Uma Abordagem Usando *Branch and Bound*, *Cutting Planes* e *Branch and Cut* para o Problema de Planejamento da Operação de Sistemas Elétricos com Critério de Segurança n-K

Thuener Silva
Departamento de Engenharia Industrial – Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 225 / 4º andar RDC – Gávea, Rio de Janeiro / RJ – 22451-900
thuener@esp.puc-rio.br

Murilo Soares
Departamento de Engenharia Elétrica – Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 225 / L960 – Gávea, Rio de Janeiro / RJ – 22451-900
murilopsoares@gmail.com

RESUMO

O planejamento da operação de sistemas elétricos de potência busca, em sua etapa de estruturação da operação, realizar a programação que minimize o valor esperado do custo total de operação, sujeito ao atendimento da demanda em cada barra do sistema. Para que a demanda seja atendida, mesmo no caso de falha de alguns componentes do sistema, é necessário considerar modelos de otimização nos quais as possíveis falhas são modeladas. Nesse trabalho apresentamos a formulação desse problema, assim como implementação e aplicação de métodos de *Branch and Bound*, *Cutting Planes* e finalmente a junção ambos os métodos para construção do *Branch and Cut*. Por fim, avaliamos o desempenho desses métodos em três instâncias diferentes, sob diferentes condições de falhas, em relação ao tempo necessário e ao atingimento ou não da solução ótima.
PALAVRAS CHAVE. *Branch and Cut*. *Branch and Bound*. Planos de Cortes. Despacho curto-prazo. Planejamento sistemas elétricos. critério de segurança n-k.

ABSTRACT

The planning of the operation of electric power systems seeks, in its stage of structuring the operation, to carry out the programming that minimizes the expected value of the total cost of operation, subject to the demand of each bar in the system. In order for the demand to be met, even in the event of failure of some system components, it is necessary to consider optimization models in which the possible faults are modeled. In this work, we present the formulation of this problem, as well as the implementation and application of methods of Branch and Bound, Cutting Plans and after joining both methods we building the Branch and Cut approach. Finally, we evaluate the performance of these methods in three different instances, under different fault conditions, in relation to the time required and to the achievement, or not, the optimal solution.
KEYWORDS. Branch and Cut. Branch and Bound. Cutting Planes. Short-term dispatch. Power system planning. n-k security criterion.

https://proceedings.science/p/85385?lang=pt-br

Introdução

O problema de despacho de curto prazo tem importância indiscutível no planejamento e operação de sistemas elétricos. Ao contrário do planejamento de médio prazo, onde incertezas relacionadas à hidrologia das usinas hidrelétricas e à demanda são de fundamental importância, na programação diária a principal incerteza é a configuração do sistema, que está sujeita a falhas. Apesar de grandes falhas serem improváveis, são eventos nos quais o atendimento à demanda é comprometido e pode deixar diversas regiões sem energia elétrica por horas.

Eventos extremos deste tipo já ocorreram em passado recente no Brasil, como por exemplo em 2009, quando o desligamento simultâneo (por condições meteorológicas adversas) de três linhas que conectavam Itaipu ao Sistema Interligado Nacional ocasionou um apagão em grande parte do território brasileiro.

Diante da impossibilidade de prever eventos deste tipo, é necessária a consideração de falhas em equipamentos do sistema durante a programação da operação das usinas, de forma que, em eventuais situações de falha, seja possível continuar suprindo a demanda do sistema com uso de reserva girante alocada nas usinas programadas que estiverem disponíveis.

O principal objetivo é, portanto, dada uma configuração da rede elétrica, com suas usinas geradoras, linhas de transmissão, barras e demandas em cada barra, encontrar a programação de cada gerador acoplado ao sistema de modo que seja possível atender a demanda de cada barra ao menor custo possível, mesmo em situações de falhas de elementos do sistema.

No modelo que tratamos nesse trabalho, a segurança é incorporada através da possibilidade de alocação de reservas girantes em cada gerador, assim como por meio da adição de restrições que garantam o atendimento da demanda nas possíveis situações de falha. Reservas girantes representam uma parcela disponível para aumento ou redução na produção dos geradores que já fazem parte do sistema e que estão despachados. Assim, a reserva de um gerador foi modelado como de dois subtipos: reserva de subida e reserva de descida, referindo-se, respectivamente, à quanto ele pode aumentar ou diminuir à sua geração programada inicialmente.

Formulação do problema de Despacho com critério n-K

A formulação que será apresentada trata do problema de programação da operação de sistemas elétricos com N elementos passíveis de falhas, tais como linhas de transmissão e geradores, com critério de segurança que permita que, mesmo no caso de falha de K elementos, a demanda de cada barra do sistema seja atendida plenamente.

Neste problema, originalmente descrito em Street et al. [2014] e similar ao Street et al. [2011], cada gerador i pode produzir p_i para atender as demandas das barras, D_b. Caso seja despachado ($v_i = 1$), o gerador i deve gerar pelo menos ($\underline{P}_i$) e no máximo ($\overline{P}_i$). Adicionalmente, é possível alocar energia de reserva no gerador de forma que, em caso de contingência, seja possível aumentar (r_i^U) ou reduzir (r_i^D) sua geração. São admitidas até K falhas simultâneas no sistema, o que define o chamado critério n-K.

As equações (1)–(6) descrevem a formulação considerada neste trabalho. A função objetivo (1) é minimizar o custo total de operação, dado pela soma dos custos fixos e de produção de cada unidade despachada ($C_i^P(p_i, v_i)$), mais os custos das reservas de subida ($C_i^U r_i^U$) e descida ($C_i^D r_i^D$) de cada usina. As equações (2) e (10) garantem o atendimento à demanda D_b em cada barra na condição de nenhuma falha e com K falhas, respectivamente. As equações (3) e (11), por sua vez, determinam o fluxo circulante em cada linha de transmissão do sistema em condições normais (f_ℓ), e nas condições de falhas(f_ℓ^k). Sendo $\theta_{fr(\ell)}$ o ângulo de fase na barra $fr(\ell)$. Os fluxos nas linhas são limitados pela equação (4).

https://proceedings.science/p/85385?lang=pt-br

$$\underset{f_\ell, p_i, \theta_b, v_i, r_i^U, r_i^D, f_\ell^k, p_i^k, \theta_b^k}{\text{Minimizar}} \sum_{i \in I} (C_i^P(p_i, v_i) + C_i^U r_i^U + C_i^D r_i^D) \tag{1}$$

$$\sum_{i \in I_b} p_i + \sum_{\ell \in \mathcal{L} | to(\ell) = b} f_\ell - \sum_{\ell \in \mathcal{L} | fr(\ell) = b} f_\ell = D_b \tag{2}$$

$$f_\ell = \frac{1}{x_\ell}(\theta_{fr(\ell)} - \theta_{to(\ell)}); \forall \ell \in \mathcal{L} \tag{3}$$

$$-\overline{F}_\ell \leq f_\ell \leq \overline{F}_\ell; \forall \ell \in \mathcal{L} \tag{4}$$

$$p_i + r_i^U \leq \overline{P}_i v_i; \forall i \in I \tag{5}$$

$$p_i - r_i^D \geq \underline{P}_i v_i; \forall i \in I \tag{6}$$

$$0 \leq r_i^U \leq \overline{R}_i^U v_i; \forall i \in I \tag{7}$$

$$0 \leq r_i^D \leq \overline{R}_i^D v_i; \forall i \in I \tag{8}$$

$$v_i \in \{0, 1\}; \forall i \in I \tag{9}$$

$$\sum_{i \in I_b} p_i^k + \sum_{to(\ell) = b} f_\ell^k - \sum_{fr(\ell) = b} f_\ell^k = D_b; \forall b \in N, \forall k \in C \tag{10}$$

$$f_\ell^k = \frac{A_\ell^k}{x_\ell}(\theta_{fr(\ell)}^k - \theta_{to(\ell)}^k); \forall \ell \in \mathcal{L}, \forall k \in C \tag{11}$$

$$-\overline{F}_\ell \leq f_\ell^k \leq \overline{F}_\ell; \forall \ell \in \mathcal{L}, \forall k \in C \tag{12}$$

$$p_i^k \geq A_i^k(p_i - r_i^D); \forall i \in I, \forall k \in C \tag{13}$$

$$p_i^k \leq A_i^k(p_i + r_i^U); \forall i \in I, \forall k \in C \tag{14}$$

A geração máxima das usinas é respeitada em todas as configurações do sistema ($\overline{P}_i$) graças às equações (5) e (14). Da mesma forma, a geração mínima das usinas despachadas ($\underline{P}_i$) é garantida por meio das equações (6) e (13). As reservas usadas para atender a demanda em situações de contingência (r_i^U e r_i^D) são limitadas pelas equações (7) e (8).

O problema descrito é, portanto, de programação inteira com variáveis binárias, e com um conjunto de restrições (10) a (14) que são combinatórias em função do critério de segurança n-K adotado.

As seções a seguir tratam dos algoritmos que foram usados para resolver o problema.

Algoritmo *Branch and Bound*

O primeiro algoritmo implementado foi o *Branch and Bound* nele utilizamos uma lista com os nós a serem processados e sempre retiramos o elemento que foi inserido mais cedo na lista.

Para gerar um limite superior para o problema implementamos uma heurística simples para tentar encontrar uma solução inteira. Relaxamos a restrição de geração mínima $p_i - r_i^D \geq \underline{P}_i v_i; \forall i \in I$ para que a geração seja apenas maior ou igual a zero, ou seja, $p_i - r_i^D \geq 0; \forall i \in I$. Depois de modificar essa restrição executamos uma relaxação linear e verificamos para cada usina se a geração menos a reserva de descida que é maior que o limite mínimo de geração, $p_i - r_i^D \geq \underline{P}_i v_i$. Se sim, colocamos a usina ligada, caso contrário a usina é colocada desligada e uma programação linear decidia qual o valor das variáveis contínuas. Porém, essa configuração das usinas pode não ser uma solução viável, nesse caso ligava-se todas as usinas e obviamente essa é uma solução viável.

Atualizamos o limite inferior do *Branch and Bound* através de uma relaxação linear. Caso a relaxação linear obtivesse uma solução inteira o resultado seria um limite superior e inferior ao mesmo tempo e, nesse caso, fazemos uma poda por otimalidade.

A cada novo nó do *Branch and Bound* uma nova usina era fixada como ligada ou desligada, sendo que os valores referentes a essa variável na matriz de restrições eram multiplicados pelo valor da usina e retirados do lado direito da equação. Estes também eram retirados da função objetivo do problema e colocados como um valor fixo a ser acrescentado na função objetivo. Essa diminuição ajuda a reduzir o tamanho do problema e faz com que, quanto mais fundo estejamos na árvore do *Branch and Bound*, menor seja o problema.

https://proceedings.science/p/85385?lang=pt-br

A ordem das usinas que seriam ligas ou desligadas a cada nó foi um fator determinante no número de nós criados e no tempo de execução. Mais detalhes serão apresentados na Seção 6.

Algoritmo *Cutting Planes*

O problema em questão é um problema de programação inteira mista com variáveis limitadas. Uma possível forma de tratar este tipo de problema poderia ser com o algoritmo de *Cutting Planes*, ou plano de cortes, inserindo limites nas variáveis como restrições no problema e resolvendo conforme descrito em Wolsey [1998]. No entanto, devido ao grande número de restrições presentes, o problema de programação linear cresceria demasiadamente caso essa fosse a abordagem.

Uma forma de considerar diretamente os limites das variáveis no cálculo dos cortes de Gomory é descrita nos trabalhos Goycoolea [2006] e Cook et al. [2009]. Em Goycoolea [2006] é feita uma introdução ao algoritmo de plano de cortes e a versão para problemas inteiro mistos é descrita com detalhes. Posteriormente, Goycoolea [2006] mostra como considerar variáveis com limites superior e inferior no cálculo dos cortes, embora na parte final da dedução haja um equívoco na expressão do corte, corrigido pelo trabalho Cook et al. [2009]. Este último trabalho tem por objetivo mostrar como calcular, de forma numericamente segura, os cortes para problemas de programação inteira mista com variáveis limitadas.

Uma breve dedução do cálculo dos cortes para o problema com variáveis limitadas será mostrada a seguir, sendo fortemente baseada nos trabalhos Goycoolea [2006] e Cook et al. [2009].

Considere o conjunto de variáveis inteiras e contínuas:

$$Q = \{x \in \mathbb{R}, \omega \in \mathbb{Z} : \omega + x = b, x \geq 0\}. \tag{15}$$

A restrição

$$x \geq \hat{b}(\lceil b \rceil - \omega) \tag{16}$$

é válida para Q, conforme mostrado em Goycoolea [2006].

Considere também $a \in \mathbb{Q}^n, b \in \mathbb{Q}$ e os conjuntos disjuntos I, C tal que $I \cup C = \{1, ..., n\}$. Seja $C^+ = \{j \in C : a_j > 0\}$, $C^- = \{j \in C : a_j \leq 0\}$ e considere o poliedro:

$$P^= = \left\{ x \in \mathbb{Q}^n : \sum_{j=1}^{n} a_j x_j = b, x \geq 0, x_j \in \mathbb{Z} \forall j \in I \right\} \tag{17}$$

Este conjunto $P^=$ é definido por valores de x que satisfaçam a equação $\sum_{j=1}^{n} a_j x_j = b$. No caso do plano de cortes, essa equação será uma linha do *tableau* no qual se tenha uma variável $x_j \in I$ que seja fracionária no ótimo da relaxação linear do problema.

Se $S \subset I$ e $\hat{q} = q - \lfloor q \rfloor$, repare que para todo $x \in P^=$, podemos reescrever $P^=$ como:

$$\underbrace{\left(\sum_{S} \lfloor a_j \rfloor x_j + \sum_{I \setminus S} \lceil a_j \rceil x_j \right)}_{w} + \left(\sum_{S} \hat{a}_j x_j - \sum_{I \setminus S} (1 - \hat{a}_j) x_j + \sum_{C} a_j x_j \right) = b \tag{18}$$

Como $-(1 - \hat{a}_j)x_j$ e $\sum_{C^-} a_j x_j$ são não-positivos, temos que:

$$w + \left(\sum_{S} \hat{a}_j x_j + \sum_{C^+} a_j x_j \right) \geq b.$$

https://proceedings.science/p/85385?lang=pt-br

Sabemos que $w \in \mathbb{Z}$ e $\left(\sum_S \hat{a}_j x_j + \sum_{C^+} a_j x_j\right) \in \mathbb{R}_+$, e podemos, a partir de 16, escrever:

$$\left(\sum_S \hat{a}_j x_j + \sum_{C^+} a_j x_j\right) \geq \hat{b}(\lceil b \rceil - w). \tag{19}$$

Substituindo w obtemos a MIR (*Mixed Integer Rounding*):

$$\sum_S (\hat{a}_j + \hat{b}\lfloor a_j \rfloor)x_j + \sum_{I \setminus S} (\hat{b} + \hat{b}\lfloor a_j \rfloor)x_j + \sum_{C^+} a_j x_j \geq \hat{b}\lceil b \rceil.$$

que pode ser escrita como:

$$\sum_{i \in I} f(a_i, b)x_i + \sum_{i \in C} h(a_i)x_i \geq \hat{b}\lceil b \rceil \tag{20}$$

onde $f(q_1, q_2) = min\{\hat{q}_1, \hat{q}_2\} + \hat{q}_2\lfloor q_1 \rfloor$ e $h(q) = max\{q, 0\}$.

As inequações do tipo MIR consideram apenas variáveis não negativas, mas não permitem considerar variáveis com limites superiores, por exemplo. Acrescentando limites inferiores e superiores teríamos a seguinte formulação:

$$P_B = \left\{ x \in \mathbb{Q}^n : \sum_{j=1}^{n} a_j x_j = b, l_j \leq x_j \leq u_j,\ \forall j \in \{1, ..., n\},\ x \geq 0, x_j \in \mathbb{Z} \forall j \in I \right\} \tag{21}$$

Definindo as variáveis $x_j^l = (x_j - l_j)$ e $x_j^u = (u_j - x_j)$ para $j = 1, ..., n$, se $u_j < \infty$, podemos reparametrizar o problema em x_j^l e x_j^u, e aplicar a metodologia MIR definida anteriormente para problemas inteiro mistos com variáveis não negativas.

Considerando que U, L definem uma partição de $\{1, ..., n\}$ tal que $\forall j \in U, u_j < \infty$, substituímos:

- x_j por $(u_j - x_j^u)$ quando $j \in U$;
- x_j por $(x_j^l + l_j)$ quando $j \in L$.

Obtendo:

$$\sum_{j \in U} (-a_j)x_j^u + \sum_{j \in L} a_j x_j^l + \sum_{j \notin U \cup L} a_j x_j \geq b - \sum_{j \in U} a_j u_j - \sum_{j \in L} a_j l_j. \tag{22}$$

Repetindo os passos que fizemos para o *MIR*, fazendo
$r = b - \sum_{j \in U} a_j u_j + \sum_{j \in L} a_j l_j$ e retornando às variáveis originais obtemos:

$$-\sum_{U \cap I} f(-a_j, r)x_j + \sum_{I \setminus U} f(a_j, r)x_j - \sum_{U \cap C} h(-a_j)x_j + \sum_{C \setminus U} h(a_j)x_j \geq R \tag{23}$$

onde,

$$R = \hat{r}\lceil r \rceil - \sum_{U \cap I} f(-a_j, r)u_j + \sum_{L \cap I} f(a_j, r)l_j - \sum_{U \cap C} h(-a_j)u_j + \sum_{L \cap C} h(a_j)l_j.$$

A expressão 23 define o corte que será utilizado no problema em questão. O plano de cortes implementado possui as seguintes características:

https://proceedings.science/p/85385?lang=pt-br

- Permite fornecer restrições de igualdade e de desigualdade;
- Implementa o adição de diversos cortes simultaneamente no problema, desde que tenham uma violação mínima;
- Implementa a remoção de cortes inativos durante a solução do problema;
- Permite fornecer variáveis limitadas inferior e superiormente;
- Utiliza como critérios de parada o tempo de processamento e o número de cortes calculados.

O plano de cortes calcula cortes para todas as variáveis que estejam fracionárias em uma mesma iteração, mas somente adiciona os cortes que tem violação maior ou igual a 0.0001. Após a adição dos cortes verificamos se algum dos cortes que foram adicionados desde o início tem variável dual igual a zero. Nesse caso, retiramos esses cortes e suas respectivas variáveis de folgas se as mesmas não estiverem presentes em outras restrições.

Como o plano de cortes pode demorar muito tempo, adicionamos diversos critérios de parada, como:

- tempo máximo utilizado com o plano de cortes;
- número máximo de cortes calculados;
- número máximo de cortes criados que não retiram nenhuma solução;

Quando o valor da função objetivo não alterasse significativamente após uma iteração o plano de cortes também era interrompido.

Algoritmo Branch and Cut

Para construção do algoritmos de *Branch and Cut* utilizamos o planos de cortes ao invés da relaxação linear, o que fornece uma solução inteira ou um limite inferior para o problema. Então a cada nó eram feitos diversos cortes na solução até atingir uma solução inteira ou algum critério de parada.

Se a solução retornada pelo plano de cortes fosse inteira verificávamos se era necessário atualizar o limite superior global e depois o nó atual era podado por otimalidade. Se a solução fosse fracionária utilizamos a solução como um limite inferior e caso a solução fosse inviável o nó era podado por inviabilidade.

Utilizamos o limite superior global dentro do plano de cortes para evitar fazer cortes que não sejam necessários. Então, se em qualquer momento do *Branch and Cut* obtivéssemos uma solução maior que o limite superior, o plano de cortes é interrompido e o nó que está sendo verificado é podado.

Experimentos e Resultados

Para validar e comparar os algoritmos implementados, realizamos experimentos em 3 instâncias diferentes. Uma delas contendo três barras e duas contendo 24 barras. Para essas últimas instâncias, uma delas corresponde à instância RTS 24, sistema padrão de 24 barras do IEEE Grigg et al. [1999], identificada como “24 barras reduzido”. A segunda se refere à instância apresentada em Street et al. [2014], construída a partir da RTS 24, mas duplicando-se algumas linhas de transmissão para que a topologia permitisse atender a demanda sem cortes de carga para até $k = 3$.

Para cada uma das instâncias variamos o critério de segurança entre $k = 0$ e $k = 1$.

A implementação dos algoritmos foi feita na linguagem Matlab R2018a com CPLEX 12.8.0 e linguagem Julia 0.6.2 com Gurobi 7.5.1, importante ressaltar que o CPLEX e Gurobi foram utilizados apenas para resolver os problemas lineares e obter as variáveis básicas da solução. A máquina utilizada para os experimentos foi um AMD FX-8300 3.3GHz com 24GB de memória, importante ressaltar que a execução não foi feita em paralelo.

https://proceedings.science/p/85385?lang=pt-br

A eficácia da heurística foi crucial para o resultado final. Em diversas instâncias o ótimo era obtido através da heurística em poucas iterações.

No plano de cortes dentro Branch and Cut colocamos os seguintes valores para os critérios de parada: tempo de execução máxima de 2 minutos, número máximo de cortes adicionados 300, número máximo de cortes criados que não retiram nenhuma solução 50.

A tabela abaixo descreve, para cada uma das instâncias, o número de identificação da instância, o número de linhas, colunas, o número total de células que a formulação do problema atinge e o seu ótimo. Fica claro, então, como a introdução do critério de segurança aumenta enormemente o tamanho do problema, ainda que seja apenas para $k = 1$.

ID	**Instâncias**	**Linhas**	**Colunas**	**Matriz**	**Ótimo**
1	3 Barras, k = 0	24	17	408	240
2	3 Barras, k = 1	96	65	6240	375.78
3	24 Barras reduzido, k = 0	107	106	11 556	9063.45
4	24 Barras reduzido, k = 1	3614	3010	$1.1 * 10^7$	12123.36
5	24 Barras, k = 0	189	292	55 188	16230.89
6	24 Barras, k = 1	11995	14247	$1.7 * 10^8$	17032.47

Tabela 1

Como mencionando da Seção 3 um fator de suma importância foi a ordem das usinas que eram ligadas a cada nó. Para exemplificar a diferença vamos mostrar o tempo de execução e o número de nós criados no *Branch and Bound* utilizando três tipos de ordenação para as 5 primeiras instâncias na Tabela 2. As usinas eram escolhidas pela ordem que a variável correspondente era referenciada na função objetivo ou pelo valor de cada uma na função objetivo em ordem crescente ou decrescente.

ID.	**Objetivo**		**Crescente**		**Decrescente**	
	Nós	**Tempo**	**Nós**	**Tempo**	**Nós**	**Tempo**
1	9	0.73956	11	0.82515	5	0.59167
2	9	0.60689	11	0.61197	7	0.50899
3	291	21.1886	247	16.3807	11	0.98125
4	967	175.9288	617	115.8882	19	3.9578
5	8197	749.8875	8113	838.6651	25	2.5978

Tabela 2: Diferenças entre os tipos de ordenação.

Analisando a tabela acima verificamos que a diferença entre os tipos de ordenação é exorbitante e que o melhor resultado foi obtido quando escolhemos as usinas em ordem decrescente em relação ao seu custo na função objetivo. Escolhendo os maiores valores podamos nós mais próximos da raiz da arvore de *Branch and Bound* e com isso conseguimos diminuir significativamente o número de *Branchs*.

A principal vantagem do *Branch and Cut* é obtenção de melhores limites superiores e inferiores, o que resulta na diminuição do número de nós que são criados. Esse comportamento pode ser averiguado na Tabela 3, na qual comparamos o número de nós criados pelo *Branch and Bound*(BB) e *Branch and Cut*(BC).

https://proceedings.science/p/85385?lang=pt-br

Instâncias	BB	BC
3 Barras, k = 0	5	3
3 Barras, k = 1	7	7
24 Barras reduzido, k = 0	11	1
24 Barras reduzido, k = 1	19	9
24 Barras, k = 0	25	19
24 Barras, k = 1	53	47

Tabela 3: Número de nós criados.

Como são criados menos nós no *Branch and Cut* o tamanho da lista de elemento a serem processados é reduzido. Como podemos ver na Figura 1 o número de elementos na lista cada iteração é menor quando utilizamos o plano de cortes comparado ao *Branch and Bound*. Se comparamos o maior número de elementos na lista durante toda execução o valor é 2.4 vezes menos, ou seja, uma redução de mais de 50%.

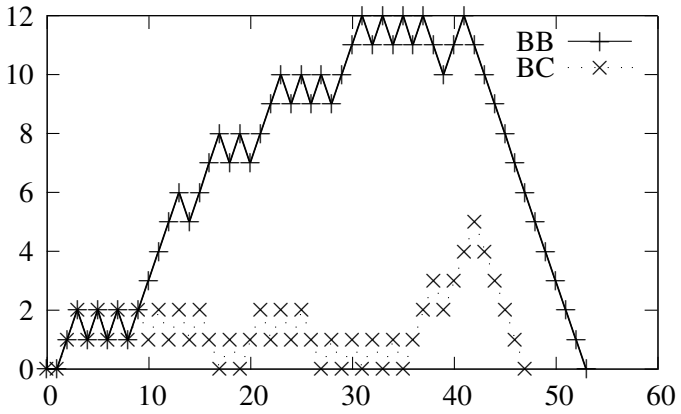


Figura 1: Nós criados e ainda não processado.

Para avaliarmos como o plano de cortes está alterando o limite inferior do problema, apresentamos na Tabela 4 o valor da solução obtida com a relaxação linear e o valor da solução após o plano de cortes.

Instâncias	Relaxação	CP	Diff. %
3 Barras, k = 0	235.45	240	101.93
3 Barras, k = 1	293.94	375.784	127.84
24 Barras reduzido, k = 0	8805.55	9063.45	102.93
24 Barras reduzido, k = 1	11172.28	11597.97	103.81
24 Barras, k = 0	15792.7329	15792.73	100.00
24 Barras, k = 1	16455.9829	16789.65	102.03

Tabela 4: Melhora no limite inferior.

Na Tabela 5 comparamos os tempos de execução de cada algoritmo. Pode-se perceber que o tempo do plano de cortes aumenta muito quando o tamanho da instância aumenta, o que afeta profundamente o tempo do *Branch and Cut*. Esse aumento foi ocasionado principalmente pelo tempo para determinar a matriz inversa utilizada para calcular o dicionário a cada iteração. Em média, para a maior instância, o tempo gasto com essa inversão era maior que 50% do tempo total de execução do algoritmo. Esse problema poderia ser solucionado se a biblioteca do CPLEX para

https://proceedings.science/p/85385?lang=pt-br

Matlab oferecesse o dicionário da solução do problema de programação linear.

Instância	CP	BB	BC
3 Barras, k = 0	0.0549	0.5097	0.4033
3 Barras, k = 1	0.0630	0.4418	0.3527
24 Barras reduzido, k = 0	0.0301	0.7904	0.1589
24 Barras reduzido, k = 1	34.9632*	4.1414	67.2182
24 Barras, k = 0	0.03274*	2.0091	1.6042
24 Barras, k = 1	223.2275*	61.9511	1030.2343

Tabela 5: Tempo de execução para cada algoritmo em Matlab(CPLEX) em segundos.

Para que fosse possível comprar os métodos de solução com maior rigor os algoritmos foram implementados na linguagem Julia utilizando o resolvedor Gurobi. Nesse caso é possível obter o dicionário do problema diretamente da biblioteca, assim evita-se desperdiçar tempo de execução com o cálculo da matriz inversa. Os novos resultados são apresentados na Tabela 6.

Instância	CP	BB	BC
3 Barras, k = 0	0.0033	0.0045	0.0090
3 Barras, k = 1	0.0058	0.0093	0.1059
24 Barras reduzido, k = 0	0.0046	0.0511	0.0494
24 Barras reduzido, k = 1	12.3156*	1.9527	81.1888
24 Barras, k = 0	0.0073*	0.0811	0.2912
24 Barras, k = 1	11.6186*	38.9537	121.8370

Tabela 6: Tempo de execução para cada algoritmo em Julia(Gurobi) em segundos.

Comparando a coluna BB das Tabelas 5 e 6 podemos ver que somente a troca de linguagem e resolver já diminuiu significativamente o tempo de execução, em média a nova implementação ficou 80% mais rápida. Também pode-se verificar nas colunas CP e BC de ambas as tabelas que de obter o dicionário diretamente do resolvedor melhora a performance significativamente, em alguns casos a execução ficou quase 10 vezes mais rápida.

Conclusões

O problema de despacho de usinas a curto-prazo é de grande importância para o planejamento de sistemas elétricos. Sua importância torna crucial que critérios de segurança sejam adicionados, permitindo que as demandas do sistema sejam atendidas ainda que falhas conjunturais ocorram. No critério de segurança n-K, até K elementos do sistema podem falhar, sem que a demanda seja comprometida.

Nesse trabalho apresentamos o problema de despacho com critério de segurança n-K como um problema de Programação Linear Inteira Mista. Adicionalmente, implementamos algoritmos de *Branch and Bound*, *Cutting Planes* e *Branch and Cut* e os aplicamos a esse problema. Para que pudéssemos comprar as diferentes implementações utilizamos três instâncias do problema de despacho à curto prazo.

O algoritmo *Branch and Cut* teve um tempo maior que o *Branch and Bound* mesmo com a criação de menos elementos. Por fim, notamos que nossas implementações *Branch and Cut* e *Branch and Bound* foram eficientes para resolver instâncias do problema de despacho de usinas a curto-prazo com contingência.

*Não chegou no ótimo

https://proceedings.science/p/85385?lang=pt-br

Referências

Cook, W., Dash, S., Fukasawa, R., and Goycoolea, M. (2009). Numerically safe gomory mixed-integer cuts. *INFORMS Journal on Computing*, 21(4):641–649.

Goycoolea, M. G. (2006). *Cutting planes for large mixed integer programming models*. PhD thesis, Georgia Institute of Technology.

Grigg, C., Wong, P., Albrecht, P., Allan, R., Bhavaraju, M., Billinton, R., Chen, Q., Fong, C., Haddad, S., Kuruganty, S., et al. (1999). The ieee reliability test system-1996. a report prepared by the reliability test system task force of the application of probability methods subcommittee. *IEEE Transactions on power systems*, 14(3):1010–1020.

Street, A., Moreira, A., and Arroyo, J. M. (2014). Energy and reserve scheduling under a joint generation and transmission security criterion: An adjustable robust optimization approach. *IEEE transactions on Power Systems*, 29(1):3–14.

Street, A., Oliveira, F., and Arroyo, J. M. (2011). Contingency-constrained unit commitment with $n - k$ security criterion: A robust optimization approach. *IEEE Transactions on Power Systems*, 26(3):1581–1590.

Wolsey, L. A. (1998). *Integer programming*. Wiley-Interscience, New York, NY, USA.

https://proceedings.science/p/85385?lang=pt-br