Algorithmic Methods for Mathematical Models (AMMM)

Some Applications

Luis Velasco

(lvelasco @ ac.upc.edu)

Campus Nord D6-107

Algorithmic Methods for Mathematical Models (AMMM)

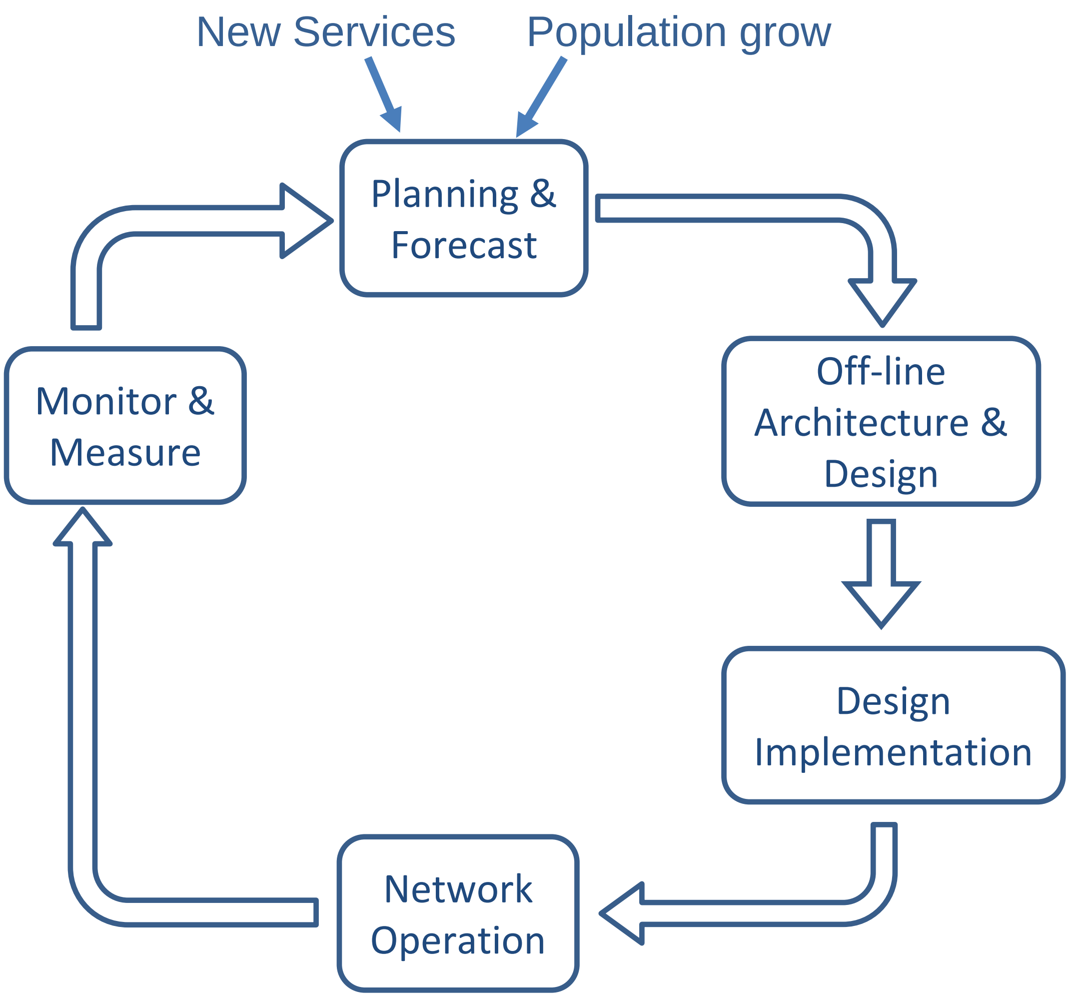
Network Planning

Luis Velasco

(lvelasco @ ac.upc.edu)

Campus Nord D6-107

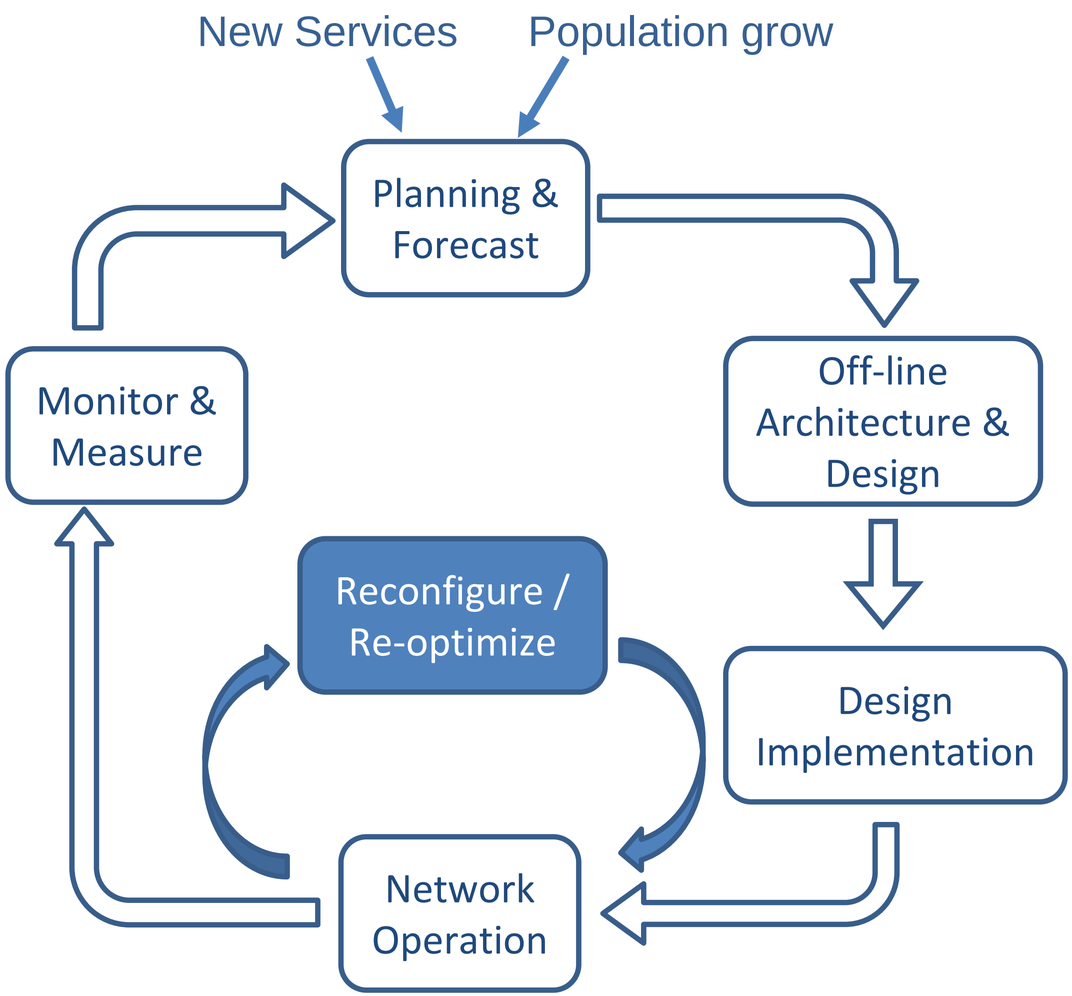
Classical Network Life-Cycle



Off-line Network Planning

- Long planning cycles, e.g. yearly.
- Reducing total cost of ownership.
- Can take time to explore different scenarios.

Augmented Network Life-Cycle



<u>Off-line Network Planning</u>

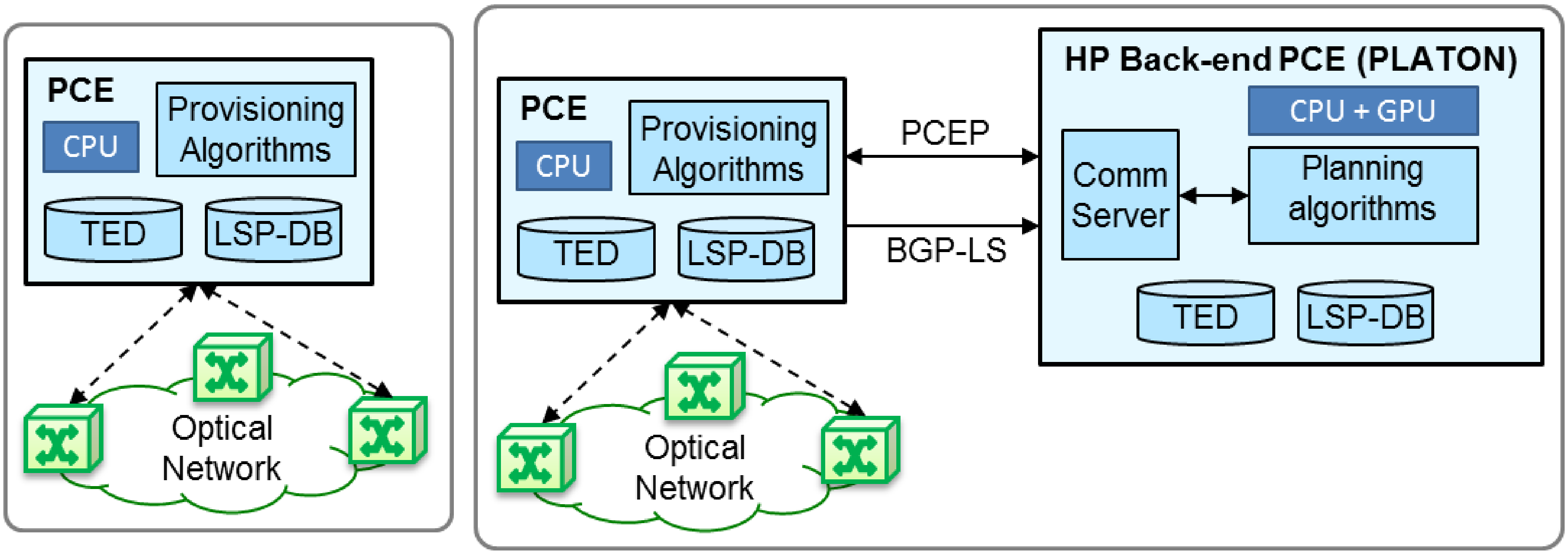
- Long planning cycles, e.g. yearly.
- Reducing total cost of ownership.
- Can take time to explore different scenarios.

<u>In-operation Network Planning</u>

- Re-optimizing the network while it is in operation.
- Good-enough solutions in stringent computation times, e.g. ms.

L. Velasco *et al.*, “In-Operation Network Planning,” IEEE Communications Magazine, vol. 52, pp. 52-60, 2014.

Centralized PCE vs. fPCE/bPCE Architecture



Ll. Gifre *et al.*, "Experimental Assessment of a High Performance Back-end PCE for Flexgrid Optical Network Re-optimization," in Proc. IEEE/OSA Optical Fiber Communication Conference (OFC), 2014.

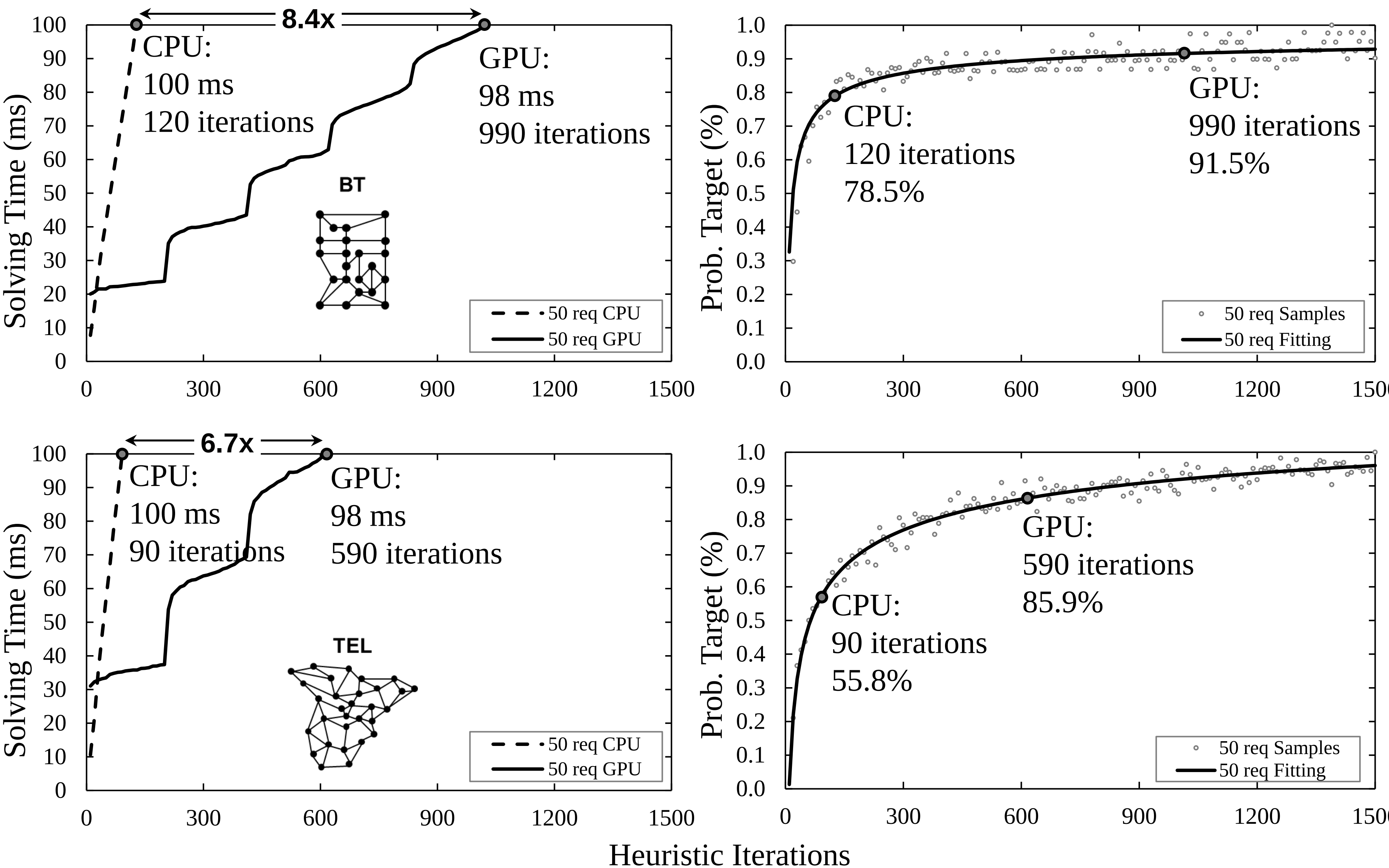
Restoration Problem Statement

- Given:
  - An optical network topology, represented by a graph G(*N*, *E*)
    - *N:* set of nodes
    - *E:* set of fiber links each one connecting two nodes
  - the spectrum characteristics of each link, including the set *S* of slices
  - a set *D* of demands to be routed and allocated on the network; each demand *d* identified by the tuple $<s_d, t_d, n_d>$
    - s_d and t_d: source and destination nodes
    - n_d: amount of slices required for the bitrate requested.
- Find:
  - The route and spectrum allocation for each demand *d* in the set.
- Objective:
  - Minimize the amount of spectrum resources used and blocked demands.

Heuristic Algorithm

```
INPUT G(N, E), D, maxIter
OUTPUT bestS
bestS ← Ø
for i = 1..maxIter do
  S ← Ø; G' ← G
  sort(D, random)
  for each d ∈ D do
    {<k, c_k>} ← getRSA(G', d.src, d.tgt, d.bw)
    if k = Ø then continue
    allocate(G', d, <k, c_k>)
    S ← S ∪ {d}
  Compute Φ(S)
  if Φ(S) > Φ(bestS) then
    bestS ← S
return bestS
```

Performance Evaluation

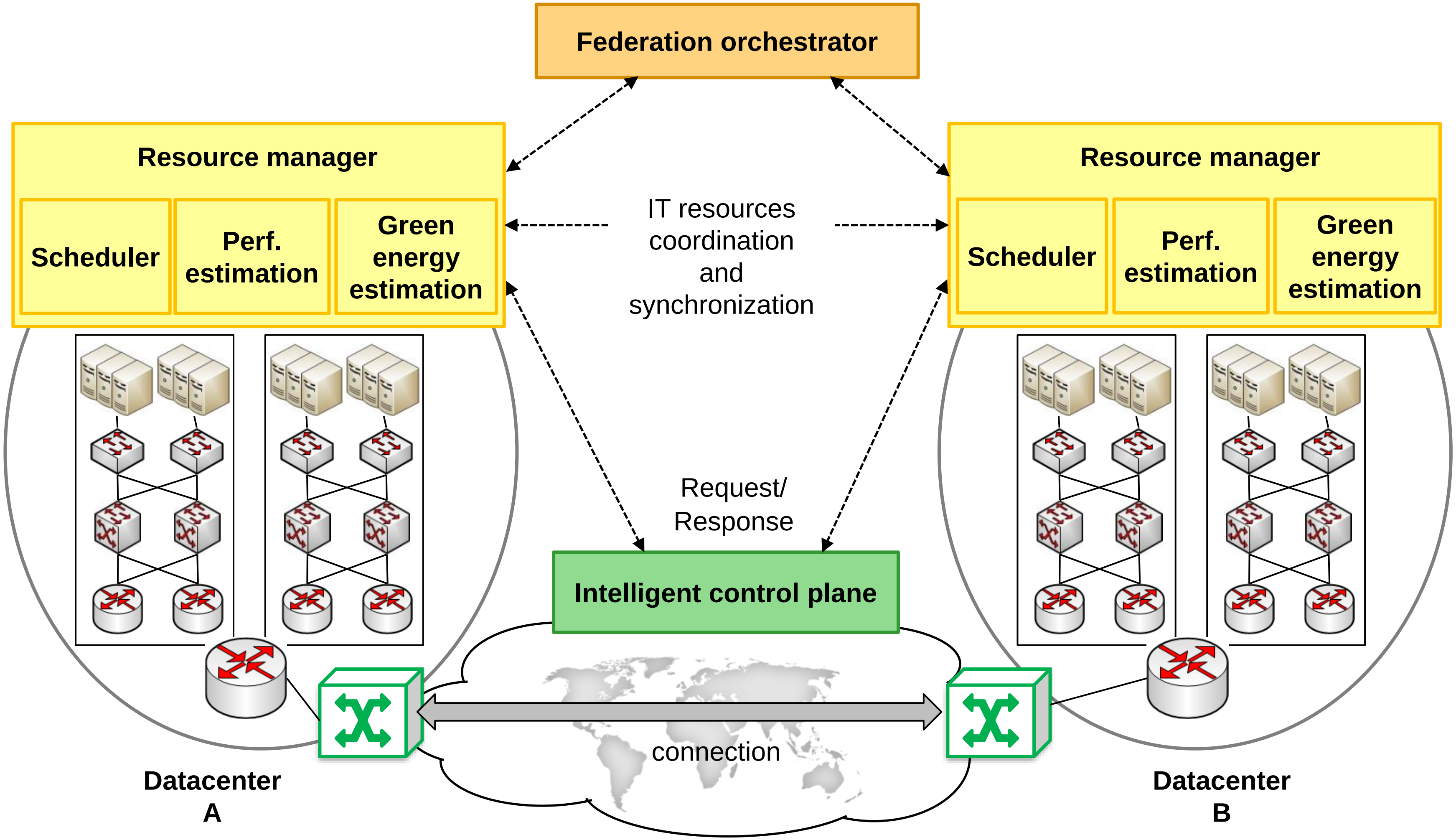


Algorithmic Methods for Mathematical Models (AMMM)

DataCenter Federations

Luis Velasco
(lvelasco @ ac.upc.edu)
Campus Nord D6-107

Orchestrating DC federations



L. Velasco *et al.*, "Elastic Operations in Federated Datacenters for Performance and Cost Optimization," Elsevier Computer Communications, vol. 50, pp. 142-151, 2014.

Elastic Operations in Federated DCs for Performance and Cost optimization (ELFADO)

Given:

- set of federated DCs *D*;
- set of optical connections *E* that can be established between two DCs;
- set of VMs $V(d)$ in each DC *d*;
- set of client locations *L*, and their number of users to be served in the next period;
- brown energy cost and the available green energy g_d in each DC in the next period;
- characteristics of each VM (data volume to migrate k_v, cores needed…), servers (cores, energy consumption…);
- performance p_{ld} perceived in location *l* when served from a VM placed in DC *d*;
- threshold th_v for the performance required at any time for accessing the service in VM *v*.

Output:

- DCs where each VM will be placed.

Objective:

- minimize energy **and communications** cost while ensuring the performance objective for each service.

Algorithms

Distributed

INPUT: d_1, $V(d_1)$, D | **OUTPUT**: *Sol*

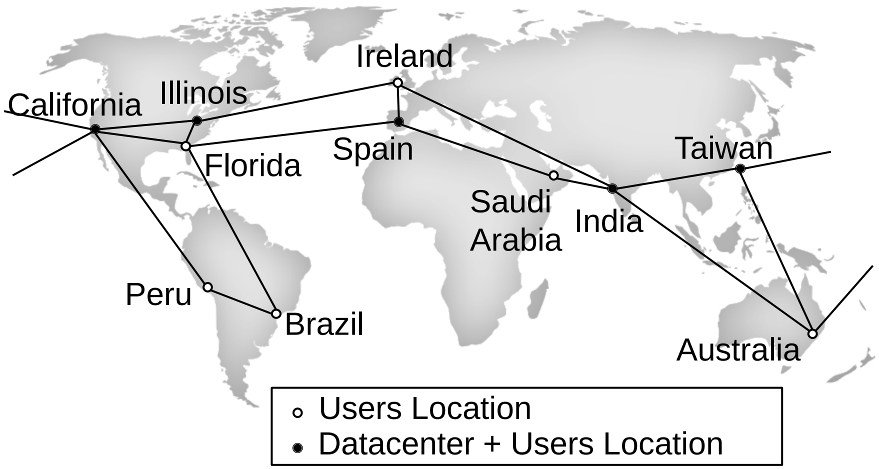
Initialize
for each $v \in V(d_1)$ **do**
  for each $d \in D$ **do**
    if $p_{vd} \leq th_v$ **then**
      if $d \neq d_1$ **then**
        let $e = (d_1, d)$
        $C[v]$.list ← costWithComm()
      else
        $C[v]$.list ← costWithoutComm()
  sort($C[v]$.list, Ascending)
for each $v \in V(d_1)$ **do**
  for i = 1..$C[v]$.list.len **do**
    $\{d, e\}$ ← $C[v]$.list(i)
    if $e \neq \emptyset$ && $z_e + k_v > k_e$ **then**
      continue
    if $e \neq \emptyset$ **then** $z_e \leftarrow z_e + k_v$
    $Sol \leftarrow Sol \cup \{(v, d)\}$
    break
return *Sol*

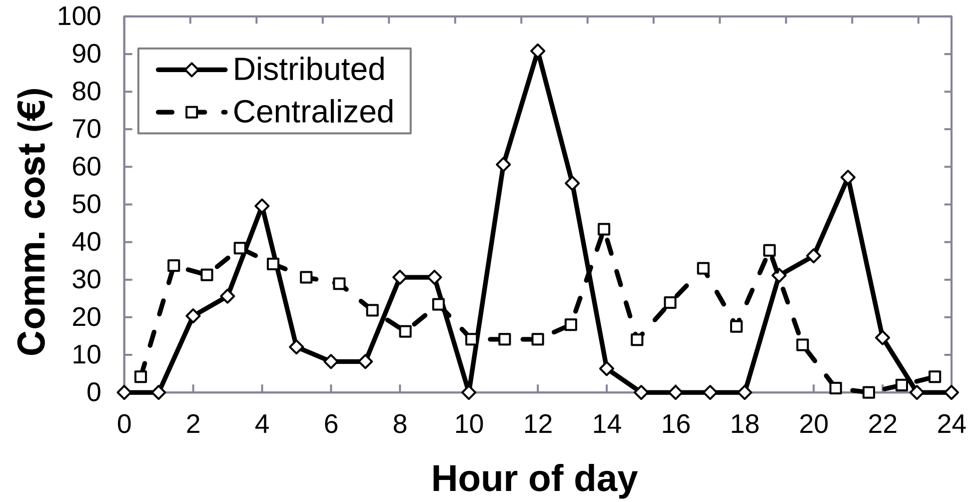
Centralized

INPUT: V, D | **OUTPUT**: *Sol*

Initialize
for each $d \in D$ **do**
  U_d ← unfeasPerformanceInCurrentPlacement(d, $V(d)$);
  F_d ← feasiblePerformanceInCurrentPlacement (d, $V(d)$);
  $r_d \leftarrow g_d$ - computeEnergyConsumption(F_d)
  $U \leftarrow U \cup U_d$; $F \leftarrow F \cup F_d$
  updateRemainingGreen(r_d, R)
if $R \neq \emptyset$ **then**
  for each $(v, d_1) \in U$ **do**
    if find $(d_2, r_{d2}) \in R$ feasible for v such that $r_d > w_v$ with min comm. cost **then**
      update(U)
      $Sol \leftarrow Sol \cup \{(v, d_2)\}$
      updateRemainingGreen(r_{d2}, R)
if $R \neq \emptyset$ **then**
  for each $(v, d_1) \in F$ **do**
    if find $(d_2, r_{d2}) \in R$ feasible for v such that $r_d > w_v$ with min comm. cost **then**
      update(F)
      $Sol \leftarrow Sol \cup \{(v, d_2)\}$
      updateRemainingGreen(r_{d2}, R)
$F \leftarrow F \cup U$
for each $\{v, d_1\} \in F$ **do**
  distributedOnlyBrownSources(d_1,v,D)
return *Sol*

Illustrative numerical results



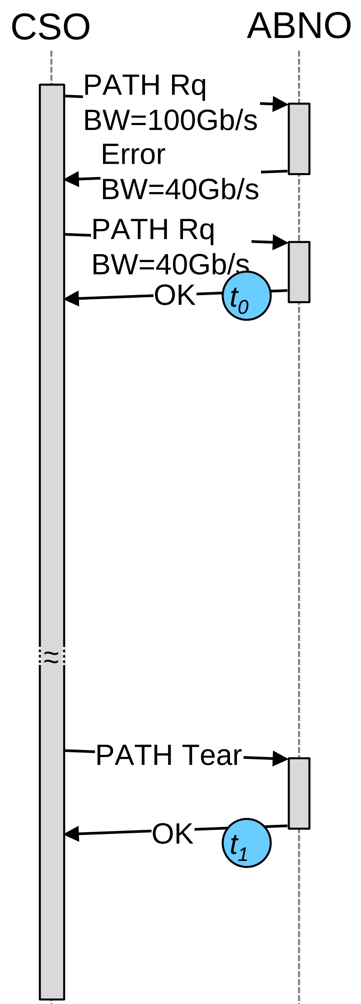


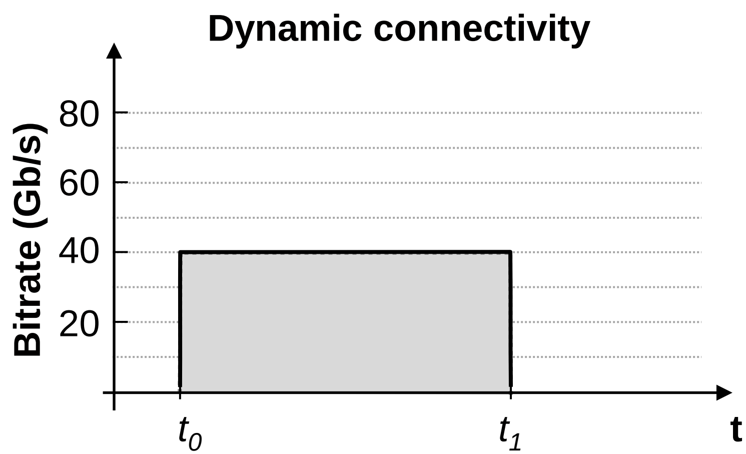
	c_d (on/off peak) (€/kWh)
Taiwan	0.0700 / 0.0490
India	0.0774 / 0.0542
Spain	0.1042 / 0.0729
Illinois	0.0735 / 0.0515
California	0.0988 / 0.0692

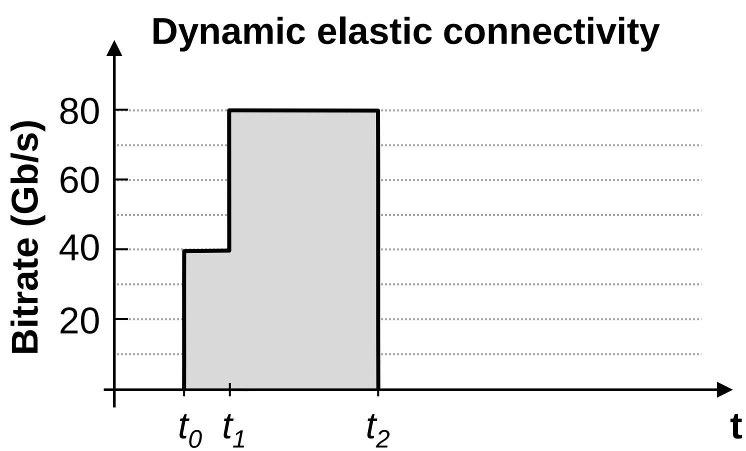
	Energy cost	Comm. cost	Total cost
Fixed	6,048 €	--	6,048 €
Dist.	5,374 € (11.1%)	537 €	5,912 € (2.3%)
Cent.	2,867 € (52.6%)	508 € (5.8%)	3,376 € (44.2%)

Dynamic connectivity for DC interconnection

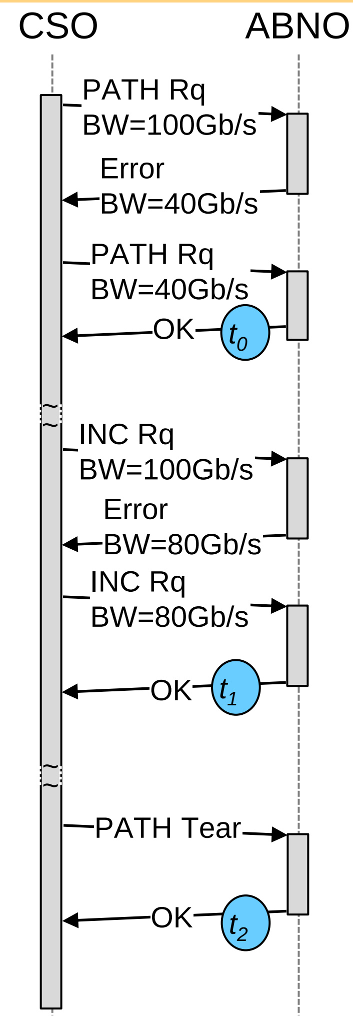
Dynamic connectivity







Dynamic elastic connectivity



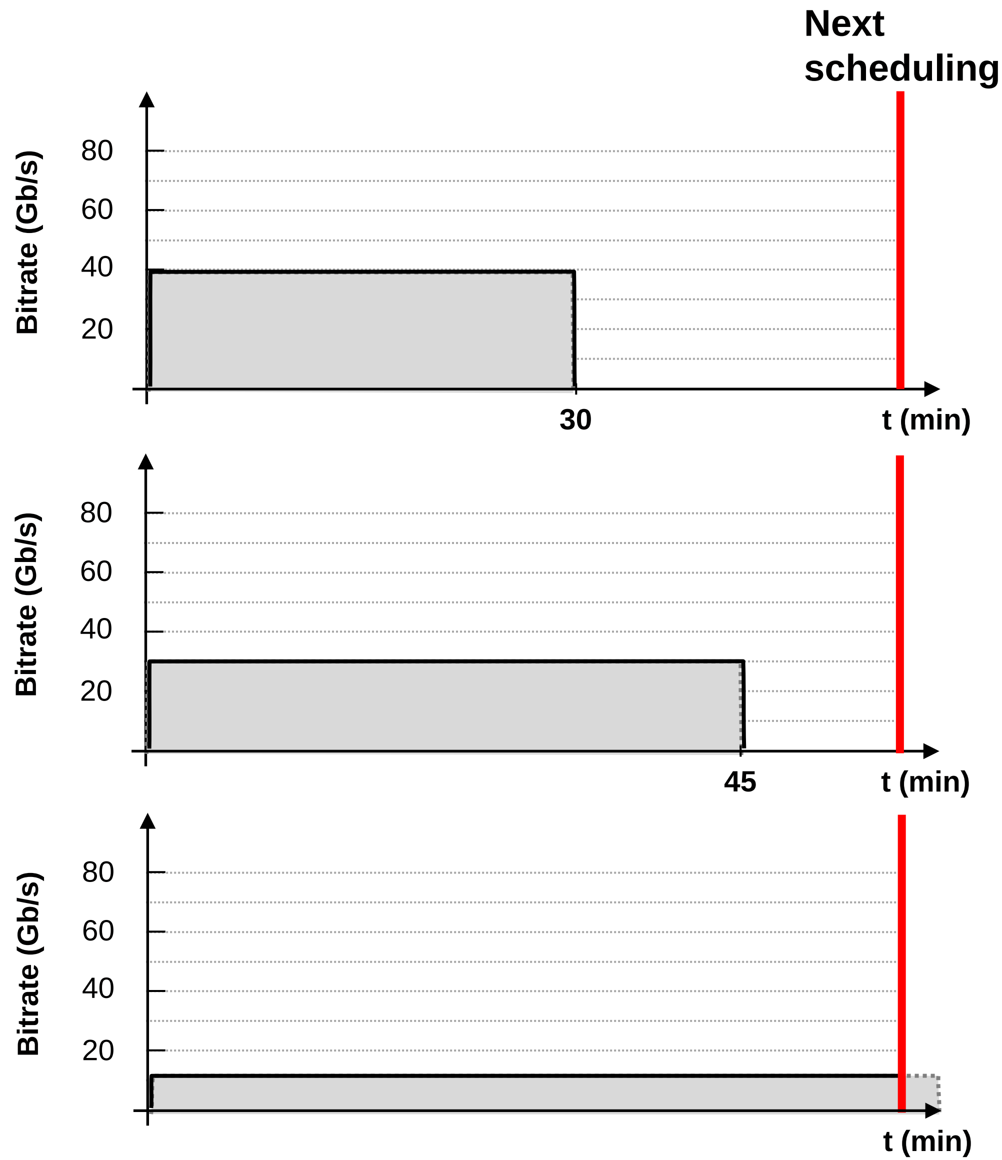
L. Velasco *et al.*, "Cross-Stratum Orchestration and Flexgrid Optical Networks for Datacenter Federations," IEEE Network Magazine, vol. 27, pp. 23-30, 2013.

Impact of the lack of resources at requesting time

- 1600 VMs migrated in 30 min.

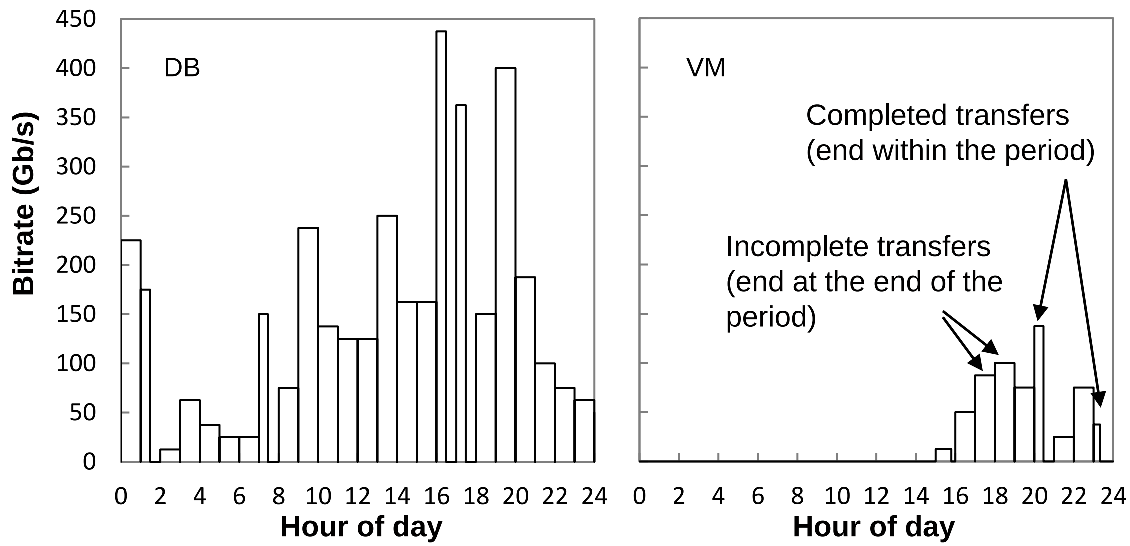
- 1600 VMs migrated in 45 min.

- 800 VMs migrated in 1 hour.
- Migration stopped due to overlapping with next scheduling period

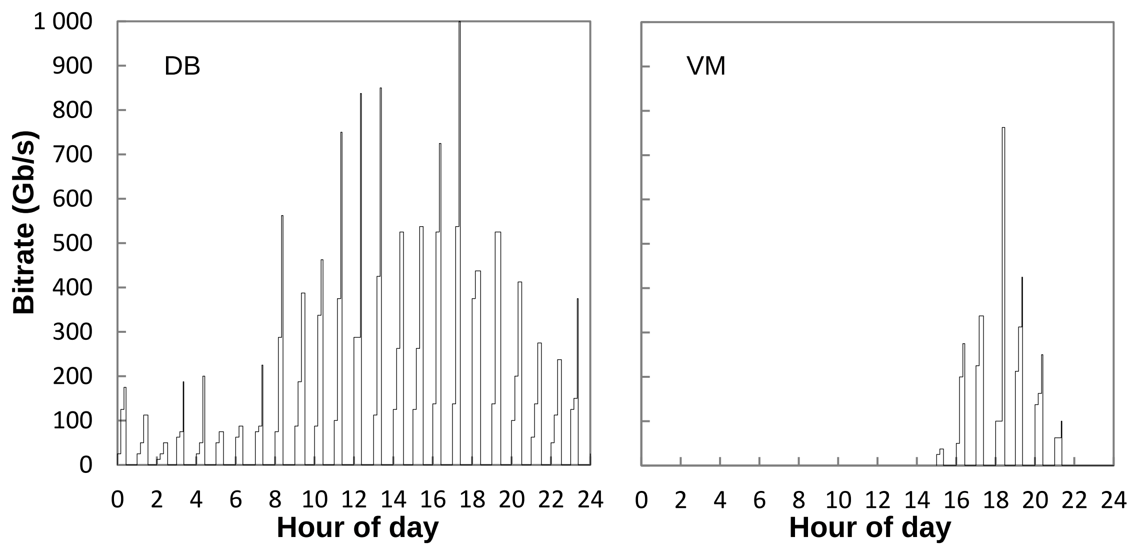


Dynamic vs. dynamic elastic

Dynamic



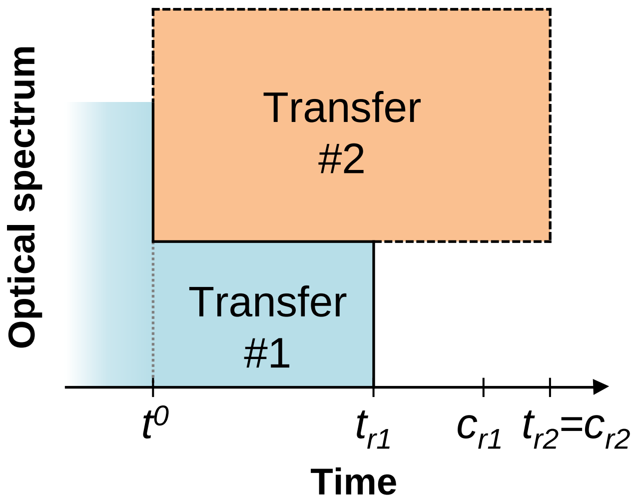
Dynamic elastic

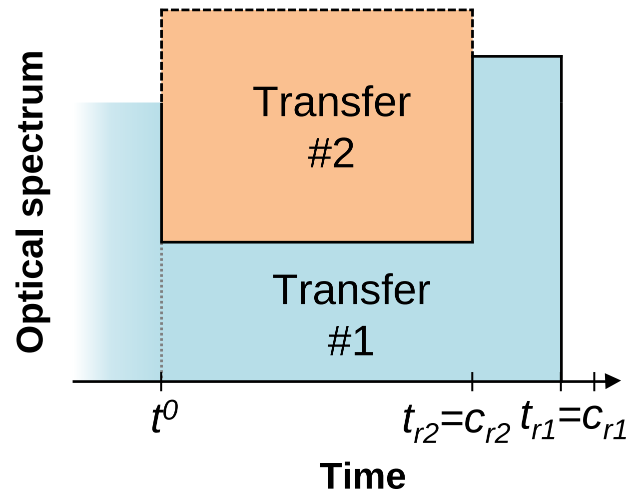


Performance evaluation

		Time-to-transfer (minutes)		
		Static	**Dynamic**	**Elastic**
Time-to-transfer (minutes)	**DB synchronization (Max. / Avg.)**	56.0 / 28.5	58.0 / 39.9	49.0 / 24.7
	VM migration (Max. / Avg.)	50.0 / 28.7	48.0 / 39.9	40.0 / 24.4
Bitrate savings		-	59.1%	57.9%

Managing transfer-based connections





A. Asensio and L. Velasco, “Managing Transfer-based Datacenter Connections,” IEEE/OSA Journal of Optical Communications and Networking, vol. 6, pp. 660-669, 2014.

Routing and Scheduled Spectrum Allocation (RSSA)

Given:

- $G(L, E)$, being L the set of locations and E the set of fiber links connecting two locations;
- set R with the transferences currently in progress in the network. For each transference $r \in R$, the tuple $<o_r, d_r, v_r, c_r, r_r, s^0_r, s^1_r, t^1_r, t_r>$ specifies the origin (o_r) and destination (d_r) DCs, the remaining amount of data (v_r) to be transferred, the requested completion time (c_r), the route (r_r), and slot currently allocated (s^0_r), the scheduled slot allocation (s^1_r) to be performed at time t^1_r, and the scheduled completion time (t_r);
- new transfer request, described by the tuple $<o_r, d_r, v_r, c_r>$.

Output:

- route (r_r), slot allocation (s^0_r), and scheduled completion time (t_r) for the new transference request;
- new spectrum allocation (w^0_r), scheduled reallocation (w^1_r, t^1_r), and completion time (t_r) for each transference request to be re-scheduled.

Objective:

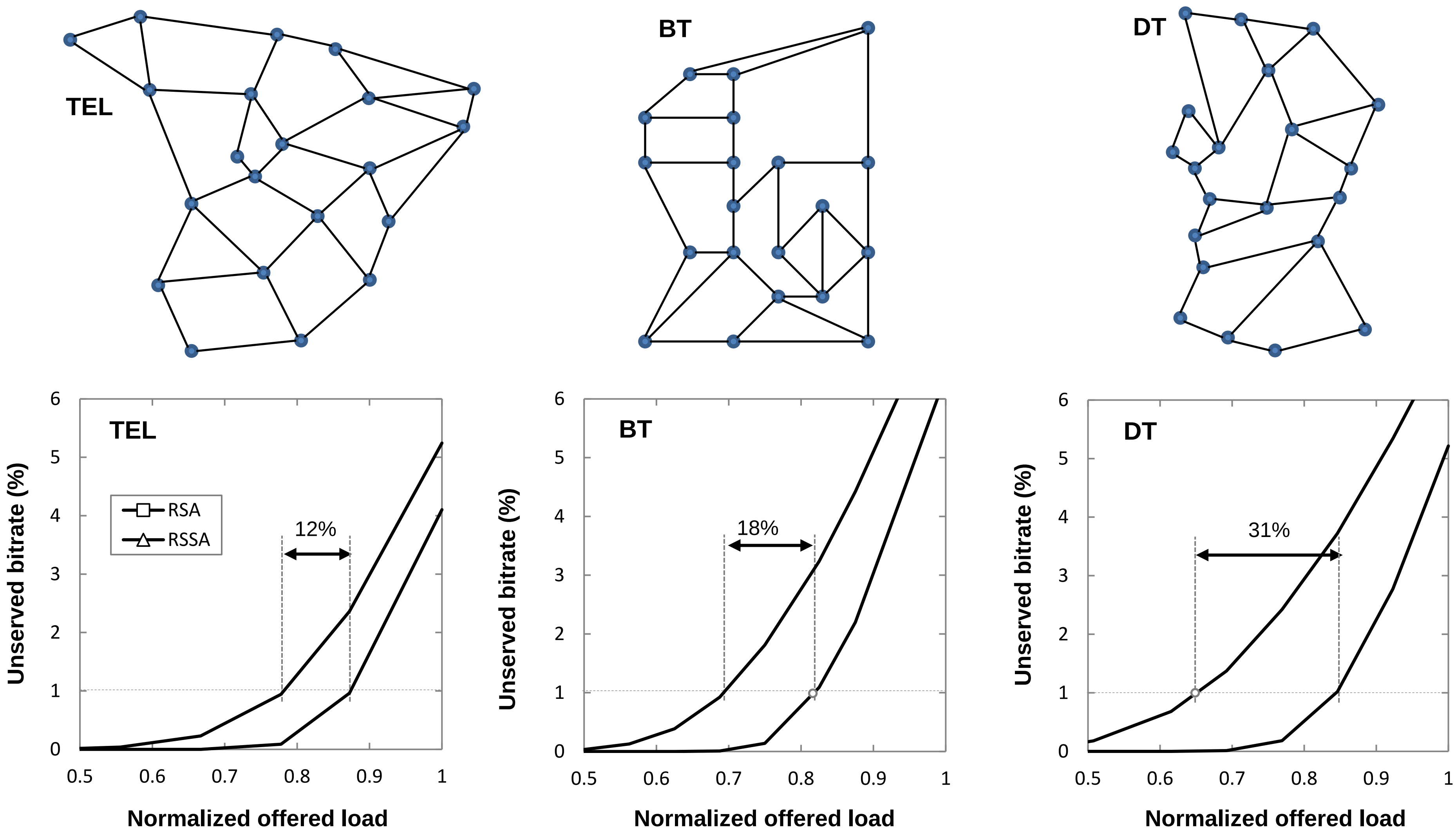
- minimize the number of connections to be re-scheduled to make room for the incoming request.

Algorithm

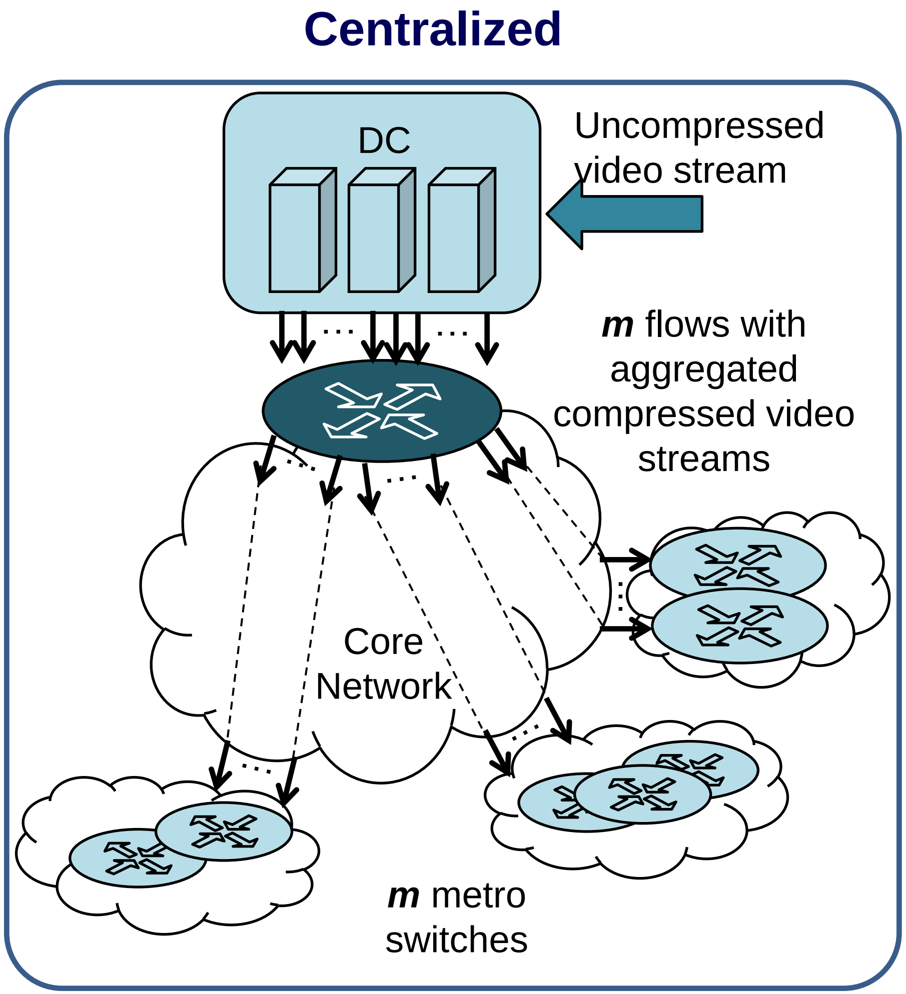
INPUT: *o*, *d*, *v*, *c*
OUTPUT: *route*, *slot*

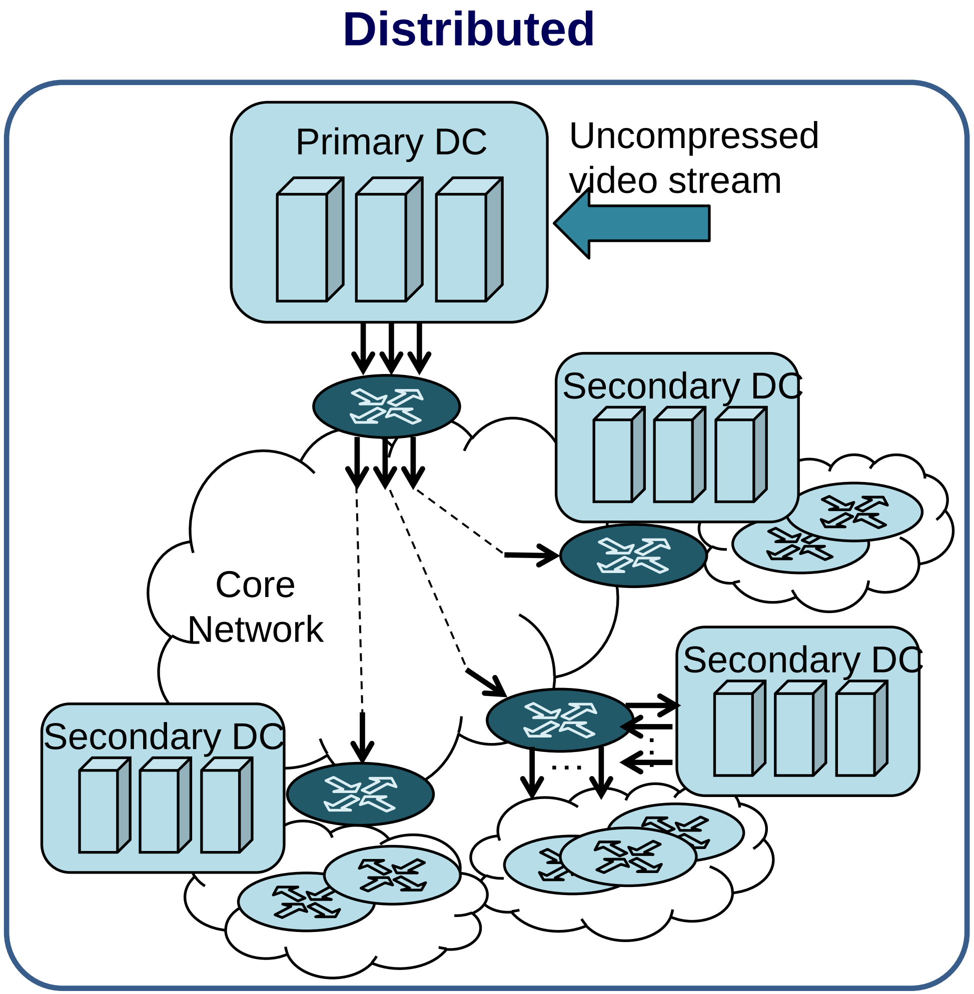
```
1:  br ← getMinBitrate(v, c)
2:  {route, slot} ← computeRSA(o, d, br, sTED)
3:  if route ≠ Ø then
4:    slot ← getMaxSA(route, sTED)
5:    return {route, slot}
6:  P ← computeKSP(o, d)
7:  for each p ∈ P do
8:    sw ← getSlotWidth(p.length, br)
9:    S ← getFreeSlots(p, sTED)
10:   sort(S, slot width, DESC)
11:   for each s ∈ S do
12:     Q ← getAdjacentPaths(p, s, sTED)
13:     R ← findSlot(Q, s, sw, sTED)
14:     if R ≠ Ø then
15:       doReschedule (R)
16:       slot ← getSA(p, br, sTED)
17:       return {p, slot}
18: for each p ∈ P do
19:   sw ← getSlotWidth(p.length, br)
20:   S ← getFeasibleBorderPathSlots(p, sTED)
21:   sort(S, slot width, DESC)
22:   repeat from line 11 to line 17
23: return {Ø, 0}
```

Illustrative numerical results



Architectures for live-TV distribution and processing





A. Asensio *et al.*, "Scalability of Telecom cloud Architectures for Live-TV Distribution," in Proc. IEEE/OSA Optical Fiber Communication Conference (OFC), 2015.

Problem statement

Given:

- primary location *p;*
- set *M* of metro locations each containing a set of metro switches;
- subset $D \subseteq M$ of locations that can host a secondary DC;
- topology *G* of the optical network interconnecting locations in *D*;
- cost structure for core switches, FTs, and SBVTs;
- uncompressed video stream to be distributed from *p* to every secondary DC (*n*) and a set of aggregated video streams to be distributed from every secondary DC to the corresponding metro switches.

Output:

- set $D^* \subseteq D$ of locations, where secondary DCs are placed;
- configuration of every core switch in terms of capacity, number of card slots, and number and type of transponders to be installed;
- set of optical connections to be set up in the optical network.

Objective:

- minimize CAPEX when *n* secondary DCs are equipped.

Planning procedure

INPUT: *M*, *D’*, *G*, *cost structure*
OUTPUT: *cost*

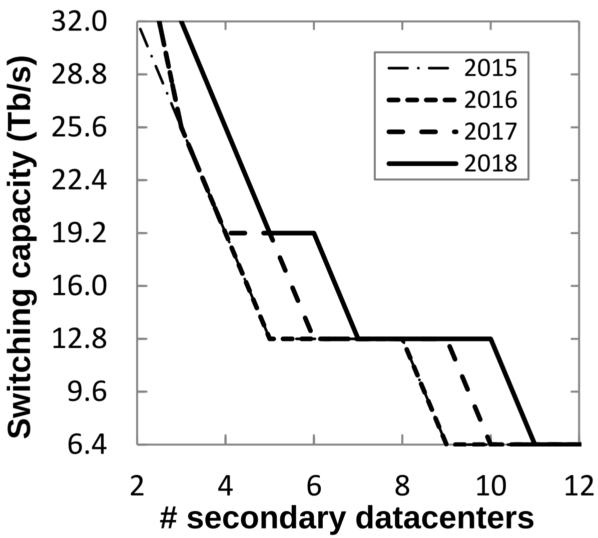
1: **for each** *d* ∊ *D’* **do**
2: Find a connection from *p* to *d* on *G*
3: **for each** *m* ∊ *M* **do**
4: **if** *m* ∊ *D’* **then** assign all metro switches in *m* to the local secondary DC
5: **else** Find a connection from metro switches in *m* to the nearest *d* ∊ *D’* on *G*
6: **for each** *d* ∊ *D’* **do**
7: Compute # FTs and SBVTs
8: Find the cheapest feasible switch
9: **return** Compute network CAPEX using *cost structure*

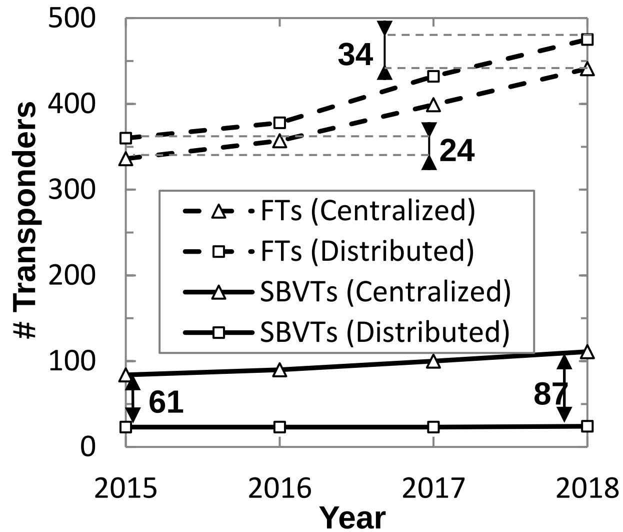
Illustrative numerical results

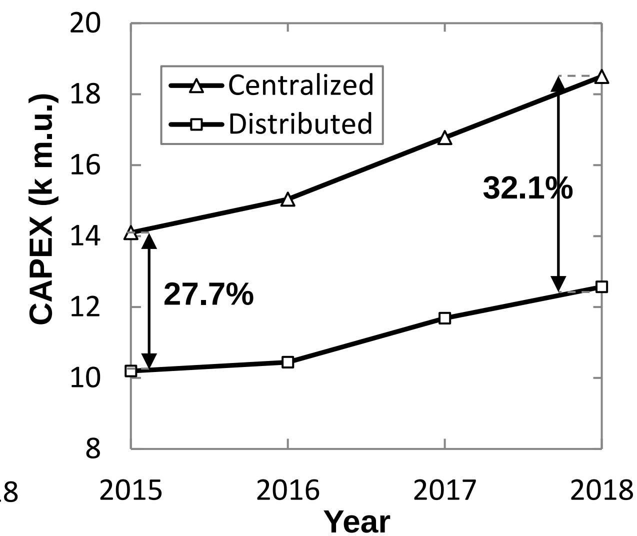
- Telefonica's Spanish national network
  - 5 regional 30-node optical networks
  - 1 core 21-node optical network
- SD, HD and UHD video quality adoption
  - 7-8 millions of users
  - From 2015 to 2018
- 8 TV channels
- 100 Gb/s connections

Centralized

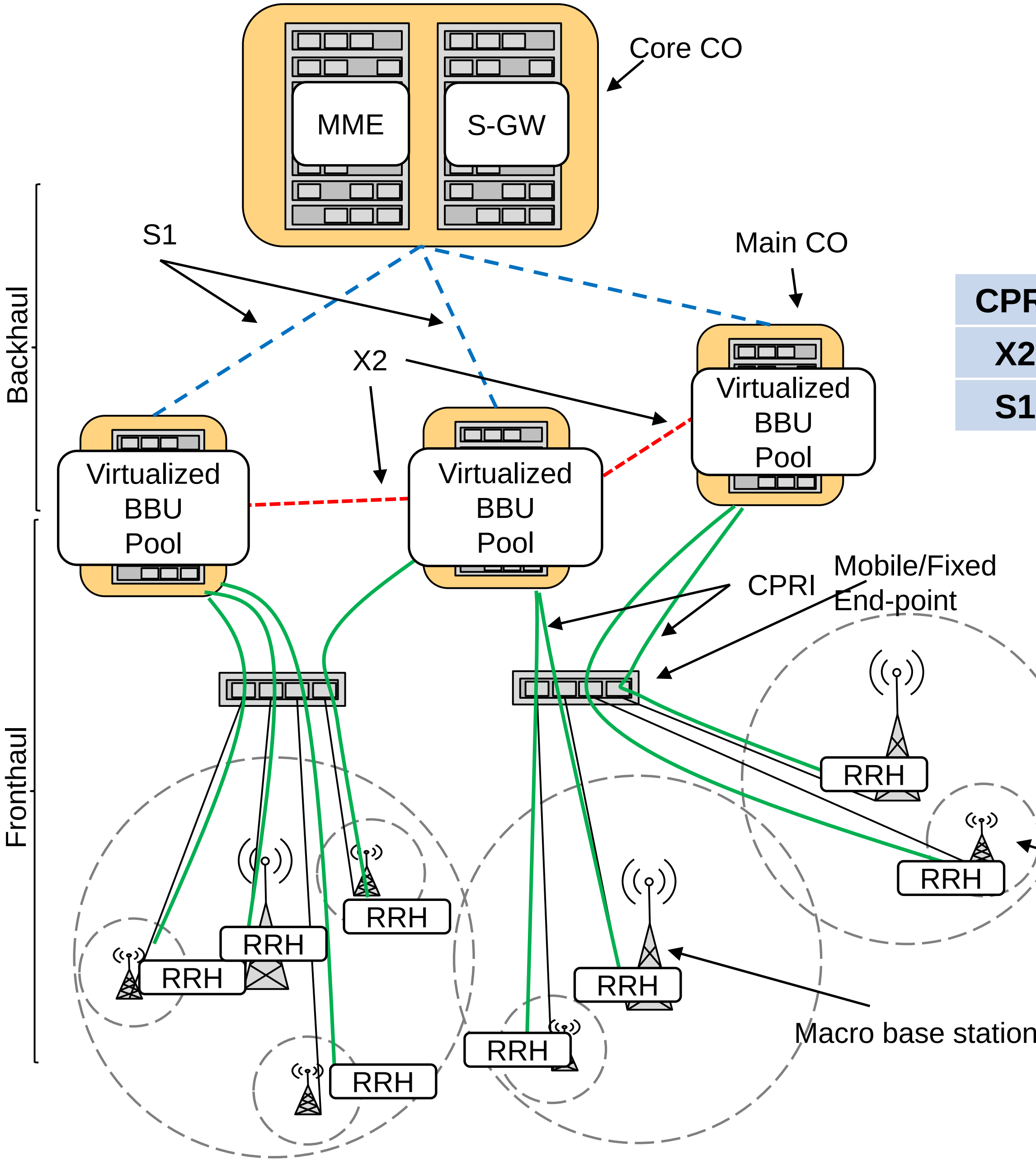
Parameter	2015	2016	2017	2018
Capacity (Tb/s)	70.4	76.8	83.2	89.6
# SBVTs	84	90	100	111
# FTs	336	357	399	441







C-RAN architecture



	LTE-A 4x4 MIMO		
	40 MHz	**100 MHz**	**Distance**
CPRI	10 Gb/s	25 Gb/s	< 20 km
X2	200 Mb/s	500 Mb/s	< 100 km
S1	600 Mb/s	1.5 Gb/s	~ 1000's km

A. Asensio *et al.*, "Study of the Centralization Level of Optical Network-Supported Cloud RAN," accepted in IEEE International Conference on Optical Network Design and Modeling (ONDM), 2016.

The C-RAN CAPEX Minimization (CRAM) problem

Given:

- set of geographically distributed RRHs H;
- tuple $<\alpha_h, \beta_h, \gamma_h>$ representing the required capacity by RRH h for CPRI, S1, and X2 interfaces, respectively, in the case it is active;
- set V of VMs' configurations with capabilities for BBU pools virtualization;
- set of transponders P; each transponder p is defined by its cost κ_p and capacity φ_p;
- set of line cards C; each line card c can support one type of transponder and it is defined by its cost κ_c, and the number of ports to plug-in transponders ξ_{cp};
- set of IP/MPLS routing equipment E; each router e is defined by its cost κ_e, its switching capacity σ_e, and the number of available slots ρ_e to plug-in line cards;
- set O of main COs that can be equipped with a predefined configuration of VMs and with an IP/MPLS router;
- core CO with functions for MME, S-GW.

Output:

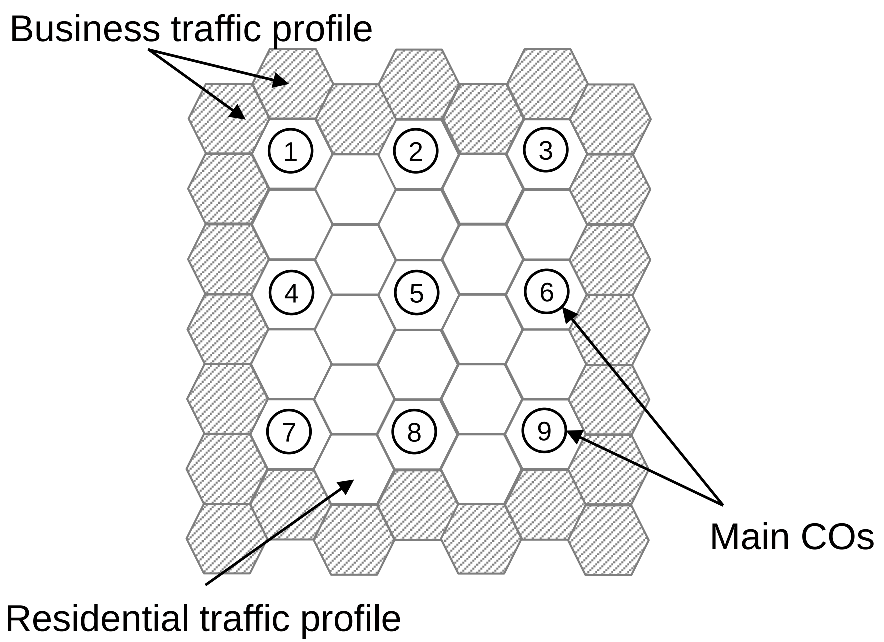
- equipment to install in each main CO.

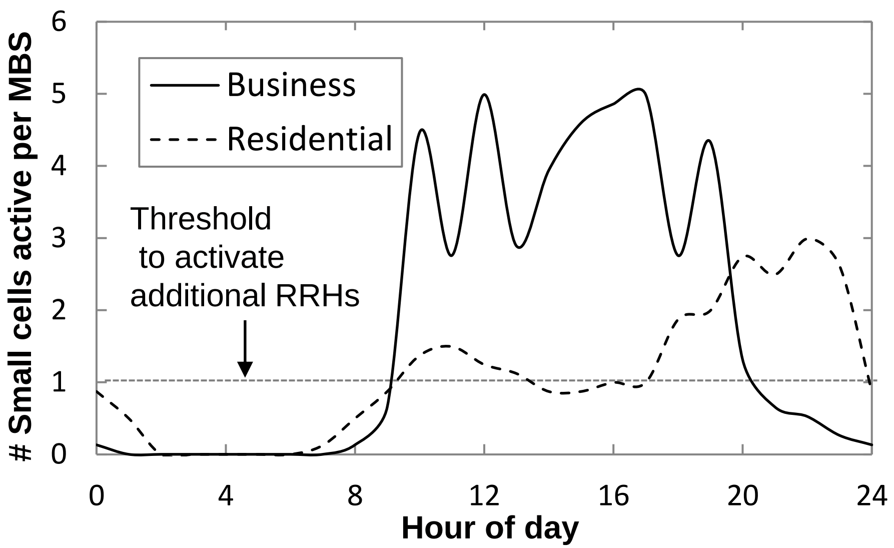
Objective:

- minimize the cost of the equipment used.

Scenario

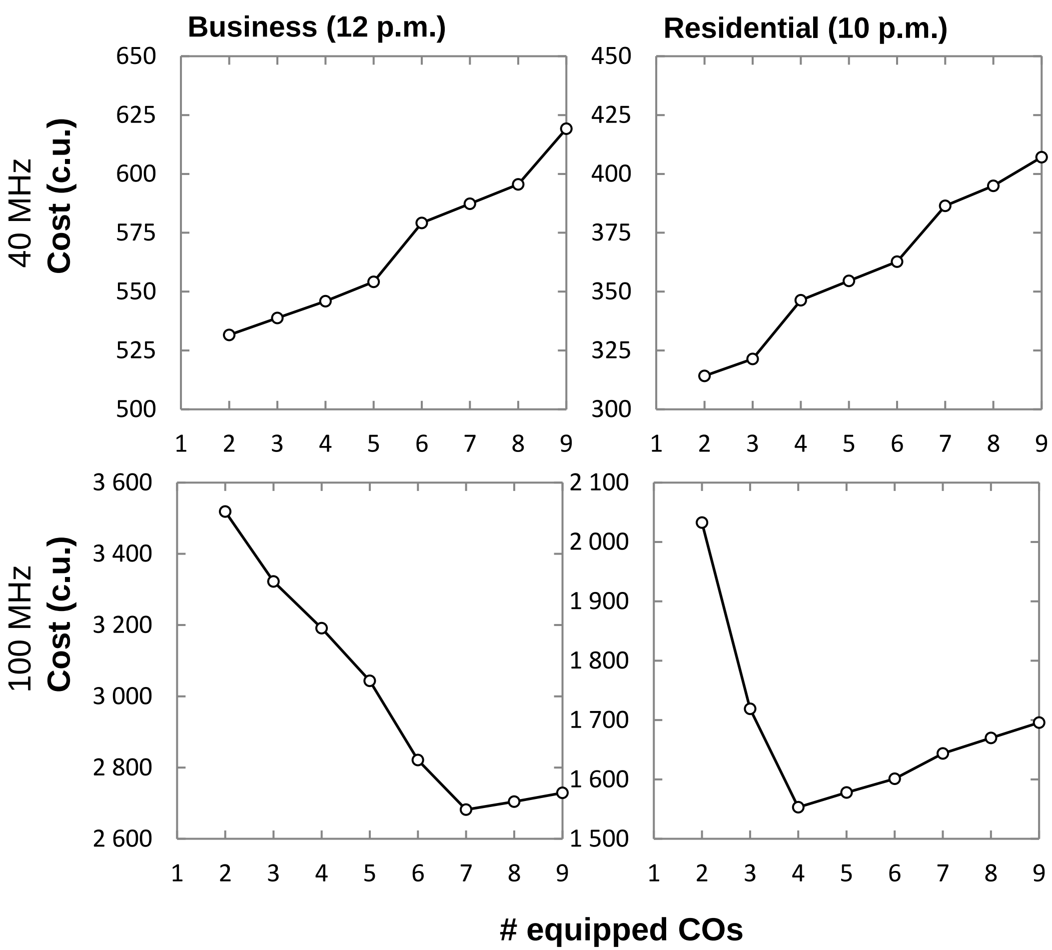
- 49 MBS + small cells that can be activated or deactivated
- ~200 RRHs (Max. distance to CO limited to 15 Km)
- Traffic profiles: Business and residential
- LTE-A configurations: 4x4 MIMO 40 MHz and 100 MHz
- Network equipment cost model: F. Rambach *et al*.





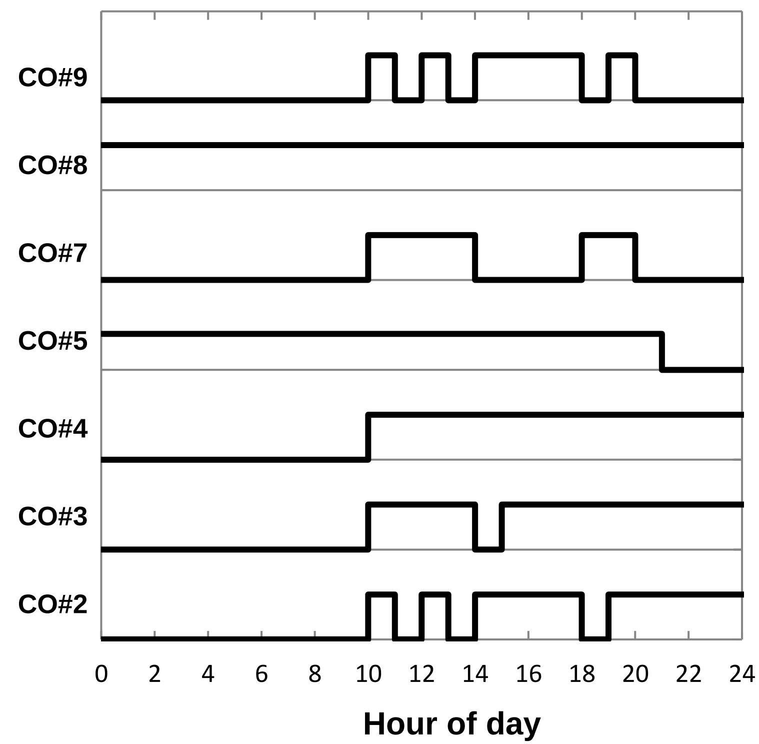
Illustrative numerical results for LTE-A 4x4 MIMO

Cost @ peak hours



CAPEX

- LTE-A 4x4 MIMO 100 MHz
- Minimum CAPEX solution was obtained by **equipping 7 COs**



Algorithmic Methods for Mathematical Models (AMMM)

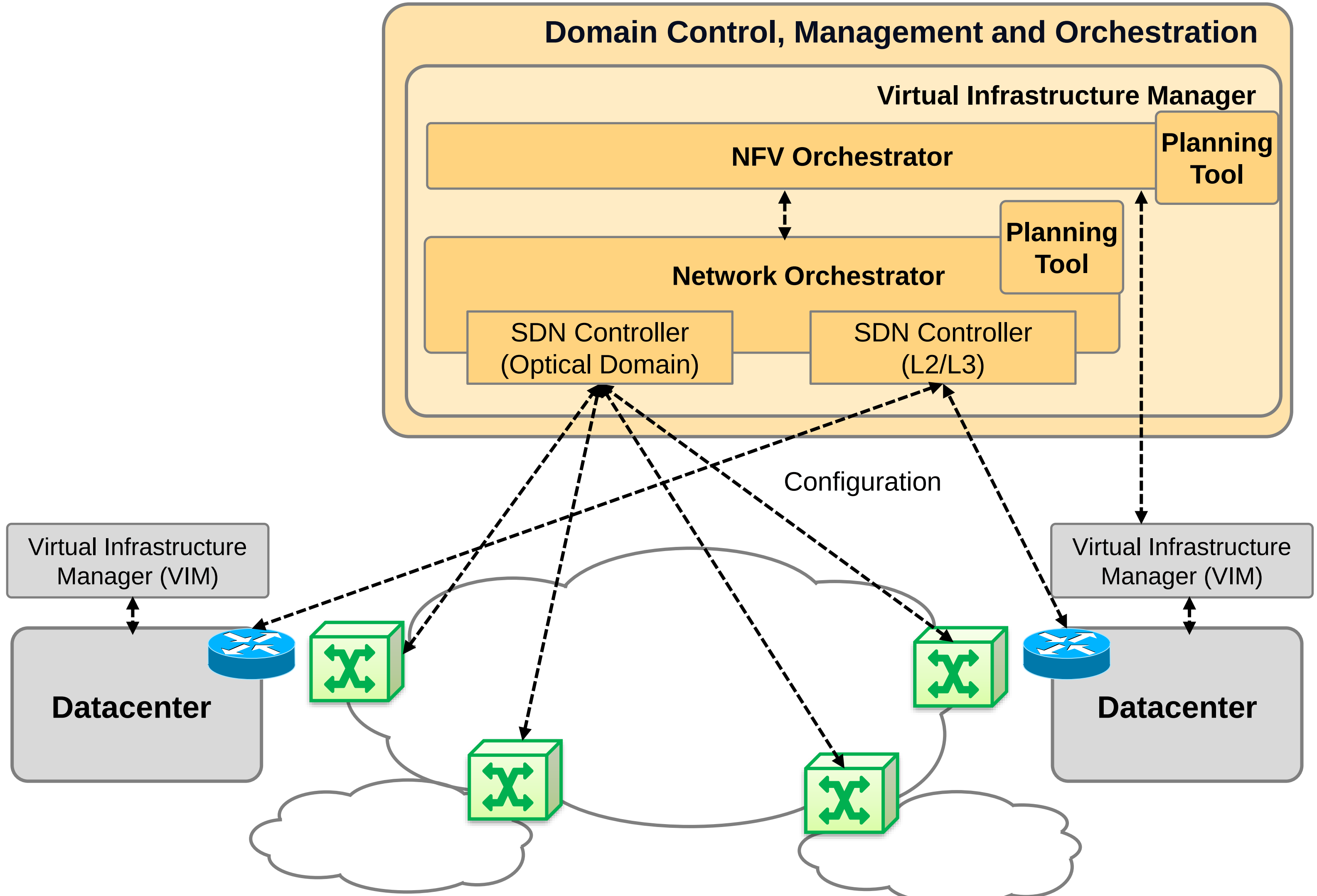
Autonomic Infrastructures

Luis Velasco

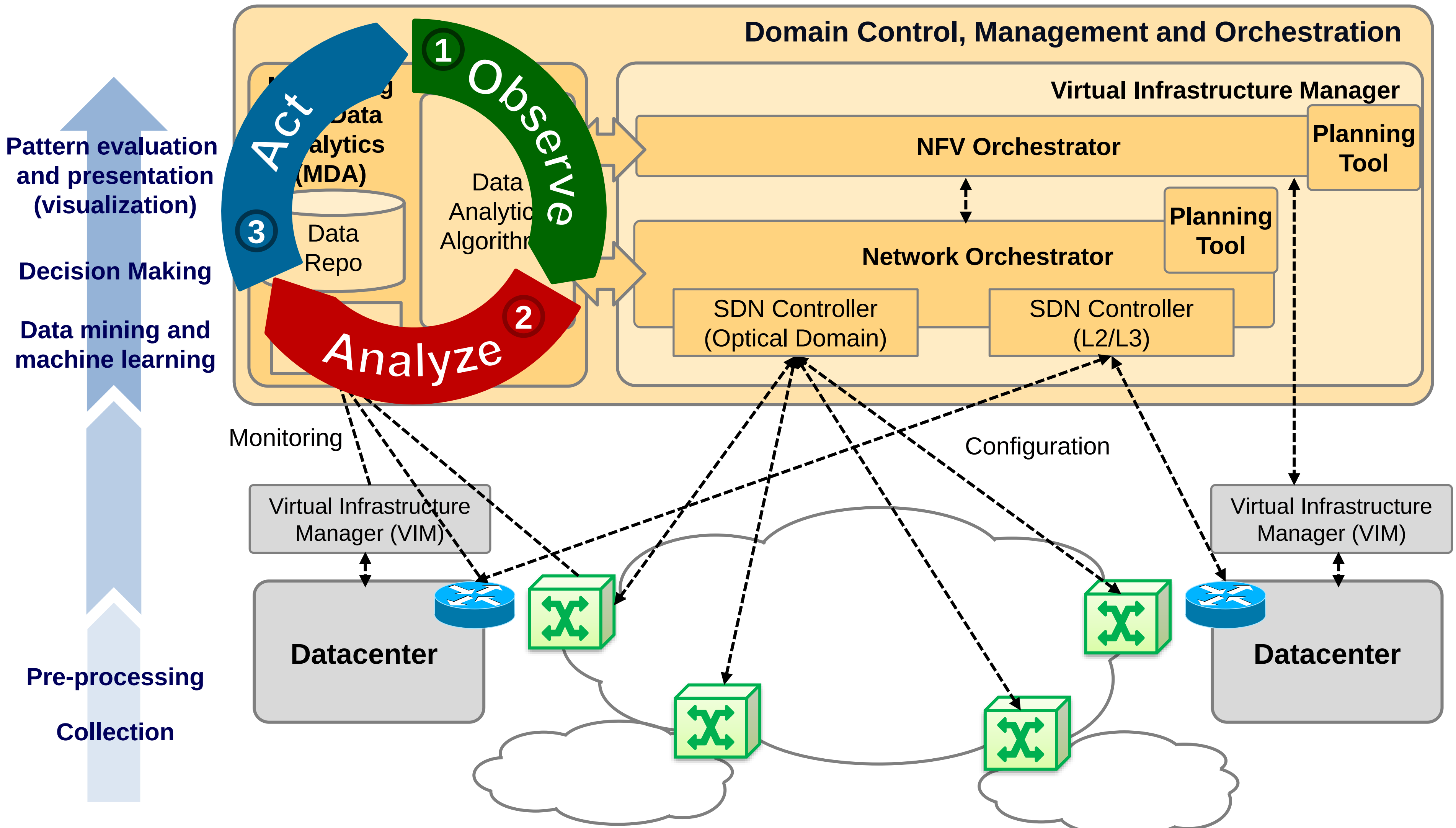
(lvelasco @ ac.upc.edu)

Campus Nord D6-107

Telecom Cloud Operation

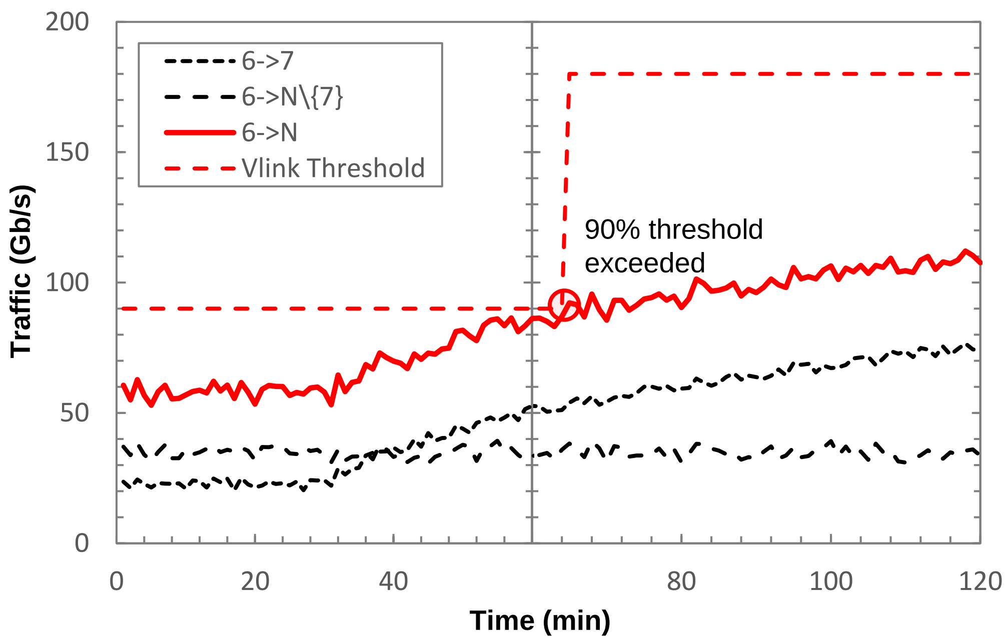


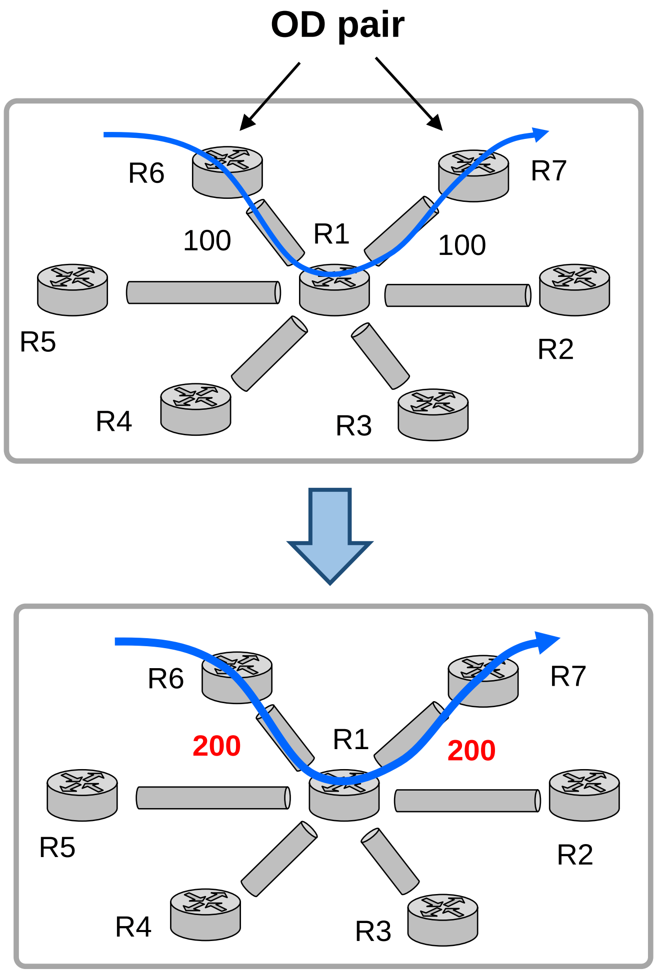
Telecom Cloud Operation



VNT reconfiguration: Threshold-based

- Typical VNT reconfiguration is based on:
  - **Monitoring vlink capacity** utilisation.
  - **Adjust it** based on a **usage threshold** (e.g., 90%).

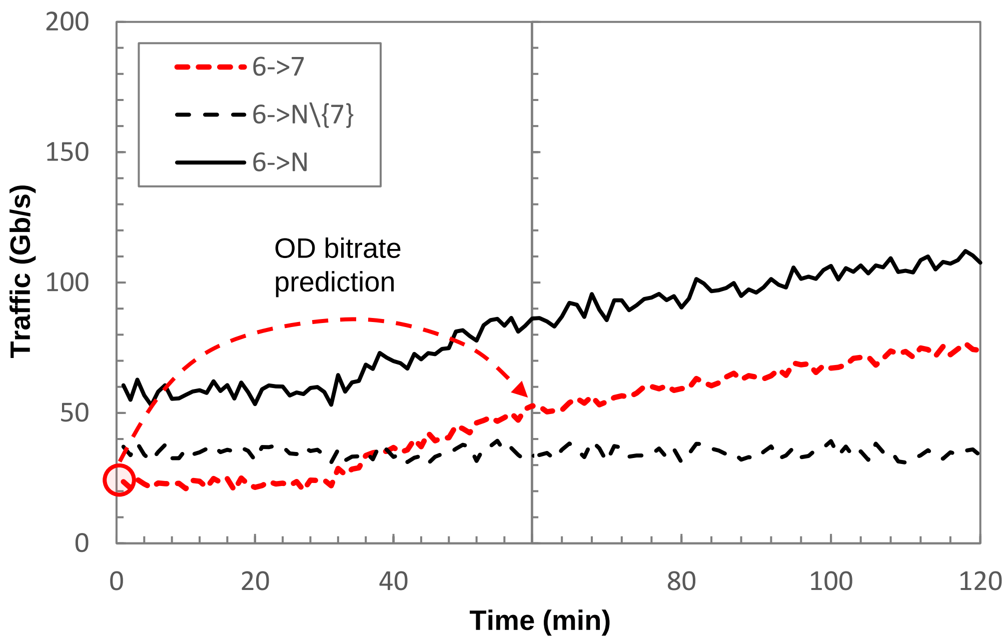


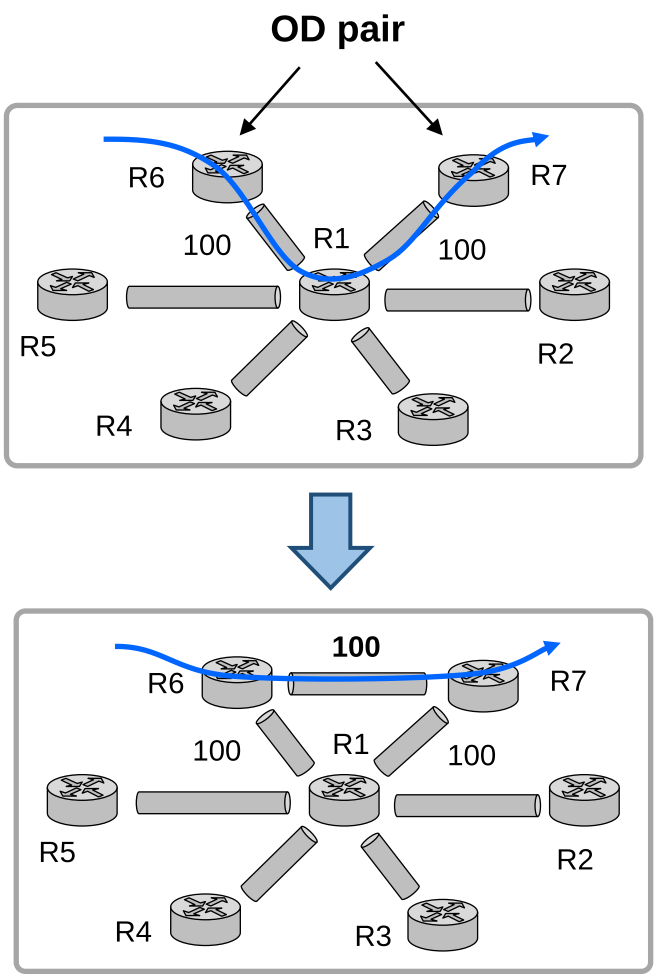


F. Morales *et al.*, "Virtual Network Topology Adaptability based on Data Analytics for Traffic Prediction," IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 9, pp. A35-A45, 2017.

VNT reconfiguration: VENTURE

- We propose a new approach:
  - **To know** not only the current, but also the **near-future origin-destination (OD) traffic**.
  - Thus **adapting the VNT** more efficiently.





VENTURE problem statement

- **Given**:
  - The current VNT represented by a graph G(*N*, *E*'), being *N* the set of routers and *E*'⊆ *E* the set of current vlinks. Set *E* is the set of all possible vlinks connecting two routers.
  - The set *P* with the transponders available in the routers; every transponder with capacity *B*.
  - The current traffic matrix *D*.
  - The **predicted** traffic matrix *OD*. The bitrate b_o of *OD* pair *o* must be served following one single path. Only in the case that b_o is enough to fill transponders with an amount over a given boundary usage *tbu*, the bitrate of pair *o* can be split into two flows and served through different paths.
- **Output:** The reconfigured VNT G*(*N*, *E**), where *E**⊆ *E*, and the paths for the traffic on G*.
- **Objective:** Maximize current and predicted served traffic matrices, whilst minimizing the total number of transponders used.

VENTURE heuristic algorithm

INPUT $G(N,E')$, D, OD, B, tbu
OUTPUT G^*, F

$Q \leftarrow \emptyset$, $U \leftarrow \emptyset$
for each $d \in D$ **do** dealloc(G, d)
for each $e \in E'$ **do**
  setCapacity(e, 0)
  releaseTransp(e)
for each $o \in OD$ **do**
  $Q \leftarrow Q \cup \{<o, g_o, l_o> = \text{splitOD}(o, B, tbu)\}$
$<Q, F'> \leftarrow PhaseI(G, Q)$
$<G', Q, F''> \leftarrow PhaseII(G, Q)$
for each $q \in Q$ **do**
  $U \leftarrow U \cup \{<o, u_o = g_o + l_o>\}$
if $U=\emptyset$ **then return** $<G', F' \cup F''>$
$<G^*, F'''> \leftarrow PhaseIII(G', U, thr)$
if $F'''=\emptyset$ **then return** INFEASIBLE
return $<G^*, F' \cup F'' \cup F'''>$

Optimal solutions **cannot be obtained** for large instances...

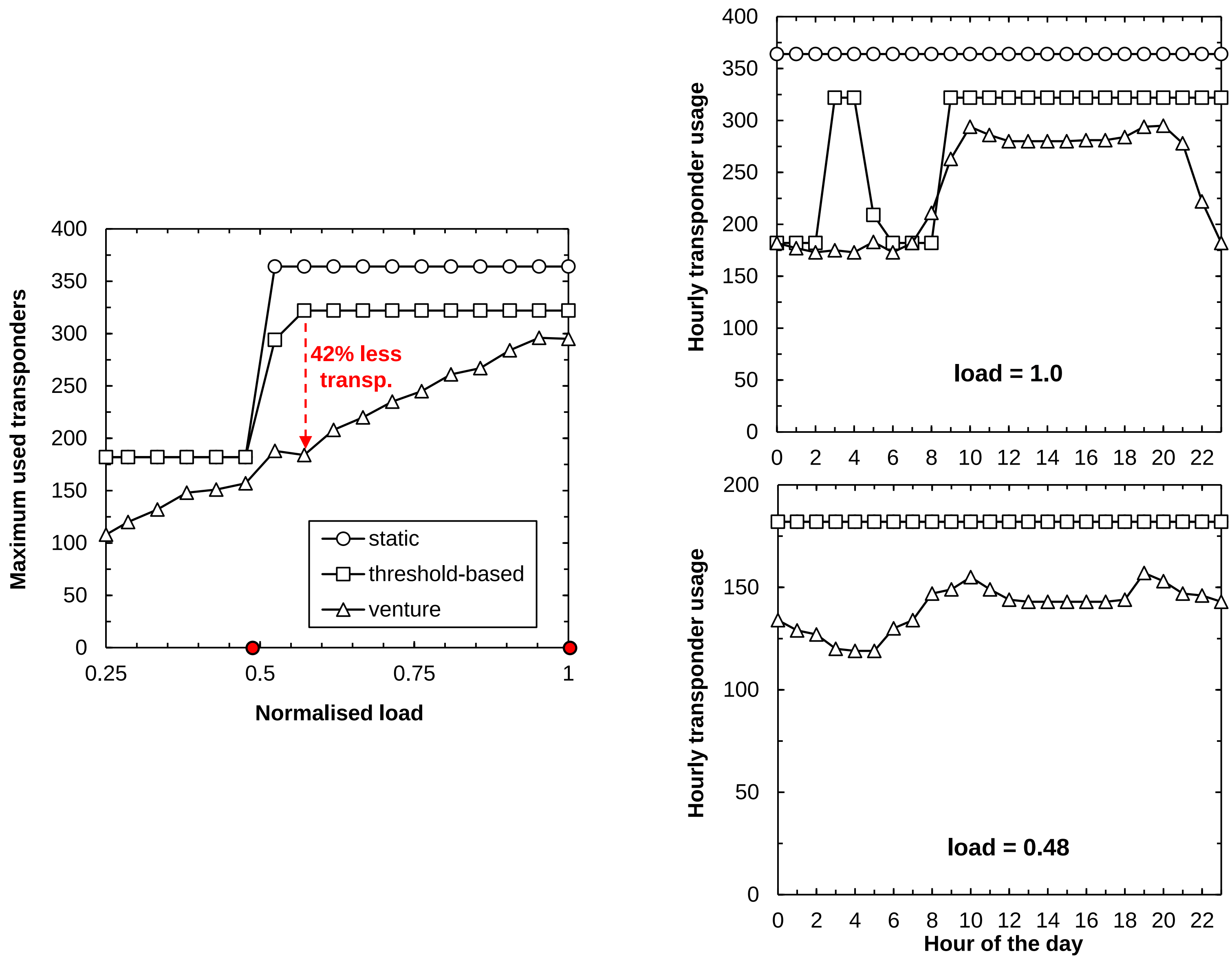
A heuristic algorithm

1. **Split** the **bitrate** of each OD pair into **high** and **low** capacity flows.
2. Route **high bitrate** flows through **direct vlinks** (Phase 1 and 2).
3. Route **low bitrate flows** minimising the transponder usage (Phase 3).

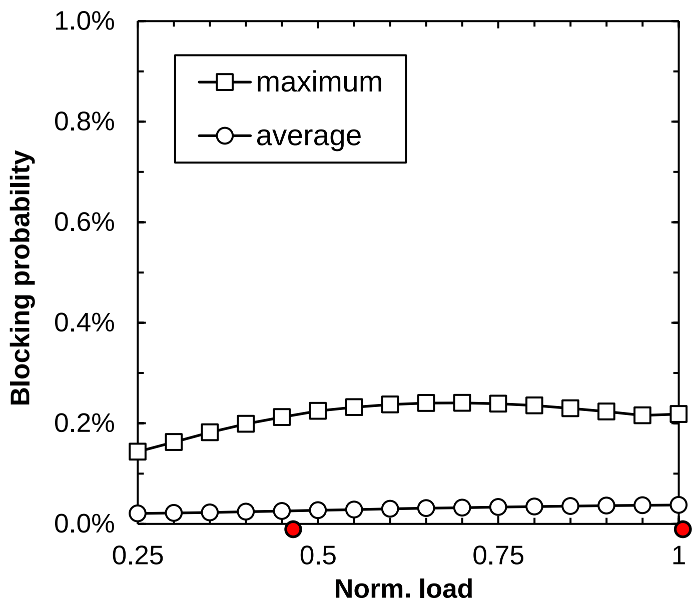
Simulation results – Scenario

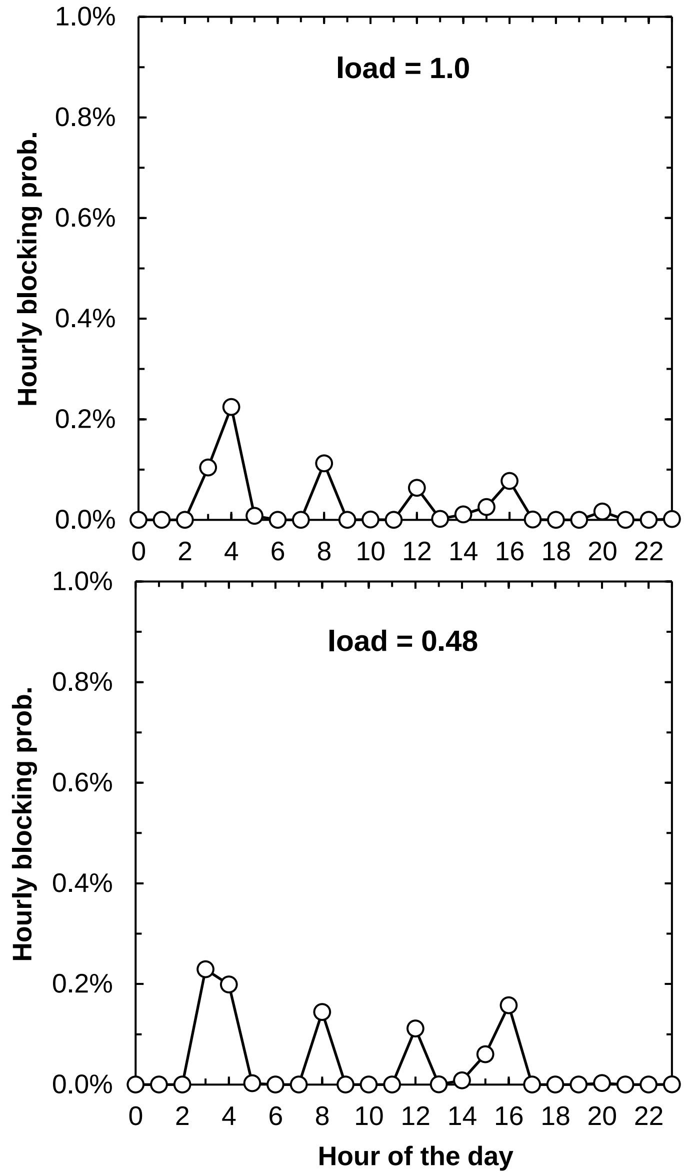
- Simulations run in an OMNeT++-based network simulator.
- 14-node, full-mesh VNT where the capacity of each vlink ranges from 100 to 200 Gb/s and each node is equipped with 26 × 100 Gb/s transponders.
- OD traffic is generated following Users and DC traffic profiles; piece-wise linear OD predictive models estimated based on monitoring data.
- We compared the following three approaches to adapt the VNT:
  - The threshold-based approach (using 90% threshold).
  - The VENTURE algorithm triggered hourly.
  - A static VNT approach, without reconfiguration.
- Simulated a network operation of several weeks.

Simulation results – Transponder utilisation



Simulation results – Blocking probability





Conclusions

- VENTURE was proposed to reconfigure the VNT, combining:
  - **Traffic prediction** based on monitoring data.
  - Combinatorial network **optimisation**.
- The VENTURE problem was formally stated and formulated as a **MILP**. A **heuristic algorithm** was proposed to solve it in practical times.
- The performance of VENTURE was compared against threshold-based VNT reconfiguration:
  - Obtaining savings between **8% and 42%** in the number of needed transponders.
  - VENTURE can **deactivate transponders** during **low traffic hours** thus, decreasing the energy consumption and releasing resources from the optical layer.

Basic phase 3 algorithm

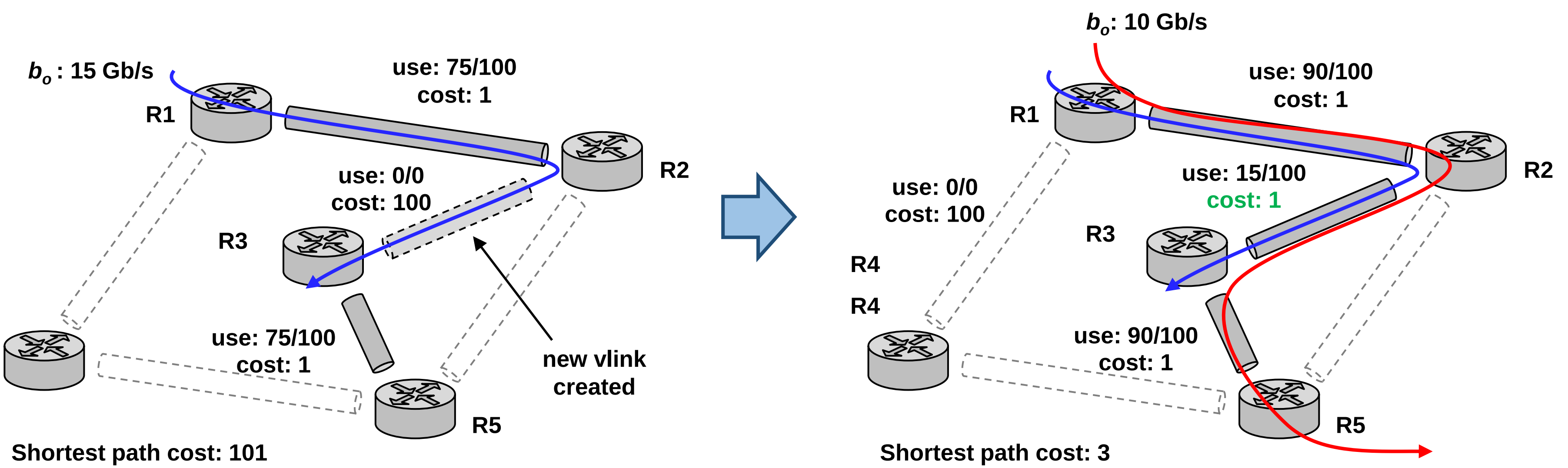
Unserved OD traffic matrix

order(o)	o_{src}	o_{dst}	b_o	
1	R2	R3	15	
2	R1	R5	10	
...				

- OD pairs with unserved bitrate are allocated based on a greedy-randomized sorting:

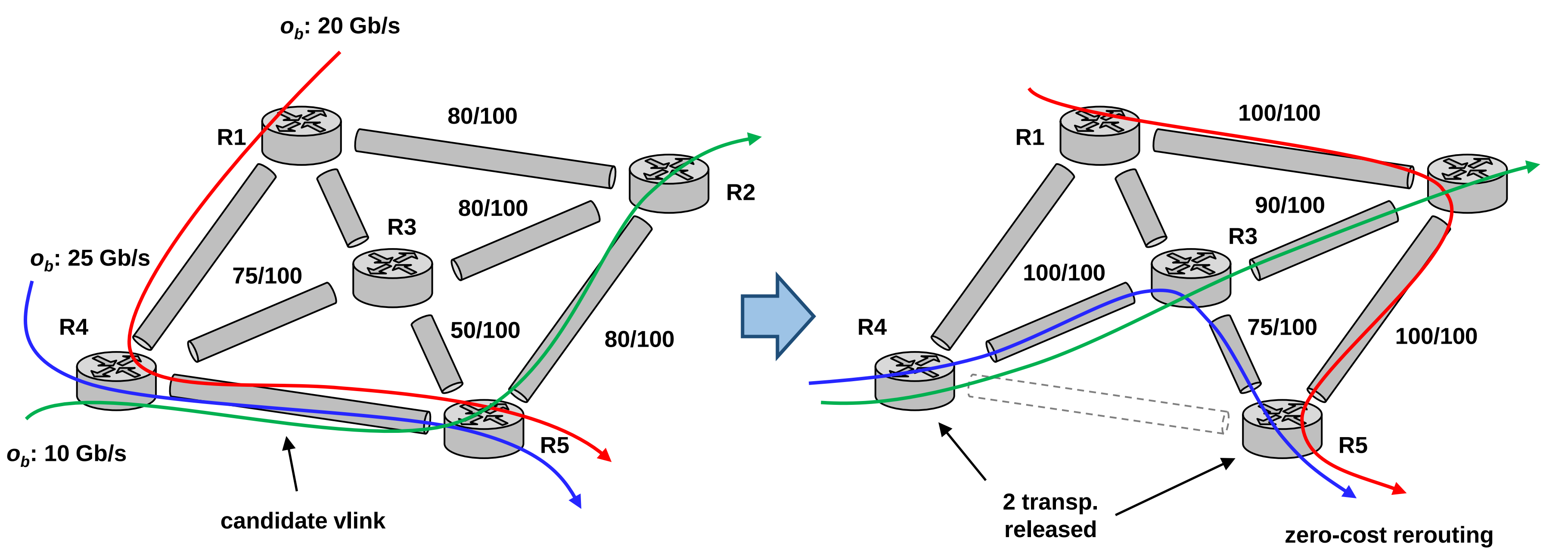
$$order(o) \propto o_b \cdot Unif(0,1)$$

- The shortest path is computed for each OD pair.
- Vlink cost is a function of the available capacity and o_b.
- The matrix is processed for several random seeds and the best solution found is returned.



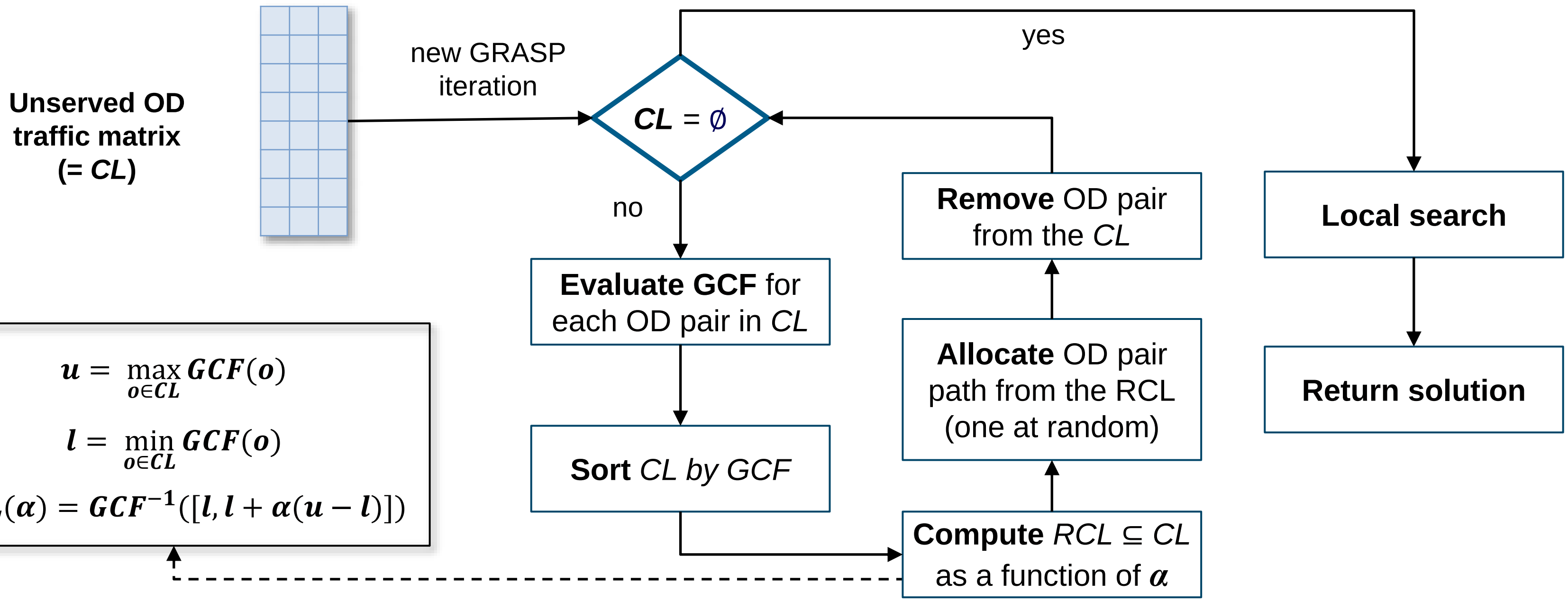
Local search procedure

- Runs at the end of every iteration of Basic VENTURE.
- Iteratively removes one vlink from the VNT.
- The **vlink with higher transponder activation** is selected.
- **Disrupted paths must be rerouted at zero cost** (greedy rerouting: high bitrate first).
- Stops when zero-cost rerouting is not posible (i.e., local optimum reached).



GRASP-based VENTURE

- Runs **multiple iterations** to find quality solutions to the problem.
- At each iteration, OD pairs are processed according to:
  - A ***greedy cost function*** (GCF) → **shortest path cost** of the OD pair.
  - A ***candidate list*** (*CL*) → the **unserved OD traffic matrix**.
  - A metaparameter $\boldsymbol{\alpha} \in [0,1]$.



Detail:

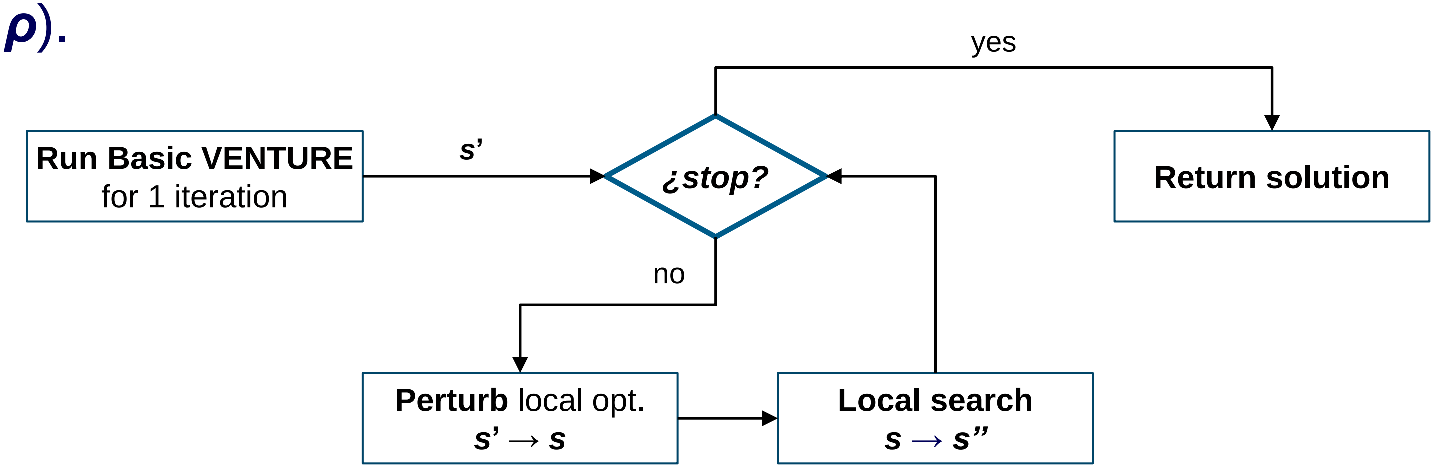
$$\boldsymbol{u} = \max_{\boldsymbol{o} \in \boldsymbol{CL}} \boldsymbol{GCF}(\boldsymbol{o})$$

$$\boldsymbol{l} = \min_{\boldsymbol{o} \in \boldsymbol{CL}} \boldsymbol{GCF}(\boldsymbol{o})$$

$$\boldsymbol{RCL}(\boldsymbol{\alpha}) = \boldsymbol{GCF}^{-1}([\boldsymbol{l}, \boldsymbol{l} + \boldsymbol{\alpha}(\boldsymbol{u} - \boldsymbol{l})])$$

Iterated local search (ILS)-based VENTURE

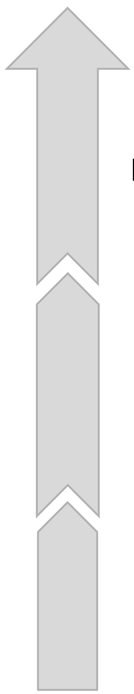
- Starts with a **constructive** phase + local search → local optimum ***s'***.
- At each iteration:
  - ***s'* is perturbed** into a different feasible solution ***s*** (escape local optimum).
  - A **local search is applied on *s*** to reach a new local optimum ***s''***.
- A perturbation **removes *n* vlinks** at random and reroutes the affected paths greedily.
- A ***strength*** metaparameter $\boldsymbol{\rho} \in [0,1]$ defines the perturbation strength ($\boldsymbol{n} \propto \boldsymbol{\rho}$).



Numerical study – Instance generation

1

OD bitrate (vlink capacity: 100 Gb/s)



High (~130 Gb/s)

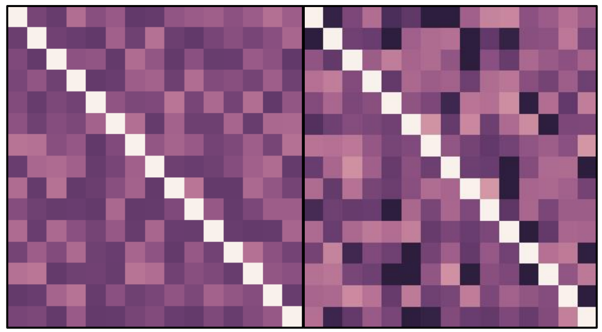
Average (~100 Gb/s)

Low (~70 Gb/s)

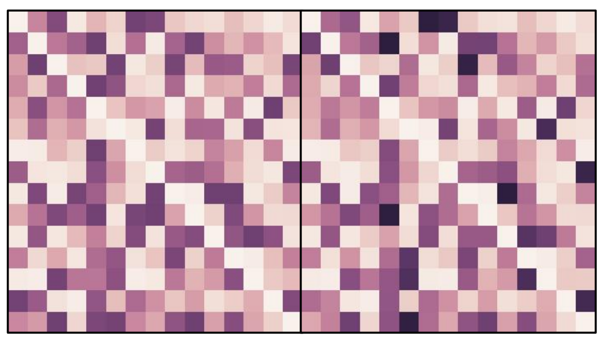
2

Volumetric

Uniform bitrate, noise fluctuation (A)

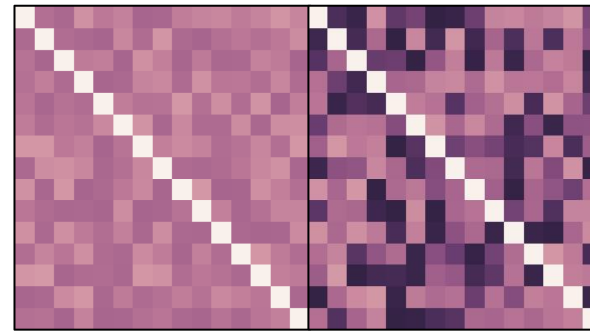


Non-uniform bitrate, noise fluctuation (B)

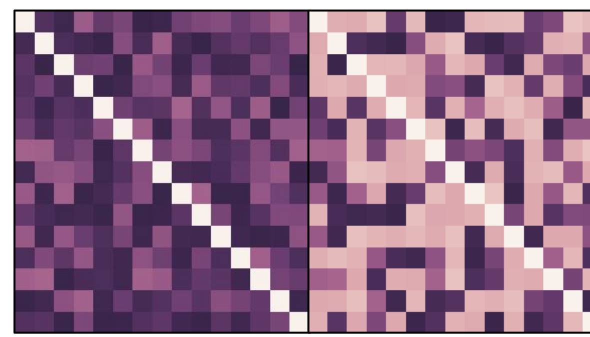


Directional

Non-uniform increment (C)



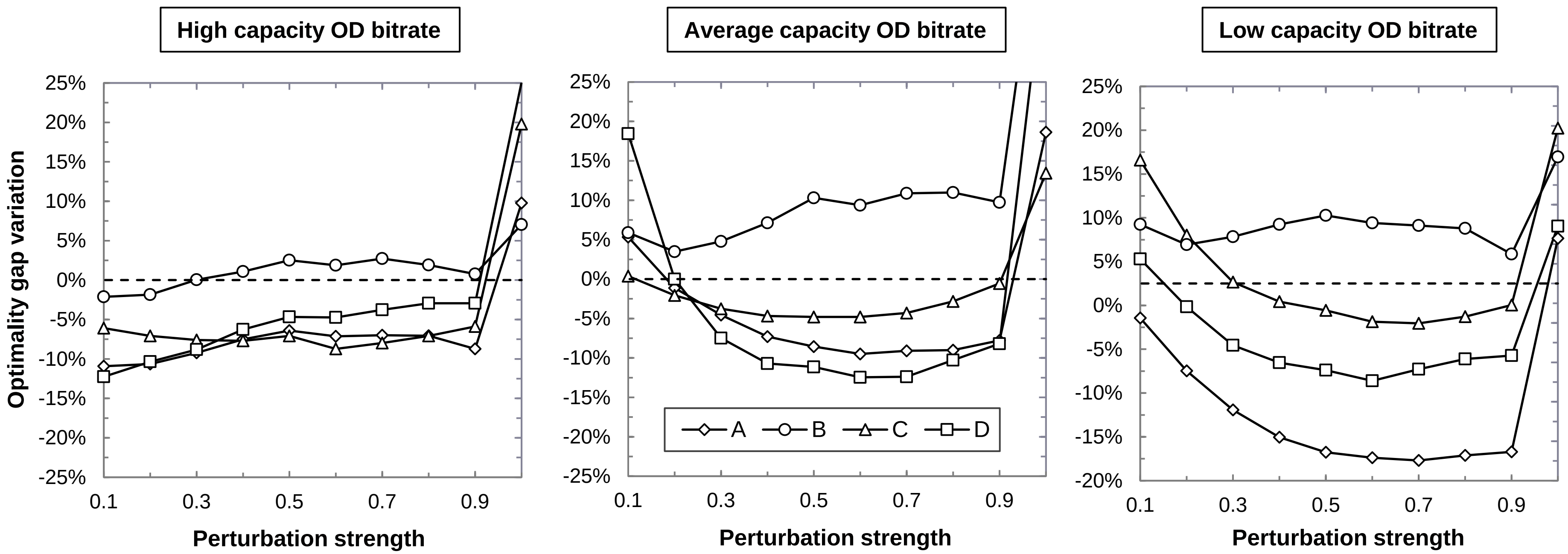
Non-uniform decrement (D)



Numerical study - Optimality

Instance		Objective function cost					Optimality gap (%)			
OD bitrate	Type	B&B (4 hours)		Heuristics (60 seconds)			B&B	Basic	GRASP	ILS
		Best bd	Best int	Base	GRASP	ILS				
High	A	455.44	484	484.14	488.25	480.99	6.27%	6.30%	7.20%	**5.61%**
	B	721.24	745	746.06	749.59	745.81	**3.29%**	3.44%	3.93%	3.41%
	C	590.00	618	618.98	624.15	616.65	4.75%	4.91%	5.79%	**4.52%**
	D	426.22	457	457.26	460.89	453.56	7.22%	7.28%	8.13%	**6.41%**
Average	A	230.74	250	250.81	255.46	249.23	8.35%	8.70%	10.71%	**8.01%**
	B	357.80	377	376.92	379.77	377.70	5.37%	**5.34%**	6.14%	5.56%
	C	291.07	315	314.57	320.24	313.55	8.22%	8.07%	10.02%	**7.72%**
	D	217.08	235	235.53	237.83	233.53	8.26%	8.50%	9.56%	**7.58%**
Low	A	142.00	165	167.37	164.91	165.63	16.20%	17.87%	**16.13%**	16.64%
	B	190.75	212	212.08	214.40	213.25	**11.14%**	11.18%	12.40%	11.80%
	C	166.69	187	187.56	186.50	187.42	12.18%	12.52%	**11.88%**	12.44%
	D	136.10	148	157.98	153.81	154.28	**8.74%**	16.08%	13.01%	13.36%

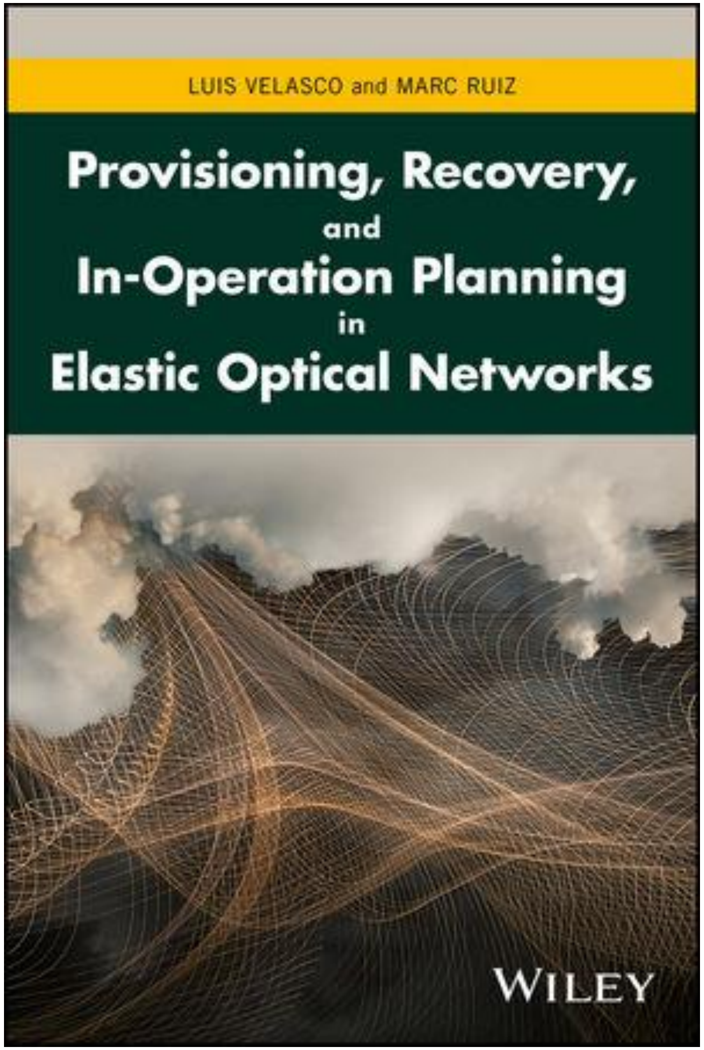
Numerical study - ILS



Conclusions

- The performance of the VENTURE algorithm was studied.
- **Three algorithms** were proposed to solve phase 3:
  - Basic VENTURE.
  - GRASP-based VENTURE.
  - ILS-based VENTURE.
- These algorithms were evaluated under different instances using an **exact solving method as reference**.
- The heuristics present an average **optimality gap of 9% (5% under low granularity instances)**.
- Among them, **ILS-based VENTURE** presents the **best performance**.

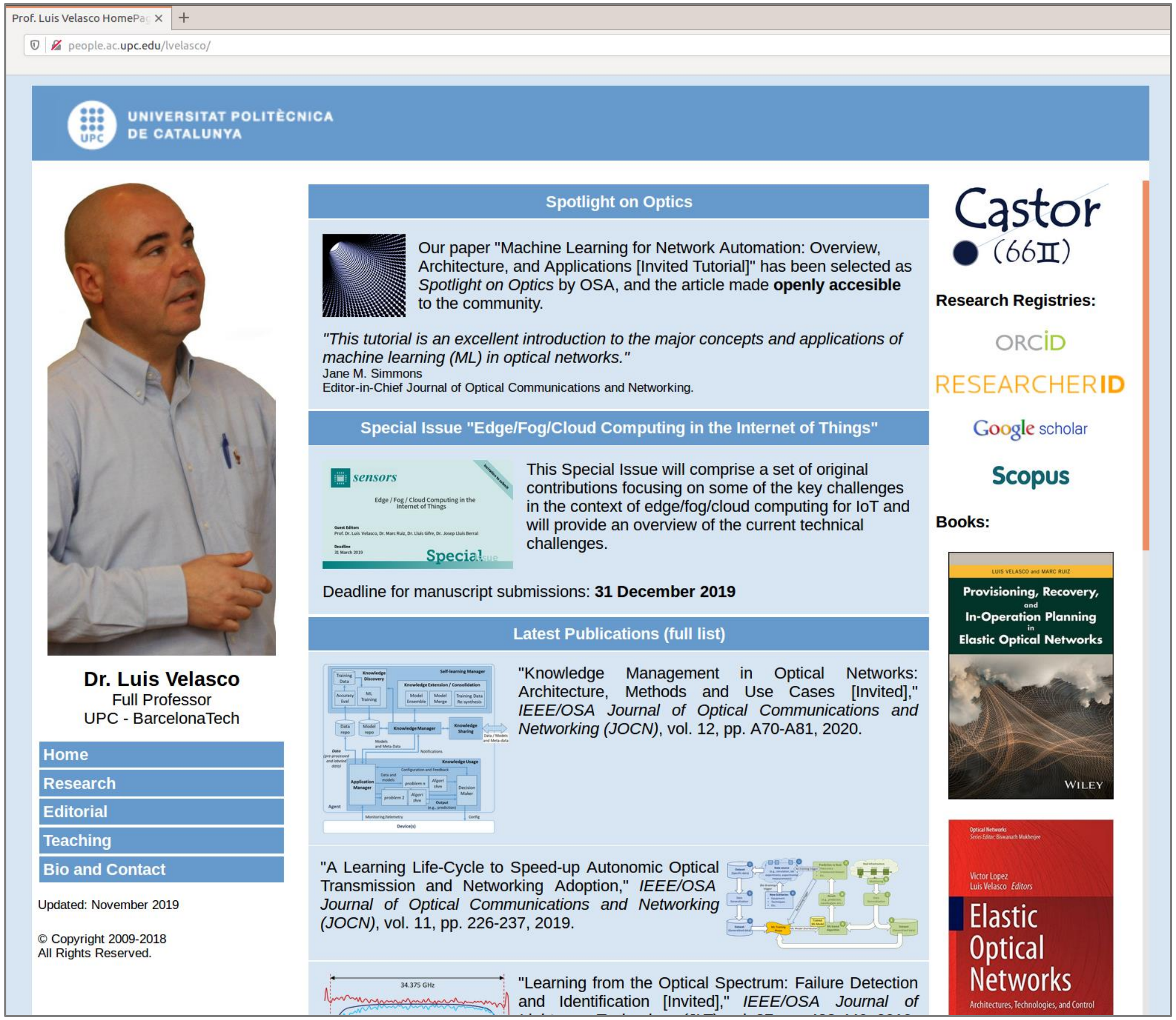
Optimization in Optical Networks



- The book discusses Elastic Optical Networks (EON) from an operational perspective.
- It presents algorithms that are suitable for real-time operation and includes experimental results to further demonstrate the feasibility of the approaches discussed. It covers practical issues such as provisioning, protection, and defragmentation. It also presents provisioning and recovery in single layer EON.
- The authors review algorithms for provisioning point-to-point, anycast, and multicast connections, as well as transfer-based connections for datacenter interconnection. They also include algorithms for recovery connections from failures in the optical layer and in-operation planning algorithms for EONs.

Personal web page

http://people.ac.upc.edu/lvelasco/



UNIVERSITAT POLITÈCNICA DE CATALUNYA

DAC

Algorithmic Methods for Mathematical Models (AMMM)

Some Applications

Luis Velasco

(lvelasco @ ac.upc.edu)

Campus Nord D6-107