SOLICITUDE DE APROBACIÓN DE ANTEPROXECTO DO TRABALLO DE FIN DE GRAO GRAO EN ENXENARÍA INFORMÁTICA.

Entréguese na Administración da ETSE antes da data indicada no Artigo 15 do Regulamento de traballo fin de grao enxeñaría informática: **O Anteproxecto terá que ser aprobado pola Comisión de Traballos Fin de Carreira de Enxeñaría Informática como mínimo tres meses antes da data de depósito e de solicitude de trámite de defensa do TFG**

Datos do/a Alumno/a			
Nome: Cristofer Canosa Domínguez			**DNI:** 35592837S
Enderezo: Rua Francisco Romero Blanco nº4 3ºB			**Localidade:** Noia
Provincia: A Coruña	**C.P.:** 15200	**Teléfono:** 689032436	**Correo-e USC:** cristofer.canosa@rai.usc.es

Requisitos
O alumnado poderá ter pendentes como máximo 75 créditos para completar os estudos, excluídos os correspondentes ao TFG

Datos do Traballo de Fin de Grao	
Título: Paralelización del algoritmo Progressive Hedging para la resolución de problemas estocásticos	
Titor(a): Juan Carlos Pichel Campos	**Correo-e:** juancarlos.pichel@usc.es
Cotitor(a): Diego Rogríguez Martínez	**Correo-e:** diego.rogriguez@usc.es
Cotitor(a):	**Correo-e:**
Áreas de coñecemento: Arquitectura y Tecnología de Computadores	
Departamento: Electrónica e Computación	
Empresa (se procede):	

A persoa que asina e cos datos que se indican solicita á Comisión de Traballos Fin de Grao de Enxeñaría Informática a aprobación do anteproxecto que se acompaña.

Santiago de Compostela, 22 de Febrero de 2018

O/A alumno/a

Asdo: Cristofer Canosa Domínguez

Vº e Pr. O/A Titor(a) e Cotitor(a) do proxecto

Asdo:

Á atención da Comisión de Traballos Fin de Grao de Enxeñaría Informática

DESCRICIÓN DO ANTEPROXECTO

TÍTULO:

Paralelización del algoritmo Progressive Hedging para la resolución de problemas estocásticos.

Introdución

La paralelización de un código busca adaptar la ejecución secuencial de forma que se ejecuten varias instrucciones al mismo tiempo. Esto no solo reduce el tiempo si no que permite ejecutar algoritmos en clústeres de computación con múltiples nodos. De esta forma el código se puede escalar más fácilmente y, dado que podemos asignar un mayor número de recursos a su ejecución, nos permitirá abordar problemas de mayor tamaño en un tiempo razonable.

El proyecto consiste en la adaptación de un módulo de un proyecto existente para su ejecución paralela. En particular, se trabajará sobre el proyecto *Pyomo*[1], un paquete de software basado en Python destinado a la formulación y solución de modelos de optimización. Más concretamente adaptaremos una función del módulo *Pysp*, que es el paquete de *Pyomo* dedicado a la solución de programas estocásticos con múltiples fases y escenarios.

La *Programación Estocástica*[2] resuelve problemas de optimización donde existe un cierto grado de incertidumbre. Esta incertidumbre genera varios escenarios distintos para el problema que, además, pueden depender entre sí (en función del valor real sobre el que existía la incertidumbre). Un ejemplo sencillo de un problema de este tipo sería un plan de inversiones. Contamos con un presupuesto fijo al inicio y podemos invertir en las empresas A o B. Dentro de 3 años podremos invertir en las empresas C o D. Actualmente sabemos cuánto podemos invertir en A o B y una aproximación de lo que ganaremos en beneficio con cada una, pero no podemos asegurarlo. En 3 años tendremos escenarios distintos en función del beneficio que nos aporten A o B. Esto influirá en la inversión posterior a C o D. De esta forma tenemos múltiples escenarios interdependientes con un grado de incertidumbre.

Este tipo de problemas pueden subdividirse en problemas individuales (en función del escenario al que pertenecen) lo que los hace fácilmente paralelizables. Estos problemas individuales deben converger sobre una solución única, lo que se consigue mediante un algoritmo como el *Progressive Hedging*[3]. Este algoritmo hace uso de las variables individuales que afectan a cada escenario y mediante un valor *rho* las hace converger a un valor único, equivalente a la resolución del problema sin subdividirlo.

Obxectivos

El objetivo principal del TFG es el de paralelizar el algoritmo *Progressive Hedging* del módulo Pysp. Este objetivo principal puede subdividirse en los siguientes objetivos específicos:

- Estudiar y analizar el funcionamiento del algoritmo *Progressive Hedging* en Pysp.
- Análisis de las diferentes alternativas de paralelización disponibles que mejor se adapten al problema. Se tendrán en cuenta tecnologías Big Data (Apache Spark[4]) o modelos tradicionales (MPI[5]).
- Diseño e implementación del nuevo módulo e integración con Pyomo.
- Análisis y evaluación del rendimiento.

Descrición técnica

El resultado final del proyecto será un módulo de ejecución de trabajos escrito en Python integrado en Pyomo que pueda sustituir a la función PySP original.

Comenzaremos analizando el código Python actual en detalle. Debemos conocer el funcionamiento del algoritmo secuencial para identificar cómo se crean y ejecutan las distintas tareas del algoritmo. En Pysp, para la resolución del problema, se ejecutan múltiples etapas para analizar la entrada, establecer variables de control, etc. Una vez definidas las entradas y salidas de cada tarea, así como la comunicación entre las mismas (por ejemplo, mediante variables de control) tendremos una idea clara de todos los componentes que intervienen en la resolución del problema. Para este estudio haremos uso de los ejemplos contenidos en el repositorio de Pyomo. Algunos de estos ejemplos se pueden ejecutar de forma paralela haciendo uso de *Pyro*[6], una librería de Python para el uso de objetos remotos.

Una vez conocida la estructura del algoritmo estudiaremos las herramientas disponibles para su implementación en paralelo. Consideraremos tecnologías Big Data como Spark, que nos permitirá gestionar de forma sencilla la gestión y ejecución de tareas de forma distribuida sobre un clúster de computación. También facilita la manipulación de grandes cantidades de datos, permitiendo abordar problemas de mayor tamaño. En caso de que las tecnologías Big Data no se adaptasen de forma satisfactoria al problema, se estudiarán métodos tradicionales como MPI, haciendo uso de objetos remotos y comunicación mediante paso de mensajes.

Tras analizar las alternativas disponibles para la implementación paralela se establecerá un diseño para aprovechar al máximo las características de la herramienta escogida, con la finalidad de conseguir el mejor rendimiento en términos de tiempo de ejecución y escalabilidad.

Una vez se implemente este diseño, se realizará una extensiva batería de pruebas tanto funcionales como de rendimiento. Comenzaremos generando un plan de pruebas que permita evaluar que el software cumple con los requisitos establecidos para su adecuado funcionamiento. Debemos asegurar también una correcta cobertura del código implementado. Cuando se asegure que el software funciona adecuadamente, pasaremos a las pruebas de rendimiento. Se diseñarán unas pruebas que permitan evaluar correctamente el rendimiento del nuevo software, así como de la versión secuencial con la finalidad de comparar las ganancias.

Los datos de rendimiento recabados en la fase de pruebas, pudiendo ser ampliados si se considera necesario, serán recopilados en un informe final detallando las mejoras conseguidas con respecto a la implementación original.

Medios materiais necesarios.

- **Ordenador de desarrollo**. Con sistema Windows y Linux, haciendo uso del siguiente software:
  - **PyCharm:** Para el desarrollo y pruebas del software en Python.
  - **Editor de texto:** Para la documentación del proyecto, en Markdown y LaTex.
  - **Trello:** Para la gestión de las tareas.
  - **Python 2.7:** Versión del intérprete sobre la que trabajaremos.
- **Clúster de pruebas**. Donde se realizarán las pruebas de rendimiento y escalabilidad de la implementación paralela.

Bibliografía.

[1] Pyomo: https://github.com/Pyomo/pyomo {última consulta 16/02/18}
[2] Programación Estocástica: John R. Birge and François V. Louveaux. *Introduction to Stochastic Programming*. Springer Verlag, New York, 1997.
[3] Progressive Hedging: Watson, Jean-Paul and Woodruff, David L. and Strip, David R., *Progressive Hedging Innovations for a Class of Stochastic Resource Allocation Problems* (September 15, 2008). UC Davis Graduate School of Management Research Paper No. 05-08. Available at SSRN: https://ssrn.com/abstract=1268385 or http://dx.doi.org/10.2139/ssrn.1268385 {última consuta 16/02/18}
[4] Apache Spark: http://spark.apache.org/docs/latest/ {última consulta 20/02/18}
[5] MPI *Message Passing Interface*: http://mpi-forum.org/ {última consulta 20/02/18}
[6] Pyro *Python Remote Objects*: https://pypi.python.org/pypi/Pyro4 {última consulta 20/02/18}

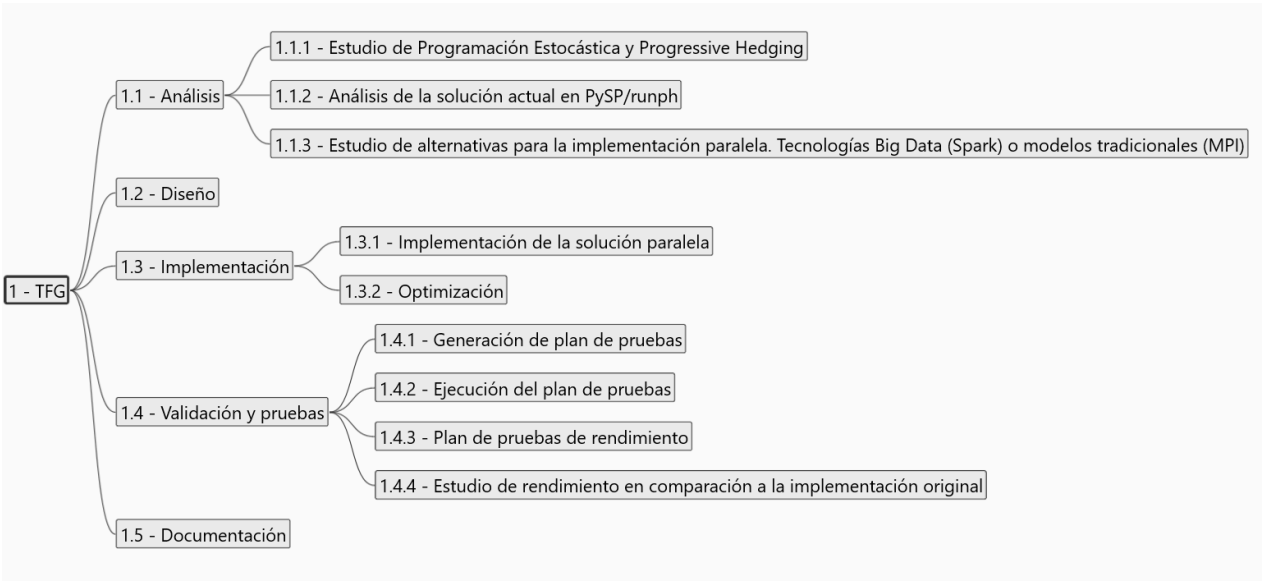
Observacións.

FASES DO TRABALLO E ESTIMACIÓN TEMPORAL

Un traballo de fin de grao suporá 401,25 horas de traballo autónomo do alumnado e 11,25 horas de traballo presencial (titorías e avaliación).

Dedicación semanal prevista (en horas/semana): _26_

Tarea	**Estimación temporal** (en semanas)
Análisis	3,5 semanas
Diseño del nuevo módulo	4 semanas
Implementación	4 semanas
Validación y pruebas	2 semanas
Generación de la documentación	2 semana


1 - TFG
1.1 - Análisis
1.1.1 - Estudio de Programación Estocástica y Progressive Hedging
1.1.2 - Análisis de la solución actual en PySP/runph
1.1.3 - Estudio de alternativas para la implementación paralela. Tecnologías Big Data (Spark) o modelos tradicionales (MPI)
1.2 - Diseño
1.3 - Implementación
1.3.1 - Implementación de la solución paralela
1.3.2 - Optimización
1.4 - Validación y pruebas
1.4.1 - Generación de plan de pruebas
1.4.2 - Ejecución del plan de pruebas
1.4.3 - Plan de pruebas de rendimiento
1.4.4 - Estudio de rendimiento en comparación a la implementación original
1.5 - Documentación