ID	**PR - 07.04**
Fase	Análisis
Cumplimiento	100%

Objetivos de la semana

En esta primera fase se ha realizado un estudio del código, concretamente del módulo "*pysp*" para entender su funcionamiento.

Comenzamos por un estudio teórico del algoritmo, seguido de un análisis del código que, finalmente, se realizó apoyado en una traza resultante de ejecutar el ejemplo *farmer*.

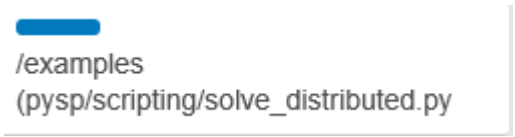
Tareas realizadas

Análisis general

Inicialmente se realizó una inspección general del código, apuntando métodos u objetos relevantes para un posterior análisis en mayor profundidad.

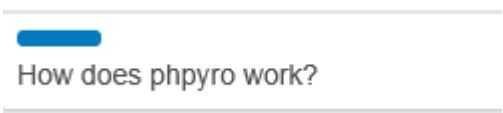
> **Ver:** */Documentación/Notas/Análisis/Análisis general.pdf*

Ejemplo de ejecución distribuida



Se estudió este ejemplo como posible ayuda de la implementación paralela del algoritmo. Sin embargo no funciona de la misma forma que utilizando el solver manager phpyro. Por lo tanto, fue descartado.

Funcionamiento phpyro



Este solver manager es el encargado de distribuir el trabajo entre varios objetos *phsolverserver*. Funciona utilizando una cola de acciones que son transmitidas a los objetos distribuidos con *Pyro*.

Método solve del objeto PH

Este es el método principal del algoritmo. Como expresión formal, se ejecuta lo siguiente:

$$k := 0$$

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

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

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

$$k := k + 1$$

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

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

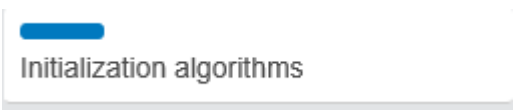
$$\Delta s \in S, w_s^{(k)} := w_s^{(k-1)} + \rho(x_s^{(k)} - \hat{x}^{(k)})$$

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

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

Este método se apoya en métodos auxiliares, donde el principal es *queue_subproblems*, donde se ejecuta la solución para cada escenario.

Algoritmos de inicialización



Estudiando la traza generada por la ejecución del ejemplo *farmer*, vemos que se utiliza el contexto *PHFromScratchManagedContext*, localizado en *phinit.py::PHFromScratch*.

Esta inicialización se divide en tres fases:

1. *ScenarioTreeInstanceFactory*, que genera cada escenario.

2. *GenerateScenarioTreeForPH*, que genera el árbol completo.
3. *PHAlgorithmBuilder*, que genera el objeto PH y otros objetos asociados como solver manager, solver server, etc.

> **Ver:** */Documentación/Notas/Análisis/Salida verbose inicialización.pdf*

Propuesta de solución inicial

Generar un esquema de funcionamiento general del algoritmo en paralelo. Adicionalmente se identifican objetivos susceptibles a ser modificados en nuestra implementación.

> **Ver:** */Documentación/Análisis/Funcionamiento PH paralelo.pdf

Notas

El análisis inicial se acabó el día 07/04/2018, con dos semanas de retraso sobre la planificación inicial.