サービスコンピューティング研究会

Invited PhD Talk @ 2012年度第4回研究会

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

（ネットワークを考慮した長期間に渡るサービス合成におけるスケーラブルな最適化の適用）

Adrian Klein

The University of Tokyo

D3, Honiden Lab

Outline

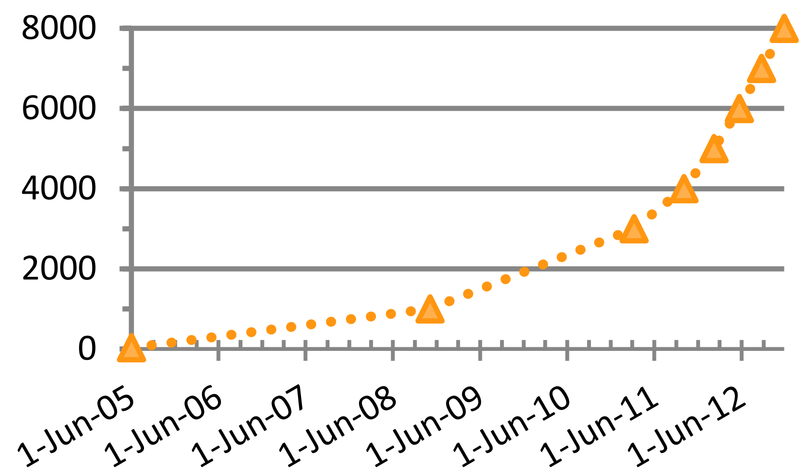
1. PRELIMINARIES

General **Overview** -> **Definition** of the **QSC** Research Problem

Services?

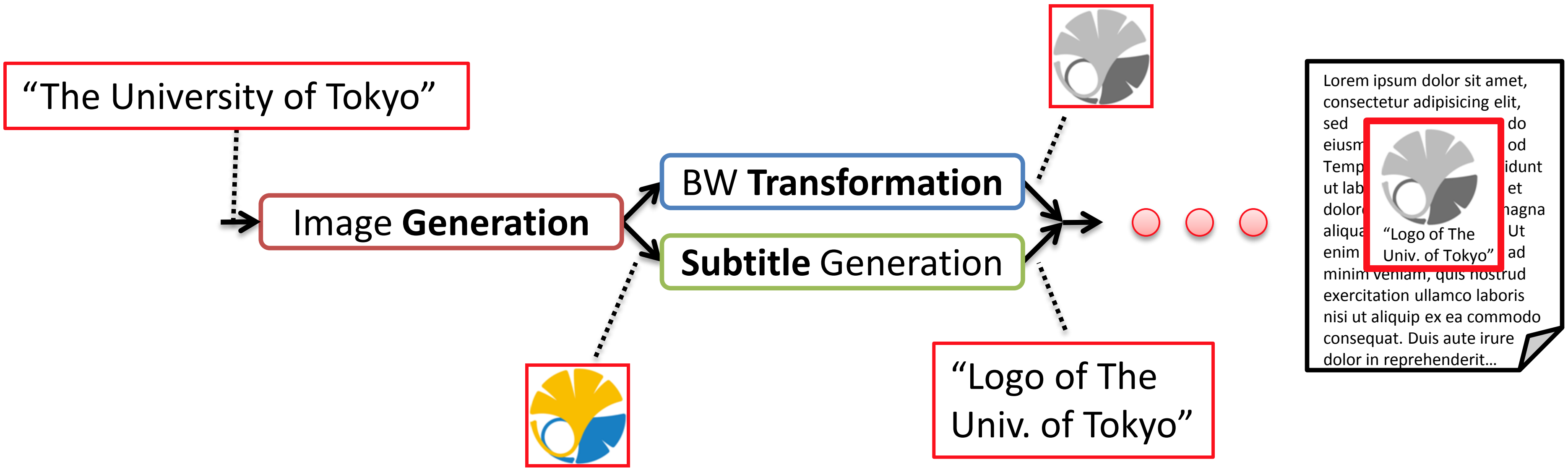
1. A **service** is a **software component** that encapsulates business logic and is accessible over the **network**.

2. **Atomic** services can be **composed** to achieve complex functionality.
3. The **number** of available services keeps **increasing**.



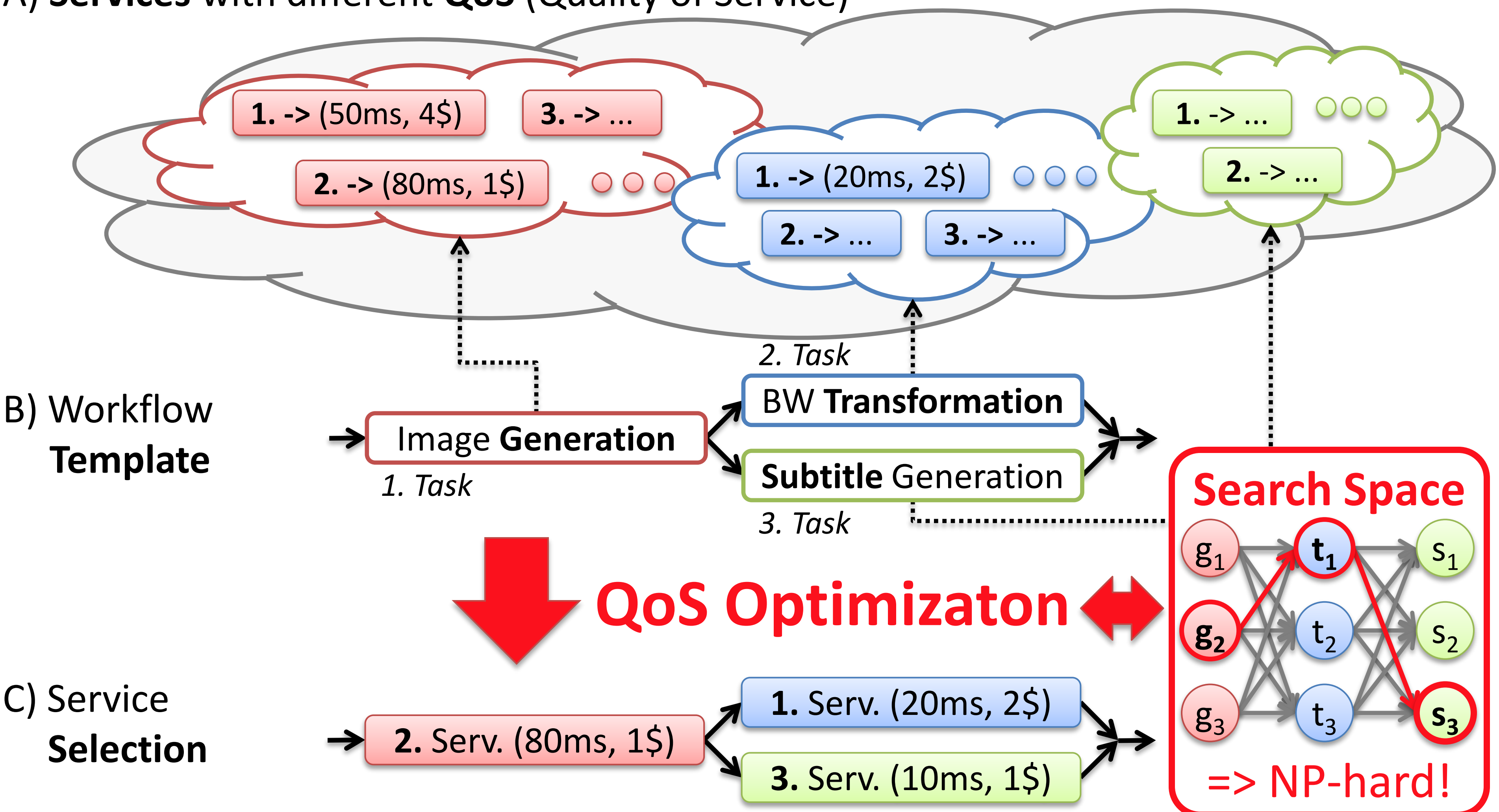
Number of **Service APIs**
@ www.programmableweb.com

Example of a Service Composition



QoS-aware Service Composition (QSC) Problem

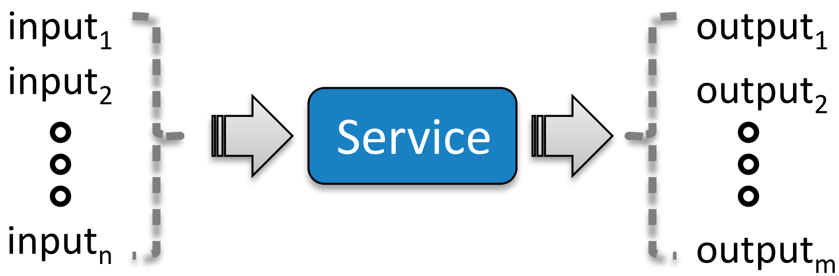
A) **Services** with different **QoS** (Quality of Service)



-> DEFINITION

Service

Form:



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:	**int** x
Outputs:	list of **ints** $\{d_1, \ldots, d_n\}$
Precondition:	$x < 10^{10}$
Postcondition:	$x = \prod d_i$ *with all d_i prime*

Price:	1$
Response time:	≤ 100 ms
Reliability:	99.5%

Functional Description

Non-Functional Description
≈ **Quality of Service (QoS)**

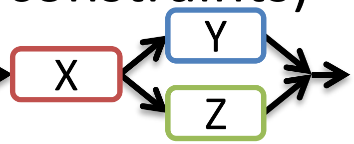
QSC Problem [ZBD+03]

Inputs (from the service user ☺)

– QoS **preferences** (=weights) → $util(q, wf, sel) := \sum_i w_i Q_i$ (simplified)

– QoS **requirements** (=constraints)

– Workflow **template** →



Other Formalizations

– Linear Integer Programming Prob.
– Knapsack Problem
– Graph Problem
– Scheduling Problem
=> **Sufficient** to some degree, **NP-hard**

QSC Problem.

$$\begin{array}{lrl} \text{given} & & (wf, (w_T, w_P, w_R), (c_T, c_P, c_R)) \in WF \times \mathbb{R}^3 \times \mathbb{R}^3 \\ \text{optimize} & & sel \in SEL \\ \text{such that} & & util(q, wf, sel) = \max_x \; util(q, wf, x) \\ & \textit{and} & q(wf, sel)(T) \leq c_T \\ & \textit{and} & q(wf, sel)(P) \leq c_P \\ & \textit{and} & q(wf, sel)(R) \geq c_R \end{array}$$

=> Search Space := #S$^{\#WF}$

#S := Services per Task

#WF := Size of Workflow

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:

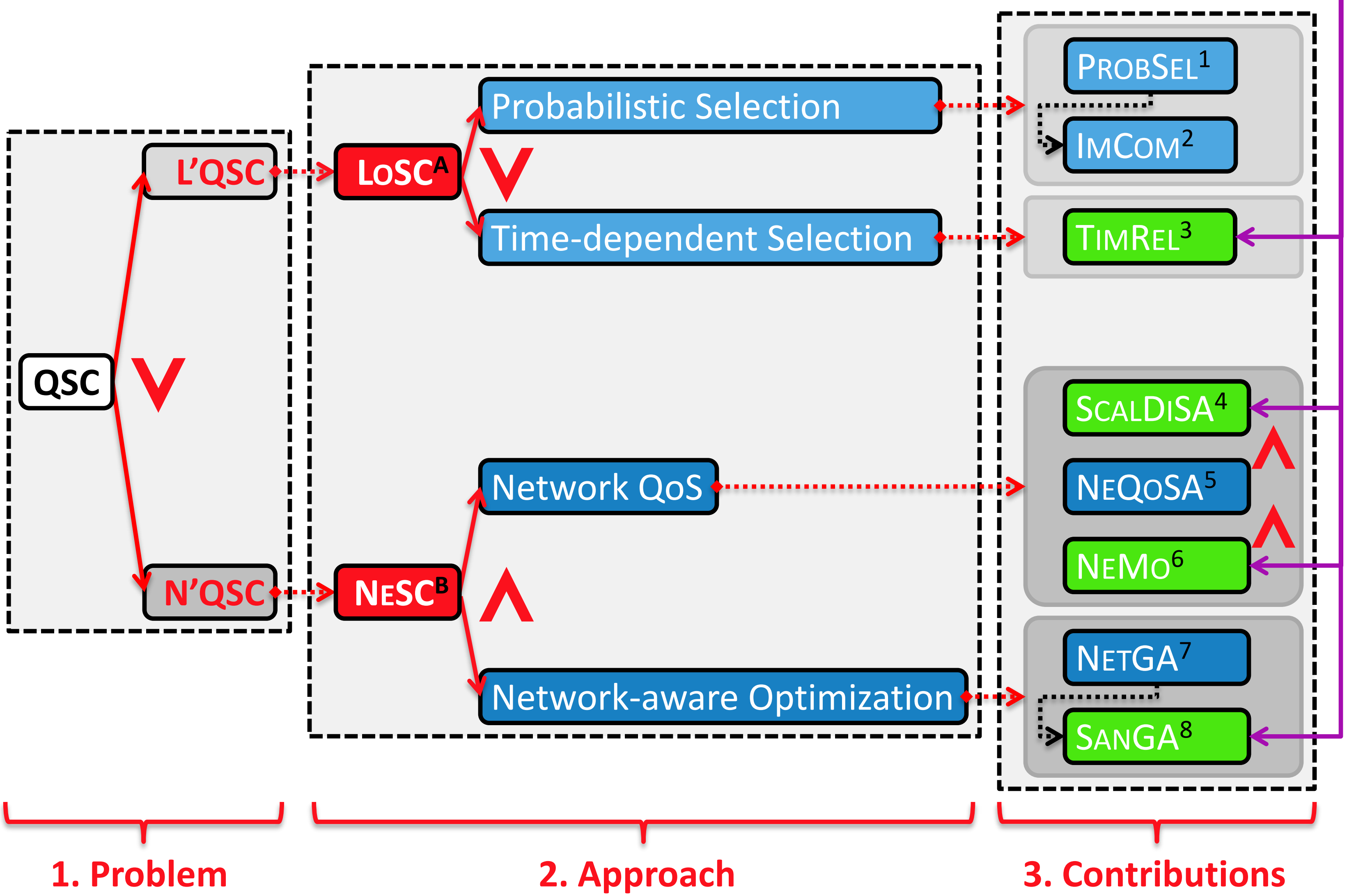
A) **Long-term** SC (**L'QSC** / **LOSC** ⇨ ①)

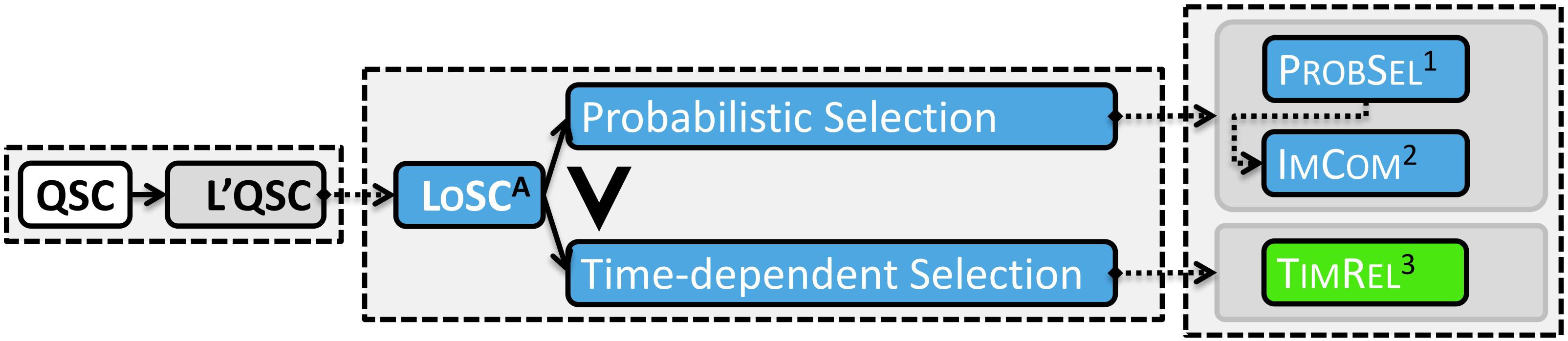
B) **Network-aware** SC (**N'QSC** / **NESC** ⇨ ②+③)

=> **Extend** **QSC** problem to **L'QSC**+**N'QSC** problems

=> Propose **effective** and **efficient** **LOSC**+**NESC** approaches

Approach: Focus





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
- Optimality+**Scalability**

2.
- **Better QoS** (+high reliability)
- **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)

PROBSEL[1]: Long-term QoS with Probabilistic Service Selection

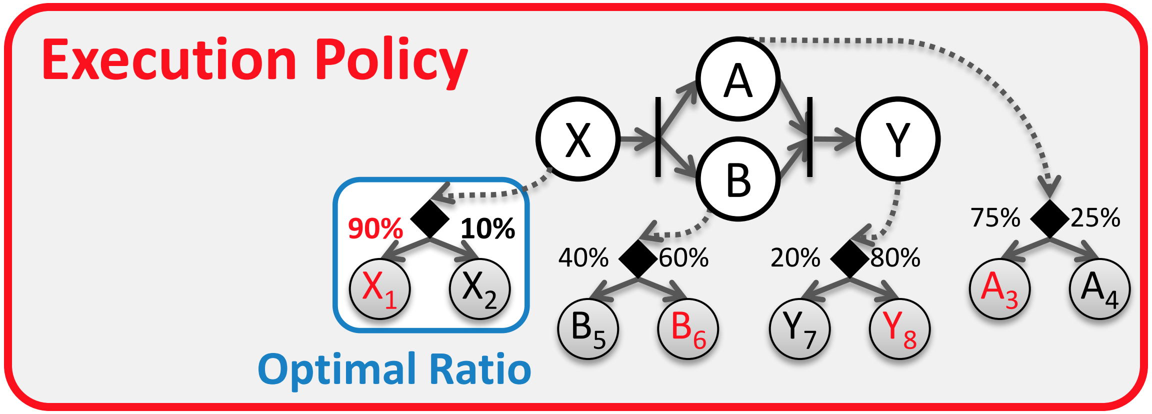
↳[KIH10a]

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

=> **Better QoS** + **Scalability**

=> Address part of **Challenge** ① (static service selection)

Probabilistic Service Selection



Executions

1. $X_1+A_3+B_5+Y_8$
2. $X_1+A_4+B_6+Y_7$
3. $X_1+A_3+B_6+Y_8$
4. $X_2+A_3+B_5+Y_8$
5. $X_1+A_3+B_6+Y_8$
6. $X_1+A_3+B_5+Y_7$

X_1:	**100%,**	**X_2:**	**0%**
A_3:	50%,	A_4:	50%
B_5:	50%,	B_6:	50%
Y_7:	50%,	Y_8:	50%

X_1:	**83%,**	**X_2:**	**17%**
A_3:	67%,	A_4:	33%
B_5:	50%,	B_6:	50%
Y_7:	33%,	Y_8:	68%

IMCOM²: Improved Time Complexity for the QSC Problem

↳[KIH11]

Custom Hill Climbing algorithm (**HC*/√SF**) with PROBSEL[1] as initial bias

=> Observed time complexity vs. HC: **$O(SF^4)$** -> **$O(SF^{1.5})$**

(...linear in problem size = #S * #WF = SF^2) (**SF** := #S := #WF)

Side Result!

TIMREL^3: Time-dependent QoS for Long-term Compositions

↳[KWIH12]

Solve complete L'QSC with a **Custom Genetic Algorithm**

+ a **variable number** of **time-dependent services** (doubling as backup services)

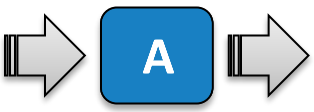
=> **Better QoS** (incl. high reliability) + **Scalability**

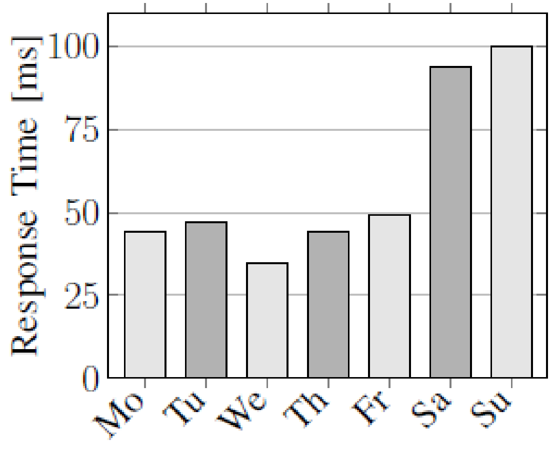
=> Address **Challenge** ① (static service selection)

↳ Cooperation with Florian Wagner

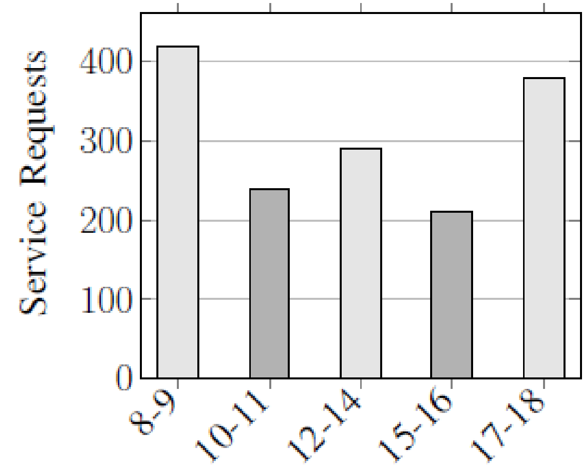
Time-dependent Service Selection

Probabilistic Patterns (QoS + Use)

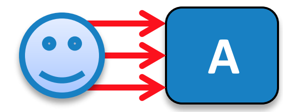




(a) Qos Pattern



(b) Usage Pattern



Time-dependent Execution Policies

- Mo-Fr Service A1 → Service A5 [in case of failure] → ○ ○ ○ [in case of failure]
- Sa/Su Service A5 → Service A1 → ○ ○ ○

=> Search Space: $\#S^{\#WF}$ -> $(\#S^{\#WF})^{\#TD}$

#S := Services per Task

#TD := Nr. of Time Discretizations

#WF := Size of Workflow

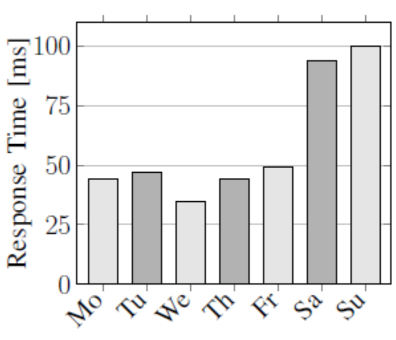
Custom Genetic Algorithm

Adaptive GA

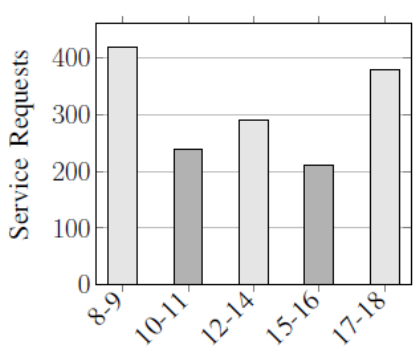
- Mutate+ → Choose values $\notin$ group
- Crossover+ → Avoid **duplicate/empty** group members
- AdaptGroups → Due to **reliability** constr. and current **rel.** (per Task, increase or decrease group size)

Probabilistic QoS Model

Probabilistic Patterns (QoS + Use)

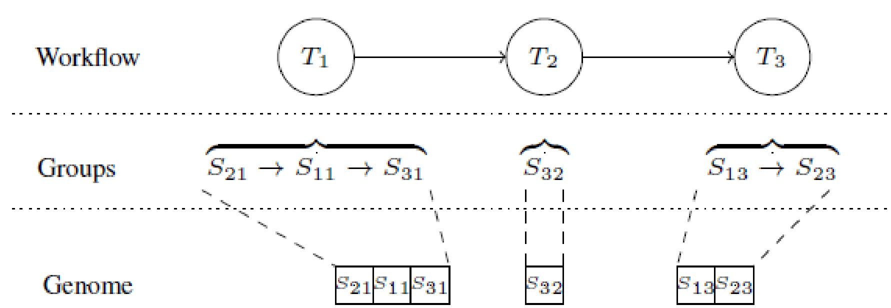


(a) Qos Pattern



(b) Usage Pattern

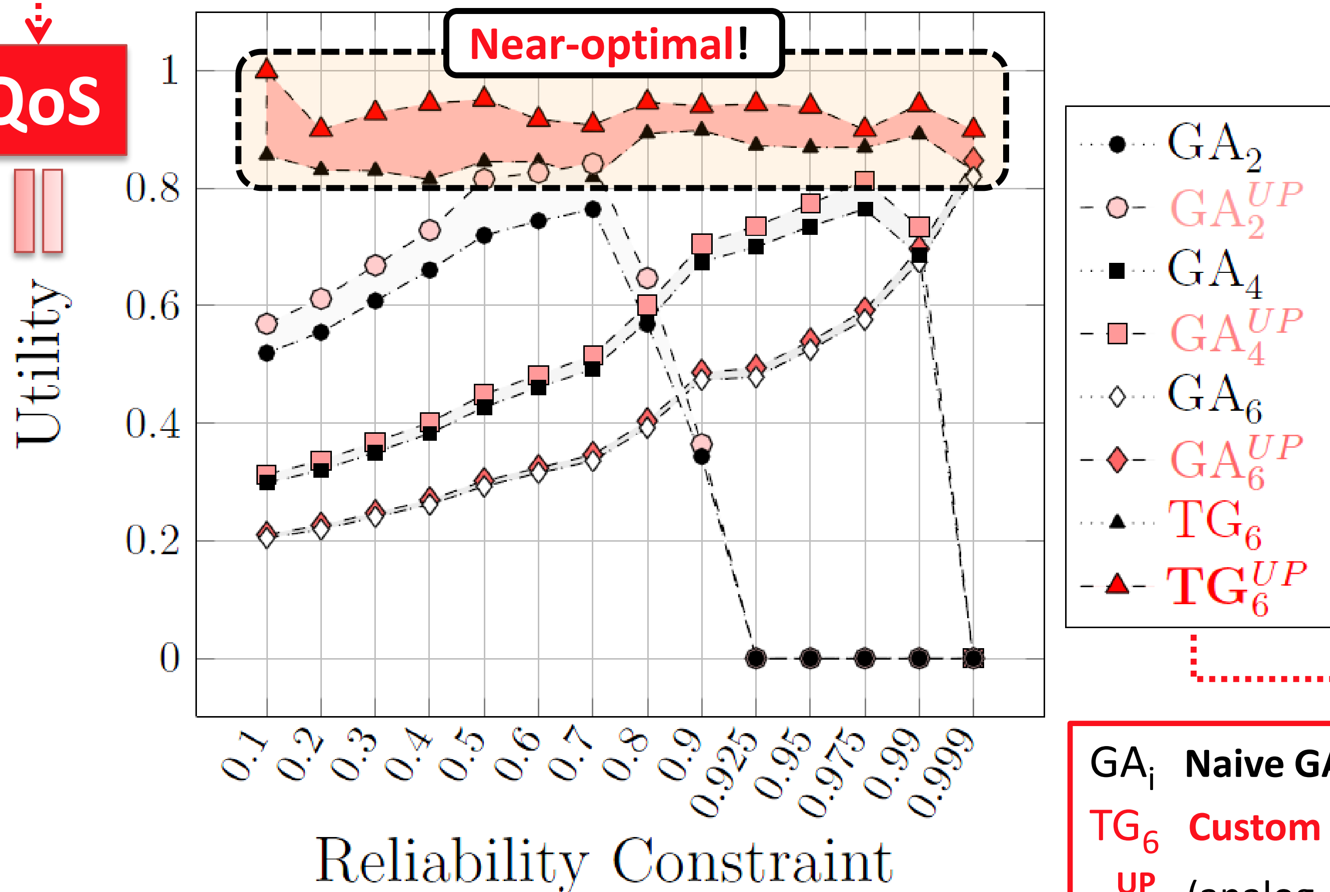
Encoding with variable length



Evaluation: Better QoS (=Utility)

QoS

||



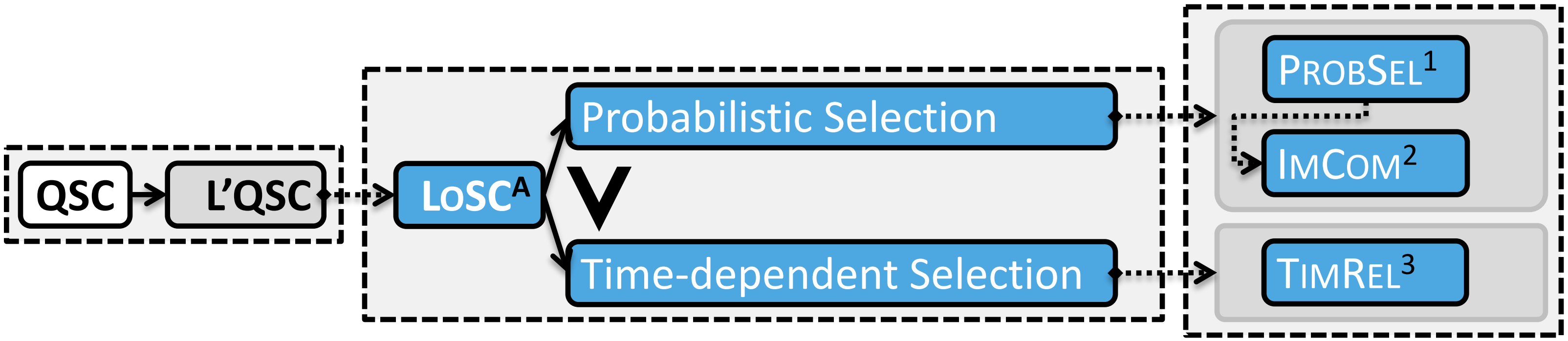
GA_i **Naive GA** with **static group size i**

TG_6 **Custom Adaptive GA**

_UP (analog, with *time-dependency*)

50^{10}

3-A) LONG-TERM SCs: Conclusion



1. Solved partial **L'QSC**

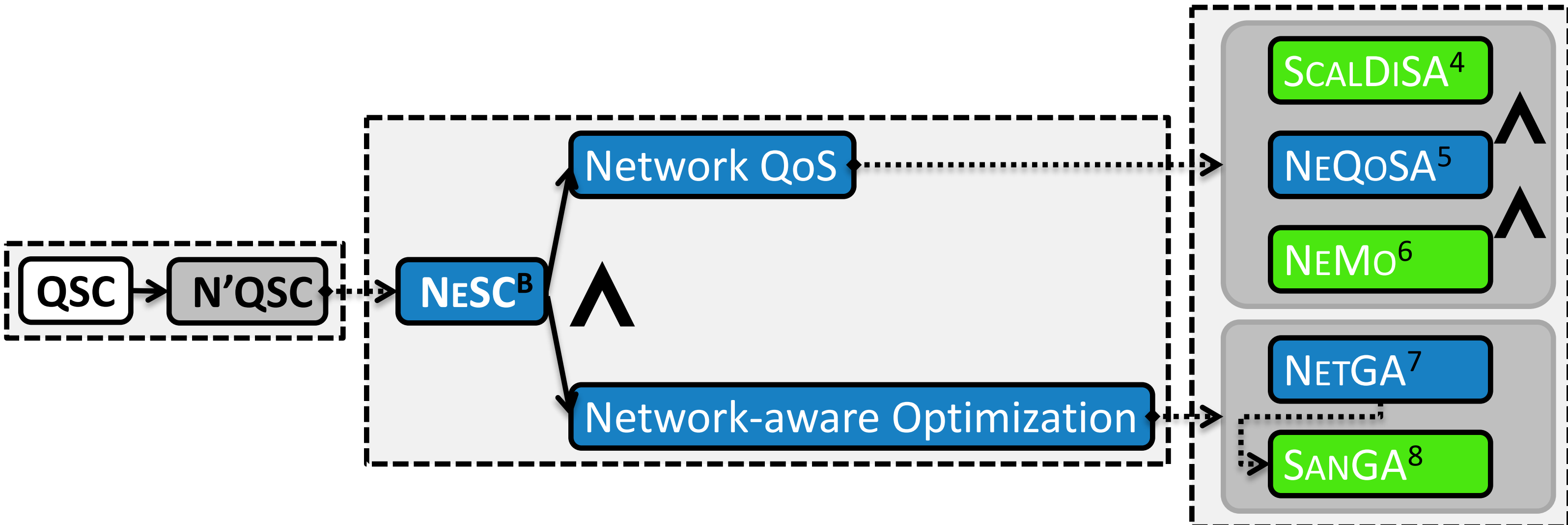
– Approximated orig. **QSC**

2. Solved complete **L'QSC**

=> **Better QoS** + **Scalability**!

=> Addressed **Challenge ①**!

(static service selection)



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**

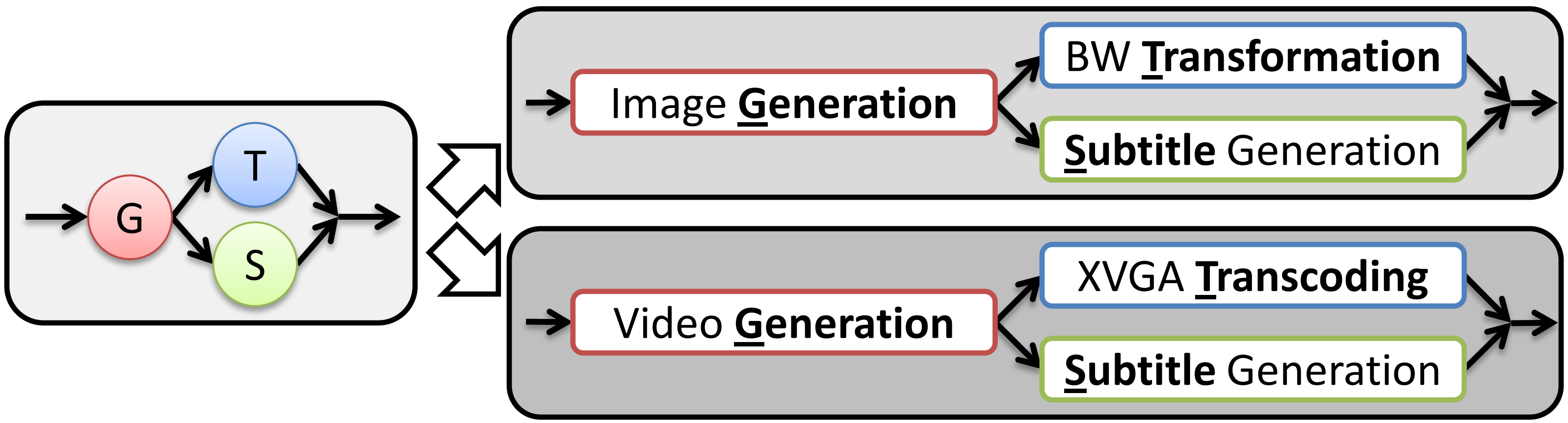
– Distributed service overlay networks (Dijkstra+QoS ratio), Li et al. [LHD+07]
=> **no** QoS constraints + **no** global QoS optimization
+ does **not scale** vs. our appr. (-> service **instances** per task, evaluated in 2.)

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

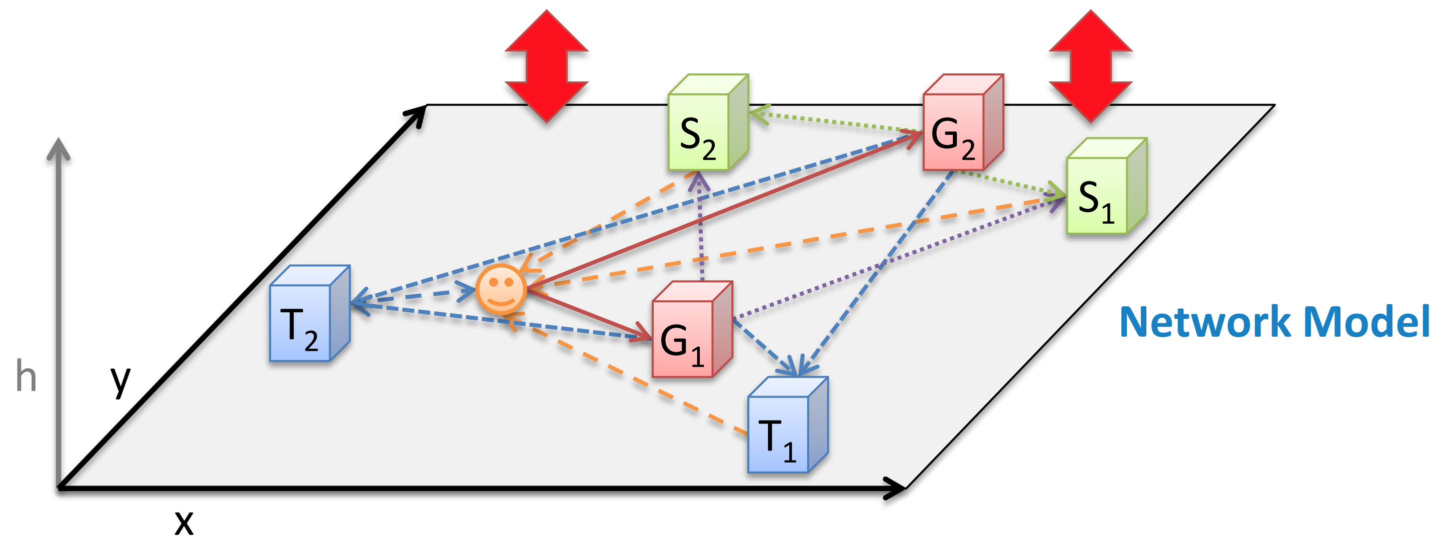
– Distributed execution acc. to chemical paradigm [LMJ10] or by agent model [FPT10] (both not solved)
=> **no** QSC formalization + **no network QoS**

Motivation (1/4): Distributed Scenario

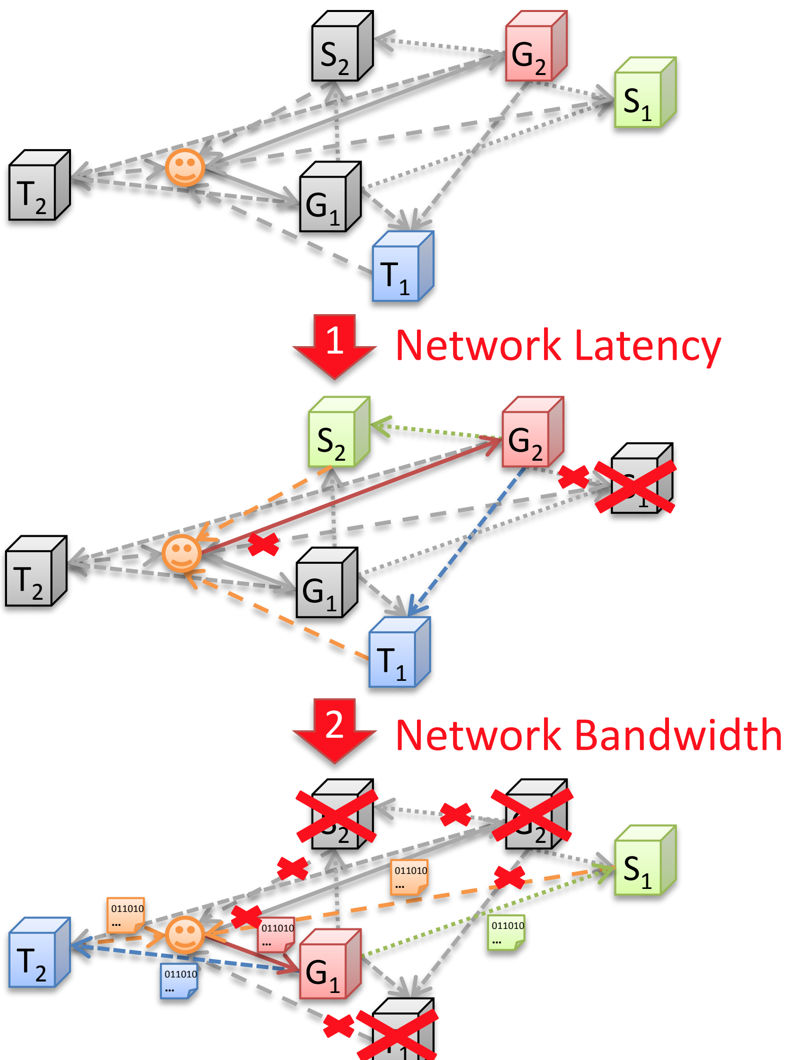
Media Tr. ➠ **Workflow**



Deployment Situation



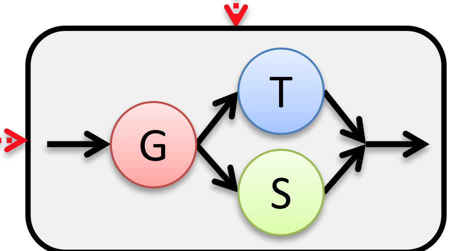
Motivation (2/4): Network-aware QoS



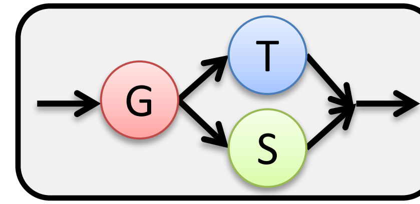
=> 80ms
(just exec.)

=> 155ms
(distributed)

=> 1691ms
(100 MB)

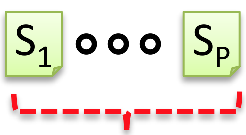


Motivation (3/4): Complexity ->Network-aware Optimization



Standard

– P providers for Task T
(e.g. Amazon, Google, …)



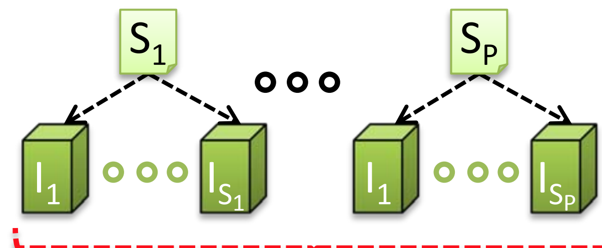
P choices **per Task**

=> **[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, …)



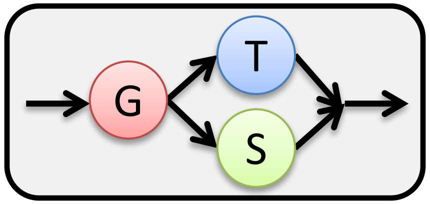
$P \cdot \sum I_k$ choices **per Task**

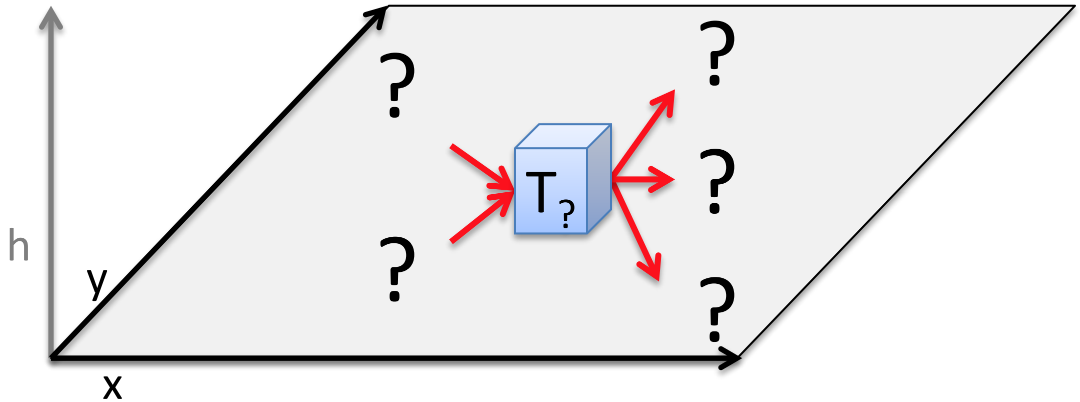
=> [50-100] x **[20-120]** = **[1000-12000]** choices (per Task)

=> Search Space: $\#S^{\#WF}$ -> $(\#I \cdot \#S)^{\#WF}$

#S := Services per Task
#I := Instances per Service
#WF := Size of Workflow

Motivation (4/4): Dependencies
->Network-aware Optimization

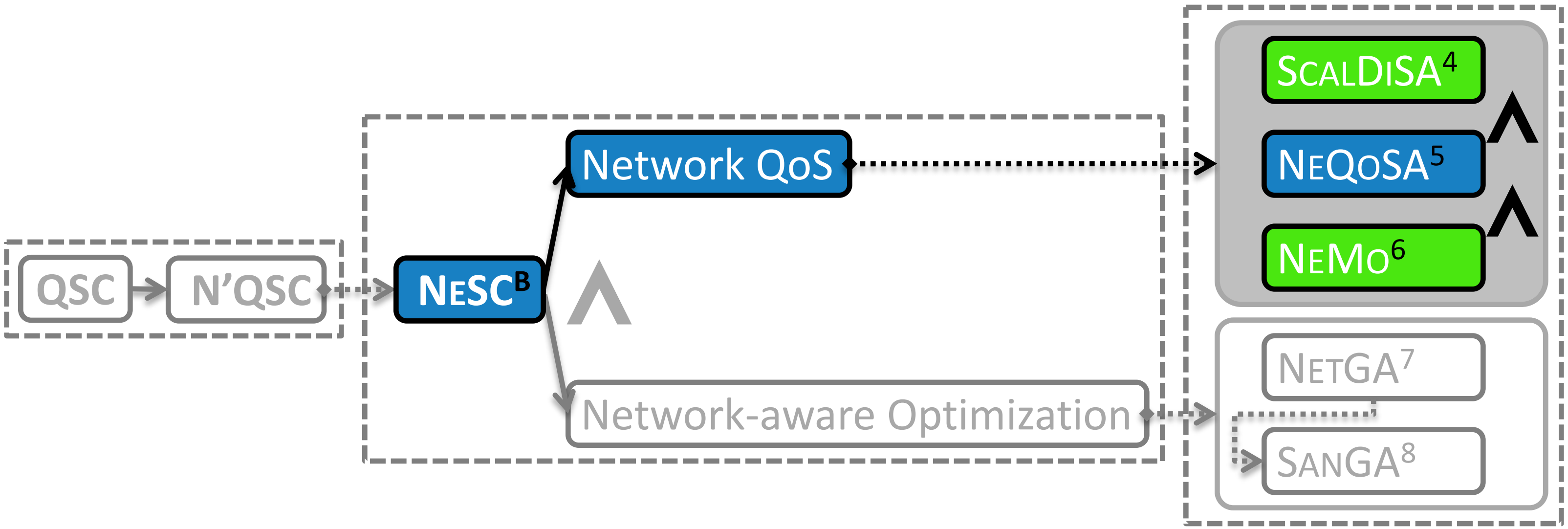




Change of service for task **T**

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

New dependencies between **each task** and its predecessors and successors**!**



NETWORK QOS

Distributed Architecture

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

+ **Network Model**

=> Challenge ②

(network-independent QoS)

SCALABLE DISTRIBUTED SERVICE ARCHITECTURE[4]

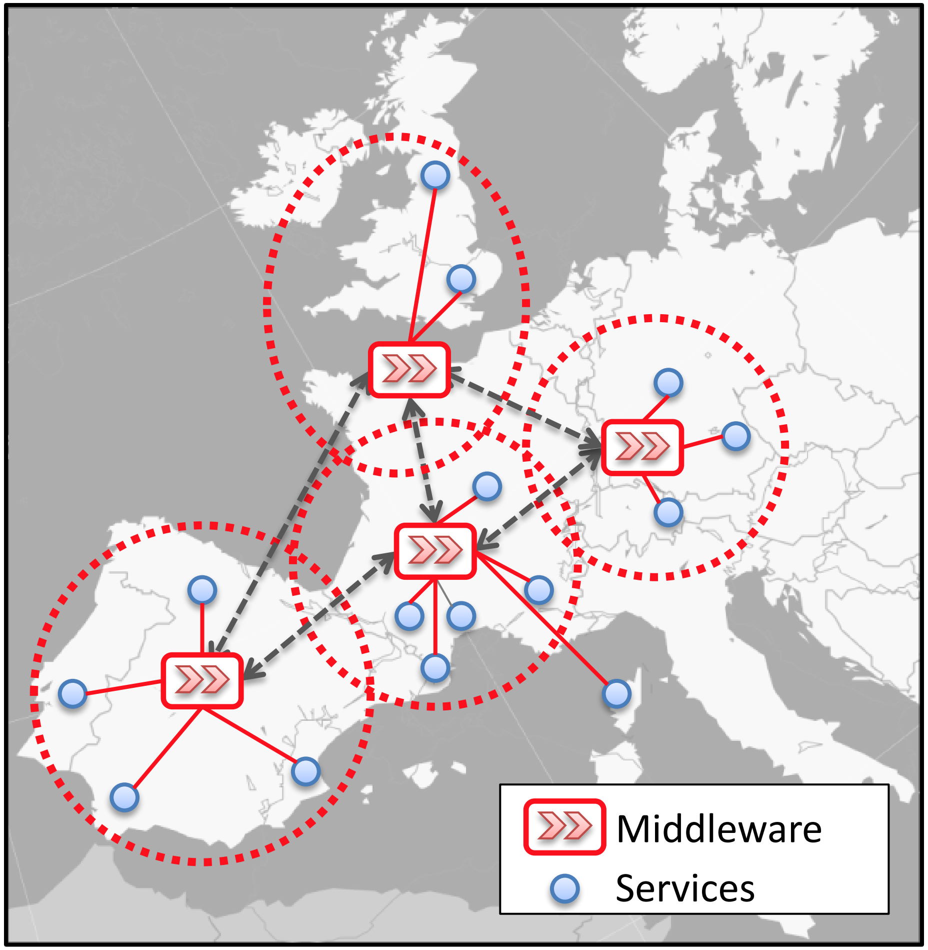
↳[KIH12a]

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

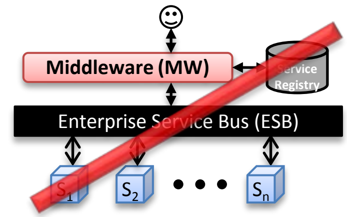
=> **Near-optimal** latency + **Scalability** in terms of distributed-ness

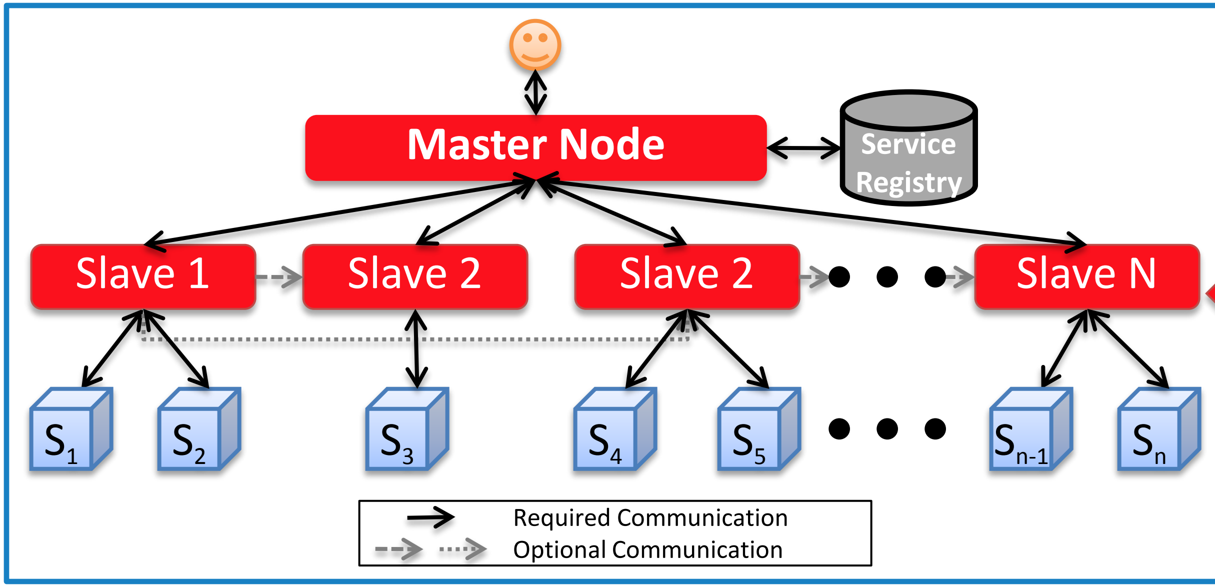
=> Address part of **Challenge ②** (network-independent QoS)

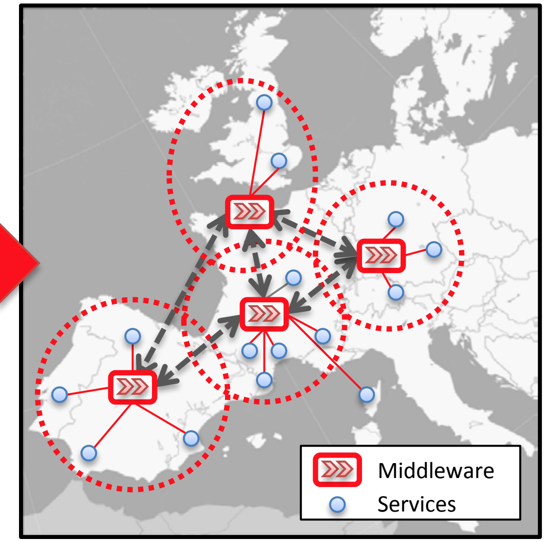
Distributed Architecture -> Distr. Exec.



Architecture for a Middleware

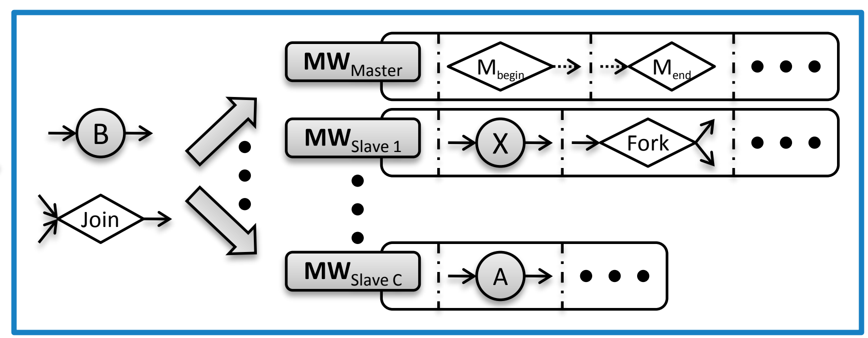






Distributed Architecture

– one **master node** (=user)
– **flexible** number of **slave nodes**

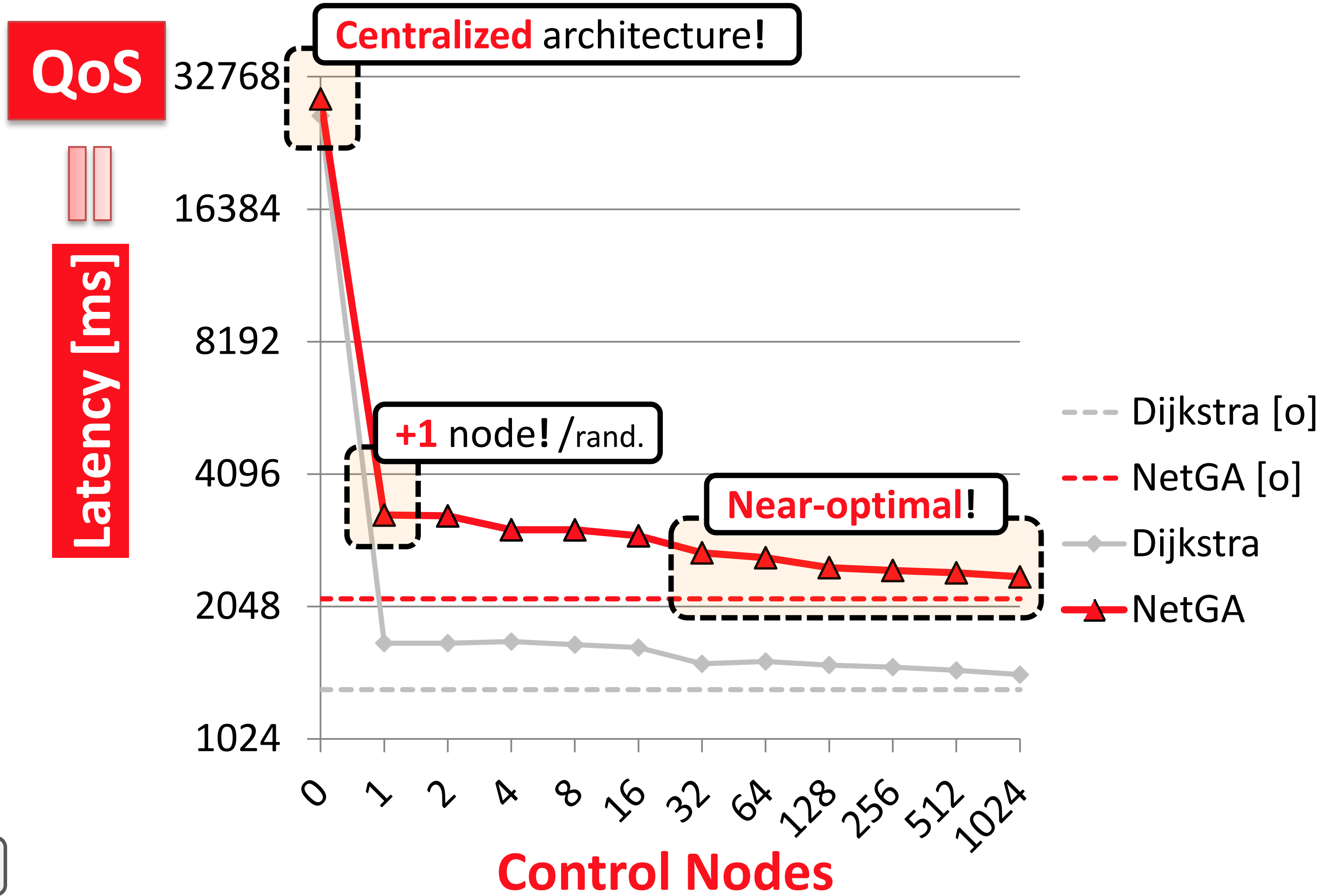


Distributed Execution

↳ **Integration** with **NeQoSA**[5] and **NeMo**[6]

Evaluation: Near-Optimality

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



500^{40}

NETWORK MODEL (NEMO[6])

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

=> Address part of **Challenge ②** (network-independent QoS)

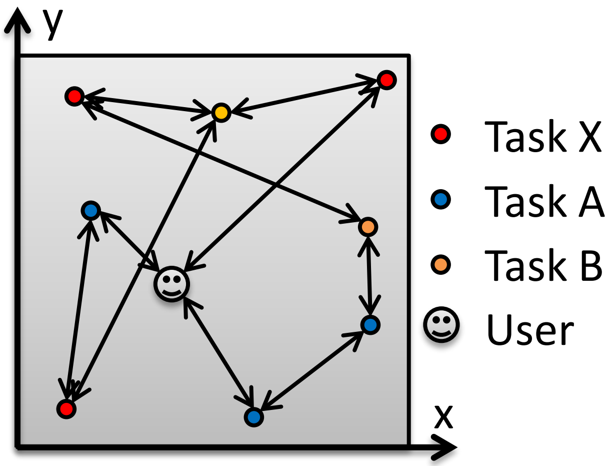
General Model [Framework]

Challenges

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

Framework

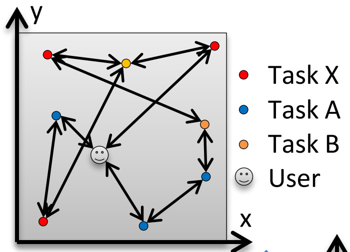
- Use a **Euclidean network model**, e.g. Vivaldi [DCKM04]
- **Project** onto **2D plane** for optimization algorithms
- **Augment** 2D representation



Augmentation

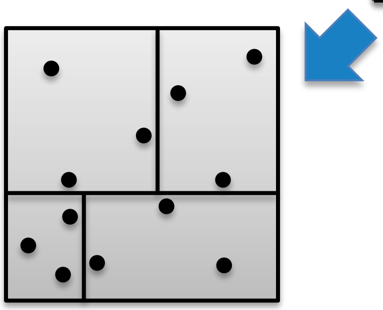
Use **K-D Tree**

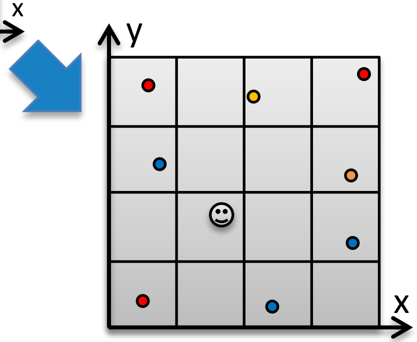
(↳ **SANGA**[8])

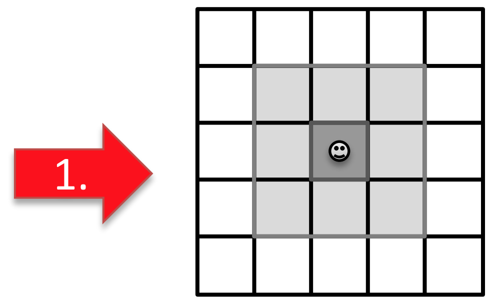


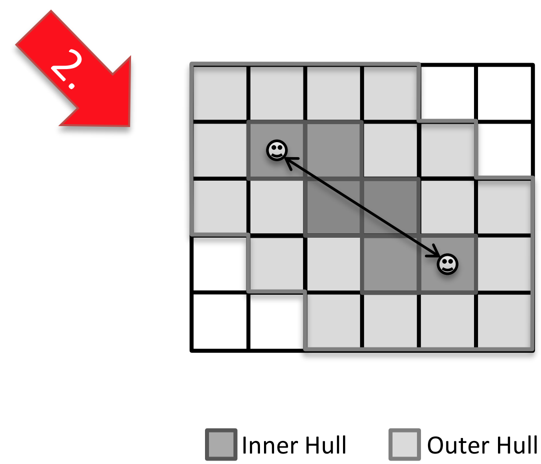
Custom Locality-sensitive Hashing Scheme

(↳ **NETGA**[7])



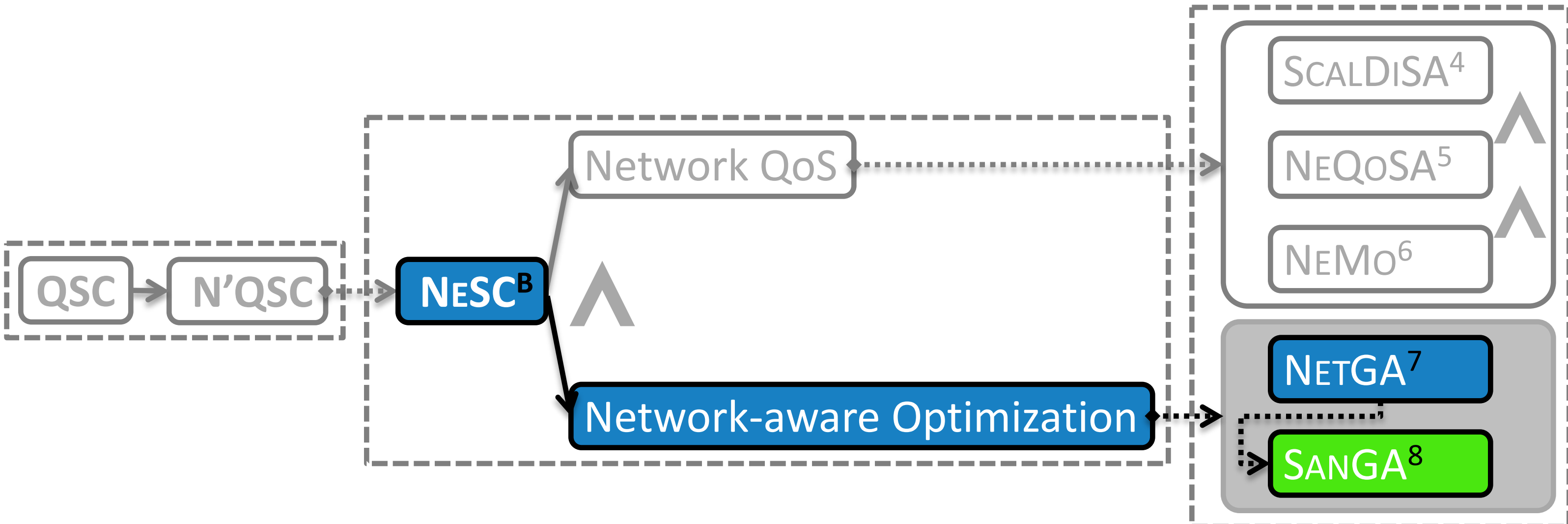






↳ **Find close locations/services**

1. Around a network location
2. Between two network locations



NETWORK-AWARE OPTIMIZATION

Network-aware Genetic Algorithm (NETGA[7])

-> + Self-adaptive network-aware Genetic Algorithm (SANGA[8])

=> Challenge ③
(network-unaware Optimization)

SELF-ADAPTIVE NETWORK-AWARE GA[8]

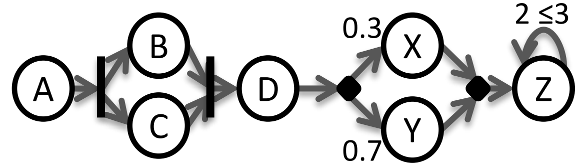
↳[KIH13]

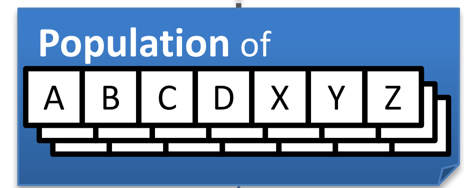
Genetic Algorithm which **self-adaptively** balances
network-aware **operators** vs. *standard operators* (extending **NETGA**[7])

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

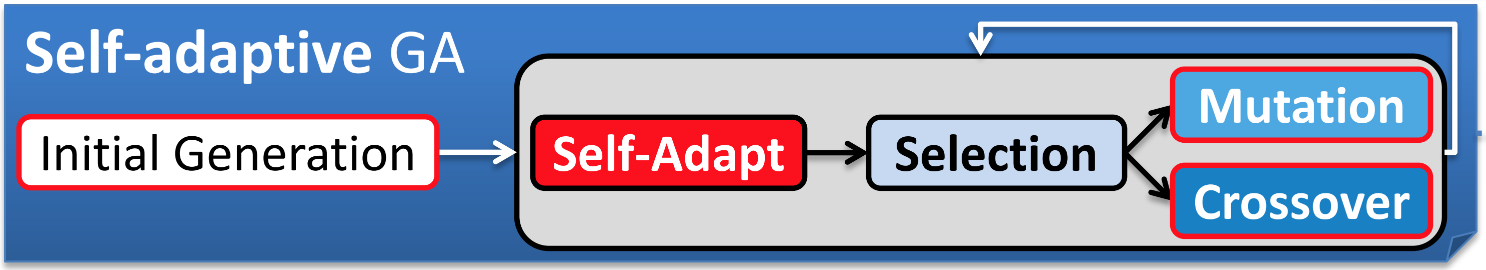
=> Address **Challenge ③** (network-unaware Optimization)

SANGA[8]





Standard Self-adaptive GA:



+ Custom Operators for:

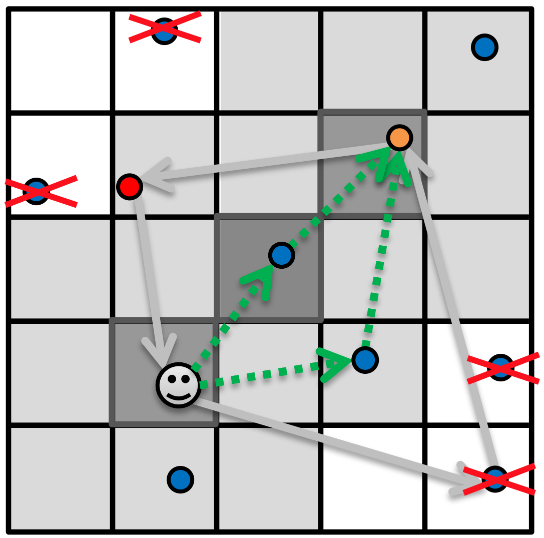
- Initial Generation (network-aware + general)
- **Mutate** (network-aware)
- **Crossover** (2x: network-aware + other QoS)

+ Custom Self-adaptation Rules

↳ Use **GA** for **QSC** [CDPEV05]

NetMutation: Mutate Operator

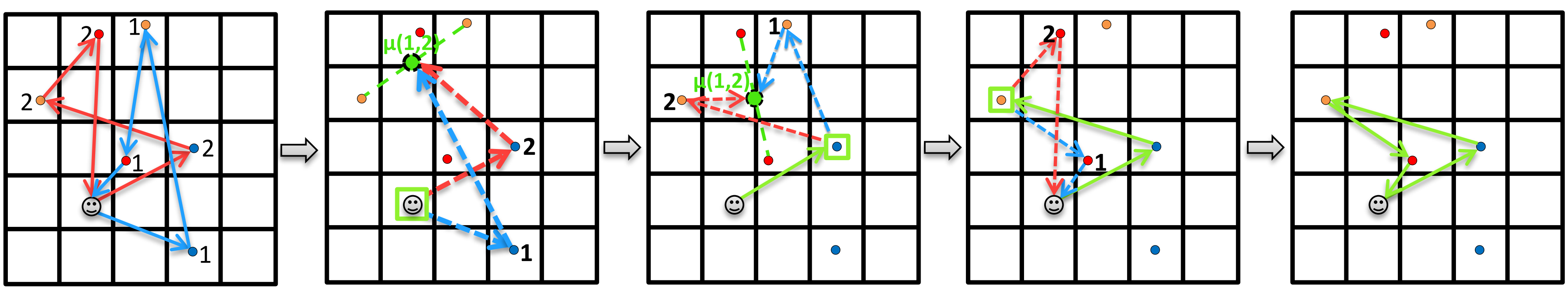
Replace one service of the current selection with one of the **close services** by using NEMO[6] (consider previous and next service in workflow)



– **Standard mutate**: search space
– **NetMutation**: search space
=> most **probable** choices

NetCrossover: Crossover Operator

Combine close services from parents by **NEMO**[6] (randomly in proportion to distance from previous + next service)



Parents 1 & 2 | **Choose 1. Service** | **Choose 2. Service** | **Choose 3. Service** | Final Offspring

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

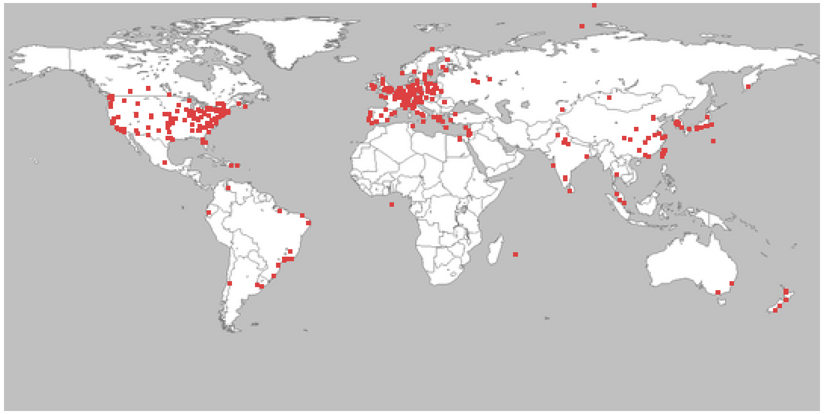
③ Local mutate/crossover ops

- **Useful** ops? -> 100%
- **Max** imp. ratio -> 80%
- **Avg** imp. ratio -> 20%

Based on Probability Matching, Adaptive Pursuit, and Power Probability Matching. (see [KIMK12])

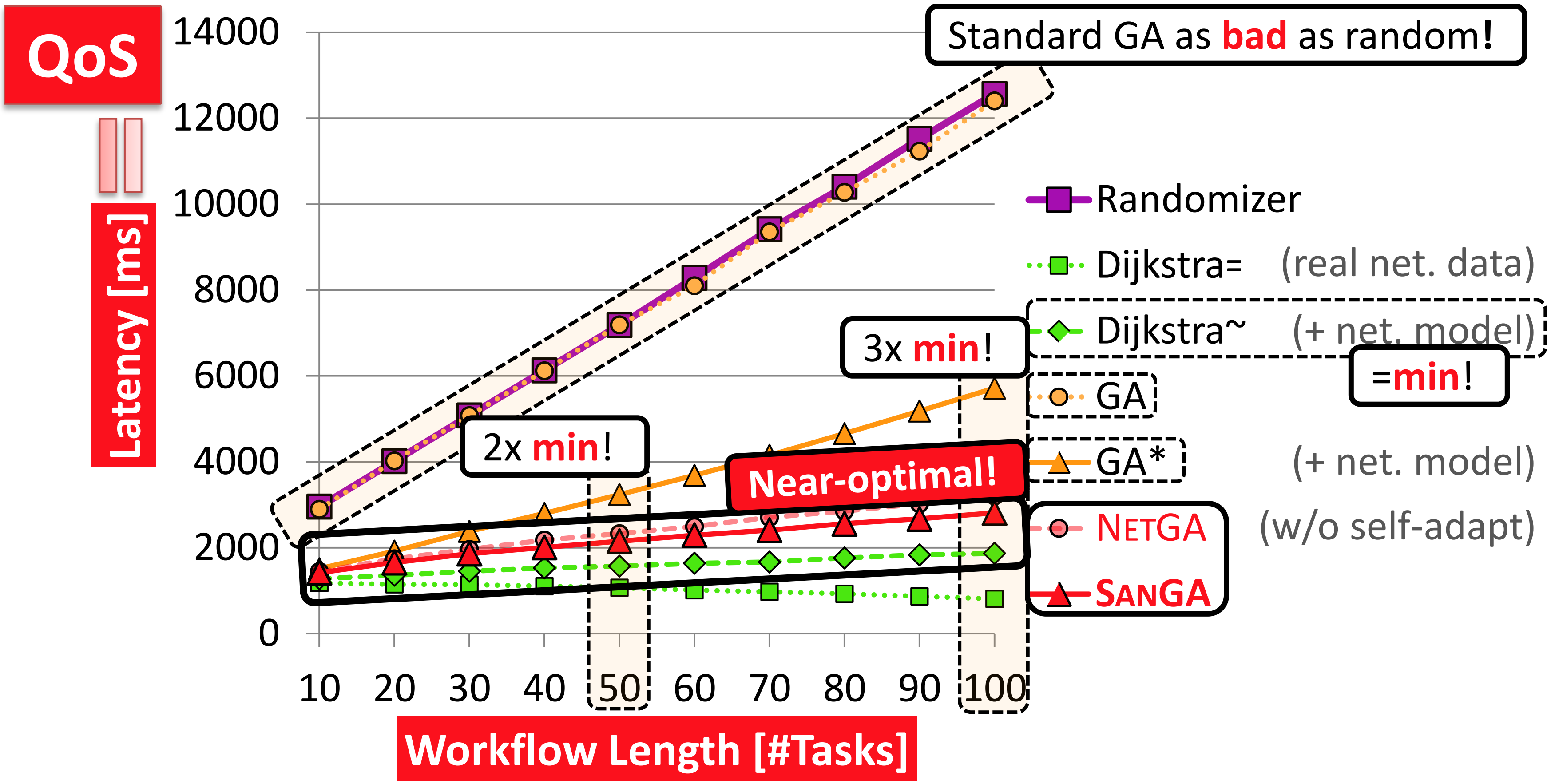
Evaluation: Network Dataset [Pro08]

- **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



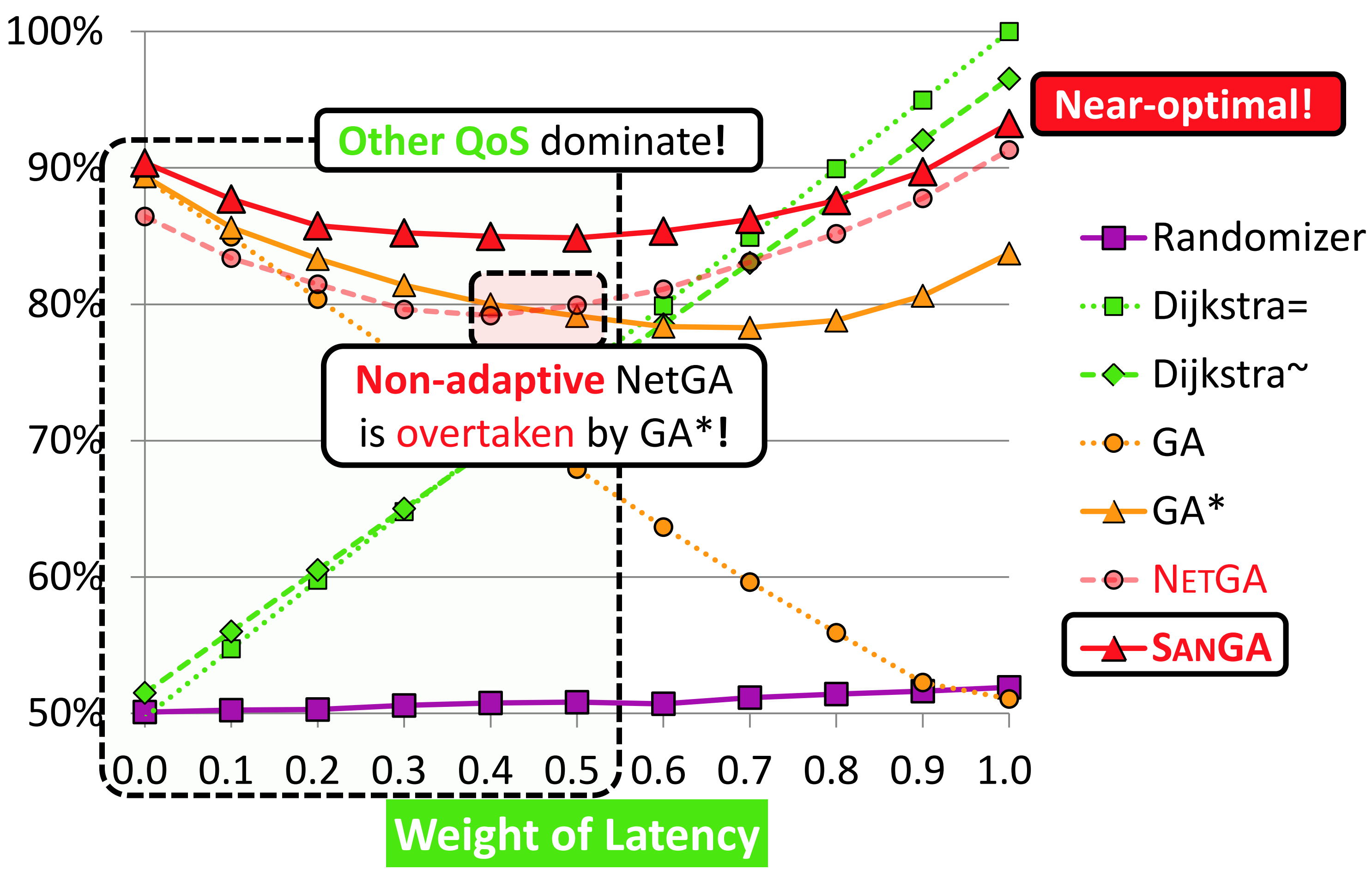
500#WF

Evaluation: Near-optimal Other QoS

QoS

=

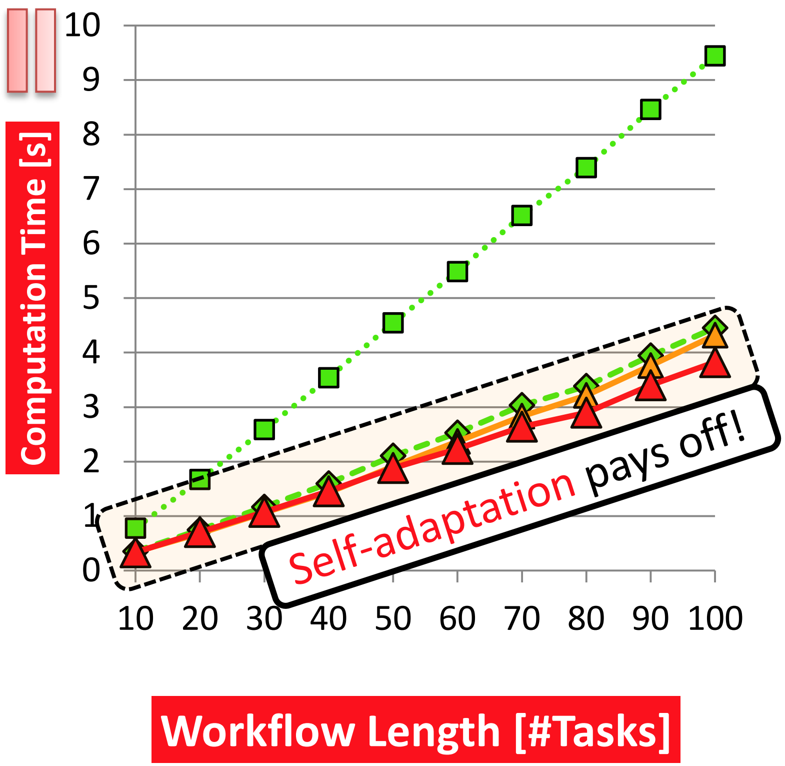
Utility

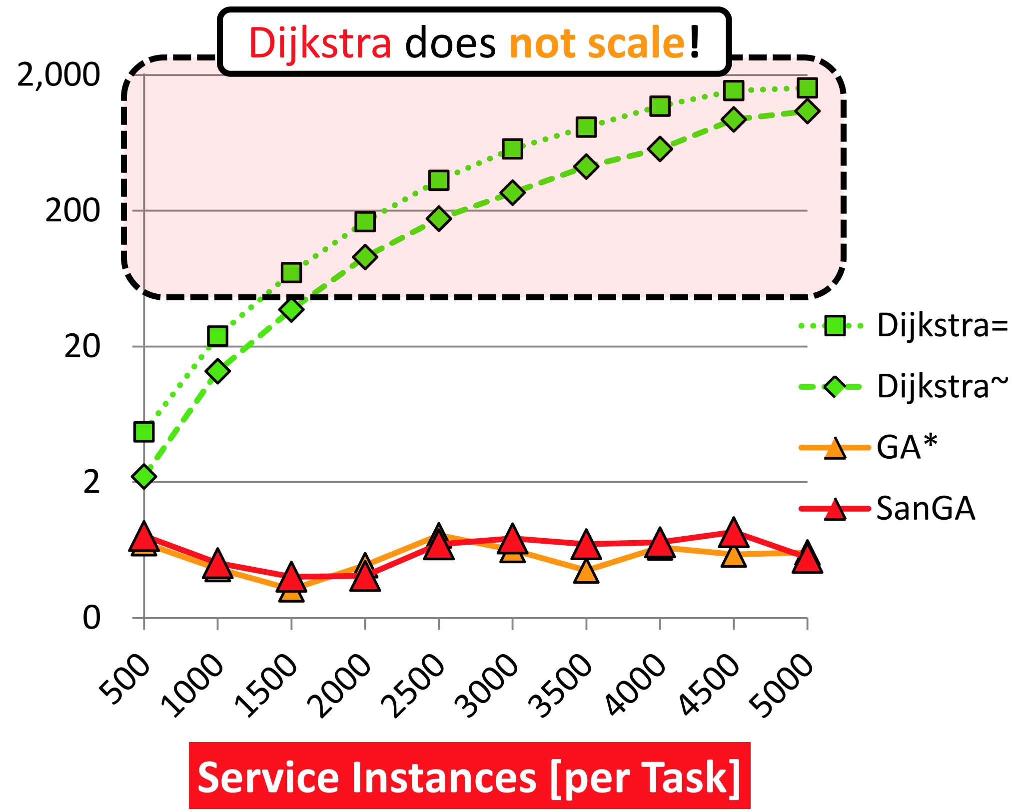


500^{50}

Evaluation: Scalability

Time

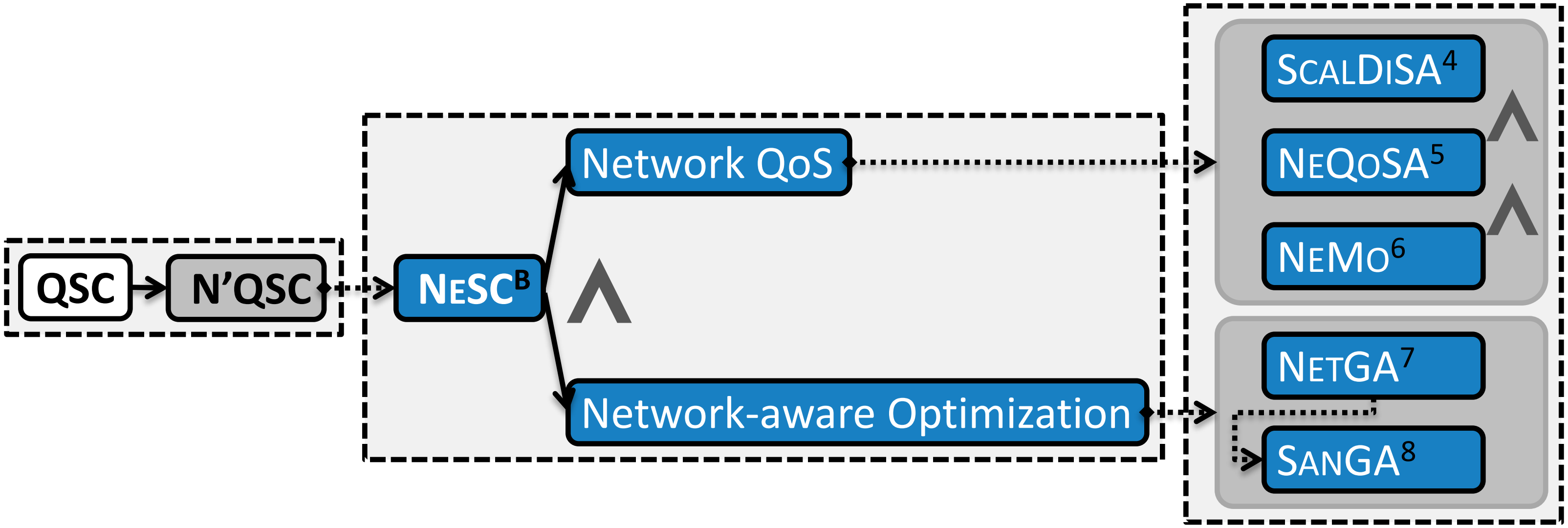




500#WF

#S50

3-B) NETWORK-AWARE SCs: Conclusion



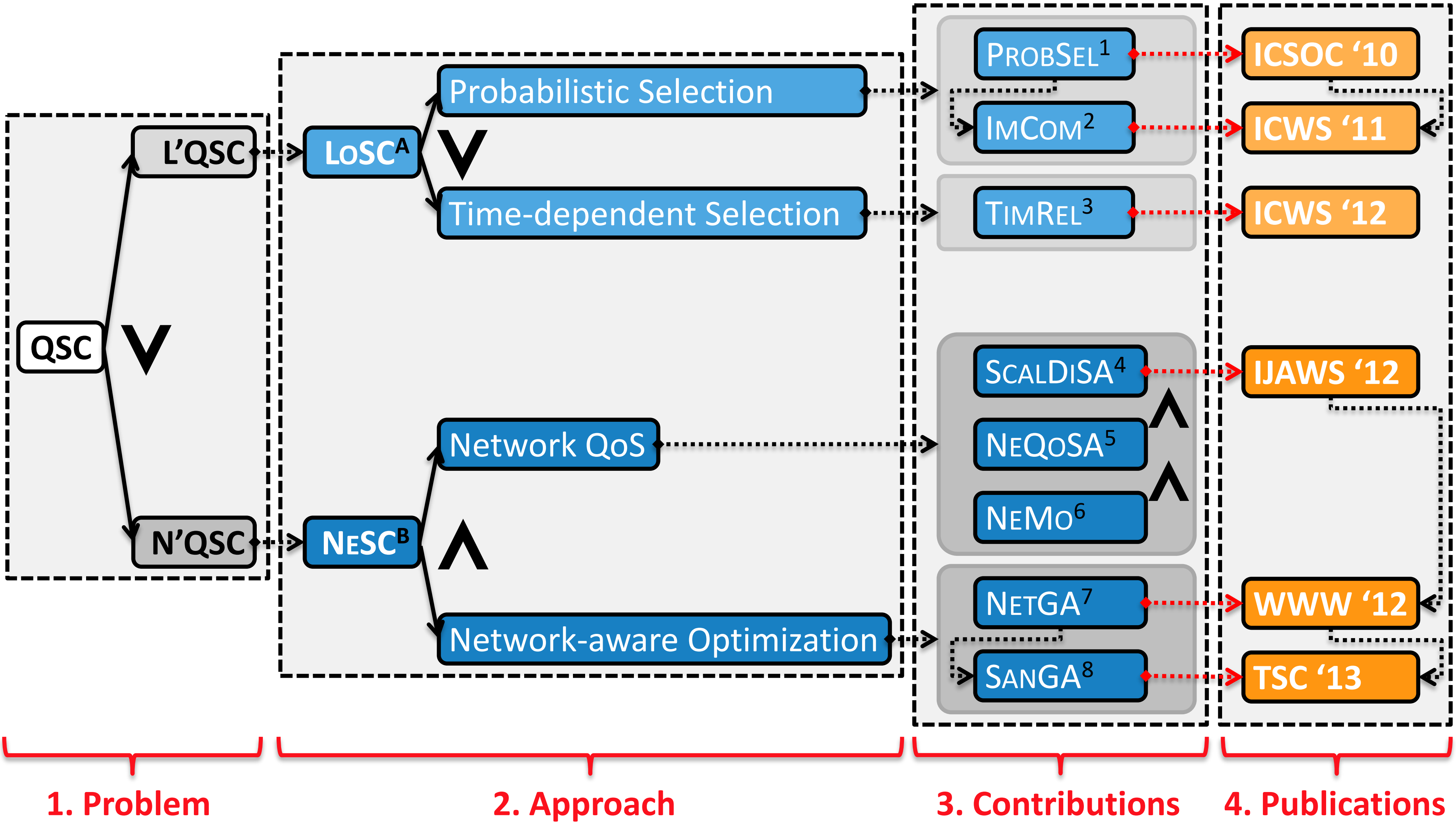
1. Solved **N'QSC** for **latency**
   (**bandwidth** ext. possible out-of-the-box)
2. Balanced **specialized operators** (net.)
   versus general operators (general QoS)

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

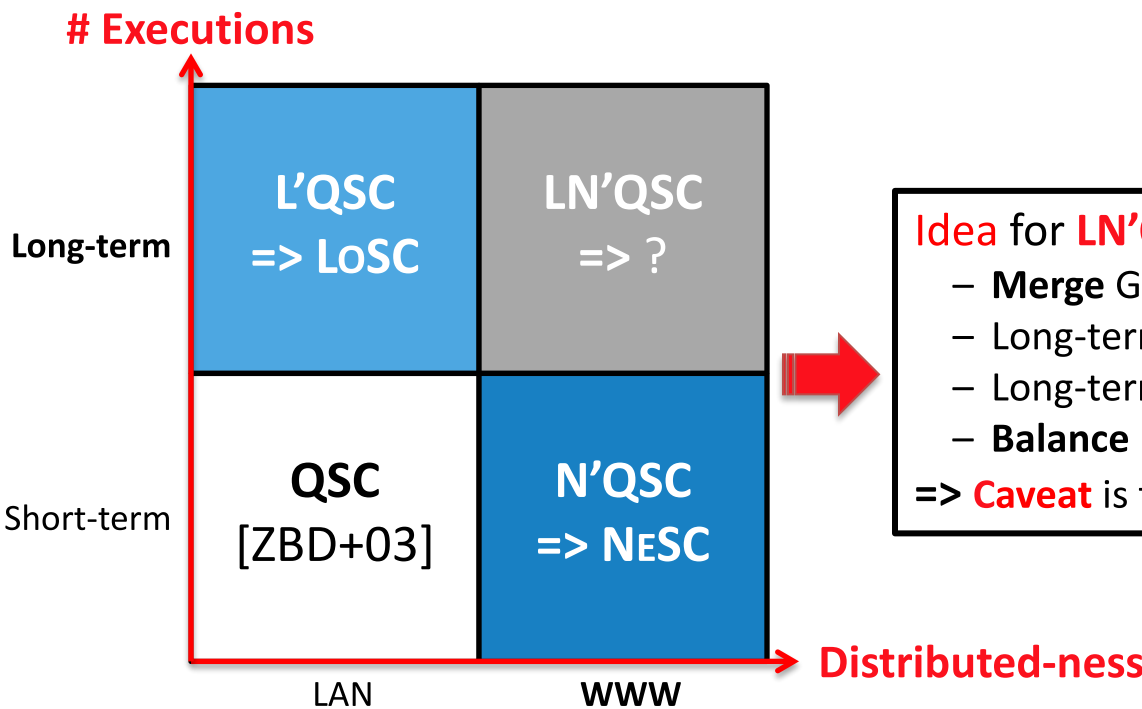
4. Conclusion

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

PhD Overview



Big Picture



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**

Outlook

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!