MCLP: BSG+SC

> doc · tabla

Implementación de BinPacking en python

La idea es modular algoritmo e implementar la resolución del bin packing usando *python* y *Jupyter notebooks*.
El objetivo es facilitar la edición, modificación y pruebas de este módulo.

A grandes rasgos el algoritmo hace los siguiente:

1. Generación de bins iniciales usando **BSG**
2. Selección de bin a desarmar y almacenar cajas en C
3. Mientras C no quede vacío o máximo de iteraciones:
   a. Seleccionar caja c de C
   b. Seleccionar bin de destino B
   c. Usar **BSG** para generar bin B' usando cajas $B \cup \{c\}$, priorizando c. Es posible que **BSG** retorne conjunto de cajas residuales R
   d. Si R es mejor que c, B se reemplaza por B' en el conjunto de bins y $C \leftarrow C \cup R$
4. Volver a 2 (seleccionar otro bin para desarmar)

Comunicación con solver BSG

El solver BSG debería realizar dos tareas principales:

Generación de bins iniciales
Se podría implementar un solver básico para MCLP (`basicsolver_mclp.cpp`). Lo que haría este solver sería:

- cargar la instancia desde un archivo en el servidor
- generar bins sin repetir cajas (como ya lo hace una de las funciones del solver)

- retornar por salida estándar la lista de bins generados con los ids de cajas y sus porcentajes de llenado (para ser leído usando *python*)

Los parámetros pueden ser los que tienen relación con la función heurística, la generación de bloques (`min_fr` , `max_blocks`), `n_beams` , y quizás alguno que permita randomizar un poco la generación de bins.

BSG priorizando subconjunto de cajas
Idealmente, el solver puede quedar escuchando instrucciones usando el protocolo TCP.
Si es así, el solver no necesitaría volver a cargar la instancia cada vez que es utilizado.

Para hacerlo, en el main se puede implementar una función `listen_instruction()` , la cuál se quede escuchando un puerto específico y retorne el string de instrucciones que reciba.
Aquí puedes ver como implementar un socket en c++ (parte del servidor).

Este string podría tener un formato como este:

```
generate_bin [lista_id_cajas] [lista_cajas_prioritarias]
```

Luego, el string es leído y delegado a la función correspondiente del solver.
Finalmente, el solver debería retornar (por el mismo puerto) una la lista de cajas residuales.

Para enviar instrucciones al servidor por consola:

```
echo <instruccion> | netcat localhost <puerto>
```