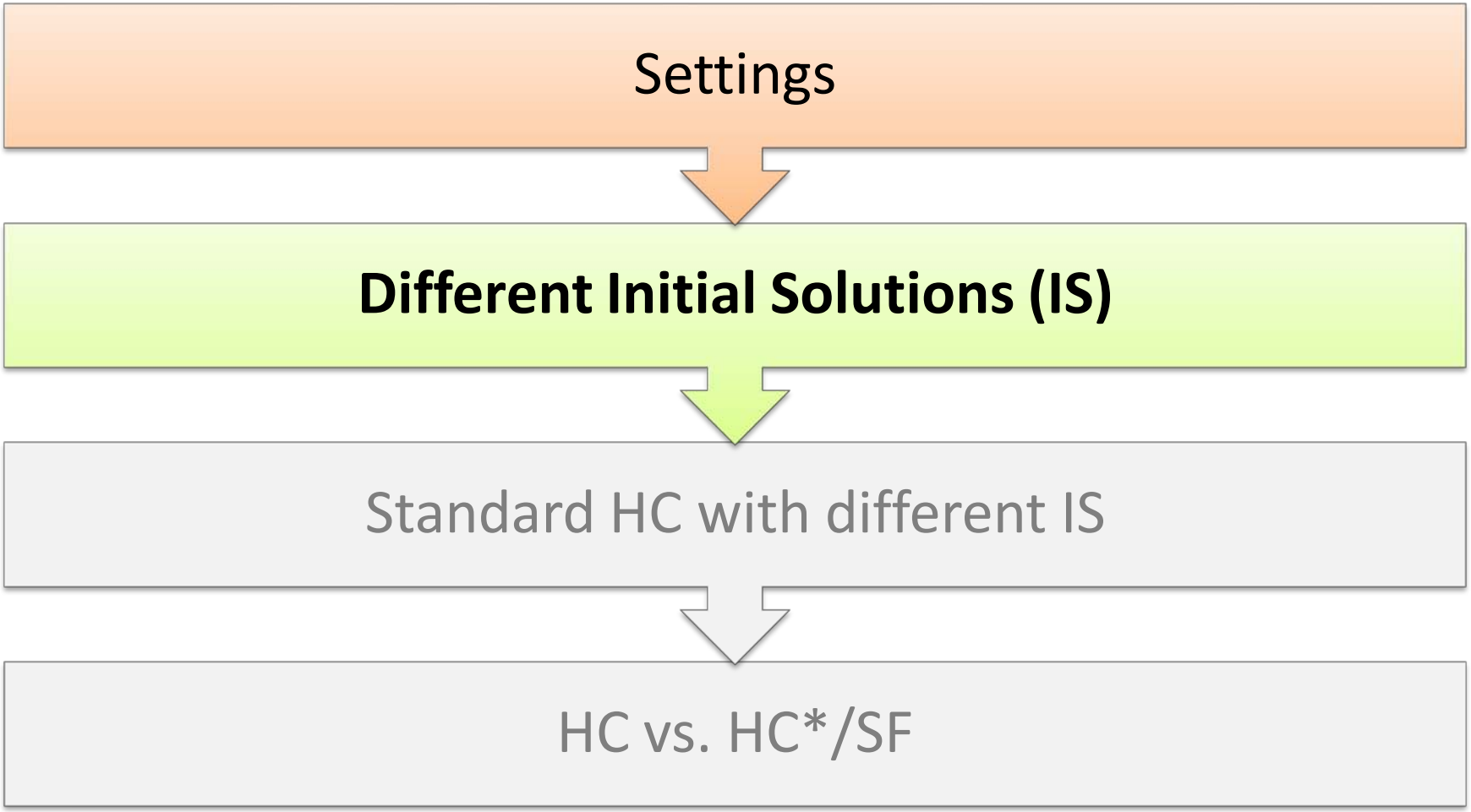
Evaluation


Settings
Different Initial Solutions (IS)
Standard HC with different IS
HC vs. HC*/SF

Generated Workflows



Evaluation


Settings
Different Initial Solutions (IS)
Standard HC with different IS
HC vs. HC*/SF

Different **Initial Solutions**

Algorithm 2: Random (R)

Input: Repetitions n
do n times
  generate random solution s
end
pick solution s with **max** $u(s)$

Algorithm 3: Greedy Weights (GW)

$s :=$ empty solution
foreach *task* $t \in s$ **do**
  set service x on solution s for task t with **max** $u(x)$
end
pick solution s

Algorithm 4: Greedy Bounds (GB)

$s :=$ empty solution
foreach *task* $t \in s$ **do**
  set service x on solution s for task t
    with **max** distance to constraints of s with x
end
pick solution s

Algorithm 5: Linear Programming (LP)

$s :=$ empty solution
$lp :=$ compute solution with Linear Programming
foreach *task* $t \in s$ **do**
  pick service x of solution lp for task t
    with **max** probability
  set service x on solution s for task t
end
pick solution s

Utility of *different* **Initial Solutions**

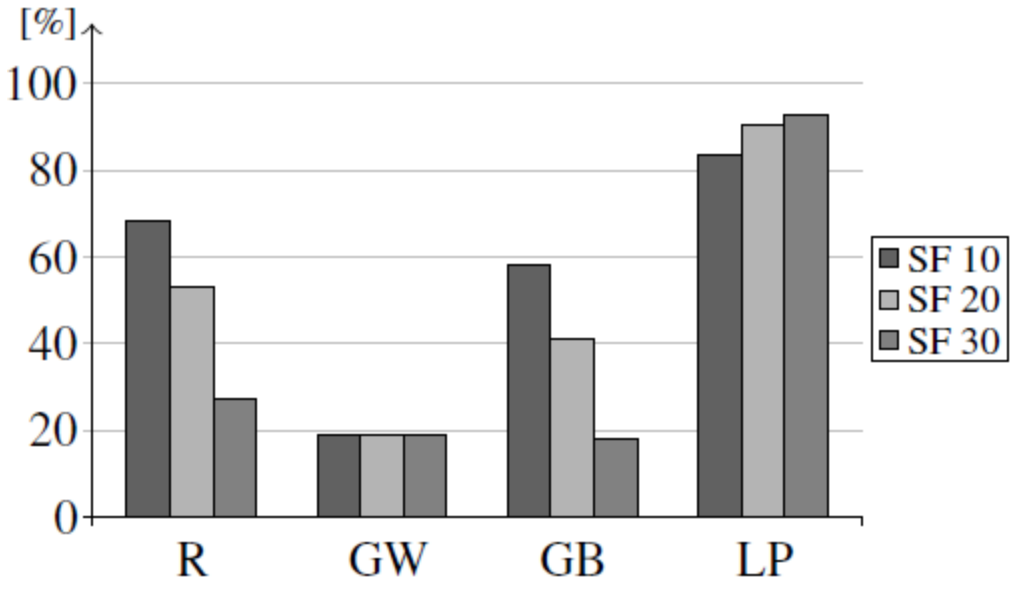
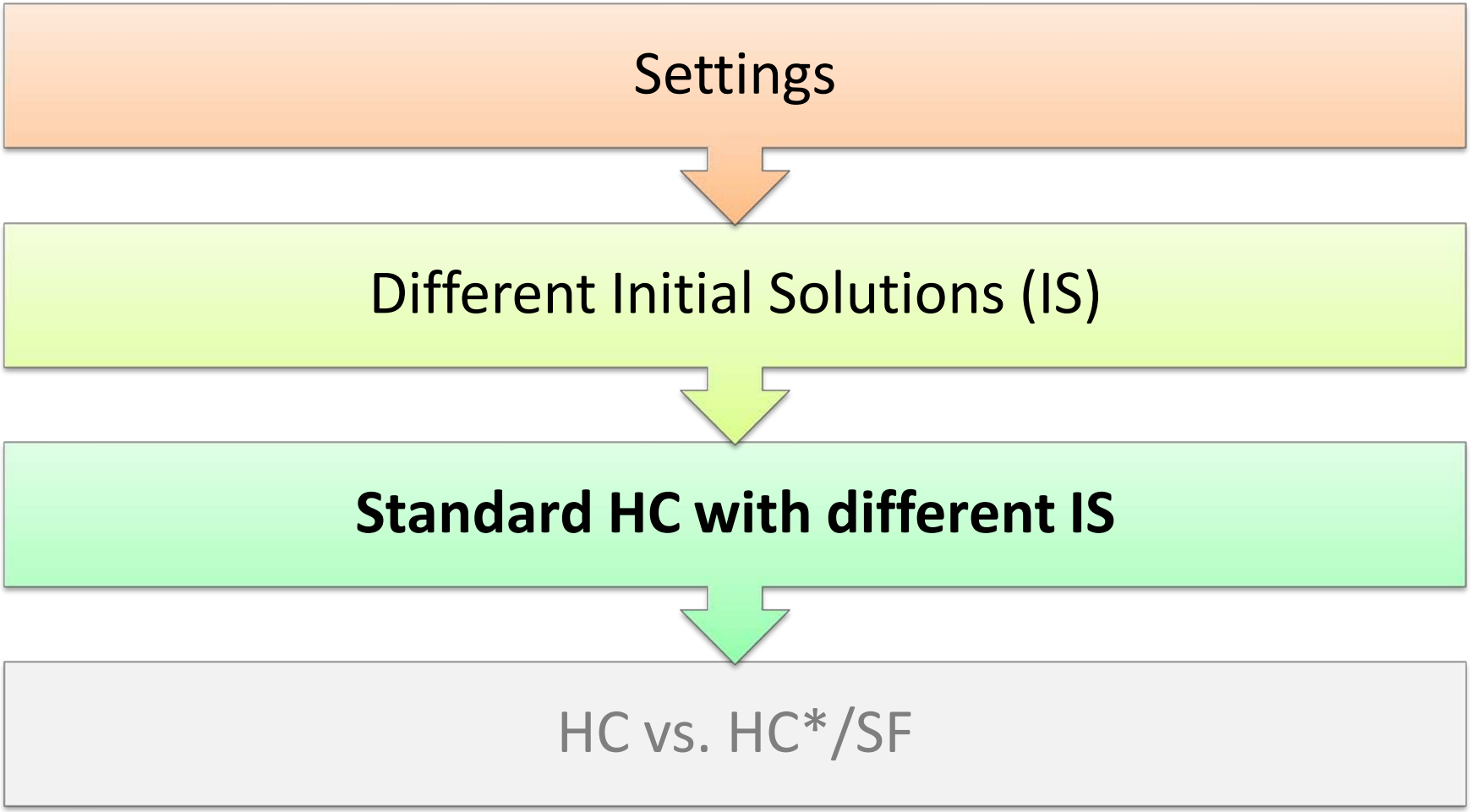
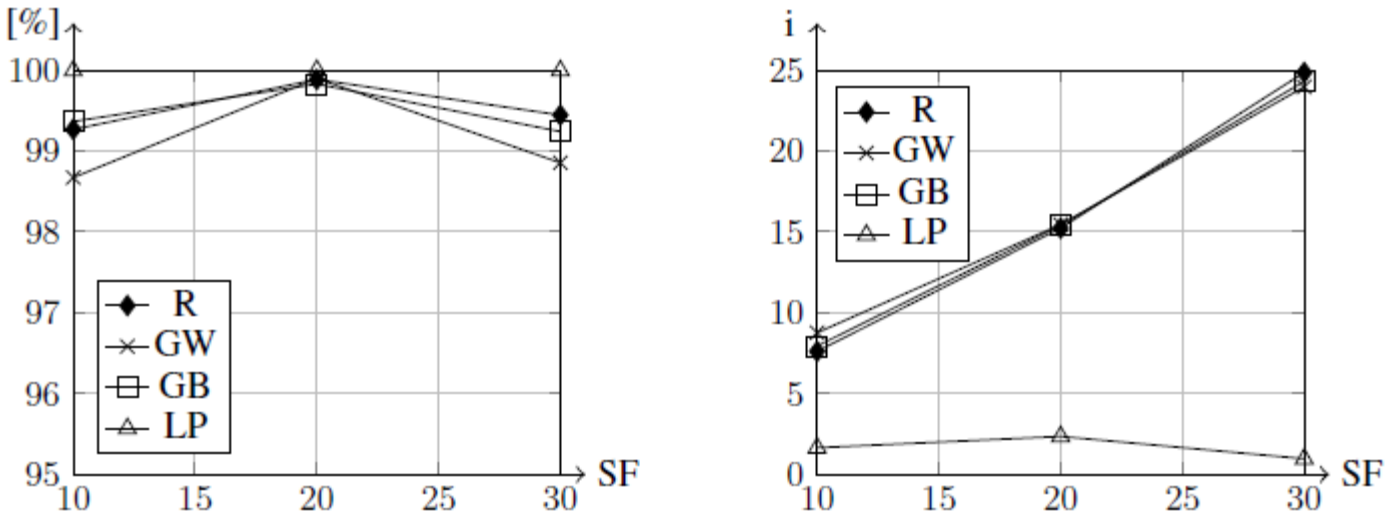


Figure 9. Average Utility of IBs for the SFs

Evaluation


Settings
Different Initial Solutions (IS)
Standard HC with different IS
HC vs. HC*/SF

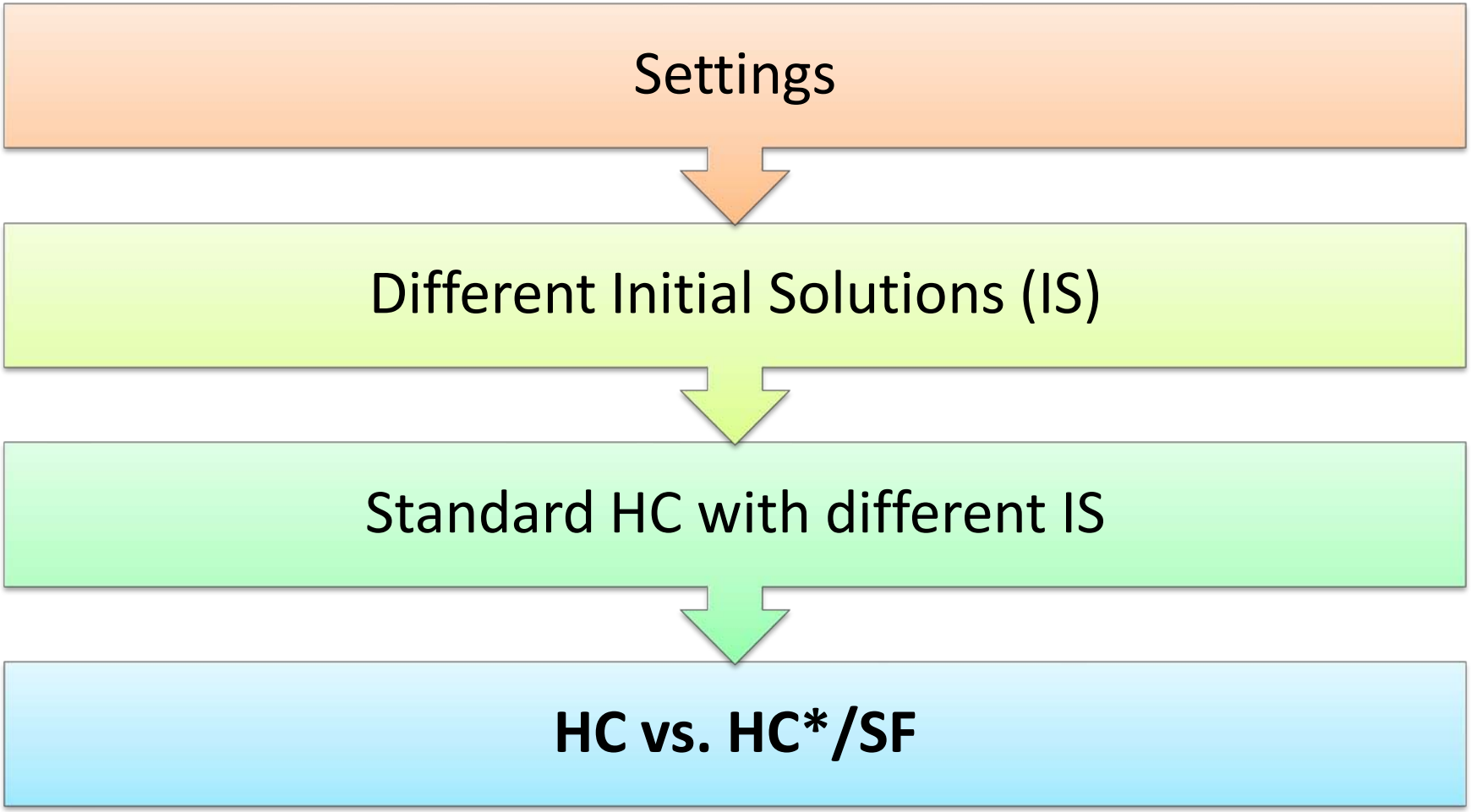
Standard **HC** with different IS



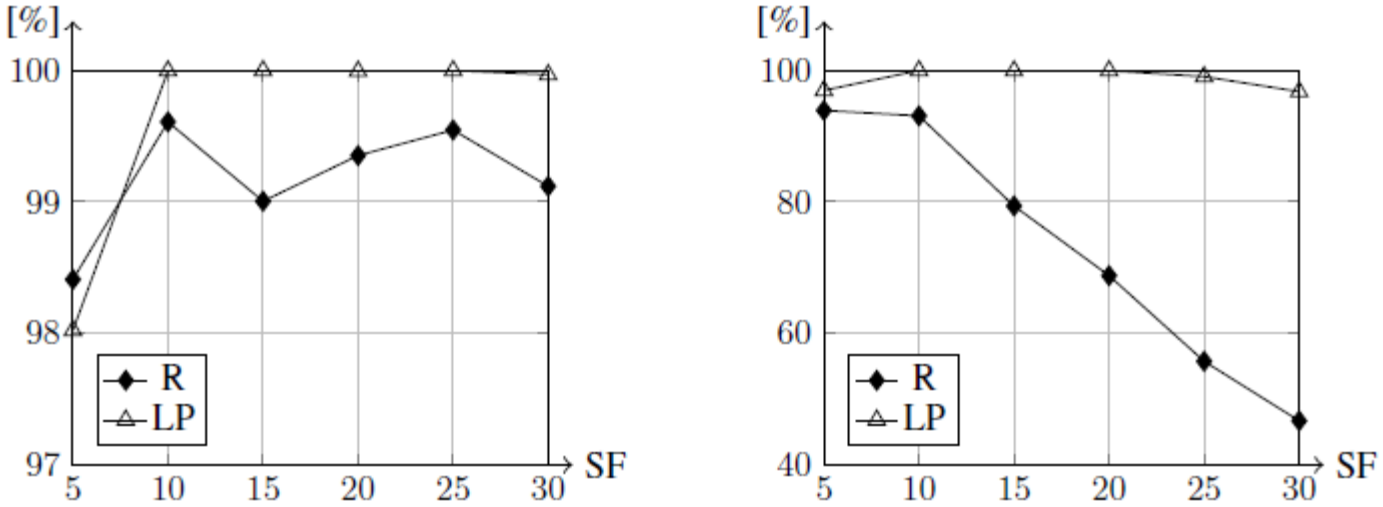
(a) Utility (b) Iterations

Figure 10. HC with respective IBs and SFs

Evaluation


Settings
Different Initial Solutions (IS)
Standard HC with different IS
HC vs. HC*/SF

HC vs. HC*/SF: Utility



(a) HC (b) *HC*/SF*

Figure 12. Average Utility of HC and *HC*/SF* for the *SF*s

HC vs. HC*/SF: Runtime

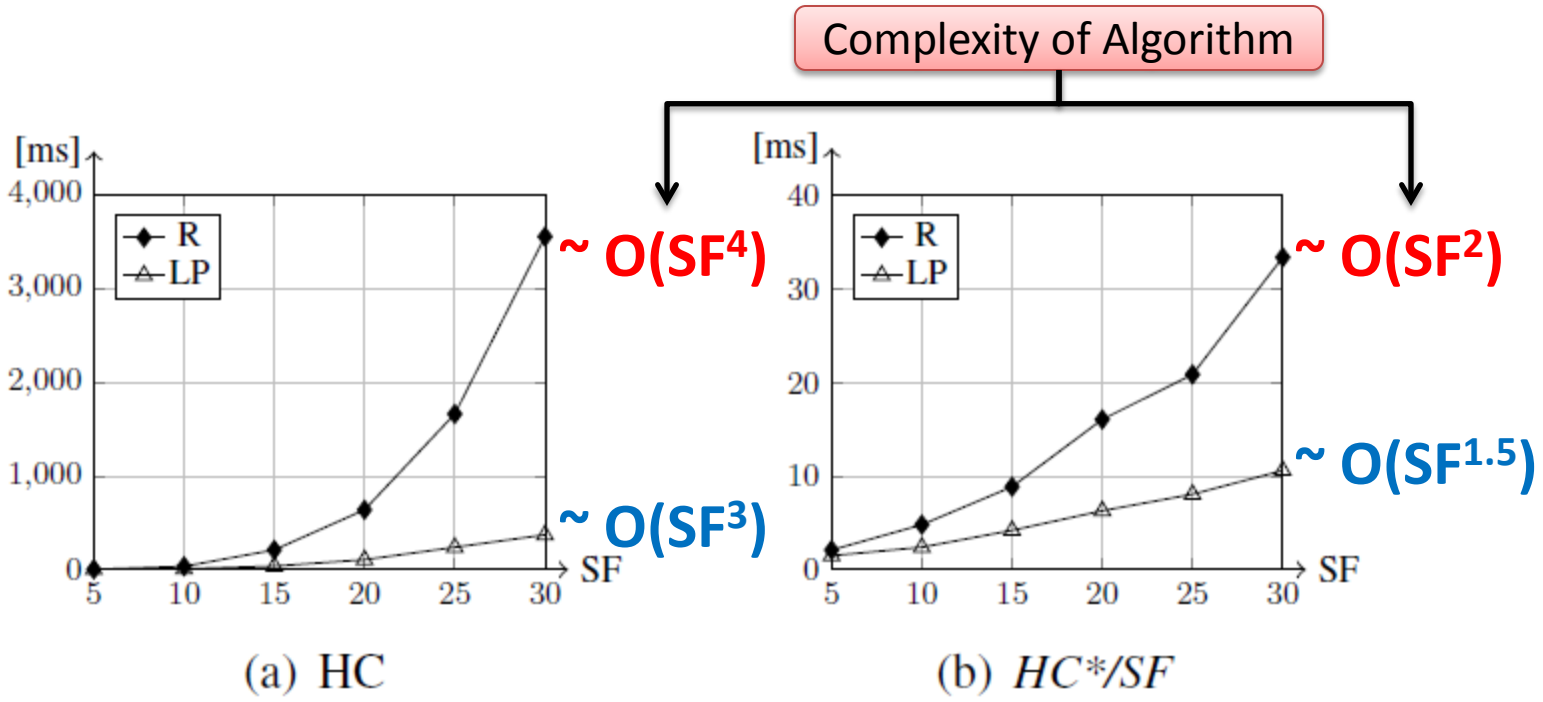


Figure 11. Runtime of HC and *HC*/SF* for the *SF*s

Outline

1. Introduction
   - QoS-aware Service Composition
   - Optimization Problem
2. Challenges
   - Computation Time
   - Solution Quality
3. Approach
   - Initial Bias
   - HC* Algorithm
4. Evaluation
5. **Conclusion**

Conclusion

Heuristic Approach based on *Hill-Climbing*

- Initial Solution by *Linear Programming*
- **HC*** with *Reduced Search Space*

*** Benefits**

- **Near-Optimal** *Solutions*
- *Improved* **Time Complexity**

 # Future Work

- Use *LP* Initial Solution with *Genetic Algorithm*

東京大学
THE UNIVERSITY OF TOKYO

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

Thank you for your attention!

Any questions/comments?