# UMA HEURÍSTICA BUSCA TABU PARA O PROBLEMA DO MOCHILEIRO VIAJANTE

**Matheus R. R. Oliveira**
Universidade Federal de Viçosa
Viçosa - MG - Brasil
matheus.roberti@ufv.br

**André Gustavo dos Santos**
Universidade Federal de Viçosa
Viçosa - MG - Brasil
andre@dpi.ufv.br

**Matheus de Freitas Araujo**
Universidade Federal de Viçosa
Viçosa - MG - Brasil
matheus.f.freitas@ufv.br

**RESUMO**

O Problema do Mochileiro Viajante (PMV) é um problema multicomponente de otimização combinatória que combina dois outros problemas bem conhecidos na literatura: o Problema do Caixeiro Viajante (PCV) e o Problema da Mochila (PM). Neste trabalho é proposta uma abordagem usando uma Busca Tabu adaptada para a interdependência do problema e uma heurística para a definição de um plano de coleta para uma dada rota. Foram realizados testes computacionais para calibrar os algoritmos e os resultados obtidos pela abordagem proposta são comparados com outra abordagem existente na literatura, mostrando através da análise dos resultados que a Busca Tabu consegue encontrar resultados superiores em quase todas as instâncias solucionadas.

**PALAVRAS CHAVE. Problema do Mochileiro Viajante, Heurísticas, Problemas Multicomponentes.**

**Área principal: MH - Metaheurística, OC - Otimização Combinatória.**

**ABSTRACT**

The Traveling Thief Problem (TTP) is a multicomponent combinatorial optimization problem that combines two well-known problems in the literature: the Traveling Salesman Problem (TSP) and the Knapsack Problem (KP). This paper proposes an approach using a Tabu Search adapted to the interdependence of the problem and a heuristic for defining a collection plan for a given route. Computational experiments were realized to calibrate the algorithms and the results obtained were compared to another existing approach in the literature, showing that the proposed Tabu Search based heuristic reached better results in almost all instances tested.

**KEYWORDS. Traveling Thief Problem, Heuristics, Multicomponent Problems.**

**Main area: MH - Metaheuristics, OC - Combinatorial Optimization.**

## 1. Introdução

Diversos problemas são estudados pelo campo da otimização combinatória e alguns deles, devido a sua alta complexidade e aplicabilidade prática, são conhecidos como problemas clássicos da literatura, por exemplo: o Problema do Caixeiro Viajante (DANTZIG; FULKERSON; JOHNSON, 1954), o Problema de Roteamento de Veículos (RADIY, 2010), o Problema de Sequenciamento de Tarefas (MIHAILA, 2011), o Problema da Mochila (PISINGER, 1995), entre outros. Esses e outros problemas de otimização combinatória foram extraídos do mundo real e são estudados por pesquisadores com o objetivo de encontrar formas satisfatórias de solucioná-los e, com isso, contribuir para os setores industriais onde esses problemas são encontrados.

Em alguns casos os problemas clássicos estudados pela otimização combinatória eram parecidos com outros problemas encontrados no mundo real, mas, devido a uma pequena diferença entre eles, uma boa solução para esses problemas clássicos poderia não ser uma boa solução para suas versões alternativas, ou sequer uma solução viável. Para resolver esse conflito foram criadas variações dos problemas clássicos com alguma adaptação, como a adição de alguma restrição ou uma variação em sua função de avaliação.

A criação das variações de diversos problemas serviu para encurtar a distância entre os problemas estudados na teoria e os problemas encontrados na prática. Entretanto, alguns pesquisadores ainda enxergam uma grande distância entre esses dois tipos de problemas e argumentam que parte dessa distância existe porque muitos dos problemas encontrados no mundo real são multicomponentes (BONYADI *et al*, 2014), e não somente uma variação de outro problema. Um problema multicomponente consiste em dois ou mais problemas individuais, que, se resolvidos separadamente de forma ótima, não levarão necessariamente a uma solução ótima quando colocados em união (BONYADI; MICHALEWICZ; BARONE, 2013). Todo problema multicomponente possui duas características importantes: combinação e interdependência. A combinação existe porque um problema multicomponente é um problema composto de dois ou mais componentes conectados, de forma que esses componentes sejam problemas de otimização quando avaliados individualmente. A interdependência ocorre quando a combinação dos componentes é feita de forma com que um afete o resultado do outro.

A indústria tem se mostrado mais interessada em ter seus problemas solucionados como um todo, em vez de obter a solução dos diversos subproblemas que o compõe separadamente, ou, ainda pior, solucionar somente parte de seus problemas (BONYADI *et al*, 2014). Foram realizadas diversas pesquisas em otimização combinatória sobre formas de solucionar problemas individuais, e, com isso, existem hoje diversas técnicas para resolvê-los, que foram sendo aprimoradas com o passar do tempo e em vários casos nos fornecem uma garantia de qualidade da solução encontrada. Existem estudos com o objetivo de identificar quais os fatores que elevam a complexidade dos problemas de otimização e quais as principais dificuldades encontradas na hora de propor abordagens para solucioná-los (WEISE, 2009), entretanto, pouco se tem pesquisado sobre a aplicabilidade e efetividade das técnicas tradicionais quando aplicadas a problemas multicomponentes (BONYADI; MICHALEWICZ; BARONE, 2013). Este trabalho irá tratar sobre o Problema do Mochileiro Viajante (do inglês: *Traveling Thief Problem*), que é um problema multicomponente composto da combinação de dois problemas bem conhecidos: o Problema do Caixeiro Viajante (PCV) e o Problema da Mochila (PM).

## 2. O Problema do Mochileiro Viajante

Nesta seção o PMV é definido e um exemplo é usado para mostrar possíveis soluções e como sua interdependência afeta a qualidade da solução. Em seguida serão informadas as contribuições dos trabalhos relacionados disponíveis na literatura.

### 2.1. Definição do problema

O Problema do Mochileiro Viajante (PMV) foi proposto inicialmente com o intuito de dar início a exploração da efetividade das técnicas atuais de otimização combinatória para problemas

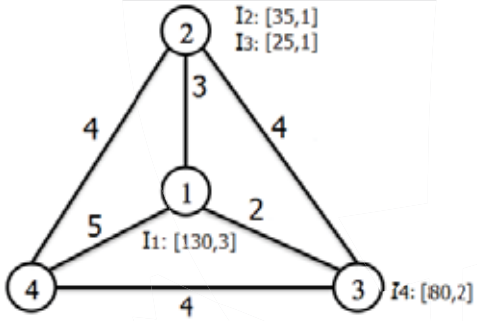
multicomponentes (BONYADI; MICHALEWICZ; BARONE, 2013). O PMV pode ser definido da seguinte forma: (i) dado um conjunto de cidades, cada qual com uma distância atribuída para se mover a qualquer outra cidade, defina uma rota de forma com que cada cidade seja visitada uma única vez, saindo de sua cidade de origem e voltando para ela no final; (ii) cada cidade possui seu próprio conjunto de itens, cada qual com seu próprio valor e peso, e os mesmos podem ser coletados quando a cidade na qual eles se encontram for visitada, exceto ao chegar na cidade de destino, contanto que a soma dos pesos de todos os itens coletados não ultrapasse a capacidade máxima de peso suportada pela mochila do mochileiro; (iii) há também uma velocidade máxima e mínima as quais o mochileiro usa como limites de sua velocidade, e a velocidade usada para percorrer a distância entre uma cidade e outra é dada em função dessas duas velocidades e o estado atual da mochila, e pode ser definida através da seguinte fórmula: $V_a = V_{max} - W_a(\frac{V_{max}-V_{min}}{W})$, sendo $V_a$ a velocidade atual, $V_{max}$ a velocidade máxima, $V_{min}$ a velocidade mínima, $W_a$ o peso atual da mochila e $W$ a capacidade máxima da mochila. Desta forma, a velocidade do mochileiro será $V_{max}$ quando a mochila estiver vazia ($W_a = 0$), diminuindo sempre que um novo item for coletado até alcançar $V_{min}$ quando a mochila estiver cheia; (iv), por fim, o mochileiro deverá pagar uma taxa $R$ para cada unidade de tempo gasta para percorrer toda a rota carregando os itens coletados. A solução ótima para este modelo é aquela que retorne o maior lucro possível, sendo o lucro a diferença entre a soma dos valores dos itens coletados e o custo para percorrer a rota. A função objetivo para obter o lucro é dada como: $L(x, y) = g(y) - R * f(x, y)$, sendo $x$ a rota escolhida, $y$ os itens coletados, $R$ o valor da taxa de aluguel da mochila, a função $f$ o custo em percorrer a rota $x$ contendo os itens $y$ e a função $g$ o valor total dos itens coletados $y$ (BONYADI; MICHALEWICZ; BARONE, 2013). É percebido que ao aumentar a taxa $R$ aumenta-se também o impacto do custo de percorrer a rota, havendo casos onde o $R$ pode ser grande o suficiente para fazer com que nenhum item aumente a qualidade da solução, reduzindo assim o problema para a versão clássica do PCV. Por outro lado, se $R = 0$, o PMV se reduz ao PM, pois o custo de percorrer a rota não interferirá na qualidade da solução.

A interdependência no PMV não o tornou um problema multiobjetivo. Sua função objetivo deve somente maximizar o lucro obtido pelo mochileiro ao término de sua viagem. Entretanto, uma segunda definição do problema foi proposta (BONYADI; MICHALEWICZ; BARONE, 2013) onde sua função de avaliação possui dois objetivos. Enquanto na primeira definição a interdependência funciona somente em uma via, na segunda definição temos uma interdependência entre os componentes do problema de forma com que a solução de um afete a solução do outro: como na primeira definição, os itens coletados vão interferir na velocidade para percorrer a rota, mas, além disso, os itens vão ter seu valor deteriorado em razão do tempo em que ficarem na mochila.

Por ser um problema recente, até o presente momento não há qualquer trabalho na literatura que propõe uma abordagem para solucionar a segunda definição do problema. Neste artigo somente a primeira definição do PMV será tratada.

## 2.2. Exemplo

**Figura 1: exemplo do PMV com 4 cidades e 4 itens**




Considere o exemplo apresentado na Figura 1. É possível visualizar que existem 4 cidades, sendo a cidade 1 a origem e destino final do mochileiro; há 4 itens, sendo o item 1 com valor de

130, peso 3 e localizado na cidade 1, os itens 2 e 3 localizados na cidade 2 e o item 4 localizado na cidade 3. O mochileiro deve definir sua rota de forma a sair da cidade 1, passar em todas as outras cidades e, por fim, retornar à cidade 1.

O mochileiro pode tentar diminuir o seu prejuízo ao percorrer a rota coletando itens, mas deve estar atento para evitar os itens que aumentarão o custo de percorrer a rota ao ponto de superar os valores obtidos ao coletá-los, diminuindo assim o seu lucro final. Considere um cenário para o exemplo da Figura 1 onde a capacidade de peso da mochila ($W$) seja 3, o custo do aluguel ($R$) da mochila por unidade de tempo seja $2,5$, a velocidade máxima seja 1 e a mínima $0,1$. Uma possível solução do PMV para esse cenário pode ser composta de: $x = 1, 4, 3, 2$ e $y = 0, 1, 0, 1$, isto é, o mochileiro passará na seguinte ordem pelas cidades 1, 4, 3 e 2, coletando os itens 2 e 4 ao passar pelas cidades 2 e 3, respectivamente. Ao sair da cidade 1 e ir para a cidade 4 o mochileiro não coleta item algum, então, sua velocidade é a velocidade máxima e o tempo necessário para percorrer esse trajeto é igual a: $5 \div \{1 - 0 \times [(1 - 0,1) \div 3]\} = T_{14} = 5$. O mochileiro segue sua rota saindo da cidade 4 e indo para a cidade 3 sem coletar item algum, portanto, segue em velocidade máxima e tem o seu tempo de percurso calculado como: $4 \div \{1 - 0 \times [(1 - 0,1) \div 3]\} = T_{43} = 4$. Ao chegar na cidade 3 o item 4 é coletado, portanto, o peso atual da mochila passa a ser 2, afetando assim a velocidade do mochileiro, fazendo com que o tempo gasto para ir da cidade 3 até a cidade 2 seja: $4 \div \{1 - 2 \times [(1 - 0,1) \div 3]\} = T_{32} = 10$. Chegando na cidade 2 o mochileiro coleta o item 3 tornando seu peso atual igual 3 $(1 + 2)$ e alcançando a capacidade máxima da mochila e atingindo sua velocidade mínima como visto no cálculo do tempo levado para fazer a viagem de volta a cidade 1 (origem e destino): $3 \div \{1 - 3 \times [(1 - 0,1) \div 3]\} = T_{21} = 30$. Quando o mochileiro retorna a cidade de origem o seu lucro total com a viagem é equivalente a $[(80 + 35) - 2,5 \times (T_{14} + T_{43} + T_{32} + T_{21})] = -7,5$. No exemplo acima o lucro da viagem ficou negativo porque o valor dos itens coletados não supera o custo da viagem, mas, se considerarmos a mesma rota sem coletar item algum, o resultado final será de $-(5 + 4 + 4 + 3) = -16$, mostrando que houve um ganho de $8,5$ por causa dos itens coletados.

É importante notar que a forma como a interdependência dos subproblemas age não permite que o PMV possa ser resolvido de forma ótima encontrando a solução ótima para o componente PCV e a solução ótima para o componente PM e as unindo em uma só solução. Essa situação pode ser vista quando tenta-se unir a rota ótima $x = 1, 2, 4, 3$ e a mochila ótima $y = 1, 0, 0, 0$ e a solução final para o PMV obtém um lucro de $-195$, valor menor do que o encontrado pela solução anterior. Nota-se também que a mesma solução sem coletar itens possui um lucro de $0 - 2,5 \times 13 = -32,5$, mostrando que o item coletado ocasionou em uma perda de $-162,5$ no lucro final. Encontrar a solução ótima para a rota e depois encontrar a solução ótima para a mochila usando a rota encontrada também não garante a solução ótima para o PMV.

### 2.3. Trabalhos Relacionados

Foi definido um modelo matemático para o PMV onde o mesmo é apresentado como um problema de otimização não linear inteira pertencente a classe NP-Difícil (MEI; LI; YAO, 2014b). Se PNP, problemas dessa natureza não podem ser resolvidos de forma determinística em tempo polinomial, tornando impossível a solução de instâncias grandes em tempo aceitável (GAREY; JOHNSON, 1979). Abordagens aproximativas como metaheurísticas tem se mostrado eficazes para encontrar soluções de boa qualidade (TALBI, 2009; BLUM; ROLI, 2003). Neste trabalho é proposta uma abordagem usando uma Busca Tabu (GLOVER, 1989) com uma busca local 2-OPT e uma heurística para criação da mochila para uma determinada rota.

Em (BONYADI; MICHALEWICZ; BARONE, 2013) os autores argumentaram que o volume dos dados tratados não é o fator que mais influencia na complexidade dos problemas encontrados hoje no mundo real, sendo esse fator a característica multicomponente desses problemas. Um problema multicomponente foi descrito como sendo um problema de otimização composto de dois ou mais problemas. Foram questionadas a efetividade e a aplicabilidade das técnicas de otimização

combinatória que são utilizadas em problemas 1-componente quando utilizadas em problemas multicomponentes. Para iniciar os estudos sobre essa classe de problemas os autores propuseram o Problema do Mochileiro Viajante. Foram propostas duas definições para o problema, cada qual com uma pequena diferença em relação ao outro, mas nenhuma forma de solucioná-los foi apresentada.

Em (POLYAKOVSKIY *et al*, 2014) os autores realizaram um *benchmark* de instâncias para o PMV, e propuseram uma heurística construtiva e duas metaheurísticas, uma busca local e um algoritmo genético, para encontrar soluções para esse *benchmark*. Para testarem as duas abordagens criaram também um conjunto de instâncias para o problema contendo 9720 instâncias. Os autores informaram no artigo os resultados para algumas instâncias de tamanhos diferentes, mas somente para instâncias de um determinado tipo. Entretanto, os resultados de suas três abordagens para cada uma das 9720 instâncias foram gentilmente disponibilizados por e-mail pelos autores do referente trabalho. Na seção 4 deste artigo serão mostrados os resultados para algumas dessas instâncias usando a metaheurística Busca Tabu definida na seção 3, em seguida será feita uma análise comparando os resultados obtidos com os resultados fornecidos pelos autores.

Em (MEI; LI; YAO, 2014a) os autores propuseram um Algoritmo Memético (AM) que usa uma estratégia de solução em duas etapas: uma rota é obtida através de uma busca local 2-opt e em seguida uma heurística é chamada para selecionar quais os itens que serão coletados para a rota fornecida. Uma estratégia similar é apresentada nesse trabalho.

Além do algoritmo proposto, os autores também informaram algumas otimizações que foram feitas para reduzir a complexidade assintótica do algoritmo de modo que o mesmo consiga resolver instâncias grandes do problema de forma eficiente. As três estratégias de redução de complexidade propostas foram:

1. Redução do tamanho da vizinhança: em vez de utilizar todos os vizinhos gerados pelo operador 2-opt, somente os vizinhos que também pertenciam a Triangulação de Delaunay (ŽALIK, 2005) são analisados.
2. Avaliação incremental: quando são aplicados os operadores de vizinhança sobre uma solução um novo vizinho é gerado e muitas vezes poucos componentes da solução são modificados em relação a solução de origem. Os autores criaram um algoritmo de avaliação que pudesse tirar proveito dessa característica, evitando recalcular partes das soluções vizinhas que não foram modificadas.
3. Modificação eficiente da solução: os autores informaram que inicialmente para modificar uma rota estava sendo utilizado um algoritmo de complexidade O(n), mas, ao modificar a estrutura de dados da mesma para uma *splay tree* (FREDMAN *et al*, 1995) houve uma redução de complexidade para O(log n).

Os autores destacaram a importância da redução da complexidade para o tratamento de problemas de larga escala, mesmo que para isso seja necessário abrir mão de maiores garantias, como não avaliar todos os vizinhos 2-OPT e somente levar em consideração aqueles com maior possibilidade de formarem rotas mais curtas, isto é, os vizinhos pertencentes a Triangulação de Delaunay (MEI; LI; YAO, 2014a; ŽALIK, 2005).

## 3. Heurísticas para o PMV

O PMV é redutível ao PCV quando o valor e peso de todos os itens for zero; e o PCV, por sua vez, foi demonstrado ser um problema NP-Difícil (PAPADIMITRIOU, 1977). Se $P \neq NP$, então não é possível usar métodos exatos para obter uma solução ótima de forma determinística em tempo polinomial para grandes instâncias deste problema, entretanto, métodos aproximados como metaheurísticas podem ser utilizados para obter soluções competitivas em tempo computacional aceitável. A seguir é descrita uma abordagem usando a metaheurística Busca Tabu para solucionar o problema.

A Busca Tabu (BT) é uma metaheurística que parte de uma solução inicial e utiliza uma estrutura de vizinhança para realizar buscas locais de modo a aperfeiçoar a qualidade da solução

atual. Para escapar de ótimos locais a Busca Tabu faz uso de uma Lista Tabu (LT) que contém os últimos N movimentos realizados dentro da vizinhança, sendo N o tamanho da LT. Esses movimentos são conhecidos como movimentos tabu e não podem ser efetuados para gerar novos vizinhos enquanto estiverem na lista (GLOVER, 1989).

Para este trabalho foi realizada uma adequação para adaptar essa metaheurística a característica multicomponente do problema. Além da busca local que está presente em qualquer implementação da Busca Tabu, há também o uso de uma heurística construtiva que serve para guiá-la, criando assim uma abordagem composta de duas etapas: uma busca local 2-OPT realizada somente sobre a rota da solução atual, sem levar em consideração os itens, e uma heurística construtiva que recebe as rotas e cria um plano de coleta de itens para elas.

### 3.1. Solução Inicial

Para criar uma solução inicial a rota foi criada de forma independente, mas para definir o plano de coleta a rota é levada em consideração. A heurística do vizinho mais próximo foi usada para criar uma rota válida para o componente PCV sem levar em consideração os itens das cidades. Em seguida essa rota é fornecida para uma heurística construtiva que define quais itens serão coletados, modificando somente o componente PM da solução. Foram realizados testes com diferentes heurísticas de criação de rotas para a geração da solução inicial: vizinho mais próximo, seleção aleatória de cidades, o Algoritmo de Kruskal (KRUSKAL, 1956) adaptado para as restrições do PCV e o Algoritmo de Economias (CLARKE; WRIGHT, 1964). O Algoritmo de Kruskal e o Algoritmo de Economia encontraram rotas melhores do que a heurística do vizinho mais próximo, porém, não houve efeito sobre a solução final para o PMV, então, a heurística do vizinho mais próximo foi usada por ser a mais rápida entre as três. A seleção aleatória de cidades para a criação da rota levou a uma solução final do PMV pior do que as outras abordagens. Esses resultados mostram que a qualidade de uma solução PMV não é completamente dependente da qualidade da solução de seus componentes de forma individual, mas uma má solução para o PCV ou para o PM pode sim afetar a qualidade do PMV: uma rota ruim pode levar muito tempo para ser percorrida, aumentando assim o custo da viagem até que nenhum plano de coleta consiga reverter o prejuízo.

### 3.2. Heurística Construtiva para o Plano de Coleta

No Algoritmo 1 é mostrado a heurística construtiva proposta para solucionar o componente da mochila pertencente ao PMV. Esse algoritmo se assemelha as heurísticas para o PM que selecionam os itens que possuam a maior razão entre seu valor e peso até que a capacidade da mochila seja alcançada ou os itens esgotados. Porém, esta estratégia de seleção de itens não é apropriada para o PMV, uma vez que coletar um item traz consigo não só um efeito positivo (seu próprio valor), mas também um negativo: o aumento de tempo para percorrer a rota e, consequentemente, o preço a ser pago pelo aluguel da mochila, podendo este aumento ser maior que o valor do item.

Uma heurística de seleção de itens para o PMV precisa avaliar tanto os efeitos positivos quanto os negativos de se coletar um determinado item, portanto, uma rota deve ser fornecida para obter essas informações. Para uma determinada rota é possível saber qual o verdadeiro impacto que um item possui na solução do problema, bastando verificar a diferença entre os valores retornados pela função de avaliação para uma solução contendo esse item e outra não.

A estratégia de seleção usada no Algoritmo 1 consiste em escolher os itens que terão o maior impacto positivo se acrescentados a solução. Para isso o impacto de um item foi definido como: $item_{IMP} = [AVAL(x, y \cup item) - AVAL(x, y)] \div item_{PESO}$. Dessa forma, os itens com os maiores impactos serão os que mais beneficiarão a solução. Entre as linhas 1 e 5 os itens são ordenados de forma decrescente pelo seu impacto se acrescentado na solução e as variáveis necessárias para a busca são inicializadas. Na linha 6 é definido o critério de parada do algoritmo de seleção dos itens verificando se não há mais itens a serem analisados, se a capacidade da mochila foi alcançada ou se os itens restantes possuem um impacto negativo; caso alguma dessas condições seja verdadeira, não é possível coletar qualquer item restante de modo a afetar de forma positiva a

solução. Na linha 7 é verificado se o item i pode ser coletado sem violar a restrição de capacidade da mochila, se sim, o mesmo é selecionado para a solução, pois, devido a reordenação da lista de itens, há uma garantia de que o impacto deste seja maior ou igual a qualquer outro item posterior.

**ALGORITMO 1** HEURÍSTICA PARA SELEÇÃO DE ITENS

**Entrada**: rota: uma rota válida para o problema
itens: os itens a serem coletados
N: a quantidade de itens
W: a capacidade de peso da mochila
**Saída**: um plano de coleta para a rota fornecida
**início**
Ordene os itens em relação ao seu impacto na solução
planoColeta $\leftarrow \varnothing$
w $= 0$
pesoCicloAtual $= 0$
i $= 1$
**enquanto** $i \leq N \wedge itens[i].impacto > 0 \wedge w \leq W$ **faça**
**se** $itens[i].peso + w \leq W$ **então**
planoColeta $\leftarrow$ itens[i]
w $=$ w $+$ itens[i].peso
pesoCicloAtual $=$ pesoCicloAtual $+$ itens[i].peso
Remova o item coletado
N $=$ N $-$ 1
**se** $pesoCicloAtual \geq W \times 0{,}5$ **então**
pesoCicloAtual $= 0$
Ordene os itens em relação ao seu impacto na solução
i $= 1$
**fim**

Se um item é coletado, o seu peso é acrescentado a solução e dessa forma o custo para percorrer a rota é aumentado. Quanto mais cheia a mochila estiver, maior será o custo para percorrer a rota, isto é, coletar um item quando o peso da mochila for 10% de sua capacidade aumentará em x o tempo do percurso, mas coletá-lo quando a capacidade da mochila for maior que 10% aumentará o tempo em uma quantidade maior que x. Essa propriedade mostra a importância em reordenar os itens quando o peso da mochila for atualizado, principalmente quando o peso atual da mochila estiver se aproximado da capacidade máxima da mesma. Entretanto, existem itens com peso tão baixo que se coletados não causam grande impacto no tempo para percorrer a rota, então, reordenar toda a lista de itens quando os mesmos forem coletados se torna uma operação desnecessária. Há uma verificação na linha 13 do Algoritmo 1 que tem como objetivo permitir que a lista de itens seja reordenada somente se 5% da capacidade máxima da mochila foi preenchida após a última reordenação, caso contrário o algoritmo continuará sua execução com a lista de itens não sendo reordenada.

### 3.3. Busca Tabu

A estratégia dessa abordagem consiste em utilizar a estrutura de vizinhança 2-OPT para gerar vizinhos baseando-se somente nas rotas e em seguida usar a heurística para criar um plano de coleta para cada vizinho, tornando-os assim uma solução completa para o PMV. Fazer uso da heurística de construção da mochila para os vizinhos gerados pelo 2-OPT serve também como um guia para a Busca Tabu explorar mais o espaço de busca do PMV como um todo, em vez de explorar somente as regiões do espaço de busca que sejam mais interessantes para seu componente PCV de forma individual. A busca sempre será direcionada para a melhor solução completa para o PMV, que pode ser ou não um boa solução para os componentes PCV e PM de forma individual.

No Algoritmo 2 a Busca Tabu é mostrada em mais detalhes. A lista tabu mantém um histórico dos últimos L movimentos 2-OPT que foram realizados para gerar os melhores vizinhos usados pelas últimas L iterações da Busca Tabu, sendo o valor de L definido através de uma calibração do algoritmo descrita na seção 4. Se a lista tabu estiver cheia, o movimento 2-OPT mais antigo é removido para dar lugar ao mais recente. A BT possui a característica de aspiração por objetivo, isto é, se uma solução for gerada através de um movimento tabu, mas seu valor for maior do que o valor da melhor solução encontrada até o momento, a lista tabu é ignorada e essa solução é usada para dar sequência na busca.

No momento de criar a mochila para os vizinhos gerados pelo 2-OPT a BT seleciona os 100 vizinhos com a menor rota para essa tarefa. Essa estratégia serve para diminuir o tempo de cada iteração da Busca Tabu, permitindo com que uma maior parte do espaço de busca seja explorado. Foram realizados testes computacionais e essa estratégia obteve melhores resultados do que criar um plano de coleta e avaliar cada vizinho 2-OPT, uma vez que existem diversos vizinhos nessa vizinhança com rotas compridas e que dificilmente serão boas soluções para o PMV: rotas maiores implicam em um maior tempo para ser percorrido, e se o tempo for muito grande o custo do aluguel da mochila será mais alto. Ao contrário do que poderia ser uma boa estratégia para o PCV, selecionar somente o vizinho com a menor rota não é interessante para o PMV, uma vez que a melhor solução para este pode não ser a menor rota possível, mas sim uma rota maior que tenha as cidades que possuam itens promissores mais para o final do percurso.

**ALGORITMO 2** BUSCA TABU

**Entrada**: solucaoInicial: uma solução válida para o PMV
**Saída**: melhor solução encontrada pela BT
**início**
melhorSolucaoAtual = solucaoInicial
listaTabu $\leftarrow \varnothing$
**enquanto** *critério de parada não for alcançado* **faça**
vizinhos = 2OPT(melhorSolucaoAtual)
Defina o plano de coleta para os vizinhos usando a heurística da mochila
Selecione o melhor vizinho para o PMV
Selecione o melhor vizinho não tabu para o PMV
**se** *melhorVizinho.lucro* $>$ *melhorSolucao.lucro* **então**
melhorSolucao = melhorVizinho
melhorSolucaoAtual = melhorVizinho
**senão**
melhorSolucaoAtual = melhorVizinhoNaoTabu
Atualize a lista tabu com o movimento 2OPT da melhorSolucaoAtual
**fim**

## 4. Experimentos Computacionais

Nesta seção serão apresentados os resultados de todos os testes computacionais realizados com as heurísticas propostas. As instâncias usadas são descritas na seção 4.1. A calibração para encontrar o melhor tamanho para a lista tabu é apresentada na seção 4.2. Na seção 4.3 é feita uma análise das soluções encontradas pela Busca Tabu em relação aos resultados obtidos pelo Algoritmo Evolucionário (EA) proposto por Polyakovskiy *et al* (2014).

Um trabalho mais recente na literatura (MEI; LI; YAO, 2014b) propôs duas abordagens evolucionárias para solucionar o PMV, sendo elas um algoritmo *Cooperative CoEvolution* e um Algoritmo Memético. Os autores criaram suas próprias instâncias e compararam os resultados de suas duas abordagens somente entre elas, havendo instâncias com 10 cidades e 10 itens até 100 cidades e 150 itens. Entretanto, ao realizamos uma busca exaustiva para as 3 menores instâncias encontramos a solução ótima, sendo este inferior aos resultados informados pelos autores como

solução das heurísticas. Por isso não usamos seus resultados para fins de comparação. Os autores foram informados sobre esse ocorrido.

Todos os testes computacionais foram realizados em um computador com a CPU Intel(R) Core(TM) i5-3337U 1.80GHz, 4 GB memória RAM e o sistema operacional Windows 8.1. Os algoritmos propostos neste artigo foram programados usando a linguagem C++.

### 4.1. Instâncias

Para realizarem seus testes computacionais Polyakovskiy *et al* (2014) criaram 9720 instâncias para o PMV. Foram criadas 10 instâncias para cada uma das seguintes três características: mesma quantidade de cidades, mesma quantidade de itens por cidade e mesmo tipo de itens disponíveis. Para cada número de cidades diferentes existem instâncias com 1, 3, 5 ou 10 itens por cidade, e para cada um desses subgrupos existem instâncias com as seguintes três características de itens disponíveis:

- ***uncorrelated* (u)**: Os valores dos itens não estão relacionados ao seu peso.
- ***uncorrelated-similar-weights* (usw)**: Os valores dos itens não estão relacionados ao peso, mas os pesos de todos os itens são similares.
- ***bounded-strongly-correlated* (bsc)**: Os valores dos itens são altamente relacionados ao peso.

Um exemplo de instância é a eil51_n50_bsc_10 e sua leitura pode ser feita da seguinte forma: é a décima instância do grupo eil com 51 cidades e 50 itens com valor e peso fortemente relacionados. Dentro do grupo eil existem também instâncias com 150 itens, 250 itens e 500 itens, sendo estas as instâncias com 3, 5 e 10 itens por cidade, respectivamente. Há também instâncias do tipo u, usw e bsc para cada número de cidades e de itens diferentes.

### 4.2. Calibração

A Busca Tabu proposta para este trabalho possui dois parâmetros: o tamanho da lista tabu e o critério de parada. Entretanto, para realizar uma comparação justa, o critério de parada usado será o mesmo utilizado pela abordagem existente na literatura, que é de 10 minutos.

A Lista Tabu contém os últimos N movimentos 2-OPT que não poderão ser realizados enquanto não se passarem N iterações. O valor de N não pode ser pequeno a ponto de permitir que a Busca Tabu não consiga sair de um mínimo local e não pode ser alto o bastante para não proibir movimentos 2-OPT por um grande período de tempo e impedindo assim que melhores resultados sejam encontrados (TALBI, 2009). Para definir o tamanho foram realizados testes computacionais com a lista tabu de tamanho 10%, 20%, 30%, 40% e 50% do número de cidades. Os testes foram realizados com as instâncias do grupo st70 e kroA100, cada uma contendo, respectivamente, 70 e 100 cidades. Foi selecionada uma instância contendo todas as combinações dos seguintes subgrupos: 1, 5 e 10 itens por cidade e itens com a característica u, usw e bsc, totalizando em 1 (quantidade de instâncias deste tipo) $\times$ 5 (tamanho da lista) $\times$ 2 (quantidade de cidades) $\times$ 3 (itens por cidade) $\times$ 3 (tipos de itens) = 90 execuções.

**Tabela 1: número de melhores soluções encontradas para cada configuração do parâmetro.**

| Tamanho | Melhores Soluções |
|---|---|
| 10% | 8 |
| 20% | **10** |
| 30% | 9 |
| 40% | 8 |
| 50% | 3 |

Na Tabela 1 é apresentado o número de melhores soluções encontradas pela BT para cada tamanho da lista tabu experimentado. Para o tamanho da lista com 20% do número de cidades a BT conseguiu encontrar 10 dos melhores resultados para as instâncias analisadas, sendo esta configuração a que retornou as melhores soluções, portanto, sendo assim definida como a configuração usada para executar os testes computacionais apresentados na próxima seção. Além disso, foram realizados testes computacionais para mostrar a importância da reordenação da lista

de itens disponíveis na heurística de seleção de itens. Na Tabela 2 são apresentados os resultados obtidos usando a heurística que reavalia e reordena os itens quando o peso da mochila aumenta (H1) e uma heurística que ordena os itens somente no início de sua execução, não os reavaliando e reordenando quando o peso da mochila aumenta (H2). Esses resultados foram obtidos usando a mesma rota, sendo modificado apenas o plano de coleta definido por cada abordagem. É possível observar que o plano de coleta quando usada a heurística que reordena os itens a cada ciclo consegue obter resultados significativamente superior aos resultados da heurística que não reordena os itens, ordenando-os somente no início de sua execução. Por obter os melhores resultados a heurística H1 foi usada durante a Busca Tabu.

**Tabela 2: resultados usando as heurísticas H1 e H2 para uma mesma rota.**

| Instância | H1 | H2 |
|---|---|---|
| berlin52_n255_bsc_10 | **87235.29** | 82955.65 |
| berlin52_n255_u_10 | **42608.35** | 34815.26 |
| berlin52_n255_usw_10 | **47572.81** | 38601.91 |
| eil51_n250_bsc_10 | **49781.12** | 17702.47 |
| eil51_n250_u_10 | **34000.52** | 17177.70 |
| eil51_n250_usw_10 | **31747.54** | 8328.47 |
| kroA100_n297_bsc_10 | **58010.61** | 17119.87 |
| kroA100_n297_u_10 | **44628.83** | 26007.19 |
| kroA100_n297_usw_10 | **43294.09** | 19260.35 |

### 4.3. Análise dos Resultados

Na Tabela 3 são apresentados os resultados obtidos pela metaheurística BT e o Algoritmo Evolucionário (EA) existente na literatura (POLYAKOVSKIY *et al*, 2014). Ambos algoritmos possuem um critério de parada de 10 minutos, isto é, a busca é interrompida após esse tempo ser alcançado e a melhor solução encontrada até o momento é retornada. Outra abordagem usando uma random local search (RLS) foi proposta pelos mesmos autores no mesmo trabalho, entretanto, obteve resultados inferiores ao EA para as instâncias analisadas neste artigo e por isso seus resultados não serão analisados.

Os resultados para cada instância com característica diferente para os grupos eil51, berlin52, st70, rat99 e kroA100 são mostrados na Tabela 3 e as melhores soluções estão destacadas. É possível notar que a BT consegue obter os melhores resultados em 83 das 90 instâncias, sendo mostrado na coluna "BT – EA" a diferença entre o seu resultado e o resultado encontrado pela heurística EA. Na coluna "BT s/ Itens" é informado o valor da solução final encontrada pela BT se os itens forem removidos. Essa situação é equivalente a percorrer toda a rota em velocidade máxima e o seu custo será sempre negativo: não existem itens para acrescentar valor a solução, mas o custo do aluguel da mochila para percorrer a rota deve ser pago. Na coluna "Sol. Inicial" é apresentado o valor da solução que é usada pela BT como solução inicial para a busca ser iniciada, mostrando em diversas instâncias que a BT consegue sair de uma solução com lucro negativo e encontrar uma rota suficientemente boa para haver um plano de coleta que torne o lucro da viagem positivo.

As instâncias com 1 item ou 3 itens por cidade do grupo rat99 tiveram todas as melhores soluções encontradas pela BT, enquanto as instâncias com 5 itens ou 10 itens por cidade tiveram as melhores soluções encontradas pelo EA. É possível observar que houve poucas iterações da heurística BT para as instâncias em que ela não obteve melhores resultados do que o EA, mostrando assim o motivo disso ocorrer e uma limitação dessa abordagem em relação ao tamanho das instâncias a serem solucionadas.

**Tabela 3: resultados das instâncias analisadas.**

| Instância | EA | BT | BT - EA | BT s/ Itens | Sol. Inicial | It. Tabu |
|---|---|---|---|---|---|---|
| eil51_n50_bsc_10 | 9604.35 | **11180.05** | 1575.70 | -19295.54 | -15423.53 | 65 |
| eil51_n50_u_10 | 6130.09 | **7167.77** | 1037.68 | -10153.07 | -6782.67 | 64 |
| eil51_n50_usw_10 | 5418.48 | **5959.69** | 541.21 | -10591.33 | -12330.51 | 63 |
| eil51_n150_bsc_10 | 24121.29 | **26997.77** | 2876.48 | -64661.12 | -104825.55 | 62 |
| eil51_n150_u_10 | 18780.23 | **21522.96** | 2742.73 | -28449.60 | -13440.53 | 72 |
| eil51_n150_usw_10 | 14843.10 | **17927.85** | 3084.75 | -31782.37 | -34694.83 | 66 |
| eil51_n250_bsc_10 | 33708.71 | **49781.12** | 16072.41 | -118443.42 | -180971.28 | 71 |
| eil51_n250_u_10 | 29508.08 | **34000.52** | 4492.44 | -46487.79 | -21596.53 | 60 |
| eil51_n250_usw_10 | 26483.19 | **31747.54** | 5264.34 | -51862.34 | -46723.30 | 61 |
| berlin52_n51_bsc_10 | 7874.70 | **15331.30** | 7456.59 | -22021.96 | -23329.16 | 58 |
| berlin52_n51_u_10 | 7979.65 | **9151.72** | 1172.07 | -10374.89 | -856.02 | 64 |
| berlin52_n51_usw_10 | 7231.54 | **8655.27** | 1423.73 | -10239.98 | -12819.62 | 61 |
| berlin52_n153_bsc_10 | 31767.99 | **49373.15** | 17605.15 | -64777.05 | -110727.88 | 61 |
| berlin52_n153_u_10 | 20754.83 | **23589.93** | 2835.11 | -31025.40 | -6253.15 | 60 |
| berlin52_n153_usw_10 | 23208.87 | **26940.18** | 3731.31 | -31275.16 | -26292.19 | 65 |
| berlin52_n255_bsc_10 | 55942.48 | **87235.28** | 31292.80 | -120496.71 | -216521.13 | 60 |
| berlin52_n255_u_10 | 39386.03 | **42608.34** | 3222.31 | -47460.83 | -1988.90 | 61 |
| berlin52_n255_usw_10 | 40536.51 | **47572.81** | 7036.30 | -51386.48 | -33370.64 | 63 |
| berlin52_n510_bsc_10 | 117278.61 | **179376.75** | 62098.13 | -228530.31 | -392026.26 | 29 |
| berlin52_n510_u_10 | 86049.77 | **94354.89** | 8305.12 | -99844.63 | -13972.87 | 34 |
| berlin52_n510_usw_10 | 79072.25 | **93972.00** | 14899.74 | -101023.96 | -79443.80 | 32 |
| eil51_n500_bsc_10 | 75874.84 | **103220.98** | 27346.14 | -222064.01 | -325101.82 | 72 |
| eil51_n500_u_10 | 62588.19 | **72825.86** | 10237.67 | -100284.68 | -43659.11 | 60 |
| eil51_n500_usw_10 | 52793.76 | **64960.24** | 12166.48 | -101614.41 | -84754.56 | 60 |
| st70_n69_bsc_10 | 10212.02 | **12048.55** | 1836.53 | -25968.27 | -29989.64 | 67 |
| st70_n69_u_10 | 12234.81 | **14356.49** | 2121.68 | -10937.26 | 4645.94 | 78 |
| st70_n69_usw_10 | 9077.16 | **10626.42** | 1549.26 | -13774.71 | -5298.98 | 63 |
| st70_n207_bsc_10 | 42385.48 | **49422.32** | 7036.84 | -91551.32 | -89694.41 | 65 |
| st70_n207_u_10 | 30231.71 | **34612.74** | 4381.03 | -36227.90 | 713.43 | 64 |
| st70_n207_usw_10 | 28591.82 | **29398.59** | 806.77 | -40000.31 | -22237.67 | 65 |
| st70_n345_bsc_10 | 69353.76 | **89075.65** | 19721.89 | -139116.44 | -161515.27 | 49 |
| st70_n345_u_10 | 48568.99 | **49533.55** | 964.56 | -65706.27 | 1366.78 | 58 |
| st70_n345_usw_10 | 44832.20 | **51504.40** | 6672.20 | -65406.51 | -44066.83 | 49 |
| st70_n690_bsc_10 | 142550.43 | **159478.11** | 16927.68 | -283010.81 | -301522.69 | 18 |
| st70_n690_u_10 | 94962.56 | **96015.28** | 1052.72 | -137582.47 | -16766.69 | 21 |
| st70_n690_usw_10 | 89263.78 | **103315.15** | 14051.37 | -128944.05 | -73242.86 | 20 |
| rat99_n98_bsc_10 | 19910.91 | **159018.64** | 139107.74 | -457160.89 | -750563.32 | 72 |
| rat99_n98_u_10 | 13975.62 | **119652.97** | 105677.35 | -205607.03 | -83448.67 | 74 |
| rat99_n98_usw_10 | 12527.14 | **105671.57** | 93144.43 | -219221.95 | -207681.28 | 82 |
| rat99_n294_bsc_10 | 57418.50 | **82861.53** | 25443.03 | -216116.53 | -369395.37 | 78 |
| rat99_n294_u_10 | 39675.44 | **69423.99** | 29748.55 | -96913.81 | -36910.55 | 77 |
| rat99_n294_usw_10 | 37202.45 | **58207.53** | 21005.08 | -102149.72 | -92685.29 | 69 |
| rat99_n490_bsc_10 | **93366.59** | 17335.31 | -76031.28 | -43570.64 | -69560.17 | 26 |
| rat99_n490_u_10 | **70168.30** | 12365.18 | -57803.13 | -21145.58 | -12725.07 | 29 |
| rat99_n490_usw_10 | **60821.47** | 12972.58 | -47848.89 | -21560.59 | -22357.53 | 26 |
| rat99_n980_bsc_10 | **196269.46** | 51109.27 | -145160.19 | -139700.89 | -245364.13 | 9 |
| rat99_n980_u_10 | **138636.78** | 34560.47 | -104076.31 | -64392.98 | -54479.89 | 11 |
| rat99_n980_usw_10 | **126800.27** | 39954.74 | -86845.53 | -56703.11 | -21888.89 | 10 |
| kroA100_n99_bsc_10 | 14749.02 | **20353.12** | 5604.10 | -41626.23 | -40989.68 | 73 |
| kroA100_n99_u_10 | 14149.22 | **15432.37** | 1283.15 | -19978.38 | -2852.24 | 79 |
| kroA100_n99_usw_10 | 13879.76 | **14133.86** | 254.10 | -22193.25 | -18072.63 | 68 |
| kroA100_n297_bsc_10 | 39072.07 | **58010.61** | 18938.54 | -140829.92 | -156800.45 | 77 |
| kroA100_n297_u_10 | 41997.23 | **44628.82** | 2631.59 | -55274.26 | -5620.59 | 71 |
| kroA100_n297_usw_10 | 39463.00 | **43294.08** | 3831.08 | -59590.18 | -42561.08 | 71 |
| kroA100_n495_bsc_10 | 73185.28 | **79311.13** | 6125.85 | -235810.01 | -260023.56 | 68 |
| kroA100_n495_u_10 | 73997.67 | **77778.99** | 3781.32 | -95542.58 | -11046.39 | 81 |
| kroA100_n495_usw_10 | 66751.00 | **71705.39** | 4954.39 | -96875.74 | -69911.86 | 71 |
| kroA100_n990_bsc_10 | 167126.14 | **171872.05** | 4745.91 | -454649.87 | -454330.06 | 27 |
| kroA100_n990_u_10 | **148490.64** | 143345.51 | -5145.13 | -194579.38 | -30361.15 | 28 |
| kroA100_n990_usw_10 | 133913.51 | **141737.38** | 7823.86 | -197345.56 | -117567.24 | 29 |

## 5. Conclusão

Este trabalho propôs uma adaptação da heurística Busca Tabu para solucionar o PMV levando em consideração sua característica multicomponente. Foi proposta uma heurística para definir um plano de coleta e os experimentos computacionais mostraram a importância de verificar a rota e o peso atual da mochila antes de coletar qualquer item, confirmando a interdependência dos problemas. Foram realizados diversos experimentos para a calibração das heurísticas e análise dos resultados. A abordagem aqui proposta conseguiu superar quase todos os resultados encontrados na literatura para instâncias com até 100 cidades e 990 itens. Entretanto, essa diferença se torna menor conforme as instâncias vão crescendo devido ao baixo número de iterações que a Busca Tabu consegue realizar dado o limite de tempo imposto. Como trabalhos futuros propomos investigar

formas de tornar as iterações mais eficientes para assim permitir que a Busca Tabu explore um maior pedaço do espaço de busca a procura de melhores soluções.

**Agradecimentos:** os autores agradecem a CAPES e a FAPEMIG pelo apoio financeiro para a realização e publicação desta pesquisa.

**Referências**

**Blum, C.; ROLI, A.** (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, v. 35, n. 3, p. 268-308.

**Bonyadi, Mohammad Reza; Michalewicz, Zbigniew; Barone, Luigi.** (2013) The Travelling Thief Problem: The First Step in the Transition From Theoretical Problems to Realistic Problems. *IEEE Congress on Evolutionary Computation* (CEC), Cancún, p. 1037-1044.

**Bonyadi, M. R. et al.** (2014). Socially Inspired Algorithms for the Travelling Thief Problem. *Conference on Genetic and Evolutionary Computation* (GECCO), Vancouver, p. 421-428.

**Clarke, G. u; Wright, John W.** (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, v. 12, n. 4, p. 568-581.

**Dantzig, G.; Fulkerson, R.; Johnson, S.** (1954). Solution of a Large-Scale Traveling Salesman Problem. *Operations Research*, 2, 393–410.

**Fredman, Michael L.; Johnson, D. S.; Mcgeoch, L. A.** (1995). Data structures for traveling salesmen. *Journal of Algorithms*, v. 18, n. 3, p. 432-479.

**Garey, Michael R.; Johnson, David S.** (1979). A Guide to the Theory of NP-Completeness. San Francisco: WH Freeman Co. **Glover, Fred.** (1989). Tabu Search - Part I. *ORSA Journal on computing*, v. 1, n. 3, p. 190-206.

**Kruskal, Joseph B.** (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, v. 7, n. 1, p. 48-50.

**Lin, Shen.** (1965). Computer Solutions of the Traveling Salesman Problem. *Bell System Technical Journal*, v.44, n. 10, p. 2245-2269.

**Martello, Silvano; Toth, Paolo.** (1990). *Knapsack problems* New York: Wiley.

**Mei, Yi; Li, X.; Yao, X.** (2014a). Improving Efficiency of Heuristics for the Large Scale Traveling Thief Problem. *Simulated Evolution and Learning*. Springer International Publishing. p. 631-643.

**Mei, Yi; Li, Xiaodong; Yao, Xin.** (2014b). On investigation of interdependence between subproblems of the Travelling Thief Problem. *Soft Computing*, p. 1-16.

**Mihaila, Cristina.** (2011). Evolutionary Computation in Scheduling. 54 p. *Tese de doutorado* - Babes-Bolyai University. Cluj-Napoca.

**Papadimitriou, Christos H.** (1977). The Euclidean Travelling Salesman Problem is NP-Complete. *Theoretical Computer Science*, v. 4, n. 3, p. 237-244.

**Pedro, Odivaney R.** (2013). Uma abordagem de Busca Tabu para o Problema do Caixeiro Viajante com Coleta de Prêmios. 108 p. *Dissertação de mestrado* - Universidade Federal de Minas Gerais. Belo Horizonte.

**Pisinger, David.** (1995). Algorithms for Knapsack Problems. *Tese de doutorado* - University of Copenhagen, Copenhagen.

**Polyakovskiy, S. et al.** (2014). A Comprehensive Benchmark Set and Heuristics for the Travelling Thief Problem. *Genetic and Evolutionary Computation Conference* (GECCO), Vancouver.

**Talbi, E. G.** (2009). *Metaheuristics: from design to implementation*. New Jersey: John Wiley.

**Weise, Thomas. et al.** (2009). Why is optimization difficult? Nature-Inspired Algorithms for Optimisation. *Studies in Computational Intelligence 193*, p. 1-50.

**Žalik, Borut.** (2005). An efficient sweep-line Delaunay triangulation algorithm. *Computer-Aided Design*, v. 37, n. 10, p. 1027-1038.