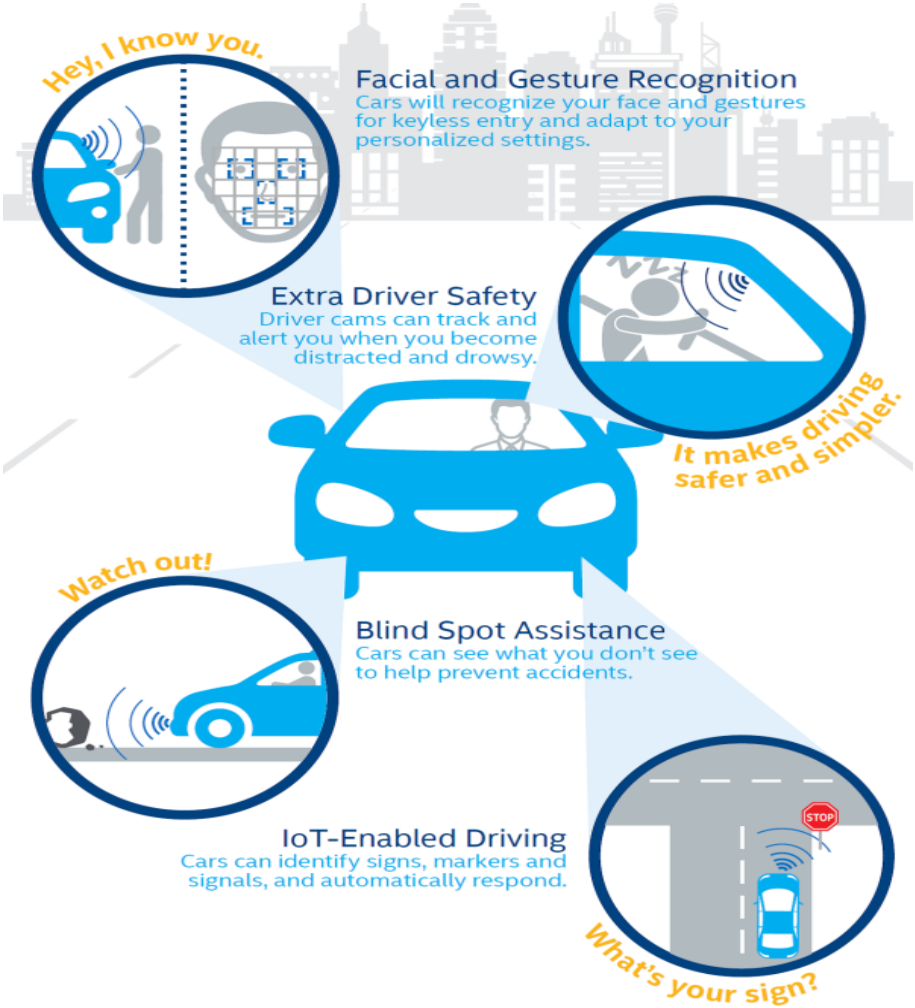
PARALLEL Q-LEARNING FOR IOT ENVIROMENTS

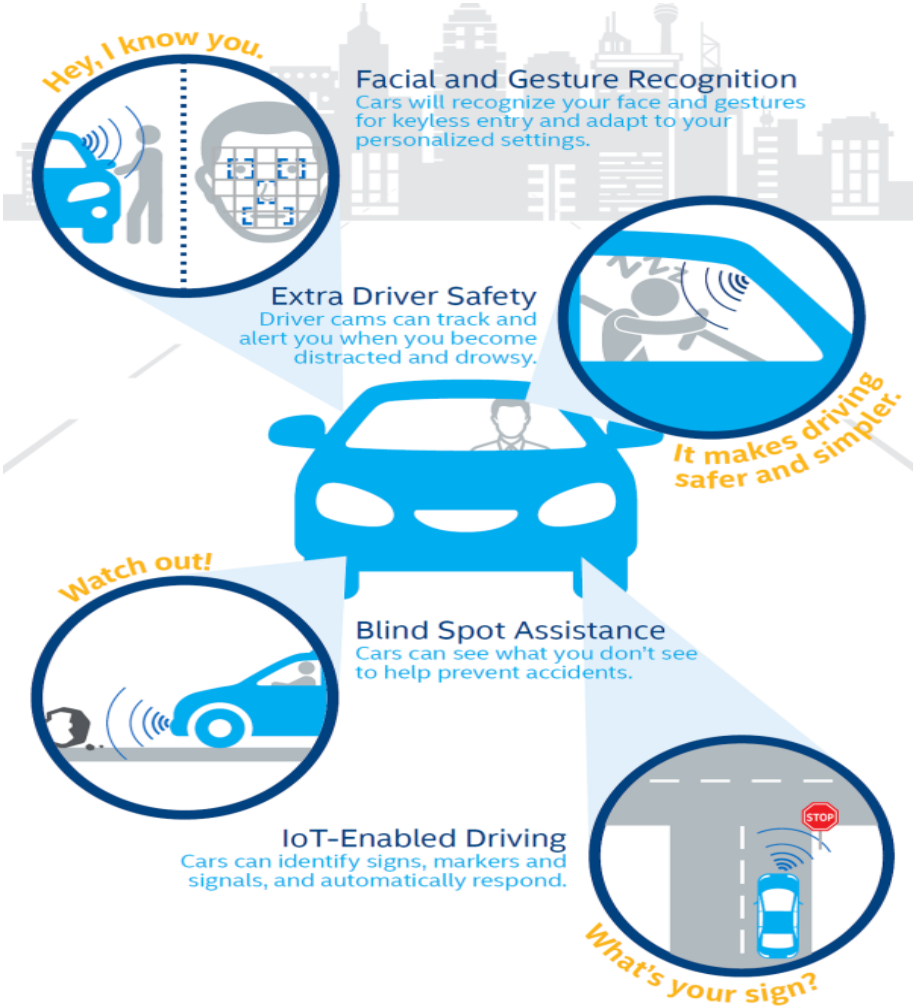
MIGUEL CAMELO, JEROEN FAMAEY AND STEVEN LATRÉ

MANY IOT APPLICATIONS REQUIRE ADAPTIVE AND SMART CONTROL….



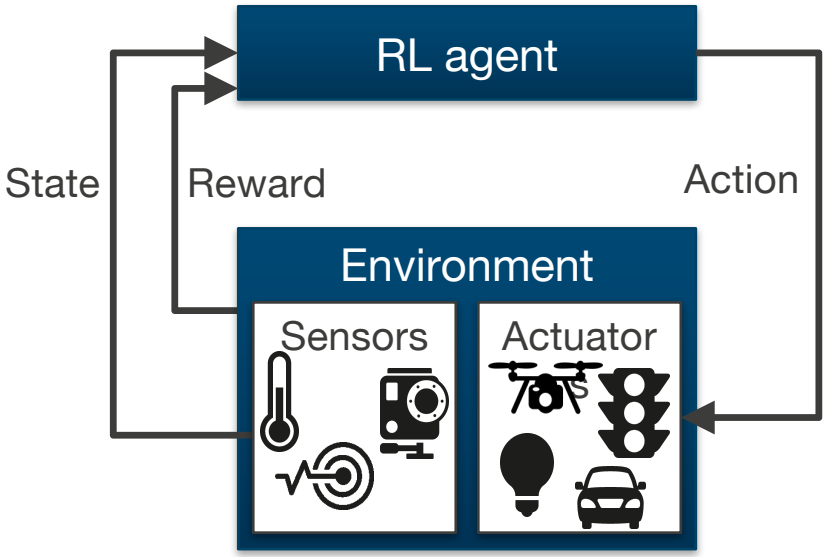
imec

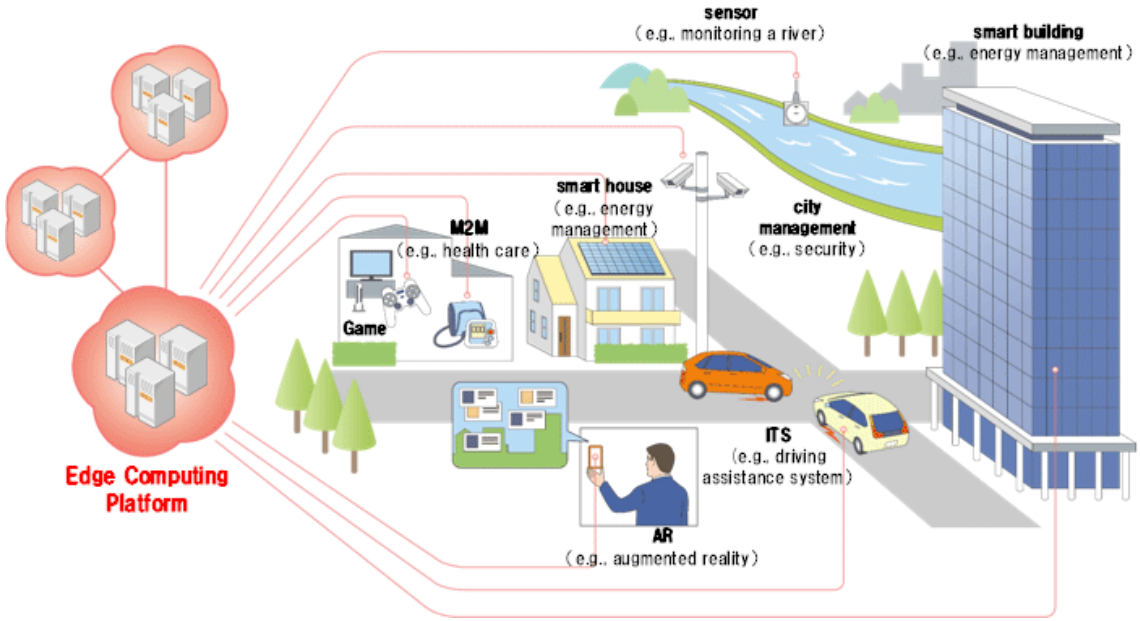
MANY IOT APPLICATIONS REQUIRE ADAPTIVE AND SMART CONTROL…. WITH LOW LATENCY



imec

RL AS A MEANS TO CONTROL IOT DEVICES





imec

PARALLEL Q-LEARNING FOR CONSTRAINED COMPUTING ENVIRONMENTS

Q-Learning (QL): Strengths and weaknesses

Traditional Parallel QL: the need of a new model for PQL

PQL with co-allocation of processing and data

Performance evaluation

PARALLEL Q-LEARNING FOR CONSTRAINED COMPUTING ENVIRONMENTS

Q-Learning (QL): Strengths and weaknesses

Traditional Parallel QL: the need of a new model for PQL

PQL with co-allocation of processing and data

Performance evaluation

Q-LEARNING (QL) IS A SIMPLE AND EFFICIENT RL ALGORITHM

Model free RL -> No previous information of the environment is needed.

Guarantee of convergence if tabular representation of Q-values.

Initialize Q(s, a) arbitrarily

1. Repeat (for each episode):
2. Initialize s
3. Repeat (for each iteration inside of the episode):
4. Choose a from s using policy derived from Q (e.g. ε-greedy)
5. Take action a, observe r, s'
6. $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
7. $s \leftarrow s'$
8. Until s is terminal

S/A	a_1	a_2	a_3	...	a_n
s_1	$Q(s_1, a_1)$	$Q(s_2, a_2)$	$Q(s_3, a_3)$	...	$Q(s_1, a_n)$
s_2	$Q(s_2, a_1)$	$Q(s_2, a_2)$	$Q(s_2, a_3)$	...	$Q(s_2, a_n)$
s_3	$Q(s_3, a_1)$	$Q(s_3, a_2)$	$Q(s_3, a_3)$	...	$Q(s_3, a_n)$
....	...	...	...	...	...
s_m	$Q(s_m, a_3)$	$Q(s_m, a_3)$	$Q(s_m, a_3)$	...	$Q(s_m, a_n)$

imec

MOST OF RL ALGORITHMS (INCLUDING QL) HAVE THREE MAIN LIMITATIONS

Sequential and designed for centralized environments.

Requires long training time to learn optimal policy.

Table-based algorithms have scalability issues in large scale problems.

PARALLEL QL AND COLLABORATIVE MA-QL REDUCE THE TRAINING TIME

Collaborative Multi-agent (MA) QL

- A unique environment is altered by actions of agents.
- Agents need to coordinate actions.

Parallel QL

- All agents learn same task
- Episodes can be reduced in a factor of 1/n
- There is guarantee of convergence

PARALLEL QL AND COLLABORATIVE MA-QL REDUCE THE TRAINING TIME

Collaborative Multi-agent (MA) QL

- A unique environment is altered by actions of agents.
- Agents need to coordinate actions.

Parallel QL

- All agents learn same task
- Episodes can be reduced in a factor of 1/n
- There is guarantee of convergence

Both approaches allow decentralized implementation

PARALLEL QL AND COLLABORATIVE MA-QL REDUCE THE TRAINING TIME

Collaborative Multi-agent (MA) QL

- A unique environment is altered by actions of agents.
- Agents need to coordinate actions.

Parallel QL

- All agents learn same task
- Episodes can be reduced in a factor of 1/n
- There is guarantee of convergence

Both approaches allow decentralized implementation

FUNCTION APPROXIMATORS (E.G. NEURONAL NETWORKS) DO NOT GUARANTEE CONVERGENCE

Neuronal Networks:

- Provide a compact representation of the Q-Table
- Allow solving problems with large number of states.

Neuronal Networks **do not**:

- Guarantee convergence.
- Provide simple and intuitive implementation.
- Slower to find optimal policy than table-based approach.

FUNCTION APPROXIMATORS (E.G. NEURONAL NETWORKS) DO NOT GUARANTEE CONVERGENCE

Neuronal Networks:

- Provide a compact representation of the Q-Table
- Allow solving problems with large number of states.

Neuronal Networks **do not:**

- Guarantee convergence.
- Provide simple and intuitive implementation.
- Slower to find optimal policy than table-based approach.

We need to go back to table-based approaches

PARALLEL Q-LEARNING FOR CONSTRAINED COMPUTING ENVIRONMENTS

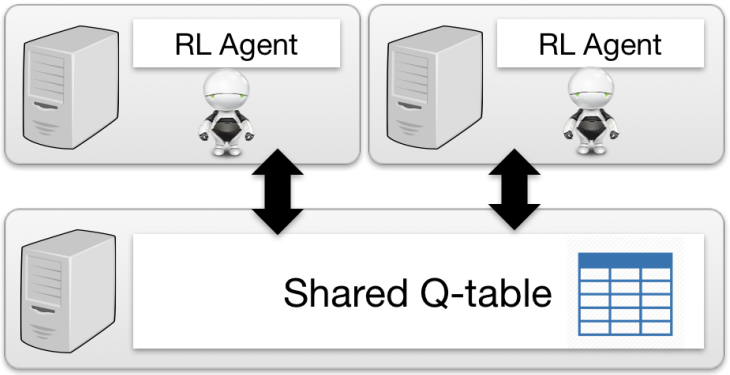
Q-Learning (QL): Strengths and weaknesses

Traditional Parallel QL: the need of a new model for PQL

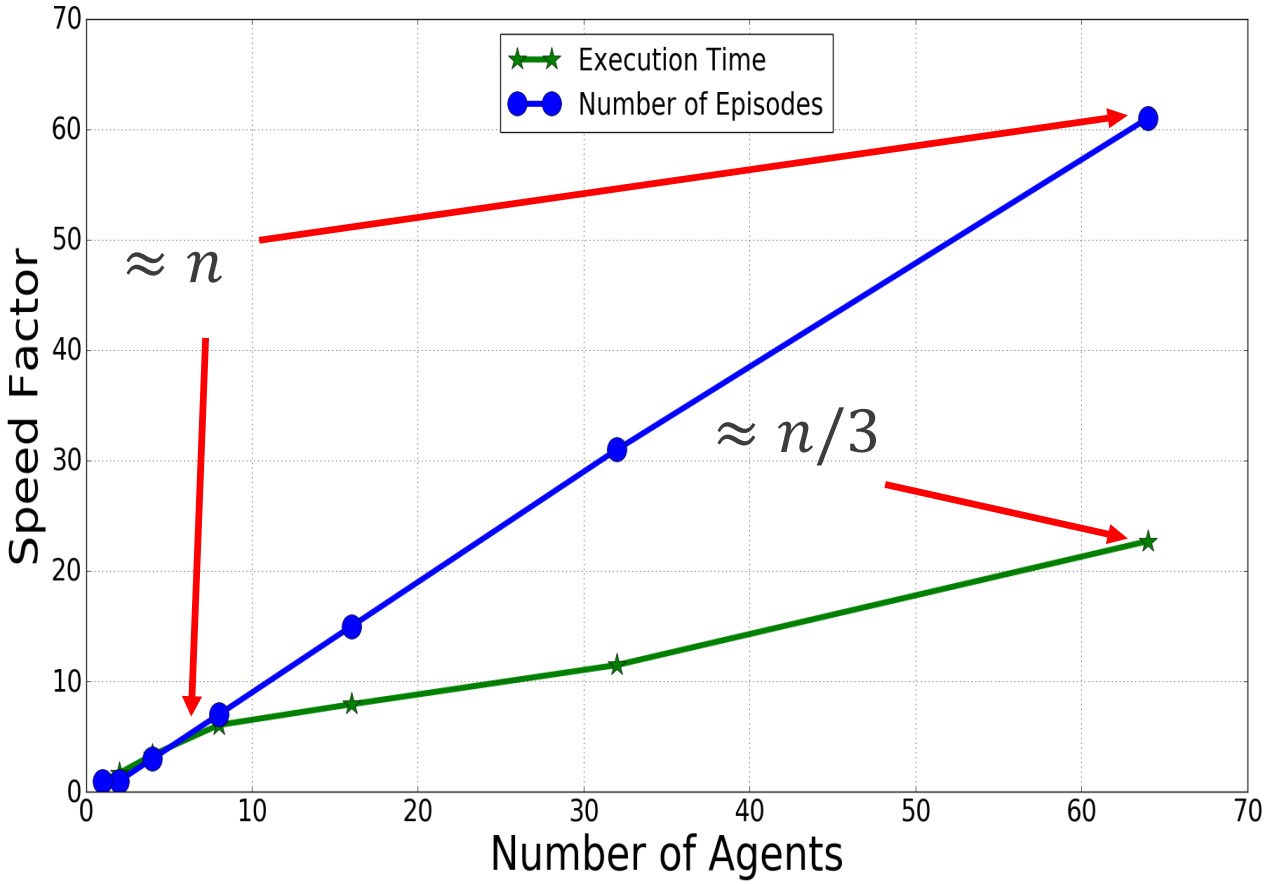
PQL with co-allocation of processing and data

Performance evaluation

TRADITIONAL PQL: CENTRAL QT BOTTLENECK WITH HIGH-COST COMMUNICATION SCHEME



Constant Share RL algorithm with centralized table



imec

A MA-RL MODEL BASED ON FUNCTIONS PROVIDES THE FLEXIBILITY THAT WE NEED

Many MA-RL algorithms (including PQL) execute several (underlying) tasks while they are running

- Allocation of resources.
- Creation and deployment of RL agents.
- Synchronization among agents.
- Data collection.
- Execution of RL algorithm.
- Management of QT.

A MA-RL MODEL BASED ON FUNCTIONS PROVIDES THE FLEXIBILITY THAT WE NEED

Many MA-RL algorithms (including PQL) execute several (underlying) tasks while they are running

Function	
Management	▪ Allocation of resources. ▪ Creation and deployment of RL agents. ▪ Synchronization among agents. ▪ Data collection.
Processing	▪ Execution of RL algorithm.
Storage	▪ Management of QT.

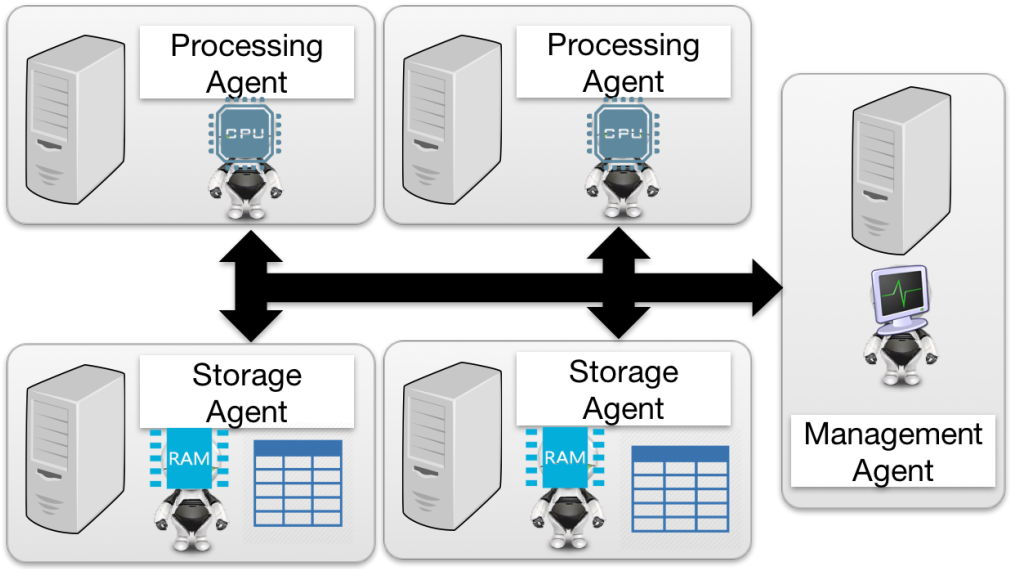
EACH FUNCTION IS ASSIGNED TO A SPECIFIC KIND OF AGENT

Processing agent (PrA)

- This is a QL agent that works on a (possible external) QT.

Storage Agent (StA)

- It stores (part of) the QT.
- It serves requests from PrA.



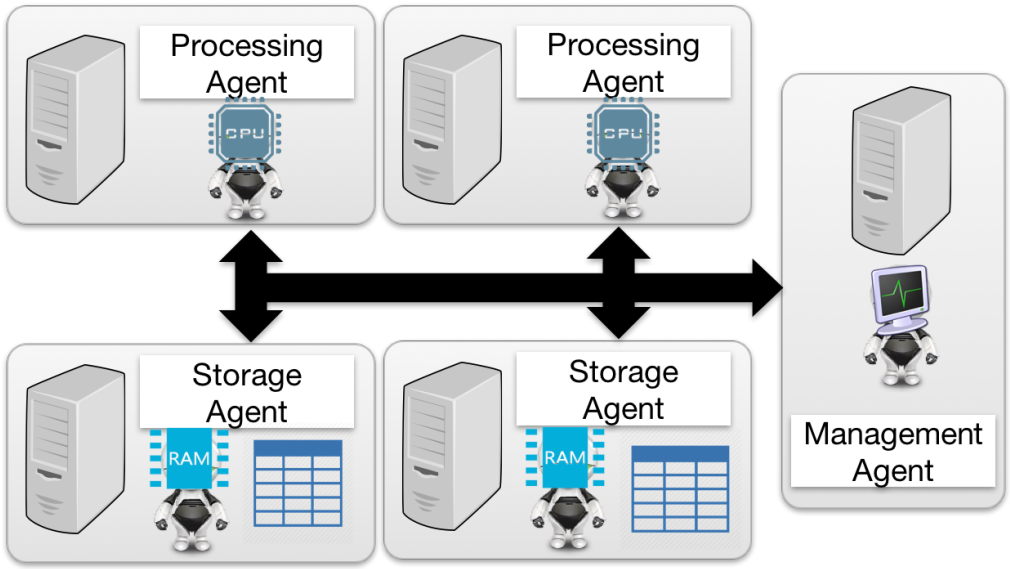
Constant Share RL algorithm modeled as a function of management, storage and processing agent

imec

EACH FUNCTION IS ASSIGNED TO A SPECIFIC KIND OF AGENT

Management agent (MngtA)

- Deploy StAs on available storage resources.
- Build and distributed QT among StAs.
- Deploy PrA on available processing resources.



Constant Share RL algorithm modeled as a function of management, storage and processing agent

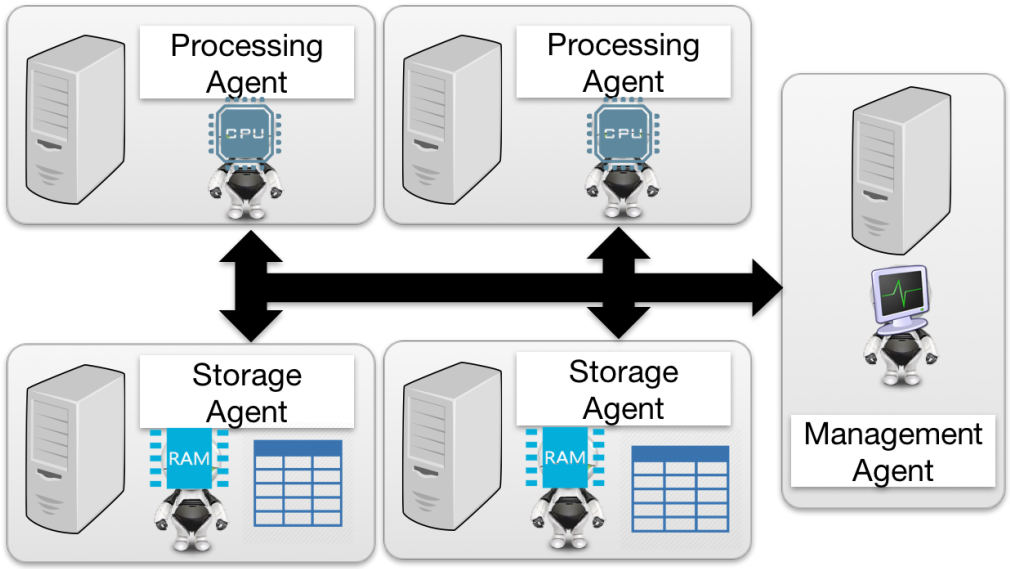
imec

THE NEW MULTI-AGENT MODEL ALLOWS A COMPLETE DECENTRALIZED EXECUTION

Applicable to many MA-RL algorithm

QT is not a bottleneck anymore

It can guarantee convergence



Constant Share RL algorithm modeled as a function of management, storage and processing agent

imec

PARALLEL Q-LEARNING FOR CONSTRAINED COMPUTING ENVIRONMENTS

Q-Learning (QL): Strengths and weaknesses

Traditional Parallel QL: the need of a new model for PQL

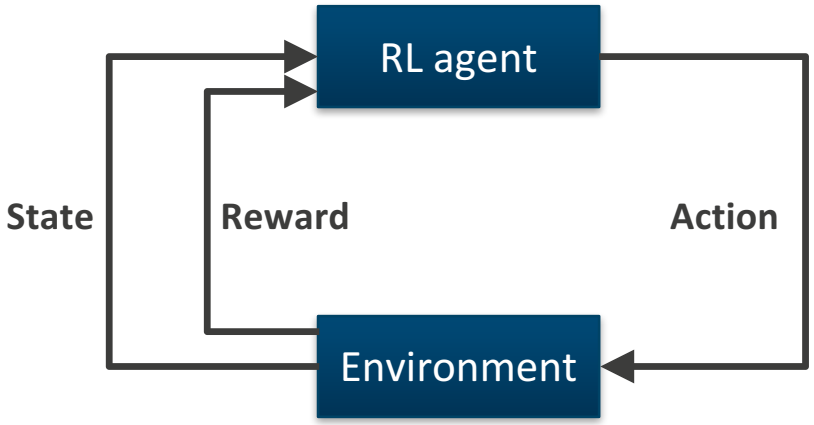
PQL with co-allocation of processing and data

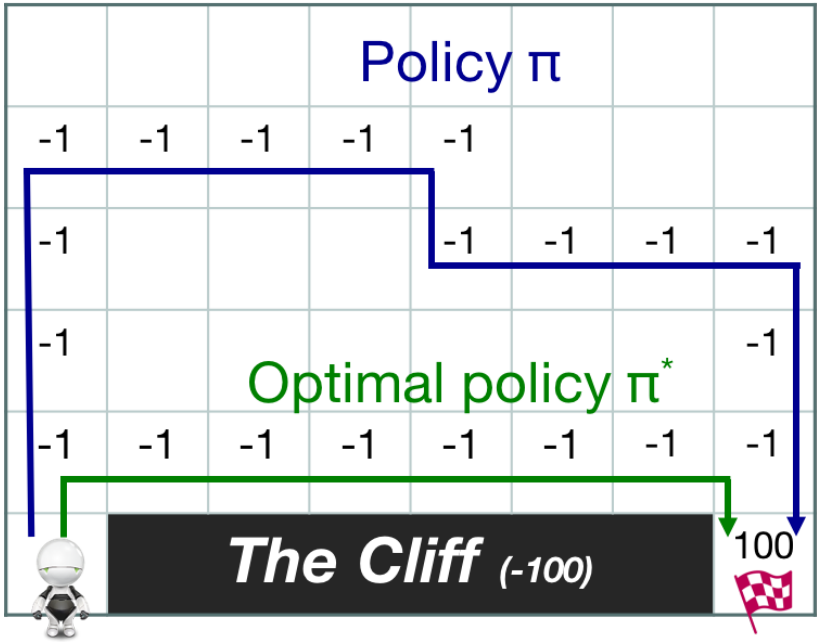
Performance evaluation

EXAMPLE: THE CLIFF PROBLEM

- Goal: The agent navigates a grid from start to goal without falling into the cliff.

Reward function:
- -100 if robot falls into the cliff
- -1 for each movement of the robot
- +100 if robot reaches the goal





imec

PCSP : PQL WITH CO-ALLOCATION OF STORAGE AND PROCESSING

Two main procedures:

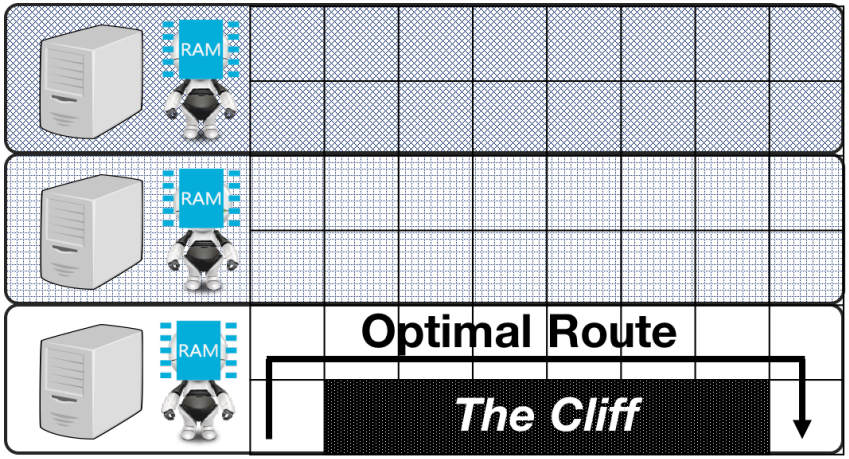
- Creation and deployment of agents
- QL execution

Algorithm 2 PQL with Co-allocation of Storage and Processing (PCSP)

Require: Number of states $|S|$ and actions $|A|$,
Require: Set of physical nodes M
1: **if** $|M| > 0$ **then**
2: Deploy a MngtA
3: $q \leftarrow$ total available memory across physical nodes.
4: **if** $O(|S||A|) \leq q$ **then**
5: Compute table partitions P
6: Deploy $n_s = |P|$ StA
7: Initialize the QT
8: Create and deploy n_p PrA
9: **while** p has not converged, $\forall p \in PrA$ **do**
10: Execute QL on p, $\forall p \in PrA$
11: **end while**
12: Obtain statics from p, $\forall p \in PrA$
13: Retrieve the optimal policy π from QT
14: **end if**
15: **end if**
16: **return** π

imec

STORE CONSECUTIVES STATES IN THE SAME STA REDUCES THE COMMUNICATION COST



Two states s and s' are consecutives if s' is the next state of s' after execution some action.

Operations such as $\max_{a'} Q(s', a')$ are completely local.

States of a (possible optimal) solution are managed by a few (maybe unique) StAs.

THE NUMBER OF STATES STORED PER NODE IS WELL-BALANCED ACROSS THE RESOURCES

n_s (number of StAs) $= P$ (number of partitions)

$P = \left\lceil \frac{c}{2} \right\rceil$, c is the total processing resources.

In homogeneous environments StA stores $\frac{|S||A|}{n_s}$

In heterogeneous environments each StA i stores s_i bits

$$\sum_{i=1}^{n_S} s_i = \Omega(|S||A|), 0 \leq s_i \leq |S||A|$$

EXPLOIT THE STORAGE OF CONSECUTIVES STATES BY DEPLOYING PRA NEXT TO STA

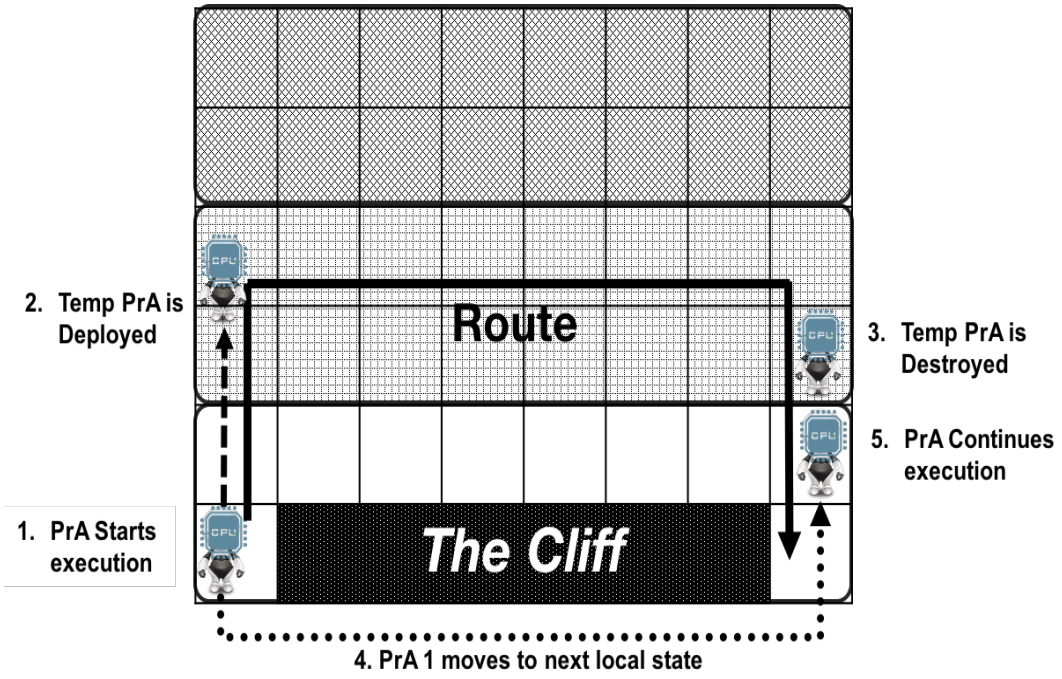
Deploy PrAs on nodes where StA have been already deployed

n_p (number of PrAs) = n_s (number of StAs)

Minimum communication cost if PrA is exploring states managed by the co-allocated StA

In case of requiring states from remote StAs, a temporal PrA is deployed

DEPLOYING TEMPORAL PRA MINIMIZES NUMBER OF REQUESTS FROM ORIGINAL PRA TO REMOTE STAS.



Algorithm 3 Modified Q-Learning

Require: Access to Distributed Q-Table with $|S||A|$ values

repeat(for each episode)
  Initialize s
  repeat(for each iteration)
    Use behavior policy to choose a
    Take action a, observe r, s'
    if s' is not stored in any local StA **then**
      Deploy temp PrA on node where s' is stored.
      repeat
        Wait
      until temp PrA has finished
      $s \leftarrow s'$
    else
      $TDerror = \max_{a'} Q(s', a') - Q(s, a)$
      $Q(s, a) \leftarrow Q(s, a) + \alpha \cdot (r + \gamma \cdot TDerror)$
      $s \leftarrow s'$
    end if
  until s is goal state
until max episodes = true or convergence = true

imec

PARALLEL Q-LEARNING FOR CONSTRAINED COMPUTING ENVIRONMENTS

Q-Learning (QL): Strengths and weaknesses

Traditional Parallel QL: the need of a new model for PQL

PQL with co-allocation of processing and data

Performance evaluation

WE EVALUATED PCSP AGAINST SEVERAL QL ALGORITHMS UNDER TWO METRICS (EPISODES AND EXECUTION TIME)

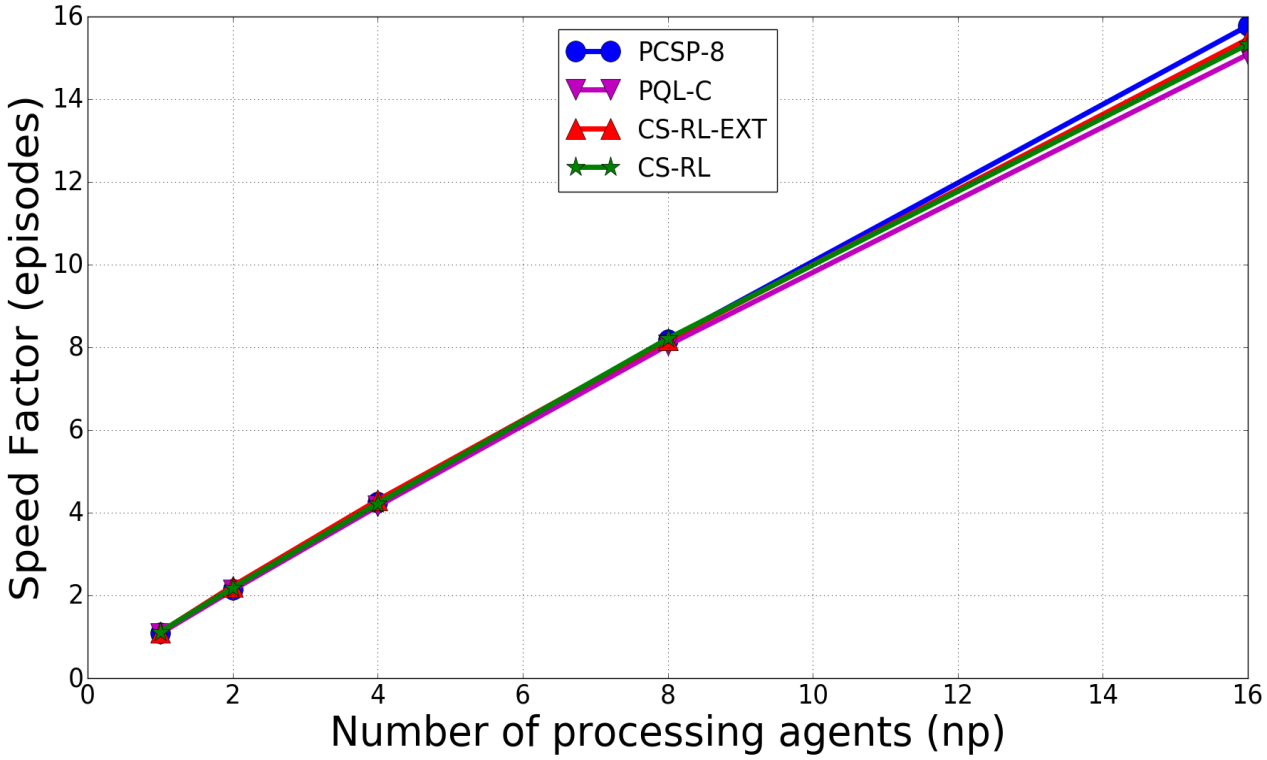
The single-agent Q-learning algorithm (SQL)

A naive implementation of the Constant Share RL (CS-RL) algorithm

An extended version of the CS-RL with local cache on the processing agents (CS-RL-EXT)

The PQL algorithm via a communication scheme with local cache (PQL-C).

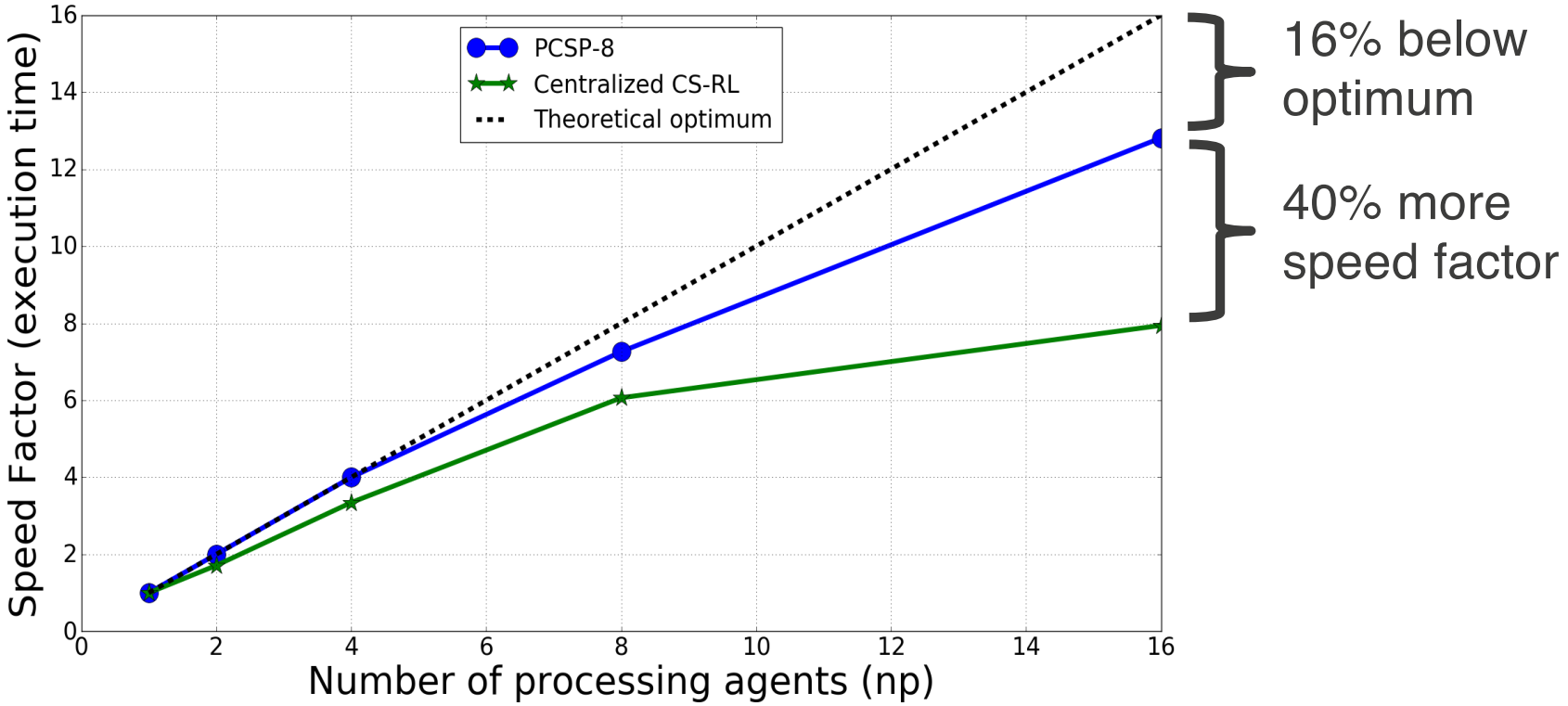
PCSP REDUCE THE NUMBER OF EPISODES IN SAME RATE AS PQL WITH CENTRAL QT



Speed factor with respect to number of episodes of a SAQ execution

imec

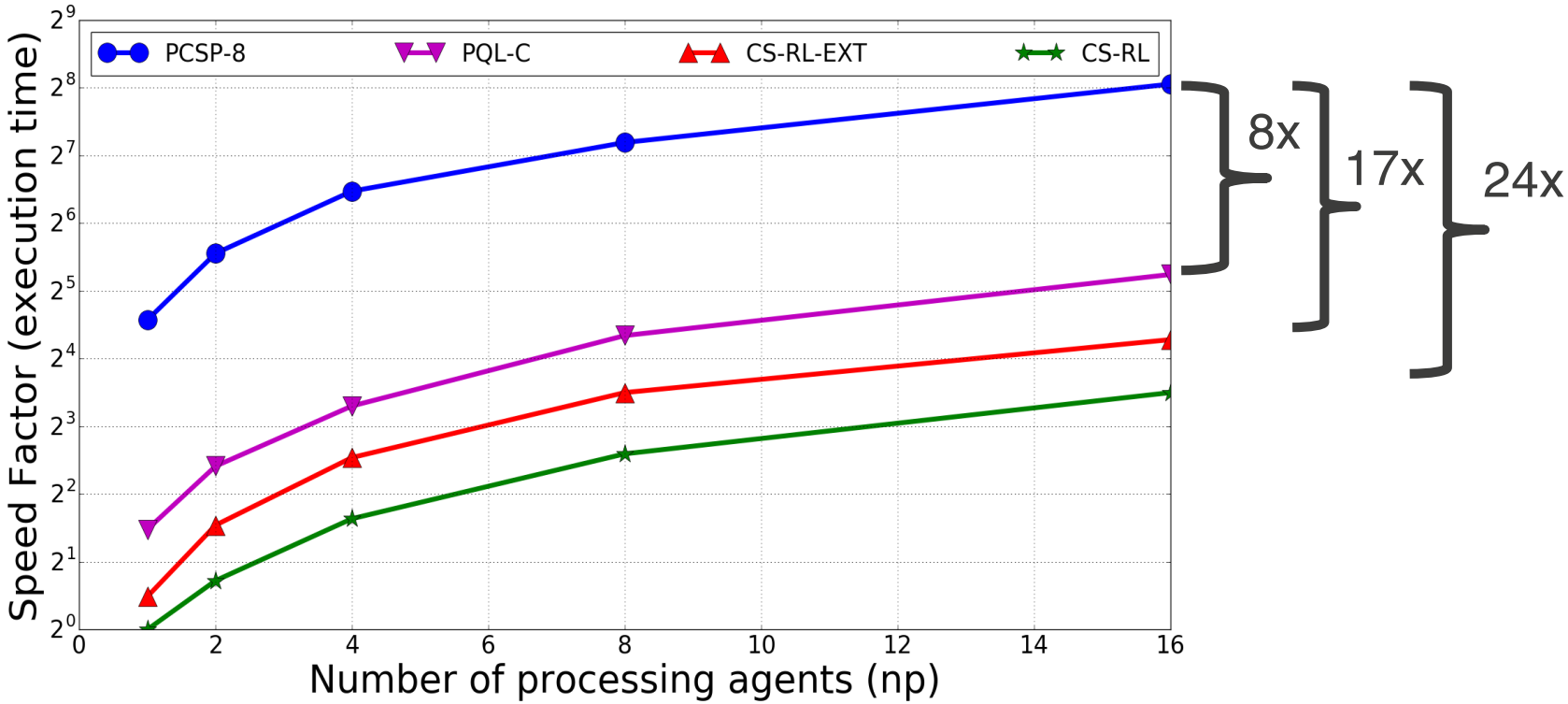
PCSP SCALES BETTER THAN CENTRALIZED PQL



Speed factor with respect to the execution time of the same algorithm with only one agent

imec

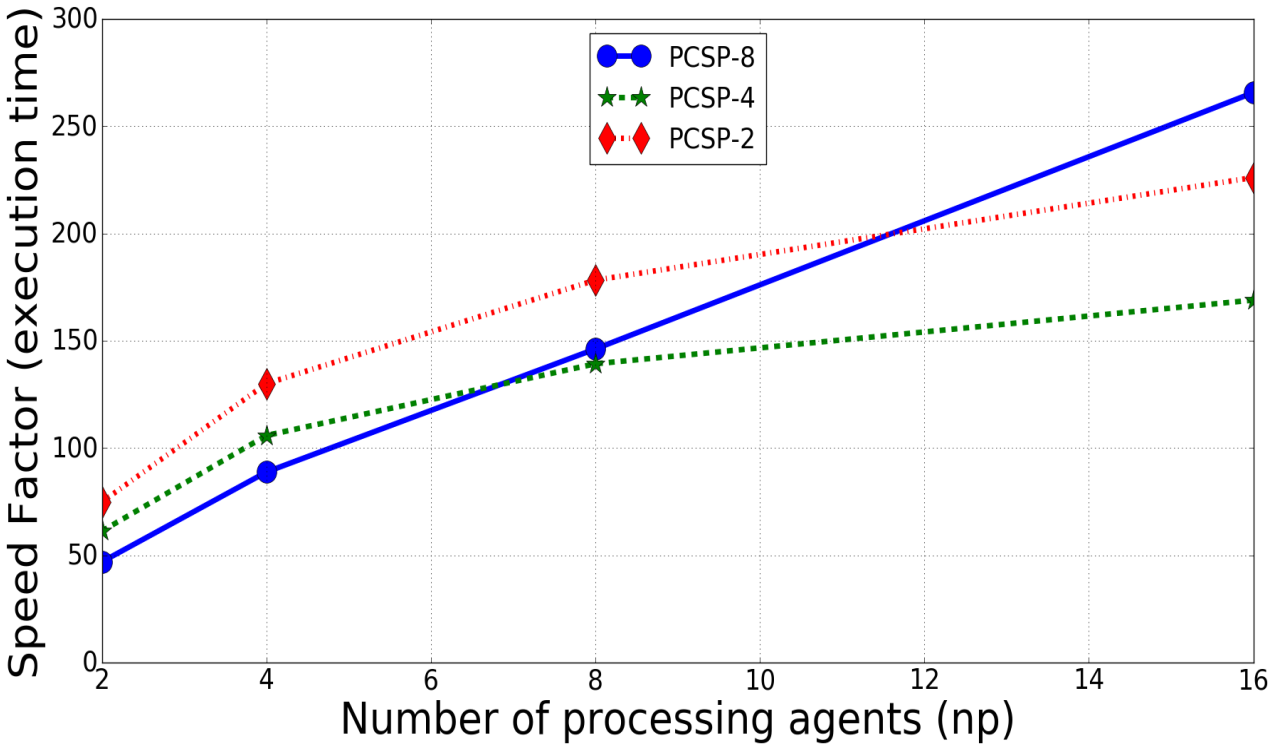
PCSP OVERCOMES ALL THE EVALUATED ALGORITHMS IN TERMS OF EXECUTION TIME



Speed factor with respect to the execution time of CS-RL with one StA and one PrA

imec

INCREASING BOTH n_p AND n_s SIMULTANEARLY ALLOWS A BETTER PERFORMANCE OF PCSP



Speed factor with respect to the execution time of CS-RL with one StA and one PrA

imec

PARALLEL Q-LEARNING FOR CONSTRAINED COMPUTING ENVIRONMENTS

- ✓ Q-Learning (QL): Strengths and weaknesses
- ✓ Traditional Parallel QL: the need of a new model for PQL
- ✓ PQL with co-allocation of processing and data
- ✓ Performance evaluation

CONCLUSIONS

imec

SUMMARY

Based on a flexible MA framework for developing efficient C-MAQL and PQL algorithms

PSCP minimizes the communication cost by combining a table partition strategy with a co-allocation of both processing and storage

PCSP can be deployed in both high performance (e.g. cloud data centers) and resource constrained devices (edge or fog computing devices)

FUTURE WORK: IMPROVE TABLE PARTITION AND ALLOCATION OF AGENTS

Design and implement partition algorithms that do not require any previous domain knowledge

Define new schemes for co-allocating agents different to pure load balancing of StA such as processing load, communication cost, etc.

Implement and evaluate the performance of other table-based RL algorithms.

embracing a better life