5

Heurística de Melhoria via Geração de Colunas (HMGC)

Neste capítulo apresentamos a Heurística de Melhoria via Geração de Colunas (HMGC). A HMGC se baseia na geração atrasada de colunas e em heurísticas de busca local, que foram apresentadas no capítulo 3. O principal componente da HMGC é a transformação do espaço de busca utilizando as variáveis duais resultantes da resolução da geração de colunas. Esta transformação tem por objetivo permitir a busca local escapar de mínimos locais de forma a explorar o maior número de regiões de qualidade do espaço de busca. Além da apresentação da HMGC, mostramos uma aplicação da mesma para os problemas de códigos de cobertura, de forma a permitir a avaliação de sua qualidade através da comparação com a busca tabu reativa apresentada no capítulo 4.

5.1

Heurística HMGC

Numa busca local dizemos que chegou-se em um mínimo local quando na exploração da vizinhança não é possível melhorar a solução atual. Neste caso a busca fica presa nesta região de mínimo local, se as soluções desta região estiverem distantes da otimalidade então o resultado da busca será comprometido. A principal característica da HMGC é sua estratégia de escape destas regiões, de forma que seja possível para a busca explorar o maior número possível de regiões de qualidade, aumentando as chances da busca de encontrar soluções próximas da otimalidade. A estratégia de escape é baseada em uma transformação do espaço de busca a partir do problema de pricing da geração de colunas. A idéia da estratégia consiste em aplicar a transformação do espaço de busca e então reiniciar a busca neste novo espaço, na esperança de que nele a busca não esteja em um mínimo local e ao explorar o mesmo seja possível atingir soluções de melhor qualidade.

5.1.1

Transformação do Espaço de Busca

Na geração atrasada de colunas temos o chamado problema de pricing, quando as colunas da formulação são soluções viáveis do problema e o problema de pricing é o próprio problema original, porém com custos diferentes dependentes dos valores das variáveis duais, podemos considerar o espaço de busca do problema de pricing como uma transformação do espaço de busca do problema original. A transformação utilizada na HMGC é justamente esta, utiliza-se uma formulação do problema onde as colunas sejam soluções viáveis e o problema de pricing associado seja o próprio problema original, tendo apenas custos diferentes. Para exemplificar a transformação mostraremos o caso do problema clássico de otimização chamado caixeiro viajante.

O problema do caixeiro viajante consiste em um conjunto de cidades e distâncias entre as mesmas, onde o objetivo é visitar cada cidade exatamente uma vez retornando a cidade inicial tendo percorrido a menor distância possível. O problema do caixeiro viajante equivale a encontrar o ciclo hamiltoniano (que são ciclos em grafos em que cada nó do mesmo aparece uma vez) de menor custo em um grafo $G(V, A, C)$, sendo V o conjunto de vértices do grafo (um por cidade), A o conjunto de arestas, e C uma função de custo que atribui a cada aresta $(i, j) \in A$ um custo $C(i, j)$ correspondendo a distância da cidade i para j. Seja P o conjunto de todos os tours (ciclos hamiltonianos) possíveis, p um elemento desse conjunto, $l(p)$ o tamanho do tour p, q_{ij}^p uma constante que será 1 se no caminho p a aresta (i, j) é usada, ou seja, se em algum momento do caminho o viajante vai da cidade i para j, λ_p uma variável de decisão que indica se o caminho p foi escolhido como solução, x_{ij} uma variável de decisão que indica se a aresta (i, j) está sendo usada na solução, então podemos escrever uma formulação para o caixeiro viajante com as características necessárias à HMGC:

$$\min \sum_{p \in P} l(p)\lambda_p \tag{5-1}$$

sujeito a

$$\sum_{p \in P} \lambda_p = 1 \tag{5-2}$$

$$\sum_{p \in P} q_{ij}^{p} \lambda_p - x_{ij} = 0, i, j = 1, ..., n \tag{5-3}$$

$$\lambda_p \geq 0, p \in P \tag{5-4}$$

$$x_{ij} \geq 0, i, j = 1, ..., n \tag{5-5}$$

Vamos analisar o problema de pricing associado a esta formulação. O objetivo da resolução do problema de *pricing* é a obtenção de colunas de custo reduzido negativo. No caso da formulação em questão consideramos as colunas como tours no grafo G, assim devemos encontrar tours cuja variável λ_p associada tenha custo reduzido negativo. O custo reduzido de uma variável é dado pela seguinte equação, onde c_v representa o custo da variável, α representa o valor das variáveis duais associadas a cada restrição e a_v corresponde ao vetor de coeficientes da variável v em cada restrição, [7]:

$$\pi_v = c_v - \alpha a_v \tag{5-6}$$

O custo reduzido de uma coluna λ_p é dado então pelo custo do tour p subtraído do somatório do valor das variáveis duais associadas a cada restrição do tipo 5-3 em que p tem coeficiente não-nulo, o que corresponde as restrições sobre as arestas (i, j) de p. Dessa forma o problema de *pricing* passa a ser encontrar um tour no grafo $G' = (V, A, C')$, onde C' é a função que atribui um custo a cada aresta (i, j) de G', sendo a mesma definida da seguinte forma, onde α_{ij} é o valor da variável dual associada a restrição do tipo 5-3 da aresta (i, j):

$$C'(i, j) = C(i, j) - \alpha_{ij} \tag{5-7}$$

Assim o problema de *pricing* é o próprio caixeiro viajante com custos sobre as arestas de A alterados pelo valor das variáveis duais. Com isto, podemos aplicar a busca local da HMGC neste novo problema transformado para tentar escapar do mínimo local, ou seja, quando a busca local atingir um mínimo local aplicamos a transformação no espaço de busca e reiniciamos a mesma, na expectativa de que na busca por colunas de custo reduzido negativo possamos encontrar soluções melhores que a atual para o problema original, escapando portanto do mínimo local.

Um detalhe importante a ser notado na transformação do espaço de busca é que podemos aplicar esta transformação sucessivas vezes, ou seja, se ao transformarmos o espaço e efetuarmos a busca nele não conseguirmos fugir do mínimo local podemos aplicar novamente a transformação só que desta vez em

cima do próprio espaço transformado, já que o espaço transformado é apenas uma alteração nos custos do problema original, e reiniciar novamente a busca na tentativa de escapar do mínimo local.

5.1.2

Algoritmos para a HMGC

Existem inúmeras formas de se combinar a busca local com a transformação do espaço de busca apresentada acima de forma a tentar fugir de mínimos locais na tentativa de melhorar a qualidade da busca. A HMGC é justamente esta combinação, ela não fixa a utilização da busca local nem da transformação do espaço de busca, sendo portanto infindáveis as possibilidades de aplicação da HMGC. Aqui são ilustradas duas formas de se utilizar a HMGC, apresentando seus respectivos algoritmos. A primeira forma funciona como uma espécie de algoritmo populacional para a HMGC, ou seja, se trabalha com uma população (conjunto) de soluções ao longo das iterações que evoluem de forma a melhorar sucessivamente sua qualidade. Já a segunda forma é baseada no conceito de transformações sucessivas no espaço de busca, conforme explicado na seção anterior, sendo mais simples que a primeira forma.

Algoritmo Populacional para a HMGC

Na primeira forma ilustrada o algoritmo da HMGC se inicia criando um conjunto Ini de soluções iniciais, em seguida aplica-se a busca local em cada solução de Ini criando-se o conjunto inicial de colunas Col a serem utilizadas na formulação da geração de colunas e guardando a melhor solução encontrada s pela busca. Com o programa linear montado a partir de Col resolve-se o mesmo, e a partir das variáveis duais da solução obtemos o problema de pricing, que corresponde ao novo espaço de busca. Então, aplica-se a busca local neste novo espaço em um conjunto Sel de soluções que obedecem a um critério escolhido de seleção. Se alguma das soluções obtidas tiverem custo menor que s no espaço original então s é atualizada. Além disto, estamos interessados nas soluções obtidas com custo reduzido negativo. Seja $Cneg$ o conjunto destas soluções, então atualizamos Col adicionando as soluções de $Cneg$ no mesmo e removendo colunas em Col que obedeçam a algum critério escolhido. Caso $Cneg$ seja vazio, então a busca se encerra retornando s, caso contrário monta-se um novo programa linear com o Col e reinicia o processo de escape de mínimos locais. Abaixo mostramos o pseudo-código do algoritmo da HMGC:

Inicializações.

- Construir um conjunto inicial Ini de soluções.
- Para toda solução x em Ini aplicar busca local gerando o conjunto inicial de colunas Col.
- $s \leftarrow [x \in Col | C(x) = min_{x' \in Col}(C(x'))]$.

Loop Principal. Repita até que o critério de parada seja satisfeito.

- Criar programa linear $Prog$ utilizando as colunas em Col.
- Resolver $Prog$ de forma a criar o problema de pricing (novo espaço de busca).
- Selecionar conjunto de colunas Sel de Col para aplicar a busca local no problema de pricing.
- Para toda solução em Sel aplicar busca local no novo espaço, gerando o conjunto P de novas soluções.
- **Se** $s' \in P$ e $C(s') < C(s)$
    - $s \leftarrow s'$.
- $Cneg \leftarrow [x \in P | C'(x) < 0]$.
- **Se** $Cneg = \emptyset$
    - Retornar s.
- Remover de Col colunas satisfazendo a um critério escolhido.
- $Col \leftarrow Col \cup Cneg$.

Duas importantes características desta aplicação da HMGC são os critérios de aplicação da busca local e o gerenciamento das colunas ao longo da busca.

A aplicação da busca local nas soluções deve ser feita de forma a maximizar a qualidade das soluções gastando o menor tempo computacional possível. Entre algumas opções de critérios temos:

– Aplicar a busca em todas as colunas de Col.

– Aplicar a busca apenas nas n melhores soluções de Col.

– Aplicar a busca aleatoriamente em n soluções de Col.

– Aplicar a busca apenas em um percentual x de melhores soluções de Col.

– Aplicar a busca aletoriamente em um percentual x de soluções de Col.

Outro critério importante é a maneira em que as colunas são gerenciadas ao longo da busca, de forma que o programa linear resultante não se torne muito grande nem perca colunas que contribuam positivamente para o resultado da busca. Entre opções de gerenciamento de colunas temos:

– Manter apenas as colunas que estiveram na base nas últimas n iterações.

– Manter todas as colunas diferentes obtidas durante a busca.

– Manter apenas as n melhores colunas a cada iteração.

– Sempre tornar Col igual a P.

Algoritmo de Sucessivas Transformações para a HMGC

A segunda forma de aplicação da HMGC ilustrada se baseia no conceito de aplicação de sucessivas transformações no espaço de busca, conforme descrito na seção anterior. A vantagem desta forma sobre a anterior é sua simplicidade, onde é necessário definir apenas o critério de seleção das colunas para sua utilização na transformação do espaço de busca. O algoritmo desta segunda forma de aplicação da HMGC é auto-explicativo, sendo o mesmo apresentado logo abaixo:

Inicializações.

- Construir uma solução inicial Ini.
- Aplicar busca local em Ini coletando durante a busca um conjunto de soluções (colunas) Col que obedeçam ao critério de seleção.
- Atribuir para s o resultado da busca local.

Loop Principal.

- Aplicar transformação no espaço de busca atual utilizando as colunas em Col.
- Aplicar busca local em s no novo espaço de busca coletando durante a busca o conjunto de soluções (colunas) Col que obedeçam ao critério de seleção.

- Se durante a busca for encontrada uma solução melhor que s (com relação a função objetivo do espaço original), então atualizar s para esta solução.
- **Se** critério de parada satisfeito
    - Retornar s.

Esta segunda forma também se mostra mais apropriada para a prova de conceito da HMGC, pois o principal componente dela é a transformação do espaço de busca, o que motivou a escolha da mesma para a aplicação nos problemas de códigos de cobertura, além de sua simplicidade de implementação.

5.2
HMGC aplicada a Problemas de Códigos de Cobertura

Nesta seção apresentamos uma aplicação da HMGC para os problemas de códigos de cobertura. Como dito anteriormente a HMGC proposta utiliza como busca local a busca tabu reativa apresentada no capítulo 4. Com isto, o espaço de busca também corresponde ao problema de conjuntos dominantes em grafos, como descrito em 2.3.2. Além disto, a HMGC usa o algoritmo de sucessivas transformações apresentado em 5.1.2, devido a sua simplicidade e por ser mais adequado para prova de conceito da HMGC.

Apresentamos inicialmente a modelagem utilizada para o problema de geração de colunas, por conta desta modelagem mostramos uma forma de evitar o uso de resolvedores lineares na geração de colunas para a transformação do espaço, o que torna a HMGC proposta mais eficiente. Em seguida apresentamos o critério de seleção de colunas durante a busca para aplicação na transformação do espaço. Este é o único critério necessário para definição quando utilizamos o algoritmo descrito em 5.1.2.

5.2.1
Modelagem para Geração de Colunas

A formulação utilizada é semelhante à formulação do caixeiro viajante descrita em 5.1.1 para ilustrar a aplicação da HMGC. Pequenas modificações foram feitas para adaptá-la ao contexto dos conjuntos dominantes. A seguir descrevemos esta formulação.

Seja $G = (V, E)$ o grafo obtido a partir de uma instância dos problemas de códigos de cobertura pela construção descrita em 2.3.2, P o conjunto de todos

os conjuntos dominantes possíveis de G, p um elemento desse conjunto, $c(p)$ o número de vértices de p (custo de p), q_i^p uma constante que será 1 se o vértice i pertence ao conjunto dominante p, λ_p uma variável de decisão que indica se o conjunto dominante p foi escolhido como solução, x_i uma variável de decisão que indica se o vértice i pertence a solução, então podemos escrever uma formulação para o problema de conjuntos dominantes com as características necessárias à HMGC:

$$\min \sum_{p \in P} c(p)\lambda_p \tag{5-8}$$

sujeito a

$$\sum_{p \in P} \lambda_p = 1 \tag{5-9}$$

$$\sum_{p \in P} q_i^p \lambda_p - x_i = 0, i = 1, ..., n \tag{5-10}$$

$$\lambda_p \geq 0, p \in P \tag{5-11}$$

$$x_i \geq 0, i = 1, ..., n \tag{5-12}$$

Vamos analisar o problema de pricing associado a esta formulação. Como explicado anteriormente o objetivo da resolução do problema de *pricing* é a obtenção de colunas de custo reduzido negativo. As colunas neste caso são os conjuntos dominantes. O custo reduzido de uma coluna λ_p é dado pelo custo do conjunto dominante p subtraído do somatório do valor das variáveis duais associadas a cada restrição do tipo 5-10 em que p tem coeficiente não-nulo, o que corresponde as restrições sobre os vértices i que fazem parte de p. Podemos então decompor o custo reduzido de p por cada vértice que faz parte do mesmo, seja $cv'(i)$ este custo decomposto por vértice i, $c'(p)$ o custo reduzido de p e $d(i)$ o valor da variável dual associada a restrição do tipo 5-10 do vértice i, então:

$$c'(p) = \sum_{i \in p} cv'(i) \tag{5-13}$$

$$cv'(i) = 1 - d(i) \tag{5-14}$$

Assim o problema de pricing corresponde a encontrar o conjunto dominante p em G com menor somatório de custos por vértice i de p, sendo o custo

por vértice i igual a $cv'(i)$. Isto equivale ao problema de conjuntos dominantes onde ao invés de termos um de custo por vértice i temos $cv'(i)$.

Como no problema de pricing precisamos considerar custos diferentes por vértice, a busca tabu reativa usada na HMGC foi adaptada de modo a considerar o custo por solução como o somatório de custos por vértices da solução, sendo este custo por vértice igual a um no caso do problema original e $cv'(i)$ no caso do problema de pricing.

5.2.2

Evitando o Uso de Resolvedores Lineares

Analisando a transformação do espaço de busca da HMGC percebemos a necessidade de obtenção de valores ótimos para as variáveis duais de forma a se criar o problema de pricing. A obtenção destes valores pode ser feita pela resolução do problema primal de geração de colunas utilizando resolvedores de programas lineares. Dependendo da complexidade e do tamanho do problema a ser resolvido pelo resolvedor linear, o tempo computacional gasto pode ser elevado. Uma maneira de tentar evitar o uso dos resolvedores é analisar diretamente o problema dual ao problema primal da geração de colunas e tentar desenvolver um algoritmo para resolver diretamente este problema, de forma a obter os valores ótimos das variáveis duais. É justamente isto que fazemos para a HMGC aplicada aos códigos de cobertura.

Vamos analisar o problema dual ao problema primal da geração de colunas apresentado na seção anterior, seja L a variável dual associada a restrição 5-9 do problema primal e sejam os demais símbolos os mesmos apresentados na seção anterior, temos o problema dual da seguinte forma:

$$\max \ L \tag{5-15}$$

sujeito a

$$\sum_{i \in p} d(i) + L \leq c(p), p \in P \tag{5-16}$$

$$d(i) \geq 0, i = 1, ..., n \tag{5-17}$$

Como temos que $d(i) \geq 0$ então $\sum_{i \in p} d(i) \geq 0$, assim podemos afirmar

também que $\sum_{i \in p} d(i) + L \geq L$, então para satisfazermos a restrição do tipo 5-16 temos que ter necessariamente:

$$L \leq c(p), p \in P \tag{5-18}$$

Como desejamos maximizar L então as restrições do tipo 5-18 forçam $L = min(p \in P | c(p))$, ou seja, L é igual ao custo da coluna de menor custo, o que é coerente com a programação linear, já que na otimalidade o valor da solução ótima do dual e do primal são iguais. Tendo determinado o valor de L basta então atribuirmos valores para as variáveis $d(i)$ de forma que as restrições do tipo 5-16 não sejam violadas, já que a atribuição de valores a $d(i)$ não influencia a função objetivo. Esta atribuição pode ser feita de infinitas formas, o que caracteriza a existência de infinitas soluções ótimas para o problema dual. Uma atribuição trivial é $d(i) \leftarrow 0 (i = 1, \ldots, n)$, porém esta solução não serve a nossos propósitos, já que para ela o problema de pricing é exatamente igual ao problema original.

A seguir apresentamos o algoritmo utilizado para atribuição de valores às variáveis $d(i)$ que são utilizadas na transformação do espaço de busca da HMGC aplicada aos códigos de cobertura. Para que as restrições do tipo 5-16 não sejam violadas devemos garantir que $\sum_{i \in p} d(i) \leq c(p) - L$. A diferença $c(p) - L$ na restrição associada a coluna p representa o quanto podemos atribuir para as variáveis $d(i)$ dos vértices i de p. No algoritmo a seguir guardamos para cada coluna p o valor $s(p)$ que representa justamente o quanto se pode adicionar as variáveis duais dos vértices de p. A cada iteração determina-se a solução min com menor valor $s(p)$, então se seleciona aleatoriamente um vértice i de min para o qual ainda se possa atribuir algum valor, no algoritmo equivale a não pertencer ao conjunto $elim$. Com isto, se atribui $s(min)$ para $d(i)$, o que implica na atualização de todos os valores $s(p)$ de colunas que contenham i, nesta atualização se garante que $s(p) \geq 0$ pois $s(min) \leq s(p)$, evitando a violação das restrições 5-16. Seja opt a solução (coluna) de menor custo, podemos descrever o algoritmo da seguinte forma:

Inicializações.

- $elim \leftarrow [i = 1, \ldots, n | i \in opt]$.
- $s(p) \leftarrow c(p) - L$, para todo $p \in P$.
- $d(i) \leftarrow 0$, para $i = 1, \ldots, n$.
- $A \leftarrow [p \in P | p \neq opt]$.

Loop Principal. Enquanto $A \neq \emptyset$.

- $min \leftarrow$ solução p de A com menor valor $s(p)$.
- Selecionar aleatoriamente um vértice $i \in min$ onde $i \notin elim$.
- Adicionar a $elim$ os vértices de min.
- $d(i) \leftarrow s(min)$.
- Para todo $p \in A$ onde p possui o vértice i
  - $s(p) \leftarrow s(p) - s(min)$.
- Remover min de A.

A atribuição de valores às variáveis duais resultante da execução deste algoritmo resulta em um problema de pricing com espaço de busca diferente do problema original, o que é justamente a finalidade da utilização da geração de colunas na HMGC, sendo portanto apropriada para nossos propósitos.

5.2.3

Critério de Seleção de Colunas

Como a HGMC aplicada aos códigos de cobertura utiliza o algoritmo de sucessivas transformações do espaço de busca descrito em 5.1.2, precisamos definir o critério de seleção de colunas durante a busca local, de forma a serem utilizadas na geração de colunas.

O critério adotado trabalha montando um conjunto de soluções Col ao longo da busca. A cada iteração da busca tabu reativa se verifica se a solução atual s é viável, ou seja, se é um conjunto dominante. Caso s seja viável é calculado o grau de similaridade de s com relação as soluções p pertencentes a Col. O grau de similaridade $G(s, p)$ é calculado da seguinte forma, onde S e P representam os conjuntos de vértices de s e p respectivamente e E representa o conjunto formado pelos vértices pertencentes tanto a s quanto a p:

$$G(s) = \frac{|E|}{|S \cup P|} \tag{5-19}$$

Se existir algum $p \in Col$ para qual o grau de similaridade $G(s, p)$ seja maior que um determinado limiar t, então s não é inserido em Col, caso contrário s é incluído em Col. A idéia é tentar manter em Col soluções representativas das regiões do espaço explorado pela busca, evitando o armazenamento

de soluções similares. O valor adotado para t foi de 0.6, ele foi determinado de forma empírica avaliando o comportamento da busca ao se utilizar um conjunto diferente de valores para t.