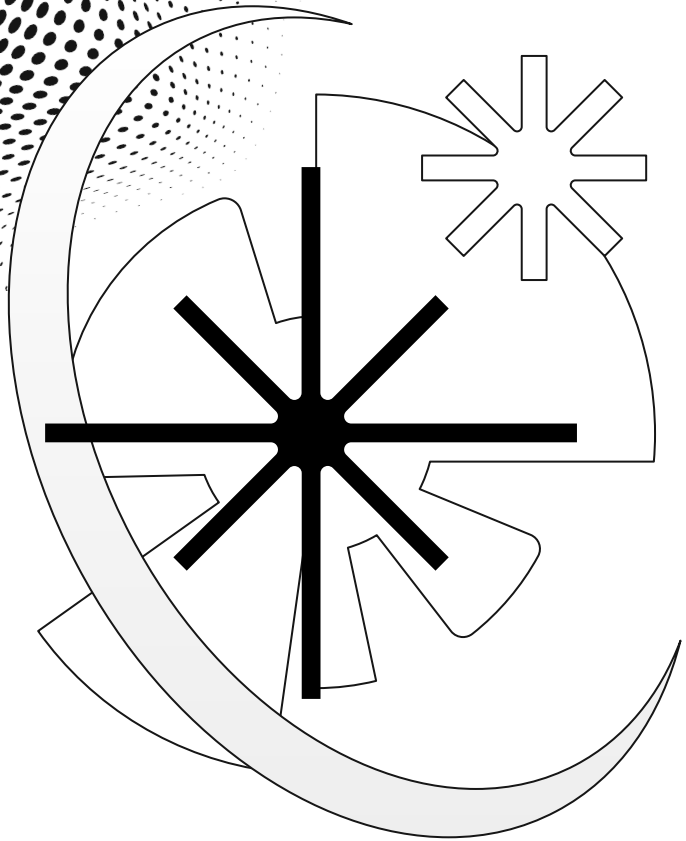
Redes MLP y recurrentes para estimar el precondicionador Simplex en problemas de optimización continua

FIC: UDP

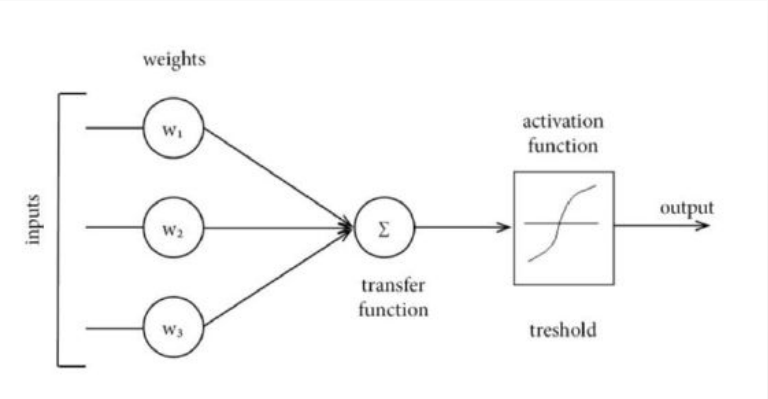


Proyecto Práctica Electiva: Parte final

Estudiante: Joaquín Fernández.
Memorista: Nicolás Claderón.
Profesor: Victor Reyes.

Contenidos de esta presentación

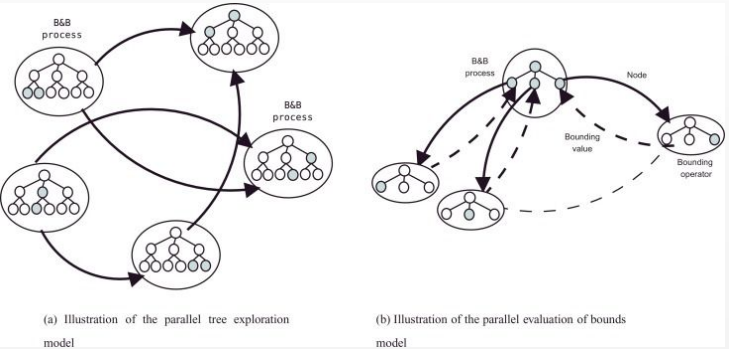
1. Documentación y estado del Arte.
    a. Anteproyecto Nicolás Calderón.
    b. Interval Branch-and-Bound algorithms for optimization and constraint satisfaction: a survey and prospects by Ignacio Araya & Victor Reyes.
    c. The Simplex-simulated Annealing Approach To Continous Non-Linear Optimization by Margarida F. Cardoso, R. L. Salcedot and S. Feyo De Acevedo.
    d. Predicting the Execution Time of the Primal and Dual Simplex Algorithms Using Artificial Neural Networks.
    e. Nonlinear Models and Problems in Applied Sciences from Differential Quadrature to Generalized Collocation Methods
2. Profundizando la herramienta Ibex (Contexto de instalación y uso general).
3. Metodología propuesta.
    a. Esquema de solución (pseudo-algoritmo)
    b. Arquitectura de la aplicación.
    c. Criterios aplicados
    d. Análisis y resultados.
        i. Gráficas en 2D.
        ii. Matriz de Confusión
4. Conclusiones y acotaciones.



a

Anteproyecto Nicolás Calderón

- Branch and Bound (B & B).
- Modelos No Lineales.
- Dominios Continuos.
- Simplex.
- MLP.



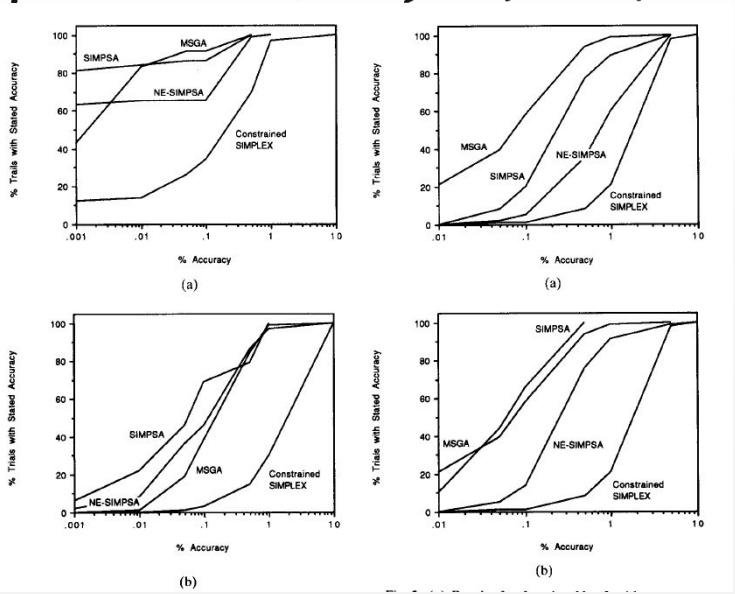
(a) Illustration of the parallel tree exploration model

(b) Illustration of the parallel evaluation of bounds model

b

Interval Branch-and-Bound algorithms for optimization and constraint satisfaction: a survey and prospects por Ignacio Araya y Victor Reyes

- Branch and Bound (B & B).
- Contractors.
- Bisectores.
- CP.



C

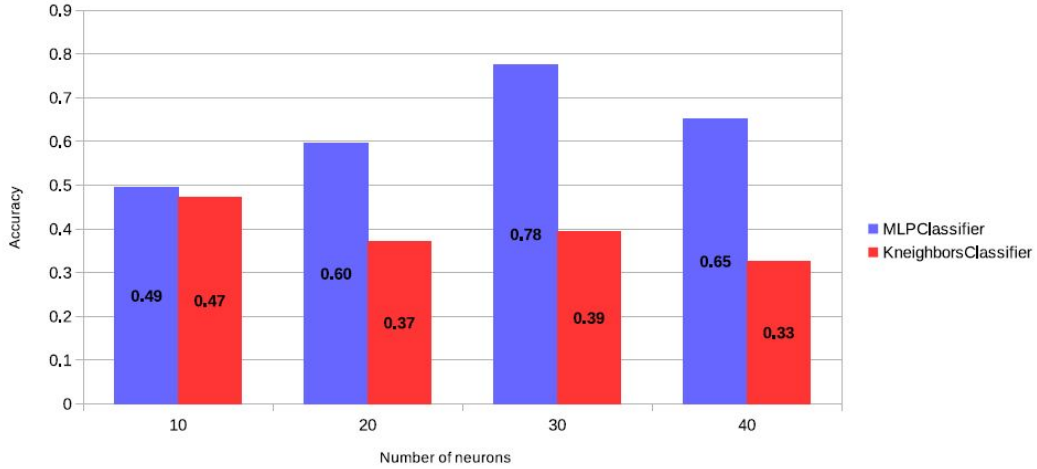
The Simplex-simulated Annealing Approach To Continous Non-Linear Optimization by Margarida F. Cardoso, R. L. Salcedot and S. Feyo De Acevedo (1994).

- Press y Teukolski (1991).
- SIMPSA.
- NE-SIMPSA.
- MSGA.

Comparativa de resultados

Function	Starting point	Simplex	SIMPSA	NE-SIMPSA
7	$(25, 1, 1)^T$	845	44413	23839
	$(30, 0, 0.6)^T$	631	44622	24651
8	$(1, 1, 1, 0.3, 0)^T$	954	59842	23970
	$(0, 0, 0.3, 1.2, 0)^T$	1546	46734	23214
9	$(28, 14, 14, 110, 110, 110, 110)^T$	2040	52402	13780
			298228*	60037*

* $\delta = 10^{-2}$ for SIMPSA and $\delta = 10^{-5}$ for NE-SIMPSA.



d

Predicting the Execution Time of the Primal and Dual Simplex Algorithms Using Artificial Neural Networks

- Simplex Primal y Simplex Dual.
- Algoritmos de Regresión y Clasificación.
  - 4 clases para el tiempo de ejecución.
- Desplazamiento en politopo.

Comparativa de resultados

$$\text{Accuracy} = (TP + TN)/(TP + TN + FP + FN)$$

$$\text{Precision} = TP/(TP + FP)$$

Table 7. Classes of the primal and dual simplex algorithm execution time (in seconds).

Class	Primal	Dual
0	$0 < \text{time} < 0.1$	$0 < \text{time} < 0.1$
1	$0.1 \leq \text{time} < 0.5$	$0.1 \leq \text{time} < 1$
2	$0.5 \leq \text{time} < 4$	$1 \leq \text{time} < 10$
3	$4 \leq \text{time}$	$10 \leq \text{time}$

Comparativa de resultados

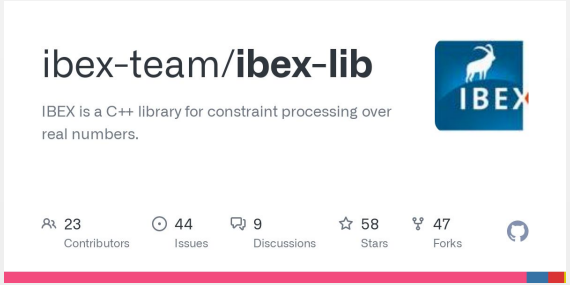
Table 10. Classification reports for the primal and dual simplex algorithm execution time.

Class (Primal)	Precision	Recall	F_1	Support
0	0.92	0.92	0.92	24
1	0.91	0.88	0.89	24
2	0.74	0.87	0.80	23
3	0.73	0.61	0.67	18
avg/total	0.83	0.83	0.83	89
Class (Dual)	**Precision**	**Recall**	F_1	**Support**
0	0.89	1.00	0.94	33
1	0.80	0.83	0.82	24
2	0.88	0.71	0.79	21
3	0.70	0.64	0.67	11
avg/total	0.84	0.84	0.84	89

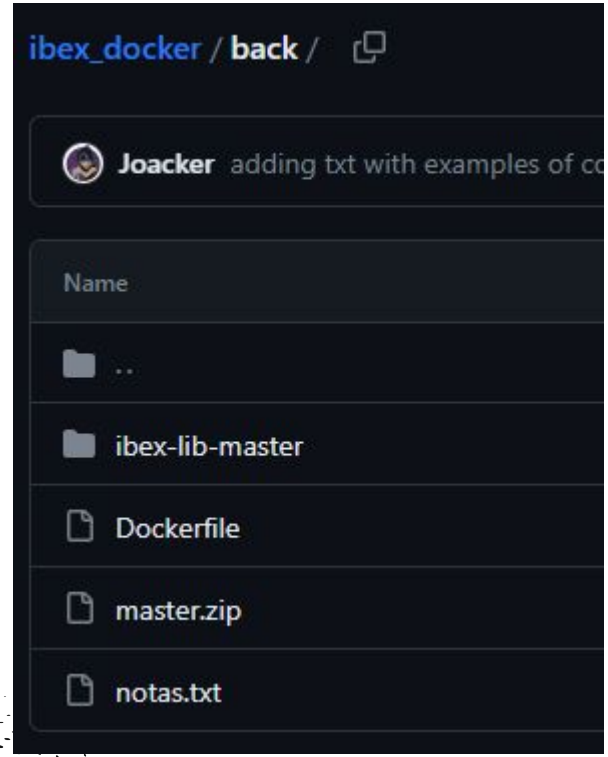
02

Profundizando Ibex y reuniones con profesor y memorista

http://www.ibex-lib.org



Herramienta aplicada en contenedor



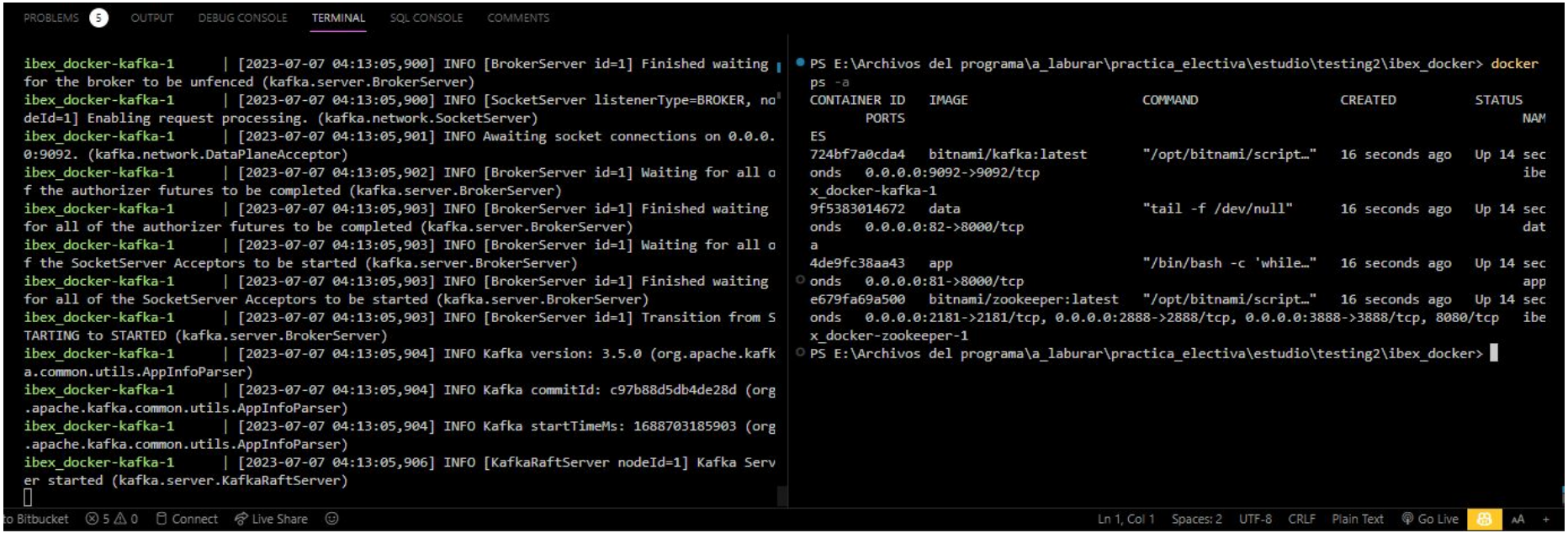

```
FROM ubuntu:22.04


RUN apt-get update && apt-get install -y \
    bash \
    g++ \
    gcc \
    flex \
    bison \
    python2-dev \
    make \
    git \
    build-essential \
    cmake

RUN apt install -y \
     python-is-python3

RUN apt install -y \
    python3-pip

RUN apt install libeigen3-dev -y

RUN apt update
RUN apt upgrade -y

# Continue with your Dockerfile instructions
WORKDIR /app
COPY ./ibex-lib-master /app

RUN pip install -r requirements.txt

RUN chmod +x waf

# RUN ./waf configure --interval-lib=gaol --lp-lib=soplex \
# && ./waf install

# ./__build__/src/ibexopt benchs/optim/medium/alkylbis.bch --random-seed=1
# ./__build__/src/ibexopt benchs/optim/hard/chembis.bch --random-seed=1 >> commands.log
# ./waf distclean

CMD ["tail", "-f", "/dev/null"]
```

Herramienta aplicada en contenedor

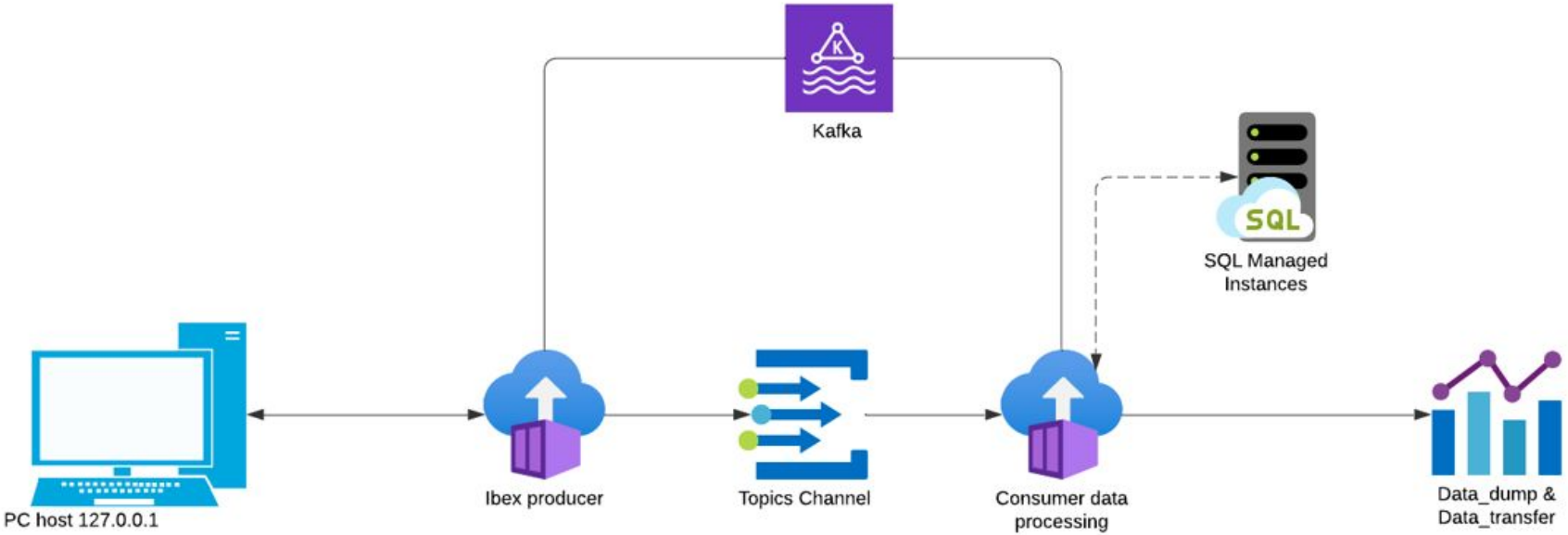
```yaml
version: '3.7'

services:

  zookeeper:
    image: "bitnami/zookeeper:latest"
    restart: always
    environment:
      - ALLOW_ANONYMOUS_LOGIN=yes
    ports:
      - "2181:2181"
      - "2888:2888"
      - "3888:3888"

  kafka:
    image: "bitnami/kafka:latest"
    restart: always
    ports:
      - "9092:9092"
    environment:
      - KAFKA_CFG_ZOOKEEPER_CONNECT=zookeeper:2181
      - ALLOW_PLAINTEXT_LISTENER=yes
    depends_on:
      - zookeeper

```

```yaml
    app:
      image: app
      build: ./back
      container_name: app
      ports:
        - 81:8000
      #make a volumes for an entire folder of cpp files
      # volumes:
      #   - ./back:/app&ibex-lib/examples
      #make still runing container
      stdin_open: true
      tty: true
      #make a container restart when it stops
      restart: always
      #make a container run in background
      volumes:
        - ./back/ibex-lib-master:/app
        - ./results:/app/results
      command: /bin/bash -c "while true; do sleep 1; done"


    data:
      image: data
      build: ./data
      container_name: data
      ports:
        - 82:8000
      volumes:
        - ./data:/app
        - ./results:/app/results
        - ./models:/app/models
        - ./back/ibex-lib-master/benchs:/app/benchmarks

```

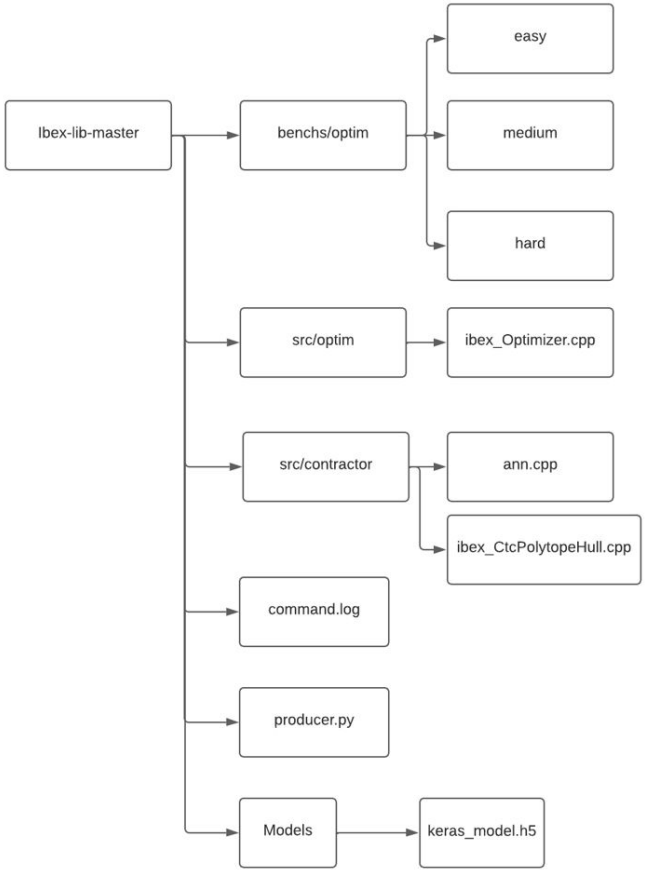
Herramienta aplicada en contenedor

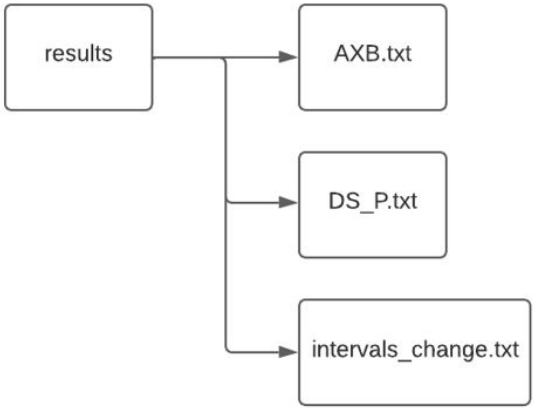


Ambiente de desarrollo y de pruebas



Arquitectura de ibex y en que se prestará atención





03

Metodología

a. Pseudo-algoritmo propuesto

1) Recopilación Preparación de datos

2) Diseño y entrenamiento de la red neuronal

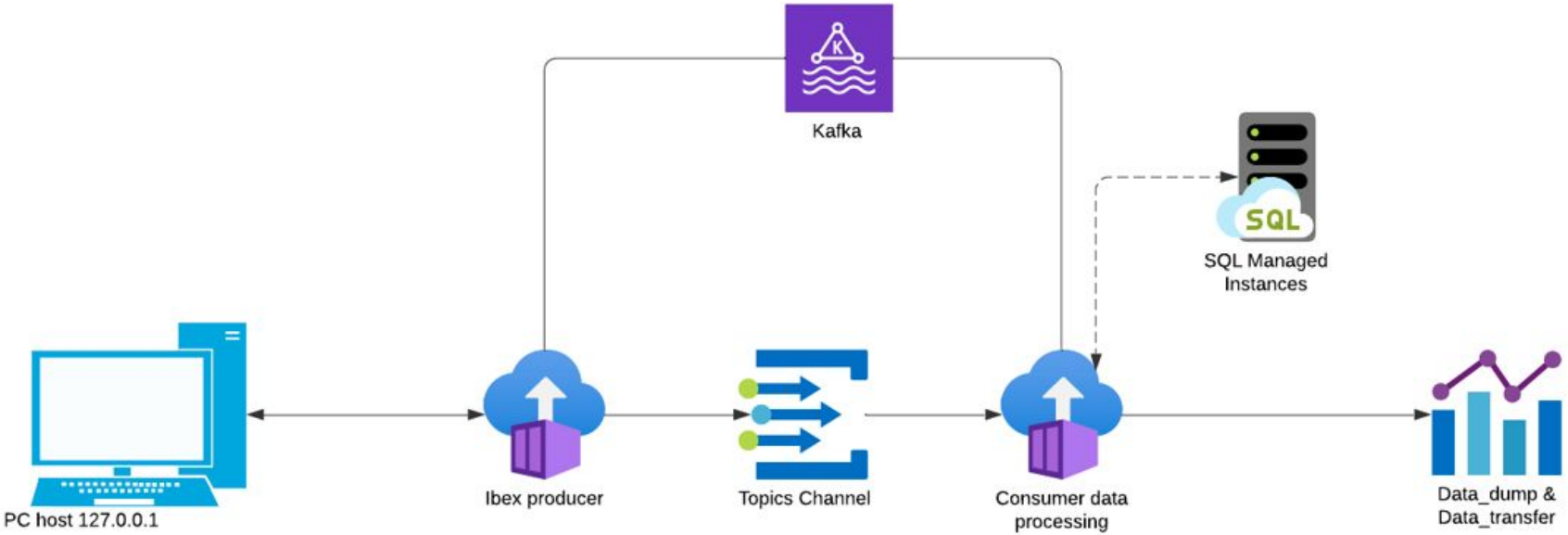
3) Validación cruzada y ajuste de hiperparámetros

4) Evaluación del rendimiento

5) Optimización de los vértices del politopo

6) Resultados y conclusiones

B. Arquitectura de la aplicación



C. Criterios aplicados

- Variar probabilidad con respecto a una distribución srand

```
void CtcPolytopeHull::optimizer(IntervalVector& box, float proba) {

    Interval opt(0.0);
    int* inf_bound = new int[nb_var]; // indicator inf_bound = 1 means the inf bound is feasible or already contracted, call to simplex useless (cf Baharev)
    int* sup_bound = new int[nb_var]; // indicator sup_bound = 1 means the sup bound is feasible or already contracted, call to simplex useless

    for (int i=0; i<nb_var; i++) {

        if (contracted_vars[i]) {
            inf_bound[i]=0;
            sup_bound[i]=0;
        } else {
            inf_bound[i]=1;
            sup_bound[i]=1;
        }
    }

    int nexti=-1;   // the next variable to be contracted
    int infnexti=0; // the bound to be contracted contract  infnexti=0 for the lower bound, infnexti=1 for the upper bound
    LPSolver::Status stat=LPSolver::Status::Unknown;

    // Update the bounds the variables
    mylinearsolver.set_bounds(box);
    //make random to choice probas with srand
    float probas[2]={proba,1-proba};
    if (proba >= probas[0]){
    for(int ii=0; ii<(2*nb_var); ii++) {  // at most 2*n calls
```

C. Criterios aplicados

- Considerar la cantidad de variables de entrada para el cálculo de la precisión

```
variables

x1 in [-32,32];
x2 in [-32,32];
x3 in [-32,32];
x4 in [-32,32];
x5 in [-32,32];

minimize

-20 * exp (-0.2 * sqrt (0.2 * (x1^2 + x2 ^2 + x3^2 + x4^2 + x5 ^2))) - exp (0.2 * (cos (2*pi*x1) + cos (2*pi*x2) +cos (2*pi*x3)
+ cos (2*pi*x4) + cos (2*pi*x5))) + 20 + 2.718281;

constraints
x1 <= 500;
end

```

C. Criterios aplicados

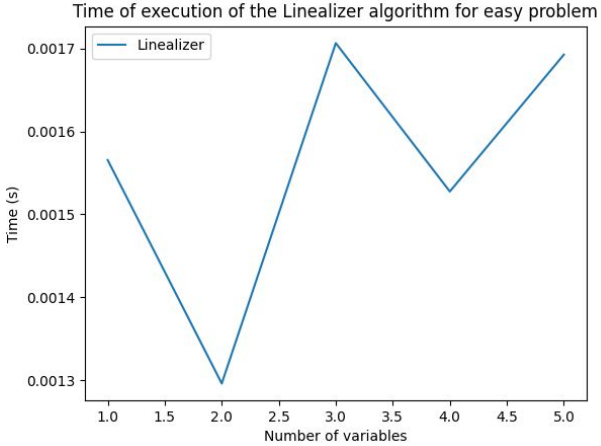
- Timeout tras el paso de 45 (2700 s) minutos o 1 hora (3600 s)

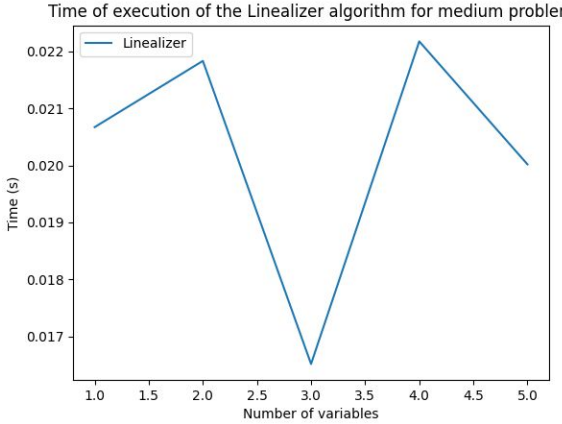
```
clock_t t1,t2;
t1=clock();
for (int i=0;i<lr2->L_b.size();i++){
        AXB_for_red.push_back(lr2->L_b[i]);
        // stop post time 2700 s pass
        if (i==2700){
            t2=clock();
            float diff ((float)t2-(float)t1);
            float seconds = diff / CLOCKS_PER_SEC;
            if (seconds>2700){
                break;
            }
        }
    }
```

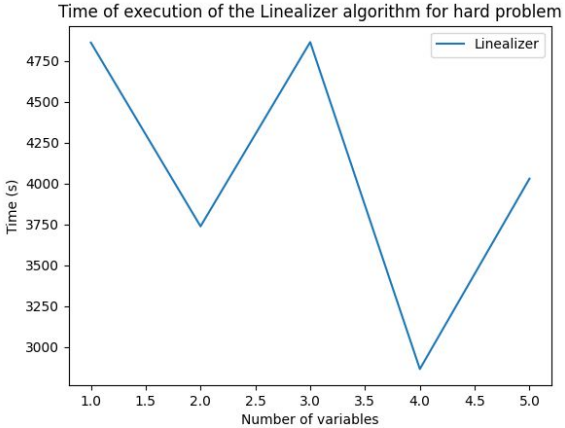
04

Resultados

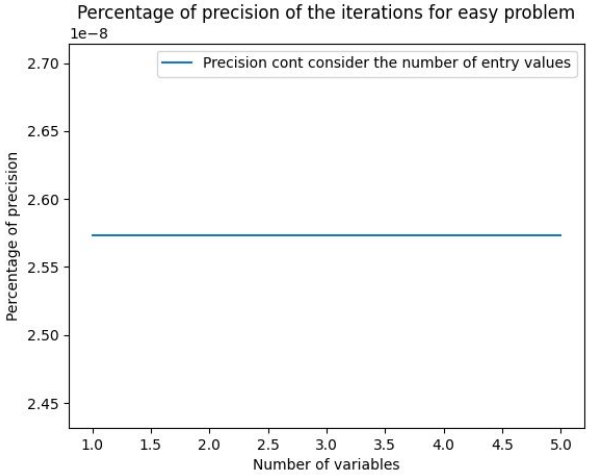
Graficas linealizer: Problemas easy, medium y hard

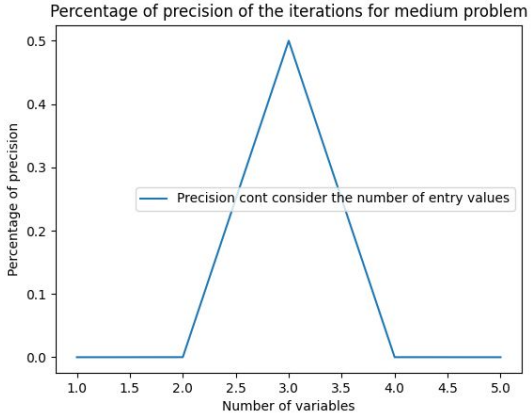


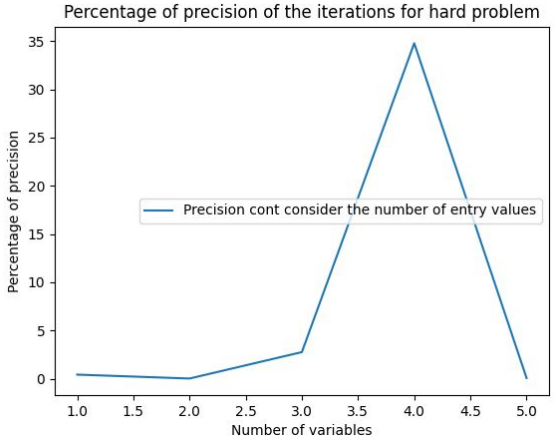




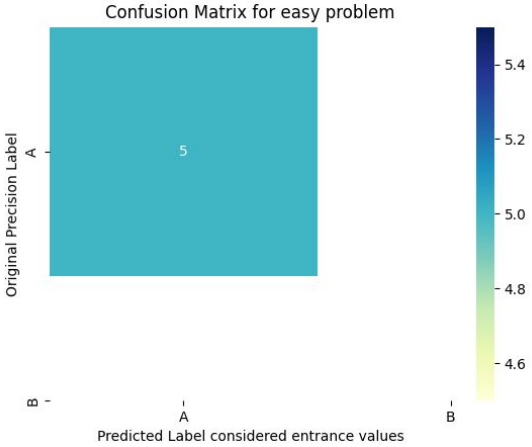
Gráficas Precisión: Problemas easy, medium y hard

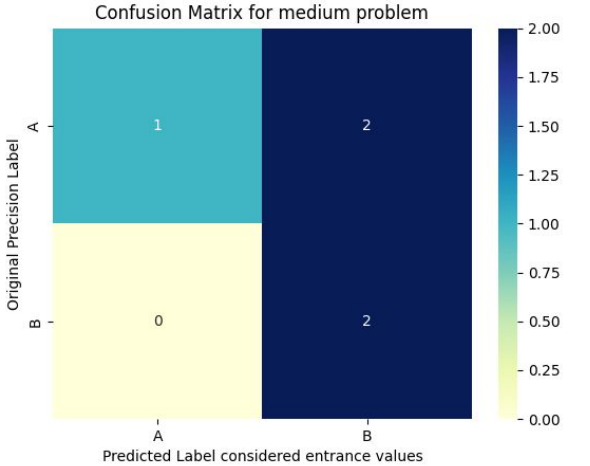


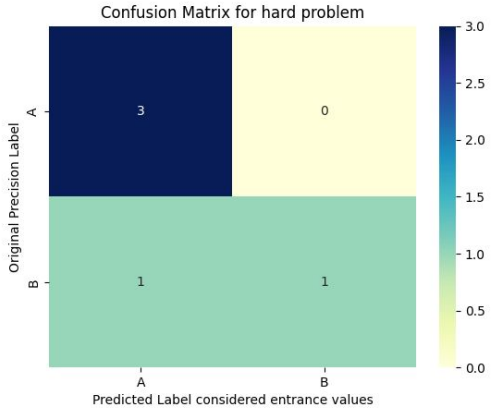




Matriz de Confusion: Problemas easy, medium y hard







4. Conclusiones y acotaciones

1. Como posible solución podría verse el tema de la parametrización para tener control del flujo y estructura.
2. Probar nuevos modelos de redes neuronales ubicadas en keras.
3. Establecer mejores criterios para el armado de precondicionadores.
4. Evaluar usar más problemas dentro de ibex para establecer probabilidades o coeficientes de aprendizaje más apropiados.

Enlaces de videos de demostración

Parte 1: https://www.youtube.com/watch?v=sgeV153Rnaw

Parte 2: https://www.youtube.com/watch?v=nC12ZCvvi-g

Enlaces de repositorios

https://github.com/Joacker/ibex_docker

https://github.com/Joacker/Preconditioners_Simplex

https://github.com/hohenheim99/Preconditioners_Simplex

https://github.com/moof2k/kerasify

GRACIAS TOTALES

¿Alguna duda? Pregunta?
Consulta? Incertidumbre?
Vacilación? Interrogante?
Dubitación ? Disyuntiva?