# 2
# Heurísticas e métodos aproximados

Métodos aproximados, comumente chamados de métodos heurísticos [33], são algoritmos que, ao invés de buscar uma solução comprovadamente ótima para um problema, buscam uma solução que seja próxima o suficiente do ótimo. Como vários problemas interessantes de otimização combinatória são NP-difíceis, há pouca esperança para que estes sejam resolvidos em tempo polinomial. Para esta classe de problemas, há uma fronteira no tamanho dos problemas que os algoritmos exatos resolvem: até essa fronteira, o comportamento destes costuma ser polinomial; após a fronteira, o tempo computacional explode. Embora os tamanhos de instância que representam essa fronteira de comportamento tenham aumentado consideravelmente nas últimas décadas, ainda é difícil resolver instâncias de tamanho razoável [7]. Os algoritmos heurísticos trocam a garantia da solução ótima em tempo exponencial pelo cálculo de uma boa solução aproximada em tempo polinomial. A medida em que a esperança de encontrar um algoritmo polinomial para problemas NP-completos diminui, os métodos heurísticos vêm recebendo cada vez mais atenção dos pesquisadores da área nos últimos anos [3].

Este capítulo fará uma rápida revisão dos métodos heurísticos mais comuns (os algoritmos de busca local) e tratará também das metaeurísticas, uma família de algoritmos heurísticos "genéricos" que tem apresentado excelentes resultados para difíceis problemas de otimização combinatória nos últimos anos. Em especial se falará das metaeurísticas VNS e *Path Relink*(PR), que serão importantes em capítulos mais a frente. A última seção do capítulo fala um pouco sobre as principais dificuldades e desvantagens no uso de métodos heurísticas.

## 2.1
## Busca local

Algoritmos de busca local são baseados no que é talvez o método de otimização mais antigo: tentativa e erro. A idéia é tão simples e natural que chega a ser surpreendente o grande sucesso que esses algoritmos alcançam em uma grande variedade de problemas [29].

O conceito de busca local está intimamente ligado ao conceito de vizi-

nhança. Uma vizinhança $N$ é uma função que mapeia de uma solução viável $f$ a um sub-conjunto de soluções viáveis $2^F$ (onde $F$ é o conjunto de todas as soluções viáveis). Em outras palavras, uma vizinhança nada mais é do que um conjunto de soluções que estão próximas (para alguma medida de proximidade) da solução inicial. Um exemplo famoso de vizinhança é a vizinhança 2-troca para o problema do caixeiro viajante, onde a vizinhança consiste em todos os *tours* em que, a partir do *tour* viável inicial, duas arestas são removidas e a rota é posteriormente conectada por outras duas arestas.

Dado uma estrutura de vizinhança $N$ e uma solução inicial $s$, um algoritmo de busca local é extremamente simples: se existe uma solução em $N(s)$ que seja melhor que $s$, substituir a solução $s$ e continuar a busca. Caso contrário, a busca chegou ao fim. A última solução encontrada em uma busca local é chamada de solução ótimo local para a vizinhança $N$ examinada.

Dada a simplicidade do algoritmo de busca local, fica claro que a performance do algoritmo depende crucialmente de suas duas entradas: qual a solução inicial e qual a estrutura de vizinhança a ser explorada. Nenhuma das duas perguntas possui uma resposta definitiva, elas dependem da intuição de quem está resolvendo o problema. Existe um claro *trade-off* entre vizinhanças grandes e pequenas: as grandes demoram a ser exploradas e produzem potencialmente melhores soluções, enquanto as pequenas resultam em uma busca local rápida porém potencialmente com piores resultados.

## 2.2 Metaeurísticas

Nos últimos 20 anos, um novo tipo de algoritmo aproximado surgiu. Este tipo de algoritmo tenta combinar métodos heurísticos básicos em um *framework* de alto nível, com o objetivo de explorar o espaço de buscas de maneira eficiente e efetiva. Esses métodos são atualmente chamados de metaeurísticas. O termo, introduzido inicialmente por Glover [14], vem da composição do verbo grego *heuriskein* (que significa "encontrar") com o prefixo meta (que significa "além, em um nível acima").

Algumas propriedades fundamentais que caracterizam um algoritmo metaeurístico são [3]:

- metaeurísticas são estratégias que guiam o processo de busca;
- seu objetivo é explorar o espaço de busca para encontrar soluções ótimas ou próximas da ótima;
- metaeurísticas não são específicas para um problema;
- metaeurísticas fazem uso de informações específicas do problema através do uso de heurísticas, que são controladas pela estratégia geral.

Em suma, podemos dizer que metaeurísticas são estratégias de alto nível para percorrer o espaço de busca utilizando diferentes métodos. Uma das características mais importantes de uma metaeurística é como ela consegue manter um balanço entre métodos de intensificação e diversificação. O termo "intensificação" normalmente se refere a como explorar o espaço de busca que já foi descoberto (por exemplo, utilizando algoritmos de busca local nas soluções encontradas), enquanto o termo "diversificação" se refere a exploração de regiões desconhecidas do espaço de busca. O balanço é importante para que o algoritmo metaeurístico não fique preso a ótimos locais e também não tente explorar demasiadamente o espaço de busca e deixe de melhorar soluções já conhecidas [3].

As estratégias de busca utilizadas variam fortemente entre cada metaeurística. Neste trabalho, iremos nos focar nas metaeurísticas VNS e *PathRelink*, descritas nas seções a seguir.

### 2.2.1 Variable Neighborhood Search (VNS)

A metaeurística de busca em vizinhança variável (*Variable Neighborhood Search*, VNS) foi proposta inicialmente por Hansen and Mladenovic [28]. Partindo do fato de que a maioria dos problemas possuem várias estruturas de vizinhanças interessantes e, ocasionalmente, disjuntas, o VNS toma proveito dessa coleção de vizinhanças fazendo uma busca que explora sistematicamente todas as vizinhanças, até encontrar uma solução que seja ótimo local para todas elas. O algoritmo 1 mostra a estrutura geral do algoritmo VNS [3].

Mesmo sendo uma metaeurística que explora fortemente o conceito de busca local, na linha 4 pode-se ver que a metaeurística tem um passo de diversificação, onde uma solução qualquer da vizinhança é escolhida. Entretanto, o passo de diversificação ainda leva em conta a melhor solução encontrada, na esperança de que uma solução vizinha desta possa levar a novos ótimos locais (ou globais).

Para que o VNS não explore repetidamente as mesmas soluções, é interessante que a lista de vizinhanças escolhidas seja de cardinalidade crescente ($|\mathcal{N}_1| < |\mathcal{N}_2| < \ldots < |\mathcal{N}_{k\_max}|$) e que, idealmente, as vizinhanças sejam disjuntas [28]. Existem diversas variações da busca VNS, aplicadas a diversos problemas diferentes - um bom *survey* sobre esta metaeurística pode ser encontrado em [20].

**Algoritmo 1** Variable Neighborhood Search

**Entrada:** conjunto de vizinhanças $\mathcal{N}_k$ ($k = 1 \ldots k\_max$), solução inicial $s$
**Saída:** solução ótimo local em todas as vizinhanças $s$

```
1:  while condições de término não atingidas do
2:    k = 1
3:    while k < k_max do
4:      s' = PickAtRandom(N_k(s))
5:      s'' = LocalSearch(s', N_k)
6:      if s'' melhor que s then
7:        s = s''
8:        k = 1
9:      else
10:       k = k + 1
11:     end if
12:   end while
13: end while
14: return s
```

### 2.2.2 Path Relink

Desenvolvida originalmente por Glover et al. [15], a metaeurística *Path Relink* (PR) parte de princípios semelhantes às metaeurísticas evolucionárias, como algoritmos genéticos [3]: a cada passo do algoritmo um conjunto de soluções de qualidade é mantido e, durante a chamada fase de *relinking*, algumas destas soluções são escolhidas e um método é utilizado para combiná-las e gerar outras soluções. O funcionamento básico do PR é descrito no algoritmo 2.

Uma das inovações do PR é a maneira proposta para combinar várias soluções: a partir de, por exemplo, duas soluções distintas, uma é chamada de solução-base e outra de solução-guia. Partindo da solução-base vão sendo geradas novas soluções que se "distanciam" da solução-base e se "aproximam" da solução-guia. A idéia é que as soluções que se encontram no caminho entre as soluções base e guia sejam semelhantes e, logo, de alta qualidade. Este conceito também pode ser extendido para fazer uso de múltiplas soluções-guia.

De maneira similar ao VNS, o PR tem conseguido excelentes resultados em uma variada seleção de problemas. Existem também diversas variações a partir do algoritmo aqui descrito - para uma boa visão de várias técnicas e usos de PR, o trabalho de Glover et al. [16] é recomendado.

**Algoritmo 2** Path Relink

**Entrada:** uma ou mais soluções iniciais (*seeds*) $S$, tamanho do conjunto de referência $n$
**Saída:** solução $s$

```
1: inicializar conjunto-referência de soluções R com soluções de S
2: while |R| < n do  // inicialização
3:    x = solução qualquer de R
4:    a partir de x, gerar uma ou mais soluções distintas (V)
5:    for cada solução v ∈ V do
6:       v = BuscaLocal(v)
7:       talvez adicionar v a R
8:    end for
9: end while
10: while condições de término não atingidas do  // path relink
11:    T = sub-conjunto de R
12:    V = combinar as soluções de T
13:    for cada solução v ∈ V do
14:       v = BuscaLocal(v)
15:       talvez adicionar v a R
16:    end for
17: end while
18: return  melhor solução de R
```

## 2.3 Dificuldades

O uso de métodos heurísticos para resolver problemas de otimização combinatória possui algumas sérias dificuldades, como o fato de que não há certeza se a solução é a melhor possível ou não (ou seja, heurísticas não conseguem provar que uma solução é ótima). Apesar de grave na teoria, esse problema é mitigado nas aplicações práticas, onde não se busca necessariamente a melhor solução possível, mas sim uma solução de excelente qualidade. Mais grave é o fato de que, muitas vezes, heurísticas (e metaeurísticas) não conseguem calcular quão distantes as soluções retornadas estão da solução ótima.

Outra dificuldade é o fato de que metaeurísticas não têm critérios de parada bem estabelecidos. Na maioria das vezes um limite de tempo é utilizado, o que é problemático quando não se sabe exatamente quão distante a busca está, naquele momento, de encontrar uma solução próxima da solução ótima.

Finalmente, talvez o mais grave problema no uso de heurísticas e metaeurísticas é que não existe um ferramental genérico que possa ser usado para qualquer problema (ou ao menos para um grande conjunto de problemas). Embora sejam genéricas na prática, a qualidade de uma metaeurística dependende diretamente da qualidade das heurísticas empregadas. Isso faz com que, para que os algoritmos consigam bons resultados, heurísticas altamente

especializadas para o problema tenham de ser utilizadas. Essas heurísticas conseguem ótimos resultados práticos para o problema em questão, mas não podem ser transportadas para outros problemas sem grande esforço. Essa faceta das metaeurísticas vai em contraste direto às técnicas de programção inteira, descritas no próximo capítulo.