An introduction to energy optimization in SMS++
Part IV: decomposition & energy optimization in SMS++

Antonio Frangioni

Dipartimento di Informatica, Università di Pisa

EdF Labs — May 25-26, 2023

Meta–Outline

- Part I: SMS++ basics & energy-related components
- Part II: hands-on with SMS++ for energy optimization
- Part III: a quick recap of decomposition techniques
- Part IV: decomposition & energy optimization in SMS++

Outline – Part IV

A general Investment Problem

- Set N of (generation/distribution) units, κ_i identical copies of each $i \in N$
- For $\kappa = [\,\kappa_i\,]_{i \in N}$, investment cost $F(\,\kappa\,)$ ("easy") and operational cost

$$\begin{aligned} O(\,\kappa\,) = \min\ & \textstyle\sum_{i \in N} \sum_{j=1}^{\kappa_i} c_i(\,x_{i,j}\,) \\ \text{s.t. } & x_{i,j} \in X_i \qquad\qquad j = 1, \ldots, \kappa_i\ ,\ i \in N \qquad (1) \\ & \textstyle\sum_{i \in N} \sum_{j=1}^{\kappa_i} A_i x_{i,j} \geq d \end{aligned}$$

- Everything convex $\implies$ all $i \in N$ produce identically at optimality $\implies$

$$\begin{aligned} O(\,\kappa\,) = \min\ & \textstyle\sum_{i \in N} \kappa_i c_i(\,x_i\,) \\ \text{s.t. } & x_i \in X_i \qquad\qquad i \in N \qquad (2) \\ & \textstyle\sum_{i \in N} \kappa_i A_i x_i \geq d \end{aligned}$$

- Can extend to stochastic setting (S = scenarios, $\mathcal{N}$ = nonanticipativity)

$$\begin{aligned} O(\,\kappa\,) = \min\ & \textstyle\sum_{i \in N} \kappa_i \sum_{s \in S} \pi^s c_i^s(\,x_i^s\,) \\ \text{s.t. } & x_i^s \in X_i^s \qquad\qquad i \in N\ ,\ s \in S \qquad (3) \\ & \textstyle\sum_{i \in N} \kappa_i \sum_{s \in S} A_i^s x_i^s \geq d\ \ ,\ \ x \in \mathcal{N} \end{aligned}$$

A general Investment Problem

- Investment problem $\min\{ F(\kappa) + O(\kappa) : \kappa \in K \}$: extremely hard
  as even (2) / (3) hard ((1) harder), since convexity assumption untrue
- Lagrangian relaxation triply clever:
  $\phi(\lambda, \kappa) = \lambda d + \sum_{i \in N} \kappa_i \min\{ c_i(x_i) - \lambda A_i x_i : x_i \in X_i \}$
  - decomposes into (many, easier, smaller) independent subproblems
  - automatically convexifies c and X[1]
  - $\phi(\lambda, \kappa)$ is concave in λ and affine in κ
- Convexified version: $\underline{O}(\kappa) = \max\{ \phi(\lambda, \kappa) : \lambda \geq 0 \} = \phi(\lambda^*(\kappa), \kappa)$
- Convexified Investment Problem: $\min\{ F(\kappa) + \underline{O}(\kappa) : \kappa \in K \}$
  possibly the best trade-off between computability and accuracy
- Crucial: $[c_i(x_i^*(\lambda^*(\kappa))) - \lambda^*(\kappa) A_i x_i^*(\lambda^*(\kappa))]_{i \in N} \in \partial \underline{O}(\kappa)$
  $\Longrightarrow$ can use bundle methods[2] or stabilised Benders' ones[3]

[1] Lemaréchal, Renaud "A geometric study of duality gaps, with applications" *Math. Prog.*, 2001

[2] van Ackooij, F. "Incremental bundle methods using upper models" *SIOPT*, 2018

[3] van Ackooij, AF., de Oliveira "Inexact Stabilized Benders' Decomposition Approaches [...]" *COAP*, 2016

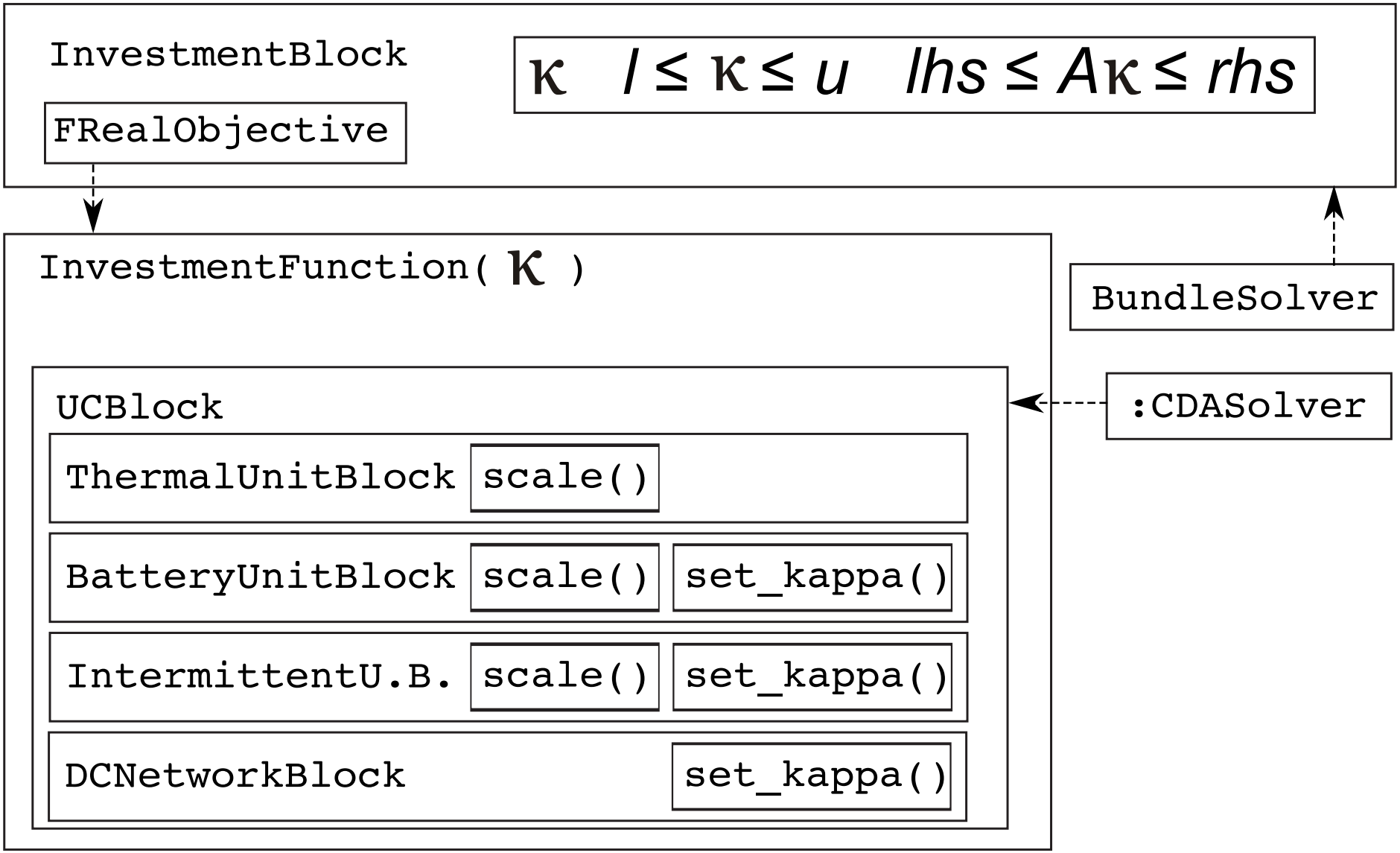
Outline

The new InvestmentBlock component

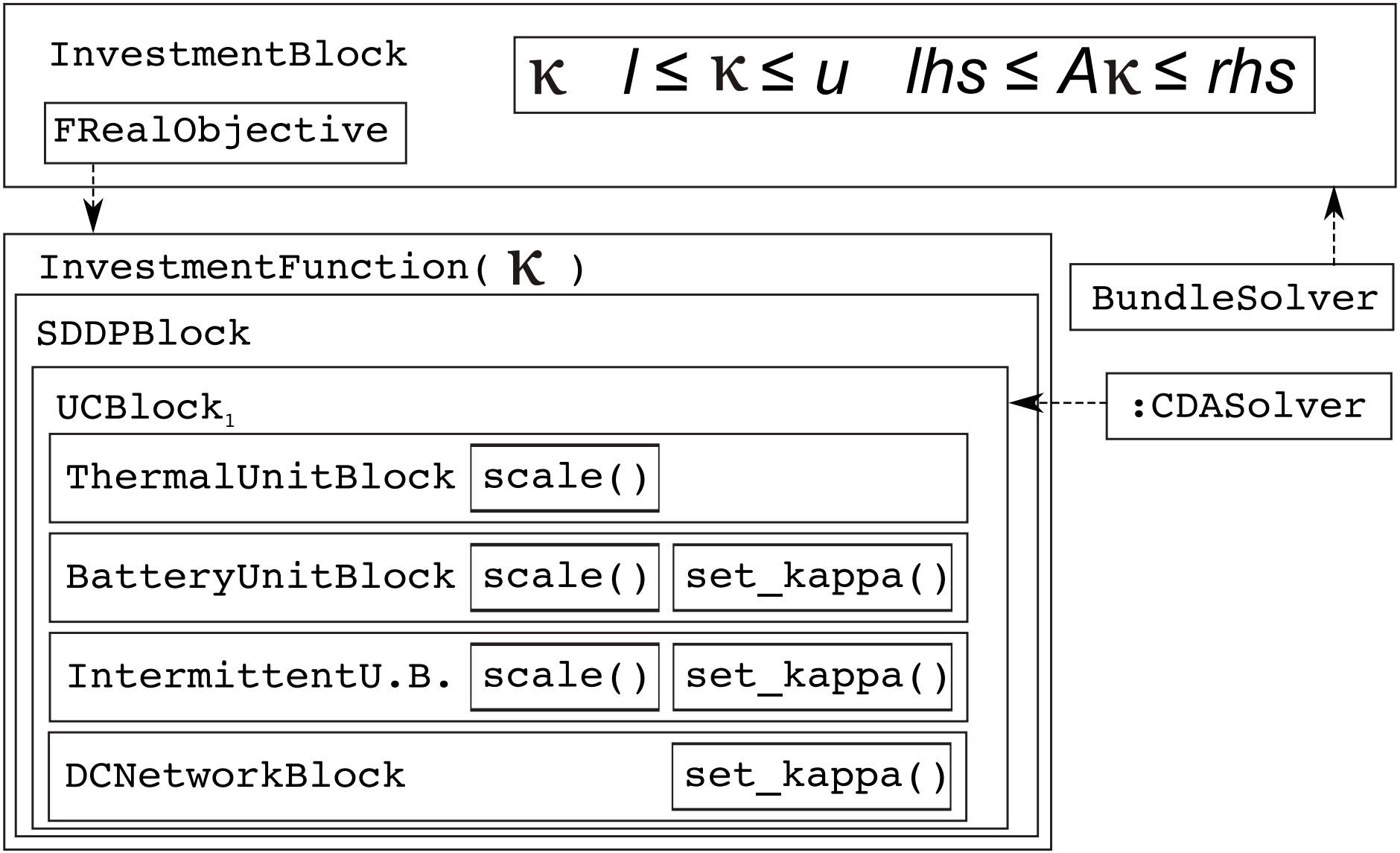
- Specific component for the Investment Problem, public as of two days ago
- Mostly relays on specialised InvestmentFunction: both a C05Function and a Block (sounds familiar?), computing $\underline{O}(\kappa)$
- Two kinds of continuous resources (κ_i):
  - upper / lower bounds on variables (network / unit capacities)
  - number of copies of existing :UnitBlock
- Requires support from UCBlock and the :UnitBlock / :NetworkBlock:
  - scale() in ThermalUnitBlock, BatteryUnitBlock, IntermittentUnitBlock (scales the whole unit)
  - set_kappa() in BatteryUnitBlock, IntermittentUnitBlock, DCNetworkBlock (only scales some RHSs)

  then UCBlock has to scale the linking constraints when κ_i change
- Supports SDDPBlock with UCBlock inside (same changes to all stages)
- Can use any CDASolver (LagrangianDualSolver, :MILPSolver)

InvestmentBlock schematics



InvestmentBlock schematics



- Scaling a `:Block` a general concept, may be upcasted to base `Block`

InvestmentFunction netCDF structure

- `ReplicateBatteryUnits` and `ReplicateIntermittentUnits`: 1 is `scale()`, 0 if `set_kappa()`
- `NumAssets`, `AssetType`$[\,i\,] = 0$ if `UnitBlock`, 1, if transmission line
- `Assets`$[\,i\,] =$ either sub-`Block` number or line number
- `LowerBound` and `UpperBound` on investment assets
- `InstalledQuantity` at start of investment decision
- `Cost` for each unit above `InstalledQuantity`
- `DisinvestmentCost` for each unit below `InstalledQuantity`
- `InnerBlock` to be invested upon, either a `UCBlock` or a `SDDPBlock` with `UCBlock` in each stage
- `NumConstraints`, `Constraints_A`, `Constraints_LowerBound` and `Constraints_UpperBound` for $l \leq A\kappa \leq u$

Outline

The `InvestmentBlock` exercise

- You are given a `netCDF` file for a small instance (5 units, 5 timestamps, HVDC network)
- Run it to see the results
- Take any (small but decent-sized) UC instance from the repos
- Produce an Investment Problem replicating a few of that UC's units
- Run it to see the results, find "interesting" values of design costs
- @home: replace `UCBlock` with a `SDDPBlock` using `UCBlock` for the stochastic version (or wait the next iteration of the course)

Conclusions
(Part IV, overall)

Conclusions Part IV

- Investment problems perhaps the biggest challenge in energy optimization
- Trade-off between system modelling accuracy and computational cost, extremely hard to navigate (simplify operational model up-front)
- Choosing what to relax nontrivial, side-effects can be hard to predict
- Lagrangian approach one interesting way: automatic convexification
- Should really be used with (convex) stochastic model inside
- Should really be used with scenarios outside
- Multilevel heterogeneous parallel decomposition a necessity
- Models must support algorithms features all along the hierarchy
- SMS++ $\approx$ the only game in town for this kind of applications

Conclusions overall

- Decomposition a large set of different techniques focusing on structure
- Not useful in all cases, often somewhat useful, at times crucial
- Theory (more or less) established, software support always been the issue
- SMS++ trying to improve on that, some steps of a long and winding road
- Useful already for huge-scale applications
- Could become very useful after having attracted mindshare (very hard)
- Yet, I think it (imperfectly) tries to address some real needs:
  - improve collaboration and code reuse, reduce huge code waste
  - help the software development process of very complex models and the corresponding highly complex algorithms
  - make decomposition approaches much more easily available
  - help a much-needed higher uptake of parallel methods
- Collaborative effort, all users / testers / contributors welcome
- In the future, one of the three legs of the ultimate modelling system[4]

[4] F., Perez Sanchez "Transforming Mathematical Models Using Declarative Reformulation Rules" *LNCS*, 2011

Please fill-in the course evaluation form