

Paralelización del algoritmo Progressive Hedging para la resolución de problemas estocásticos

Grao en Ingeniería Informática
Universidad de Santiago de Compostela

Autor: Cristofer Canosa Domínguez

Director: Juan Carlos Pichel Campos

9 de septiembre de 2018

Tabla de contenidos

- 1 Introducción
 - Programación Estocástica
 - Pyomo
 - Objetivos
- 2 Gestión del proyecto
 - Alcance y entregables
 - Planificación temporal
- 3 Análisis
 - Algoritmo *Progressive Hedging*
 - Herramientas
- 4 Diseño
- 5 Implementación
- 6 Pruebas
- 7 Conclusiones

Programación estocástica

- Problemas de optimización

Programación estocástica

- Problemas de optimización
- Existe un nivel de incertidumbre

Programación estocástica

- Problemas de optimización
- Existe un nivel de incertidumbre
- Generan múltiples escenarios

Programación estocástica

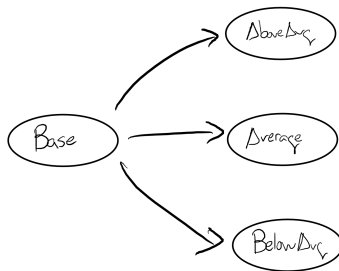


Figura: Árbol de escenarios en un problema estocástico

Ejemplos de aplicación

- 1 TFM: “Problemas de rutas de vehículos”
- 2 Modelo de optimización de la oferta de generación eléctrica para compañías eléctricas que participan en el mercado eléctrico liberalizado MIBEL.
- 3 “Progressive Hedging aplicado a coordinación hidrotérmica”

Pyomo

- Formulación y solución de modelos de optimización
- Uso de solucionadores de terceros (CPLEX, GLPK)
- *Sandia National Laboratories y University of California*
- Python



Objetivos

- 1 Estudiar y analizar el funcionamiento del algoritmo *Progressive Hedging* en PySP.
- 2 Análisis de las diferentes alternativas de paralelización disponibles que mejor se adapten al problema. Se tendrán en cuenta tecnologías Big Data (Apache Spark) o modelos tradicionales de paralelización.
- 3 Diseño e implementación del nuevo módulo e integración con Pyomo.
- 4 Análisis y evaluación del rendimiento.

Alcance

- Adaptar el módulo de programación estocástica (PySP) a una nueva implementación paralela.
- Nueva implementación más escalable que permita abordar problemas de mayor tamaño.
- Realizar un análisis de rendimiento.

Entregables

- Código de Pyomo actualizado con el módulo de ejecución de PH paralelo.
- Estudio de rendimiento.
- Memoria de realización del proyecto.
- Otra documentación asociada a la realización del proyecto.

Metodología de desarrollo

- Límite temporal estricto

Metodología de desarrollo

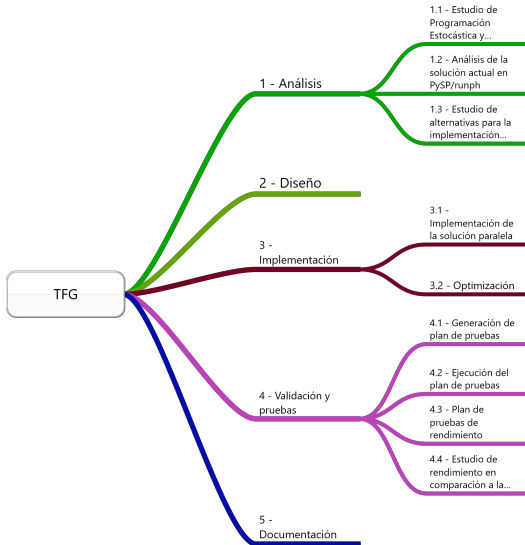
- Límite temporal estricto
- Poco peso de la fase de implementación

Metodología de desarrollo

- Límite temporal estricto
- Poco peso de la fase de implementación
- Mayor énfasis en análisis y documentación

Metodología de desarrollo

- Límite temporal estricto
- Poco peso de la fase de implementación
- Mayor énfasis en análisis y documentación
- Metodología en cascada



Cronograma

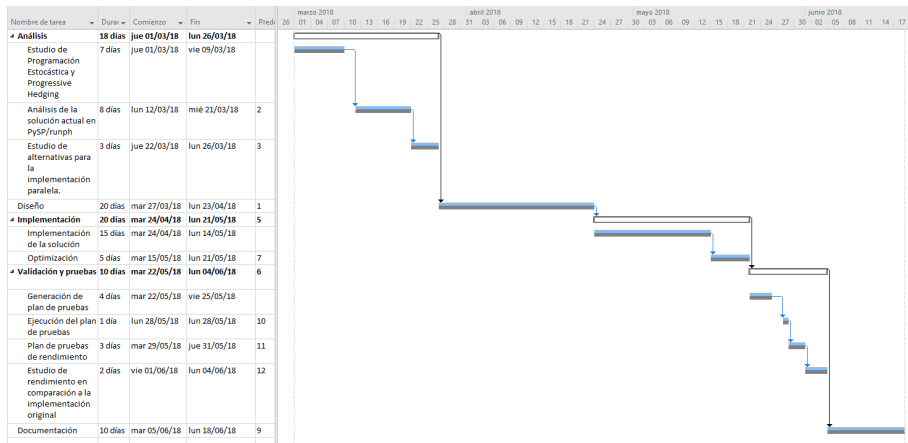


Figura: Línea base

Adaptación de la planificación

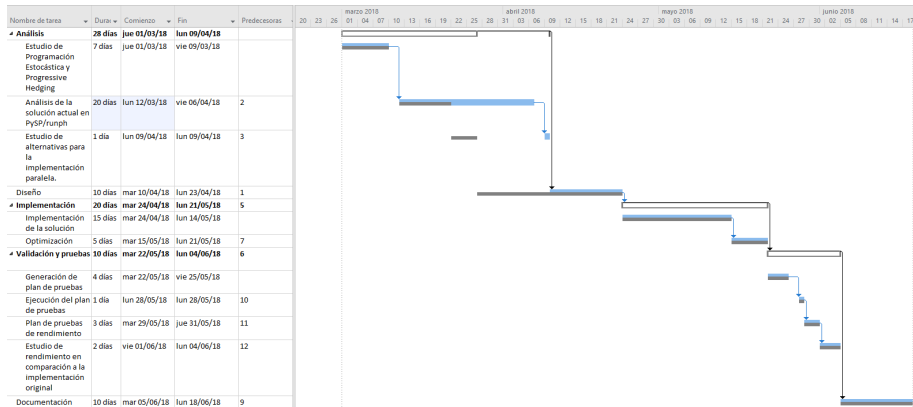


Figura: Primer retraso

Adaptación de la planificación

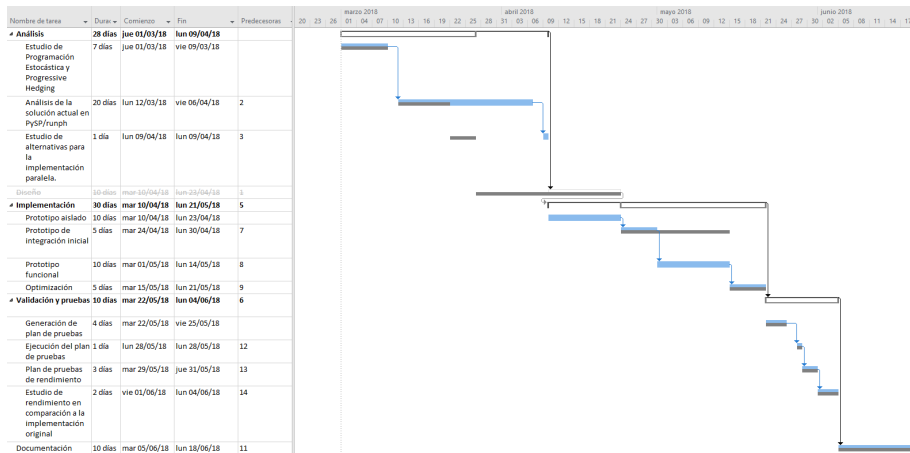


Figura: Línea base prototipos

Cronograma final

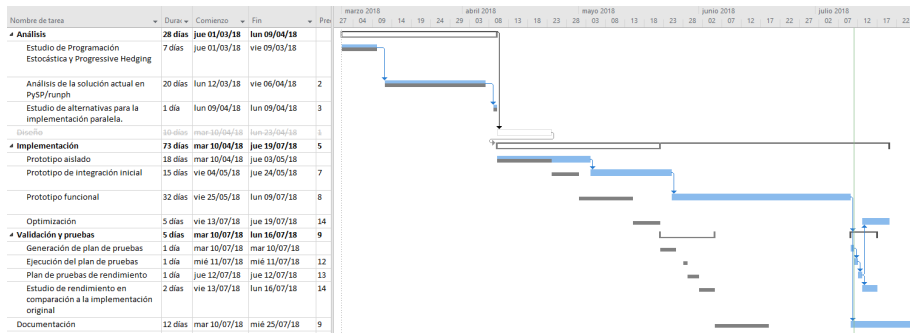


Figura: Línea base final

Progressive Hedging

- Algoritmo iterativo
- Descomposición del árbol de escenarios en múltiples problemas lineales.
- Convergencia de las múltiples soluciones

$$k := 0$$

$$\Delta s \in S, x_s^{(k)} := \operatorname{argmin}_x (c \cdot x + f_s \cdot y_s) : (x, y_s) \in Q_s$$

$$\hat{x}^{(k)} := \sum_{s \in S} \operatorname{Pr}(s) x_s^{(k)}$$

$$w_s^{(k)} := \rho(x_s^{(k)} - \hat{x}_s^{(k)})$$

$$k := k + 1$$

$$\Delta s \in S, x_s^{(k)} := \operatorname{argmin}_x (c \cdot x + w_s^{(k-1)} x + \rho/2 \|x - \hat{x}^{(k-1)}\|^2 + f_s \cdot y_s) \in Q_s$$

$$\hat{x}^{(k)} := \sum_{s \in S} \operatorname{Pr}(s) x_s^{(k)}$$

$$\Delta s \in S, w_s^{(k)} := w_s^{(k-1)} + \rho(x_s^{(k)} - \hat{x}_s^{(k)})$$

$$g^{(k)} := \sum_{s \in S} \operatorname{Pr}(s) \|x_s^{(k)} - \hat{x}_s^{(k)}\|$$

If $g^{(k)} < \epsilon$ repeat from 5, else terminate

Implementación en Pyomo

- ❶ Se importan el modelo y los escenarios.
- ❷ Genera los objetos necesarios para la ejecución.
- ❸ Cada iteración se divide en 3 fases:
 - ❶ “presolve”: Preprocesa los escenarios.
 - ❷ “apply_solver”: Ejecuta el solver externo.
 - ❸ “postsolve”: Adapta la salida del solver externo.

Implementación en Pyomo

- Arquitectura basada en tareas
- Facilita intercambiar el módulo que ejecuta la solución del problema
- Simplifica la paralelización

Herramientas

Para la nueva implementación paralela se analizarán dos posibles tecnologías:

- 1 Spark
- 2 MPI

Spark

- Herramienta Big Data de código abierto

Spark

- Herramienta Big Data de código abierto
- Abstracción del entorno distribuido

Spark

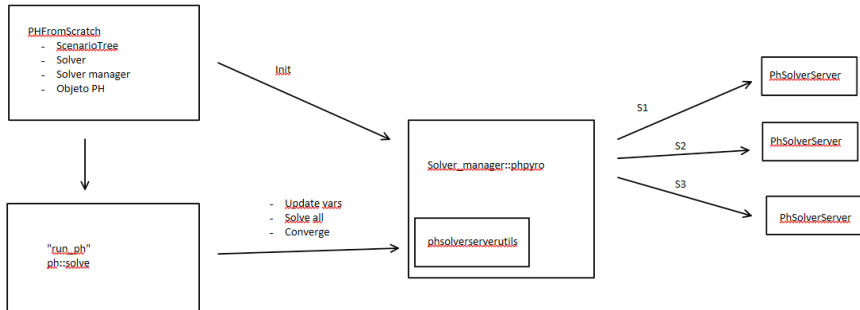
- Herramienta Big Data de código abierto
- Abstracción del entorno distribuido
- Trabaja sobre RDD (*Resilient Distributed Datasets*):
 - Transformaciones
 - Acciones

Spark

- Herramienta Big Data de código abierto
- Abstracción del entorno distribuido
- Trabaja sobre RDD (*Resilient Distributed Datasets*):
 - Transformaciones
 - Acciones
- Otros módulos (Dataframes, MLib, etc)

MPI

- Interfaz estándar de paso de mensajes
- Desacoplamiento entre trabajadores
- Implementación de bajo nivel (sincronización entre trabajadores, tipos personalizados)
- Funciones específicas para el tratamiento paralelo de datos (MPI_BCAST, MPI_SCATTER, etc)



Proceso

- Generación de 3 prototipos

Proceso

- Generación de 3 prototipos
 - ① Prototipo inicial

Proceso

- Generación de 3 prototipos
 - 1 Prototipo inicial
 - 2 Prototipo de integración

Proceso

- Generación de 3 prototipos
 - 1 Prototipo inicial
 - 2 Prototipo de integración
 - 3 Prototipo funcional

Funcionamiento

Implementación

Introducción
Gestión del proyecto
Análisis
Diseño
Implementación
Pruebas
Conclusiones

Recursos

Resultados

Lecciones aprendidas

Trabajo futuro

Conclusiones