An Edge-Centric Ensemble Scheme for Queries Assignment

Kostas Kolomvatsos, Christos Anagnostopoulos

School of Computing Science,

University of Glasgow

8th International Workshop on Combinations of Intelligent Methods and Applications

in conjunction with

30th International Conference on Tools with Artificial Intelligence

November 5-7, 2018

Volos, Greece

Outline

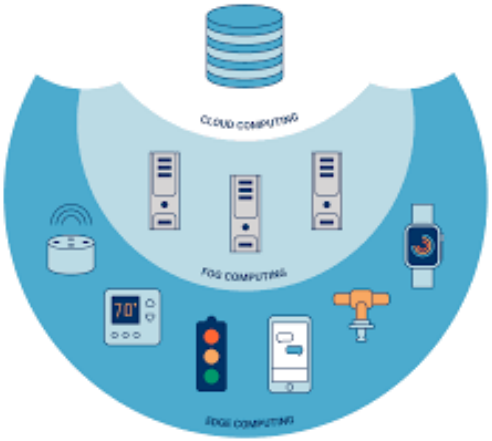
- Introduction
- Edge Nodes
- High Level Description
- Delivering the Complexity Class
- The Ensemble Scheme
- The Matching Process
- Experimental Evaluation
- Conclusions and Future Work

Introduction

- In the era of the Internet of Things (IoT), numerous devices form a vast infrastructure
- Devices can process tasks and exchange data
- Data can be processed at the devices, at the edge of the network (Edge/Fog) or at the Cloud

Edge Nodes

- Current research efforts focus on the data streams management at the edge
- Edge Nodes (ENs) act as distributed data repositories where queries can be executed
- ENs are responsible to report the results to the requested entity



Edge Nodes

- We deal with queries allocation to the appropriate ENs
- Queries are reported into a set of Query Controllers (QCs)
- It is a multi-dimensional problem involving queries and ENs characteristics

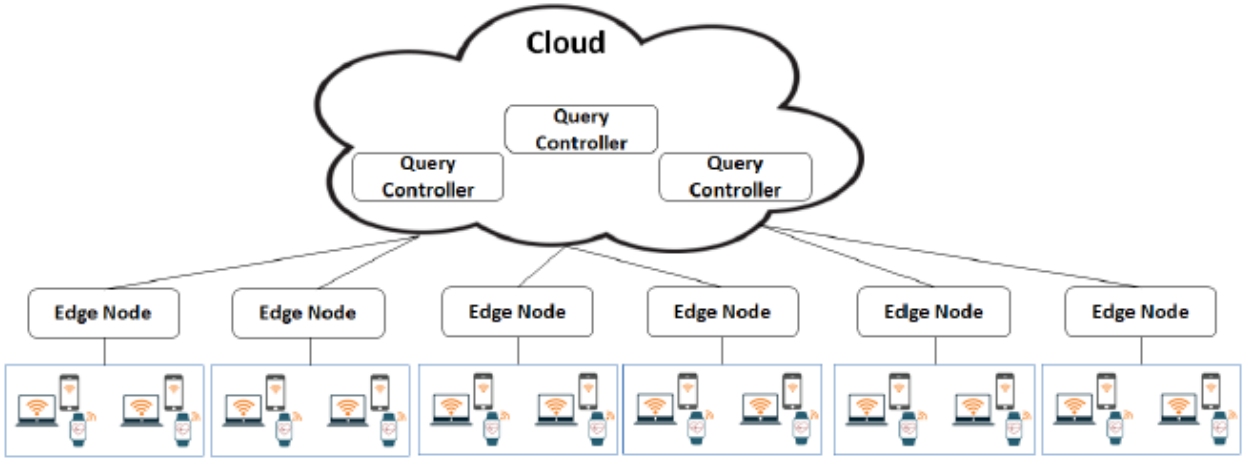
High Level Description

- Step 1. Classify queries into a set of complexity classes
- Step 2. Compare the requirements of queries with the ENs' load
- We propose models for both steps
  - An ensemble similarity scheme for the estimation of the complexity class
  - Decision for the selection of ENs based on the current and future load

High Level Description

- A Query Processor (QP) is adopted in every EN to respond to any incoming query

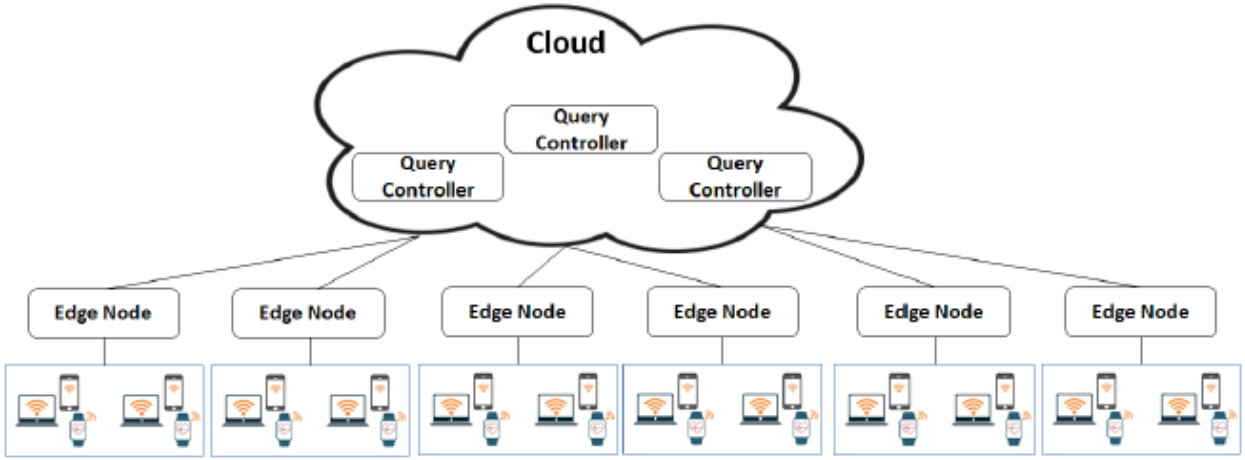
- QCs receive queries, `invoke' the appropriate QPs, get their responses and return the final result



High Level Description

- In each EN, a dataset is formulated i.e., a geodistributed local data repository

- Each dataset stores multivariate data



Matching Queries with Processors

- Every EN/QP exhibits specific characteristics
- We adopt:
  - The load
  - The speed
- Queries also have a set of characteristics
- We adopt:
  - The complexity
  - The need for instant response
- We focus on the query class; it depicts the complexity

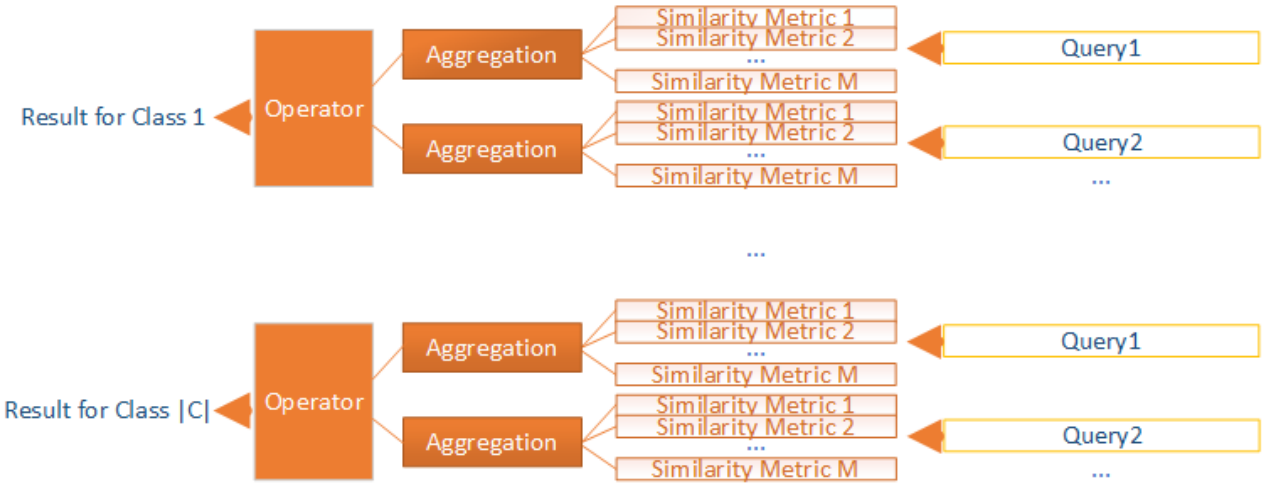
Delivering the Query Complexity

- For delivering the complexity class, we propose a `fuzzy' approach and define a Fuzzy Classification Process (FCP)
- The FCP derives the membership of a query in each of the pre-defined classes

- We also adopt a dataset of historical queries together with their corresponding classes
- The same class may be involved in multiple tuples, thus, in multiple queries

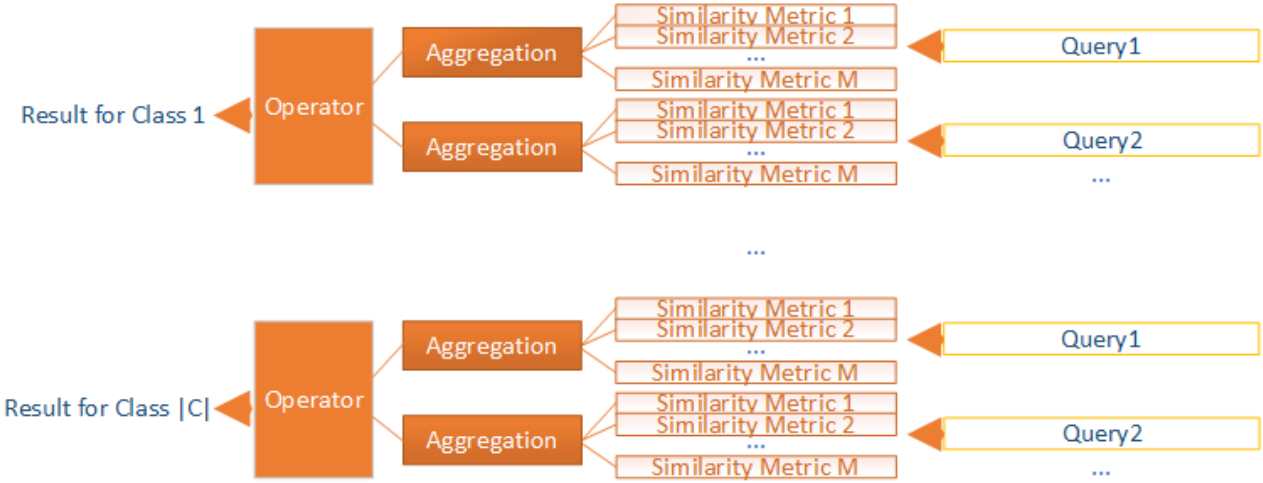
Delivering the Query Complexity

- We build on top of a function f
- f gets the query and delivers a similarity vector
- Example: q^s = <0.2, 0.8, 0.3>
- The ensemble scheme evaluates the final similarity between the query and every tuple in the training set



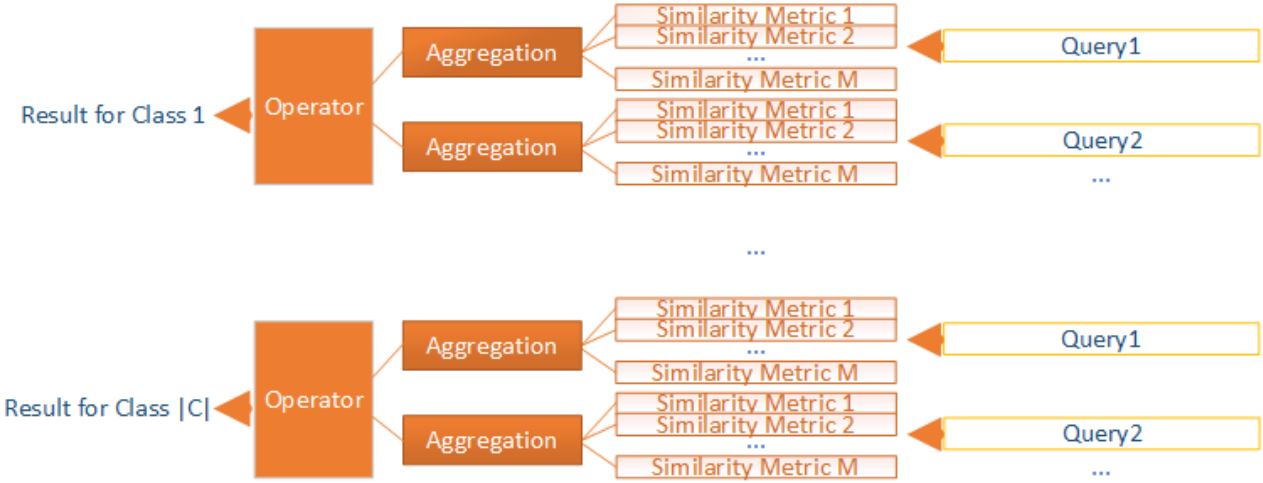
The Ensemble Scheme

- Similarity metrics are applied on each tuple classified into a class
- All the results are aggregated
- Every single result represents the membership of the query to a 'virtual' fuzzy set
- We adopt the Hamacher product for the final aggregation



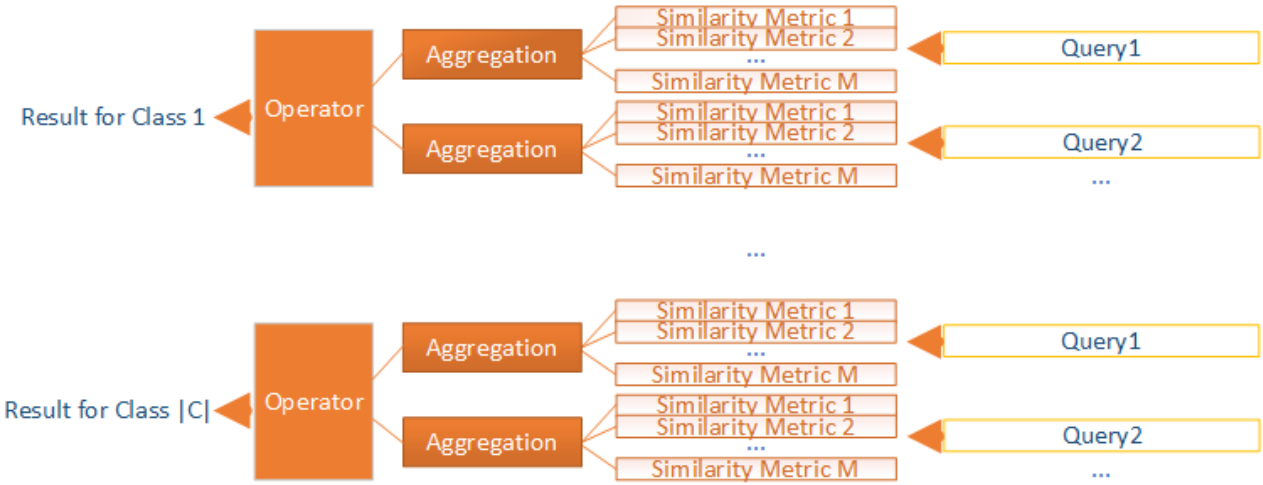
The Ensemble Scheme

- Disagreements are managed through the use of top-k similarity values based on their significance level
- The Significance Level (SL) depicts if a value is 'representative' for many other results
- Density based: Only values with a 'dense' neighborhood are considered



The Ensemble Scheme

- Over a set of aggregated similarity values for a class, we apply an operator
- We adopt the Quasi-Arithmetic mean for the second level of aggregation



The Matching Process

- We consider an additional vector containing steps for each complexity class
- The expected number of steps for a query is compared with the available load
- When the number of steps can be covered: reward
- Otherwise: penalty
- We process both, the current and the future load

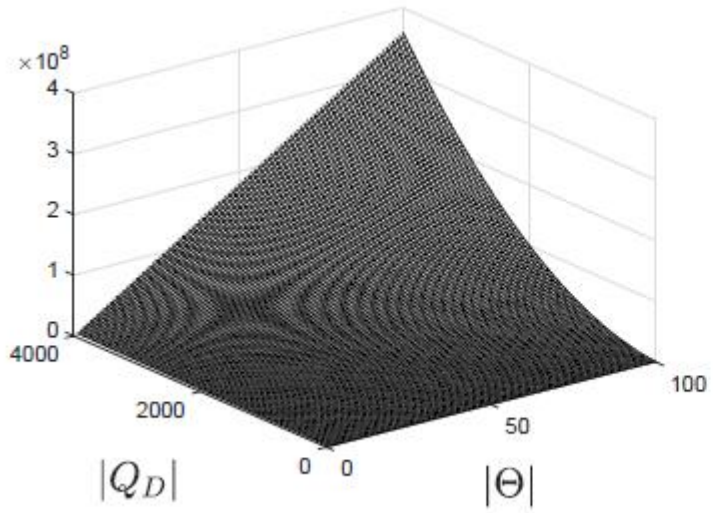
Experimental Evaluation

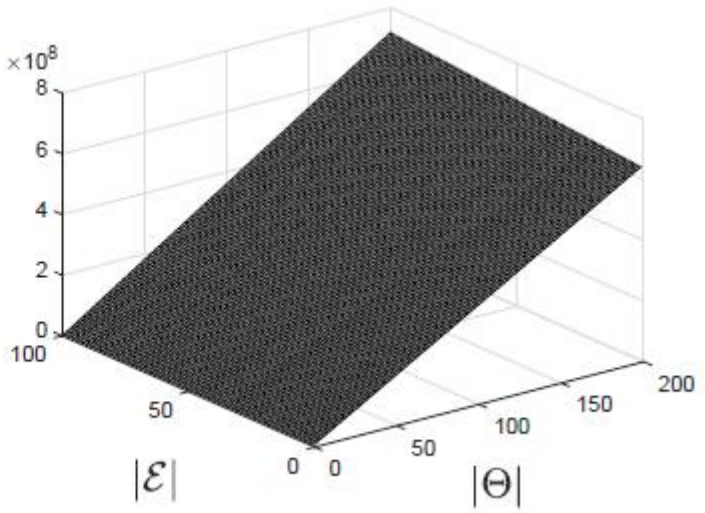
- Datasets
  - Queries found at http://www.tpc.org
  - For each, we define the complexity class (six classes)*
- Performance Metrics
  - Ψ: seconds to allocate a query
  - Ξ: difference of the selected load with the lowest
- Ties management
  - Scenario A: Random selection
  - Scenario B: The lowest load first

* Vashistha, A., Jain, S., 'Measuring Query Complexity in SQLShare Workload', Proc. of the Int. Conf. on Management of Data, 2016.

Experimental Evaluation

- Complexity of the scheme*





* $|Q_D|$: size of the training dataset, $|E|$: number of similarity metrics, $|\Theta|$: number of classes

Experimental Evaluation

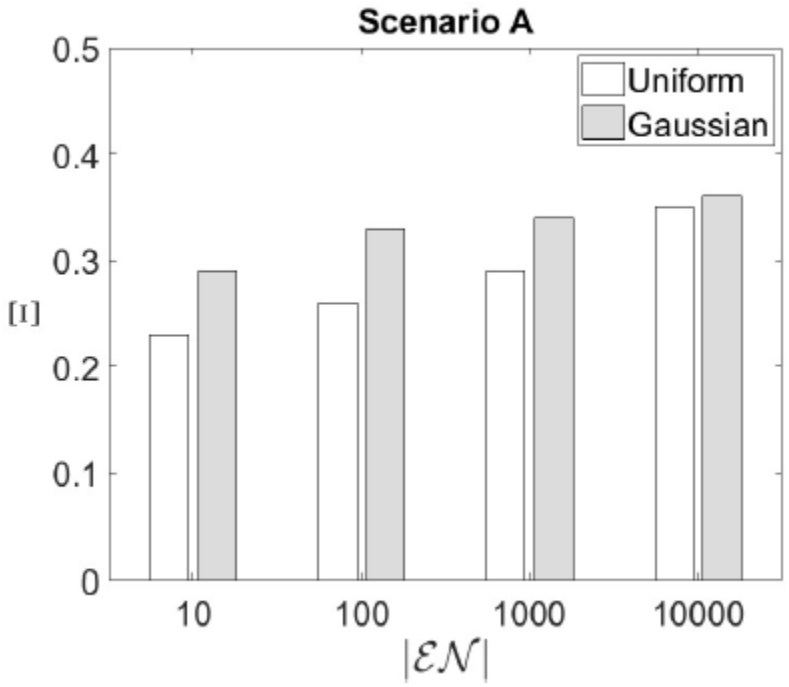
- Conclusion time (in seconds)

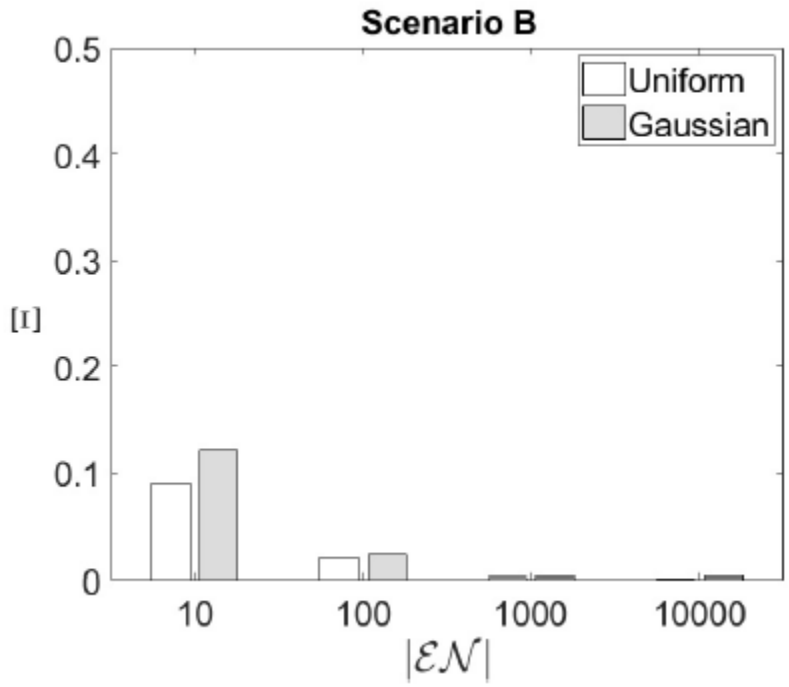
$\lvert\mathcal{EN}\rvert$	Ψ Uniform	Ψ Gaussian
10	0.008	0.008
100	0.012	0.010
1,000	0.055	0.370
10,000	0.251	0.276

* |EN|: number of nodes

Experimental Evaluation

- The load of the selected EN





Conclusions and Future Work

- The proposed model exhibits good performance
- We manage to perform efficient allocations

- Our future research plans involve the incorporation of more parameters
  - the deadline
  - the statistics of data
- The aim is to provide an adaptive mechanism

Thank You!!

Questions?