IDC 2018

15 – 17 October 2018 – Bilbao, Spain

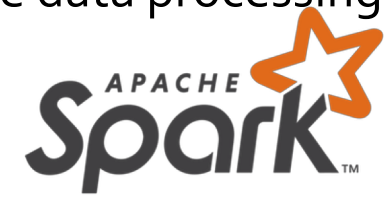
Multi-Objective Big Data Optimization with jMetalSP

Antonio J. Nebro
PhD in Computer Science
antonio@lcc.uma.es

Cristóbal Barba-González
PhD Student
cbarba@lcc.uma.es

About this tutorial

- jMetalSP is
  - A Java-based framework for multi-objective Big Data optimization with metaheuristics
  - It combines
    - jMetal: optimization engine (https://github.com/jMetal/jMetal)
    - Apache Spark: unified analytics engine for large-scale data processing (http://spark.apache.org/)
  - Open source project (MIT license)



- Web site
  - GitHub: https://github.com/jMetal/jMetalSP

About this tutorial

- Purpose of this tutorial
  - Offer a practical view of the jMetalSP framework
    - Description of its main components (algorithms, problems, streaming runtime, streaming data processing, algorithm data consumers)
    - Examples of dynamic versions of well-known algorithms (e.g., NSGA-II) and newer proposals (e.g., InDM2).
    - How to use jMetalSP to solve a dynamic optimization problem
  - Show how you can use jMetal to:
    - Configure and run dynamic multi-objective metaheuristics with and without Spark

Table of contents

- **Who, why, what, when**
- Background
- Working with jMetalSP
- Case study
- Further developments

Khaos INVESTIGACIÓN

Who, why, what, when

- Antonio J. Nebro
  - PhD in Computer Science (1999)
  - Associate professor (University of Málaga – Spain)
  - Khaos research group (http://khaos.uma.es/en)

- Research interests
  - Multi-objective optimization
  - Parallel metaheuristics
  - Applications to real-world problems (bioinformatics, civil engineering, Big Data)
  - Software tools (jMetal, jMetalSP, jMetalPy)

Khaos INVESTIGACIÓN

Who, why, what, when

- Cristóbal Barba González
  - PhD student
  - Researcher (University of Málaga – Spain)
  - Khaos research group (http://khaos.uma.es/)

- Research interests
  - Multi-objective Big Data optimization
  - Parallel metaheuristics
  - Applications to real-world problems (bioinformatics, civil engineering, Big Data)
  - Software tools (jMetal, jMetalSP)
  - Web Semantic

Khaos INVESTIGACIÓN

Who, why, what, when

- Khaos research group

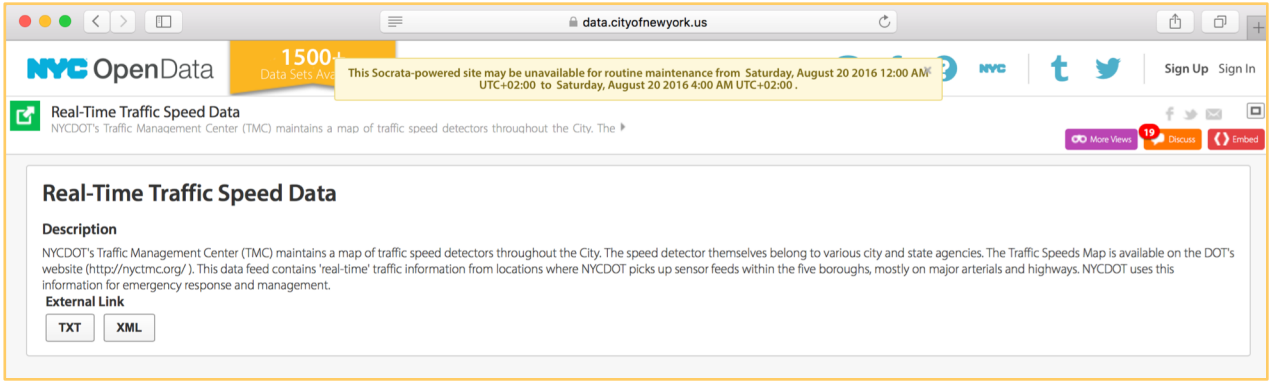
Who, why, what, when

- Research Lines:
  - Data Management and Integration (Relational Data Bases, NoSQL, Linked Data, Open Data)
  - Data Analysis (Data Mining, Test Mining, Information Recovery, Optimization Algorithms, Big Data Analytics)
  - Semantic Annotation. Integrated Data Analysis
  - Semantics (Scalable Reasoning, Semantic Relations Discovery, Semantic Assignation to Mobile Objects, Semantic Driven Contents Recommendation, Web Semantics for E-Sciences, Smart Data for Big Data interpretation)
  - **Metaheuristics. Multi-objective Optimization. Big Data Optimization**
- Applications (multi-disciplinary domains):
  - Life Sciences and Biomedicine
  - Cultural Patrimony and Tourism
  - E-commerce
  - Smart Cities

Khaos INVESTIGACIÓN

Who, why, what, when

- We have a lot of experience with jMetal
  - Java-based software for multi-objective optimization with metaheuristics
  - Proyect started in 2006
  - https://github.com/jMetal/jMetal
- We started to work with Apache Spark in 2015
  - Spark has a streaming engine
- Availability of open data -> real-time traffic speed data of New York



Who, why, what, when

- "It is of great interest to investigate the role of evolutionary multiobjective optimization (EMO) techniques for the optimization and learning involving big data, in particular, the ability of EMO techniques to solve dynamic multi-objective big data analytics problems."

Call for papers of Applied Soft Computing, Special Issue on Evolutionary Multi-objective Optimization and Applications in Big Data (2016)

jMetalSP: a framework for dynamic multi-objective big data optimization

Autores	Cristóbal Barba-González, José García-Nieto, Antonio J Nebro, José A Cordero, Juan J Durillo, Ismael Navas-Delgado, José F Aldana-Montes
Fecha de publicación	2018/8/1
Revista	Applied Soft Computing

Khaos INVESTIGACIÓN

Who, why, what, when

- What we where interested in:
  - Defining a dynamic bi-objective transportation problem using real-time traffic data of New York processed in streaming
  - Solving the problem with a dynamic evolutionary algorithm
- Our approach:
  - To design a software platform to deal with these kinds of problems: **jMetalSP**
    - The algorithms and problems are provided by jMetal
    - The streaming facilities are provided by Spark

Dynamic multi-objective optimization with jmetal and spark: a case study

Autores	José A Cordero, Antonio J Nebro, Cristóbal Barba-González, Juan J Durillo, José García-Nieto, Ismael Navas-Delgado, José F Aldana-Montes
Fecha de publicación	2016/8/26
Conferencia	International Workshop on Machine Learning, Optimization and Big Data

Khaos INVESTIGACIÓN

Who, why, what, when

- We started the project in 2016
- A redesign of the architecture was carried out in 2017

Design and architecture of the jMetalSP framework

Autores	Antonio J Nebro, Cristóbal Barba-González, José García Nieto, José A Cordero, José F Aldana Montes
Fecha de publicación	2017/7/15
Conferencia	Proceedings of the Genetic and Evolutionary Computation Conference Companion

- The jMetalSP project is continuously evolving and we are working actively on it

Table of contents

What is multi-objective optimization?

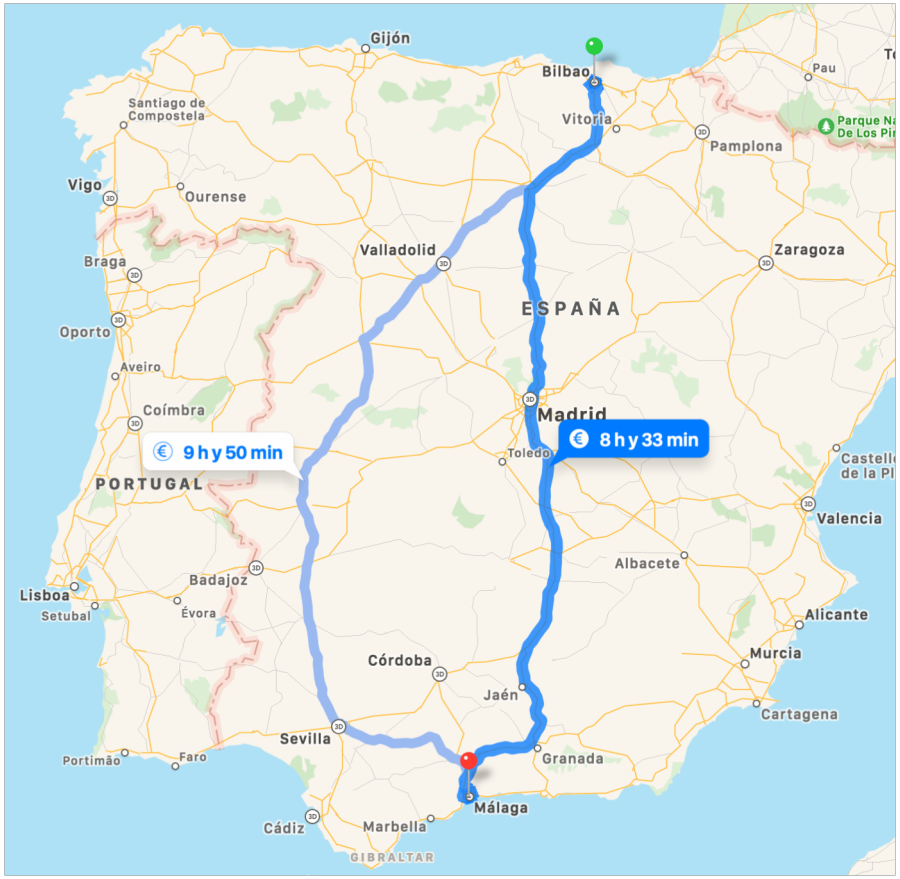
- Many real-world optimization problems require to optimize more than one objective or function at the same time
  - These objectives are usually in conflict among them
  - Improving one means worsening the others

- Multi-objective (or multi-criteria) optimization
  - Discipline focused in solving multiobjective optimization problems (MOPs)

Khaos INVESTIGACIÓN

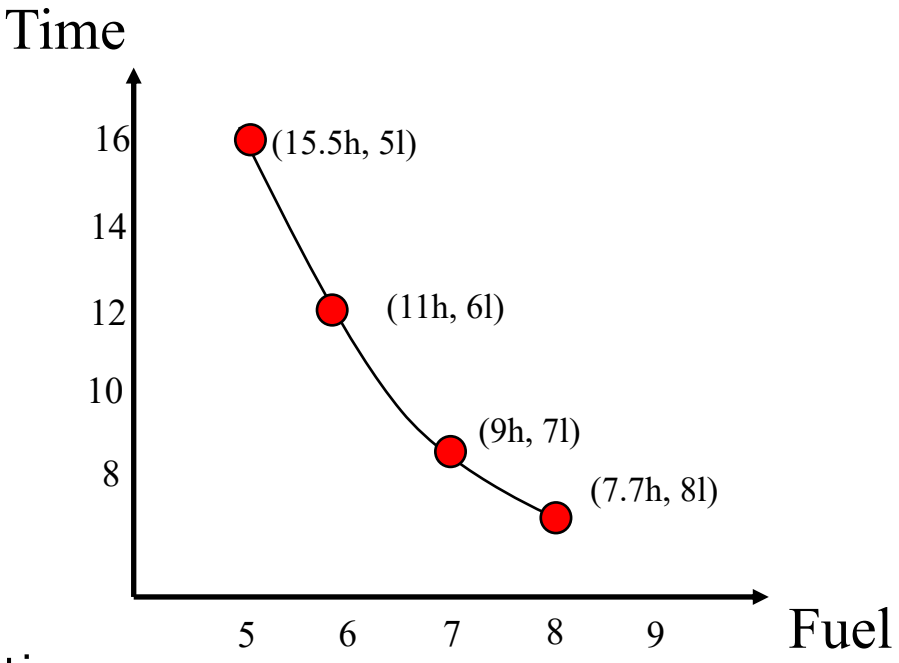
An example

- Example: travelling by car from Málaga to Bilbao (932 km)
  - Objective 1:
    - Minimizing time
  - Objective 2:
    - Minimizing fuel
  - Constraints:
    - Max. speed: 120 km/h
    - Min. speed: 60 km/h
  - Decision variable:
    - Mean car speed



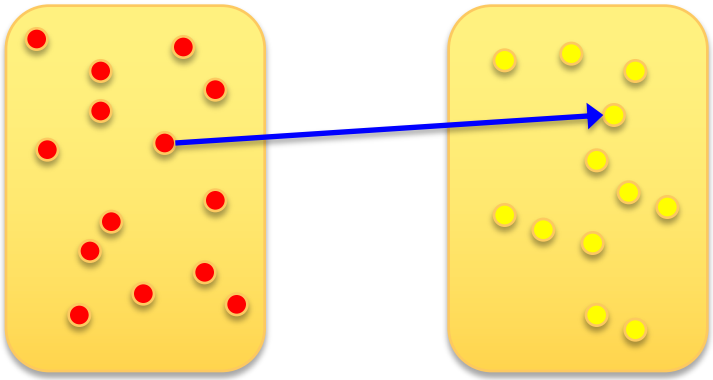
An example

- Example: travelling by car from Málaga to Bilbao (932 km)
  - We assume ideal conditions (i.e., we can keep a constant speed)
  - Extreme solutions
    - Time: 7.7 hours, fuel: 8 litres
    - Time: 15.5 hours, fuel: 5 litres
  - Other solutions
    - Time: 9 hours, fuel: 7 litres
    - Time: 12 hours, fuel: 6 litres
  - As velocity is a continuous variable
    - There are infinite compromise solutions



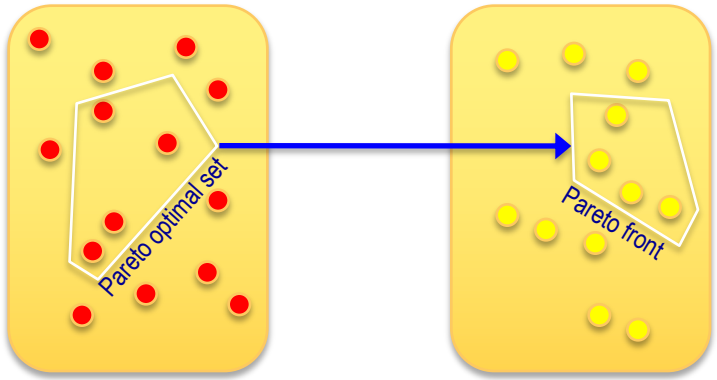
Single vs Multi-Objective Optimization

- In single-objective optimization (SO)
  - The optimum is a solution



X (Solution space) F(X) (Objective space)

- In multi-objective optimization (MO)
  - The optimum (Pareto optimal set) is a set of (non-dominated) solutions



X (Solution space) F(X), G(X), … Objective space

Concepts: Dominance and non-dominance

- In single-objective optimization
  - We look for a single solution
  - The concept of "A better than B" is trivial
- In multi-objective optimization
  - We are not restricted to find a unique optimal solution
  - the concept of "A better than B" is not trivial

A	2	3	4	5
B	4	6	5	7

A is better than B

A	3	7	4	8
B	2	1	2	5

B is better than A

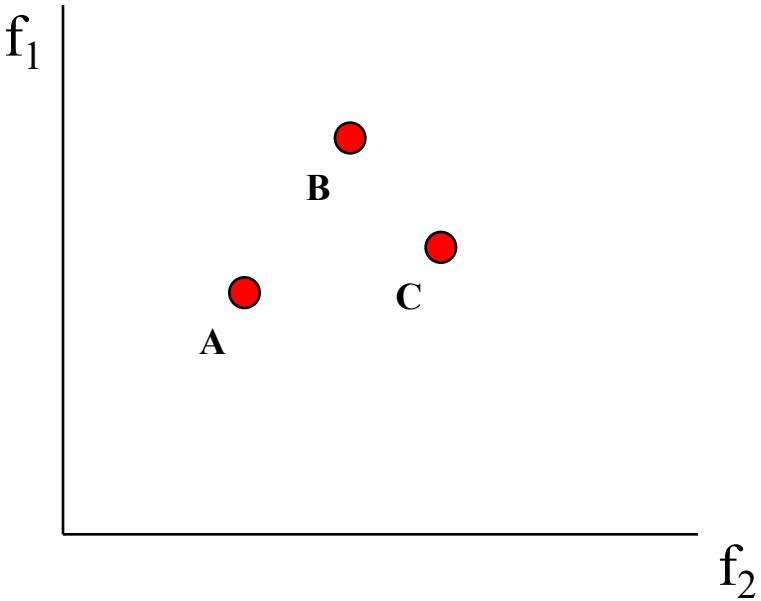
A	1	9	4	5
B	3	6	5	7

None is better

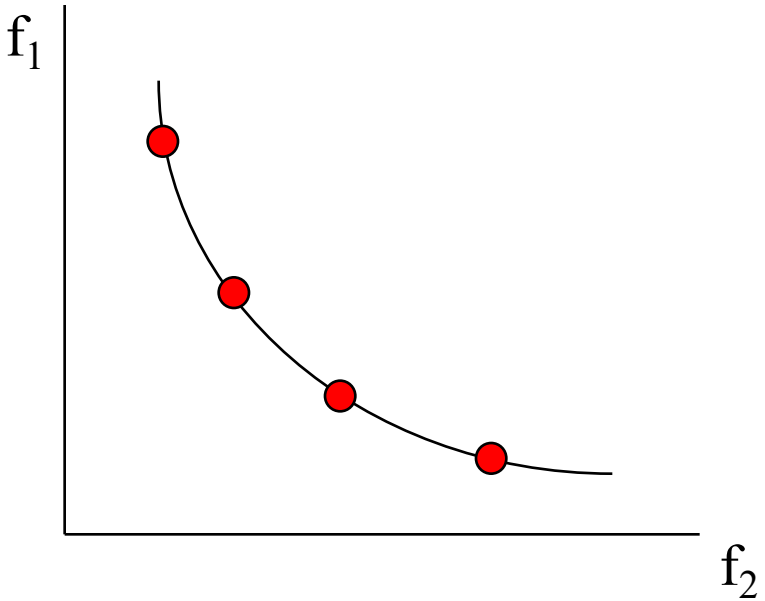
A and B are NON-DOMINATED

Concepts: Dominance and non-dominance

- Examples



A dominates B and C, B and are non-dominated

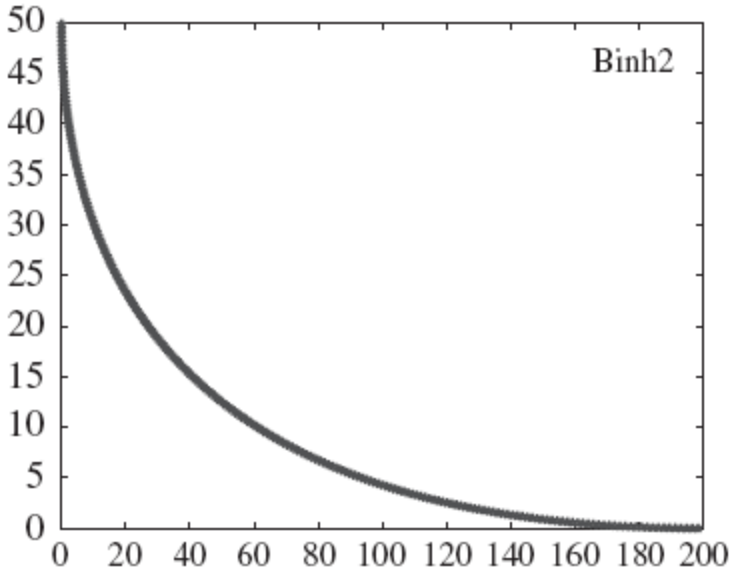


Non-dominated points

Concepts: Pareto optimal set, Pareto optimal front

- The solution of a MOP is a set composed of those non-dominated solutions that cannot be dominated by any other solution outside that set: Pareto Optimal Set
- The correspondence of the Pareto optimal set in the objective space is known as Pareto Optimal Front (or just Pareto Front)

$$
\begin{aligned}
&\text{Min } F = (f_1(\vec{x}), f_2(\vec{x}))\\
&f_1(\vec{x}) = 4x_1^2 + 4x_2\\
&f_2(\vec{x}) = (x_1 - 5)^2 + (x_2 - 5)^2\\
&\text{Subject to:}\\
&g_1(\vec{x}) = (x_1 - 5)^2 + x_2^2 - 25 \le 0\\
&g_2(\vec{x}) = -(x_1 - 8)^2 - (x_2 + 3)^2 + 7.7 \le 0\\
&0 \le x_1 \le 5\\
&0 \ge x_2 \le 3
\end{aligned}
$$

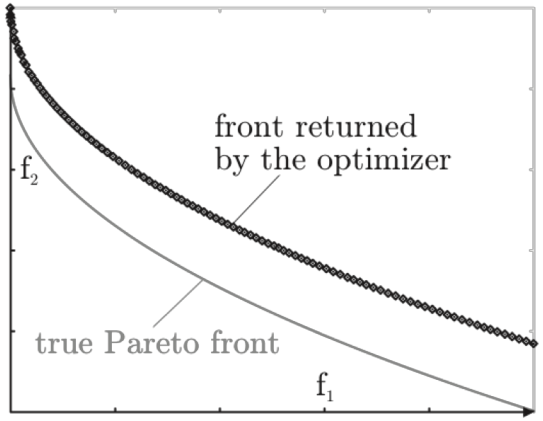


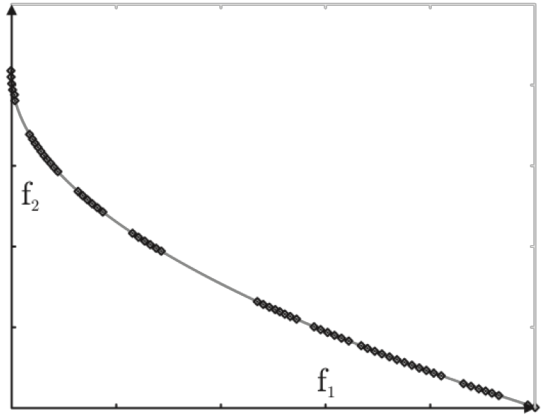
Goals of Multi-Objective Optimization

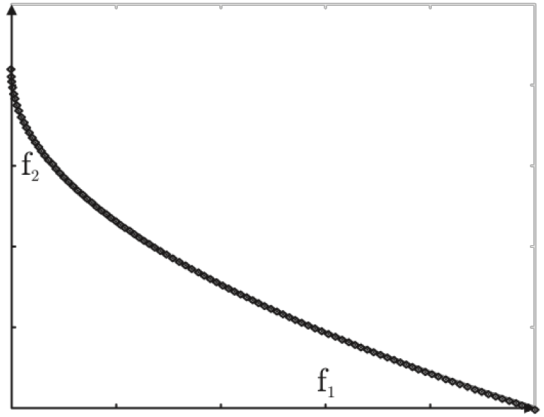
- The ideal goal is to obtain the Pareto front
- Unfortunately, this is unpractical in real-world problems
  - NP-hard complexity, non-linearity, epistasis, …
  - Frequently, exact techniques are not useful
- Alternative: Use non-exact algorithms
  - E.g. *Metaheuristics*
  - These techniques provide an *approximation* to the Pareto front

Goals of Multi-Objective Optimization

- Main goal when using non-exact tecniques:
  - Obtaining an approximation to the Pareto from with two properties
    - Convergence
    - Diversity

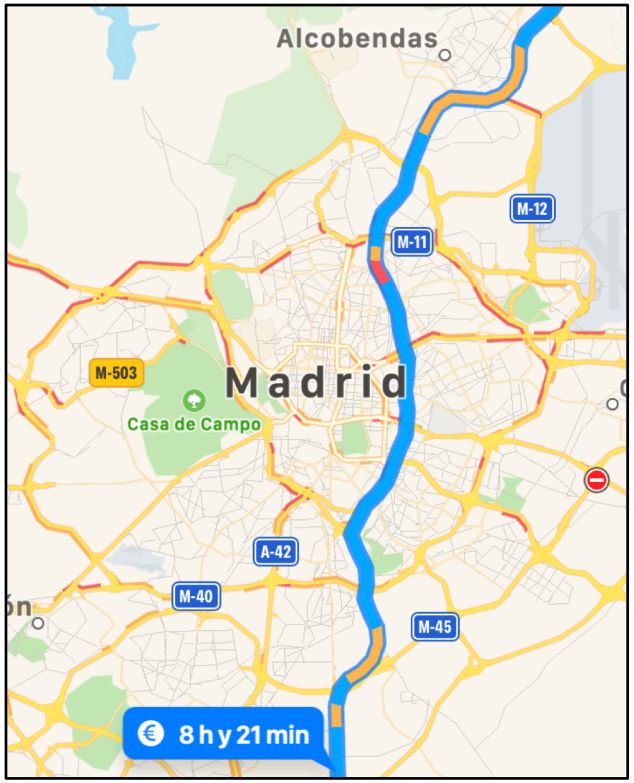






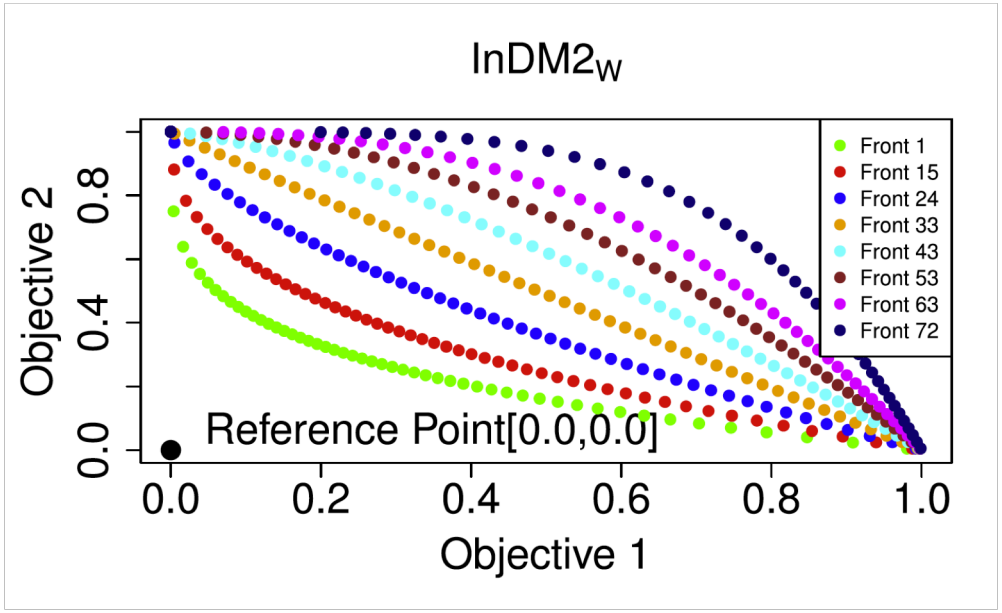
Dynamic Multi-Objective Optimization Problems

- Example: travelling by car from Málaga to Bilbao (932 km)
  - What happen if we take into consideration real-traffic information?
  - Sources:
    - Cameras
    - Sensor data
    - Web services
  - The problem data
    - Change with time
  - The optimization problem
    - Becomes **dynamic**



Dynamic Multi-Objective Optimization Problems

- The changes can affect:
  - The Pareto set
  - The Pareto front
  - Both
  - None of them, but the problem formulation can change



Metaheuristics

- Components (Greek terms):
  - Meta: "higuer level"
  - Heuristic: "to find", generic method to solve a problem
- Metaheuristic:
  - *High-level strategy that combines a set of underlying simpler operation techniques (usually heuristics) aimed at obtaining a more powerful procedure*
- They have become very popular optimization techniques
  - Evolutionary algorithms
  - Particle swarm optimization
  - Ant colony optimization
  - …

Metaheuristics

- Pseudo-codes

Algorithm 1 Template of a metaheuristic

```
1: A(0) ← GenerateInitialSolutions()
2: t ← 0
3: Evaluate(A(0))
4: while not StoppingCriterion( ) do
5:     S(t) ← Generation(A(t))
6:     Evaluate(S(t))
7:     A(t + 1) ← Update(A(t), S(t))
8:     t ← t + 1
9: end while
```

Algorithm 2 Template of an Evolutionary Algorithm (EA) algorithm.

```
1: P(0) ← GenerateInitialPopulation()
2: t ← 0
3: Evaluate(P(0))
4: while not StoppingCriterion( ) do
5:     P′(t) ← Selection(P(t))
6:     P″(t) ← Reproduction(P′(t))
7:     Evaluate(P″(t))
8:     P(t + 1) ← Replacement(P(t), P″(t))
9:     t ← t + 1
10: end while
```

Dynamic Metaheuristics

- Adopted approach
  - If a change in the problem is detected the algorithm must do a restart
  - Restart strategies
    - Some solutions can be removed
    - The deleted solutions must be replaced by new ones

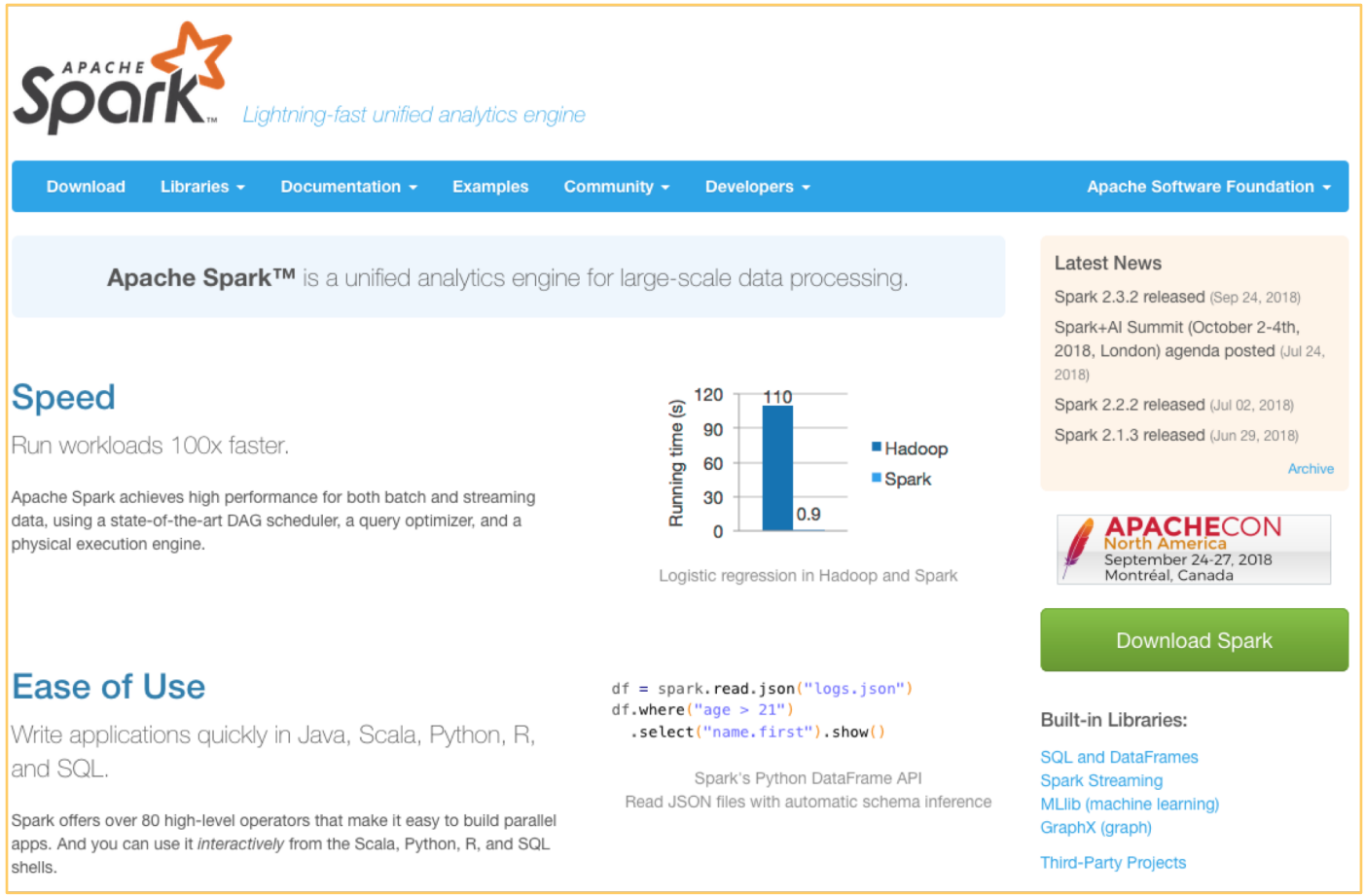
Algorithm 1 Template of a metaheuristic

1: $A(0) \leftarrow$ GenerateInitialSolutions()
2: $t \leftarrow 0$
3: Evaluate($A(0)$)
4: **while not** StoppingCriterion( ) **do**
5: $S(t) \leftarrow$ Generation($A(t)$)
6: Evaluate($S(t)$)
7: $A(t+1) \leftarrow$ Update($A(t)$, $S(t)$)
8: $t \leftarrow t+1$
9: **end while**

Change detection →

Apache Spark

- Spark is a popular technology in Big Data applications

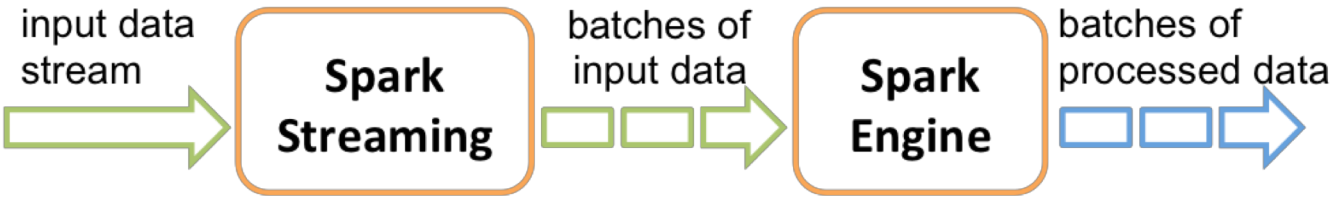


http://spark.apache.org/

Apache Spark

- Spark includes two streaming engines:
  - RDD-based (DStreams)
  - Structured streaming (dataframe-based API)

- DStreams (RDD-based API)
  - Streaming data is processed as micro-batches



Khaos INVESTIGACIÓN

Table of contents

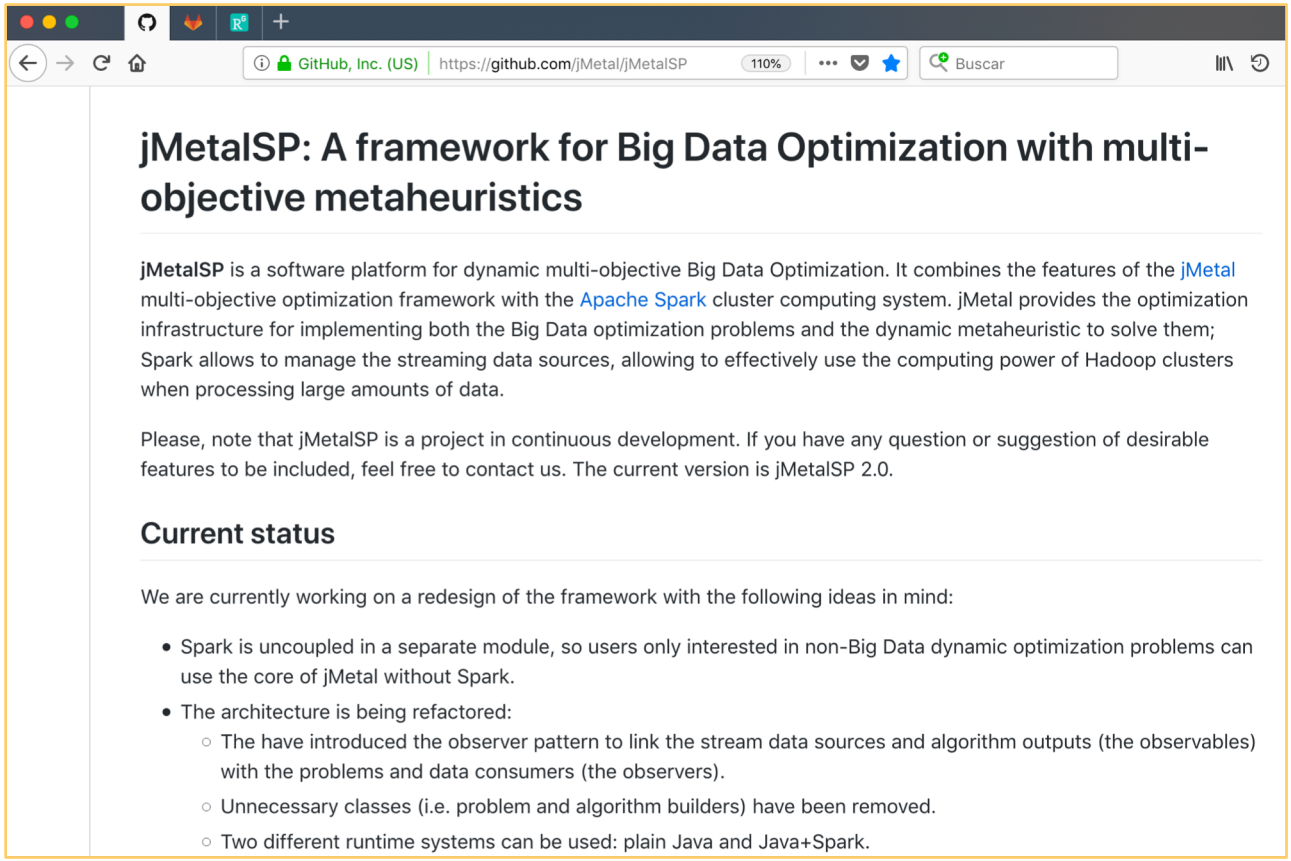
- Who, why, what, when
- Background
- **Working with jMetalSP**
- Case study
- Further developments

Software requirements

- Requirements to run jMetal
  - Java JDK 8+
  - Maven
  - An IDE (Intellij Idea, Eclipse, Netbeans)

Khaos INVESTIGACIÓN

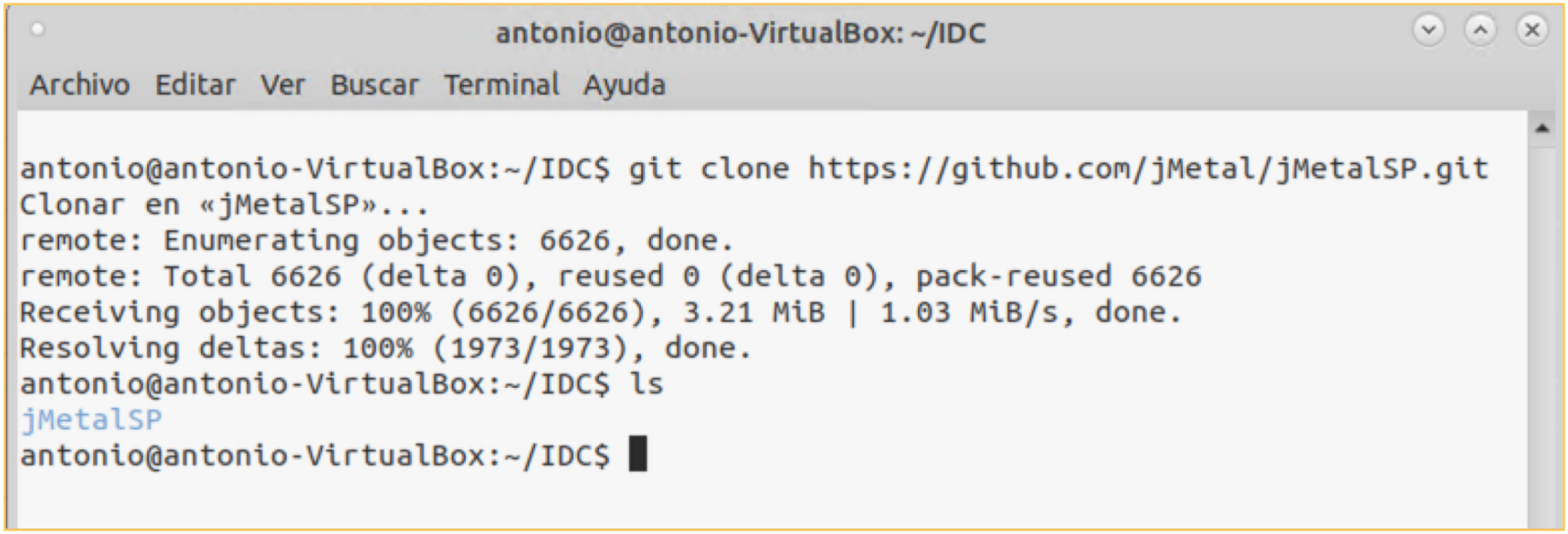
jMetalSP: a Maven Project Hosted in GitHub



https://github.com/jMetal/jMetalSP

jMetalSP: a Maven Project Hosted in GitHub

- Getting the full source code of the project

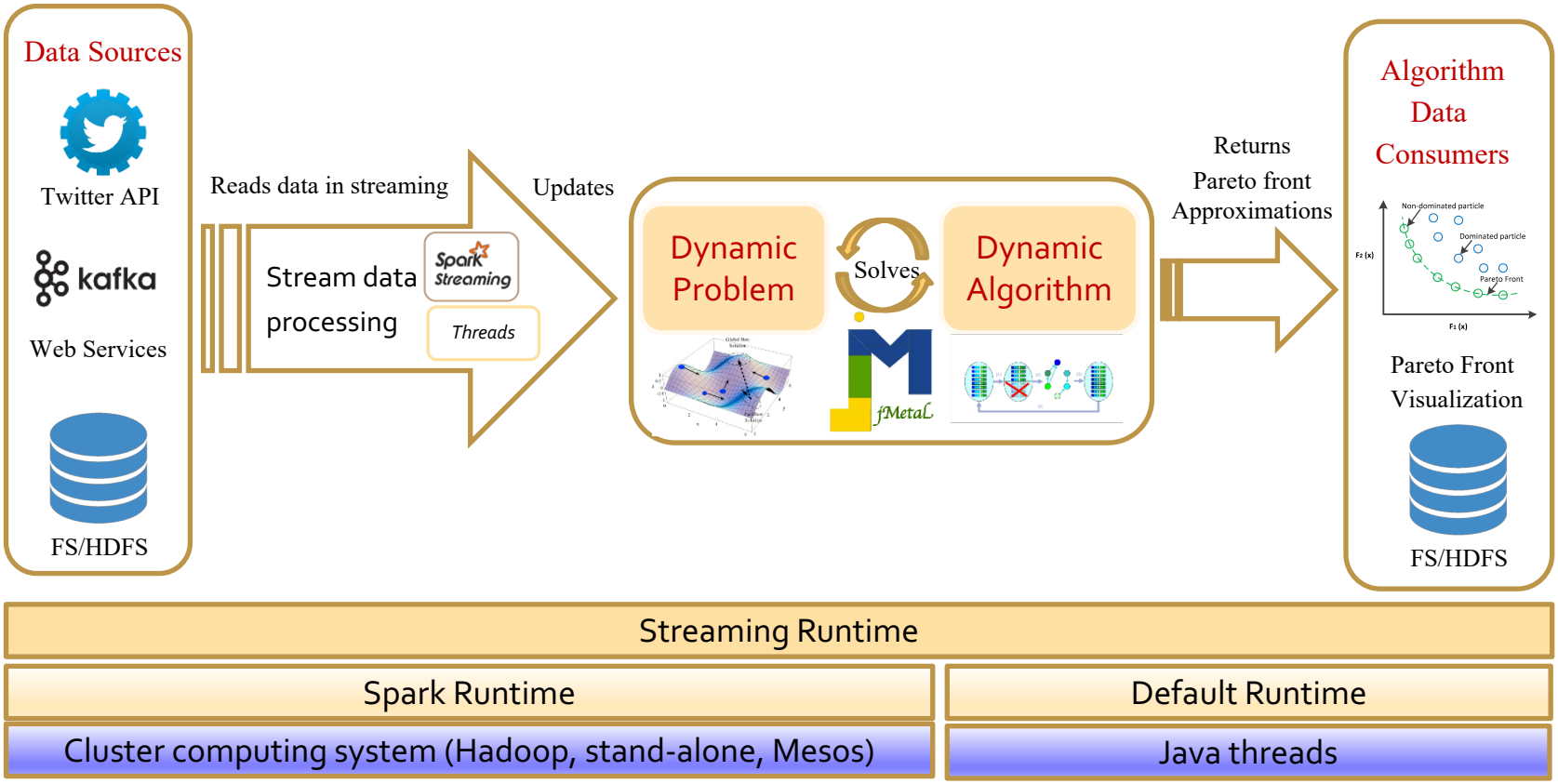


jMetalSP: a Maven Project Hosted in GitHub

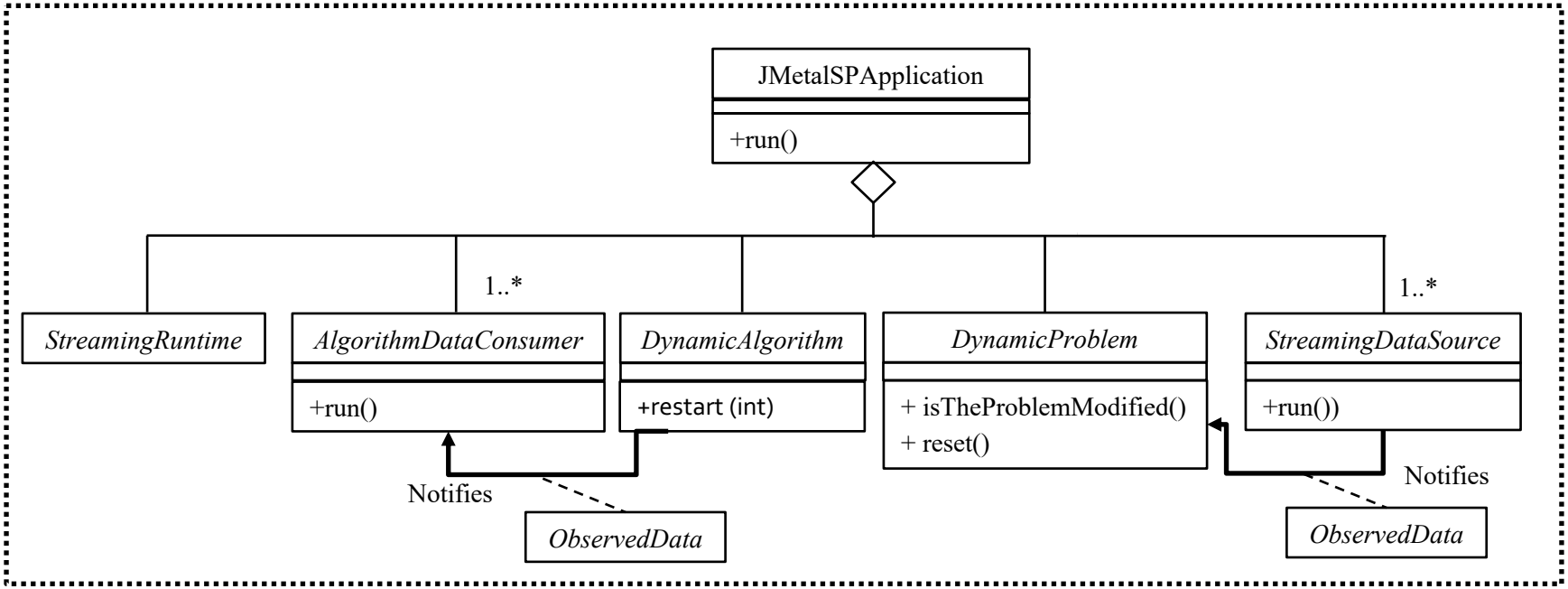
- The project can be built from the command line using Maven

```
antonio@antonio-VirtualBox: ~/IDC/jMetalSP
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
antonio@antonio-VirtualBox:~/IDC/jMetalSP$ mvn clean package
[INFO] Scanning for projects...
[INFO] Inspecting build with total of 10 modules...
[INFO] Installing Nexus Staging features:
[INFO]   ... total of 10 executions of maven-deploy-plugin replaced with nexus-staging-maven-plugin
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Build Order:
[INFO]
[INFO] jmetalsp
[INFO] org.uma.jmetalsp:jmetalsp-core
[INFO] org.uma.jmetalsp:jmetalsp-observeddata
[INFO] org.uma.jmetalsp:jmetalsp-algorithm
[INFO] org.uma.jmetalsp:jmetalsp-consumer
[INFO] org.uma.jmetalsp:jmetalsp-spark
[INFO] org.uma.jmetalsp:jmetalsp-problem
[INFO] org.uma.jmetalsp:jmetalsp-streamingdatasource
[INFO] org.uma.jmetalsp:jmetalsp-examples
[INFO] org.uma.jmetalsp:jmetalsp-externalsource
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building jmetalsp 2.1-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ jmetalsp ---
[INFO]
[INFO] --- maven-source-plugin:2.2.1:jar-no-fork (attach-sources) @ jmetalsp ---
[INFO]
[INFO] --- maven-javadoc-plugin:2.9.1:jar (attach-javadocs) @ jmetalsp ---
[INFO] Not executing Javadoc as the project is not a Java classpath-capable package
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building org.uma.jmetalsp:jmetalsp-core 2.1-SNAPSHOT
[INFO] ------------------------------------------------------------------------
```

jMetalSP (2.0) Architecture



jMetalSP (2.0) UML class diagram

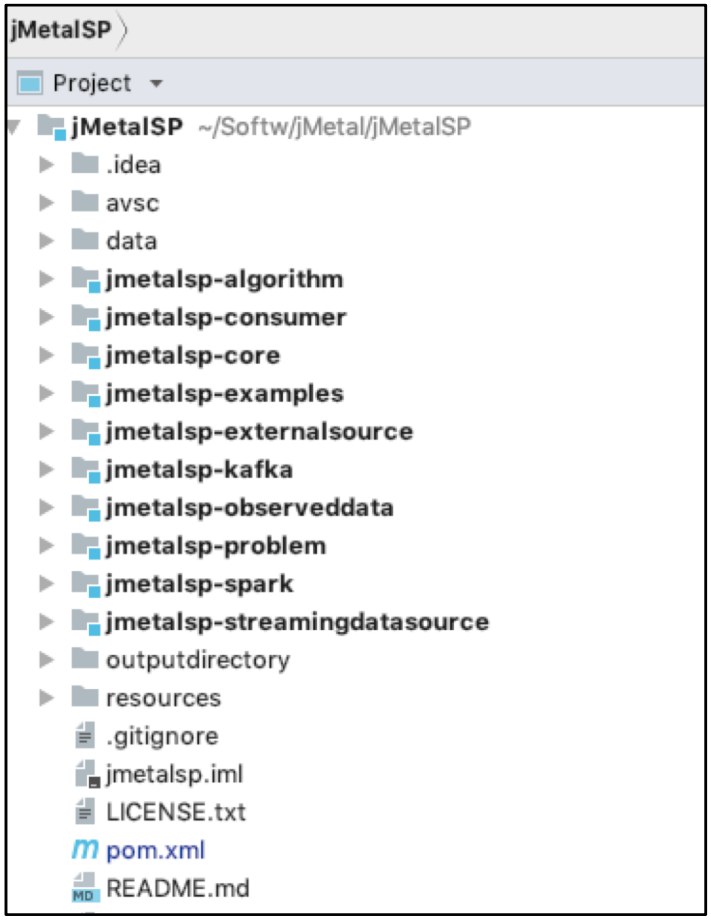



```
// Declaring application components
…
JMetalSPApplication<> application = new JMetalSPApplication<>() ;

application.setStreamingRuntime(new SparkRuntime())
     .setProblem(problem)
     .setAlgorithm(algorithm)
     .addStreamingDataSource(streamingDataSource1, problem)
     .addStreamingDataSource(streamingDataSource2, problem)
     .addAlgorithmDataConsumer(dataConsumer1)
     .addAlgorithmDataConsumer(dataConsumer2)
     .run();
```

jMetalSP: a Maven Project Hosted in GitHub

- jMetalSP is composed of a number of 10 Maven sub-packages
- In the current version (jMetalSP 2.0):
  - Multi-Objective algorithms (7)
  - Benchmark problems (5)
  - Streaming runtimes (2)

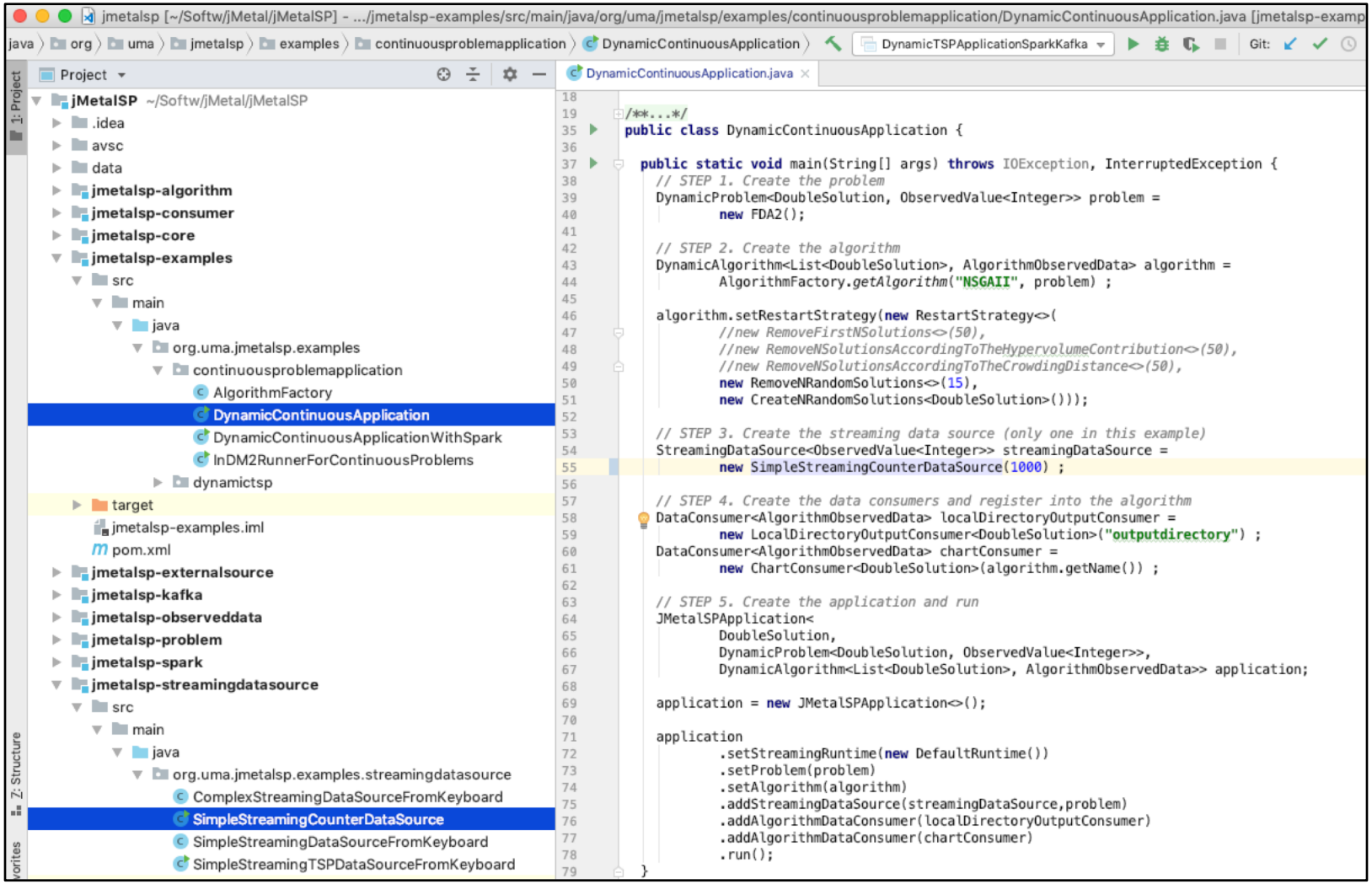


Algorithms included

- Dynamic versions of
  - NSGA-II
  - NSGA-III
  - MOCell
  - SMPSO
  - WASF-GA (Decision making)
  - R-NSGA-II (Decision making)
  - InDM2 (Decision making, interactive)

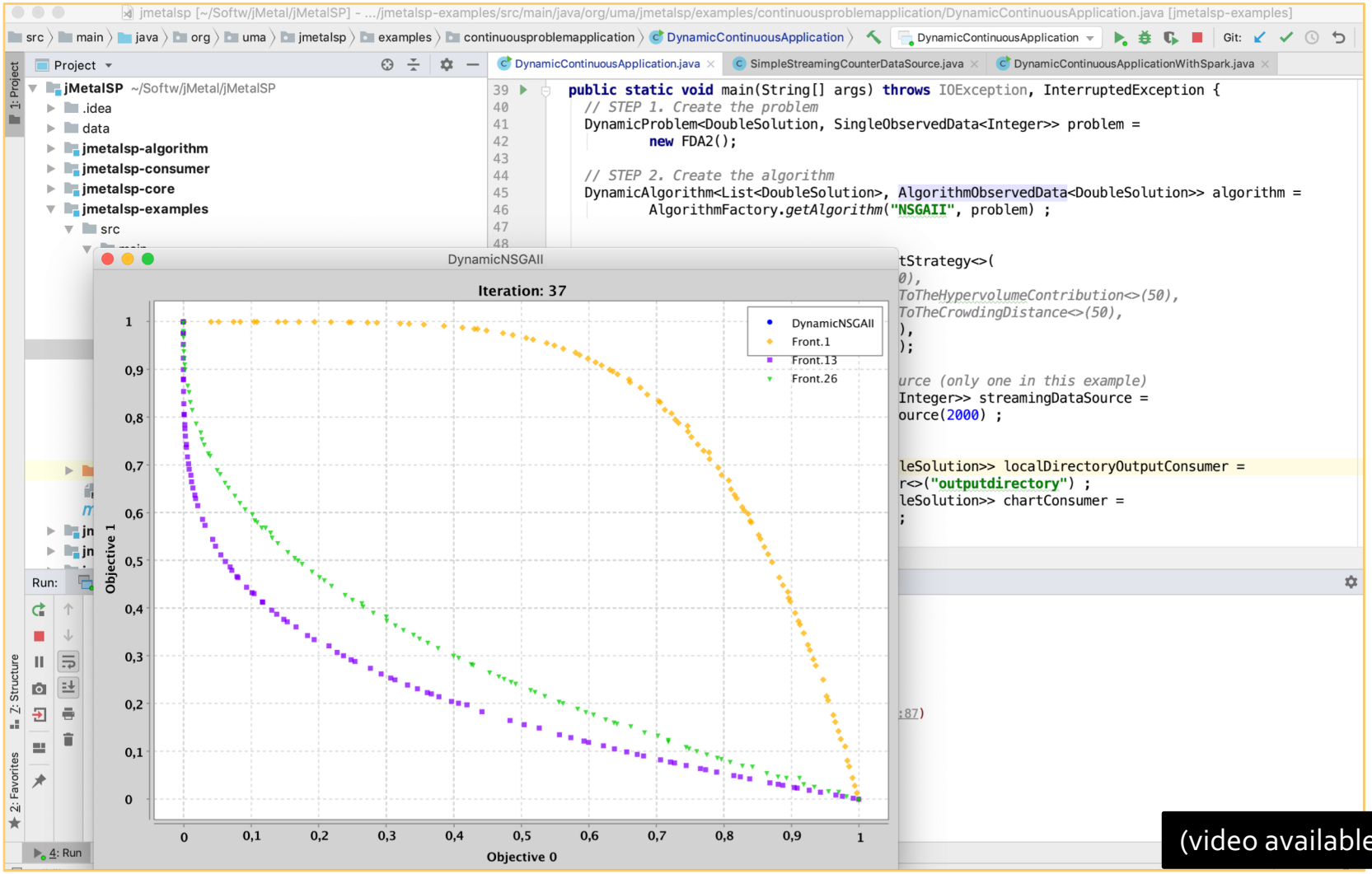
Khaos INVESTIGACIÓN

Example 1: DNSGA-II, FDA2, Default Runtime



Khaos INVESTIGACIÓN

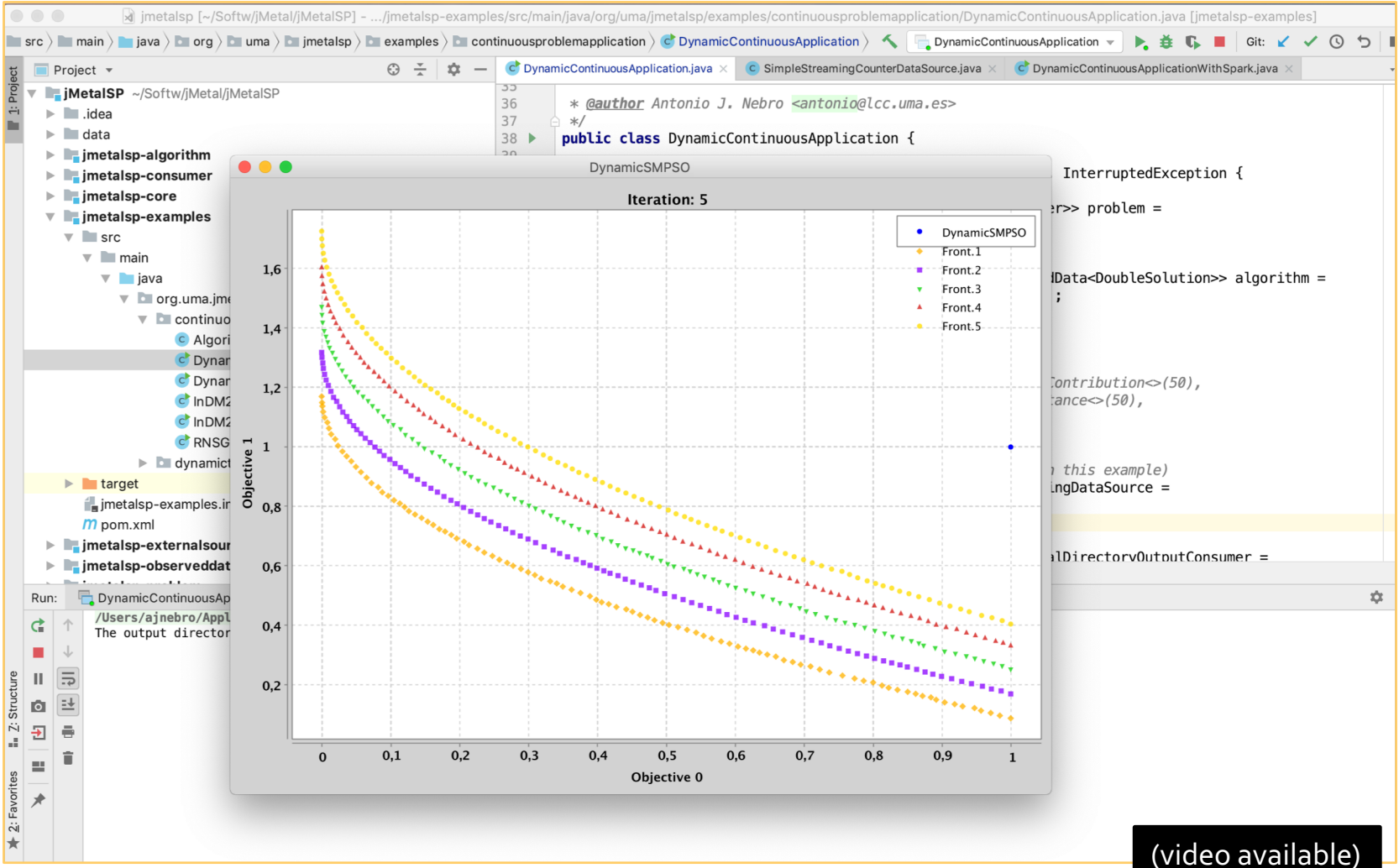
Example 1: DNSGA-II, FDA2, Default Runtime



(video available)

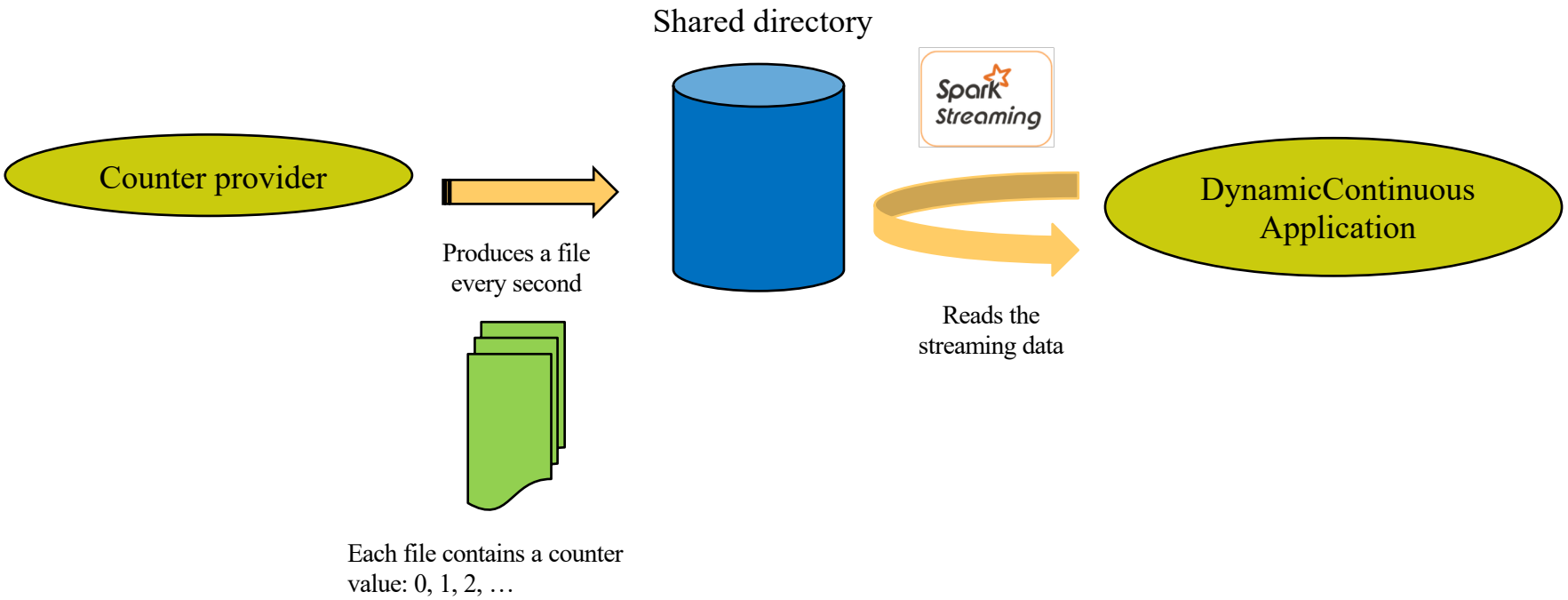
Khaos INVESTIGACIÓN

Example 2: DSMPSO, FDA3, Default Runtime



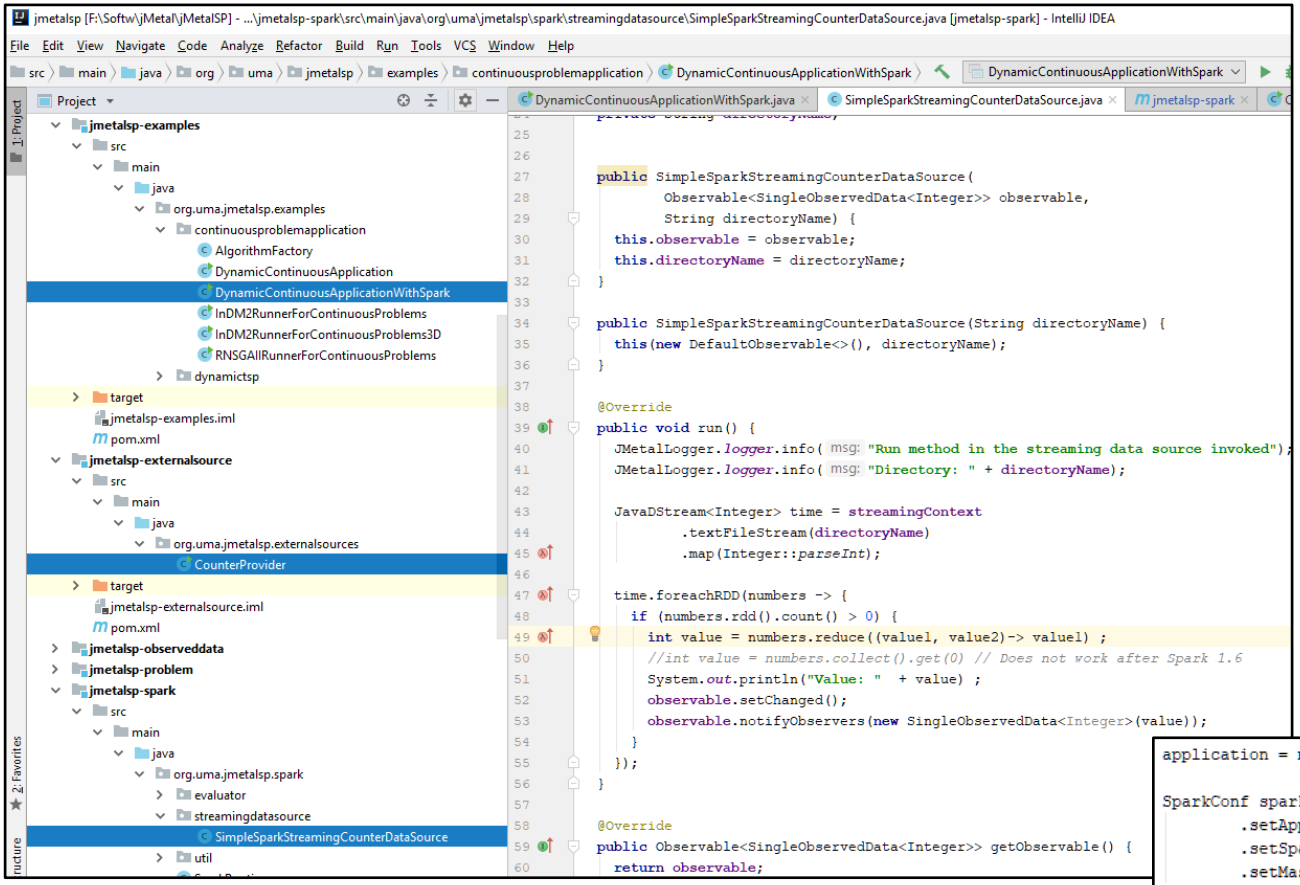
(video available)

Example 3: DSMPSO, FDA2, Spark Runtime



Khaos INVESTIGACIÓN

Example 3: DSMPSO, FDA2, Spark Runtime



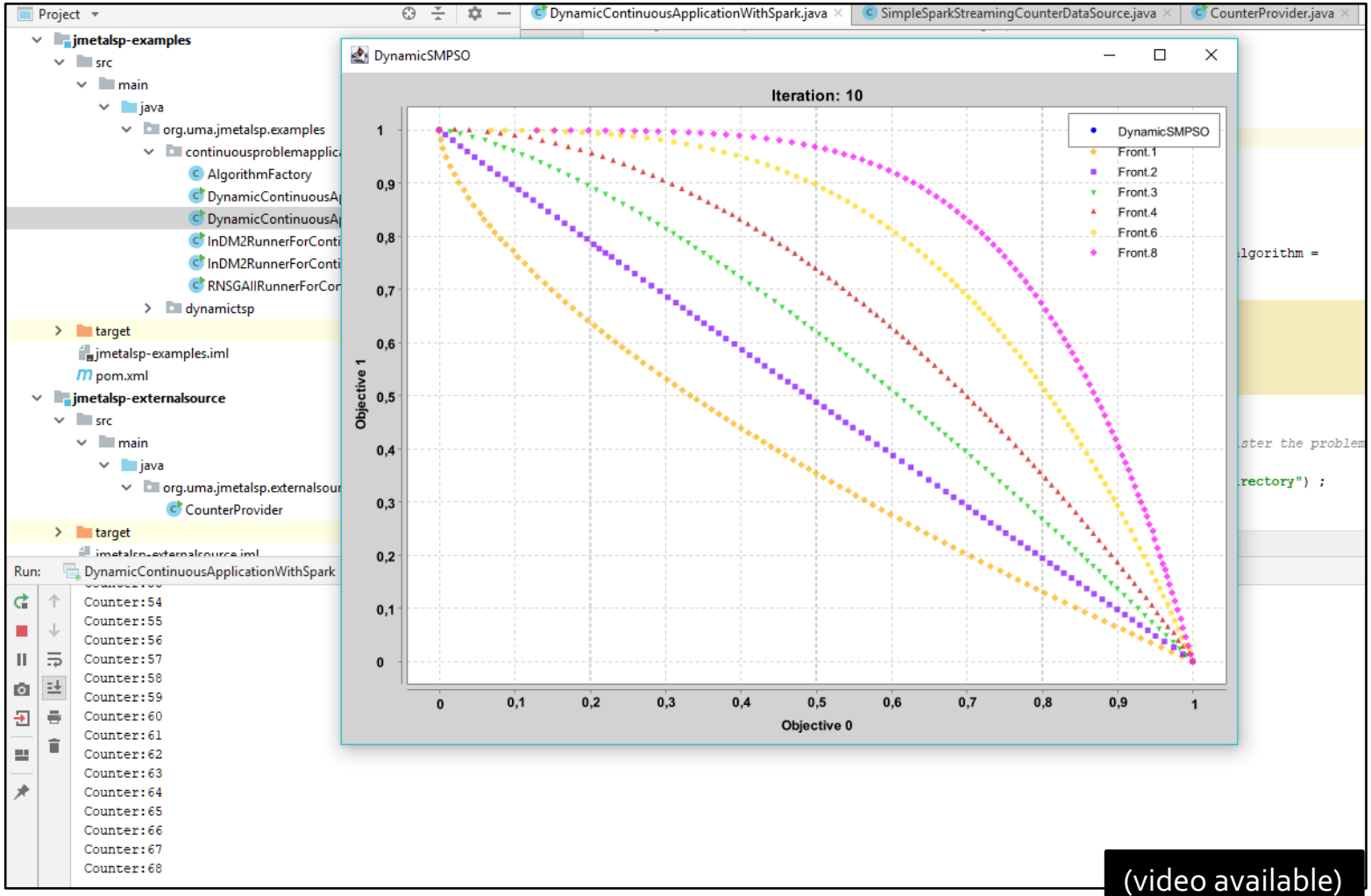

```
application = new JMetalSPApplication<>();

SparkConf sparkConf = new SparkConf()
        .setAppName("SparkApp")
        .setSparkHome("F:\\spark-2.3.1-bin-hadoop2.7")
        .setMaster("local[4]") ;

application.setStreamingRuntime(new SparkRuntime( duration: 1, sparkConf))
        .setProblem(problem)
        .setAlgorithm(algorithm)
        .addStreamingDataSource(streamingDataSource,problem)
        .addAlgorithmDataConsumer(localDirectoryOutputConsumer)
        .addAlgorithmDataConsumer(chartConsumer)
        .run();
```

Khaos INVESTIGACIÓN

Example 3: DSMPSO, FDA2, Spark Runtime

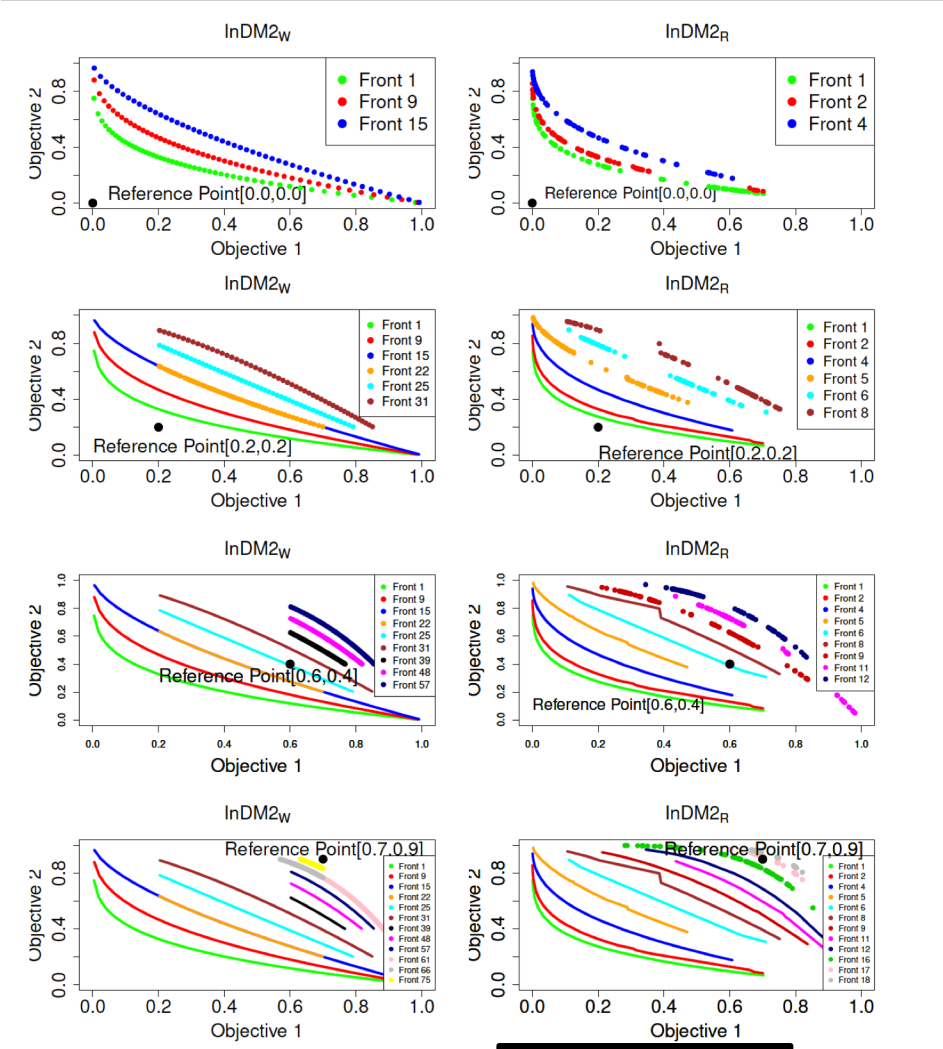


(video available)

Khaos INVESTIGACIÓN

Example 4: InDM2, FDA2, Default runtime

- InDM2
  - Interactive **D**ynamic **M**ulti-objective **D**ecision **M**aking algorithm
- Intended for
  - Large executions
  - Guide the search to the desired region



(video available)

A.J. Nebro, A.B. Ruíz, C. Barba-González, J. García Nieto, M.M. Roldán-García, M. Luque, J.F. Aldana Montes. InDM2: Interactive Dynamic Multi-Objective Decision Making Using Evolutionary Algorithms. Swarm and Evolutionary Computation, Vol. 40, pp: 184:195. June 2018.

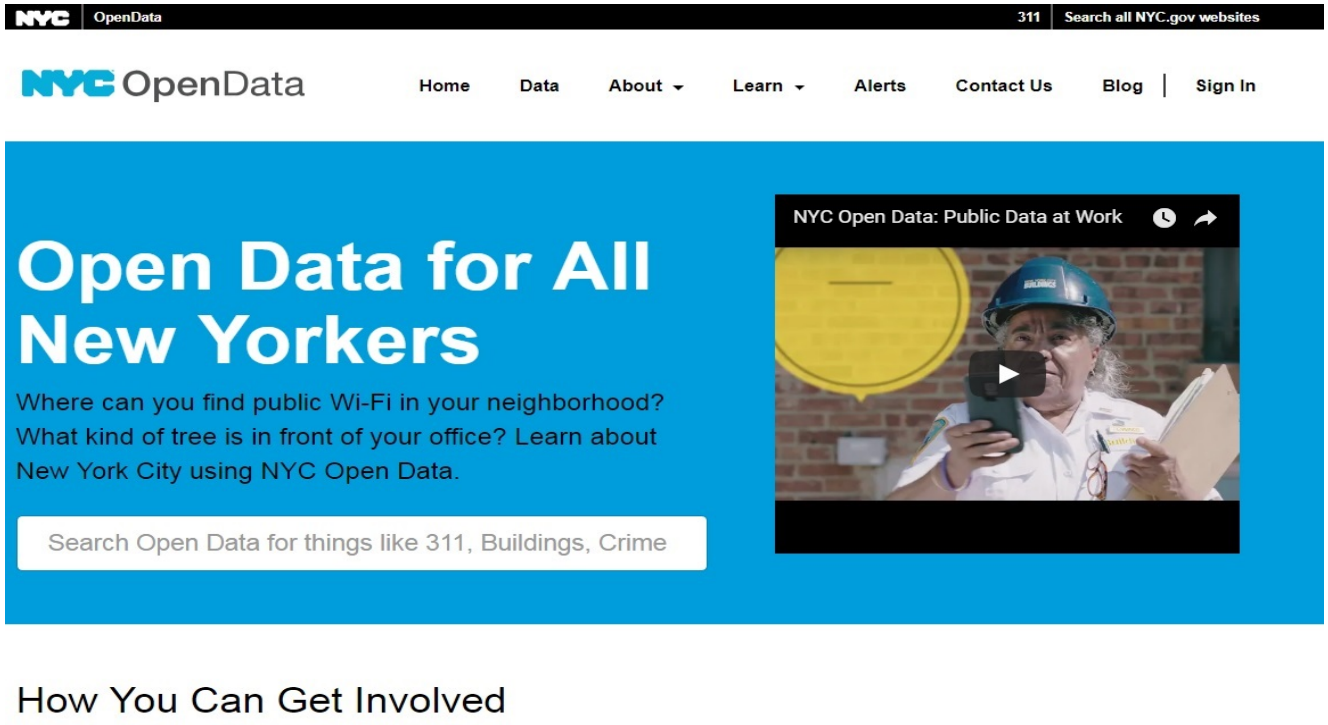
Khaos INVESTIGACIÓN

Table of contents

- Who, why, what, when
- Background
- Working with jMetalSP
- **Case study**
- Further developments

Case Study: Real Time Traffic Problem

- Traveling Salesman Problem (TSP) based on New York City's real-time traffic data.
- Data are from New York City Open Data Website (https://opendata.cityofnewyork.us/).



Case Study: Real Time Traffic Problem

- This is a classical academic problem to which we have incorporated a combination of real nodes (street geo-locations) and real traffic data from the city of New York.

- Dynamic version of the bi-objective Traveling Salesman Problem (DTSP).

- The goal is to minimize the travel time and the distance to cover all the points of the instance.

Case Study: Real Time Traffic Problem

- DTSP problem is composed of 93 locations and 315 communications between them.
- The links are bi-directional, so the resulting DTSP is asymmetric.
- To determine the distance between points, we have used a Google service.

Khaos INVESTIGACIÓN

Case Study: Real Time Traffic Problem

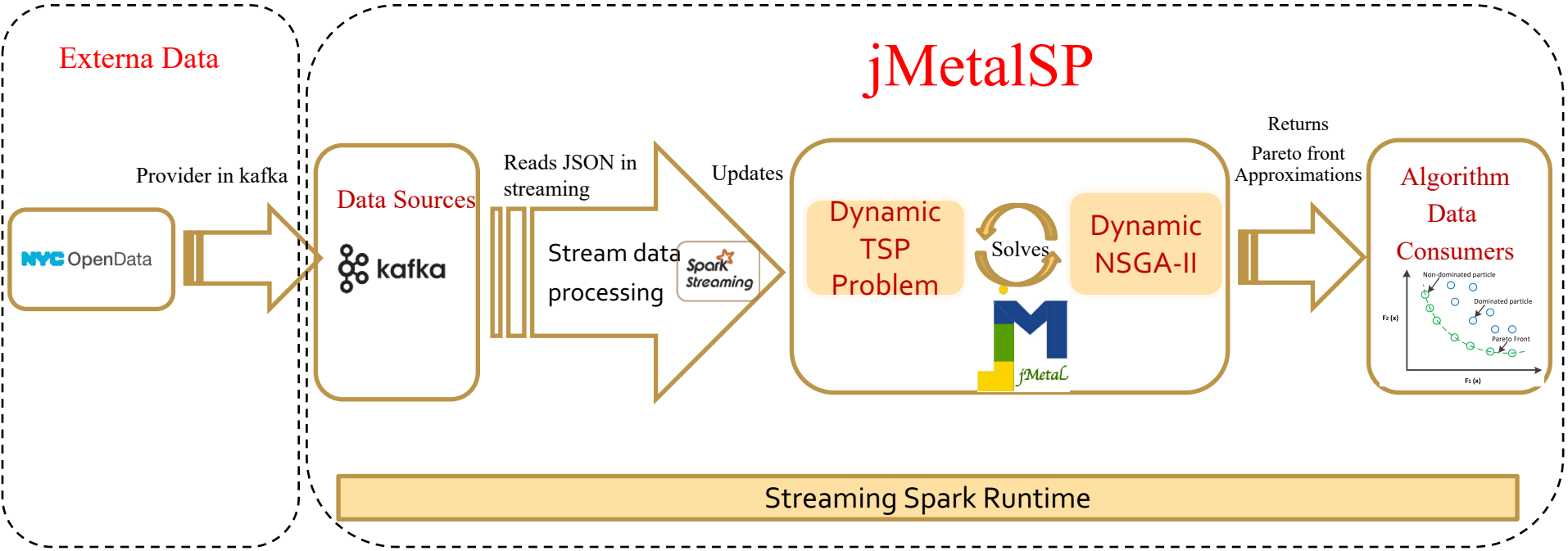
- https://data.cityofnewyork.us/resource/i4gi-tjb9.json


```
{
  "name": "NY Traffic",
  "fields": [
    {
      "name": "id",
      "type": "int"
    },
    {
      "name": "speed",
      "type": "double"
    },
    {
      "name": "travel_time",
      "type": "int"
    },
    {
      "name": "status",
      "type": "int"
    },
    {
      "name": "encoded_poly_line",
      "type": "string"
    },
    {
      "name": "link_name",
      "type": "double"
    }]
}
```

Case Study: Real Time Traffic Problem

- **Id**: link identifier (an integer number).
- **Speed**: average speed a vehicle traveled between end points on the link in the most recent interval (a real number).
- **TravelTime**: average time a vehicle took to traverse the link (a real number).
- **Status**: if the link is closed by accidents, works or any cause (a boolean).
- **EncodedPolyLine**: An encoded string that represents the GPS Coordinates of the link. It is encoded using the Google's Encoded Polyline Algorithm.
- **LinkName**: description of the link location and end points (a string).

Case Study: Real Time Traffic Problem



Case Study: Real Time Traffic Problem

SimpleSparkStructuredKafkaStreamingTSP.java

```
102
103
104         JavaDStream<List<ParsedNode>> nodes=stream.map(value ->{
105
106             List<ParsedNode> result= new ArrayList<>();
107             final JSONArray parser = new JSONArray(value.value());
108             for (int i = 0; i < parser.length(); i++) {
109                 try{
110                     final JSONObject object =  parser.getJSONObject(i);
111                     ParsedNode pNode = new ParsedNode(
112                       object.getInt( key: "id"),
113                       object.getDouble( key: "speed"),
114                       object.getInt( key: "travel_time"),
115                             (object.getDouble( key: "status")==0.0d),
116                       object.getString( key: "encoded_poly_line"),
117                       object.getString( key: "link_name"),
118                             GoogleDecode.decode(object.getString( key: "encoded_poly_line"))
119                     );
120
121                     if(nodeDistances.containsKey(pNode.getId())){
122                         pNode.setDistance(nodeDistances.get(pNode.getId()));
123                     }
124                     result.add(pNode);
125                 }catch (Exception ex){
126                     ex.printStackTrace();
127                 }
128             }
```

Case Study: Real Time Traffic Problem

SimpleSparkStructuredKafkaStreamingTSP.java

```
194         nodes.foreachRDD(aux-> {
195             if (aux != null && aux.rdd().count()>0) {
196                 List<ParsedNode> pNodes = aux.reduce((key, value) -> value);
197                 for (ParsedNode node : pNodes) {
198                     if (hashNodes.get(node.getId()) != null) {
199                         ParsedNode nodeAux = hashNodes.get(node.getId());
200                         if (nodeAux.isStatus() != node.isStatus()) {
201                             if (node.isStatus()) {
202                                 node.setDistance(Integer.MAX_VALUE);
203                                 node.setTravelTime(Integer.MAX_VALUE);
204                             } else if (nodeDistances.containsKey(node.getId())) {
205                                 node.setDistance(nodeDistances.get(node.getId()));
206                             }
207                             node.setDistanceUpdated(true);
208                             node.setCostUpdated(true);
209                             nodeAux.setStatus(node.isStatus());
210                         }
211                         if (node.getTravelTime() != nodeAux.getTravelTime()) {
212                             nodeAux.setTravelTime(node.getTravelTime());
213                             node.setCostUpdated(true);
214                         }
215                     } else {
216                         hashNodes.put(node.getId(), node);
217                         addManualEdges();
218                     }
219                     TSPMatrixData matrix = generateMatrix(node);
220                     observable.setChanged();
221                     observable.notifyObservers(new ObservedValue<>(matrix));
```

Case Study: Real Time Traffic Problem

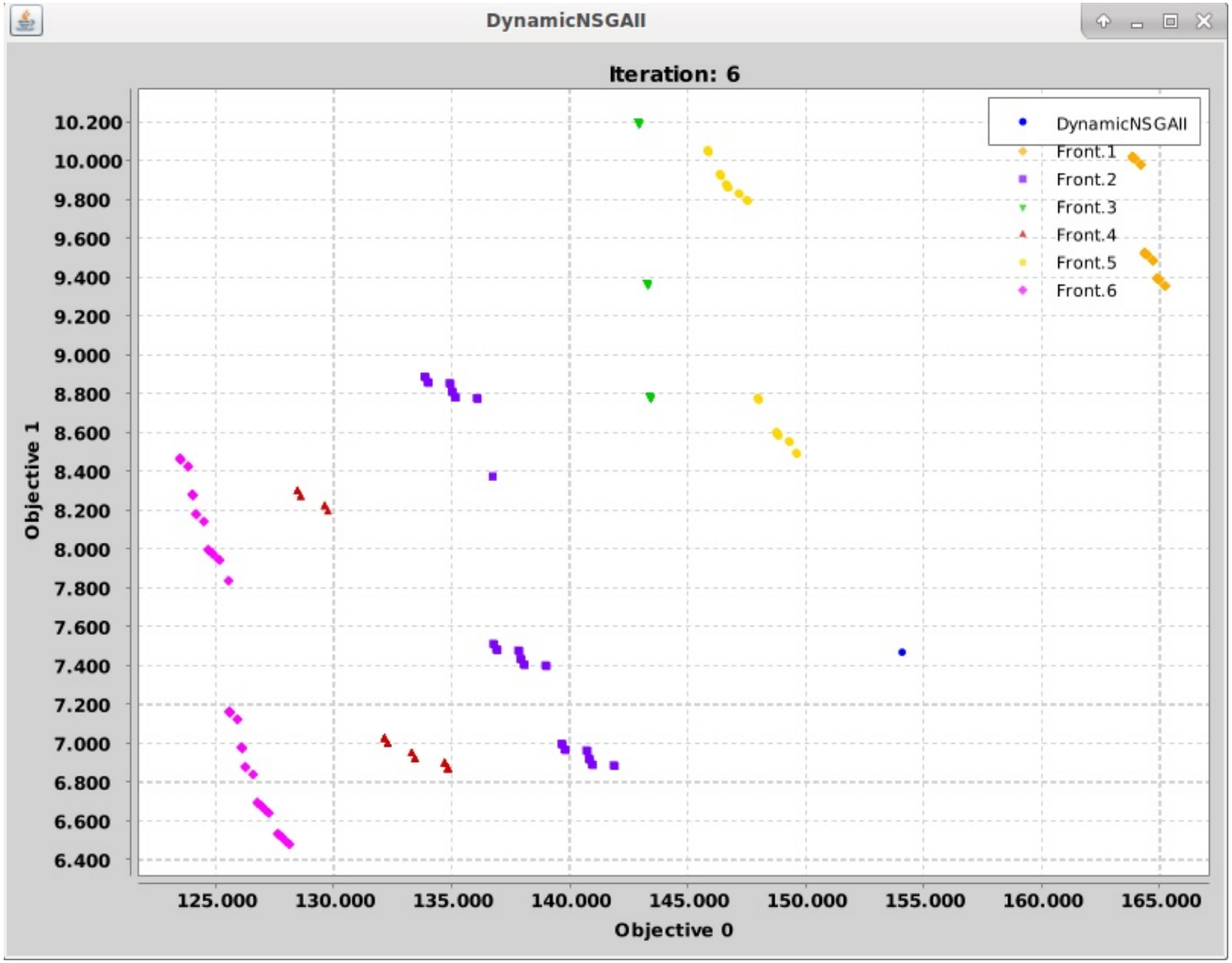


Table of contents

- Who, why, what, when
- Background
- Working with jMetalSP
- Case study
- **Further developments**

Further developments

- jMetalSP has two types of users
  - Designers of dynamic multi-objective algorithms
  - Users requiring the use of a streaming engine for processing large amounts of data
- For the second group
  - The API of Spark streaming will be replaced by Spark Structured Streaming
  - There are other streaming projects
    - Flink
    - Kafka Streams

- Line of work 1: incorporating these systems into jMetalSP

Further developments

- All the components of jMetalSP run as threads of a same process
  - This can limit the usefulness of jMetalSP applications
    - The streaming processing could be near sensors generating the data
    - The optimization algorithm could run on a high performance computer
    - The graphical output could be visualized on a Web application

- Line of work 2: using Kafka to decouple all the jMetal components

Further developments

- We have used two applications with jMetalSP
  - Dynamic benchmark problems
  - Transportation problem
    - One data source with an update frequency of about 1 minute

- Line of work 3: find real-world problems

Thanks for your attention ☺

- Comments and suggestions are welcome