Dual-Based Procedure and Subgradient method implementations

Di Battista Mattia

Università degli Studi di Roma "Tor Vergata"

July 16, 2022

Outline

1. Goal
2. Methodology
   - Primal Simplex
   - Dual-Based Procedure
   - Subgradient method
3. Performance Evaluation
   - Dualoc & Linear relaxation
   - Lagrangian relaxation
4. Conclusion

Outline

Goal

The aim is to implement and evaluate:

- **Dual-Based Procedure** for *Uncapacitated Facility Location* (UFL)
- **Subgradient method** for *Lagrangian relaxation*

Goal

The aim is to implement and evaluate:

- **Dual-Based Procedure** for *Uncapacitated Facility Location* (UFL)
- **Subgradient method** for *Lagrangian relaxation*

Using:

- w.r.t. *Linear relaxation* and *Primal Simplex*
- *Python* as programming language
- *Gurobi API*

Outline

Outline

Primal Simplex

Main commands used:

- to set the *Primal Simplex* as solution algorithm:

```
self.model.Params.Method = 0
```

- to disable presolve, cut generation and heuristics options:

```
self.model.setParam(gp.GRB.Param.Presolve, 0)
self.model.setParam(gp.GRB.Param.Heuristics, 0)
self.model.setParam(gp.GRB.Param.Cuts, 0)
```

- to use *Linear relaxation*:

```
self.model.relax()
```

Outline

Dual-Based Procedure (1)

Uncapacitated Facility Location **formulation**:

$$
\begin{aligned}
\min \;\; z = \sum_{u\in U}\sum_{v\in V} c_{vu} y_{vu} + \sum_{u\in U} f_u x_u & & & \\
\sum_{v\in V} y_{vu} &= 1 & & v\in V \\
x_u - y_{vu} &\geq 0 & & u\in U, v\in V \\
x_u \in \{0,1\}, y_{vu} \in \{0,1\} & & & u\in U, v\in V
\end{aligned}
$$

Uncapacitated Facility Location **dual formulation**:

$$
\begin{aligned}
\max \;\; g(z) = \sum_{v\in V} z_v & & & \\
z_v - w_{vu} &\leq c_{vu} & & u\in U, v\in V \\
\sum_{v\in V} w_{vu} &\leq f_u & & u\in U \\
w_{vu} &\geq 0 & & u\in U, v\in V
\end{aligned}
$$

Dual-Based Procedure (2)

Dualoc-Based procedure (created by *D. Erlenkotter* in *1978*)

- starting from the dual problem
- provides a **lower bound** for the primal problem
- high quality and efficiently
- cons: influenced by the sorting of variables

Algorithm 1

1: **for** $v \in V$ **do**
2: $\quad \bar{z}_v = min_{u \in U}\{c_{uv}\}$
3: **end for**
4: **for** $s \in V$ **do**
5: $\quad \bar{w}_{uv} = max\{0, \bar{z}_v - c_{vu}\}$
6: $\quad z_s^{max} = min_{u \in U}\{c_{su} + f_u - \sum_{v \neq s} \bar{w}_{vu}\}$
7: **end for**
8: **return** z^{max}

Outline

Subgradient method (1)

Uncapacitated Facility Location **formulation**:

$$
\begin{aligned}
\min \ z = \sum_{u\in U}\sum_{v\in V} c_{vu}y_{vu} + \sum_{u\in U} f_u x_u \\
\sum_{v\in V} y_{vu} &= 1 \quad v\in V \\
x_u - y_{vu} &\geq 0 \quad u\in U, v\in V \\
x_u \in \{0,1\}, y_{vu} \in \{0,1\} & \qquad u\in U, v\in V
\end{aligned}
$$

Uncapacitated Facility Location **Lagrangian relaxation formulation**:

$$
\begin{gathered}
min \ z = \textstyle\sum_{u\in U}(f_u - \sum_{v\in V}\lambda_{uv})x_u + \sum_{u\in U}\sum_{v\in V}(c_{uv}+\lambda_{uv})y_{uv} \\
\textstyle\sum_{u\in U} y_{uv} = 1, \ \forall v \in V \\
x, y \in \{0,1\}
\end{gathered}
$$

Subgradient method (2)

Subgradient method provides a vector multiplier for *Lagrangian relaxation*

- proved its convergence to the optimal *Lagrangian* multiplier
- considering a feasible solution of the *UFL* problem (i.e., *B*)

Algorithm 2

1: find $x^{(h)}$ in *Lagrangian* problem
2: $s^{(h)} = b - Ax^{(h)}$
3: **if** $s^{(h)} == 0$ **then**
    STOP
4: **end if**
5: $\theta^{(h)} = \frac{B - L(\lambda^{(h-1)})}{||s^{(h)}||^2}$
6: $\lambda^{(h+1)} = \lambda^{(h)} + \theta^{(h)} \frac{s^{(h)}}{||s^{(h)}||}$
7: $h = h + 1$
8: **if** lower bound did not enhance in the last K iterations **then**
    STOP
9: **end if**

Subgradient method (3)

Assumptions in the implementation:

- started from a *Lagrangian relaxation* solution
  - randomly generated multiplier
- iterations (i.e., K) equal to *100*
- B equals to the *Simplex Primal* solution
  - i.e., the best solution

```
self.model.setObjective(obj_func, gp.GRB.MINIMIZE)
self.model.optimize()
return self.model.objVal
```

Outline

1. Goal
2. Methodology
3. Performance Evaluation
    - Dualoc & Linear relaxation
    - Lagrangian relaxation
4. Conclusion

Performance Evaluation (1)

Parameters considered in the **tests**:

- $(customers; facilities) \in \{(30; 30), (29; 5), (5; 29)\}$
- $algorithm \in$
  $\{Dualoc, Linear\ relaxation, Lagrangian\ relaxation\}$
- ratios:
  - $setup\ cost \approx shipping\ cost$
  - $setup\ cost \gg shipping\ cost$
  - $setup\ cost \ll shipping\ cost$
  - $Var(setup\ cost) \gg Var(shipping\ cost)$
  - $Var(setup\ cost) \ll Var(shipping\ cost)$
- trials equal to *100*

Performance Evaluation (1)

Parameters considered in the **tests**:

- $(customers; facilities) \in \{(30; 30), (29; 5), (5; 29)\}$
- $algorithm \in \{Dualoc, Linear\ relaxation, Lagrangian\ relaxation\}$
- ratios:
  - $setup\ cost \approx shipping\ cost$
  - $setup\ cost \gg shipping\ cost$
  - $setup\ cost \ll shipping\ cost$
  - $Var(setup\ cost) \gg Var(shipping\ cost)$
  - $Var(setup\ cost) \ll Var(shipping\ cost)$
- trials equal to *100*

4500 **instances** resolved!

Performance Evaluation (2)

Generated *15* **charts**:

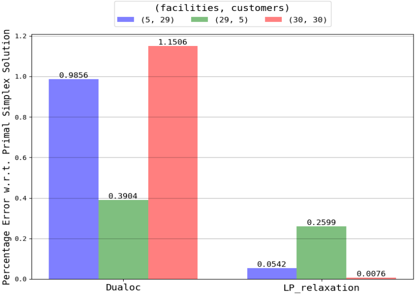
1. computation time
2. percentage error w.r.t *Primal Simplex* solution:

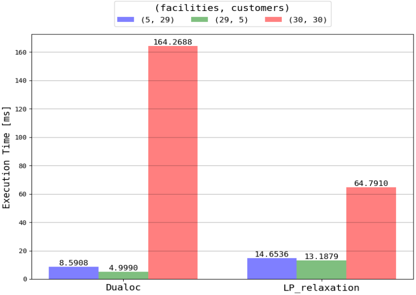
$$\frac{|Experimental\ \ Value - Theoretical\ \ Value|}{Theoretical\ \ value} \cdot 100 \qquad (1)$$

Outline

Dualoc & Linear relaxation (1)

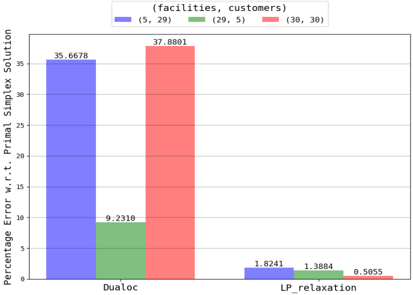
Test Case 1: $setup\ cost \approx shipping\ cost$

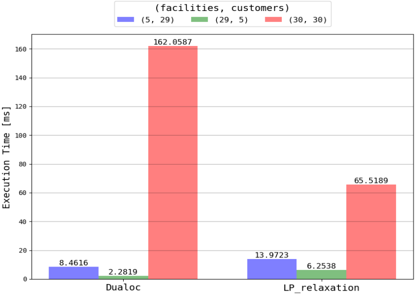




Dualoc & Linear relaxation (2)

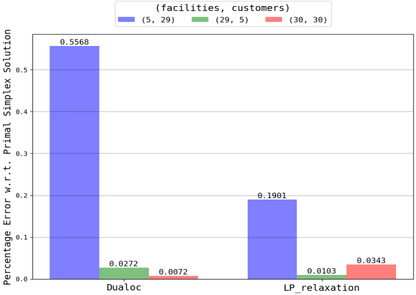
Test Case 2: $setup\ cost \gg shipping\ cost$

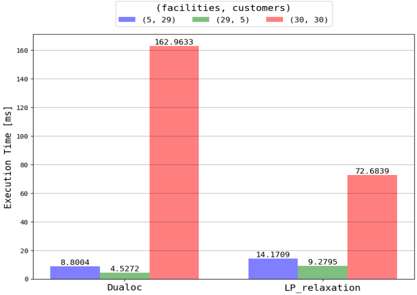




Dualoc & Linear relaxation (3)

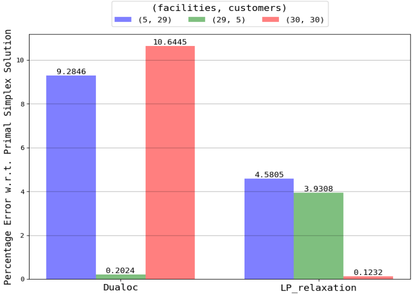
Test Case 3: *setup cost* $\ll$ *shipping cost*

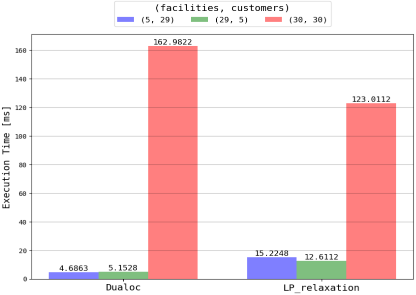




Dualoc & Linear relaxation (4)

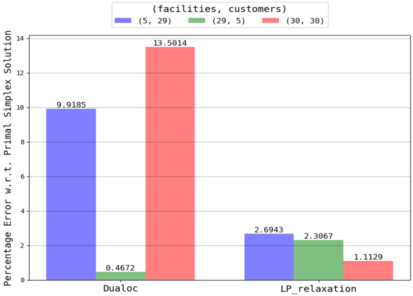
Test Case 4: $Var(setup\ cost) \gg Var(shipping\ cost)$

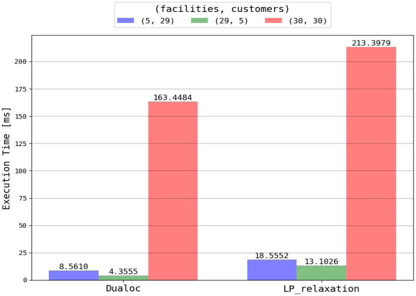




Dualoc & Linear relaxation (5)

Test Case 5: $Var(setup\ cost) \ll Var(shipping\ cost)$

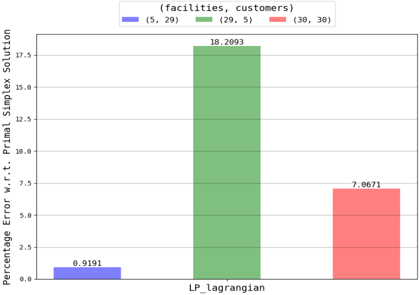


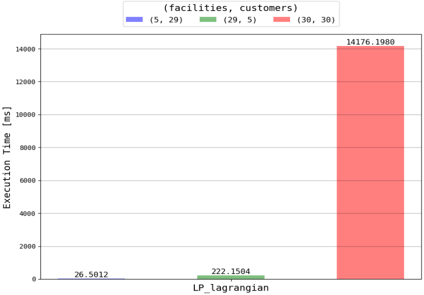


Outline

Lagrangian relaxation (1)

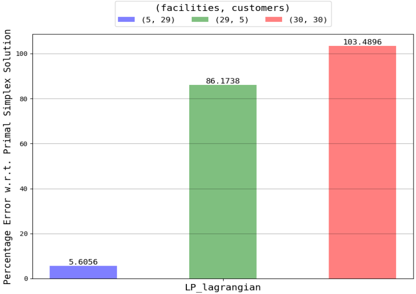
Test Case 1: $setup\ cost \approx shipping\ cost$

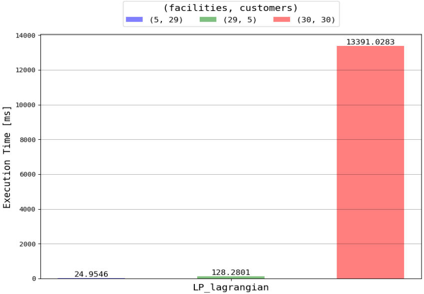




Lagrangian relaxation (2)

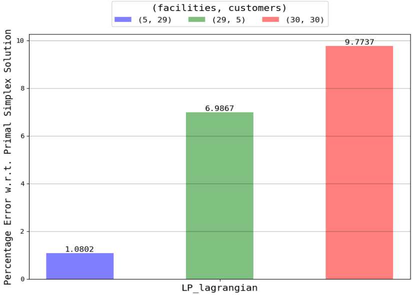
Test Case 2: $setup\ cost \gg shipping\ cost$

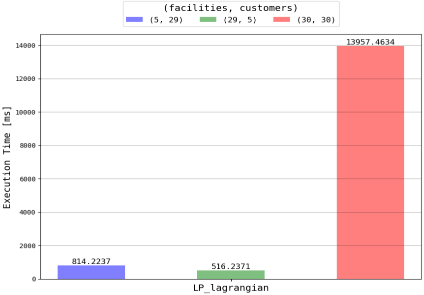




Lagrangian relaxation (3)

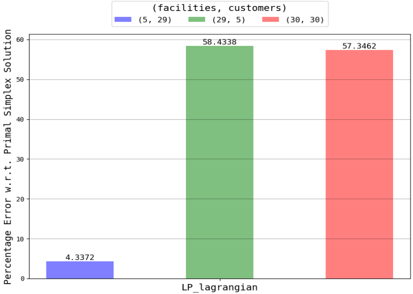
Test Case 3: $setup\ cost \ll shipping\ cost$

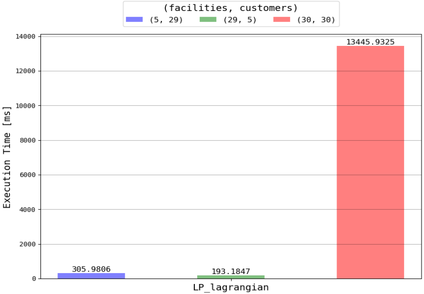




Lagrangian relaxation (4)

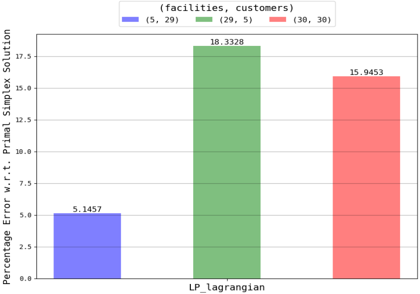
Test Case 4: $Var(setup\ cost) \gg Var(shipping\ cost)$

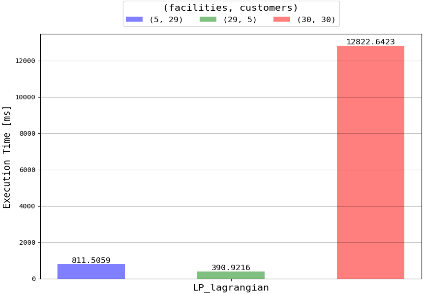




Lagrangian relaxation (5)

Test Case 5: $Var(setup\ cost) \ll Var(shipping\ cost)$





Outline

Conclusion (1)

Best algorithms in **percentage errors**:

	(5, 29)	(29, 5)	(30, 30)
Test case 1	*Linear r.*	*Linear r.*	*Linear r.*
Test case 2	*Linear r.*	*Linear r.*	*Linear r.*
Test case 3	*Linear r.*	*Linear r.*	*Dualoc*
Test case 4	*Lagrangian r.*	*Dualoc*	*Linear r.*
Test case 5	*Linear r.*	*Dualoc*	*Linear r.*

- *11* times *Linear relaxation* provides the minimum error
- *3* times *Dualoc* provides the minimum error
- *1* times *Lagrangian relaxation* provides the minimum error

Conclusion (2)

Best algorithms in **computation times**:

	(5, 29)	(29, 5)	(30, 30)
Test case 1	*Dualoc*	*Dualoc*	*Linear r.*
Test case 2	*Dualoc*	*Dualoc*	*Linear r.*
Test case 3	*Dualoc*	*Dualoc*	*Linear r.*
Test case 4	*Dualoc*	*Dualoc*	*Linear r.*
Test case 5	*Dualoc*	*Dualoc*	*Dualoc*

- *4* times *Linear relaxation* provides the minimum time
- *11* times *Dualoc* provides the minimum time