5
Vizinhança elipsoidal

Como descrito anteriormente, na seção 2.1, as vizinhanças k-OPT são bastante comuns em algoritmos de busca local devido à sua simplicidade e sua aplicabilidade a vários problemas diferentes. O algoritmo *Local Branching* (descrito com detalhes na seção 4.2) foi o primeiro a aplicar a idéia de vizinhanças k-OPT em modelos MIP. O algoritmo 6, chamado KOPT, é uma simples implementação da busca local k-OPT em modelos MIP usando a definição de distância entre soluções descrita na seção 4.2.

Algoritmo 6 KOPT

Entrada: raio de busca k, solução base x_base
Saída: solução x, possivelmente melhor que x_base

```
1: adicionar a restrição Δ(x_base, x) ≤ k
2: x = resolver MIP (com limite superior igual ao custo de x_base)
3: if encontrada solução viável ou ótima then
4:    return x
5: else
6:    return x_base
7: end if
```

A vizinhança k-OPT de uma solução x pode ser vista então como todas as soluções que estão em uma distância menor ou igual a k de x. Dessa maneira, é fácil visualizar a busca local em uma vizinhança k-OPT como sendo a busca por soluções em um espaço "circular" em torno da solução x.

O trabalho de Pigatti et al. [30] propõe uma modificação simples da vizinhança k-OPT onde, ao invés de centrar a busca em uma única solução x, centra-se a busca em torno de duas soluções a e b. A vizinhança proposta pode então ser definida pelo seguinte corte:

$$\sum_{j \in N_1(a) \cap N_1(b)} 2x_j + \sum_{j \in ((N_1(a) \cup N_1(b)) \setminus (N_1(a) \cap N_1(b)))} x_j \geq N + |N_1(a) \cap N_1(b)| - k \quad (5\text{-}1)$$

onde:

– a e b são as duas soluções-base da vizinhança;

- $N_1(a)$ e $N_1(b)$ são o conjunto de variáveis binárias com valor 1 em a e b, respectivamente;
- k é o parâmetro de folga da vizinhança, semelhante ao parâmetro k na vizinhança k-OPT, determinando quão distante as soluções podem estar de a e b.

A partir da inequação (5-1) percebe-se que esta vizinhança dá mais importância (peso no lado esquerdo do corte) às variáveis que estão presentes em ambas as soluções. A vizinhança formada pelo corte (5-1) é chamada de vizinhança elipsoidal [30] devido a seu forte paralelo com o conceito matemático de elipse.

Uma elipse é o conjunto de todos os pontos no plano cuja soma das distâncias r_1 e r_2 para dois pontos fixos F_1 e F_2 (chamados de focos da elipse) é um valor constante e positivo a [37]. Supondo que as soluções do modelo MIP são pontos em um espaço binário multi-dimensional que sempre possuem N bits iguais a 1, a inequação (5-2) define uma vizinhança elipsoidal em torno de duas soluções a e b com folga k.

$$\begin{aligned} \Delta(x,a) + \Delta(x,b) &\leq k \\ \sum_{j \in N_1(a)} (1 - x_j) + \sum_{j \in N_1(b)} (1 - x_j) &\leq k \quad \text{por (4-2)} \\ (N - \sum_{j \in N_1(a)} x_j) + (N - \sum_{j \in N_1(b)} x_j) &\leq k \\ 2N - (\sum_{j \in N_1(a)} x_j + \sum_{j \in N_1(b)} x_j) &\leq k \\ \sum_{j \in N_1(a)} x_j + \sum_{j \in N_1(b)} x_j &\geq 2N - k \end{aligned} \tag{5-2}$$

O corte (5-2) é equivalente ao corte (5-1): as variáveis binárias que têm valor 1 em ambas as soluções possuem peso dois no lado esquerdo da inequação, enquanto as variáveis que têm valor 1 em somente uma solução possuem peso um. O lado esquerdo calcula quantas variáveis em 1 a solução a ser encontrada possui em comum com as soluções-base (ou seja, quão semelhante elas são) e o corte exige que a solução a ser encontrada seja distante das soluções-base. A organização da inequação, entretanto, deixa claro como esta vizinhança se relaciona com uma elipse: a vizinhança delimitada pelo corte é formada por todas as soluções cuja soma das distâncias até as soluções-base a e b (os focos da elipse) são, no máximo, um valor fixo k.

5.1 Múltiplas soluções

A partir do corte (5-2) pode-se naturalmente extender o conceito de vizinhança elipsoidal para múltiplas soluções. A inequação (5-3) define um corte elipsoidal a partir de um conjunto de soluções-base $S = \{s_1, s_2, \ldots, s_p\}$.

$$\begin{aligned} \sum_{s \in S} \Delta(x, s) &\leq k \\ \sum_{s \in S} \sum_{j \in N_1(s)} (1 - x_j) &\leq k \quad \text{por (4-2)} \\ \sum_{s \in S} (N - \sum_{j \in N_1(s)} x_j) &\leq k \\ |S|N - \sum_{s \in S} \sum_{j \in N_1(s)} x_j &\leq k \\ \sum_{s \in S} \sum_{j \in N_1(s)} x_j &\geq |S|N - k \end{aligned} \tag{5-3}$$

Para exemplificar as definições anteriores e deixar claro o conceito de vizinhanças elipsoidais, tome o exemplo a seguir: calcular as soluções que fazem parte da vizinhança elipsoidal de duas soluções a e b, com $N = 3$ - em outras palavras, encontrar as soluções que se encontram na elipsoide entre a e b e que possuem $N = 3$ bits de valor 1. Abaixo, x_i significa o conjunto de soluções que encontra-se na vizinhança elipsoidal de a, b a uma folga i - x_3 por exemplo mostra o conjunto de soluções cuja distância até a e b, somada, é igual a 3. Isso significa que, ao colocar as soluções de x_3 no lado esquerdo da inequação

5-3, o resultado será 3.

$$
\begin{aligned}
a &= 001011 \\
b &= 101010 \\
x_0 &= \{\} \\
x_1 &= \{a, b\} \quad (\sum_{s \in S} \sum_{j \in N_1(s)} x_j = 5,\ k = 1) \\
x_2 &= \{101001, 100011\} \quad (\sum_{s \in S} \sum_{j \in N_1(s)} x_j = 4,\ k = 2) \\
x_3 &= \{101100, 111000, 001101, 011001, \\
&\qquad 100110, 110010, 000111, 010011\} \quad (\sum_{s \in S} \sum_{j \in N_1(s)} x_j = 3,\ k = 3) \\
x_4 &= \{100101, 110001, 010110, 011100\} \quad (\sum_{s \in S} \sum_{j \in N_1(s)} x_j = 2,\ k = 4) \\
x_5 &= \{110100, 010101\} \quad (\sum_{s \in S} \sum_{j \in N_1(s)} x_j = 1,\ k = 5) \\
x_6 &= \{\}
\end{aligned}
$$

O exemplo acima ajuda a ilustrar outro detalhe importante da vizinhança elipsoidal: para um conjunto de soluções-base S, nem todos os valores de k são viáveis. No exemplo acima, não existe solução viável para $k = 1$, simplesmente porque não existe vetor binário que possua $N = 3$ bits iguais a 1 e cuja soma das "interseções" com as soluções-base a e b (lado esquerdo do corte) seja igual a 1. Pode-se perceber então que, para uma vizinhança definida por múltiplas soluções, existem limites mínimos e máximos para o valor de k. O algoritmo 7, chamado de EllipsoidalBounds, exemplifica como pode ser feito o cálculo dos limites mínimos e máximos da vizinhança elipsoidal, dado um conjunto de soluções S. Este algoritmo basicamente calcula o maior e menor valor possível para o lado esquerdo do corte (5-3) e, aplicando este valor na inequação, calcula os valores, respectivamente, mínimo e máximo de k.

Finalmente, o algoritmo 8, chamado EllipsoidalSearch, define uma busca local na vizinhança elipsoidal. Este algoritmo utiliza o mesmo conceito de busca em "disco" utilizada pelo algoritmo VNS-Disk (algoritmo 5), fazendo também uma espécie de busca em vizinhança variável como no algoritmo VND (algoritmo 4).

Na linha 1, calculam-se os limites para os valores da folga k. Na linha 2, calcula-se o limite superior a ser utilizado na busca - para manter um balanço entre intensificação e diversificação, usa-se a média aritmética dos custos das soluções em S. As linhas 4 e 5 inicializam o valor das variáveis k e

Algoritmo 7 EllipsoidalBounds

Entrada: conjunto de soluções-base S, número N de variáveis de valor 1
Saída: limite mínimo min e máximo max para os valores de k em uma vizinhança elipsoidal

1: $possible\_lhs = \{\}$
2: **for** cada variável x_j **do**
3: $num\_ones = 0$
4: **for** cada solução $s \in S$ **do**
5: **if** x_j tem valor 1 em s **then**
6: $num\_ones = num\_ones + 1$
7: **end if**
8: **end for**
9: $possible\_lhs = possible\_lhs \cup num\_ones$
10: **end for**
11: **return** $min = |S|N-$ soma dos N maiores valores de $possible\_lhs$
12: **return** $max = |S|N-$ soma dos N menores valores de $possible\_lhs$

Algoritmo 8 EllipsoidalSearch

Entrada: conjunto de soluções-base S, número de variáveis em 1 N, tamanho vizinhança opt, $max\_failures$
Saída: x_best, solução ótimo local na vizinhança elipsoidal, ou erro

1: $min, max =$ EllipsoidalBounds(S, N)
2: $UB =$ média dos custos das soluções em S
3: $num\_failures = 0$
4: $k = min$
5: **while** $num\_failures < max\_failures$ e $rhs + opt \leq max$ **do**
6: adic. restrição: $N|S| - (k + opt) \leq \sum_{s \in S} \sum_{j \in N_1(s)} x_j \leq N|S| - k$
7: $x =$ resolver MIP (com limite superior $= UB$)
8: **if** encontrada solução viável ou ótima **then**
9: $k = min$
10: $x\_best = x$
11: **else**
12: $k = k + opt$
13: $num\_failures = num\_failures + 1$
14: **end if**
15: remover restrição adicionada na linha 6
16: **end while**
17: **return** x_best, se encontrado, ou erro caso contrário

$num\_failures$: a primeira é a folga a ser utilizada na vizinhança elipsoidal, a segunda guarda o número de tentativas de busca que não tiveram sucesso. O laço principal do algoritmo é controlado pelo número de falhas (que não pode exceder o parâmetro $max\_failures$) e a folga a ser utilizada (que não pode exceder o valor de max). A linha 6 adiciona o corte que define a vizinhança elipsoidal, enquanto as linhas 7 a 15 lidam com a otimização do modelo e com o resultado: caso encontre-se uma solução viável, guarda-se a solução e o valor de k volta a min; caso contrário, aumenta-se k e incrementa-se o contador de falhas. No último passo do laço, remove-se a restrição adicionada na linha 6. Ao final do laço, na linha 17, retorna-se a melhor solução encontrada ou um valor de erro, caso nenhuma solução viável tenha sido achada.

5.2 Paralelo com Relinking

O paralelo entre a vizinhança elipsoidal e o passo de *relinking* do algoritmo PR (seção 2.2.2) é claro: uma busca em uma vizinhança elipsoidal (como a do algoritmo EllipsoidalSearch) parte de um conjunto de soluções-base e gera soluções que são semelhantes às deste conjunto. Uma diferença é que, normalmente, o passo de *relink* gera somente soluções que estão no "caminho" entre as soluções do conjunto-base. No uso da vizinhança elipsoidal descrito acima, soluções fora deste caminho também podem estar presentes na resposta, embora soluções que sejam semelhantes às do conjunto-base são consideradas mais próximas e, logo, são encontradas com valores de folga k menores.