An introduction to energy optimization in SMS++
Part II: hands-on with SMS++ for energy optimization

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 II

1. Exercise: `NuclearUnitBlock`
2. Home Exercise
3. Conclusions (Part II)

The NuclearUnitBlock exercise

- Nuclear unit = a thermal unit with modulation constraints
- Modulation = (significant) change of power output
- Modulation constraint: modulation must happen "sparsely"
- $0 \leq \Delta_+^M \ll \Delta_+$ / $0 \leq \Delta_-^M \ll \Delta_-$ upper / lower modulation limits
- $\tau^- \geq \tau^M \geq 2$ modulation interval: $p_t - p_{t-1} > \Delta_+^M$ or $p_{t-1} - p_t > \Delta_-^M$ $\Longrightarrow$ must not happen again until $t + \tau^M$
- Startup and shutdown do not count as modulation
- Recall τ^0 initial state: $\tau^0 > 0 \Longrightarrow$ on since τ^0 instants, $\tau^0 \leq 0 \Longrightarrow$ off since $-\tau^0$ instants ($= 0$ is "just off")
- $m^0 > 0$ initial modulation state = modulated m^0 instants before the first (default τ^M = irrelevant)
- Shutdown automatically satisfy modulation constraint

NuclearUnitBlock constraints

- Need modulation variables $m_t \in \{\,0\,,\,1\,\}$

$$
\begin{aligned}
& p_t - p_{t-1} \leq \Delta_+^M u_{t-1} + (\Delta_+ - \Delta_+^M) m_t + \bar{l} v_t && t \in T && (1)\\
& p_{t-1} - p_t \leq \Delta_-^M u_t + (\Delta_- - \Delta_-^M) m_t + \bar{u} w_t && t \in T && (2)\\
& m_t = 0 && 0 \leq t < \tau^M - m^0 && (3)\\
& m_t \leq u_t && t \in T && (4)\\
& m_t \leq (1 - v_t) && t \in T && (5)\\
& \textstyle\sum_{h=\max\{\,0\,,\,t-\tau^M+1\,\}}^{t} m_h \leq 1 && t \in T && (6)
\end{aligned}
$$

- (1) / (2): ramp rates are Δ_+^M / Δ_-^M unless modulation ($m_t = 1$)
- (3): prohibit initial modulations based on past ones
- (4): cannot modulate if off, (5): startup is no modultion (shutdown version not needed as $w_t = 1 \implies u_t = m_t = 0$)
- (6): modulation constraint proper

Data for `NuclearUnitBlock`

- `netCDF::NcGroup` for `NuclearUnitBlock` = `ThermalUnitBlock` plus:
    - positive scalar variable "`ModulationTime`" of type `netCDF::NcUint` for τ^M (default 2)
    - positive scalar variable "`InitModulation`" of type `netCDF::NcUint` for m^0 (default = `ModulationTime`)
    - non-negative scalar variable "`ModulationDeltaRampUp`" of type `netCDF::NcDouble` for Δ^M_+ (default 0)
    - non-negative scalar variable "`ModulationDeltaRampDown`" of type `netCDF::NcDouble` for Δ^M_- (default 0)

- How to produce one:
    - `ncdump file.nc4 > file.txt`
    - edit section(s) corresponding to `ThermalUnitBlock(s)` to add missing data
    - `ncgen -o newfile.nc4 file.txt`

Outline

Further food for thoughts

- Consider supporting changes to τ^M and m^0 from the physical representation:
  - what changes would it entail in the current implementation?
  - would there be trade-offs?
- Consider supporting changes to Δ_+^M and Δ_-^M from the abstract representation:
  - what changes would it entail in the current implementation?
  - would there be trade-offs?
- How about extending the former to the abstract representation?
- Extend `ThermalUnitDPSolver` to support `NuclearUnitBlock`

Conclusions

(Part II)

Conclusions (for now)

- New `:UnitBlock` can be added without `:UCBlock` knowing / bothering
- `C++` inheritance an interesting approach to build complex models
- Hierarchy of units sharing common parts?
- `No free lunch`:
  - significant amount of `C++` coding involved
  - base classes need be written for being extended
  - significant analysis always required
- Trade-off between flexibility and complexity: wisely choose your use cases
- There is a reason why these are called mathematical programs
- Once done and tested is there for good: good programming practices crucial in complex projects, optimization not an exception