

# **Security Constrained Economic Dispatch With Post-Contingency Corrective Rescheduling**

**Herminio Pinto, Brian Stott**

FERC Conference, June 23-24, 2010, Washington DC

*Enhanced real-time optimal power flow market models*

# SCED Security in the Future Smart Grid?

- **SCED-P**

- Dispatch as now, with “preventive” security
- System survives contingencies under automatic control action

- **SCED-C**

- Dispatch with “corrective” security – lower operating costs
- Post-contingency corrective rescheduling removes (mostly thermal) operational violations
- Currently research-level, complex algorithms, big computation

- **SCED-PC**

- Most practical: SCED-P for some cases and SCED-C for others

# SCED-C Timelines

When	What	Who
1978	SCED-C mode posited	Stott, Hobson
1985-88	Algorithmic breakthroughs via Benders decomposition	Granville, Pereira, Monticelli, et al
1990	Parallel computing implementations	Pinto, Pereira, et al
1991-97	Several papers in minor publications	Various authors
1998	Method for valuing demand-side corrective security	Strbac, Ahmed, Kirschen, Allan
2007	Viability constraints	Capitanescu, Wehenkel
2009	Re-examination by industry	Litvinov, Zhao

## Dispatching in SCED-C Mode

- Corrective rescheduling can include dispatchable:
  - Generator MWs
  - Phase-shifting transformer angles
  - HVDC flows, FACTS settings
  - Loads (curtailable/interruptible – big Smart Grid potential)
  - Transmission switching
- Preventive dispatching is often needed for Q-V constraints

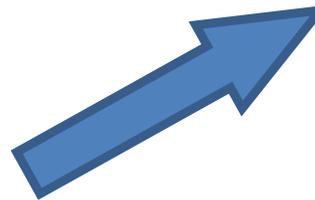
## Key SCED-C Questions

1. Will future EMS real-time reliability permit such operation?
2. How much will SCED-C change security margins?
3. How big is the economic incentive to consider SCED-C?
4. Is real-time SCED-C actually possible?
5. How can we find out about questions 2 - 4?

Step 1:  
Analyze SCED-C  
algorithms and  
computation

Step 2:  
Build a full-scale,  
full-featured, high-  
efficiency SCED-PC  
prototype

Step 3:  
Start studies



# SCED-C Pre-requisites for Real-Time Use

- Functional versatility
  - Modeling versatility
  - Algorithmic reliability
  - Computational speed
  - Solution practicality
- We have been addressing these questions
- All complex engineering optimization problems depend on:
    - A myriad of heuristic algorithmic customizations
    - Superimposed on a suitable base optimizing technology (e.g. LP, NLP, MIP)
    - Canned packages on their own will generally not give usable engineering solutions

# SCEC-PC Problem Formulation (simplified)

Dispatch base-case controls  $\mathbf{u}_o$  within their limits to minimize a piecewise linear (including staircase bids) or nonlinear objective function (generally non-concave)

$$\mathbf{c}(\mathbf{u}_o)$$

subject to security constraints:

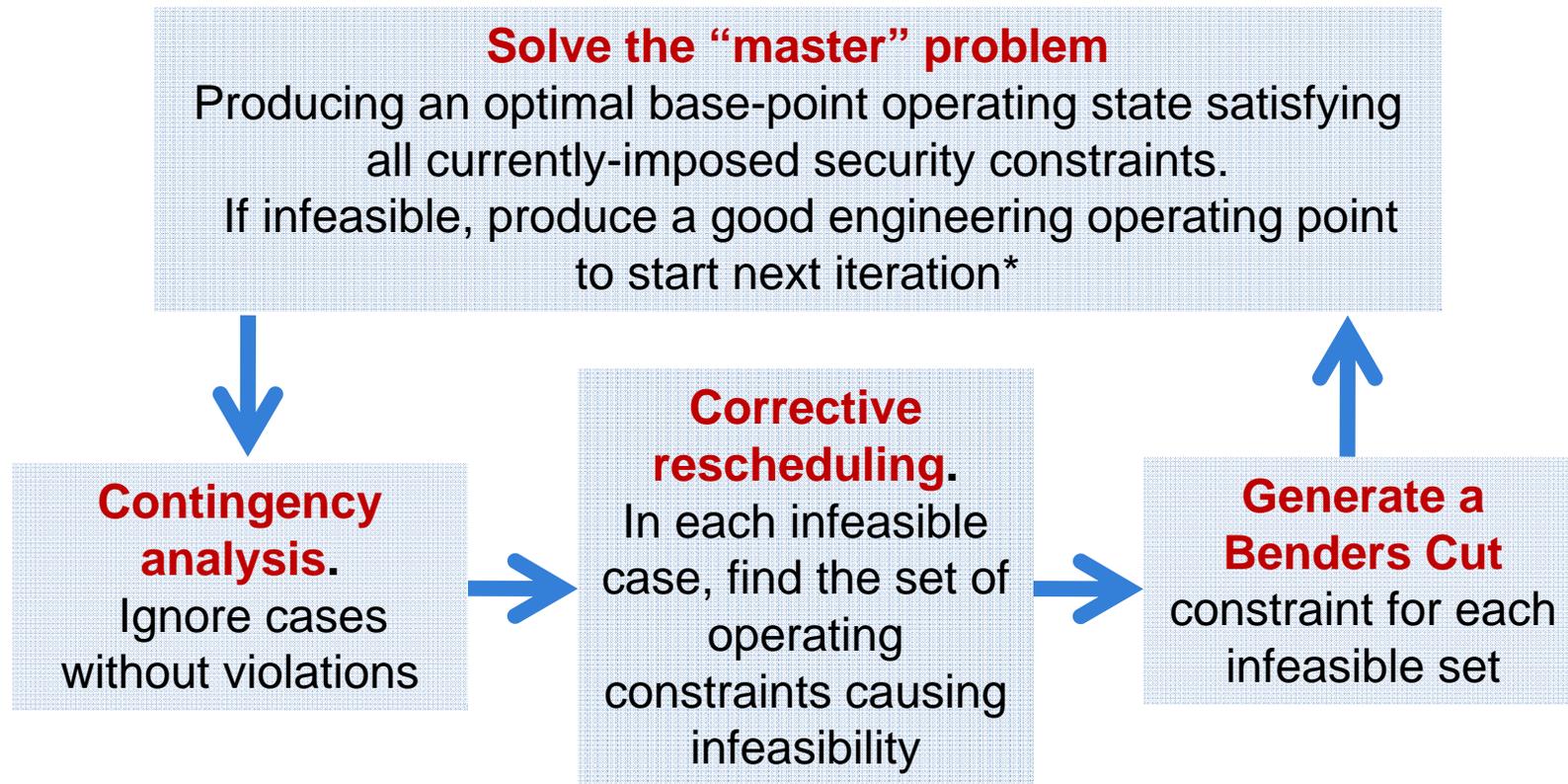
constraints	type	description
$\mathbf{a}_o(\mathbf{u}_o, \mathbf{x}_o) \geq \mathbf{0}$	base case	pre-contingency operation
$\mathbf{a}_i(\mathbf{u}_o, \mathbf{x}_i) \geq \mathbf{0}$	preventive	u not rescheduled
$\mathbf{a}_i(\mathbf{u}_i, \mathbf{x}_i) \geq \mathbf{0}$	corrective	u rescheduled to remove post-contingency violations
$\Delta_{Li} \leq \mathbf{u}_i - \mathbf{u}_o \leq \Delta_{Ui}$	coupling	u is subject to ramp rates

x = state variables, subscript i = for each contingency i = 1...n

# Some Practical SCED Solution Requirements

- Model discontinuities
  - Even linearized networks undergo abrupt model discontinuities
- Equitable dispatch under (frequent) degeneracy
  - With fixed-price bids and/or linearly dependent constraints
- Infeasibility
  - Identify the set of operating constraints contributing to infeasibility
  - Produce a sensible engineering solution (typically WLS violations)
- Ineffective rescheduling
  - Avoid massive rescheduling for little effect on constraints
- LMP decontamination (an imperative in markets, and not straightforward)
  - Locally-acting controls, breakpoint solutions, penalties, dummy objectives

# Main Steps in a SCED-C Solution



*The above has 4 concurrent convergence processes: for nonlinearities, discontinuities, binding set search, and Benders approximations. It is critical to keep the number of cycles low*

*\* Without this, the no. of cycles can become v. large*

## Benders Decomposition for SCED-C

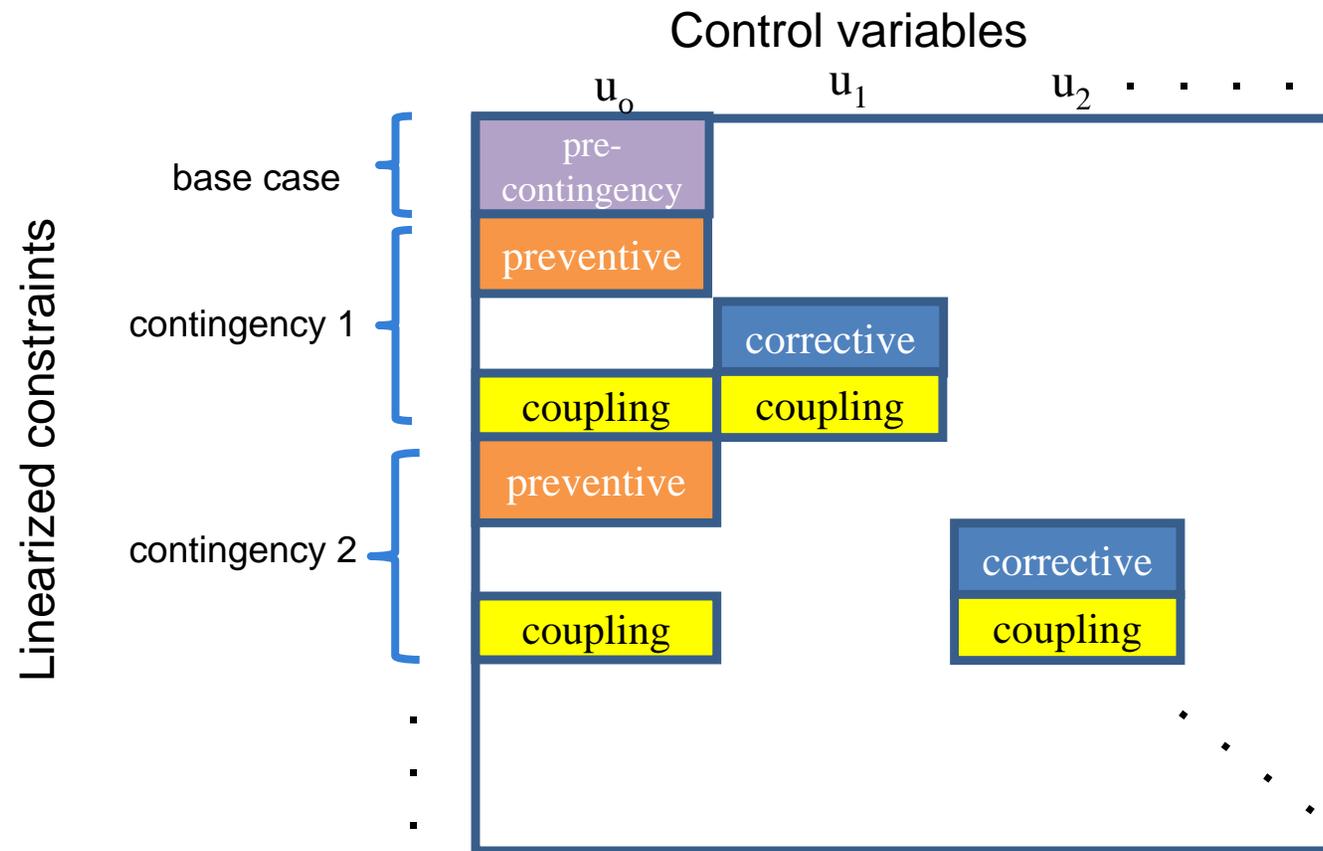
- Applied after an infeasible corrective rescheduling case
- Aggregates all the case's violated constraints into a single linear approximating Benders Cut constraint
- All such Benders Cut constraints are then added to the master problem
- The process is iterated until convergence



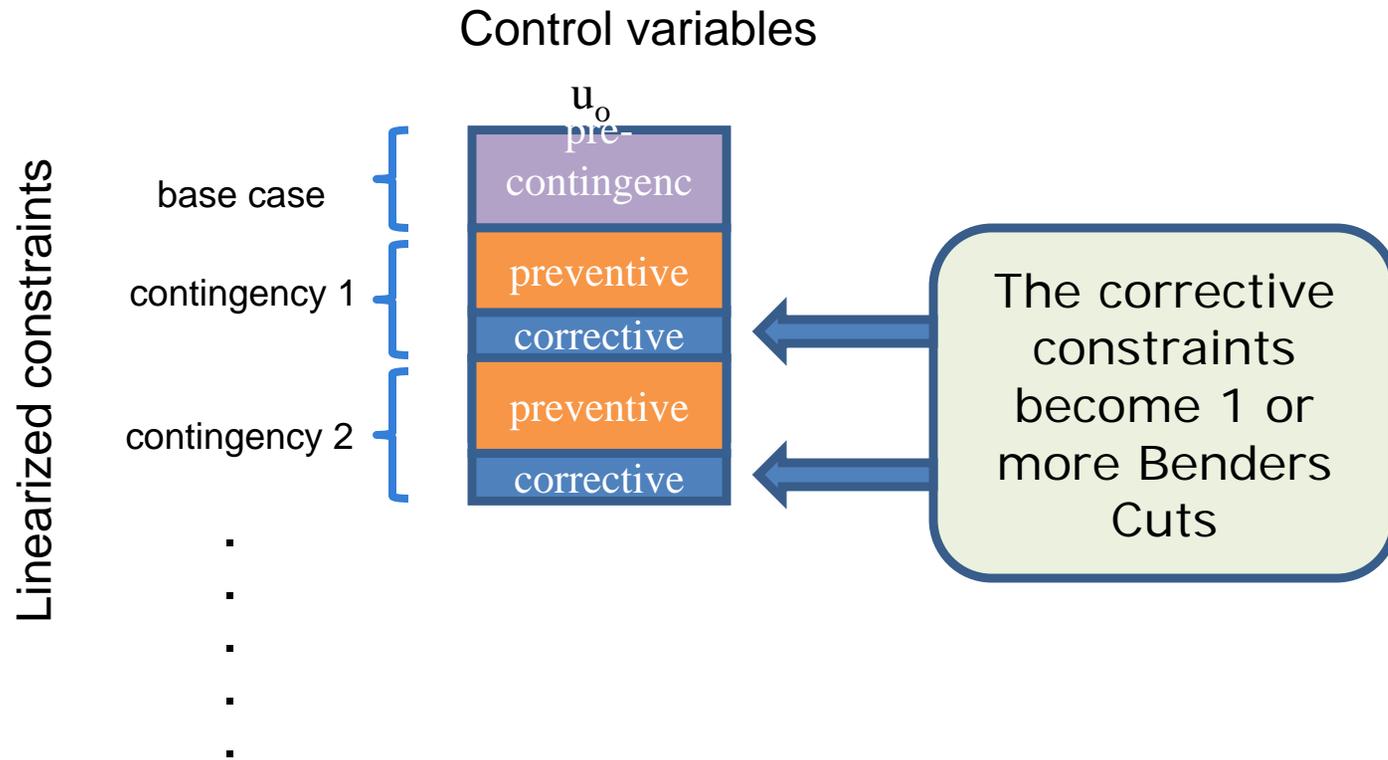
Corrective objective can be coupling constraint violations or WLS operating violations

The master solution should "see" as many bottleneck constraints as possible, to avoid excess iterations

# Control-Constraint Structure Before Decomposition



# Control-Constraint Structure After Decomposition



The master problem is now reduced to the optimization of the base-case variables.

## Versatility

- Nonlinear and discontinuous models can be handled
- Benders decomposition also handles corrective switching actions (mixed-integer)
- Suitable for real-time SCED, Day-Ahead SCUC, planning

## Reliability - Robustness

- Linear network models and non-concave cost curves
  - Good reliability, higher chance of uniqueness
- AC network models, non-convex cost curves, binary controls
  - No guarantees for convergence or uniqueness

# Computing Effort

Iterative Step	Description	Comments
Master problem	Large no. of nonsparse post-contingency constraints, both preventive and corrective	Only one Benders Cut constraint per infeasible contingency per iteration. Hundreds will be no problem if the master optimization is very efficient
Contingency analysis	Same per iteration as in conventional SCED-P	Reduced by screening, critical set heuristics. Ideal for minimization of elapsed time by MPI multi-processing
Corrective rescheduling	Large no. of network-constrained optimizations	The greatest computing effort. Ideal for minimization of elapsed time by MPI multi-processing
Generating Benders Cuts	One cut per infeasible contingency case	Little computing effort

# Real-Time Performance

- Prerequisites
  - Fast optimizing and contingency analysis engines
  - Enough processors for corrective rescheduling (MPI)
  - Multi-threaded master optimization
  - Good iteration-reducing heuristics
- Already known
  - SCED-P already achieves real-time speed requirements with incrementally linearized network models
- A guesstimate
  - SCED-PC can take only 2-3 times longer elapsed time than SCED-P
  - If real time SCED-P is currently achievable, SCED-PC with similar models should be no problem in future

# Conclusions

- The corrective SCED calculation
  - Is not the monster it was once thought to be
    - Even on the biggest power systems
    - At least, with simplified models (e.g. linearized network)
  - A high degree of problem-specific implementation is needed
- SCED calculation reliability
  - Is likely to be OK with well-behaved formulations
- Large-scale prototyping and testing is needed
  - On security margins, economic benefits, algorithmic reliability, performance