4
Métodos Existentes

A hibridização de diferentes métodos é em geral utilizada para resolver problemas de escalonamento, por fornecer empiricamente maior eficiência na busca de soluções. Ela pode ser obtida mediante a composição de heurísticas baseadas em população, tais como, o algoritmo genético, *particle swarm optimization* (PSO), *ant colony optimization* (ACO), entre outros; ou com heurísticas de busca local, tais como busca tabu (TS), *simulated annealing* (SA) e *hill climbing* (HC).

A seguir é feita uma breve revisão dos métodos existentes para resolver os problemas de escalonamento e também a descrição do método de Newton multiobjetivo.

4.1
Algoritmo Genético

O algoritmo genético é um método de busca usado para resolver problemas de otimização, fazendo analogia à Teoria da Evolução de Darwin. Ele simula o mecanismo de sobrevivência natural (continuidade) do indivíduo ou cromossomo (da solução candidata) mais apto em uma população (conjunto de soluções canditadas), segundo uma função aptidão de acordo com o problema que se deseja resolver (KAMIRI *et al*., 2010).

O algoritmo genético tem sido aplicado em uma grande quantidade de problemas de escalonamento. Nesta adaptação, cada seqüência de operações e máquinas é vista como um indivíduo pertencente a uma população. Em ambientes de máquinas flexíveis, um cromossomo é composto por dois sub-cromossomos, onde um contém a informação da seqüência e o outro a informação da alocação das máquinas. O algoritmo genético começa com uma população inicial de soluções candidatas. As soluções candidatas são avaliadas segundo as funções objetivo e depois combinadas em uma única função denominada função aptidão. O valor da função aptidão na solução candidata indica a chance que ela tem em gerar novas soluções; quanto melhor a aptidão, maior é a chance. Operadores de

seleção, cruzamento e mutação são aplicados sobre as soluções da população (pais) de modo a gerar uma nova população de soluções candidatas (descendentes), que seja mais promissora ou mais apta que a população anterior. Este processo se repete até que o número de gerações atinja um limite máximo pré-fixado, que é a condição de parada do algoritmo $cond$. A Figura 24 mostra o algoritmo genético básico, onde gen denota o contador de gerações. A seguir, descrevem-se os operadores de geração de novas soluções.

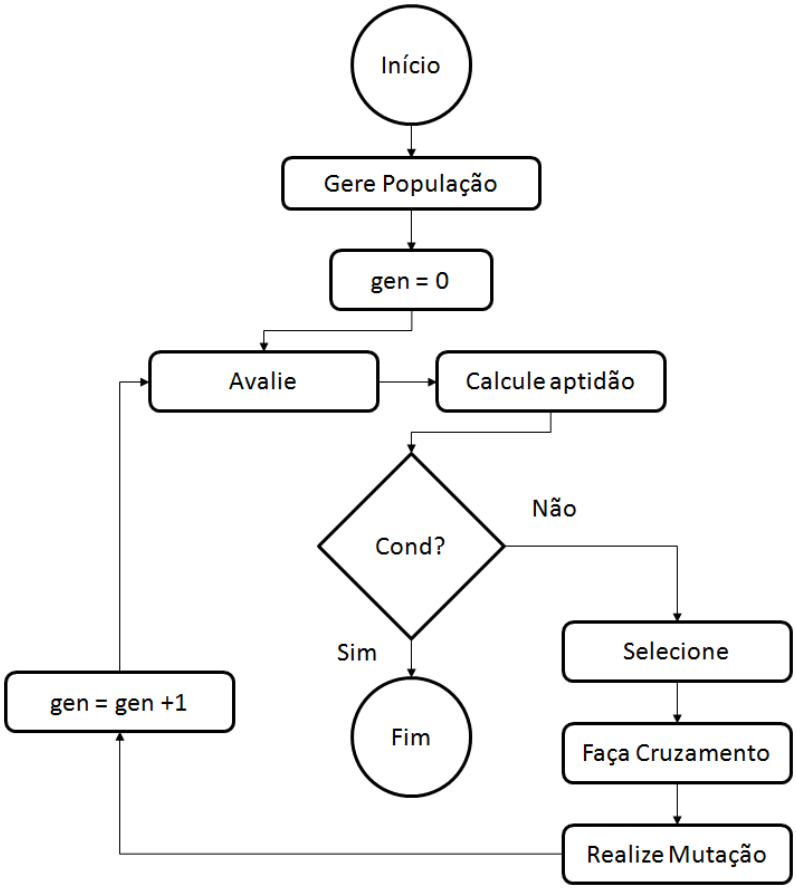

Figura 24 – Fluxograma do algoritmo genético básico
Fonte: Deb (2001, p.87)

4.1.1 Cruzamento

O Cruzamento é o operador de troca entre um par de soluções candidatas da geração corrente, que por sua vez gera um par de soluções candidatas descendentes semelhante às soluções candidatas pais. A propagação de características dos mais aptos em cada nova geração é a idéia desse cruzamento. O

cruzamento simples ou de um ponto foi usado em vários problemas de escalonamento (PONNAMBALAM *ET AL.*, 2004; PASUPATHY *ET AL.*, 2006; ZHANG *ET AL.*, 2006B). O operador cruzamento simples escolhe aleatoriamente um ponto de cruzamento, e a partir desse ponto é realizado o cruzamento de subseqüências dos cromossomos pais. No entanto, para assegurar a viabilidade dos novos descendentes, seu resultado deve ser combinado com a estratégia de mapeamento, para preencher os elementos restantes dos descendentes seguindo a ordenação do outro pai de tal forma que as novas soluções geradas não contenham elementos repetidos. A Figura 25 mostra um exemplo da aplicação do operador cruzamento simples de um FSP com 10 tarefas.

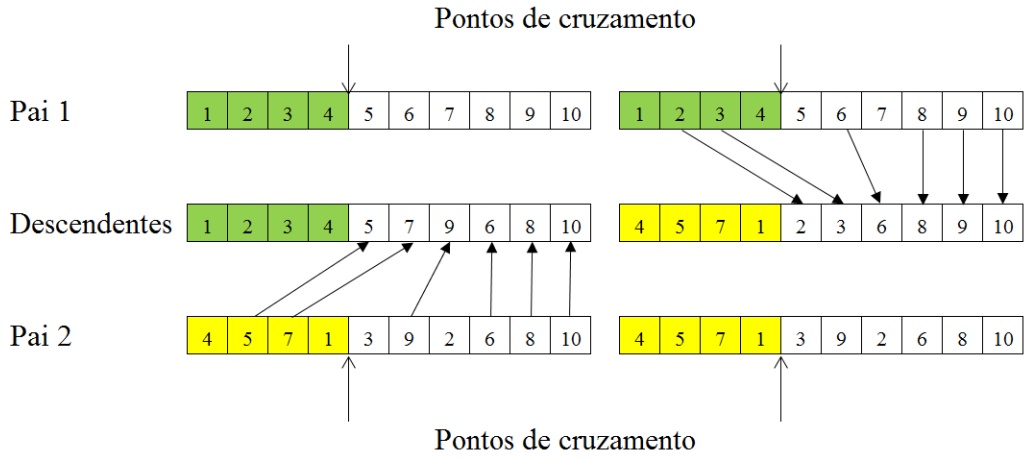

Figura 25 – Exemplo de cruzamento simples

4.1.2 Mutação

O operador mutação gera uma modificação arbitrária em uma solução candidata. É responsável pela diversidade dos indivíduos da população. A mutação aplicada para uma seqüência é realizada escolhendo-se duas posições aleatoriamente e trocando suas posições na seqüência (PONNAMBALAM *ET AL.*, 2004; PASUPATHY *ET AL.*, 2006; ZHANG *ET AL.*, 2006b). A Figura 26 mostra um exemplo do operador de mutação de um FSP com 10 tarefas.

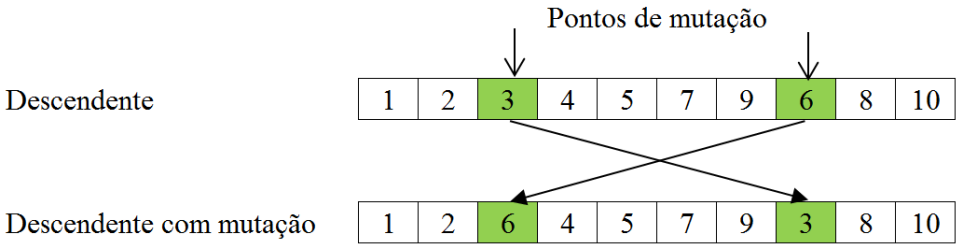

Figura 26 – Exemplo de mutação por troca de duas posições

Para problemas com máquinas flexíveis, a mutação é aplicada escolhendo-se uma posição aleatória e trocando a máquina alocada por outra selecionada aleatoriamente dentre as máquinas disponíveis para operação ou posição (ZHANG *ET AL.*, 2006b; LIU *ET AL.*, 2009; YANG E TANG, 2009). A Figura 27 mostra um exemplo de aplicação.

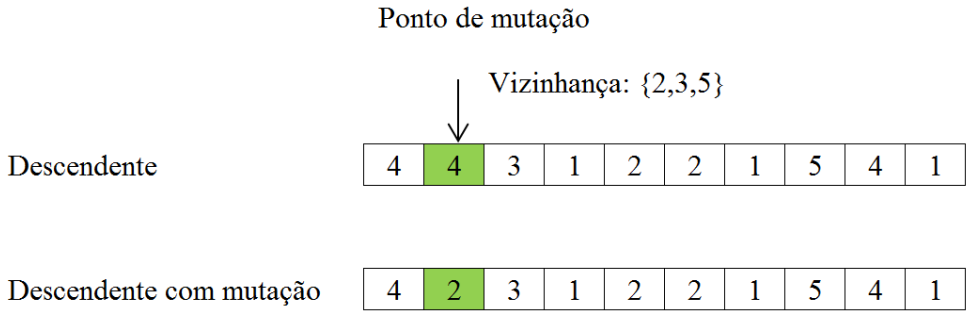

Figura 27 – Exemplo de mutação por troca de máquinas

4.2
Particle Swarm Optimization Algorithm

Proposto por Eberhart e Kennedy (1995), *particle swarm optimization* (PSO) é um método para resolver problemas de otimização inspirado no comportamento social de um enxame de partículas (XIA E WU, 2005). Originalmente, PSO foi apresentado como uma técnica de otimização para espaços contínuos.

Para os problemas de escalonamento, PSO é usado para determinar a melhor alocação de máquinas para as operações (XIA E WU, 2005; ZHANG *ET AL.*, 2009). Ele parte de uma população inicial de soluções candidatas (enxame). Cada

solução candidata (partícula) tem uma velocidade que é ajustada dinamicamente de acordo com suas experiências e das demais soluções candidatas.

Um modelo matemático usual para definir o PSO, proposto por Shi e Eberhart (1999), é dado por:

$$V_{i,t+1} = wV_{i,t} + Rand\ C_1(P_i - X_{i,t}) + rand\ C_2(P_g - X_{i,t})$$
$$X_{i,t+1} = X_{i,t} + V_{i,t+1},$$

onde , i é a i-ésima partícula; $X_{i,t}$ é a posição da partícula i na iteração t; $V_{i,t}$ é a velocidade da partícula i na iteração t; P_i é a melhor posição previa da partícula i até agora, chamado de $pbest$; P_g é a melhor posição prévia de todas as partículas, chamado de $gbest$; w é o peso inercial; C_1 e C_2 são constantes de aceleração; $Rand$ e $rand$ são números aleatórios no intervalo $[0,1]$.

Os passos do procedimento padrão para o PSO são descritos a seguir:

1. Inicialize o exame de partículas com posições aleatórias e velocidades.
2. Para cada partícula avalie o valor da função objetivo, atribua a $pbest$ a posição corrente e atribua a $gbest$ a posição da melhor partícula inicial.
3. Atualize a velocidade e a posição de cada partícula.
4. Avalie o valor da função objetivo de cada partícula.
5. Para cada partícula, compare o valor da função objetivo corrente com o valor de $pbest$. Se o valor corrente for melhor, então atualize $pbest$ com a posição corrente.
6. Determine a melhor partícula do exame e o melhor valor da função objetivo. Se o valor da função objetivo for melhor que $gbest$, então atualize $gbest$ com a melhor posição corrente.
7. Se a condição de parada for atingida, então o procedimento termina com resultado $gbest$ para o seu valor objetivo; senão retorne ao Passo 3.

4.3 Busca Tabu

Inicialmente, esta técnica foi apresentada por Glover (1989) e sua aplicação para resolver FSP foi feita por Widmer e Hertz (1989). A busca tabu é útil para

resolver problemas de otimização combinatória. O procedimento começa com uma solução inicial factível s e a vizinhança de s, $N(s)$. Em cada iteração é realizado um movimento de s para um vizinho $s^* \in N(s)$, permitindo que s^* seja pior que s em relação à função objetivo.

Com a finalidade de evitar possíveis ciclos associados aos movimentos, é definida a lista T de movimentos tabu ou proibidos na iteração corrente. Esta lista exclui movimentos que levam de volta para soluções anteriores, evitando que o procedimento cicle. No entanto, o tamanho da lista tabu $|T|$ não pode aumentar indefinidamente, devendo ter um limite. Assim, se $|T|$ esta no seu limite, antes de adicionar um novo elemento a T, um de seus elementos é removido, geralmente escolhe-se o elemento mais antigo.

4.4
Simulated Annealing Algorithm

Proposta por Kirkpatrick, Gelatt e Vecchi (1983), *simulated annealing* (SA) é um método de busca local probabilística, que tem sua origem nas áreas de ciência dos materiais e da física. Ele faz uma analogia a um processo térmico (recozimento), usado para obtenção de estados de baixa energia de um sólido.

O procedimento começa com uma solução inicial s e seleciona uma nova solução s' que pertence à vizinhança de s. Esta seleção pode ser feita de forma aleatória ou organizada. Por exemplo, para o problema *job shop*, s é a seqüência de tarefas. Então, a variação do valor da função objetivo é calculada como: $\Delta = f(s') - f(s)$. Para problemas de minimização do critério de desempenho, se $\Delta < 0$, a transição para a nova solução é aceita. Se $\Delta \geq 0$, então a transição para a nova solução é aceita com probabilidade $\exp(-\Delta/T)$, onde T é um parâmetro de controle chamado de temperatura. O SA geralmente começa a partir de uma temperatura elevada que é gradualmente reduzida. A transição para soluções piores em relação ao critério de desempenho é permitida para dar a oportunidade de se afastar de um mínimo local e encontrar uma melhor solução. Para cada temperatura, uma busca é feita para certo número de iterações. Quando a condição de parada é satisfeita, o algoritmo para.

4.5
Método de Newton multiobjetivo

Desenvolvido por Fliege *et al.* (2008), este método é utilizado para resolver problemas multiobjetivos com variáveis contínuas sem restrições. Sua característica principal é pertencer à classe dos algoritmos ditos de descida, isto é, cada novo iterado melhora simultaneamente todas as funções objetivo. A seguir, o método de Newton multiobjetivo é brevemente descrito.

4.5.1
Direção de Newton

Dada uma função $F: U \subset R^n \rightarrow R^m$ duas vezes continuamente diferenciável e um ponto não estacionário $x \in U$, a direção de Newton em x, denotada por $s(x)$, é obtida resolvendo-se o seguinte problema auxiliar:

$$\begin{cases} \min \max\limits_{j=1,\ldots,m} \nabla F_j(x)^T s + \frac{1}{2} s^T \nabla^2 F_j(x) s \\ s.t. \;\; s \in \mathbb{R}^n. \end{cases}$$

Denote por $\theta(x)$ o valor ótimo do problema auxiliar dado por:

$$\theta(x) = \inf_{s \in \mathbb{R}^n} \max_{j=1,\ldots,m} \nabla F_j(x)^T s + \frac{1}{2} s^T \nabla^2 F_j(x) s$$

e a direção de Newton:

$$s(x) = \operatorname*{argmin}_{s \in \mathbb{R}^n} \max_{j=1,\ldots,m} \nabla F_j(x)^T s + \frac{1}{2} s^T \nabla^2 F_j(x) s$$

4.5.2
Algoritmo de Newton Multiobjetivo

O algoritmo de Newton com estratégia de convergência global para minimizar F sem restrições é descrito a seguir.

1. (Inicialização) Escolha $x_0 \in U$, $0 < \sigma < 1$, $k = 0$ e defina $J = \{1/2^n | n = 0, 1, 2, \dots\}$.

2. (Ciclo principal)

   (a) Resolva a o problema auxiliar para obter $s(x_k)$ e $\theta(x_k)$.

   (b) Se $\theta(x_k) = 0$, então, pare. Senão, vá para o passo 2. (c).

   (c) Escolha t_k que seja o maior $t \in J$, tal que

$$x_k + ts(x_k) \in U,$$

$$F_j(x_k + ts(x_k)) \leq F_j(x_k) + \sigma t\theta(x_k), \qquad j = 1, \dots, m.$$

   (d) (Atualização) Defina

$$x_{k+1} = x_k + t_k s(x_k)$$

   e faça $k = k + 1$. Vá para o passo 2. (a).

Agora, vamos apresentar um exemplo da aplicação do método de Newton, para um problema multiobjetivo encontrado em Preuss *et al.* (2006):

$$Min\ f(x_1, x_2) = x_1^4 + x_2^4 - x_1^2 + x_2^2 - 10x_1x_2 + 20$$

$$Min\ g(x_1, x_2) = (x_1 - 1)^2 + x_2^2$$

$$s.a$$

$$-2 \leq x_1 \leq 2$$

$$-2 \leq x_2 \leq 2$$

A Figura 28 apresenta as imagens de x_1 e x_2 num gráfico no espaço dos valores dos objetivos.

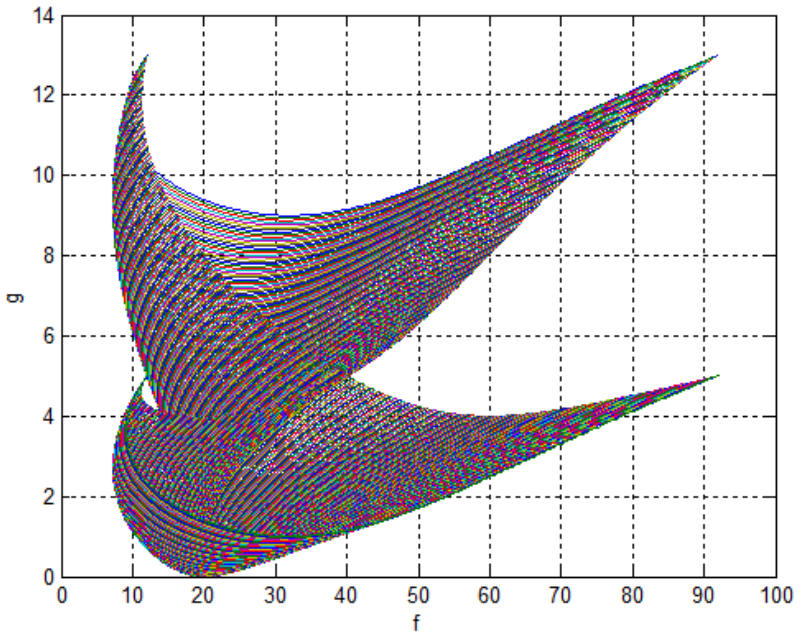

Figura 28 – Região no espaço dos valores dos objetivos para o exemplo

A Figura 29 apresenta a aplicação do método de Newton, para 150 pontos iniciais. Os iterados seguem em direção ao sudoeste com o progresso do algoritmo. A Figura 30 mostra a aproximação da *fronteira de Pareto*.

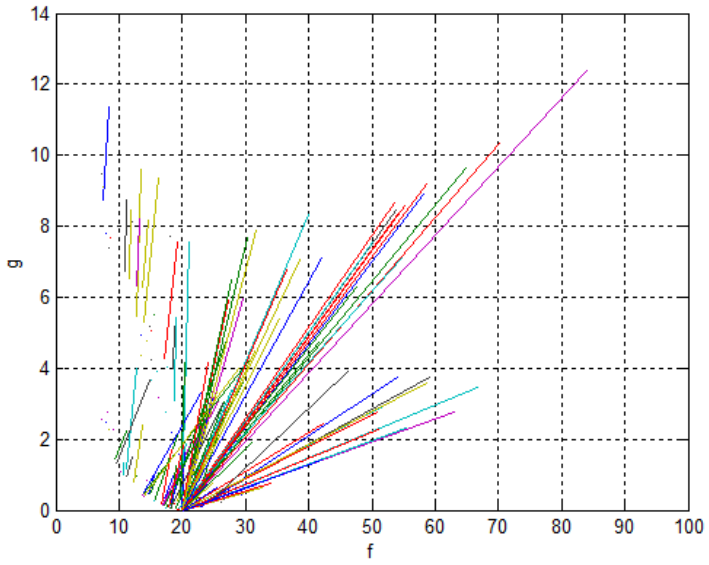

Figura 29 – Aplicação do método de Newton para o exemplo

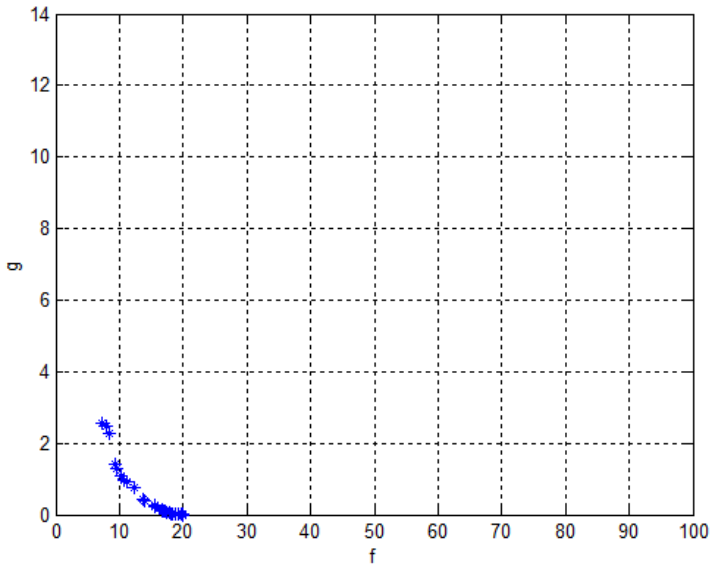

Figura 30 – Aproximação da *fronteira de Pareto* do exemplo