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

15 de septiembre de 2018

Tabla de contenidos

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

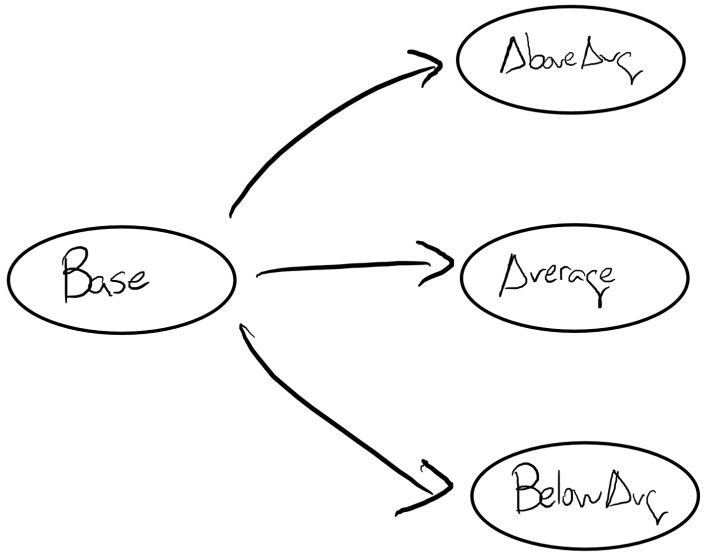


Figura: Árbol de escenarios en un problema estocástico

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.

Casos de uso

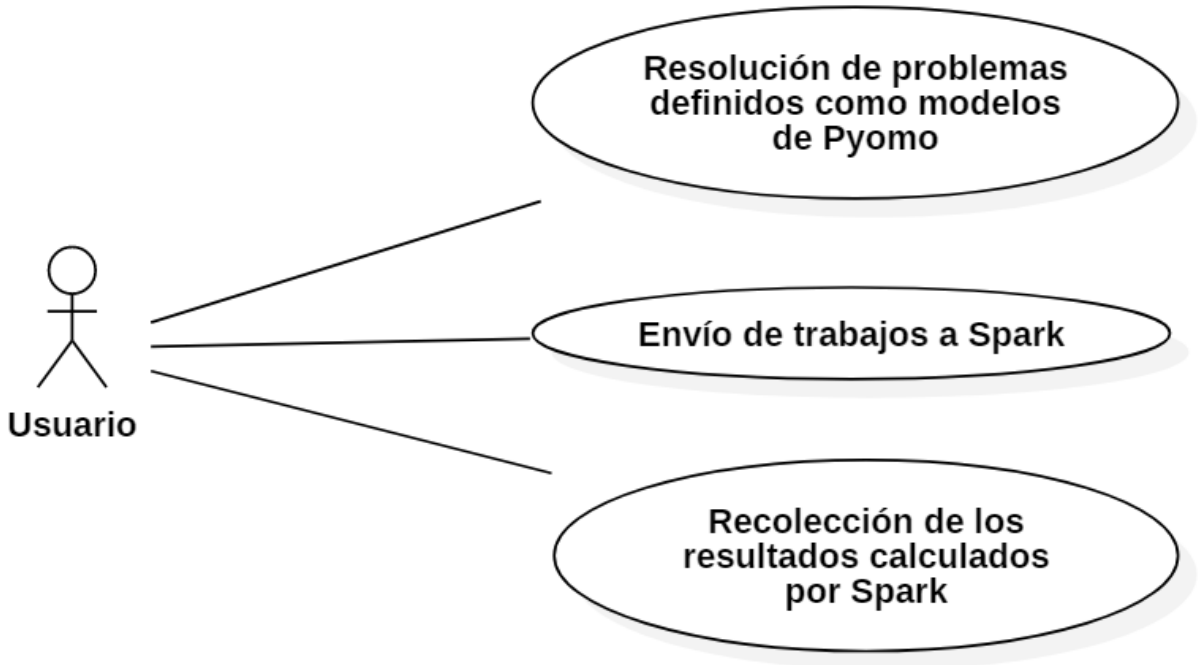


Figura: Casos de uso

Metodología de desarrollo

- Límite temporal estricto

Metodología de desarrollo

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

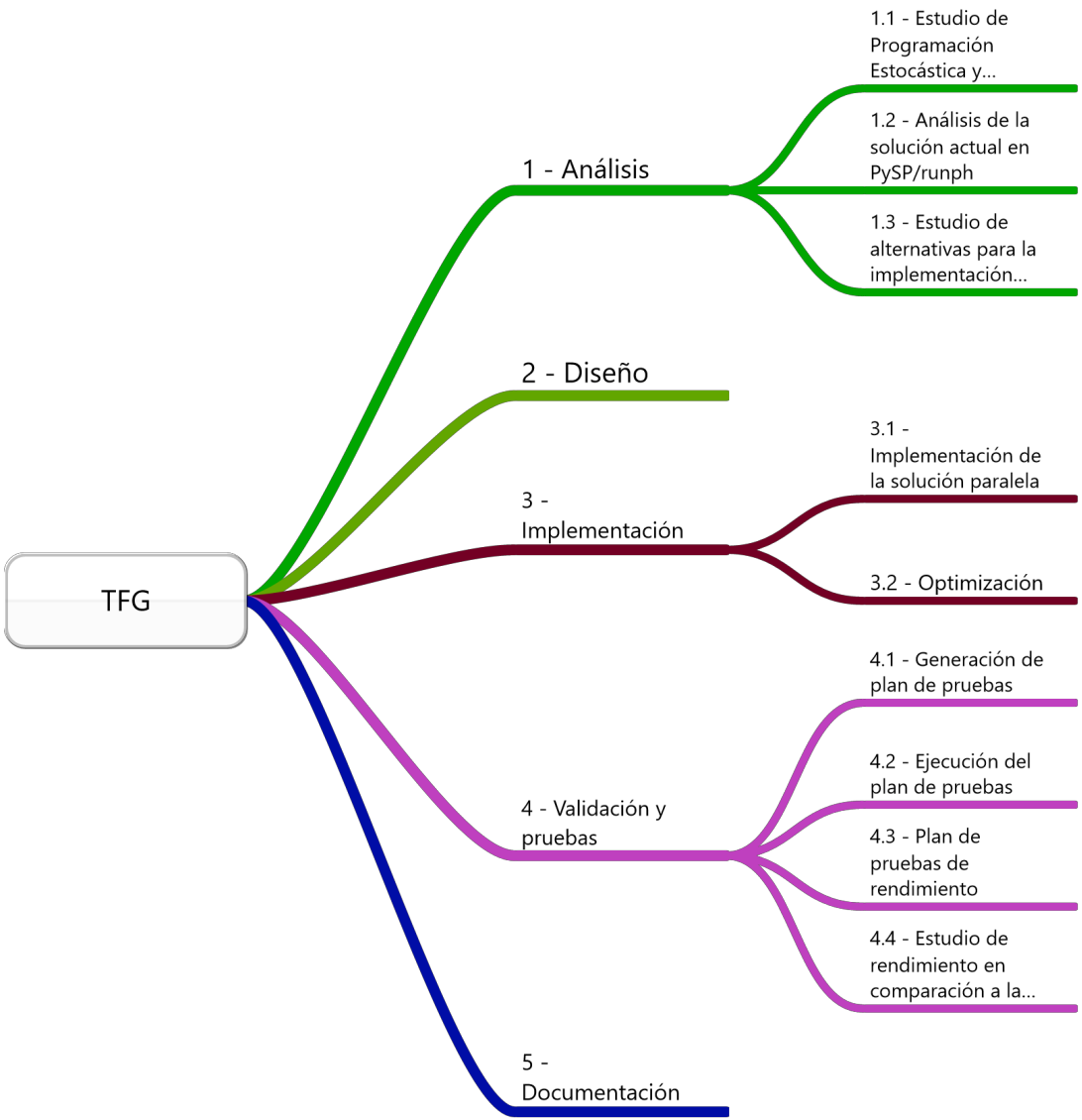
Metodología de desarrollo

- 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

EDT


TFG
1 - Análisis
1.1 - Estudio de Programación Estocástica y...
1.2 - Análisis de la solución actual en PySP/runph
1.3 - Estudio de alternativas para la implementación...
2 - Diseño
3 - Implementación
3.1 - Implementación de la solución paralela
3.2 - Optimización
4 - Validación y pruebas
4.1 - Generación de plan de pruebas
4.2 - Ejecución del plan de pruebas
4.3 - Plan de pruebas de rendimiento
4.4 - Estudio de rendimiento en comparación a la...
5 - Documentación

Cronograma

Nombre de tarea	Dura	Comienzo	Fin	Pred
Análisis	**18 días**	**jue 01/03/18**	**lun 26/03/18**	
Estudio de Programación Estocástica y Progressive Hedging	7 días	jue 01/03/18	vie 09/03/18	
Análisis de la solución actual en PySP/runph	8 días	lun 12/03/18	mié 21/03/18	2
Estudio de alternativas para la implementación paralela.	3 días	jue 22/03/18	lun 26/03/18	3
Diseño	20 días	mar 27/03/18	lun 23/04/18	1
Implementación	**20 días**	**mar 24/04/18**	**lun 21/05/18**	**5**
Implementación de la solución	15 días	mar 24/04/18	lun 14/05/18	
Optimización	5 días	mar 15/05/18	lun 21/05/18	7
Validación y pruebas	**10 días**	**mar 22/05/18**	**lun 04/06/18**	**6**
Generación de plan de pruebas	4 días	mar 22/05/18	vie 25/05/18	
Ejecución del plan de pruebas	1 día	lun 28/05/18	lun 28/05/18	10
Plan de pruebas de rendimiento	3 días	mar 29/05/18	jue 31/05/18	11
Estudio de rendimiento en comparación a la implementación original	2 días	vie 01/06/18	lun 04/06/18	12
Documentación	10 días	mar 05/06/18	lun 18/06/18	9

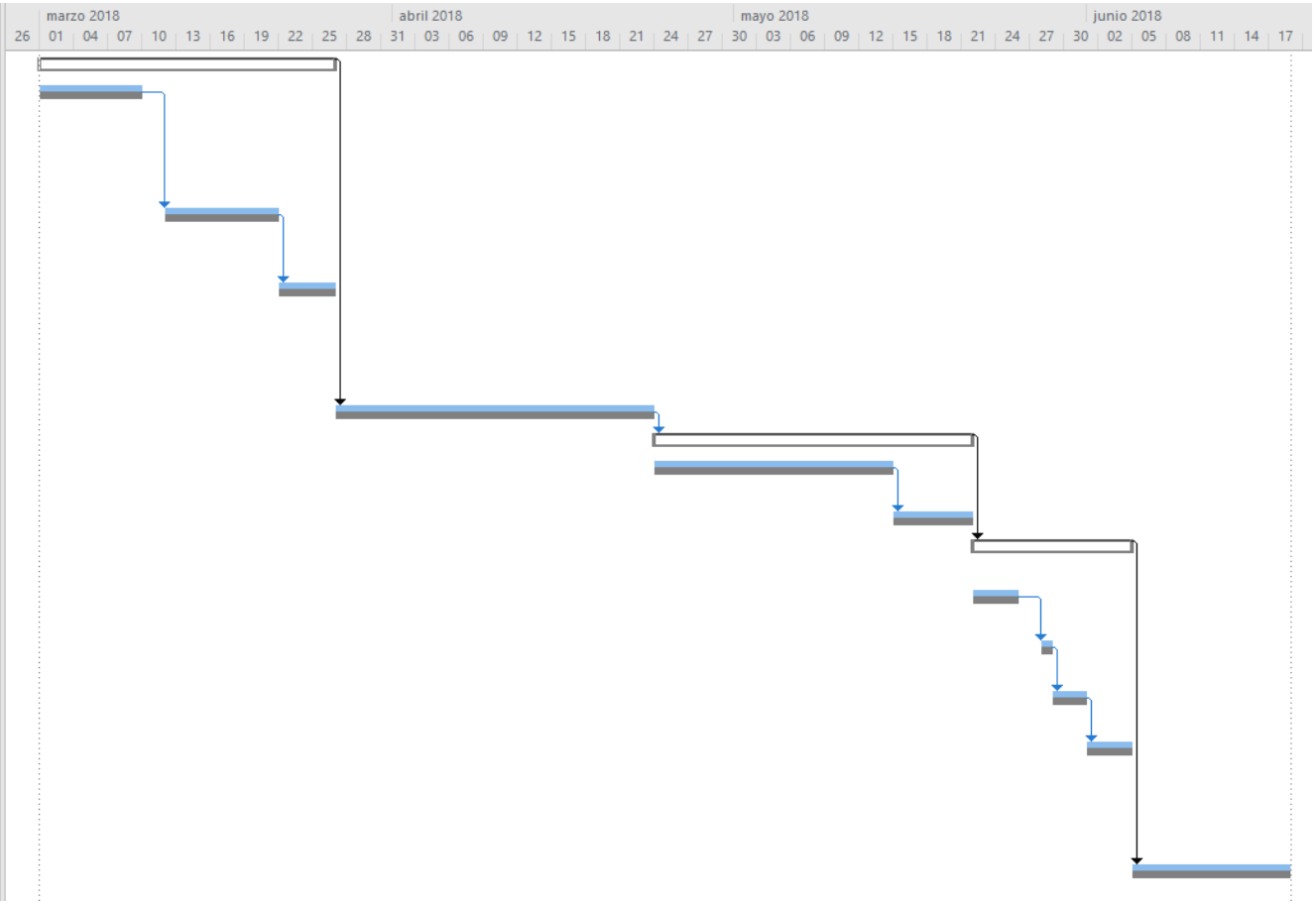


Figura: Línea base

Adaptación de la planificación

Nombre de tarea	Dura	Comienzo	Fin	Predecesoras
Análisis	**28 días**	**jue 01/03/18**	**lun 09/04/18**	
Estudio de Programación Estocástica y Progressive Hedging	7 días	jue 01/03/18	vie 09/03/18	
Análisis de la solución actual en PySP/runph	20 días	lun 12/03/18	vie 06/04/18	2
Estudio de alternativas para la implementación paralela.	1 día	lun 09/04/18	lun 09/04/18	3
Diseño	10 días	mar 10/04/18	lun 23/04/18	1
Implementación	**20 días**	**mar 24/04/18**	**lun 21/05/18**	**5**
Implementación de la solución	15 días	mar 24/04/18	lun 14/05/18	
Optimización	5 días	mar 15/05/18	lun 21/05/18	7
Validación y pruebas	**10 días**	**mar 22/05/18**	**lun 04/06/18**	**6**
Generación de plan de pruebas	4 días	mar 22/05/18	vie 25/05/18	
Ejecución del plan de pruebas	1 día	lun 28/05/18	lun 28/05/18	10
Plan de pruebas de rendimiento	3 días	mar 29/05/18	jue 31/05/18	11
Estudio de rendimiento en comparación a la implementación original	2 días	vie 01/06/18	lun 04/06/18	12
Documentación	10 días	mar 05/06/18	lun 18/06/18	9

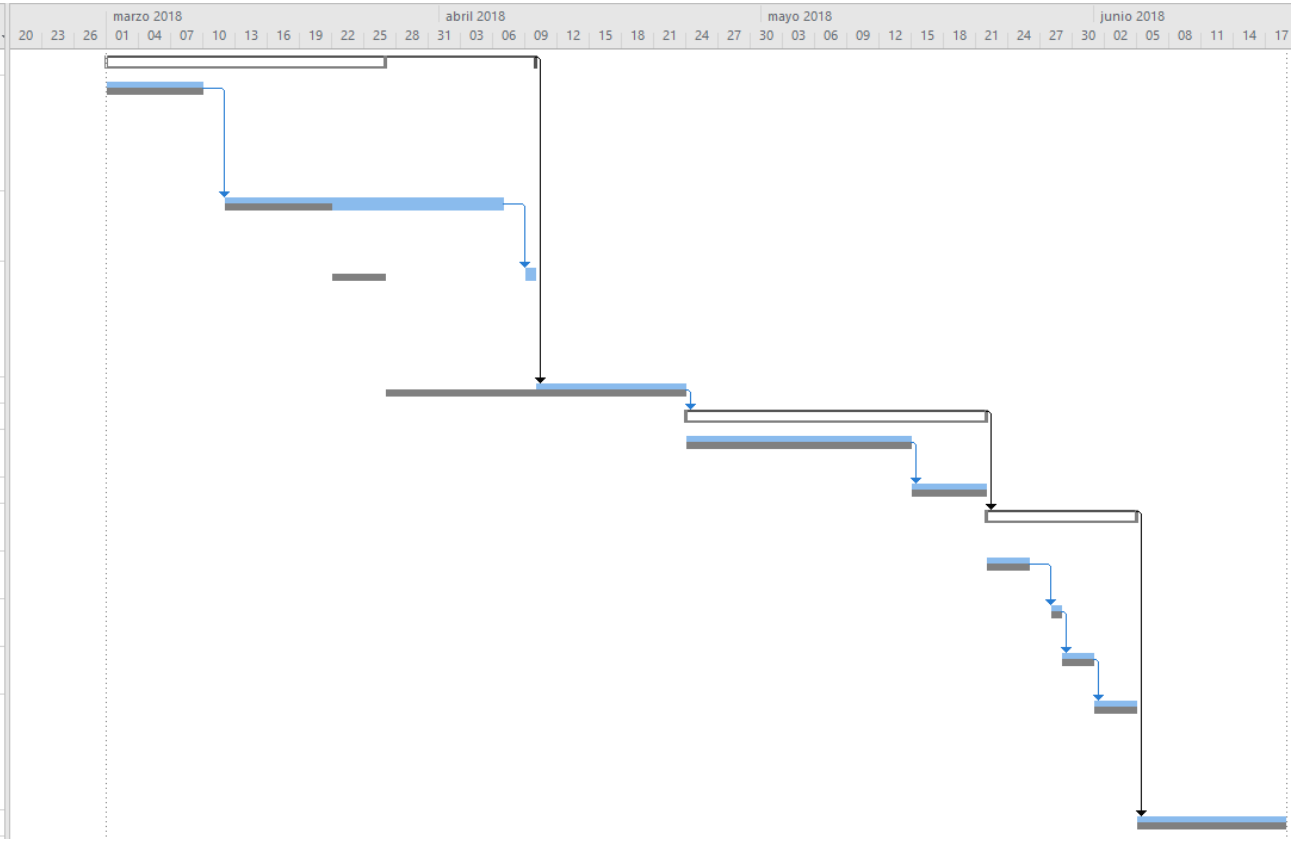


Figura: Primer retraso

Adaptación de la planificación

Nombre de tarea	Dura	Comienzo	Fin	Predecesoras
Análisis	**28 días**	**jue 01/03/18**	**lun 09/04/18**	
Estudio de Programación Estocástica y Progressive Hedging	7 días	jue 01/03/18	vie 09/03/18	
Análisis de la solución actual en PySP/runph	20 días	lun 12/03/18	vie 06/04/18	2
Estudio de alternativas para la implementación paralela.	1 día	lun 09/04/18	lun 09/04/18	3
~~Diseño~~	~~10 días~~	~~mar 10/04/18~~	~~lun 23/04/18~~	~~1~~
Implementación	**30 días**	**mar 10/04/18**	**lun 21/05/18**	**5**
Prototipo aislado	10 días	mar 10/04/18	lun 23/04/18	
Prototipo de integración inicial	5 días	mar 24/04/18	lun 30/04/18	7
Prototipo funcional	10 días	mar 01/05/18	lun 14/05/18	8
Optimización	5 días	mar 15/05/18	lun 21/05/18	9
Validación y pruebas	**10 días**	**mar 22/05/18**	**lun 04/06/18**	**6**
Generación de plan de pruebas	4 días	mar 22/05/18	vie 25/05/18	
Ejecución del plan de pruebas	1 día	lun 28/05/18	lun 28/05/18	12
Plan de pruebas de rendimiento	3 días	mar 29/05/18	jue 31/05/18	13
Estudio de rendimiento en comparación a la implementación original	2 días	vie 01/06/18	lun 04/06/18	14
Documentación	10 días	mar 05/06/18	lun 18/06/18	11

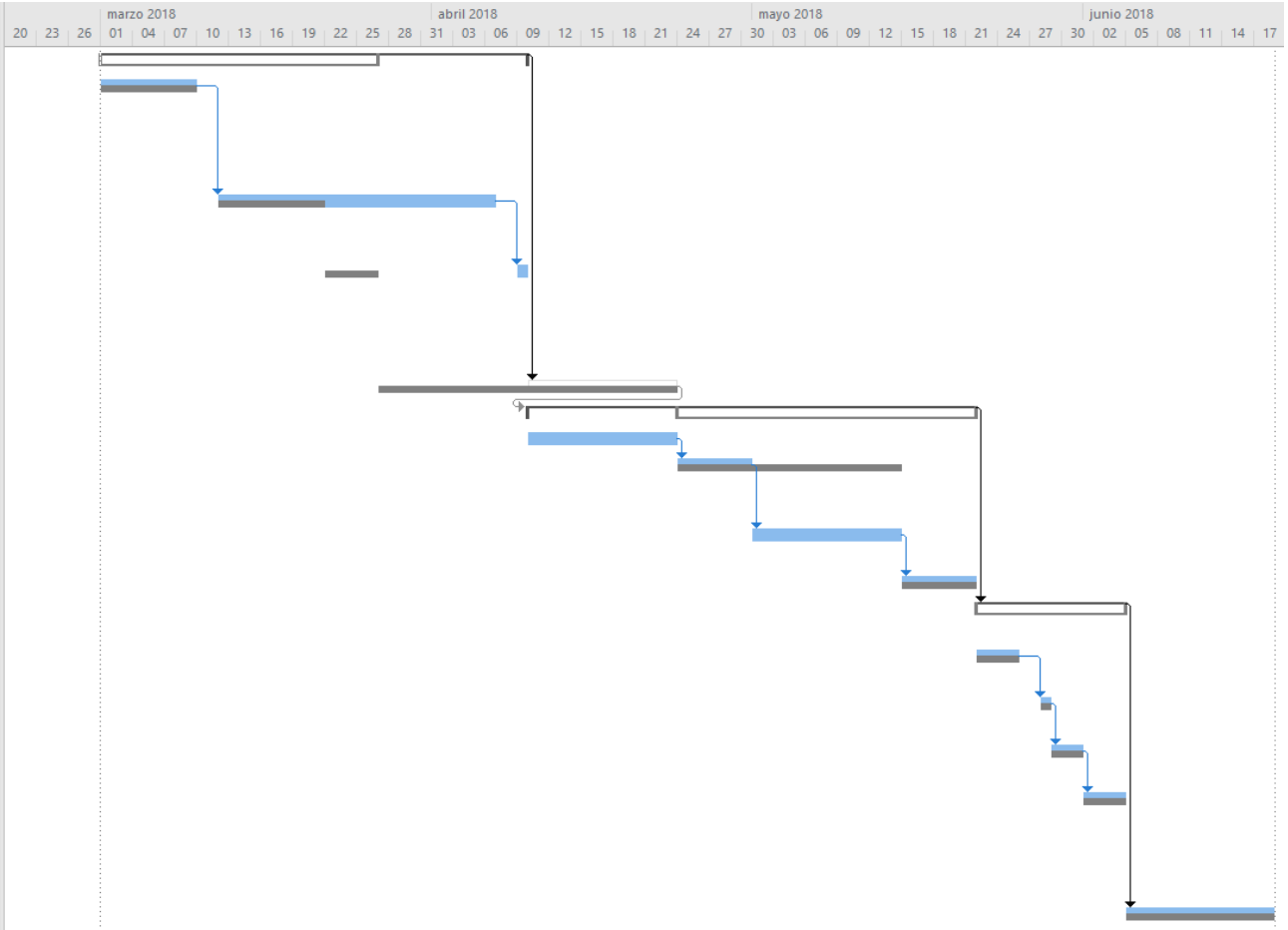


Figura: Línea base prototipos

Cronograma final

Nombre de tarea	Dura	Comienzo	Fin	Pre
Análisis	**28 días**	**jue 01/03/18**	**lun 09/04/18**	
Estudio de Programación Estocástica y Progressive Hedging	7 días	jue 01/03/18	vie 09/03/18	
Análisis de la solución actual en PySP/runph	20 días	lun 12/03/18	vie 06/04/18	2
Estudio de alternativas para la implementación paralela.	1 día	lun 09/04/18	lun 09/04/18	3
~~Diseño~~	~~10 días~~	~~mar 10/04/18~~	~~lun 23/04/18~~	~~1~~
Implementación	**73 días**	**mar 10/04/18**	**jue 19/07/18**	**5**
Prototipo aislado	18 días	mar 10/04/18	jue 03/05/18	
Prototipo de integración inicial	15 días	vie 04/05/18	jue 24/05/18	7
Prototipo funcional	32 días	vie 25/05/18	lun 09/07/18	8
Optimización	5 días	vie 13/07/18	jue 19/07/18	14
Validación y pruebas	**5 días**	**mar 10/07/18**	**lun 16/07/18**	**9**
Generación de plan de pruebas	1 día	mar 10/07/18	mar 10/07/18	
Ejecución del plan de pruebas	1 día	mié 11/07/18	mié 11/07/18	12
Plan de pruebas de rendimiento	1 día	jue 12/07/18	jue 12/07/18	13
Estudio de rendimiento en comparación a la implementación original	2 días	vie 13/07/18	lun 16/07/18	14
Documentación	12 días	mar 10/07/18	mié 25/07/18	9

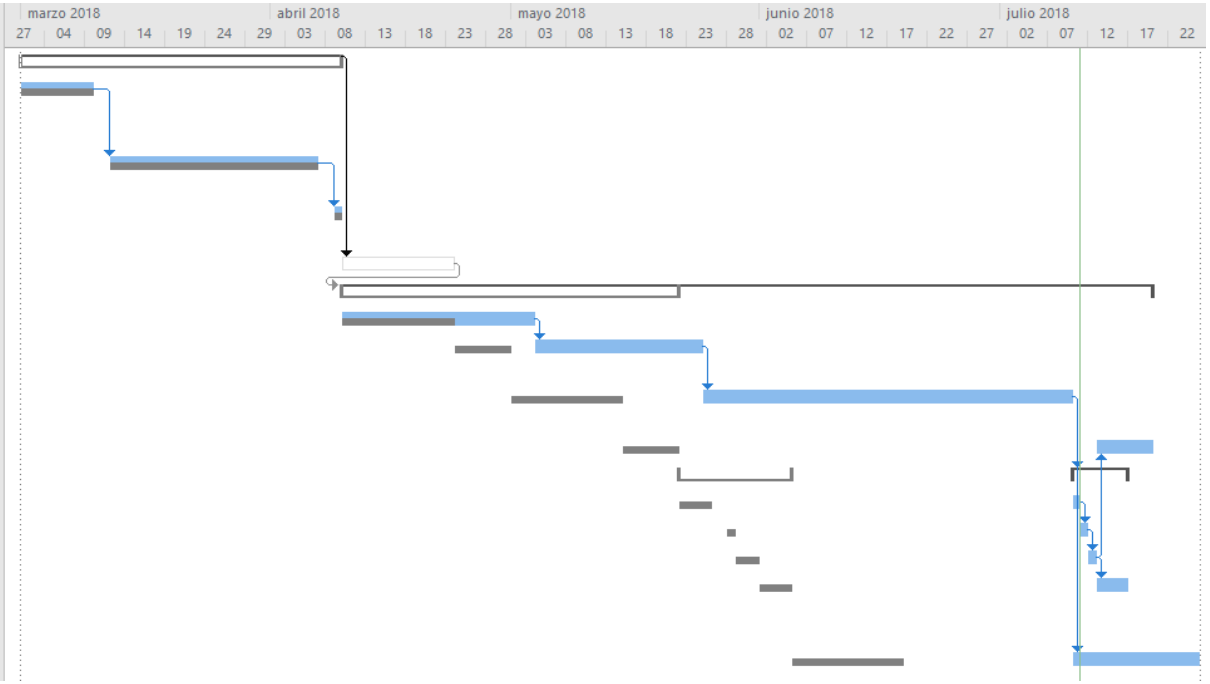


Figura: Línea base final

Progressive Hedging

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

Progressive Hedging

$k := 0$

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

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

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

$k := k + 1$

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

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

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

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

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

Figura: Pseudocódigo del algoritmo

Implementación en Pyomo

1. Se importan el modelo y los escenarios.
2. Genera los objetos necesarios para la ejecución.
3. Cada iteración se divide en 3 fases:
   1. "presolve": Preprocesa los escenarios.
   2. "apply_solver": Ejecuta el solver externo.
   3. "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

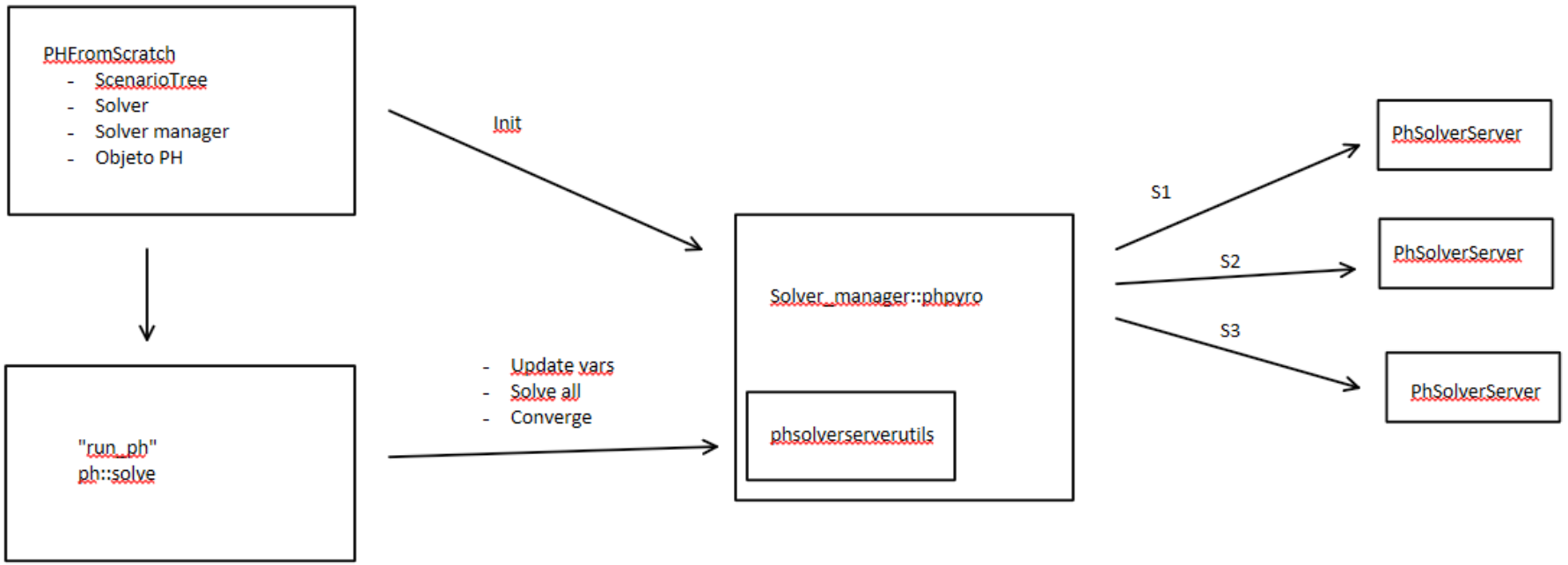
Spark

- 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)

Diseño


PHFromScratch
- ScenarioTree
- Solver
- Solver manager
- Objeto PH
Init
"run_ph"
ph::solve
- Update vars
- Solve all
- Converge
Solver_manager::phpyro
phsolverserverutils
S1
S2
S3
PhSolverServer
PhSolverServer
PhSolverServer

Proceso

- Generación de 3 prototipos

Proceso

- Generación de 3 prototipos
  1. Prototipo inicial

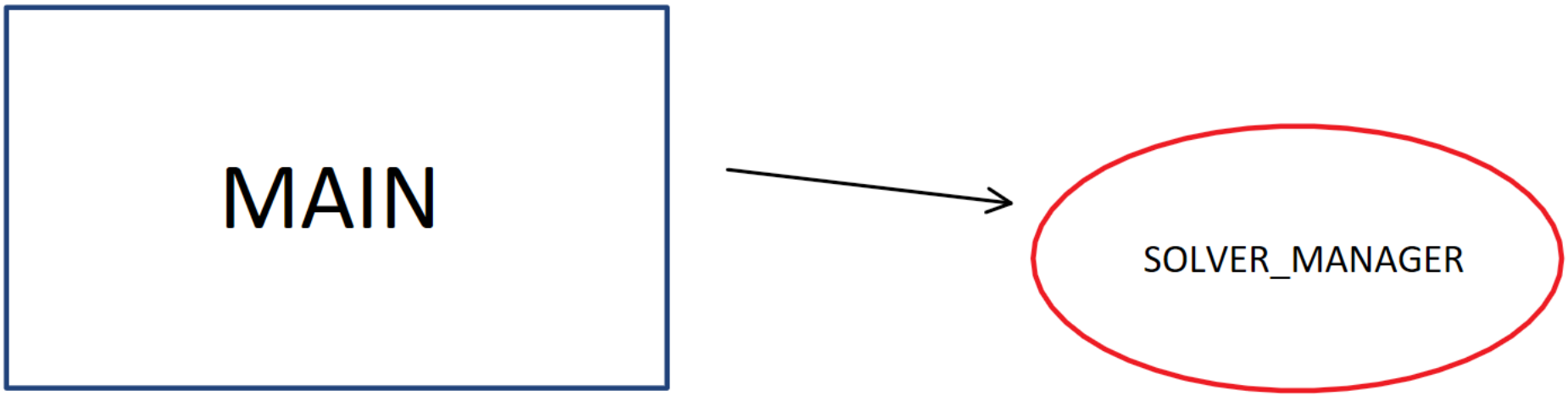
Proceso

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

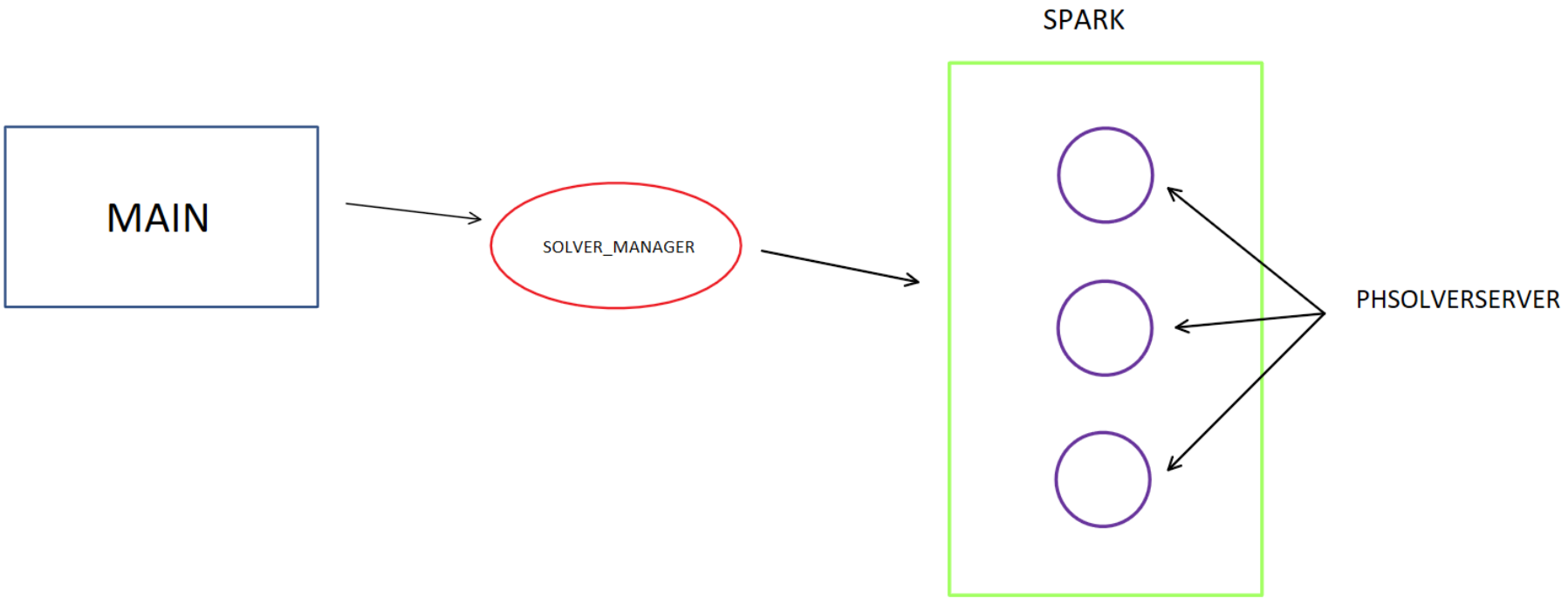
Proceso

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

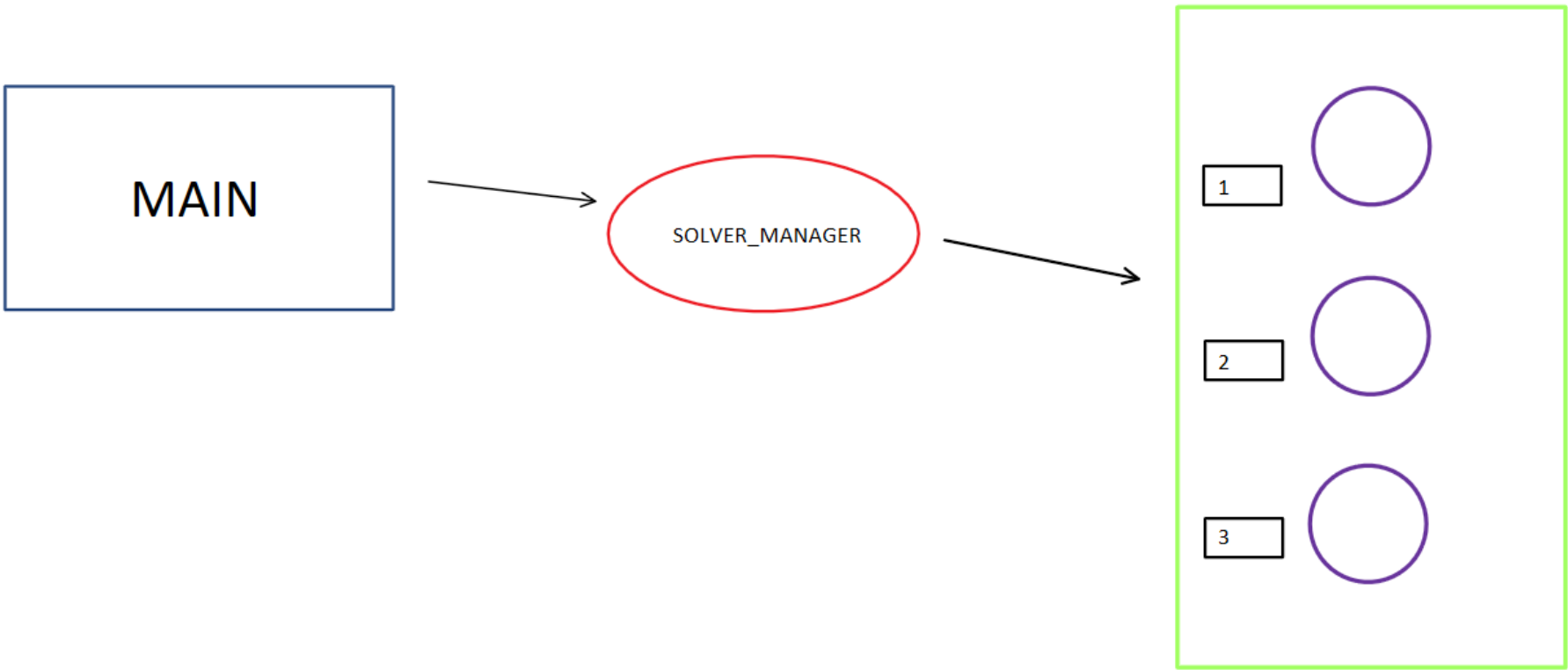
Funcionamiento-1



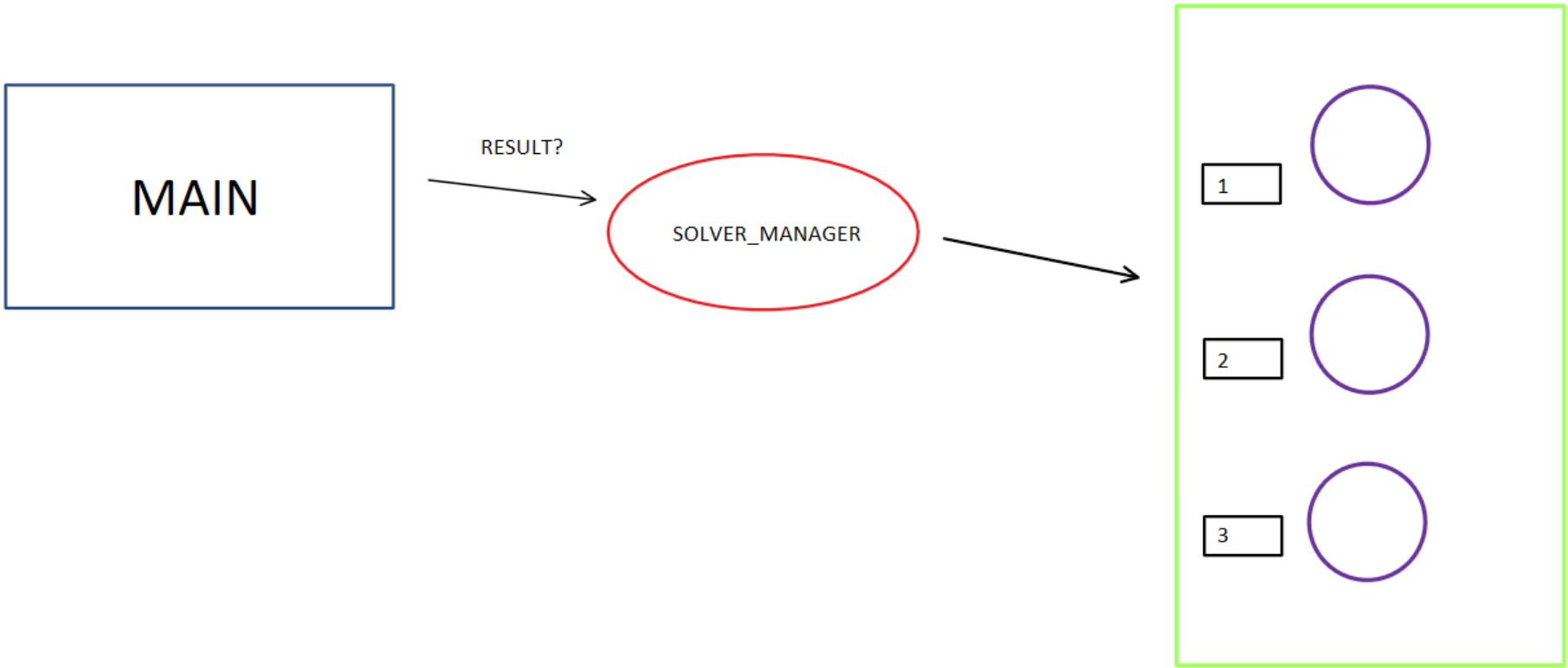
Funcionamiento-2



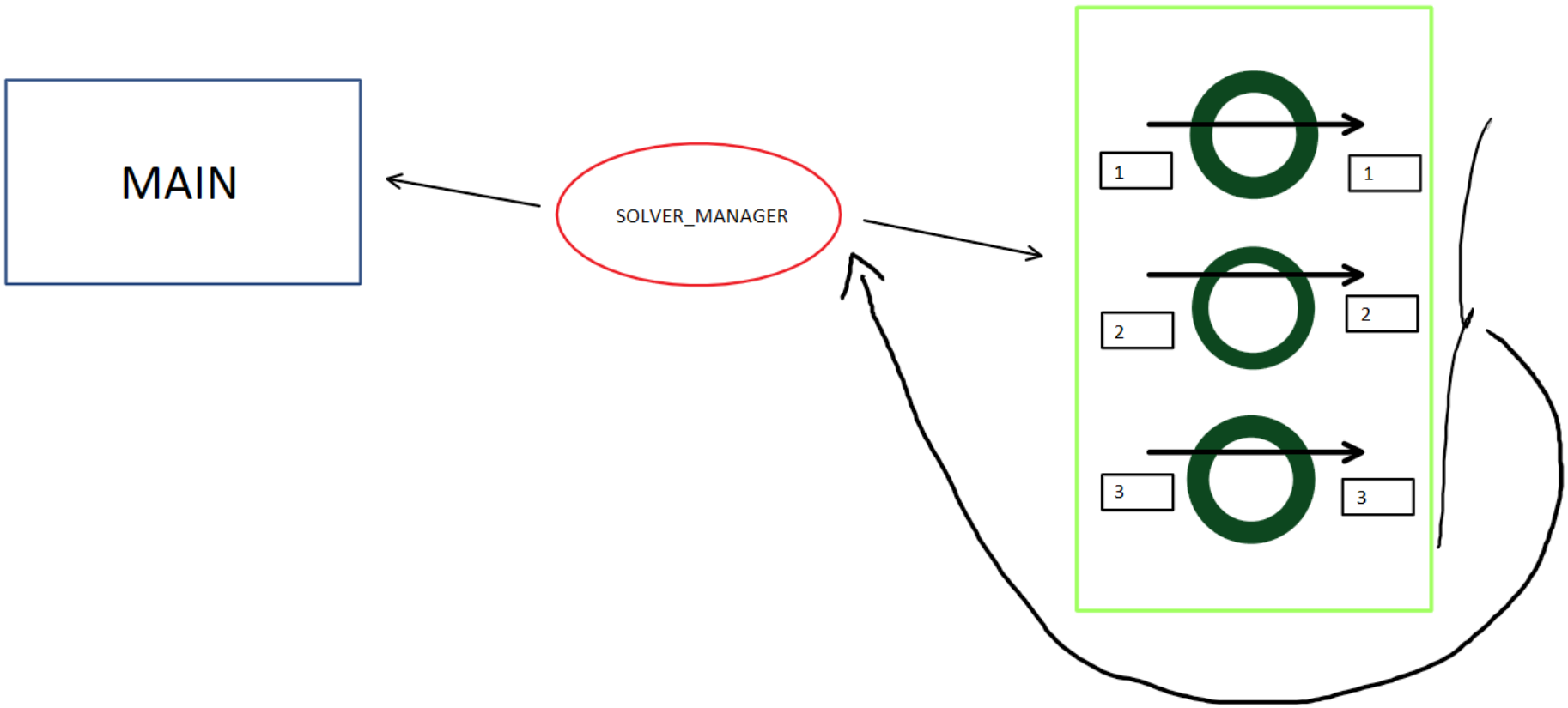
Funcionamiento-3



Funcionamiento-4



Funcionamiento-5



Pruebas

- Diseño de pruebas funcionales
  - Diferentes opciones de configuración que modifican la ejecución
  - Resolución de los ejemplos
- Diseño de pruebas de rendimiento

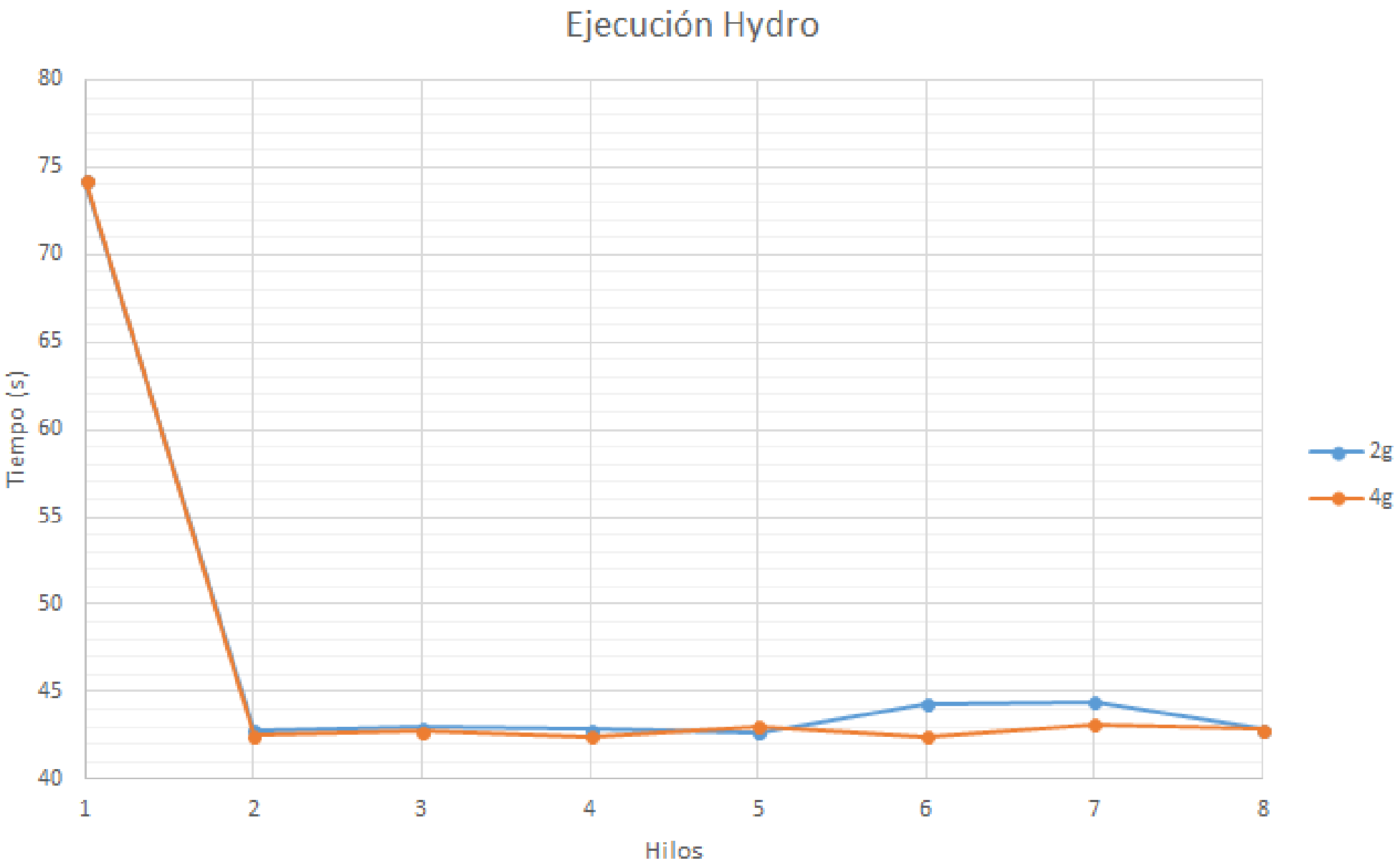
Recursos

Se probarán los ejemplos *Finance* y *Hydro*

Se prueba la escalabilidad en 1 y 8 hilos

Se medirá el tiempo total de ejecución y el tiempo medio por iteración

Resultados


Ejecución Hydro
Tiempo (s)
80
75
70
65
60
55
50
45
40
1
2
3
4
5
6
7
8
Hilos
2g
4g

Resultados

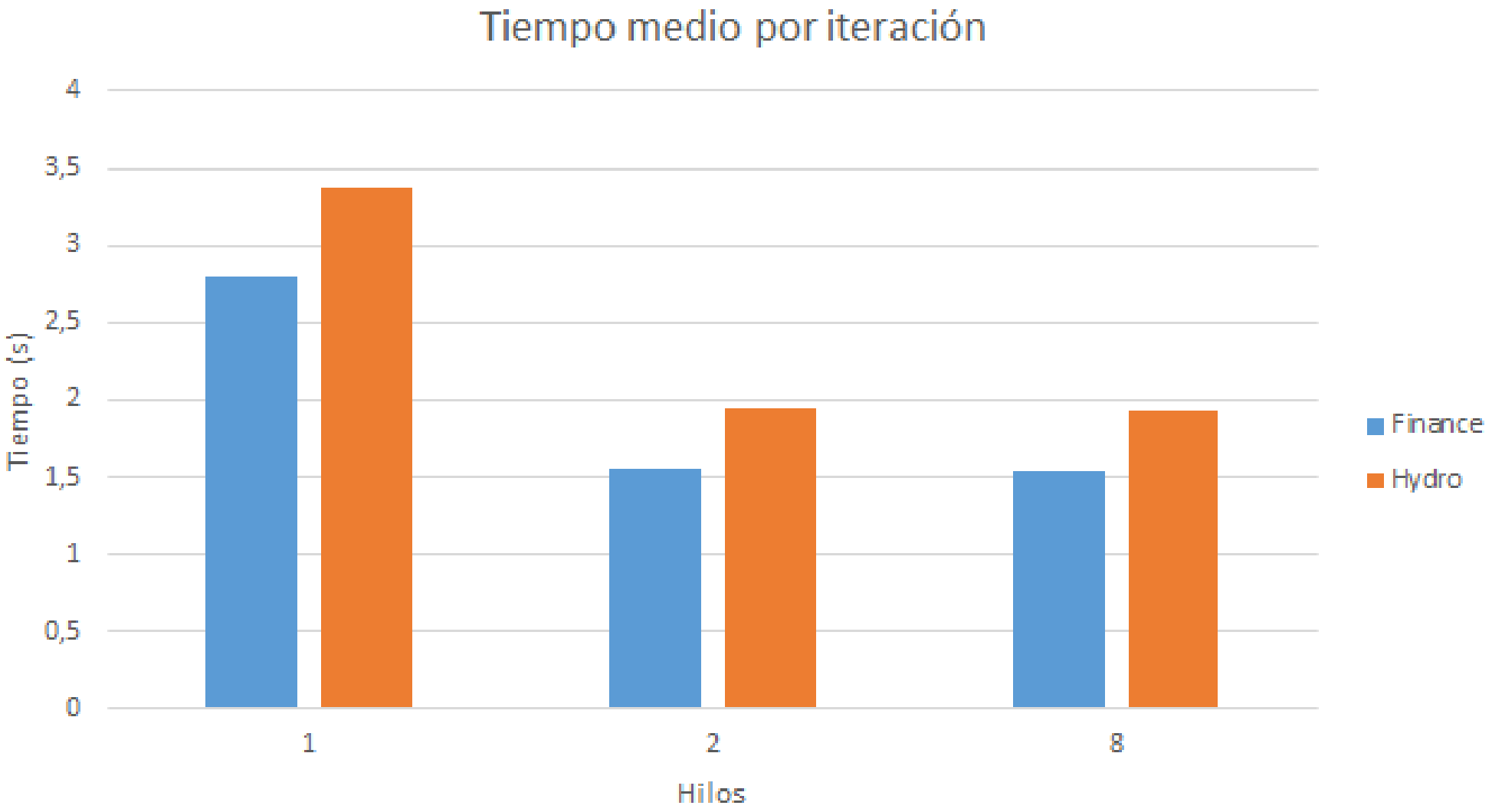


Figura: Tiempo medio por iteración

Conclusiones

- Extensibilidad de las herramientas Big Data y Spark en concreto

Conclusiones

- Extensibilidad de las herramientas Big Data y Spark en concreto
- Integración satisfactoria con Pyomo

Conclusiones

- Extensibilidad de las herramientas Big Data y Spark en concreto
- Integración satisfactoria con Pyomo
- Escalabilidad conseguida

Lecciones aprendidas

- Importancia de la planificación

Lecciones aprendidas

- Importancia de la planificación
  - Uso de un ciclo de vida adecuado
  - Tener en cuenta el tiempo de formación cuando se usan nuevas herramientas

Lecciones aprendidas

- Importancia de la planificación
  - Uso de un ciclo de vida adecuado
  - Tener en cuenta el tiempo de formación cuando se usan nuevas herramientas
- Adaptación de una arquitectura orientada a objetos a Spark

Lecciones aprendidas

- Importancia de la planificación
  - Uso de un ciclo de vida adecuado
  - Tener en cuenta el tiempo de formación cuando se usan nuevas herramientas
- Adaptación de una arquitectura orientada a objetos a Spark
- Herramientas de gestión de la configuración y planificación

Lecciones aprendidas

- Importancia de la planificación
  - Uso de un ciclo de vida adecuado
  - Tener en cuenta el tiempo de formación cuando se usan nuevas herramientas
- Adaptación de una arquitectura orientada a objetos a Spark
- Herramientas de gestión de la configuración y planificación

Trabajo futuro

- Pyomo es Open Source

Trabajo futuro

- Pyomo es Open Source
- Entorno de pruebas adecuado
- Mayor optimización en el uso de Spark

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

15 de septiembre de 2018