Incorporando Mineração de Dados a uma Heurística Estado-da-Arte para o Problema da Mínima Latência*

Ítalo G. Santana, Alexandre Plastino, Isabel Rosseti
Instituto de Computação - Universidade Federal Fluminense
Rua Passo da Pátria, 156, Bloco Computação - CEP 24210-240 - Niterói - RJ - Brasil
{isantana,plastino,rosseti}@ic.uff.br

RESUMO

A hibridização de heurísticas com técnicas de mineração de dados tem fornecido resultados significativos para diversos problemas de otimização combinatória. Neste artigo, uma proposta de incorporação de mineração de dados em uma heurística estado-da-arte para o problema da mínima latência (PML) é apresentada. Experimentos computacionais realizados em 150 instâncias demonstraram que a heurística híbrida com mineração de dados foi capaz de igualar ou superar os resultados originais com menos esforço computacional. Além disso, 19 soluções com valores de custo menores que os da literatura foram obtidas.

PALAVRAS CHAVE. Metaheurísticas, Problema da Mínima Latência, Mineração de Dados.

Tópicos: MH - Metaheuristicas, OC - Otimização Combinatória

ABSTRACT

Hybridization of heuristics with data mining techniques has provided significant results for several combinatorial optimization problems. In this paper, an approach of incorporating data mining into a state-of-the-art heuristic for the minimum latency problem (MLP) is presented. Computational experiments carried out on 150 instances showed that the hybrid heuristic with data mining was able to match or improve the original results with less computational effort. Besides, 19 solutions with cost values less than those from literature were obtained.

KEYWORDS. Metaheuristics, Minimum Latency Problem, Data Mining.

Paper topics: MH - Metaheuristics, OC - Combinatorial Optimization

*Trabalho parcialmente financiado pela CAPES e CNPq.

1. Introdução

Como alternativa aos métodos exatos, metaheurísticas têm sido extensivamente aplicadas a problemas de otimização combinatória, pois oferecem soluções de boa qualidade em tempos de execução aceitáveis. Além disso, o interesse em versões híbridas de metaheurísticas tem aumentado bastante nos últimos anos, uma vez que, para muitos problemas de otimização, os melhores resultados da literatura são encontrados por essas metaheurísticas híbridas [Talbi, 2002]. Um exemplo bem-sucedido de hibridização de metaheurísticas foi empregado a uma heurística baseada em GRASP (*Greedy Randomized Adaptive Search Procedures*) [Feo e Resende, 1995] com técnicas de mineração de dados para o problema de empacotamento de conjuntos [Ribeiro et al., 2006], onde essa versão híbrida proposta foi capaz de alcançar melhores resultados em relação à versão original, tanto em termos de qualidade de solução quanto em tempo computacional.

Mineração de dados (MD), do inglês *Data Mining* (DM), é um processo de extração automática de conhecimento de bases de dados que pode ser representado por meio de padrões e regras [Han et al., 2011]. Esse processo foi primeiramente incorporado ao GRASP em [Ribeiro et al., 2006], nomeado de *Data Mining* GRASP (DM-GRASP). Essa proposta era baseada em identificar padrões em soluções de alta qualidade que poderiam ser utilizados como guia no espaço de busca do problema e, assim, de maneira eficiente, alcançar melhores resultados em menor tempo computacional, comparado com a heurística original. Posteriormente, o DM-GRASP foi aplicado ao problema de diversidade máxima e problema de replicação de servidores para transmissão *multicast* confiável [Santos et al., 2008], problema das p-medianas [Plastino et al., 2011], problema de projeto de redes a 2-caminhos [Barbalho et al., 2013] e ao problema do caixeiro viajante com coleta e entrega envolvendo um único tipo de produto [Guerine et al., 2016]. A incorporação de mineração de dados também foi realizada com sucesso na metaheurística *Iterated Local Search* (ILS) para o problema das p-medianas [Martins et al., 2014].

Neste trabalho, uma heurística estado-da-arte para o problema da mínima latência (PML), denominada GILS-RVND, desenvolvida por [Silva et al., 2012], é utilizada como heurística-base na proposta de hibridização com mineração de dados deste trabalho. A heurística original reúne componentes de GRASP, ILS e do método de busca local RVND (*Variable Neighborhood Descent with Random Neighborhood Ordering*). Resultados computacionais demonstraram que a versão híbrida com mineração de dados foi capaz de não somente melhorar os resultados da heurística original bem como obter uma redução considerável do tempo de execução, reduzindo, em média, 31.23% no grupo de instâncias de maior dimensão da literatura.

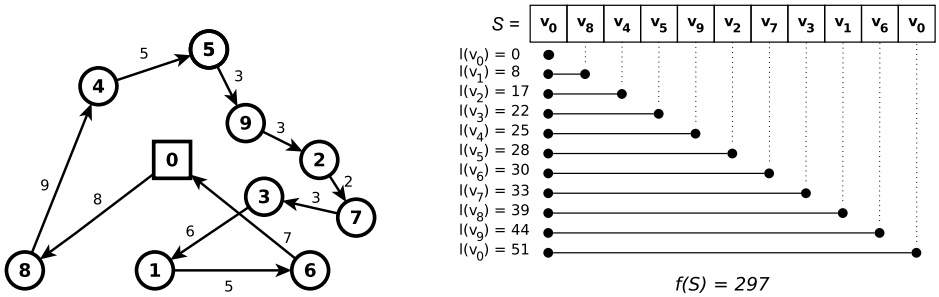
Este trabalho está organizado como se segue. A Seção 2 descreve o problema da mínima latência. As Seções 3 e 4 apresentam, em detalhes, o GILS-RVND e a nova proposta híbrida com mineração de dados, respectivamente. Em seguida, a Seção 5 reporta os experimentos computacionais sobre os conjuntos de instâncias utilizados por [Silva et al., 2012], bem como análises de significância estatística e de comportamento entre as duas heurísticas. Por fim, a Seção 6 aponta as conclusões deste estudo e trabalhos futuros.

2. O Problema da Mínima Latência

O problema da mínima latência, do inglês *Minimum Latency Problem*, é uma variante do problema do caixeiro viajante (PCV) e é definido como se segue. Seja $G = (V, A)$ um grafo completo direcionado, onde $V = \{0, ..., n\}$ é o conjunto dos vértices, sendo o vértice 0 definido como o ponto de partida (depósito) e os outros definidos como os clientes, e $A = \{(i, j) : i, j \in V, i \neq j\}$ é o conjunto dos arcos, cada um associado a um tempo de viagem t_{ij}. Assim, o PML tem como objetivo determinar um circuito hamiltoniano que minimiza o tempo total de espera de todos os clientes. Esse tempo de espera, também chamado de latência, é definido como sendo $\Sigma_{i=0}^{n} l(i)$, onde $l(i)$ representa o tempo de viagem entre o depósito e $i \in V$. O problema ainda considera o depósito como ponto inicial e final do circuito e que $l(0) = 0$. Em alguns trabalhos, a restrição de retorno ao depósito não é considerada, e portanto, a solução se torna um caminho hamiltoniano, sendo dada por $\Sigma_{i=0}^{n-1} l(i)$. Na literatura, o PML também é conhecido como *Traveling Repairman*

Problem [Tsitsiklis, 1992], *Delivery Man Problem* [Fischetti et al., 1993], *Cumulative Traveling Salesman Problem* [Bianco et al., 1993] e *School Bus Driver Problem* [Chaudhuri et al., 2003].

Apesar de o PML ser uma variante do PCV, ele possui uma orientação voltada aos clientes do problema, e isso o torna mais difícil que o PCV, pois pequenas alterações na sequência de visitação dos clientes podem impactar em grandes mudanças na solução. Assim como o PCV, o PML também é $\mathcal{NP}$-Difícil em sua versão de otimização [Blum et al., 1994]. Um exemplo de uma solução viável do PML é ilustrada na Figura 1. Seja S a solução na Figura 1(a) e sua sequência de clientes na Figura 1(b), a latência total (valor de custo) de S, ou $f(S)$, é denotado pela soma das latências $l(i)$ de cada cliente de S, que possuem custos acumulados.



(a) S com tempos de viagem nas arestas (b) Latência por vértice e valor de custo de S dado por $f(S)$

Figura 1: Exemplo de uma solução viável do PML e seu valor de custo. Figura adaptada de [Rios, 2016]

Devido às suas características, o PML pode ser facilmente encontrado em importantes aplicações práticas, onde o tempo de espera dos clientes é o fator crucial de análise, incluindo-se resgates em situações de desastres, serviços de entrega a clientes e recuperação de dados em redes de computadores [Moshref-Javadi e Lee, 2013]. Segundo [Abeledo et al., 2013], métodos exatos foram capazes de resolver instâncias de até 107 clientes, o que os tornam inviáveis em proporções realistas. Estratégias paralelas baseadas no GILS-RVND foram desenvolvidas para o mesmo problema em [Rios, 2016].

3. A heurística GILS-RVND

GILS-RVND é uma heurística baseada em GRASP [Feo e Resende, 1995], que é uma metaheurística multipartida composta por duas etapas: uma fase construtiva e uma estratégia de busca local. Na primeira etapa, uma solução inicial é gerada por meio de um método construtivo semiguloso controlado por um valor $\alpha \in [0, 1]$, que define o nível de aleatoriedade da construção, onde valores mais próximos de zero indicam uma construção mais gulosa. Em seguida, essa solução é submetida à etapa de busca local, onde é feita uma exploração sistemática do espaço de busca do problema até que o critério de parada seja satisfeito. É importante ressaltar que essas iterações são independentes, *i.e.*, a informação gerada por uma iteração não é propagada para as demais.

O pseudocódigo do GILS-RVND é apresentado no Algoritmo 1. Esse algoritmo realiza I_{Max} iterações (linhas 2-6), onde, para cada iteração, uma solução s é gerada pelo método construtivo, considerando um valor aleatório α de um conjunto R, parâmetro de entrada, que contém os possíveis valores para α (linha 3). Em seguida, s é submetida à etapa de busca local (linha 4). Logo depois, se s é melhor que a melhor solução global s^*, então s^* é atualizada (linhas 5-6). Após todas as iterações realizadas, s^* é retornada pelo algoritmo (linha 7).

O Algoritmo 2 descreve o método construtivo do GILS-RVND. No início, uma solução s é inicializada com o cliente 0 (linha 1), chamado de depósito, e a lista de clientes (LC) é gerada com os clientes remanescentes (linha 2). No laço, os clientes de LC são ordenados de forma crescente de acordo com o tempo de viagem entre cada cliente e o último cliente inserido em s (linha 5), que está sempre associado à variável r. Depois, a lista restrita de clientes (LRC) é preenchida com os $\alpha\%$

melhores clientes de LC (linha 6). Posteriormente, um cliente $c \in LRC$ é selecionado de forma aleatória, sendo inserido em s e removido de LC (linhas 7-10). Esse algoritmo termina quando LC se tornar vazia, e então, s é retornada pelo algoritmo (linha 11).

Algoritmo 1: GILS-RVND(I_{Max}, I_{ILS}, R)

```
1  f(s*) ← ∞;
2  para i ← 1, ..., I_Max faça
3  |  s ← MetodoConstrutivo(α ∈ R);
4  |  s ← BuscaLocal(s, I_ILS);
5  |  se f(s) < f(s*) então
6  |  |  s* ← s;
7  retorna s*
```

Algoritmo 2: MetodoConstrutivo(α)

```
1  s ← {0};
2  LC ← GerarLC();
3  r ← 0;
4  enquanto LC ≠ ∅ faça
5  |  LC ← OrdenarLC(LC,r);
6  |  LRC ← PreencherLRC(LC,α);
7  |  c ← SelecionarCliente(LRC);
8  |  s ← s ∪ {c};
9  |  r ← c;
10 |  LC ← LC − {c};
11 retorna s
```

A busca local reportada em [Silva et al., 2012], baseia-se na metodologia RVND, que por sua vez é uma extensão do método VNS (*Variable Neighborhood Search*) [Mladenović e Hansen, 1997]. O VNS é um método iterativo que aplica de forma sistemática um conjunto de vizinhanças, com a finalidade de melhorar a solução corrente de um determinado problema. Diferente do VNS onde a ordem de aplicação das vizinhanças é estipulada de maneira determinística, para o RVND a ordem é definida de forma aleatória. Em [Silva et al., 2012], cinco estruturas de vizinhanças clássicas do PCV são utilizadas no RVND: *Swap* - dois clientes da solução são trocados entre si; *2-opt* - dois arcos não adjacentes são removidos e dois outros são inseridos, mantendo a viabilidade da solução; *Reinsertion* - um cliente é realocado para outra posição da solução; *Or-opt-2* - um arco é realocado para outra posição da solução; *Or-opt-3* - dois arcos adjacentes são realocados para outra posição da solução.

Além do RVND, a etapa de busca local do GILS-RVND possui um mecanismo de perturbação originalmente proposto para o PCV, denominado de *double-bridge* [Martin et al., 1991]. Esse mecanismo consiste em remover quatro arcos e inserir quatro outros arcos na solução, mantendo a viabilidade da solução.

A estrutura básica da busca local é descrita no Algoritmo 3. Antes do laço, s' recebe uma cópia de s e é definida como a melhor solução da busca local (linha 1). No laço, definido entre as linhas 3 e 9, s é submetida ao RVND (linha 4). Caso s melhore s', então s' é atualizada e o contador $iterILS$ é reiniciado (linhas 5-7). Em seguida, o mecanismo de perturbação é aplicado em s' (linha 8) e $iterILS$ é incrementado (linha 9). Esse laço é executado enquanto $iterILS$ for menor que I_{ILS} iterações sem melhora em s' (linhas 3-9), caso contrário, s' é retornada (linha 10).

4. Heurística Híbrida com Mineração de Dados: DM-GILS-RVND

Em mineração de dados, diversas técnicas podem ser aplicadas em bases de dados a fim de identificar regras e padrões. Uma dessas técnicas, conhecida como mineração de conjuntos frequentes (MCF), tem sido utilizada na criação de heurísticas híbridas com mineração de dados [Santos et al., 2008]. Basicamente, a MCF extrai conjuntos de elementos que aparecem com frequência na base de dados, de acordo com uma frequência mínima estabelecida, ou suporte mínimo ($SupMin$).

Em outras palavras, um conjunto frequente é aquele que ocorre em, pelo menos, $SupMin\%$ dos registros de uma base de dados.

Algoritmo 3: BuscaLocal(s, I_{ILS})

```
1  s' ← s;
2  iterILS ← 0;
3  enquanto iterILS < I_ILS faça
4  |  s ← RVND(s);
5  |  se f(s) < f(s') então
6  |  |  s' ← s;
7  |  |  iterILS ← 0;
8  |  s ← Perturbacao(s');
9  |  iterILS ← iterILS + 1;
10 retorna s'
```

Em geral, a incorporação dessa técnica em heurísticas é dividida em duas fases: a primeira é responsável por armazenar soluções de alta qualidade em um conjunto, denominado de conjunto elite, e a outra utiliza padrões encontrados nesse conjunto, que servirão como guia na exploração eficiente do espaço de busca do problema. Nesse contexto, esses padrões são obtidos por meio da aplicação da técnica MCF nas soluções (registros) do conjunto elite (base de dados). Para esse trabalho, o algoritmo FPMax*, que é uma implementação estado-da-arte na extração de conjuntos frequentes maximais, foi adotado na mineração desses padrões [Grahne e Zhu, 2003]. Um conjunto frequente é considerado maximal quando não existe nenhum superconjunto que o contenha.

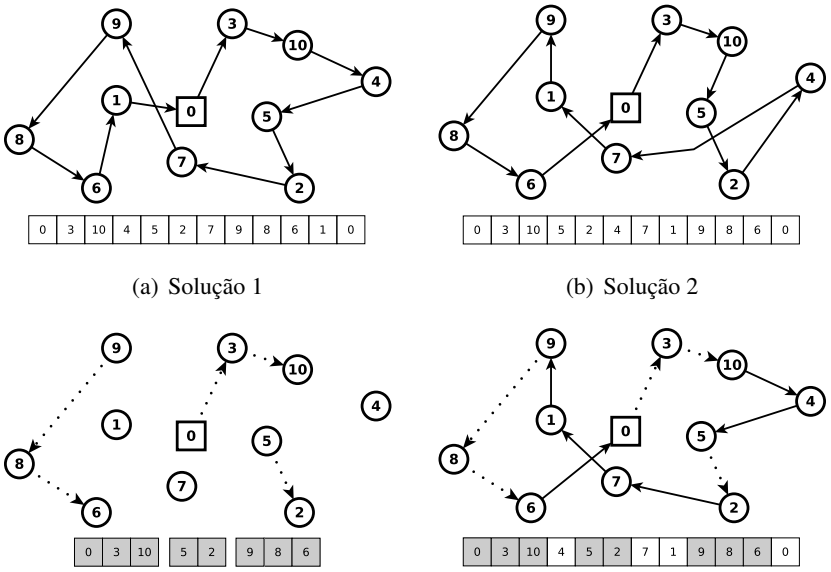
No entanto, a identificação desses conjuntos não leva em consideração a ordem dos seus elementos e, portanto, a aplicação direta dessa técnica nas soluções do conjunto elite não pode ser realizada, pois as sequências de visitação de clientes, presente nas soluções, serão perdidas. Para solucionar esse novo problema, uma abordagem proposta por [Guerine et al., 2016], que realiza a mineração de arcos frequentes, foi adaptada nesse trabalho. Essa abordagem consiste em transformar cada par de clientes consecutivos (i e j) de uma solução para o seu respectivo arco ($a_{i \to j}$) e, assim, por meio de um mapeamento de arcos, a extração de padrões baseada em arcos pela técnica MCF mantém as ordens de visitação de clientes das soluções.

A Figura 2 ilustra o uso de arcos frequentes minerados. Considere um conjunto elite com duas soluções, representadas pelas Figuras 2(a) e 2(b), as quais são submetidas ao procedimento de mineração de dados com suporte igual a dois. Logo, somente os arcos que estão presentes nas duas soluções serão considerados frequentes (ver Figura 2(c)). Em seguida, esses arcos são usados na construção de soluções iniciais em um método construtivo adaptado, que será explicado posteriormente. Portanto, por meio desse método construtivo adaptado, soluções iniciais são geradas a partir de componentes frequentes de soluções de alta qualidade, como exemplificado na Figura 2(d).

A versão híbrida do GILS-RVND com mineração de dados, denominada DM-GILS-RVND, foi desenvolvida com base na hibridização que deu origem ao DM-GRASP [Santos et al., 2008]. Na versão híbrida desenvolvida (DM-GILS-RVND), o conjunto total de iterações do algoritmo original (GILS-RVND) é executado em duas partes com metade do número total de iterações em cada e possui, entre elas, o procedimento de mineração de dados. A primeira parte da versão híbrida é idêntica ao algoritmo original, exceto pela introdução do armazenamento de soluções no conjunto elite, onde uma solução só é inserida se essa é melhor que a pior solução do conjunto e diferente de todas as soluções do conjunto ou se o conjunto elite não estiver completo. Assim como a primeira, a segunda parte é idêntica ao algoritmo original, exceto pelo fato de o método construtivo ser substituído por uma abordagem de construção híbrida.

O Algoritmo 4 descreve em detalhes a proposta deste trabalho. A primeira parte (linhas 3-7) permanece igual ao algoritmo original, salvo pela etapa de busca local, a qual é diretamente derivada da busca local original. Essa etapa é idêntica ao Algoritmo 3, exceto pela introdução da função de atualização do conjunto elite Ω de tamanho d após o RVND, que se encarrega de tentar

inserir a solução s retornada pelo RVND em Ω. Logo após a primeira parte, o conjunto elite é submetido ao FPmax* (linha 8). Neste procedimento, são requeridos Ω, d, $SupMin$ e $MaxP$. O último parâmetro se refere a um valor inteiro que limita o número de padrões a serem minerados. Após esse procedimento, o minerador retorna uma lista de padrões P de tamanho $MaxP$ ordenados de forma crescente pela quantidade de arcos minerados por padrão. Na segunda parte (linhas 9-13), toda a estrutura do algoritmo original é mantida, exceto pela substituição do método construtivo original pelo método construtivo híbrido, onde, para cada iteração, um padrão $p \in P$ e um valor aleatório α do conjunto R são utilizados (linha 10).



(a) Solução 1

(b) Solução 2

(c) Conjunto frequente de arcos minerado

(d) Solução construída com o conjunto frequente

Figura 2: Processo de mineração de conjuntos frequentes de arcos

Algoritmo 4: DM-GILS-RVND($I_{Max}, I_{ILS}, R, SupMin, MaxP, d$)

1 $f(s^*) \leftarrow \infty$;
2 $\Omega \leftarrow \emptyset$;
3 **para** $i \leftarrow 1, ..., I_{Max}/2$ **faça**
4 $\quad s \leftarrow MetodoConstrutivo(\alpha \in R)$;
5 $\quad s \leftarrow BuscaLocal(s, I_{ILS}, \Omega, d)$;
6 $\quad$ **se** $f(s) < f(s^*)$ **então**
7 $\quad\quad s^* \leftarrow s$;
8 $P \leftarrow MineraPadroes(\Omega, d, SupMin, MaxP)$;
9 **para** $i \leftarrow 1, ..., I_{Max}/2$ **faça**
10 $\quad s \leftarrow MetodoConstrutivoHibrido(\alpha \in R, p \in P)$;
11 $\quad s \leftarrow BuscaLocal(s, I_{ILS})$;
12 $\quad$ **se** $f(s) < f(s^*)$ **então**
13 $\quad\quad s^* \leftarrow s$;
14 **retorna** s^*

No método construtivo híbrido descrito no Algoritmo 5, os elementos dos padrões minerados são aproveitados na construção de soluções iniciais. Inicialmente, após o depósito ser inserido na solução s (linha 1), as listas de arcos consecutivos (LAC) encontradas em p são geradas (linha 2), onde cada lista será utilizada de forma integral quando o primeiro cliente da lista for selecionado. A lista de clientes (LC), baseada em LAC, é gerada com os clientes que não pertencem a nenhuma lista de LAC e com aqueles que estão na primeira posição de cada lista de LAC (linha 3).

Caso o depósito seja o primeiro elemento de alguma lista de LAC, então essa lista é inserida em s e removida de LAC (linhas 4-6). No laço, os clientes de LC são ordenados de forma crescente de acordo com o tempo de viagem de cada cliente e r (último cliente inserido em s) (linha 9). A seguir, LRC é preenchida com os $\alpha\%$ melhores clientes de LC, para que um cliente $c \in LRC$ seja selecionado de forma aleatória (linhas 10-11). Se c é o primeiro cliente de alguma lista de LAC, então essa lista é inserida em s e removida de LAC (linhas 12-14). Caso contrário, somente c será inserido em s (linhas 15-16). Em seguida, c é removido de LC e o último cliente inserido em s é associado a r como referência na ordenação da próxima iteração (linhas 17-18). Este laço termina quando LC estiver vazia e, então, s é retornada (linha 19).

Algoritmo 5: MetodoConstrutivoHibrido(α, p)

```
1  s ← {0};
2  LAC ← GerarLAC(p);
3  LC ← GerarLC(LAC);
4  se {0 é o primeiro cliente de lac | lac ⊂ LAC} então
5  |  s ← s ∪ lac;
6  |  LAC ← LAC − lac;
7  r ← SelecionarUltimoCliente(s);
8  enquanto LC ≠ ∅ faça
9  |  LC ← OrdenarLC(LC, r);
10 |  LRC ← PreencherLRC(LC, α);
11 |  c ← SelecionarCliente(LRC);
12 |  se {c é o primeiro cliente de lac | lac ⊂ LAC} então
13 |  |  s ← s ∪ lac;
14 |  |  LAC ← LAC − lac;
15 |  senão
16 |  |  s ← s ∪ {c};
17 |  LC ← LC − {c};
18 |  r ← SelecionarUltimoCliente(s);
19 retorna s
```

5. Experimentos Computacionais

O código-fonte do GILS-RVND desenvolvido por [Silva et al., 2012] foi cedido pelos autores para a incorporação da técnica de mineração de dados. Ambas heurísticas foram codificadas em C++ (g++ 4.4.3) e executadas em um Intel®Core™ i7 3,40GHz, com 16GB de memória RAM e GNU/Linux Ubuntu 14.04 de 64 *bits*.

Em ambas abordagens, todos os experimentos foram realizados sob o mesmo conjunto de parâmetros definidos por [Silva et al., 2012], os quais são $I_{Max} = 10$, $I_{ILS} = min\{100; n\}$ e $R = \{0,00; 0,01; ...; 0,25\}$, onde n representa o número de clientes da instância. Para a versão hibridizada com mineração de dados, os parâmetros $d = 10$, $SupMin = 7$ e $MaxP = 5$ foram utilizados com base em testes preliminares. Estes experimentos foram realizados em todas as 150 instâncias da versão de caminho hamiltoniano do PML em [Silva et al., 2012], as quais são divididas em: sete grupos de 20 instâncias de mesma dimensão com 10, 20, 50, 100, 200, 500 e 1000 clientes (Grupos 1 a 7) e um grupo de dez instâncias variando entre 70 e 532 clientes[1] (Grupo 8).

Para realizar uma avaliação justa e rigorosa, os experimentos envolvendo o GILS-RVND foram completamente reproduzidos, utilizando a mesma versão de compilador, as mesmas sementes de números aleatórios e dez execuções por instância, características definidas por [Silva et al., 2012]. Posteriormente, o DM-GILS-RVND também foi submetido a essas mesmas condições a fim de apresentar uma análise comparativa justa entre as duas heurísticas testadas.

Nas Tabelas 1 e 2, os resultados dos grupos das instâncias de 500 e 1000 clientes são detalhados, visto que esses grupos são os maiores e mais difíceis da literatura. Nessas tabelas

[1]Instâncias do TSPlib: st70, rat99, kroD100, rat195, lin105, pr107, pr226, lin318, pr439, att532. Disponíveis em: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/

são reportados os resultados de melhor solução obtida, solução média e tempo médio em segundos de ambas heurísticas sobre os Grupos 6 e 7, com os valores em negrito indicando o melhor resultado obtido entre as duas heurísticas e custos sublinhados mostrando os melhores valores de custo de solução para a literatura. Para os resultados de melhor solução e solução média, também são apresentadas as diferenças percentuais desses valores em relação à melhor solução conhecida (*BKS*), que são dadas por $\Delta_{\%} = \frac{100\times(Valor-BKS)}{BKS}$, onde $Valor$ denota o valor de custo (melhor ou médio) de solução reportado. Os valores de *BKS* reportados estão associados aos símbolos $\star$ e ‡ que indicam a origem, [Silva et al., 2012] e [Rios, 2016], respectivamente. Na última coluna são apresentadas as diferenças percentuais dos tempos médios entre as duas heurísticas, que são calculadas por $\Delta_{\%} = \frac{100\times(TempoDMGILS-TempoGILS)}{TempoGILS}$. Além disso, o nome de cada instância foi reduzido para o seu respectivo sufixo, as quais estão no formato TRP-S500-Ri e TRP-S1000-Ri para os grupos de 500 e de 1000 clientes, respectivamente. Por último, todas as médias gerais das diferenças percentuais são apresentadas na última linha.

Instância	BKS	GILS-RVND					DM-GILS-RVND					$\Delta_{\%}$
		Melhor Solução	$\Delta_{\%}$ Melhor	Solução Média	$\Delta_{\%}$ Média	Tempo Médio	Melhor Solução	$\Delta_{\%}$ Melhor	Solução Média	$\Delta_{\%}$ Média	Tempo Médio	Tempo
R1	1841386*	**1841386**	0,000	1856018,7	0,795	830,85	**1841386**	0,000	**1852238,0**	0,589	**620,18**	-25,36
R2	1815664‡	**1816568**	0,050	1823196,9	0,415	724,17	1817288	0,089	**1821667,9**	0,331	**502,99**	-30,54
R3	1826855‡	1833044	0,339	1839254,2	0,679	761,86	**1831357**	0,246	**1838557,2**	0,641	**580,34**	-23,83
R4	1804894‡	1809266	0,242	**1815876,4**	0,608	810,63	**1808563**	0,203	1816765,0	0,658	**589,33**	-27,30
R5	1821250‡	1823975	0,150	1834031,7	0,702	734,32	**1823135**	0,104	**1831575,2**	0,567	**565,36**	-23,01
R6	1782731‡	1786620	0,218	1790912,4	0,459	796,28	**1785217**	0,139	**1789675,6**	0,390	**596,65**	-25,07
R7	1847999*	1847999	0,000	1857926,6	0,537	781,01	**1847089**	-0,049	**1854324,8**	0,342	**609,98**	-21,90
R8	1819636‡	1820846	0,066	**1829257,3**	0,529	769,33	**1820639**	0,055	1829831,8	0,560	**580,50**	-24,55
R9	1733819*	1733819	0,000	1737024,9	0,185	693,82	**1730561**	-0,188	**1736278,3**	0,142	**545,16**	-21,43
R10	1761174‡	1762741	0,089	**1767366,3**	0,352	784,64	**1762197**	0,058	1767912,7	0,383	**587,06**	-25,18
R11	1797881*	**1797881**	0,000	**1801467,9**	0,200	741,50	**1797881**	0,000	1802957,9	0,282	**536,77**	-27,61
R12	1774452*	**1774452**	0,000	1783847,1	0,529	766,23	**1774452**	0,000	**1781588,2**	0,402	**564,99**	-26,26
R13	1863905‡	1873699	0,525	1878049,4	0,759	797,54	**1867803**	0,209	**1875973,1**	0,647	**556,95**	-30,17
R14	1799171*	1799171	0,000	1805732,9	0,365	835,86	**1798130**	-0,058	**1803379,2**	0,234	**585,61**	-29,94
R15	1785263‡	1791145	0,329	1797532,9	0,687	800,69	**1790524**	0,295	**1795858,3**	0,593	**587,39**	-26,64
R16	1804392‡	1810188	0,321	1816484,0	0,670	761,93	**1808183**	0,210	**1814941,6**	0,585	**590,35**	-22,52
R17	1825748*	1825748	0,000	1834443,2	0,476	738,57	**1824674**	-0,059	**1832635,8**	0,377	**533,61**	-27,75
R18	1825615‡	**1826263**	0,035	1833323,7	0,422	780,37	**1826263**	0,035	**1831998,5**	0,350	**636,26**	-18,46
R19	1776855‡	1779248	0,135	**1782763,9**	0,333	773,39	**1774846**	-0,113	1785575,3	0,491	**553,04**	-28,49
R20	1820813*	1820813	0,000	1830483,3	0,531	726,50	**1820713**	-0,005	**1829740,0**	0,490	**544,20**	-25,09
Média			0,125		0,512			**0,059**		**0,453**		-25,59

Tabela 1: Resultados sobre as instâncias do Grupo 6

Os experimentos computacionais do Grupo 6, reportados na Tabela 1, apontam que o comportamento médio do DM-GILS-RVND foi superior ao GILS-RVND com base em 15 melhores resultados de solução média. A diferença percentual média da versão híbrida com mineração de dados em relação aos valores de BKS foi 0,453%. Já a da heurística-base foi 0,512%. Assim, os resultados da versão híbrida com mineração de dados se aproximam mais do BKS do que a heurística original. Em termos de melhor solução, o DM-GILS-RVND também superou o GILS-RVND em 15 instâncias, sendo seis dessas novos valores de custos para a literatura, e com diferença percentual de 0,059% contra 0,125%. Ainda sobre este resultado, houve quatro empates e somente uma vitória do GILS-RVND. O DM-GILS-RVND também se mostrou superior em relação ao tempo computacional médio, onde obteve médias de tempo melhores que a heurística original em todas as instâncias, sendo a abordagem proposta, em geral, 25,59% mais rápida que a versão original.

A Tabela 2, que reporta os experimentos computacionais do Grupo 7, apresenta novamente o DM-GILS-RVND com comportamento médio superior ao GILS-RVND com base em 15 melhores resultados de solução média. Neste caso, a versão híbrida com mineração de dados apresentou 0,461% de percentual médio em relação ao *BKS*, ao passo que a versão original apresentou 0,566%, indicando, mais uma vez, que a nova proposta apresentou resultados mais próximos dos valores de *BKS* que a heurística original. A heurística híbrida com mineração de dados melhorou 17 melhores soluções encontradas pela heurística original, sendo 13 desses novos valores de custos

para a literatura, e com diferença percentual em relação ao BKS de -0,042% contra 0,141%. Ainda, houve um empate e duas vitórias do GILS-RVND. Em termos de tempo computacional médio, a versão proposta foi mais eficiente que a heurística original em todos os casos, reduzindo o esforço computacional, em média, 31,01% neste grupo.

Instância	BKS	GILS-RVND					DM-GILS-RVND					
		Melhor Solução	$\Delta_{\%}$ Melhor	Solução Média	$\Delta_{\%}$ Média	Tempo Médio	Melhor Solução	$\Delta_{\%}$ Melhor	Solução Média	$\Delta_{\%}$ Média	Tempo Médio	$\Delta_{\%}$ Tempo
R1	5107395*	5107395	0,000	5133698,3	0,515	19889,15	**5099593**	-0,153	**5122436,1**	0,294	**14123,42**	-28,99
R2	5106161*	5106161	0,000	5127449,4	0,417	19218,17	**5084762**	-0,419	**5112406,6**	0,122	**13532,12**	-29,59
R3	5096977*	5096977	0,000	5113302,9	0,320	18798,18	**5089701**	-0,143	**5111390,5**	0,283	**12553,89**	-33,22
R4	5112465‡	5118006	0,108	**5141392,6**	0,566	18493,11	**5110184**	-0,045	5142219,9	0,582	**12404,32**	-32,92
R5	5097991‡	5103894	0,116	5122660,7	0,484	18906,29	**5085136**	-0,252	**5116167,7**	0,357	**13124,07**	-30,58
R6	5109946‡	5115816	0,115	5143087,1	0,649	19143,87	**5102671**	-0,142	**5132201,1**	0,436	**13862,07**	-27,59
R7	4995703‡	5021383	0,514	5032722,0	0,741	17681,02	**4984950**	-0,215	**5014796,3**	0,382	**13154,44**	-25,60
R8	5109325*	**5109325**	0,000	5132722,6	0,458	18065,23	**5109325**	0,000	**5129942,9**	0,404	**12781,96**	-29,25
R9	5046566‡	5052599	0,120	5073245,3	0,529	17979,62	**5045262**	-0,026	**5070469,7**	0,474	**11758,44**	-34,60
R10	5060019‡	5078191	0,359	5093592,6	0,664	17596,33	**5070109**	0,199	**5089275,6**	0,578	**12274,40**	-30,24
R11	5031455‡	**5041913**	0,208	**5066161,5**	0,690	18307,69	5052459	0,417	5066612,8	0,699	**11826,31**	-35,40
R12	5029792*	**5029792**	0,000	5051235,2	0,426	19149,54	5030837	0,021	**5043866,1**	0,280	**12940,67**	-32,42
R13	5102520*	5102520	0,000	5131437,5	0,567	19604,18	**5098034**	-0,088	**5123032,4**	0,402	**13895,03**	-29,12
R14	5092861‡	5099433	0,129	5118980,6	0,513	18974,58	**5089565**	-0,065	**5116989,7**	0,474	**13171,29**	-30,58
R15	5131013‡	5142470	0,223	5174493,2	0,847	18889,53	**5123240**	-0,151	**5166046,9**	0,683	**13687,70**	-27,54
R16	5064094‡	5073972	0,195	**5090280,5**	0,517	18206,27	**5070422**	0,125	5091420,9	0,540	**11962,50**	-34,29
R17	5052283‡	5071485	0,380	5084450,4	0,637	18571,62	**5065952**	0,271	**5082717,2**	0,602	**11847,47**	-36,21
R18	5005789‡	5017589	0,236	**5037094,0**	0,625	19745,37	**5003047**	-0,055	5038269,6	0,649	**12966,44**	-34,33
R19	5064873‡	5076800	0,235	5097167,6	0,638	19790,68	**5065743**	0,017	**5087261,2**	0,442	**13365,27**	-32,47
R20	4977262*	4977262	0,000	**5002920,6**	0,516	18715,65	**4970735**	-0,131	5003599,3	0,529	**13150,60**	-29,73
Média			0,147		0,566			**-0,042**		**0,461**		-31,23

Tabela 2: Resultados sobre as instâncias do Grupo 7

Por último, em função dos valores de custo de solução de ambas heurísticas nos Grupos 6 e 7 se aproximarem de uma amostra com uma distribuição normal, verificados por meio do teste de normalidade de Shapiro-Wilk, o teste estatístico t Student unicaudal pareado com nível de significância igual a 5% foi aplicado. Em quatro instâncias do Grupo 6, a diferença entre as médias apresentou significância estatística a favor do DM-GILS-RVND e nenhuma para o GILS-RVND. Para o Grupo 7, em sete instâncias, a diferença entre as médias apresentou significância estatística para o DM-GILS-RVND e nenhuma para o GILS-RVND.

A Tabela 3 reúne os resultados dos experimentos de todos os grupos de instâncias entre as duas heurísticas tratadas neste trabalho. Nesta tabela, as duas primeiras colunas identificam os grupos já definidos e suas respectivas dimensões. Nas duas colunas seguintes, os tempos médios de todos os grupos para cada heurística e, nas três colunas seguintes, as diferenças percentuais de melhor solução, solução média e tempo computacional médio em segundos entre as duas heurísticas, que foram calculados usando a formula $\Delta_{\%} = \frac{100 \times (ResultadoDMGILS - ResultadoGILS)}{ResultadoGILS}$.

Origem	Dimensão	GILS-RVND Tempo Médio	DM-GILS-RVND Tempo Médio	Melhor Solução ($\Delta_{\%}$)	Solução Média ($\Delta_{\%}$)	Tempo Médio ($\Delta_{\%}$)
Grupo 1	10	0,010	0,010	0,000	0,000	0,00
Grupo 2	20	0,020	0,020	0,000	0,000	0,00
Grupo 3	50	0,350	0,340	0,000	0,000	-2,86
Grupo 4	100	4,675	4,180	0,000	0,000	-10,58
Grupo 5	200	43,372	35,475	0,000	0,000	-18,21
Grupo 6	500	770,465	573,335	-0,066	-0,059	-25,59
Grupo 7	1000	18786,305	12919,120	-0,188	-0,105	-31,23
Grupo 8	70-532	142,135	111,890	0,008	0,002	-21,28

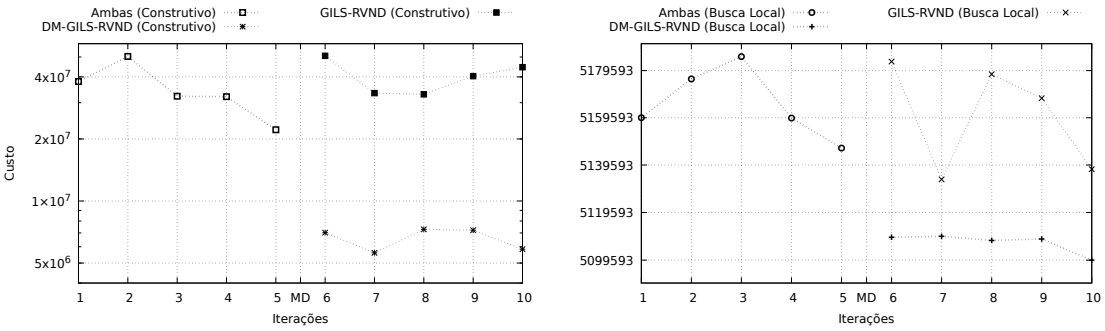
Tabela 3: Oito grupos de instâncias executados utilizando as duas heurísticas testadas

Analisando a Tabela 3, o comportamento médio do DM-GILS-RVND, medido pela diferença percentual de solução média, foi superior ao GILS-RVND nos Grupos 6 e 7, empatando nos

Grupos 1, 2, 3, 4 e 5 e perdendo no Grupo 8. É importante salientar que, a vitória do GILS-RVND apresentou uma ligeira vantagem, sendo essa heurística muito mais lenta que a versão proposta. Outro aspecto importante é a tendência de melhoria do DM-GILS-RVND, tanto em qualidade de solução, quanto em tempo computacional médio, à medida que a dimensão da instância aumenta e o problema se torna mais difícil.

5.1. Análises de Comportamento das Heurísticas

A fim de analisar e entender o impacto do uso da mineração de dados, os dois algoritmos foram executados usando-se a instância TRP-S1000-R1 com a mesma semente inicial e os valores de custo gerados pela etapa construtiva e de busca local de cada iteração são mostrados na Figura 3.



(a) Soluções geradas pelos métodos construtivos (b) Soluções encontradas pela etapa de busca local

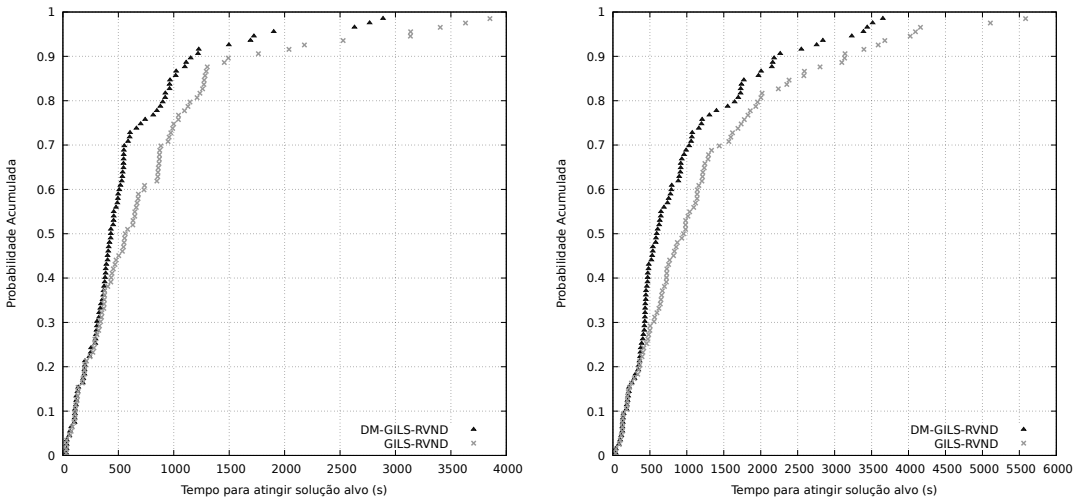
Figura 3: Gráficos Custo *versus* Iteração - TRP-S1000-R1

As duas heurísticas possuem comportamento idêntico nas cinco primeiras iterações, antes da execução do processo de mineração. Na Figura 3(a), logo após a etapa de mineração de dados (MD no eixo horizontal), o emprego dos arcos minerados reduz drasticamente os valores de custo das soluções iniciais geradas pelo método construtivo híbrido em comparação com os valores de custo das soluções iniciais geradas pelo método construtivo original. Neste caso específico, o número médio de arcos frequentes minerados é de 931 por padrão minerado. A Figura 3(b) considera apenas os custos das soluções encontradas pela etapa de busca local. Nessa figura, é possível notar que o DM-GILS-RVND, após a execução do processo de mineração de dados, possui valores de custo de solução menores e, portanto, mais próximos dos valores de custo das melhores soluções dessa instância em relação aos valores de custos das soluções apresentadas pelo GILS-RVND.

Uma maneira de avaliar comportamentos de algoritmos com componentes aleatórias é por meio de gráficos *Time-to-Target Plots* (*TTTPlots*) [Aiex et al., 2007]. Esses gráficos apresentam, no eixo das ordenadas, a probabilidade acumulada do algoritmo encontrar uma solução pelo menos tão boa quanto um dado alvo (valor de custo) dentro de um determinado tempo, representado no eixo das abscissas. A Figura 4 apresenta os gráficos *TTTPlots* para as duas heurísticas, utilizando a instância TRP-S500-R5, e dois alvos distintos: 1834031 (mais fácil) e 1831575 (mais difícil). Nesta avaliação, as heurísticas foram executadas 100 vezes com o mesmo conjunto de 100 sementes iniciais distintas.

Na Figura 4(a), o comportamento do DM-GILS-RVND, em 95% das suas execuções, atinge o alvo definido em aproximadamente 1750 segundos, enquanto, para essa mesma probabilidade, o GILS-RVND necessita de um pouco mais de 3100 segundos. Verifica-se também que o DM-GILS-RVND atinge o alvo em 60% das execuções em torno de 500 segundos, enquanto, para este mesmo tempo, a probabilidade para o GILS-RVND alcançar este alvo cai para 45%. No segundo gráfico, que apresenta um alvo mais difícil que o anterior, a Figura 4(b) mostra que a eficiência do DM-GILS-RVND em atingir o alvo é de 100% em cerca de 3600 segundos, ao passo que, para a heurística original, o tempo necessário para alcançar esse mesmo alvo em todas as execuções foi um pouco mais de 5500 segundos. Além disso, a versão híbrida com mineração de

dados atinge o alvo em 90% das execuções em menos de 2500 segundos, enquanto, para a versão original, o tempo necessário aumenta para um pouco mais de 3200 segundos.



(a) TTTplot com alvo 1834031 (b) TTTplot com alvo 1831575

Figura 4: Gráficos TTTplots para alvos da instância TRP-S500-R5

6. Conclusões e Trabalhos Futuros

A heurística híbrida com mineração de dados introduzida neste trabalho apresentou-se como uma abordagem bem-sucedida para o PML. Esse comportamento foi evidenciado por meio dos experimentos computacionais realizados em 150 instâncias. Não somente a eficiência da nova abordagem proposta em encontrar melhores soluções foi superior à da heurística original, mas também houve uma redução perceptível do tempo computacional, especialmente para instâncias de 500 e 1000 clientes. Além disso, a heurística híbrida com mineração de dados foi capaz de superar 19 melhores valores de custo de solução existentes na literatura. Portanto, a hibridização proposta neste trabalho demonstrou ganhos significativos de qualidade de solução e de tempo de execução, principalmente nas maiores e mais difíceis instâncias do PML da literatura.

Em trabalhos futuros, pretende-se aperfeiçoar a hibridização de mineração de dados no GILS-RVND. Para tanto, pretende-se executar a mineração de dados como no modelo *Multi* DM-GRASP (MDM-GRASP) [Santos et al., 2008], realizando minerações sucessivas de padrões de modo adaptativo ao longo da execução do algoritmo.

Referências

Abeledo, H., Fukasawa, R., Pessoa, A., e Uchoa, E. (2013). The Time Dependent Traveling Salesman Problem: Polyhedra and Algorithm. *Mathematical Programming Computation*, 5:27–55.

Aiex, R. M., Resende, M. G. C., e Ribeiro, C. C. (2007). TTT plots: A Perl Program to Create Time-to-target Plots. *Optimization Letters*, 1:355–366.

Barbalho, H., Rosseti, I., Martins, S. L., e Plastino, A. (2013). A Hybrid Data Mining GRASP with Path-Relinking. *Computers and Operations Research*, 40:3159–3173.

Bianco, L., Mingozzi, A., e Ricciardelli, S. (1993). The Traveling Salesman Problem with Cumulative Costs. *Networks*, 23:81–91.

Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, B., Raghavan, P., e Sudan, M. (1994). The Minimum Latency Problem. In *Proceedings of the 26th annual ACM symposium on Theory of computing*, p. 163–171, Montreal, Canadá.

Chaudhuri, K., Godfrey, B., Rao, S., e Talwar, K. (2003). Paths, Trees, and Minimum Latency Tours. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2003*, p. 36–45, Cambridge, Estados Unidos.

Feo, T. A. e Resende, M. G. C. (1995). Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, 6:109–133.

Fischetti, M., Laporte, G., e Martello, S. (1993). The Delivery Man Problem and Cumulative Matroids. *Operations Research*, 41:1055–1064.

Grahne, G. e Zhu, J. (2003). Efficiently Using Prefix-trees in Mining Frequent Itemsets. In *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, p. 236–245, Melbourne, Estados Unidos.

Guerine, M., Rosseti, I., e Plastino, A. (2016). Extending the Hybridization of Metaheuristics with Data Mining: Dealing with Sequences. *Intelligent Data Analysis*, 20:1133–1156.

Han, J., Kamber, M., e Pei, J. (2011). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, Estados Unidos, 3rd edition.

Martin, O., Otto, S. W., e Felten, E. W. (1991). Large-step Markov Chains for the Traveling Salesman Problem. *Complex Systems*, 5:299–326.

Martins, D., Vianna, G., Rosseti, I., Martins, S., e Plastino, A. (2014). Making a State-of-the-art Heuristic Faster with Data Mining. *Aceito para publicação em: Annals of Operations Research*.

Mladenović, N. e Hansen, P. (1997). Variable Neighborhood Search. *Computers and Operations Research*, 24:1097–1100.

Moshref-Javadi, M. e Lee, S. (2013). A Taxonomy to the Class of Minimum Latency Problems. In *Proceedings of the 2013 Industrial and Systems Engineering Research Conference*, p. 3896–3905, San Juan, Porto Rico.

Plastino, A., Fonseca, E. R., Fuchshuber, R., Freitas, A. A., Luis, M., e Martins, S. L. (2011). A Hybrid Data Mining Metaheuristic for the p-Median Problem. *Stat. Anal. Data Min.*, 4:313–335.

Ribeiro, M. H., Plastino, A., e Martins, S. L. (2006). Hybridization of GRASP Metaheuristic with Data Mining Techniques. *Journal of Mathematical Modelling and Algorithms*, 5:23–41.

Rios, E. (2016). *Exploração de Estratégias de Busca Local em Ambientes CPU / GPU*. Tese de Doutorado, Programa de Pós-graduação em Computação, Instituto de Computação, Universidade Federal Fluminense.

Santos, L. F. M., Martins, S. L., e Plastino, A. (2008). Applications of the DM-GRASP Heuristic: A Survey. *International Transactions in Operational Research*, 15:387–416.

Silva, M. M., Subramanian, A., Vidal, T., e Ochi, L. S. (2012). A Simple and Effective Metaheuristic for the Minimum Latency Problem. *European Journal of Operational Research*, 221:513–520.

Talbi, E.-G. (2002). A Taxonomy of Hybrid Metaheuristics. *Journal of Heuristics*, 8:541–564.

Tsitsiklis, J. N. (1992). Special Cases of Traveling Salesman and Repairman Problems with Time Windows. *Networks*, 22:263–282.