ECE 5314: Power System Operation & Control

Lecture 6: Unit Commitment

Vassilis Kekatos

R1 A. J. Wood, B. F. Wollenberg, and G. B. Sheble, Power Generation, Operation, and Control, Wiley, 2014, Chapters 3-4.

R2 A. Gomez-Exposito, A. J. Conejo, C. Canizares, *Electric Energy Systems: Analysis and Operation*, Chapter 5.

Motivation

Economic dispatch assumes all units to be online and ready to produce, but:

- ramping and must-stay-on/off constraints
- start-up/shut-down costs (cooling vs. banking)
- crew constraints (units that cannot be started together)
- spinning (coal) and offline (hydro, gas) reserves
- fuel constraints (use it or loose it)
- must-run status (due to voltage regulation or other functions)

Such constraints are accommodated by **unit commitment** problems

- bad news: non-convex, hard to solve
- good news: solved in advance (day-ahead)

Static unit commitment

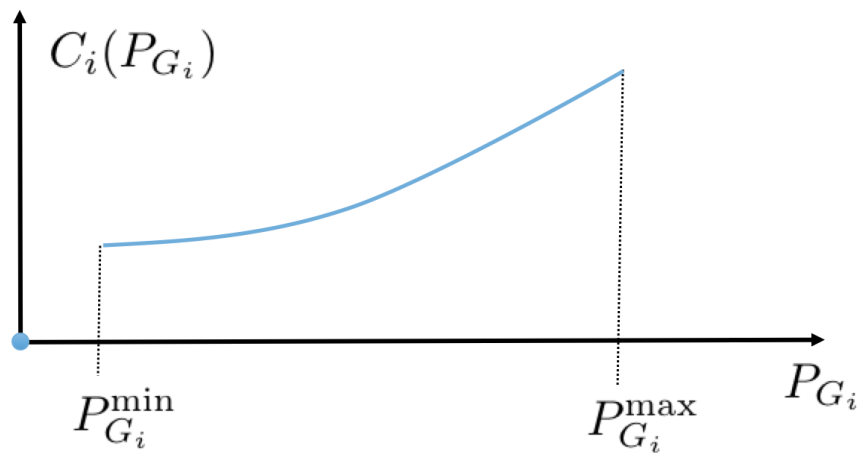
- Extend ED to include on/off scheduling of units
- Binary variable: $u_i = 1$ if unit i is scheduled to be on; 0 otherwise
- If unit i is scheduled to be on, it has to produce at least $P_i^{\min}$

$$\min_{\{P_i, u_i\}_i} \quad \sum_{i=1}^{N} u_i C_i(P_i)$$

$$\text{s.to} \quad \sum_{i=1}^{N} P_i = D$$

$$u_i P_i^{\min} \leq P_i \leq u_i P_i^{\max}, \ \forall \ i$$

$$u_i \in \{0, 1\}, \ \forall \ i$$



- For economic dispatch, all u_i's have already been decided

An example

Two generators with quadratic costs $C_i(P_i) = C_{0i} + a_i P_i + \frac{b_i}{2} P_i^2$

	C_{0i} [\$/h]	a_i [\$/MWh]	b_i [\$/(MW)^{2}h]	$P_i^{\min}$ [MW]	$P_i^{\max}$ [MW]
G1	100	20	0.05	0	400
G2	200	25	0.10	0	300

Unit commitment for different loads D

Case	D [MW]	P_1 [MW]	P_2 [MW]	λ [\$/MWh]	Cost [\$/h]
(1,0)	40	40	0	22	940
(0,1)	40	0	40	29	1,280
(1,1)	40	40	0	22	1,140
(1,0)	250	250	0	33	6,663
(0,1)	250	0	250	50	9,575
(1,1)	250	200	50	30	6,675
(1,0)	300	300	0	35	8,350
(0,1)	300	0	300	55	12,200
(1,1)	300	233	67	32	8,217

Avoiding bilinear products

- MILPs are the 'easiest' mixed-integer programs
- To remain within MILP class, need to avoid products of variables

- Generation capacity constraints

$$u_i P_i^{\min} \leq P_i \leq u_i P_i^{\max} \qquad \text{vs.} \qquad u_i P_i^{\min} \leq u_i P_i \leq u_i P_i^{\max}$$

- Generation cost $u_i C_i(P_i)$ for $C_i(P_i) = C_{0i} + a_i P_i + \frac{b_i}{2} P_i^2$

$$u_i C_{o,i} + a_i P_i + \frac{b_i}{2} P_i^2 \qquad \text{vs.} \qquad u_i C_{o,i} + a_i u_i P_i + \frac{b_i}{2} u_i P_i^2$$

Multi-period unit commitment

N generation units over T control periods (24 hours of a day)

$$\min_{\{P_i^t, u_i^t, s_i^t\}_{i,t}} \sum_{t=1}^{T} \sum_{i=1}^{N} u_i^t C_i^t(P_i^t) + s_i^t$$

$$\text{s.to} \quad \sum_{i=1}^{N} P_i^t = D^t, \ \forall\, t$$

$$u_i^t P_i^{\min} \le P_i^t \le u_i^t P_i^{\max}, \ \forall\, i, t$$

$$u_i^t \in \{0, 1\}, \ \forall\, i, t \qquad \longleftarrow \text{ integral (binary) constraint}$$

$$s_i^t \ge s_i(u_i^t - u_i^{t-1}), \ \forall\, i, t \qquad \longleftarrow \text{ startup cost } s_i$$

$$s_i^t \ge 0, \ \forall\, i, t \qquad \longleftarrow \text{ startup cost variable}$$

Decisions are coupled across time through the startup cost

What if I replace $u_i^t \in \{0, 1\}$ with $u_i^t \in [0, 1]$ for all i, t?

More unit commitment constraints

Additional constraints coupling decisions across time

- Ramp up constraint: $P_i^t - P_i^{t-1} \leq R_i^{\text{up}}$
- Ramp down constraint: $P_i^{t-1} - P_i^t \leq R_i^{\text{down}}$
- Spinning reserves:

$$\sum_{i=1}^{N} u_i^t P_i^{\max} \geq D + P_{\text{reserve}}$$

- Must-stay-on for L_i periods:

$$u_i^t - u_i^{t-1} \leq u_i^\tau, \ \tau = t+1, \ldots, \min\{t + L_i - 1, T\}$$

- Must-stay-off for ℓ_i periods:

$$u_i^{t-1} - u_i^t \leq 1 - u_i^\tau, \ \tau = t+1, \ldots, \min\{t + \ell_i - 1, T\}$$

Mixed-Integer (Non-)Linear Programs

Optimization problems with continuous and integer/binary variables

Brute-force method should solve 2^{NT} EDs!

Even MILPs are NP-hard in general! (sometimes MINLPs linearized to MILPs)

Common solution approaches:

1. Dynamic programming [Bell, 1950]
2. Branch and bound algorithms [Land & Doig, 1960]
3. Lagrangian relaxation [Muckstadt & Koenig, 1977; Bertsekas, 1983]
4. Bender's decomposition [Bender, 1962]

Branch and bound method

Smart way to enumerate possible solutions; widely used in discrete optimization

1. Find lower and upper bounds (ℓ, u) on f^*
2. Problem with all binary constraints relaxed to box cons. added in queue
3. Solve the next problem in the queue to get $(\hat{\mathbf{x}}, \hat{f})$
4. If $\hat{\mathbf{x}}$ is binary and $\hat{f} < u$, UPDATE $\bar{\mathbf{x}} \leftarrow \hat{\mathbf{x}}$ and $u \leftarrow \hat{f}$
5. If $\hat{\mathbf{x}}$ is non-binary, then
   - 5.1 If $\ell < \hat{f} \leq u$, then
     BRANCH: pick a variable with non-binary value $\hat{x}_i$ and add two problems in the queue, one with constraint $x_i = 0$ and the other with $x_i = 1$
   - 5.2 If $\hat{f} > u$ (including infeasibility with $\hat{f} = \infty$), then
     CUT this branch since the solution cannot be improved
6. If queue is empty, output minimizer $\bar{\mathbf{x}}$; else go to Step 3.

There exists variations where ℓ progressively increases

Example on branch and bound method

$$\min_{\{P_i,u_i\}_{i=1}^2} \quad \sum_{i=1}^{2} u_i C_{0i} + a_i P_i + \tfrac{b_i}{2} P_i^2$$

$$\text{s.to} \quad P_1 + P_2 = 300$$

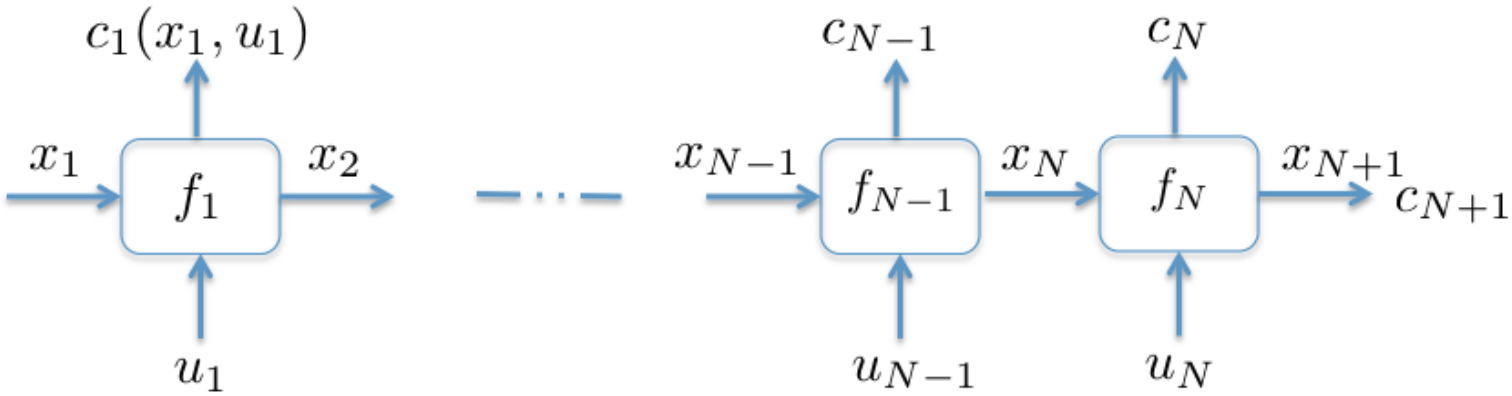
$$0 \le P_i \le u_i P_i^{\max}, \ i = 1, 2$$

$$u_i \in \{0, 1\}, \ i = 1, 2$$

Problem	$\hat{P}_1$	$\hat{P}_2$	$\hat{u}_1$	$\hat{u}_2$	$\hat{f}$	ℓ	u	$\bar{\mathbf{x}}$
$u_1, u_2 \in [0, 1]$	236.1	63.9	0.59	0.21	8,019	8,019	$+\infty$	–
$u_1 = 0, u_2 \in [0, 1]$	0	300	0	1	12,200	8,019	12,200	store
$u_1 = 1, u_2 \in [0, 1]$	237.8	62.2	1	0.21	8,059	8,059	12,200	branch
$u_1 = 1, u_2 = 0$	300	0	1	0	8,350	8,059	8,350	store
$u_1 = 1, u_2 = 1$	233	67	1	1	8,217	8,059	8,217	store

Dynamic programming method

Multi-stage problems (continuous/discrete) with a recursive structure



- **Stages**: indexed by $n = 1, \ldots, N$
- **States**: x_n (discrete or continuous)
- **Decision**: $u_n \in \mathcal{U}_n(x_n)$ (actions, controls, opt. variables)
- **Dynamic system**: $x_{n+1} = f_n(x_n, u_n)$ for $n = 1, \ldots, N$
- **Per-stage cost**: $c_n(x_n, u_n)$ for $n = 1, \ldots, N$, and final cost $c_{N+1}(x_{N+1})$

Problems solved with dynamic programming

Given state recursion, per-state costs and constraints, minimize the total cost

$$J^*(x_1) = \min_{\{u_n\}} \quad \sum_{n=1}^{N} c_n(x_n, u_n) + c_{N+1}(x_{N+1})$$

$$\text{s.to} \quad x_{n+1} = f_n(x_n, u_n), \ \forall n \qquad \text{(dynamic system)}$$

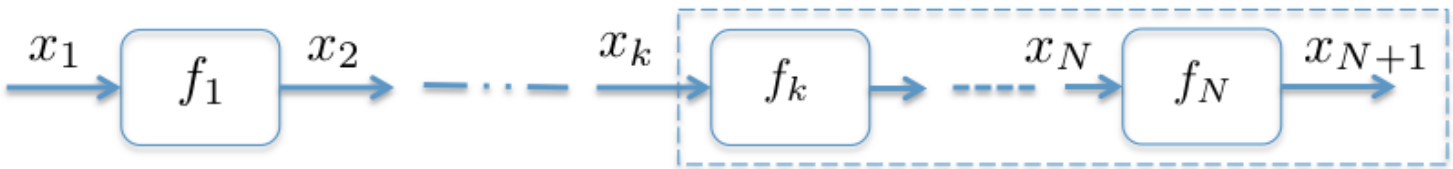
$$u_n \in \mathcal{U}_n(x_n), \ \forall n \qquad \text{(control options)}$$

DP widely used in a variety of applications:

- optimal (stochastic) control
- financial applications
- Kalman filter and hidden Markov models (HMMs)
- graph theory and networking problems
- wireless communications (Viterbi algorithm)

Optimality principle

Solving the tail problem after stage k for a state value x_k is optimal regardless how you reached x_k



$$J_k^*(x_k) = \min_{\{u_k\}_{k=1}^N} \sum_{n=k}^{N} c_n(x_n, u_n) + c_{N+1}(x_{N+1})$$

$$\text{s.to} \quad x_{n+1} = f_n(x_n, u_n), \ \forall n = k, \ldots, N \qquad \text{(dynamic system)}$$

$$u_n \in \mathcal{U}_n(x_n), \ \forall n = k, \ldots, N \qquad \text{(control options)}$$

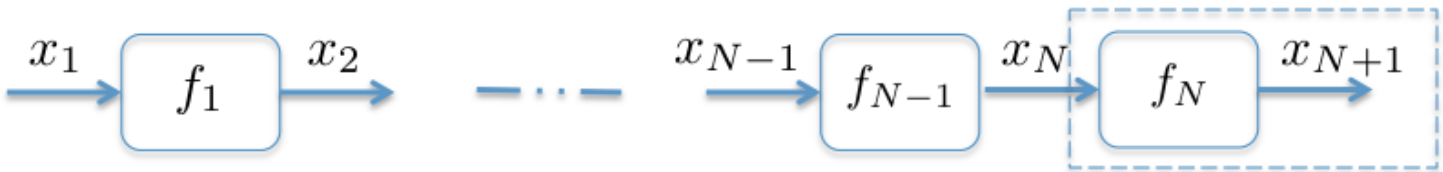
Recursive solution

Find last action u_N

$$J_N^*(x_N) = \min_{u_N} \quad c_N(x_N, u_N) + c_{N+1}(x_{N+1})$$

$$\text{s.to} \quad x_{N+1} = f_N(x_N, u_N)$$

$$u_N \in \mathcal{U}_N(x_N)$$

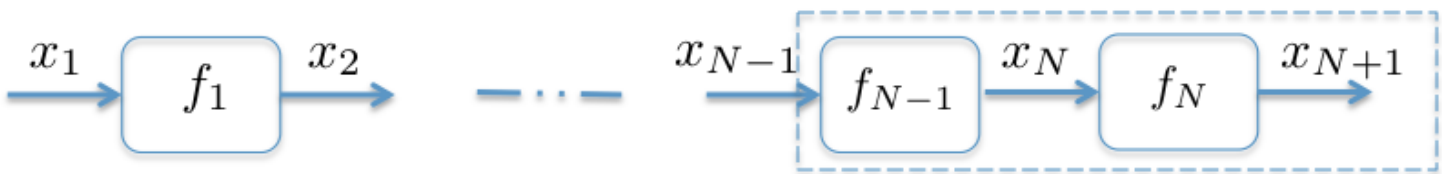


Find u_{N-1}

$$J_{N-1}^*(x_{N-1}) = \min_{u_{N-1}} \quad c_{N-1}(x_{N-1}, u_{N-1}) + J_N^*(x_N)$$

$$\text{s.to} \quad x_N = f_N(x_{N-1}, u_{N-1})$$

$$u_{N-1} \in \mathcal{U}_{N-1}(x_{N-1})$$



Dynamic programming algorithm

1. Start with $J^*_{N+1}(x_{N+1}) = c_{N+1}(x_{N+1})$
2. Go backwards: solve the previous stage for all possible values of state x_k

$$\begin{aligned} J^*_k(x_k) = \min_{u_k} \quad & c_k(x_k, u_k) + J^*_{k+1}(x_{k+1}) \\ \text{s.to} \quad & x_{k+1} = f_k(x_k, u_k) \\ & u_k \in \mathcal{U}_k(x_k) \end{aligned}$$

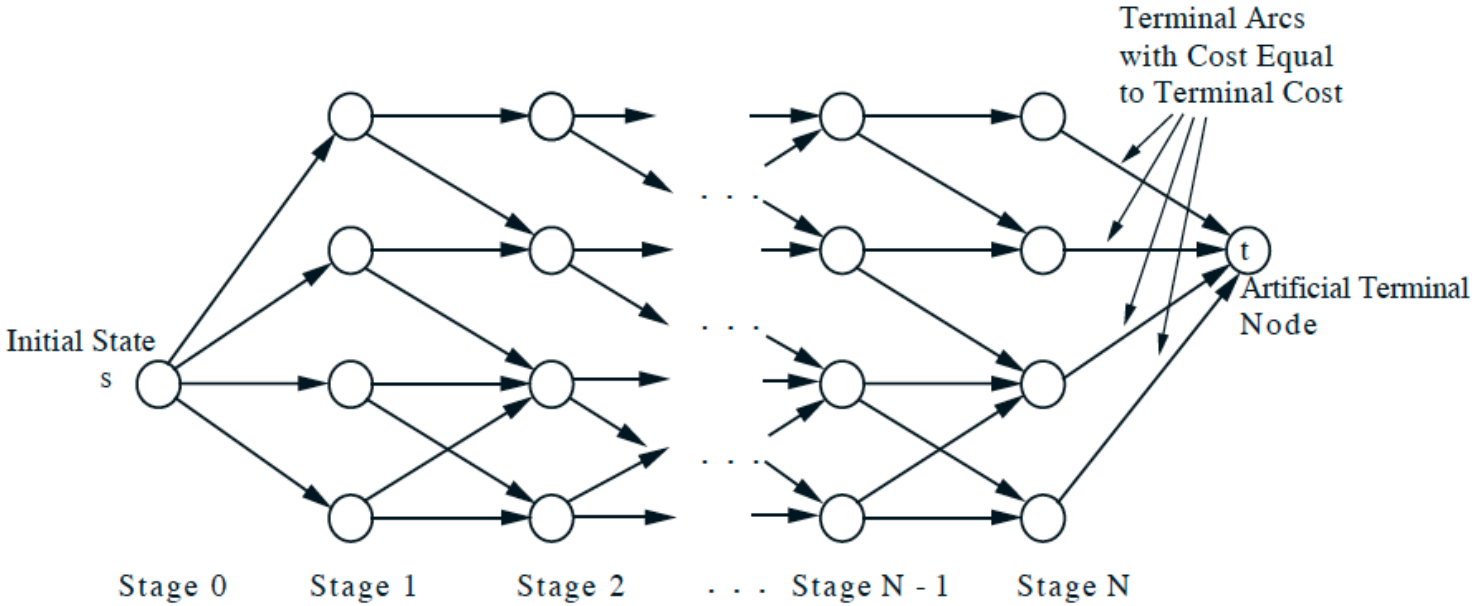
or equivalently

$$\begin{aligned} J^*_k(x_k) = \min_{u_k} \quad & c_k(x_k, u_k) + J^*_{k+1}(f_k(x_k, u_k)) \\ \text{s.to} \quad & u_k \in \mathcal{U}_k(x_k) \end{aligned}$$

until you reach the initial state x_1

Finite-state problems

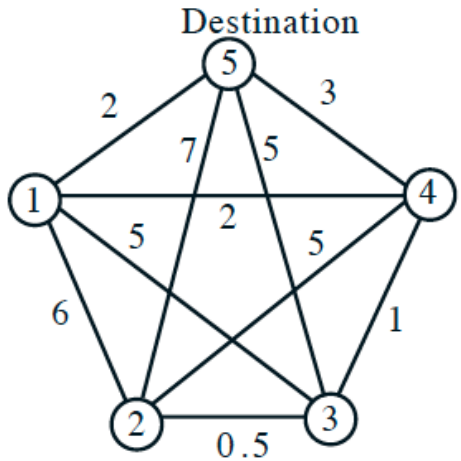
Number of possible states per stage is discrete

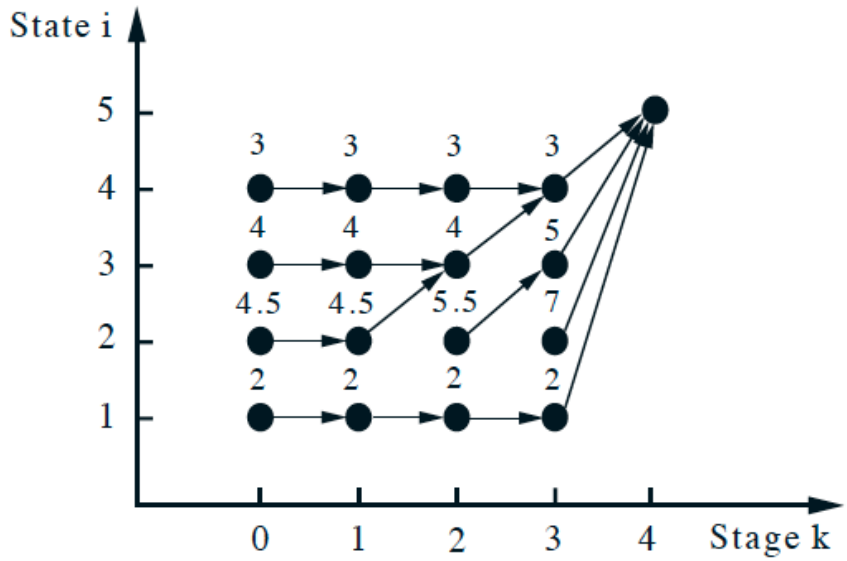


- nodes correspond to state values; arcs correspond to actions
- c_{ij}^k: transitioning cost from state i to j at stage k, i.e., $c_k(x_k = i, u_{i \to j})$
- c_{ii}^N: cost of terminating at state x_i
- minimum-cost path yields optimal cost and actions

Shortest path problem

In a graph with weighted edges and N nodes, find the shortest paths from any node to a destination node d





- **State**: the node you are at time n $\{1, 2, 3, 4, 5\}$
- **Decision cost**: c moving from one node to another (edge weight)
- **Cost** $J_k(i)$: minimum cost for moving from i to d within $N - k$ steps
- **Stages**: N since within N steps you can reach d from any node

Solving a simple UC problem with DP

Start-up costs $s_i(0,1)$; shut-down costs $s_i(1,0)$; and $s_i(0,0) = s_i(1,1) = 0$

$$\min_{\{P_i^t, u_i^t\}_{i,t}} \sum_{t=1}^{T} \sum_{i=1}^{2} u_i^t C_i(P_i^t) + s_i(x_i^t, u_i^t)$$

$$\text{s.to} \quad \left.\begin{array}{l} \sum_{i=1}^{2} P_i^t = D^t, \ \forall\, t \\ u_i^t P_i^{\min} \leq P_i^t \leq u_i^t P_i^{\max}, \ u_i^t \in \{0,1\}, \ \forall\, i,t \end{array}\right\} \quad \text{control options}$$

$$x_i^{t+1} = u_i^t \qquad \text{dynamic system!}$$

- **Stages**: time periods $t = 1, \ldots, T$
- **States**: three possible configurations $\{(0,1),(1,0),(1,1)\}$ per stage t
- **Actions**: (u_i^t, P_i^t) for $i = 1, 2$
- **Termination cost**: no cost for being online or offline at time $T+1$
- $J_t^*(x_i^t)$: minimal total cost from t to T if starting with configuration x_i

Summarizing

Consider $N = 6$ units to be dispatched over $T = 24$ hours for the next day

Exhaustive search: entails solving $2^{NT} = 2.2 \cdot 10^{43}$ ED problems!

Branch-and-bound method: avoids solving entire "sub-trees" of ED problems

- finds optimal solution, but not in deterministic time
- complexity depends on (ℓ, u) and order of visiting UC cases in the queue
- tight initial (ℓ, u) can reduce complexity significantly

Dynamic programming: exploits recursive structure to check fewer UC cases

- entails solving $2^N \times T = 1,536$ single-period UC problems
- each smaller UC involves 2^N ED problems, or can be solved with B&B
- DP still suffers from combinatorial complexity in N

Lagrangian relaxation

Consider the single-period UC problem

$$\min_{\{(P_i,u_i)\in\mathcal{S}_i\}_i} \quad \sum_{i=1}^{N} C_i(P_i,u_i)$$

$$\text{s.to} \quad \sum_{i=1}^{N} P_i = D$$

where $\mathcal{S}_i = \{(P_i,u_i) : u_i P_i^{\min} \le P_i \le u_i P_i^{\max}, u_i \in \{0,1\}\}$

Lagrangian function: $L(\{(P_i,u_i)\};\lambda) = \sum_{i=1}^{N} \left(C_i(P_i,u_i) - \lambda P_i\right) + \lambda D$

- *separable* over generators
- for fixed λ, it can be minimized independently per generator i

$$\min_{(P_i,u_i)\in\mathcal{S}_i} C_i(P_i,u_i) - \lambda P_i$$

Updating the dual variable

Dual function: $g(\lambda) = \min_{\{(P_i,u_i)\in\mathcal{S}_i\}} \sum_{i=1}^{N} \left(C_i(P_i,u_i) - \lambda P_i\right) + \lambda D$

- find λ^* by maximizing $g(\lambda)$, which is always a concave function

- minimize Lagrangian function separately per generator i for λ^k

$$(P_i(\lambda^k), u_i(\lambda^k)) = \arg \min_{(P_i,u_i)\in\mathcal{S}_i} C_i(P_i,u_i) - \lambda^k P_i \quad \forall\ i$$

  two easy problems: one for $u_i = 0$ and one for $u_i = 1$

- power imbalance $D - \sum_{i=1}^{N} P_i(\lambda^k)$ serves as sort of gradient of $g(\lambda^k)$

- update dual variable through dual ascent and iterate

$$\lambda^{k+1} = \lambda^k + \mu \left(D - \sum_{i=1}^{N} P_i(\lambda^k) \right)$$

Lagrangian relaxation for multi-period UC

$$\min_{\{\{(P_i^t,u_i^t)\}_t\in\mathcal{S}_i\}_i} \quad \sum_{i=1}^{N}\sum_{t=1}^{T} C_i^t(P_i^t,u_i^t)$$

$$\text{s.to} \quad \sum_{i=1}^{N} P_i^t = D^t \;\forall\, t \qquad \longleftarrow \quad \boldsymbol{\lambda} = [\lambda_1 \;\cdots\; \lambda_T]^\top$$

- minimize Lagrangian per unit i for $\boldsymbol{\lambda}^k$ and over the entire horizon

$$\{P_i^t(\boldsymbol{\lambda}^k), u_i^t(\boldsymbol{\lambda}^k)\}_{t=1}^T = \arg\min_{\{(P_i^t,u_i^t)\}_t\in\mathcal{S}_i} \sum_{t=1}^{T} C_i^t(P_i^t,u_i^t) - \lambda_t^k P_i^t \quad \forall\, i$$

  this problem is solved via DP with only two states $u_i^t \in \{0,1\}$ per stage!

- update dual variables through dual ascent and iterate

$$\lambda_t^{k+1} = \lambda_t^k + \mu\left(D^t - \sum_{i=1}^{N} P_i^t(\boldsymbol{\lambda}^k)\right) \quad \forall\, t$$

Comments on Lagrangian relaxation

- subgradient iterations for dual problem converge for diminishing step size
- a.k.a. *dual decomposition* if primal is convex
- not optimal for UC due to non-convexity
- LR output may be infeasible (power balance not precisely satisfied)
- relative duality gap $\frac{p^* - d^*}{d^*}$ decreases with increasing N
- LR can be used to initialize a branch-and-bound scheme or it can be heuristically adjusted to yield feasibility

D. Bertsekas, G. Laurel, N. Sandell, T. Posbergh, "Optimal Short-Term Scheduling of Large-Scale Power Systems," *IEEE Trans. on Automatic Control*, Vol. 28, No. 1, Jan. 1983, pp. 1–11.