3
Métodos de Otimização

3.1
Introdução

Os métodos de otimização são aplicados em problemas nos quais é preciso maximizar ou minimizar certa característica. Na maioria das aplicações, esta característica é representada por uma função objetivo, que depende de suas variáveis de decisão. A otimização é feita por meio das variações destas variáveis, tendo estas que cumprir certas restrições do problema.

Neste capítulo são apresentados alguns métodos de otimização global e local, tendo em consideração tanto suas características principais como também suas vantagens e desvantagens em relação a outros métodos de otimização.

3.2
Definições

Para o melhor entendimento de um problema de otimização, precisa-se de alguns conceitos prévios. Os problemas de otimização podem-se agrupar nos seguintes tipos [33]:

Problema sem restrições

Este tipo de problema não tem restrições e sua função objetivo pode chegar a ser descontínua ou inclusive infinita. A Equação (3-1) representa um típico problema de otimização sem restrições, na qual x é o vetor das variáveis de decisão e f é a função objetivo do problema.

$$\begin{array}{ll} Min & f(x) \\ sujeitoa: & x \in \mathbb{R}^n \end{array} \tag{3-1}$$

Problema com limites e restrições lineares

Este tipo de problema tem restrições lineares. O conjunto $\mathbb{X}$ define estas restrições por meio das matrizes A e b. A Equação (3-2) representa um típico

PUC-Rio - Certificação Digital Nº 1121430/CA

problema de otimização com restrições lineares, onde x é o vetor das variáveis de decisão e f é a função objetivo do problema.

$$\begin{aligned} Min \quad & f(x) \\ sujeitoa: \quad & x \in \mathbb{X} \\ \mathbb{X} = \{x \in \mathbb{R}^n : \; & Ax \leq b\} \cap [l, u] \end{aligned} \tag{3-2}$$

Problema com restrições gerais

Este tipo de problema tem restrições gerais, ou seja, suas variáveis de decisão além de cumprir restrições de limites, também tem que cumprir restrições definidas por funções não lineares $c_i(x)$. A Equação (3-3), representa um típico problema de otimização com restrições gerais, onde x é o vetor das variáveis de decisão, f é a função objetivo do problema e c_i são as restrições não lineares das variáveis de decisão. Estes tipos de otimização se encontram na maioria de aplicações de engenheira, estatística e fianças, já que consideram restrições adicionais das variáveis de decisão.

$$\begin{aligned} Min \quad & f(x) \\ sujeitoa: \quad & x \in \mathbb{X} \cap \Omega \\ \Omega = \{x : \; & c_i(x) \leq 0, \; i = 1, 2, ..., n\} \end{aligned} \tag{3-3}$$

3.3 Evolução Diferencial

A Otimização Global é aplicada amplamente em áreas como engenharia, estatística e finanças. Na maioria de problemas práticos as funções objetivos são não diferenciáveis, não-contínuas, não-lineares, possuem perturbações, multi-dimensional ou ter muitos mínimos locais ou restrições. Tais problemas são difíceis, se não mesmo impossíveis de resolver de forma analítica. O método de Evolução Diferencial (ED) [21] surge como uma ferramenta usada para encontrar soluções aproximadas para este tipo de problemas. Diferentes aplicações [34], [35] e [36] mostram a robustez e precisão do método.

3.3.1 Descrição do Método

Evolução Diferencial (ED) é um método de busca direta paralela que utiliza $NP(\geq 4)$ vetores de parâmetros D-dimensionais. Os vetores de parâmetros são da forma:

$$x_{i,G} = [x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, ..., x_{D,i,G}] \; i = 1, 2, 3, ..., NP \tag{3-4}$$

PUC-Rio - Certificação Digital Nº 1121430/CA

na qual G é a geração atual também conhecida como iteração. O número de populações NP não muda durante o processo de otimização. Como regra geral, supõe-se uma distribuição de probabilidade uniforme para todas as decisões aleatórias, salvo indicação contrária. Logo, o conjunto inicial de vetores é criado aleatoriamente, procurando que o espaço de busca seja totalmente coberto. Na ausência de um espaço de busca, utiliza-se uma distribuição uniforme para inicializar a população. Logo, ED cria novos vetores de parâmetros por meio da adição da diferença ponderada entre dois vetores de parâmetros a um terceiro indivíduo. Considera-se esta operação como uma mutação. Os vetores de parâmetros mutados são então combinados com outros vetores pré-determinados, denominados *target vectors*, a fim de criar os *trial vectors*. Esta combinação de parâmetros é referida como cruzamento em ED. É importante ressaltar que cada vetor presente na atual população deve ser usado uma vez como *trial vector*. Caso o *trial vector* forneça um valor de *fitness* maior (maximização) que aquele associado ao respectivo *target vector*, este último dará lugar ao primeiro na próxima geração. Esta operação corresponde à seleção.

Todos estes passos são resumidos na Figura 3.1. Nas seguintes subseções detalha-se cada uma das estratégias do método da Evolução Diferencial.

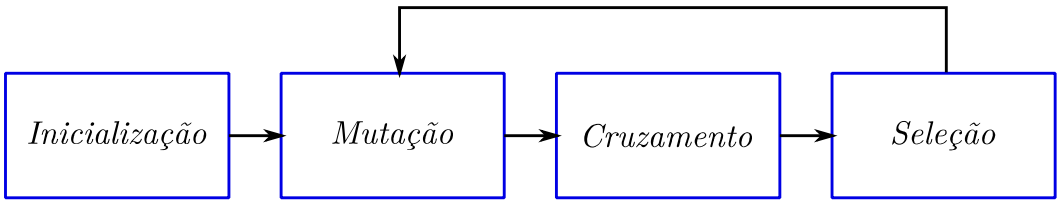

Figura 3.1: Processo geral de evolução da ED.

3.3.2 Inicialização

Antes de poder inicializar a população, devem ser definidos os limites inferior e superior para cada parâmetro (variável de decisão) do problema. Uma vez que os limites foram definidos, um gerador de números aleatórios atribui a cada parâmetro de cada vetor da população um valor que está dentro dos limites estabelecidos anteriormente. Por exemplo, para uma determinada geração g^*, o valor inicial do j-ésimo parâmetro do i-ésimo vetor da população é:

$$x_{j,i,g*} = rand(0,1)(UB_j - LB_j) + LB_j \tag{3-5}$$

PUC-Rio - Certificação Digital Nº 1121430/CA

na qual UB_j e LB_j são os limites superior e inferior do parâmetro j. O gerador de números aleatórios $rand(0,1)$, retorna um número aleatório distribuído uniformemente na faixa de $[0,1\rangle$. Além disso, mesmo que uma variável seja discreta ou integral, deve ser inicializada com um valor real, uma vez que a ED trata internamente todas as variáveis como valores de ponto flutuante, independentemente do seu tipo.

3.3.3 Mutação

Para cada *target vector* $x_{i,G},\ i = 1,2,3,...,NP$, um terceiro vetor (vetor mutante) é gerado por meio da seguinte equação:

$$v_{i,G+1} = x_{r_1,G} + F(x_{r_3,G} - x_{r_2,G}) \tag{3-6}$$

na qual $x_{r_1,G}$, $x_{r_2,G}$ e $x_{r_3,G} \in\ 1,2,3,...,NP$ são índices mutuamente diferentes e distintos do índice do *target vetor* i. F é uma constante real $\in\ [0,2]$ que determina o tamanho do passo na direção do vetor diferença $x_{r_3,G} - x_{r_2,G}$. A Figura 3.2 mostra como se constrói o vetor mutante $v_{i,G+1}$

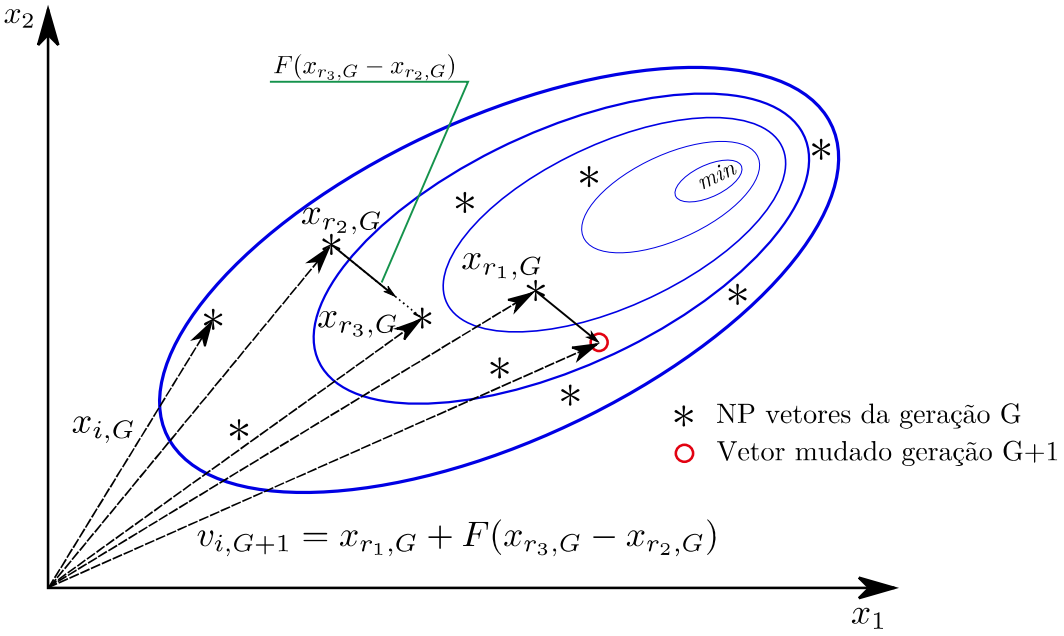

Figura 3.2: Exemplo bi-dimensional do processo de mutação.

PUC-Rio - Certificação Digital Nº 1121430/CA

3.3.4 Cruzamento

Com a finalidade de aumentar a diversidade dos novos vetores de parâmetros (mutação), o procedimento de cruzamento é aplicado. Seja $x_{i,G}$ o *target vector* da geração G e $v_{i,G+1}$ o vetor mudado obtido da Equação (3-6). Define-se um novo vetor $u_{i,G+1}$ D-dimensional denominado *trial vector*, que é determinado da seguinte maneira:

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1}, & \text{se } r_{j,i} \leq CR \text{ ou } j = I_i \\ x_{j,i,G}, & \text{se } r_{j,i} > CR \text{ ou } j \neq I_i \end{cases} \tag{3-7}$$

na qual $j = 1, 2, 3, ..., D$, $r_j = rand(0, 1)$, $CR \in [0, 1]$ é uma constante (probabilidade) definida pelo usuário e $I_i \in 1, 2, 3, ..., D$ é um índice aleatoriamente escolhido. Esta condição garante que $u_{i,G+1}$ recebe pelo menos uma componente de $v_{i,G+1}$. A Figura 3.3 esboça o processo de geração do *trial vector* $u_{i,G+1}$.

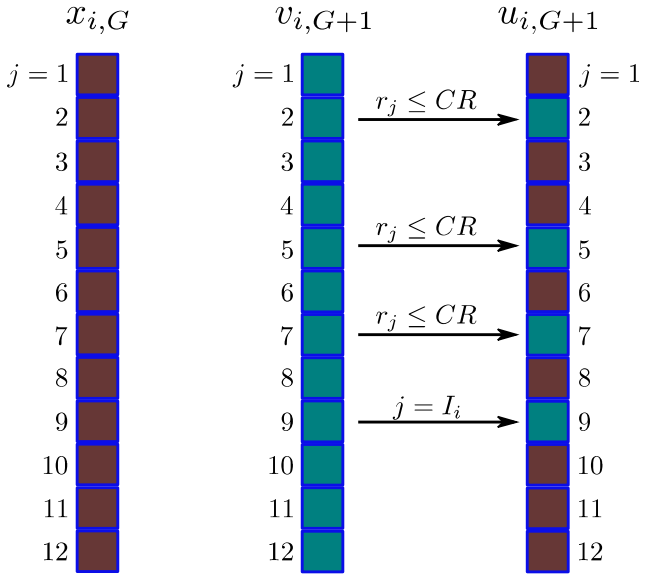

Figura 3.3: Processo de criação do *trial vector* $u_{i,G+1}$.

3.3.5 Seleção

Após das etapas de mutação e cruzamento, o *target vector* $x_{i,G}$ é comparado com o *trial vector* $u_{i,G+1}$ para determinar que vetores são preservados na próxima geração. Esta seleção é feita usando o critério do melhor *fitness*, e é determinado da seguinte maneira (problema de minimização):

PUC-Rio - Certificação Digital Nº 1121430/CA

$$x_{i,G+1} = \begin{cases} u_{i,G+1}, & \text{se } f(u_{i,G+1}) < f(x_{i,G}), \quad i = 1, 2, 3, ..., NP \\ x_{i,G}, & \text{caso contrario} \end{cases} \tag{3-8}$$

A mutação, cruzamento e seleção continuam até que seja alcançado algum critério de parada. O Algoritmo 1, a seguir mostra os passos da estratégia do método de Evolução Diferencial.

Algorithm 1 Evolução Diferencial

```
Function x = ED(NP, CR, F, range, f)
x ← f(x)
fitness_x ← random(range, NP)
while critério de parada não for satisfeito do
    for i = 1 até NP do
        v_{i,G+1} ← mutação(x_{i,G}, F)
        u_{i,G+1} ← crossover(x_{i,G}, v_{i,G+1}, CR)
    end for
    fitness_u ← f(u)
    for i = 1 até NP do
        if fitness_u(i) < fitness_x(i) then
            x_{i,G+1} ← u_{i,G+1}
        else
            x_{i,G+1} ← x_{i,G}
        end if
    end for
end while
```

PUC-Rio - Certificação Digital Nº 1121430/CA

3.4 Busca Generalizada de Padrões

O método de Busca Generalizada de Padrões (GPS, sua sigla em inglês) é uma classe de métodos de otimização chamados métodos *Pattern Search*, que foram propostos por Lewis e Torczon [37] como uma generalização de métodos anteriores, como o algoritmo *pattern search* original de Hook e Jeeves [38] e o algoritmo de busca multi-direcional de Dennis e Torczon [39]. O método GPS não requer informação do gradiente da função objetivo, então pode-se resolver problemas nos quais a função objetivo é não-diferenciável, estocástica ou até descontínua.

3.4.1 Descrição do Método

O algoritmo GPS é definido por meio de um conjunto finito de direções utilizadas em cada iteração para criar uma malha em torno do ponto atual,

sobre o qual os pontos de teste são selecionados, avaliados e comparados com a melhor solução atual. A Figura 3.4 mostra esta criação da malha. O princípio base na definição da malha é um conjunto de direções positivas $D \in \mathbb{R}^{n \times p}$ construído com as seguintes regras: cada direção $d_j \in D$ $(j = 1, 2, 3, ..., p)$ é o produto $G\bar{z}_j$ de uma matriz não singular $G \in \mathbb{R}^{n \times n}$ gerada por um vetor de números inteiros $\bar{z}_j \in \mathbb{Z}^n$. Observa-se que a matriz G (normalmente definida como uma matriz identidade) é usada para gerar todas as direções d_j. Estas condições são essenciais para a convergência do método.

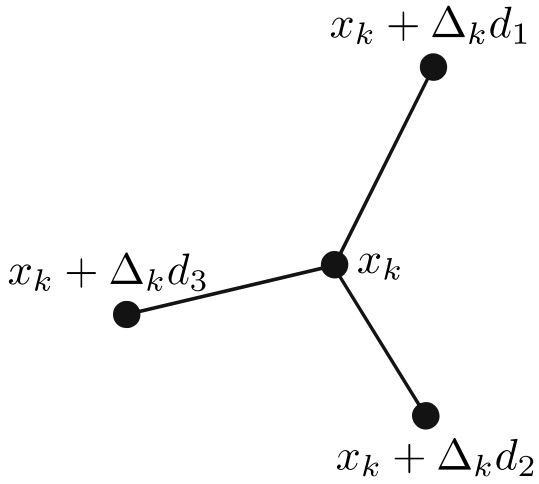

Figura 3.4: Malha ao redor da solução atual x_k.

Na iteração k, a malha M_k é centrada em torno do ponto atual $x_k \in \mathbb{R}^n$, com uma finura de malha $\Delta_k \in \mathbb{R}^n_+$. Esta malha pode ser expressa como:

$$M_k = \{x_k + \Delta_k D_z \ : \ z \in \mathbb{N}^p\} \tag{3-9}$$

Cada iteração é dividida em duas etapas: a primeira é o *Search* e a segunda é o *Poll*, que são definidas nas subseções seguintes.

3.4.2 Etapa Search

A função objetivo é avaliada em um conjunto finito de pontos de teste da malha com o intento de melhorar a função objetivo do problema. Se uma melhora da função objetivo for encontrada, esse ponto de prova torna-se o novo titular e a iteração é declarada bem sucedida. Em cada iteração do GPS, o algoritmo pode executar um ou dois passos. O primeiro é opcional e é chamado *Search*, que gera uma malha inicial de pontos que posteriormente são avaliados. Qualquer estratégia para a geração desta malha com um conjunto finito de pontos pode ser usada. Ele pode ser tão simples como nada alterar (ou seja, o

PUC-Rio - Certificação Digital Nº 1121430/CA

conjunto finito de pontos da malha é vazio), ou pode fazer uso de heurística, como algoritmos genéticos, por exemplo.

3.4.3 Etapa Poll

Se o passo *Search* é omitido ou não for bem sucedido na melhoria da função objetivo, uma segunda etapa, denominada etapa *Poll*, deve ser realizada antes de terminar a iteração. O passo *Poll* consiste de uma exploração local dos pontos de malha em torno da solução atual.

Quando o passo *Poll* é chamado, a função objetivo deve ser testada em certos pontos da malha, chamado de conjunto *Poll*, perto da solução atual $x_k \in \mathbb{R}^n$. Em cada iteração, uma matriz positiva de direções D_k composta por um subconjunto das colunas da matriz D, é usada para a construção do conjunto *Poll*. O conjunto *Poll* é composto pelos pontos da malha que se situam a um passo Δ_k da solução atual x_k ao longo da direção especificada por as colunas da matriz D_k:

$$P_k = \{x_k + \Delta_k d \ : \ d \in D_k\} \tag{3-10}$$

A Figura 3.5 mostra como o método GPS faz a busca na malha depois de determinar os conjuntos M_k e P_k.

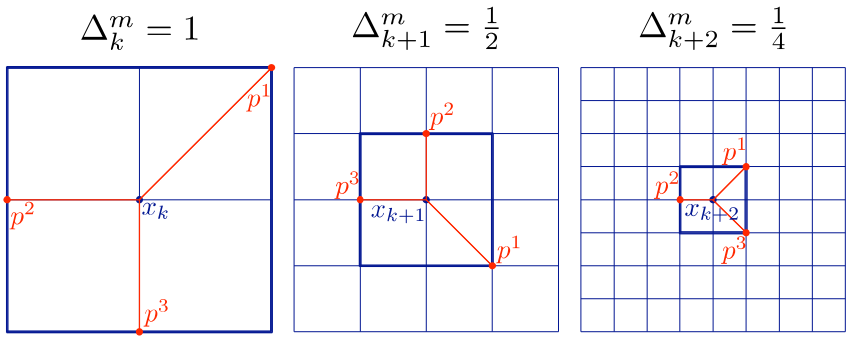

Figura 3.5: Exemplo das malhas no GPS: $P_k = \{x_k + \Delta_k d \ : \ d \in D_k\} = p_1, p_2, p_3$ para diferentes valores de Δ_k^m. Em todas a figuras M_k é a intersecção de todas a linhas.

O vetor de direções D_k pode ser uma matriz identidade I ou dependente do numero de iteração k e do valor atual da solução x_k: $D_k = D(k, x_k) \subseteq D$. O método GPS continua se não cumpre um critério de parada. Normalmente este critério é a finura de malha Δ_k. O Algoritmo 2 mostra os passos do método GPS.

PUC-Rio - Certificação Digital Nº 1121430/CA

Algorithm 2 Generalized Pattern Search (GPS)

1: **Inicialização**
2: x_0 tal que $f(x_0)$ seja finita, $\Delta_0 > 0$ e fazer $k = 0$
3: **Passos SEARCH e POLL para determinar** M_k eq. (3-9)
4: Realizar *Search* e, possivelmente, o *Poll* (ou apenas um das etapas, se um ponto de teste da malha melhore a função objetivo).
5: **SEARCH**: Avaliar f num subconjunto finito de pontos experimentais sobre a malha M_k (a estratégia que seleciona os pontos é fornecido pelo usuário).
6: **POLL**: Avaliar f num conjunto *Poll* P_k^p(ver eq. (3-10)) em torno de x_k definido por Δ_k e $D_k^p \subseteq D_k$, D_k depende de k e x_k.
7: **Atualização dos parâmetros**
8: Se os passos *Search* ou *Poll* produzem um melhor ponto, por exemplo, $x_{k+1} \in M_k$ para os quais $f(x_{k+1}) < f(x_k)$, logo atualizar $\Delta_{k+1} \geq \Delta_k$.
9: Caso contrário, x_k é um otimizador local de malha. Fazer $x_{k+1} = x_k$, diminuir $\Delta_{k+1} < \Delta_k$.
10: Fazer: $k \leftarrow k + 1$ e voltar para os passos *Search* e *Poll*.

3.5 Busca Direta com Malha Adaptativa

O método Busca Direta com Malha Adaptativa (MADS: suas siglas em inglês) [23] é uma extensão da Busca Generalizada de Padrões (GPS) descrito na secção 3.4. Estes dois métodos são algoritmos iterativos, nas quais a função objetivo do problema é avaliada num conjunto de pontos de prova pertencentes a uma malha, e estes pontos podem ser aceitos ou não para a iteração seguinte. Numa iteração k, a malha é definida na Equação (3-9).

Cada iteração do MADS está dividida em duas etapas, chamadas Search e *Poll*. A primeira etapa, o *Search*, é a mesma do GPS já foi descrito na subseção 3.4.2. A diferença entre o método MADS e o GPS reside exatamente na segunda etapa, o Poll, e será detalhada na subseção 3.5.1.

3.5.1 Etapa Poll

O método MADS introduz um novo parâmetro $\Delta_k^p \in \mathbb{R}_+$ em cada iteração k. Este novo parâmetro determina a distância dos pontos de prova gerados pela etapa *Poll* à solução atual x_k. O efeito deste novo parâmetro na malha pode ser observado na Figura 3.6 ($\Delta_k^p = n\sqrt{\Delta_k^m}$). Ao contrário do método GPS, esta distância é igual nas duas etapas: $\Delta_k = \Delta_k^p = \Delta_k^m$. No método MADS, a estratégia de atualização do Δ_k^p é que este seja sempre maior ou igual que o Δ_k^m para todo k. O conjunto gerado por esta etapa *Poll* do MADS é similar ao gerado pelo método GPS:

PUC-Rio - Certificação Digital Nº 1121430/CA

$$P_k = \{x_k + \Delta_k^m d \ : \ d \in D_k\} \subset M_k \tag{3-11}$$

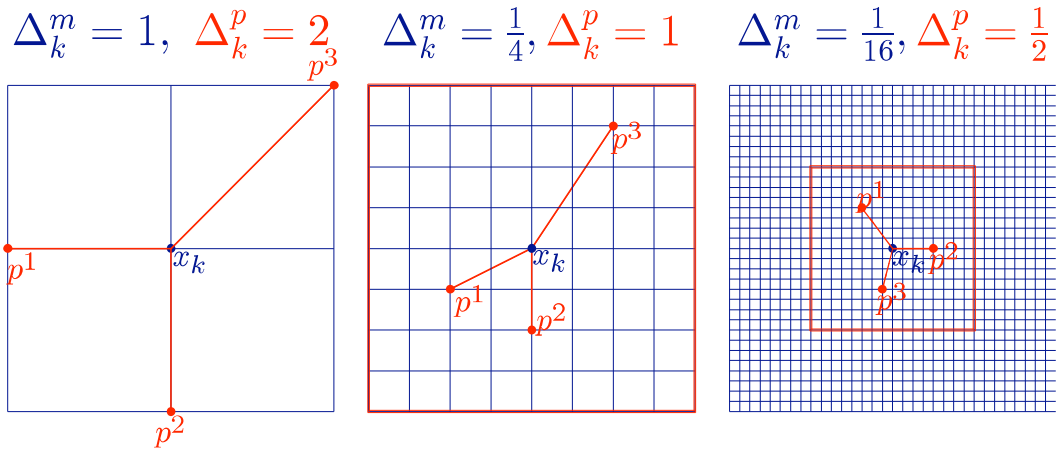

Figura 3.6: Exemplo das malhas no MADS: $P_k = \{x_k + \Delta_k d \ : \ d \in D_k\} = p_1, p_2, p_3$ para diferentes valores de Δ_k^m e Δ_k^p. Em todas a figuras M_k é a intersecção de todas a linhas.

Para definir os parâmetros do MADS utiliza-se uma implementação prática deste método, chamado LTMADS (Lower Triangle MADS). Estas regras práticas são resumidas a seguir:

1. Escolhe-se $\Delta_0^m = 0$ e $\Delta_0^p = 0$ como tamanho inicial da malha e da etapa *Poll*, respectivamente.

2. O vetor direção é definido como: $D = [I - I]$, onde I é a matriz identidade.

3. Define-se as seguintes regras de atualização:

$$\Delta_{k+1}^m = \begin{cases} \frac{\Delta_k^m}{4} & \text{se } x_k \text{ segue sendo o mínimo} \\ 4\Delta_k^m & \text{se um melhor ponto foi encontrado, e se } \Delta_k^m \leq \frac{1}{4} \\ \Delta_k^m & \text{caso contrário} \end{cases}$$

4. Faz-se $\Delta_k^p = n\sqrt{\Delta_k^m}$.

O Algoritmo 3 mostra os passos do método Mesh Adaptive Direct Search (MADS).

PUC-Rio - Certificação Digital Nº 1121430/CA

Algorithm 3 Mesh Adaptive Direct Search (MADS)

1: **Inicialização**
2: x_0 tal que $f(x_0)$ seja finita ($x_0 \in \Omega$), $\Delta_0^m \leq \Delta_0^p$ e fazer $k = 0$
3: **Passos SEARCH e POLL para determinar** M_k eq. (3-9)
4: Realizar *Search* e, possivelmente, o *Poll* (ou apenas um das etapas, se um ponto de teste da malha melhore a função objetivo).
5: **SEARCH**: Avaliar f num subconjunto finito de pontos experimentais sobre a malha M_k (a estratégia que seleciona os pontos é fornecido pelo usuário).
6: **POLL**: Avaliar f num conjunto *Poll* P_k^p (ver eq. (3-11)).
7: **Atualização dos parâmetros**
8: Determinar o tipo de sucesso da iteração k.
9: Atualizar Δ_{k+1}^m e Δ_{k+1}^p de acordo com as regras apresentadas acima.
10: Fazer: $k \leftarrow k + 1$ e voltar para os passos *Search* e *Poll*.

3.6 Método de Nelder-Mead

O método Nelder-Mead foi criado com a idéia de encontrar a solução ótima local com base no *simplex* [24]. O *simplex* é um poliedro criado por $n + 1$ pontos independentes e afins no espaço, onde n é o número de variáveis de decisão do problema, e o vetor solução $x = [x_1, x_2, x_3, ..., x_n]$. Cada operação do procedimento *simplex* (Nelder-Mead) é realizado em todo o vetor solução.

Como uma analogia com os algoritmos evolucionários, o *simplex* no procedimento Nelder-Mead pode ser visto como uma população de indivíduos. No entanto, o número de indivíduos são $n + 1$ neste caso. Para poder definir o método Nelder-Mead, deve-se especificar 4 parâmetros que estão associados com os passos deste método, que são: parâmetro de reflexão (ρ), expansão (χ), contração (γ) e redução (σ). Segundo o artigo original [24], estes parâmetros deve-se satisfazer:

$$\rho > 0,\ \chi > 1,\ \chi > \rho,\ 0 < \gamma < 1,\ \text{e} \quad 0 < \sigma < 1 \tag{3-12}$$

Uma escolha comum destes parâmetros é vista em [40]:

$$\rho = 1,\ \chi = 2,\ \gamma = \frac{1}{2},\ \text{e} \quad \sigma = \frac{1}{2} \tag{3-13}$$

Como já mencionado, o método Nelder-Mead possui 5 etapas: Inicialização, Reflexão, Expansão, Contração e Redução.

PUC-Rio - Certificação Digital Nº 1121430/CA

3.6.1 Inicialização do Simplex

Nesta etapa, numa iteração k, o algoritmo gera um *simplex* Δ_k com $n+1$ vértices, cada ponto no espaço $\mathbb{R}^n$. Logo, nesta iteração deve-se começar ordenando as funções objetivos dos vértices do *simplex* $x_1^k, x_2^k, x_3^k, ..., x_{n+1}^k$, tal que:

$$f_1^k \leq f_2^k \leq f_3^k \leq ... \leq f_{n+1}^k \tag{3-14}$$

na qual $f_i^k = f(x_i^k)$. A iteração k gera um diferente *simplex* para a seguinte iteração ($\Delta_{k+1} \neq \Delta_k$). Partindo da idéia de que se deseja minimizar a função objetivo, faze-se as seguintes definições: x_1^k é o melhor ponto ou vértice do *simplex*, x_{n+1}^k é o pior ponto, e x_n^k é o seguinte pior ponto do *simplex*, isto também para f_{n+1}^k como a pior função objetivo, e assim por diante.

3.6.2 Reflexão do Simplex

Para esta etapa computa-se o ponto de reflexão x_r com a seguinte formulação:

$$x_r = \bar{x} + \rho(\bar{x} - x_{n+1}) = (1+\rho)\bar{x} - \rho x_{n+1} \tag{3-15}$$

na qual $\bar{x} = \sum_{j=1}^{n} \left(\frac{x_i}{n}\right)$ é o centróide de todos os pontos do *simplex*, exceto o pior deles, x_{n+1}. Logo computa-se $f_r = f(x_r)$. Se $f_1 \leq f_r < f_n$, aceita-se o ponto de reflexão x_r e finaliza-se a iteração. Um exemplo de reflexão bi-dimensional é mostrado na Figura 3.7.

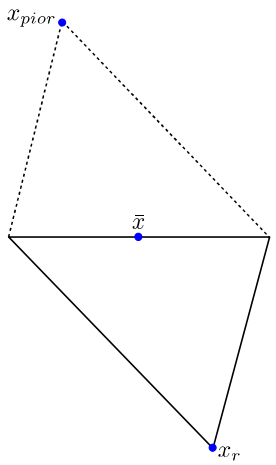

Figura 3.7: Simplex após um passo de reflexão.

PUC-Rio - Certificação Digital Nº 1121430/CA

3.6.3
Expansão do Simplex

Se $f_r < f_1$, determina-se o ponto de expansão x_e com a seguinte formulação:

$$x_e = \bar{x} + \chi(x_r - \bar{x}) = \bar{x} + \chi\rho(\bar{x} - x_{n+1}) = (1 + \rho\chi)\bar{x} - \rho\chi x_{n+1} \qquad (3\text{-}16)$$

e avalia-se $f_e = f(x_e)$. Se $f_e < f_r$, aceita-se x_e e finaliza-se a iteração; caso contrário ($f_e \geq f_r$), aceita-se x_r e finaliza-se a iteração. Um exemplo de expansão bi-dimensional é mostrado na Figura 3.8.

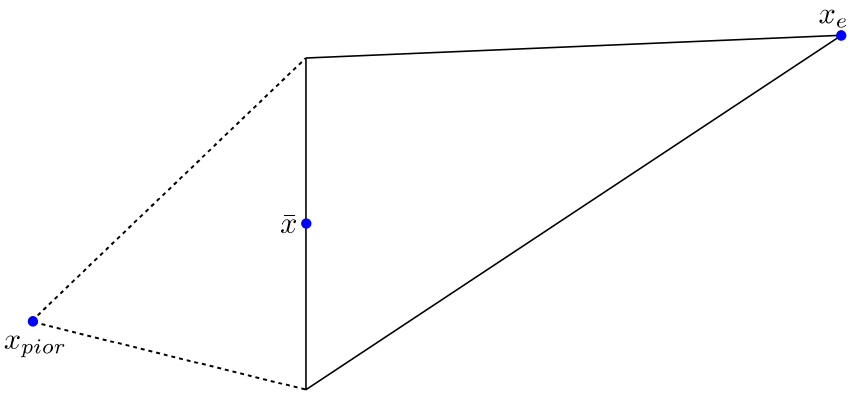

Figura 3.8: Simplex após um passo de expansão.

3.6.4
Contração do Simplex

Se $f_r \geq f_n$, realizar uma contração do simplex entre $\bar{x}$ e o melhor de x_{n+1} (o pior) e x_r. Nesta etapa podem-se encontrar dois tipos de contrações: fora do simplex e dentro dele.

Fora do Simplex

Se $f_n \leq f_r < f_{n+1}$ (x_r é estritamente melhor que x_{n+1}), realiza-se uma contração fora do simplex:

$$x_c = \bar{x} + \gamma(x_r - \bar{x}) = \bar{x} + \gamma\rho(\bar{x} - x_{n+1}) = (1 + \rho\gamma)\bar{x} - \rho\gamma x_{n+1} \qquad (3\text{-}17)$$

e avalia-se $f_c = f(x_c)$. Se $f_c \leq f_r$, aceita-se x_c e finaliza-se a iteração; caso contrário segue-se para a etapa de redução do simplex.

PUC-Rio - Certificação Digital Nº 1121430/CA

Dentro do Simplex

Se $f_r \geq f_{n+1}$, realiza-se uma contração dentro do simplex:

$$x_{cc} = \bar{x} - \gamma(\bar{x} - x_{n+1}) = (1-\gamma)\bar{x} + \gamma x_{n+1} \tag{3-18}$$

e avalia-se $f_{cc} = f(x_{cc})$. Se $f_{cc} < f_{n+1}$, aceita-se x_{cc} e finaliza-se a iteração; caso contrário segue-se para a etapa de redução do simplex.

Um exemplo bi-dimensional de contração dentro e fora do simplex é mostrado na Figura 3.9.

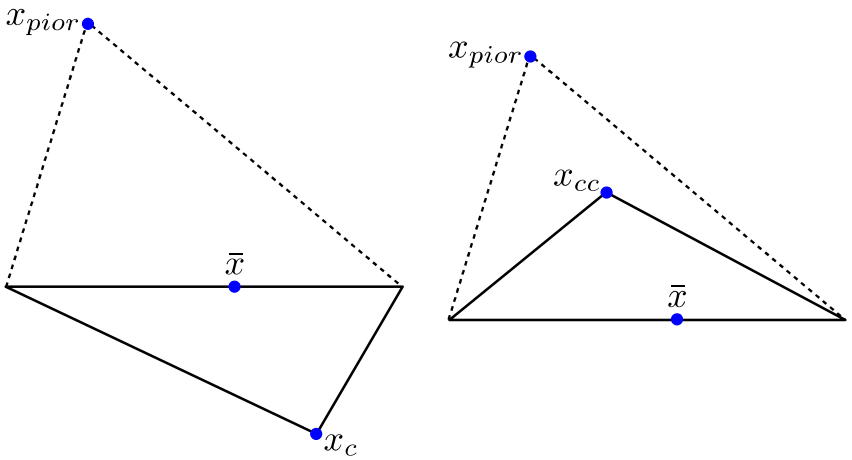

Figura 3.9: Simplex após um passo de contração.

3.6.5 Redução do Simplex

A redução do *simplex*, consiste em avaliar f nos n pontos definidos por $v_i = x_1 + \sigma(x_i - x_1)$ $(i = 2, 3, 4, ..., n+1)$. Logo, os vértices do simplex da iteração seguinte são $x_1, v_2, ..., v_{n+1}$. Um exemplo bi-dimensional de redução do simplex é mostrado na Figura 3.10.

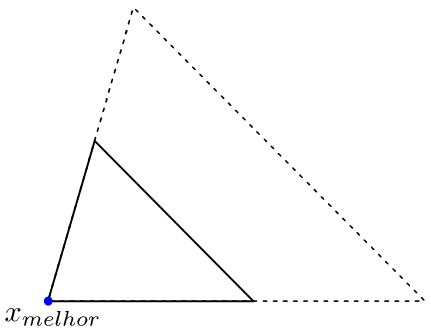

Figura 3.10: Simplex após um passo de redução.

PUC-Rio - Certificação Digital Nº 1121430/CA

O procedimento do método Nelder-Mead Simplex, é dado pelos Algoritmos 4 e 5 respectivamente. Na qual são detalhados todas as etapas do simplex incluindo a redução, que é a ultima etapa do método.

Algorithm 4 Nelder-Mead (Simplex)

while critério de parada não for satisfeito **do**
  Ordenar o Simplex $= x_1, ..., x_{n+1}$ tal que $f(x_1) \leq f(x_2) \leq ... \leq f(x_{n+1})$
  $\bar{x} \leftarrow \sum_{j=1}^{n} \frac{1}{n} x_k$
  $x_r \leftarrow \bar{x} + \rho(\bar{x} - x_{n+1})$
  if $f(x_1) \leq f(x_r) < f(x_n)$ **then**
    $Simplex \leftarrow \{Simplex \setminus \{x_{n+1}\}\} \bigcup \{x_r\}$
  else
    if $f_{x_r} < f(x_1)$ **then**
      $x_e \leftarrow \bar{x} + \chi(x_r - \bar{x})$
      if $f(x_e) < f(x_r)$ **then**
        Simplex $\leftarrow \{\text{Simplex} \setminus \{x_{n+1}\}\} \bigcup \{x_e\}$
      else
        Simplex $\leftarrow \{\text{Simplex} \setminus \{x_{n+1}\}\} \bigcup \{x_r\}$
      end if
    else
      if $f(x_r) < f(x_{n+1})$ **then**
        $x_c \leftarrow \bar{x} + \gamma(x_r - \bar{x})$
        if $f(x_c) \leq f(x_r)$ **then**
          Simplex $\leftarrow \{\text{Simplex} \setminus \{x_{n+1}\}\} \bigcup \{x_c\}$
        else
          Simplex $\leftarrow$ Redução(Simplex, σ)
        end if
      else
        $x_{cc} \leftarrow \bar{x} - \gamma(\bar{x} - x_{n+1})$
        if $f(x_{cc}) < f(x_{n+1})$ **then**
          Simplex $\leftarrow \{\text{Simplex} \setminus \{x_{n+1}\}\} \bigcup \{x_{cc}\}$
        else
          Simplex $\leftarrow$ Redução(Simplex, ρ)
        end if
      end if
    end if
  end if
end while
Retornar $(\{x_k \mid min\ f(x_k),\ x_k \in \text{Simplex}\})$

PUC-Rio - Certificação Digital Nº 1121430/CA

Algorithm 5 Redução

1: NovoSimplex $\leftarrow x_1$
2: **for** $i = 2$ até $n + 1$ **do**
3: $\quad v_i \leftarrow x_1 + \sigma(x_i - x_1)$
4: $\quad$ NovoSimplex $\leftarrow$ NovoSimplex $\bigcup v_i$
5: **end for**
6: Retornar NovoSimplex

Os métodos de otimização estudados até agora são utilizados para estimar os parâmetros dinâmicos do veículo. Estes métodos tem vantagens e desvantagens que dependem do problema de otimização a resolver, neste trabalho se faz uma comparação do desempenho de cada um destes algoritmos, com o objetivo de obter o método que da uma melhor solução ao problema.

PUC-Rio - Certificação Digital Nº 1121430/CA