Zurich University
of Applied Sciences

School of Engineering
InIT Institute of Applied Information Technology

Zurich University of
Applied Sciences (CH)

Università degli Studi di Messina

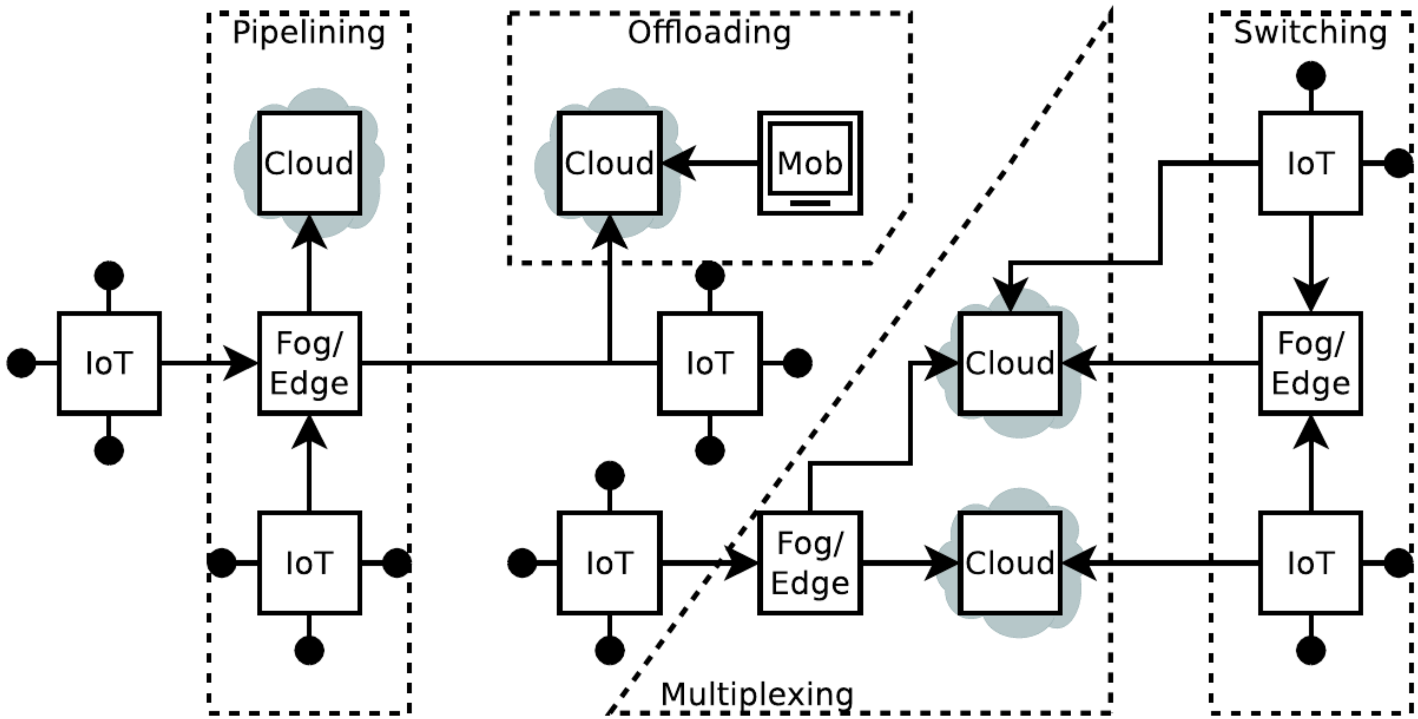
University of
Messina (IT)

Rule-Based Resource Matchmaking for Composite Application Deployments across IoT-Fog-Cloud Continuums

Josef Spillner, Panagiotis Gkikopoulos, Alina Buzachis, Massimo Villari

Dec 9, 2020 | 2nd CIFS @ UCC | ~~Leicester~~ Online

Context: Evolving Continuums...



- Operational efficiency & predictive maintenance in factories
- Control systems for optimal heating (NMPC)
- Intelligent transport systems (traffic counters & classifiers)
- Hospital automation
- many other use cases & osmotic scenarios...

Assignment Problem

Assign applications/services (A) to resources (R)...
... depending on constraints & preferences

Scope	Name	Values (examples)
A, R	memory	128 MiB, 2 GiB
A, R	runtime	python:3, java
A, R	latency	5 ms
A, R	duration	900 s
A, R	zone	intranet, dmz, internet
A	vulnerability	backdoor, CVE-477
A	consistency	true, false
A	complexity	high, medium, low
A	port	9233
R	country	gb, cn
R	trust	high, low
R	billing	monthly, pay-per-use, free
R	gpu	true, false

Assignment Problem Solutions

«Kuhn's Hungarian Method»

- **Combinatorial optimisation based on cost functions**
- **Background: assignment of military staff, 1955**

Individual	qualifies for job(s)
1	1, 2, and 3
2	3 and 4
3	4
4	4

Qualification matrix
→ row minimisation

$$Q = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

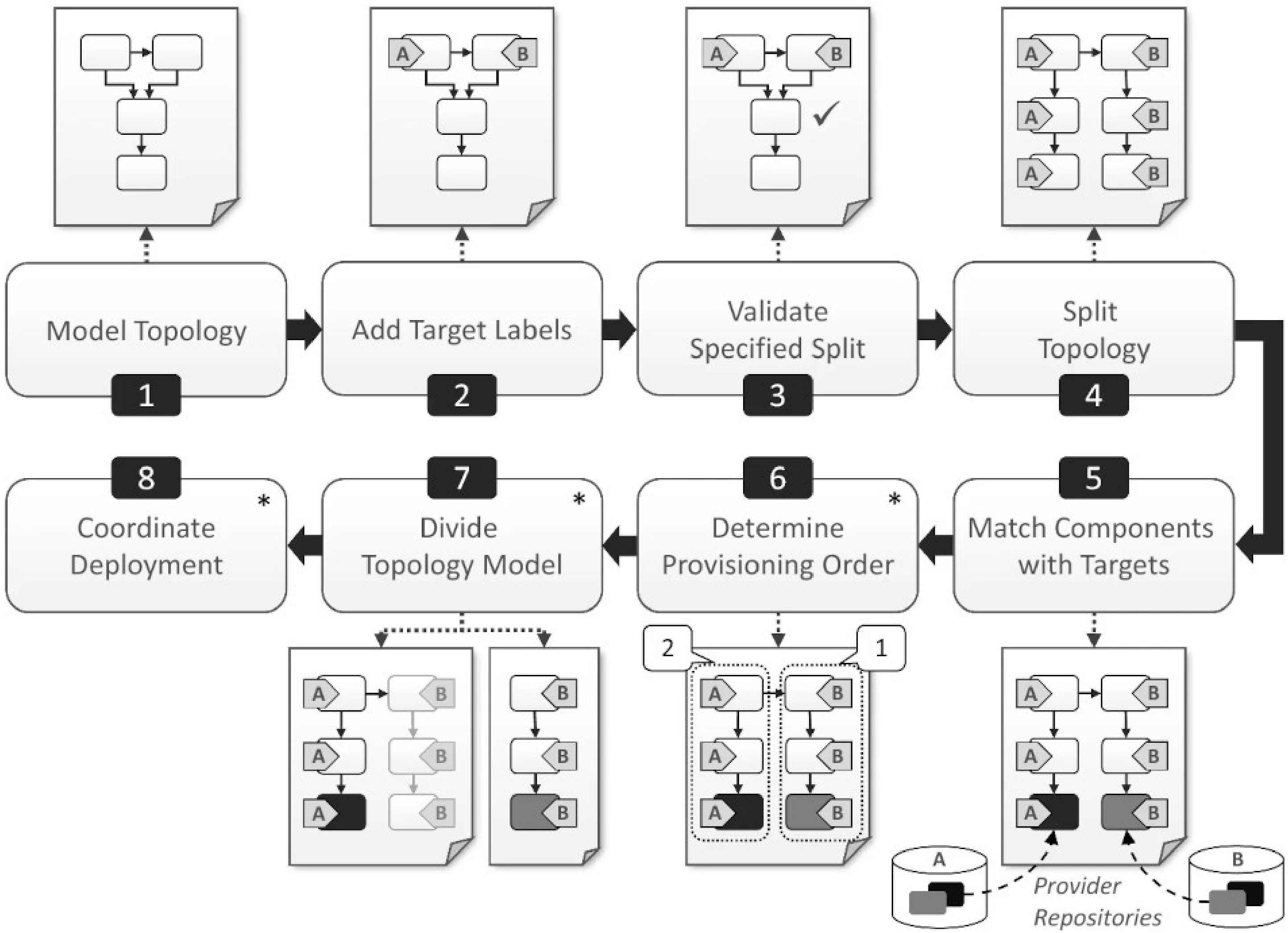
Algorithms with $O(n^3)..O(n^4)$ complexity

Assignment Problem Solutions

- **Fast iterative combinatorial algorithm ⊛**
  $O(n \times \frac{n}{2})$ for $|A| = |R| = n$
  - **greedy - may get stuck**
- **Recursive tree search algorithm ⊛**
  $O(n \times \frac{(n-1)^2}{2})$
  - **backtracking to avoid dead-ends**
- **Simple greedy algorithm**
  - **single optimisation target (e.g. CPU | memory) over sorted list**
- **SAT solver**
  - **minimise/maximise multiple targets (e.g. CPU & memory)**
  - **known property: finds solution if exists**

Assignment Problem in TOSCA

Split-and-match algorithm



J. Spillner, P. Gkikopoulos, A. Buzachis, M. Villari: Rule-Based Resource Matchmaking for Composite Application Deployments across IoT-Fog-Cloud Continuums, CIFS 2020

Knowledge Problem

A Knowledge

MAO - Microservice Artefact Observatory

- **static & dynamic assessment**
- **various artefact types (Docker images, Helm charts, Lambda functions/SAMs, ...)**
- **tracking of evolution over time**
- **consensus-based ground truth in federation**

[P. Gkikopoulos, J. Spillner, C. Mateos, A. Teyseyre: Given 2n Eyeballs, All Quality Flaws Are Shallow. Middleware 2020 / https://www.youtube.com/watch?v=nc9zLlA7Kj8]

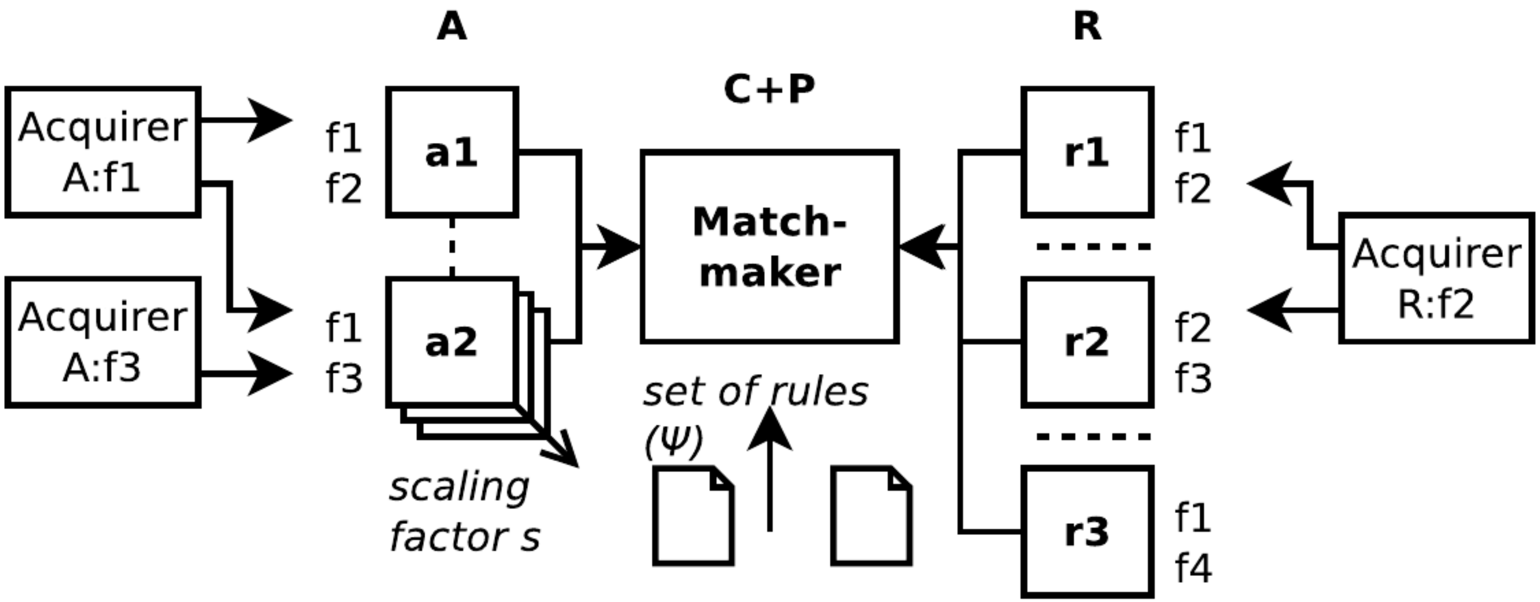
R Knowledge

CloudPick
Scraping
(e.g. García-Galán)
FaaSCC
CMPs (esp. multi-cloud)
CIM/MIB/SNMP/...

[https://github.com/serviceprototypinglab/faascc]

RBMM Approach

A: Artefacts, R: Resources, C+P: Constraints and Preferences
F: Decision Factors



→ **Deployment:** $A \mapsto_{C+P} R_{used} \subseteq R$

RBMM Rules (Ψ)

Propagation rules (Ψπ)

- complement missing factors, change existing factors
- A factors derived from single artefacts
- replication, subsumption, bounding (latency), tainting (security), ...

Skipping rules (Ψσ)

- temporarily hide factors
- context (e.g. leave CPU/memory to later stage)
- feasibility (e.g. pre-check based on trust, country/geolocation)

Aggregation rules (Ψα)

- adjust post-deployment resource characteristics
- reduction of limited resources: memory, port numbers, GPU, ...

RBMM Implementations

Matchmaker:

RBMM - Python library

- **~3.3s for |A|=|R|=10000 combinatorial, ~80.1s for |A|=|R|=200 recursive**
- **extensible for other algorithms**

[https://github.com/serviceprototypinglab/rbmm]

Emulator integration:

- **OsmoticToolkit**
- **based on MaestroNG, Docker**
- **decision factors specified in YAML**


```
ship_provider : dynamic # static
name : p1
# ships :
    # ship1 :
        #  ip : x.x.x.x
services :
    foo :
        image : ubuntu
        security_opt : [zone==intranet,
            vulnerability==backdoor, consistency
            == true]
        requires: [test]
        labels :
            constraint :
                runtime : python3
                complexity : high
                latency : 5ms
                duration : 900s
        limits :
            memory : 50m
            cpu : 1
        instances :
            foo-1 :
                # ship : ship1
```