May Update

Budget Cut Algorithm, Code Improvements, and Previous Work

Dane Lacey, Bismark Singh

FAU Erlangen-Nuremberg

May 28, 2022

Introduction

- HiWi - Chair of Economic Mathematics
- MSc. Computational and Applied Mathematics
- Past, Present, and Future work and contributions

Past Work - (Dec., Feb., Mar.)

- Getting up to speed with FINE
  - Literature review (Oliver's thesis, Lara's thesis, etc.)
  - Hackathons
  - Running ref/flex/selfScen models
  - Gurobi naïve solver, optimality parameters, and interpreting output
- Working with Jureca
  - Running python scripts remotely
  - Getting used to CentOS and Linux terminal

JÜLICH FORSCHUNGSZENTRUM FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Past Work - (Dec., Feb., Mar.)

- Understanding Oliver's contributions
  - Broad strokes of the Budget Cut algorithm (BCA)
  - Code Improvements
  - Replicating results on both laptop and Jureca
  - Help from Andreas Smolenko:
    - Introduction to batch scripts and utilization of compute nodes
    - Develop workflow with MPI to run multiple models in parallel
- Understanding performance of BCA
  - Poor performance on ref/flexScen
  - Analyze .mps files and logs, for both MIP and relaxed LPs
    - BCA hinges on MIP being **slower** than LPs
    - This should always be the case if code is sufficiently improved

JÜLICH FORSCHUNGSZENTRUM
FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Past Work - Constraint Reduction

Table 1: refScen

(Regions,Days)	LP solve time	MIP solve time	# binary vars	# binary vars after presolve
(5,45)	19725.5090	11444.2192	14	7
(5,50)	19396.6607	12444.0838	14	7
(10,20)	11487.9579	13905.8330	36	18
*(10,25)	19582.8426	21352.2400	36	18
(15,10)	2540.6025	1482.2447	56	28

Table 2: flexScen

(Regions,Days)	LP solve time	MIP solve time	# binary vars	# binary vars after presolve
(5,45)	1149.0005	11545.5334	24	17
(5,50)	1121.1858	15203.0333	24	17
(10,20)	959.1568	15397.7304	56	38
(10,25)	1197.8699	17570.1542	56	38
(15,10)	202.0206	20538.5181	86	58

Table 3: selfScen

(Typical days, Year)	LP solve time	MIP solve time	# binary vars	# binary vars after presolve
(56,95)	65.1904	3718.6419	5	5
(56,96)	68.1824	3458.1068	5	5
(56,97)	63.6835	3464.2057	5	5
(56,98)	71.3745	3417.8257	5	5
(56,99)	44.4653	2815.3125	5	5

Past Work - Constraint Reduction

- Identify and correct duplicate symmetrical capacity constraints

$$\mathrm{cap}^{comp}_{(loc_i,loc_j)} = \mathrm{cap}^{comp}_{(loc_j,loc_i)}, \quad \forall comp \in \mathcal{C}^{trans} \times loc \in \mathcal{L}^{tran}$$

- For example, take $comp = DCLines$ with 5 clustered regions
  - Assuming eligibility everywhere, we have the following constraints:

$$\begin{aligned}
&\mathrm{cap}^{DC\ Lines}_{(\mathrm{loc}_0,\mathrm{loc}_1)} = \mathrm{cap}^{DC\ Lines}_{(\mathrm{loc}_1,\mathrm{loc}_0)}, \quad &&\mathrm{cap}^{DC\ Lines}_{(\mathrm{loc}_1,\mathrm{loc}_0)} = \mathrm{cap}^{DC\ Lines}_{(\mathrm{loc}_0,\mathrm{loc}_1)},\\
&\mathrm{cap}^{DC\ Lines}_{(\mathrm{loc}_1,\mathrm{loc}_2)} = \mathrm{cap}^{DC\ Lines}_{(\mathrm{loc}_2,\mathrm{loc}_1)}, \quad &&\mathrm{cap}^{DC\ Lines}_{(\mathrm{loc}_2,\mathrm{loc}_1)} = \mathrm{cap}^{DC\ Lines}_{(\mathrm{loc}_1,\mathrm{loc}_2)},\\
&\mathrm{cap}^{DC\ Lines}_{(\mathrm{loc}_3,\mathrm{loc}_2)} = \mathrm{cap}^{DC\ Lines}_{(\mathrm{loc}_2,\mathrm{loc}_3)}, \quad &&\mathrm{cap}^{DC\ Lines}_{(\mathrm{loc}_2,\mathrm{loc}_3)} = \mathrm{cap}^{DC\ Lines}_{(\mathrm{loc}_3,\mathrm{loc}_2)},
\end{aligned}$$

$$\vdots$$

Past Work - Constraint Reduction

- Identify and correct duplicate symmetrical capacity constraints

$$\mathrm{cap}^{comp}_{(loc_i,loc_j)} = \mathrm{cap}^{comp}_{(loc_j,loc_i)}, \quad \forall comp \in \mathcal{C}^{trans} \times loc \in \mathcal{L}^{tran}$$

		cluster_j				
		0	1	2	3	4
cluster_i	0	~~(0,0)~~	(0,1)	(0,2)	(0,3)	(0,4)
	1	(1,0)	~~(1,1)~~	(1,2)	(1,3)	(1,4)
	2	(2,0)	(2,1)	~~(2,2)~~	(2,3)	(2,4)
	3	(3,0)	(3,1)	(3,2)	~~(3,3)~~	(3,4)
	4	(4,0)	(4,1)	(4,2)	(4,3)	~~(4,4)~~

Figure: Before improvement

		cluster_j				
		0	1	2	3	4
cluster_i	0	~~(0,0)~~	(0,1)	(0,2)	(0,3)	(0,4)
	1	~~(1,0)~~	~~(1,1)~~	(1,2)	(1,3)	(1,4)
	2	~~(2,0)~~	~~(2,1)~~	~~(2,2)~~	(2,3)	(2,4)
	3	~~(3,0)~~	~~(3,1)~~	~~(3,2)~~	~~(3,3)~~	(3,4)
	4	~~(4,0)~~	~~(4,1)~~	~~(4,2)~~	~~(4,3)~~	~~(4,4)~~

Figure: After improvement

- For n regions, $(n^2 - n)/2$ possible constraints eliminated per comp.
  - The exact number depends on where the given component is eligible

JÜLICH FORSCHUNGSZENTRUM
FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Past Work - Constraint Reduction

```
def symmetricalCapacity(self, pyM):
    """
    Ensure that the capacity between location_1 and location_2 is the same as the one
    between location_2 and location_1.

    :param pyM: pyomo ConcreteModel which stores the mathematical formulation of the model.
    :type pyM: pyomo ConcreteModel
    """
    compDict, abbrvName = self.componentsDict, self.abbrvName
    capVar, capVarSet = getattr(pyM, 'cap_' + abbrvName), getattr(pyM, 'designDimensionVarSet_' + abbrvName)

    def symmetricalCapacity(pyM, loc, compName):
        return capVar[loc, compName] == capVar[compDict[compName]._mapI[loc], compName]

    setattr(pyM, 'ConstrSymmetricalCapacity_' + abbrvName, pyomo.Constraint(capVarSet, rule=symmetricalCapacity))
```

- 3 implementations ideas
  - Imp. 1: Check compDict keys per component to see if cluster (i,j) present, if so, apply Constraint.Skip
  - Imp. 2: Apriori make a reduced dict of clusters for compDict to read
  - Imp. 3: Create a subset of capVarSet where j > i, and send that to setattr

Past Work - Constraint Reduction

Table 1: refScen - Time to establish symmetrical constraints (LOPF + Transmission)

(Regions,Days)	Naïve	Imp. 1	Imp. 2	Imp. 3	# rows original	# rows w/ fix
(10,10)	1.543e-3	1.929e-3	1.243e-3	3.380e-3	291972	291917
(10,20)	2.775e-3	2.074e-3	1.392e-3	5.482e-3	494532	494477
(15,10)	2.253e-3	3.253e-3	2.087e-3	6.826e-3	423269	423181
(15,20)	2.354e-3	3.066e-3	1.921e-3	5.186e-3	720629	720541

- Imp. 1: Readable and relatively simple, but medium overhead
- Imp. 2: Lowest overhead, but difficult loop structure
- Imp. 3: Mathematically most simple, but highest overhead

- Due to readability and small-ish overhead, we chose to go with Imp. 1

FORSCHUNGSZENTRUM JÜLICH
FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Past Work - Constraint Reduction

	Days	Regions	Objective		Time (s)			Constraints		
			Naïve	Imp.	Naïve	Imp.	Δ	Naïve	Imp.	Δ
refScen	10	5	35834.4	35834.4	83	77	7.2%	155229	155204	-25
	20	5	35637.7	35637.7	247	176	28.7%	260829	260804	-25
	30	5	35642.9	35642.9	403	384	4.7%	366429	366404	-25
	10	10	34778.5	34778.5	212	212	0%	291972	291917	-55
	20	10	34912.0	34912.0	1147	1018	11.2%	494532	494477	-55
	30	10	35363.3	35363.3	1758	1636	6.9%	697092	697037	-55
flexScen	10	5	47412.7	47412.7	642	542	15.6%	64436	64416	-20
	20	5	47725.0	47725.0	1524	1347	11.6%	110036	110016	-20
	30	5	48401.2	48401.2	2883	2366	17.9%	155636	155616	-20
	40	5	48208.6	48208.6	4112	3933	4.4%	201236	201216	-20
	10	10	44736.9	44736.9	2423	2024	16.5%	116922	116881	-41
	20	10	46850.9	46850.9	7434	7032	5.4%	201642	201601	-41

JÜLICH FORSCHUNGSZENTRUM

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Current Work - BCA Improvements

- BCA needs to be examined on a finer resolution to find performance "hot spots"
  - Oliver made skeleton code for timing subsections of BCA ("timesplit")
    - Room for improvement
    - Make sure we are best utilizing cache (spatial and temporal locality specifically)
    - Also want to time fixing/unfixing binaries, adding/setting cut constraints, etc.
  - Profiling tools are being explored
- Error handling improvements

Current Work - BCA Improvements

Algorithm 1 Budget-Cut Algorithm

Input: an instance of model (3); $a_i \leftarrow$objective function coefficient of element $i, \forall i \in I$; $TIME$
Output: z^{opt}; optimality gap for best known feasible solution to model (3)

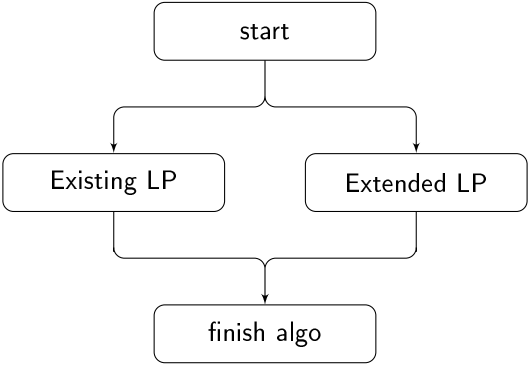
1: $iter \leftarrow 0$, search $\leftarrow$ `True`
2: **while** time $\leq TIME$
3: Solve model (5), get $\bar{z}$ — preptime1 lptime1
4: Solve model (6), get $\underline{z}$ — preptime2 lptime2
5: Update time to the cumulative wall-clock time
6: $b \leftarrow \bar{z} - \underline{z}$
7: **if** $b < min_{i \in I} a_i$
8: $z^{\text{opt}} \leftarrow \bar{z}$; Gap $\leftarrow 0\%$; go to 23
9: **if** $b > max_{i \in I} a_i$
10: search $\leftarrow$ `False`
11: **while** search — preptime3
12: $J = \cup_{i: a_i > b}$; $I \leftarrow I \setminus J$
13: **if** $J \neq \emptyset$
14: $iter \leftarrow iter + 1$
15: Solve model (6) with $bin_i \leftarrow 0, \forall i \in J$; update $\underline{z}$, Gap $\leftarrow$ optimality gap
16: $b \leftarrow \bar{z} - \underline{z}$
17: **if** $b < min_{i \in I} a_i$ — iteration[i]
18: $z^{\text{opt}} \leftarrow \bar{z}$; go to 23
19: **else**
20: go to 22
21: Update time to the cumulative wall-clock time — miptime
22: Solve model (8); $z^{\text{opt}} \leftarrow z^*$, Gap $\leftarrow$ optimality gap
23: **Return:** z^{opt}, Gap

Credit: *Oliver Rehberg et al. (2021)*

JÜLICH FORSCHUNGSZENTRUM
FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Current Work - BCA Improvements

- Calculation of "Existing" and "Extended" LPs are independent



Current Work - BCA Improvements

- Calculation of "Existing" and "Extended" LPs are independent
  - "embarrassingly parallelizable"
  - First try, ran into problems with the esM not being picklable
  - Working on a solution without needing to build a separate pyomo model instance
- Either by "multiprocessing" package or "threading" package (MPI/OpenMP not necessary)
  - Threading uses less memory, but whichever scales better

Current Work - BCA Extensions

- At the moment, we know BCA fails if the first "Existing" LP is infeasible
  - This is the case for districtScen (and others, such as greensfield)
- We are working on locating the reason for infeasibility, and generating a feasible solution apriori to "get the algorithm going" in this case
  - This can be done by solving the model with a 0 objective function, subject to the same constraints
    - Indicates what a feasible solution would look like
    - But this is (in general) not a smart thing to do, and may take longer than optimizing the model naïvely!
- Ideally, we would like to develop a general method to determine a feasible "Existing" model formulation **without** having to do this. i.e. replace line 3:

Current Work - BCA Extensions

Algorithm 1 Budget-Cut Algorithm

Input: an instance of model (3); $a_i \leftarrow$objective function coefficient of element $i, \forall i \in I$; $TIME$
Output: z^{opt}; optimality gap for best known feasible solution to model (3)
1: $iter \leftarrow 0$, search $\leftarrow$ `True`
2: **while** time $\leq TIME$
3: Solve model (5), get $\bar{z}$
4: Solve model (6), get $\underline{z}$
5: Update time to the cumulative wall-clock time
6: $b \leftarrow \bar{z} - \underline{z}$
7: **if** $b < min_{i \in I} a_i$
8: $z^{\text{opt}} \leftarrow \bar{z}$; Gap $\leftarrow 0\%$; go to 23
9: **if** $b > max_{i \in I} a_i$
10: search $\leftarrow$ `False`
11: **while** search
12: $J = \cup_{i:a_i > b}$; $I \leftarrow I \setminus J$
13: **if** $J \neq \emptyset$
14: $iter \leftarrow iter + 1$
15: Solve model (6) with $bin_i \leftarrow 0, \forall i \in J$; update $\underline{z}$, Gap $\leftarrow$ optimality gap
16: $b \leftarrow \bar{z} - \underline{z}$
17: **if** $b < min_{i \in I} a_i$
18: $z^{\text{opt}} \leftarrow \bar{z}$; go to 23
19: **else**
20: go to 22
21: Update time to the cumulative wall-clock time
22: Solve model (8); $z^{\text{opt}} \leftarrow z^*$, Gap $\leftarrow$ optimality gap
23: **Return:** z^{opt}, Gap

Credit: *Oliver Rehberg et al. (2021)*

Next Steps

- Finish developing method to fix BCA's infeasibility issue
  - This is highest priority
  - Smallest spatial clustering, districtScen has 180 optional components
    - We expect BCA to perform well
- Various Code Improvements
  - Capacity variables can be reformulated to reduce a decision variable

$$cap_{c,\ell} = \begin{cases} \text{capPerUnit}_c \cdot nbReal_{c,\ell} & \text{if } cap_{c,\ell} \text{ continuous} \\ \text{capPerUnit}_c \cdot nbInt_{c,\ell} & \text{if } cap_{c,\ell} \text{ discrete} \end{cases}$$

  - For example, given we know $cap_{c,\ell}$ continuous apriori:

$$\text{M}_c \cdot bin_{c,\ell} \geq cap_{c,\ell} \qquad \Rightarrow \qquad \text{M}_c \cdot bin_{c,\ell} \geq \text{capPerUnit}_c \cdot nbReal_{c,\ell}$$

Next Steps

- Constraints capFix and binFix can be reformulated as bounds
  - For example, what this improvement could look like for capFix

if *have capFix* **then**
    if *continuous Flag* **then**
        set nbReal.bounds$\left(\frac{capFix}{capPerUnit}, \frac{capFix}{capPerUnit}\right)$
    end
    else
        set nbInt.bounds$\left(\frac{capFix}{capPerUnit}, \frac{capFix}{capPerUnit}\right)$
    end
end