June Update

Budget Cut Algorithm extensions

Dane Lacey

FAU Erlangen-Nuremberg

June 2, 2021

Introduction

- Recap
- Current Work / Solutions to feasibility problem
- Next Steps

Recap

- What we wanted to do:
  - BCA performance profiling
  - Parallelization of "Existing" and "Extended" LPs
  - Generation of a feasible "Existing" configuration of binaries
  - Various code improvements
- Effort has been primarily focused on fixing "Existing" feasibility issue, and implementing solutions
- Therefore, the majority of this month's update primarily deals with BCA extensions
  - districtScen has been used for experimentation
  - Although a solution should be independent of model scenario

Recap

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)*

JÜLICH FORSCHUNGSZENTRUM
FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

BCA - Feasibility Problem

- The original idea was to just set all bin=0 (i.e. nothing new is built), and take that to be the "Existing" configuration of optional components
    - The optimal objective function value $\overline{z}$ is then used as a "warmstart" to the final MIP (i.e. it is used as the incumbent solution)
- Intuition tells us that this should be feasible, but as we've seen, setting all bin=0 should not be assumed so
- We needed a more flexible approach

JÜLICH FORSCHUNGSZENTRUM FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

BCA - Solution to Feasibility Problem

- BCA just needs *something* for a reasonable upper bound on the budget to operate
  - Of course for better performance, we would like this $\overline{z}$ to be close to optimal
- We've come up with three ways to generate feasible starting configurations of binary variables:
  - Naïve Minimal
  - Calculated Configuration
  - Manual/Iterative

BCA - Solution to Feasibility Problem - Naïve Minimal

- Optimizing a model such that all bin=0 *is likely to* be feasible, yet it doesn't *need* to be
  - What is also likely to be feasible, is if all components are built (i.e. all bin=1)
  - This is what I noticed when solving by the "classical obj = 0 method"
- The idea here is to first try to optimize "Existing" with all bin=0, then with all bin=1
  - If either the bin=0 case or the bin=1 case is infeasible, it is discarded
  - The minimum of these two would be our $\bar{z}$, and would be fed into the final MIP as an incumbent solution
- Pros & Cons
  - -fast (only solve two LPs), easy to understand
  - -not guaranteed to actually give feasibility (just more likely)

JÜLICH FORSCHUNGSZENTRUM FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

BCA - Solution to Feasibility Problem - Calculated Configuration

- Essentially "borrowed" a snippet from optimizeTSAmultistage.py
- We would replace line (3:) of BCA with two lines:
  - Solve a fully simplified (i.e. 1 typical day) version of the original MIP
  - Then, the resulting binary configuration is fixed on the original MIP
    - We then solve the original MIP with the original temporal aggregation (or lack thereof) naïvely, with the fixed binaries
    - This results in the upper bound $\overline{z}$
    - In this way, we "calculate" a feasible starting binary configuration
- Pros & Cons
  - -Most general of the three, automatic, this configuration can be saved externally for subsequent model runs
  - -only makes sense for models with higher TSA/ fully temporally resolved models, potentially slow (because we need to build and solve a whole "simplified" model)

BCA - Solution to Feasibility Problem - Manual/Iterative

- For a given model scenario with an infeasible "Existing" configuration (for example, districtScen), one can manually find feasible binary configurations by trial and error
  - With districtScen, we found that we can either "turn on" CHP, boiler, or heatpump and BCA will run
- But what if we don't want to manually guess and check our way to a feasible starting configuration?
- It is very quick for Gurobi to build and solve one of these LPs where all bin are fixed (on the order of a few seconds)
  - The idea here is to have BCA "flip on and off" optional components until it arrives at a feasible starting solution
- Pros & Cons
  - -Potentially very quick, if a feasible configuration exists, it will be found
  - -Also potentially slow if many binaries need to be explored

JÜLICH FORSCHUNGSZENTRUM FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

BCA - Solution to Feasibility Problem

Table 1: Binary configuration results

	BCA LPs				Naïve	
Method	Existing Obj.	Extended Obj.	Gap	Time (s)	Optimal Obj	Time (s)
Naïve Minimal						
bin = 0	infeasible	-	-	-	-	-
bin = 1	133303	58301	75002	12	69881.9	57839
Calculated Configuration						
	over time	-	-	-	-	-
Manual/Iterative						
chp	78267	58301	19966	5	69881.9	57839
boiler	123969	58301	65668	5	69881.9	57839
heatpump	269200	58301	210899	11	69881.9	57839

BCA - districtScen - Numerical Trouble

- Now the "Starting" and "Extended" LPs function as we would like
- Gurobi : Matrix range [2e-02, 1e+06], Numerical trouble encountered

$$\text{None} \leq \text{cap\_stor[transformer, Thermal Storage\_Big]} -$$
$$1000000 * \text{designBin\_stor[transformer, Thermal Storage\_Big]} \leq 0.0$$

- We know this is the only cause of numerical trouble for Gurobi
  - Because of this "Numerical trouble", the "Starting" LP is not accepted into the final MIP as an incumbent solution (warmstart)
  - If this constraint is removed, we have "Matrix range [2e-02, 9e+03]" and no "Numerical trouble encountered" warning (yet we violate other constraints when this is removed, so it is necessary to have a constraint of this form)

JÜLICH FORSCHUNGSZENTRUM FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

BCA - districtScen - Numerical Trouble

- The 100000 is a "bigM". We would love to replace it with capMax, but none is given for Thermal Storage_Big
- So in order for us to eliminate the Numerical trouble, we need to derive a bound for cap_stor[transformer,Thermal Storage_Big]
- Looking through the constraints in the literature, we decided to pursue the following constraint for typical periods and unprecise bounds:

$$SoC^{comp,min} \cdot cap_{loc}^{comp} \leq \underline{SoC}_{loc,p,t}^{comp,sup}$$

$$\text{with } \underline{SoC}_{loc,p,t}^{comp,sup} = SoC_{loc,p}^{comp,inter} \cdot (1 - \eta^{\text{self-discharge}})^{\frac{t^{per\ \ period} \cdot \tau^{hours}}{h}} + SoC_{loc,map(p)}^{comp,min}$$

BCA - districtScen - Solution to Numerical Trouble

- But,

$$\left(1-\eta^{\text{self-discharge}}\right)^{\frac{t^{per\quad period}\cdot\tau^{hours}}{h}}$$

  is a constant, so it can just be replaced with α.
- Rewriting this constraint for *cap* gives us the bound:

$$cap_{loc}^{comp} \leq \underline{SoC}_{loc,p,t}^{comp,sup} / SoC^{comp,min}$$
$$= \frac{\alpha \cdot SoC_{loc,p}^{comp,inter} + SoC_{loc,map(p)}^{comp,min}}{SoC^{comp,min}}$$

BCA - districtScen - Solution to Numerical Trouble - Aside

- If $SoC^{comp,min} = 0$?
- Then we have a constraint reduction!

$$SoC^{comp,min} \cdot cap_{loc}^{comp} \leq \underline{SoC}_{loc,p,t}^{comp,sup}$$
$$\Rightarrow 0 \leq \underline{SoC}_{loc,p,t}^{comp,sup}$$

- This is already established, since $SoC_{loc,p}^{comp,inter}$, $(1 - \eta^{\text{self-discharge}})$, and $SoC_{loc,map(p)}^{comp,min}$ are all defined to be non-negative
- So this constraint is redundant, and can be skipped
  - For example, simply with an if-then check within appropriate storage.py function, and a returned "pyomo.Skip.Constraint"

BCA - districtScen - Solution to Numerical Trouble

- Substituting our specific parameters for districtScen:

$$cap_{transformer}^{Thermal\ \ Storage\_Big} \leq \frac{\max\left(\alpha \cdot SoC_{transformer,p}^{Thermal\ \ Storage\_Big,inter}\right) + \max\left(SoC_{transformer,map(p)}^{Thermal\ \ Storage\_Big,min}\right)}{SoC^{Thermal\ \ Storage\_Big,min}}$$

with $\alpha = (1 - \eta^{\text{self-discharge}})^{24}$

$\forall p \in \mathcal{P}$

- When $SoC^{Thermal\ \ Storage\_Big,min} \neq 0$, this should give us bound smaller than 1000000, thus solving the "Numerical trouble".

Next Steps

- Have an implementation of BCA containing all three feasible binary configuration generation approaches, and letting user decide the option at execution time
  - The hard part is finished for this, it just needs to be implemented and merged
- Generate results with this derived bound on cap_stor[transformer,Thermal Storage_Big]
  - Compute and compare optimization run times on districtScen with Gurobi's naïve solver (subject to necessary time limits)

Next Steps

With the days remaining in June, implement code improvements

- $SoC^{comp,min} = 0$ constraint reduction
- Capacity variables reformulation

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

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

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