Parte II

Abordagem Heurística

4
Heurísticas

Este capítulo faz uma breve introdução a respeito de alguns tipos de heurísticas muito utilizadas na literatura. Alguns conceitos básicos como vizinhança e busca local são também destacados. Em seguida, são comentados alguns trabalhos de heurísticas aplicadas ao TOP.

4.1
Heurísticas Construtivas

Uma heurística construtiva é um método que, numa sequência de passos, gera uma solução viável pra um determinado problema de maneira incremental.

Em geral, partindo-se de uma solução trivial - no caso do TOP seria um conjunto de rotas que não faz nenhuma visita - consegue-se gerar uma solução em um baixo tempo computacional. Nessa abordagem, a cada passo, incrementa-se uma solução parcial até que, devido a alguma restrição do problema, não seja mais possível incrementar a solução.

Vale destacar que as decisões adotadas em um passo não podem mais ser desfeitas nos passos futuros. Os critérios de decisão mais utilizados são: o critério guloso, no qual a decisão se dá observando-se a melhor alternativa local, sem considerar o contexto global; o critério aleatório, no qual a decisão é tomada ao acaso e o critério híbrido, no qual seleciona-se de maneira gulosa um conjunto de k possíveis decisões e, escolhe-se, de maneira probabilística a decisão a ser tomada.

Na maioria dos casos, tais heurísticas tendem a convergir para um ótimo local. Geralmente, não são obtidas soluções de boa qualidade com essas heurísticas, todavia, elas são muito utilizadas como pontos de partida de algoritmos mais elaborados.

4.2
Busca Local

Dado que se tem uma solução viável para um determinado problema, intuitivamente, é fácil observar que realizando-se algumas modificações na solução inicial, porém mantendo boa parte da sua estrutura, pode-se obter

outras soluções viáveis. Quanto maior for o número de modificações efetuadas sobre a solução inicial, mais distante será a nova solução obtida da solução original. De maneira análoga, se a maior parte da estrutura original for mantida, a nova solução será mais próxima da original. Ao conjunto de soluções que satisfazem ao critério intuitivo que indica a noção de distância e de proximidade entre soluções, denomina-se vizinhança. Por conseguinte, o conjunto de soluções vizinhas de uma solução original difere dependendo do critério de vizinhança adotado.

Uma busca local é, portanto, uma busca por soluções de melhor qualidade realizada em um conjunto de soluções vizinhas da solução original. Esse processo é iterativo, escolhendo-se uma nova solução *pivot* a cada iteração. A busca termina na iteração na qual não for encontrada nenhuma solução melhor do que a solução *pivot*.

O principal componente de uma busca local é a estrutura da vizinhança. Além disso, a definição de espaço de busca, a representação de soluções x e uma função objetivo $f(x)$ capaz de mensurar a qualidade das soluções são itens necessários para a definição de uma busca local.

4.3 Vizinhanças

Pode-se dizer que uma vizinhança é um conjunto de soluções, que, sob algum critério têm proximidade δ de uma solução original. O valor de δ influencia diretamente no tamanho da vizinhança. Formalmente, o conceito de vizinhança é definido por Puchinger e Raidl (PR05) como segue:

Definição 1 *Uma estrutura de vizinhança $N : S \rightarrow 2^S$ é uma função que associa um conjunto de vizinhos, denominado vizinhança $N(x) \subset S$ a toda solução $x \in S$.*

Estruturas de vizinhança são definidas por um movimento específico. Um movimento consiste numa operação realizada sobre uma solução $\bar{x_1}$ gerando outra solução $\bar{x_2}$.

O Algoritmo 4 descreve uma busca local genérica.

O pseudocódigo deixa em aberto como é feita a escolha pela solução $\bar{x_2}$. Em geral essa escolha é realizada de acordo com uma das seguintes estratégias:

- *Vizinho Aleatório* : aleatoriamente seleciona-se uma solução $\bar{x_2} \in N(\bar{x_1})$;
- *Primeiro vizinho melhor*: procura-se na vizinhança $N(\bar{x_1}$ uma solução melhor do que $\bar{x_1}$. A primeira solução que atender a esse critério é selecionada;

Algorithm 4: Busca Local

```
1 x̄1 ⟵ gera solução inicial;
2 repeat
3 |   escolha x̄2 ∈ N(x̄1);
4 |   if f(x̄2) é melhor que f(x̄1) then
5 |   |   x̄1 ⟵ x̄2;
6 until condição de parada;
```

– *Melhor Vizinho*: seleciona-se a melhor solução $\bar{x_2} \in N(\bar{x_1})$.

Se a busca local genérica for aplicada usando-se o critério de seleção do melhor vizinho ou mesmo o critério do primeiro vizinho melhor, até que nenhuma melhor solução possa ser encontrada, então pode-se afirmar que a solução atual é um ótimo local em relação à vizinhança N.

Definição 2 *Uma solução $\bar{x_2}$, num problema de maximização, é localmente ótima com respeito a uma estrutura de vizinhança N se*

$$f(\bar{x_2}) \geq f(\bar{x_1}) \forall x \in N(\bar{x_1}) \tag{4-1}$$

O sucesso da busca local depende da solução inicial, da estrutura da vizinhança e da função passo que estabelece o critério de seleção da próxima solução.

4.3.1 Heurísticas Matemáticas

Há abordagens que aplicam as ideias de metaeurísticas em MIP's. Em geral, utiliza-se algum critério para fixar variáveis ou colocar restrições de modo a reduzir o espaço de busca do problema original.

Fischetti e Lodi (FL03) introduziram o *local branching*, uma abordagem exata combinando a idéia de metaeurísticas com um resolvedor de MIP. Eles consideram MIP's genéricos com variáveis binárias 0-1. A idéia central consiste em, iterativamente, resolver um subproblema correspondendo a uma vizinhança clássica *K-Opt* utilizando o resolvedor MIP.

Isso se dá através da introdução de uma restrição baseada em uma solução conhecida $\overline{x}$. Essa restrição particiona o espaço de busca na vizinhança *K-Opt* e no restante do espaço de busca original: $\Delta(x, \overline{x}) \leq K$ e $\Delta(x, \overline{x}) \geq K + 1$. O valor de Δ corresponde à distância de Hamming entre as duas soluções (dois vertores de variáveis 0-1). Esse algoritmo é executado usando a idéia de *Branch-and-Bound* até que não seja encontrada uma solução melhor para dar continuidade ao processo. O mecanismo básico pode também ser estendido para

introduzir limites de tempo, alterar dinamicamente o tamanho da vizinhança k e adicionar estratégias de diversificação.

Danna et al. (DRP05) introduziu uma abordagem denominada *RINS - Relaxation Induced Neighborhood Search* com o intuito de explorar vizinhanças promissoras de soluções conhecidas. A idéia consiste em, ocasionalmente, derivar um subproblema de um nó da árvore de *Branch-and-Bound*. Esse subproblema corresponde a uma determinada vizinhança de uma solução incumbente. Nessa abordagem são executados os seguintes passos:

- As variáveis que têm o mesmo valor tanto na solução incumbente como na solução da Relaxação Linear são fixadas;
- Um limite é passado ao resolvedor baseado no valor da função objetivo da solução incumbente;
- O MIP do subproblema é resolvido com as variáveis que não foram fixadas.

O tempo de resolução do subproblema é limitado e se for encontrada uma solução melhor, esta é passada para o MIP global. Os autores compararam RINS com CPLEX padrão, *local branching* e combinações de RINS com *local branching*. Os resultados mostraram que RINS frequentemente foi melhor do que as demais abordagens testadas.

4.4 Heurísticas para o TOP

A primeira heurística publicada para o TOP foi desenvolvida por Chao et al. (CGW96) e é semelhante à heurística de cinco passos que os mesmos autores haviam proposto para o OP. Ao invés de somente selecionar o melhor caminho, os P melhores caminhos são selecionados e dois passos de reinicialização são utilizados, ao invés de um como acontecia no OP.

Tang e Miller-Hooks (TMH05) propuseram uma metaeurística *Tabu Search* embebida em um procedimento de memória adaptativa. No passo de inicialização da Busca Tabu, os parâmetros são configurados de modo a explorar somente um reduzido número de soluções vizinhas. No passo de melhoramento, procedimentos gulosos e aleatórios geram soluções vizinhas tanto viáveis como inviáveis. Os parâmetros garantem uma alternância contínua entre buscas em pequenas e grandes vizinhanças. No passo de avaliação, a melhor solução não tabu é selecionada e os parâmetros são ajustados baseados no tamanho atual da vizinhança e na qualidade da solução. O procedimento de memória adaptativa funciona de maneira similar a um algoritmo genético. Os cromossomos

gerados podem ser oriundos de mais do que dois cromossomos ancestrais. Caminhos isolados são armazenados e combinados de modo a formar uma solução inicial para a busca tabu. O resultado da Busca Tabu atualiza os caminhos isolados.

Metaeurísticas mais recentes para o TOP são descritas em Archetti et al. (AHS07), Ke et al. (KAF08), Vansteenwegen et al. (VSBO09) e Souffriau et al. (SVV10).

Archetti et al. (AHS07) propuseram duas variações de Busca Tabu. Além disso, apresentou duas versões de *Variable Neighborhood Search*, uma rápida e uma lenta. Todas essas quatro metaeurísticas serão comparadas com o Algoritmo Evolucionário proposto neste trabalho e descrito no capítulo 5. As quatro implementações iniciam de uma solução base x_1 sobre a qual aplica-se um operador de *jump* de modo a promover um salto para uma outra solução x_2. Assim, a Busca Tabu é utilizada para melhorar x_2, gerando uma terceira solução x_3. Essa nova solução é então comparada com x_1. Na estratégia da VNS a solução x_3 é aceita somente se x_3 for melhor do que x_1. Na estratégia da Busca Tabu, por sua vez, a solução x_3 é sempre aceita. Esse processo se repete até que a condição de parada seja alcançada. De maneira análoga ao que acontece na heurística de cinco passos proposta por Chao et al. (CGW96), os vértices que não estão inclusos na solução são também organizados em caminhos. A Busca Tabu utiliza os dois tipos de operadores seguintes:

- *1-move* : move um vértice de um caminho para outro;
- *swap-vertice*: troca dois vértices de caminhos diferentes.

Os vértices são sempre incluídos na solução usando o algoritmo da inserção mais barata. Como, durante o processo, são aceitas soluções inviáveis, foi necessário desenvolver procedimentos para reduzir e outros para remover inviabilidades dos caminhos. Dois tipos de operadores de *jump* são utilizados nesses algoritmos. O primeiro deles é como se fosse uma série de operações *1-move*, porém sem incluir os vértices removidos em outro caminho. O segundo operador troca subconjuntos de vértices de dois caminhos ou ainda troca subconjuntos de vértices de um caminho por um subconjunto de vértices não inclusos na solução.

As duas versões de Busca Tabu utilizam apenas o segundo tipo de operador de *jump*. A diferença entre as duas versões é que uma aceita soluções inviáveis e a outra aceita somente soluções viáveis.

A VNS, por sua vez, também utiliza a Busca Tabu como busca local, porém com um número menor de iterações e aceita somente soluções viáveis. Sempre que uma solução é melhorada, aplica-se o algoritmo *2-opt*

(Lin e Kernighan(LK73)) com o objetivo de tentar reduzir a duração total do caminho. Para comparar diferentes soluções, foram propostas cinco métricas para mensurar a qualidade das soluções. As funções que implementam essas métricas são baseadas no prêmio coletado, no tempo consumido e na viabilidade dos caminhos. Dentre os trabalhos mencionados, o de Archetti et al. (AHS07) apresenta, em geral, os melhores resultados para o TOP. A eficácia dessa abordagem provavelmente está relacionada com o fato de serem gerados caminhos viáveis mesmo para os vértices não inclusos na solução.

Ke et al. (KAF08) implementaram um algoritmo de Colônia de Formigas aplicado ao TOP. Em cada ciclo é construída uma solução viável que é melhorada por uma busca local. O algoritmo pára quando um número máximo de ciclos é atingido. O procedimento de busca local consiste em encurtar os caminhos, usando *2-opt* (Lin e Kernighan (LK73)) e inserir posteriormente quantos vértices forem possíveis. Esse procedimento é repetido até que um ótimo local seja alcançado. Quatro métodos são propostos para gerar soluções viáveis:

- *ASe* : O primeiro deles é sequencial, pois, nessa abordagem, caminhos completos são construídos um após o outro;
- *ARC* : No aleatório-concorrente, um caminho é selecionado aleatoriamente, a cada iteração, para receber um novo vértice;
- *ADC* : No método determinístico-concorrente, a sequência de caminhos considerada para receber um novo vértice a cada iteração é fixa.
- *ASi*: No método simultâneo, a cada iteração, um vértice é adicionado a um dos caminhos até que todos os caminhos alcancem a duração limite.

O método sequencial foi o que obteve os melhores resultados para gerar a solução inicial, porém inferiores aos obtidos por Archetti et al. (AHS07). Entretanto, o tempo computacional foi menor na abordagem sequencial.

Vansteenwegen et al. (VSBO09) primeiramente propuseram uma *Guided Local Search* e uma VNS. Ambos os algoritmos aplicam uma combinação de intensificação e diversificação. Há um mecanismo de diversificação que simplesmente remove uma cadeia de clientes do caminho. Há dois mecanismos de intensificação, um voltado para maximizar o prêmio total coletado e outro visando a reduzir a duração das rotas. O VNS claramente supera o GLS e ainda com um tempo computacional inferior. O sucesso dessa abordagem se dá devido a vários fatores. Primeiramente, porque aceita soluções com valores ligeiramente piores do que o da solução base, desde que sejam soluções distantes. A importância de se ter uma boa estratégia de diversificação é também um fator relevante.

Souffriau et al. (SVV10) desenvolveram duas variações de *Greedy Randomized Adaptative for Search Procedure - GRASP* com *Path Relinking*. Alterando apenas o critério de parada, tem-se a versão lenta e capaz de gerar excelentes resultados e uma versão rápida que gera boas soluções, porém um pouco inferiores às geradas pelo algoritmo lento. No GRASP, são executados quatro algoritmos em sequência, até que nenhuma melhora seja alcançada. Primeiramente é gerada uma solução inicial. Baseado na razão entre o critério guloso e o aleatório, vértices são inseridos até que os caminhos estejam cheios a ponto de não suportar a adição de novos vértices. Devido à aleatoriedade, novas soluções iniciais são geradas a cada iteração. Em seguida realiza-se uma busca local em torno dessas soluções. O critério da busca local alterna entre maximizar o prêmio total coletado e minimizar o tempo de duração dos caminhos até que se chegue a um ótimo local em relação a todas as vizinhanças. Nesse momento, o *Path Relinking* introduz um conjunto de soluções elite. Além disso, é estabelecido um caminho virtual entre cada solução gerada pela busca local e cada solução elite. A melhor solução encontrada nesses caminhos virtuais é retornada. O conjunto de soluções elite é atualizado. A qualidade das soluções encontradas pelo algoritmo lento são compatíveis com os apresentados em Ke et al. (KAF08) e Archetti et al. (AHS07).

4.5 Conclusão

Este capítulo fez uma breve introdução acerca de heurísticas e apresentou, de maneira sucinta, as principais heurísticas já propostas na literatura para o *Team Orienteering Problem*. As abordagens heurísticas em geral consistem em gerar soluções iniciais e realizar buscas locais iterativamente. No capítulo 5, será apresentada uma abordagem heurística que segue também essa linha, porém realiza buscas em grandes vizinhanças utilizando Programação Matemática. No capítulo 6, a abordagem deste trabalho será comparada às principais metaeurísticas apresentadas neste capítulo.