O Algoritmo VNSP: Uma Solução Paralela para o Problema de Escalonamento de Motorista de Transporte Público

Mariane Affonso Medeiros
Universidade Estadual de Maringá
Av. Colombo, 5790 - UEM - Bloco C56
marianeaffonsomedeiros@gmail.com

Ademir Aparecido Constantino
Universidade Estadual de Maringá
Av. Colombo, 5790 - UEM - Bloco C56
ademir@din.uem.br

Anderson Faustino
Universidade Estadual de Maringá
Av. Colombo, 5790 - UEM - Bloco C56
anderson@din.uem.br

RESUMO

O Problema de Escalonamento de Motoristas de Transporte Público consiste em construir escalas de trabalho para motoristas, que devem respeitar restrições trabalhistas como quantidade máxima de horas trabalhadas, tempo mínimo de intervalo, entre outras. Além disto, as escalas devem minimizar os custos operacionais. Este trabalho propõe um algoritmo paralelo baseado na meta-heurística *Variable Neighborhood Search* para resolução de tal problema. O algoritmo implementado possui 36 diferentes buscas locais paralelas, que utilizam método exato para alterar uma solução. Os resultados obtidos indicam que a paralelização gera diversos benefícios, tais como redução de tempo de processamento e a possibilidade de análise de quais buscas locais são mais eficazes.

PALAVRAS CHAVE. Problema de Escalonamento de Motoristas, *Variable Neighborhood Search*, Paralelismo.

ABSTRACT

The Bus Driver Scheduling Problem consists of construct work schedules for drivers, that must respect some work restrictions like maximum of work hours and minimum time of break. The schedule needs to minimize the operational cost too. This article, propose a parallel algorithm based on Variable Neighborhood Search to solve this problem. The proposed algorithm, use 36 different local searches, that use exact methods to operate over the solution. The results obtained by the algorithm, show that parallelization offers many benefits, like reduction of processing time and possible analyzes of local search that are more effective.

KEYWORDS. Bus Driver Scheduling Problem, Variable Neighborhood Search, Parallelism.

1. Introdução

Veículos e motoristas são os principais recursos de uma companhia de transporte público, portanto a forma como são alocados tem impacto direto na qualidade e no custo do serviço (Silva e da Cunha [2010]). Sendo assim, a boa utilização destes recursos pode resultar na redução de custos, beneficiar funcionários e proporcionar jornadas de trabalho menos exaustivas. Neste contexto, o Problema de Escalonamento de Motorista de Transporte Público (PEM) consiste na geração de escalas de trabalho para um conjunto de motoristas, de forma a respeitar restrições e otimizar a função objetivo (Constantino et al. [2017]).

Considerando o tempo computacional que alguns problemas de otimização exigem, diversas abordagens para paralelizar meta-heurísticas foram propostas (Garca-Lopez et al. [2002]; Knausz [2008]). Devido a complexidade do PEM, a utilização de meta-heurísticas paralelas para sua resolução é uma área de interese dos pesquisadores.

A proposta desse artigo é apresentar um algoritmo paralelo, VNSP, o qual é baseado na meta-heurística *Variable Neighborhood Search* (VNS) e tem por objetivo resolver o PEM. O diferencial do VNSP está nas 36 buscas locais que são executadas em paralelo. Os resultados obtidos indicam que o VNSP é uma proposta promissora para resolução do PEM, devido a sua considerável melhora da solução inicial. Além disto, por meio dos experimentos realizados foi possível notar que nem todas as buscas locais utilizadas são promissoras na melhora da solução.

O restante deste artigo está organizado como segue. A Seção 2 apresenta alguns trabalhos relacionados encontrados na literatura. A Seção 3 descreve o algoritmo paralelo proposto. A Seção 4 apresenta um estudo experimental e uma discussão sobre os resultados obtidos. Finalmente, a Seção 5 traz as considerações finais.

2. Trabalhos Relacionados

Silva et al. [2004] formula o PEM como um problema de partição de conjuntos utilizando o método simplex para resolvê-lo. Yunes et al. [2005] utiliza uma abordagem baseada no problema de cobertura de conjuntos e geração de colunas com o objetivo de minimizar o número de motoristas. Uma limitação destes trabalhos está no fato de que as proposta baseadas em métodos exatos utilizam apenas instâncias de pequeno porte, as quais não ultrapassam 250 viagens. Para instâncias de grande porte a utilização de meta-heurísticas tem sido uma alternativa viável.

Na literatura é possível encontrar propostas que aplicam diferentes meta-heurísticas ao PEM. Marinho et al. [2004] utiliza a meta-heurística Busca Tabu (BT), enquanto Dos Santos et al. [2016] propõe a resolução do PEM por meio da meta-heurística VNS. Tais abordagens, possuem a característica de encontrarem soluções para instâncias de grande porte, as quais chegam a conter 2313 viagens.

No contexto de algoritmos paralelos baseado na meta-heurística VNS, se destacam os trabalhos de Garca-Lopez et al. [2002] e Sevkli e Aydin [2007]. Garca-Lopez et al. [2002] propôs três estratégias distintas para solucionar o problema *p-medianas*. A primeira estratégia foi paralelizar apenas a busca local. A segunda, envolveu execuções independentes de vários procedimentos sequenciais do VNS. Por fim, a terceira estratégia utilizou a abordagem mestre-escravo, no qual o mestre executa o VNS e os escravos, executam as etapas de *shake* e busca local. Sevkli e Aydin [2007] propuseram um VNS paralelo para solução do problema *Job Shop Scheduling*. Tal trabalho considera quatro estratégias distintas: (1) uma estratégia cooperativa sincronizada (Garca-Lopez et al. [2002]); (2) um método assíncrono com coordenação centralizada (Crainic et al. [2004]); (3) uma estratégia não centralizada utilizando a topologia de anel unidirecional; e (4) uma estratégia não centralizada utilizando topologia de anel bidirecional.

Como pode ser observado nesta seção, o presente trabalho se difere dos trabalhos de Marinho et al. [2004] e Dos Santos et al. [2016] pelo fato destes não utilizarem uma estratégia paralela para resolução do PEM. Por outro lado, o presente trabalho se aproxima das estratégias utilizadas por Garca-Lopez et al. [2002] e Sevkli e Aydin [2007] pelo uso da estratégias de paralelismo da meta-heurística VNS. É importante observar que a segunda abordagem proposta por Sevkli e Aydin

[2007] não pode ser considerada uma estratégia paralela, pois a abordagem simplesmente utiliza várias instâncias sequenciais do VNS e coleta as melhores soluções. No entanto, a primeira estratégia (Garca-Lopez et al. [2002]) é de fato uma estratégia paralela. Já a estratégia aqui proposta utiliza diferentes buscas locais de forma paralela e não apenas paraleliza uma busca local como foi feito em Garca-Lopez et al. [2002].

É possível observar que no trabalho de Sevkli e Aydin [2007] existe cooperação entre os componentes da proposta, visando uma sinergia que possibilite a obtenção de resultados satisfatórios para o problema em questão. O algoritmo aqui proposto, segue a mesma abordagem, o uso de cooperação entre os componentes paralelos. Contudo a diferença está na forma que a cooperação é feita. Enquanto que na abordagem de Sevkli e Aydin [2007] existe um fluxo determinado e fixo na comunicação, a abordagem aqui proposta relaxa essa questão, pelo fato de não haver uma sequência pré-estabelecida de comunicação entre os componentes. De fato, neste trabalho é proposto uma estratégia na qual a comunicação é realizada de forma a potencializar a convergência para uma melhor solução.

3. O Algoritmo VNSP

O PEM lida com jornadas de trabalho que devem obedecer a um conjunto de regras como descrito em Constantino et al. [2017]. A qualidade de uma escala de trabalho é avaliada por uma função objetivo. A função objetivo considerada neste trabalho corresponde a contabilizar o total de minutos pagos por uma jornada de trabalho. Uma jornada pode ser penalizada quando não obedece a alguma restrição imposta pelo problema. A função objetivo utilizada neste trabalho é descrita em Dos Santos et al. [2016].

Como mencionado anteriormente, a proposta deste artigo é apresentar um algoritmo paralelo baseado na meta-heurística VNS, chamado VNSP, que será aplicado ao PEM. Basicamente o VNSP possui duas fases. A primeira, fase construtiva, constrói a solução inicial. A segunda, fase melhorativa, modifica sucessivamente a solução inicial de forma a melhorar sua qualidade.

Com objetivo de obter sinergia entre os diferentes operadores de vizinhança propostos em Dos Santos et al. [2016], o VNSP utiliza uma abordagem paralela na qual diferentes buscas locais são instanciadas paralelamente, cada busca executando um operador diferente. Tais buscas locais cooperam entre si trocando informação sobre os resultados obtidos, com o objetivo de melhorar a qualidade da solução e alcançar uma boa solução global.

3.1. A Fase Construtiva

Inicialmente as tarefas são distribuídas em camadas, como mostra a Figura 1.

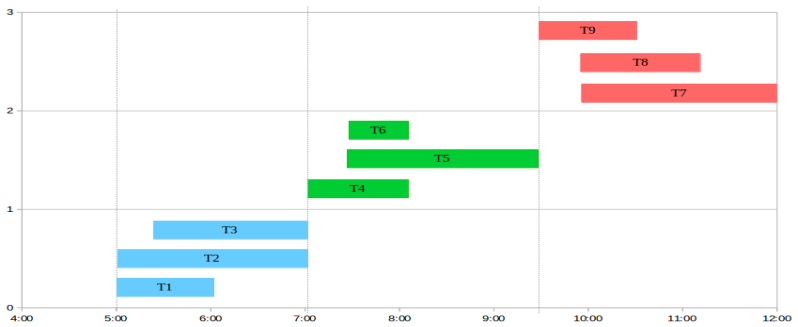

Figura 1: Divisão das camadas feitas pelo VNSP.

O processo de geração de camadas tem como entrada um conjunto T de tarefas, que são definidas por: hora de inicio e fim, um ponto de partida e parada. As tarefas devem ser reorganizadas em camadas que devem ser sequenciais entre si. Na Figura 1, existem 9 tarefas distribuídas ao longo das cinco da manhã até ao meio dia. As tarefas estão separadas em 3 camadas, cada uma contendo 3 tarefas. As tarefas da camada 1 podem ser sequenciadas com as da camada 2, que são sequenciadas

com as da camada 3. Nota-se que as tarefas contidas em cada camada, não podem ser sequenciadas entre si.

Para gerar a solução inicial, percorre-se a sequência de camadas C, para formar um conjunto de jornadas J. Para alcançar este objetivo é utilizado o Problema de Atribuição (PA) (Hillier e Lieberman [2013]), que tem como objetivo determinar entre um conjunto de possíveis atribuições de tarefas à jornadas, qual atribuição possui o menor custo associado. O PA é resolvido para cada camada da sequência C e deve verificar o custo de atribuir as tarefas da camada em questão a uma jornada existente em J. Caso haja inviabilidade de atribuição, cria-se uma nova jornada em J. Uma matriz de custos é a entrada para o PA, que armazena as possíveis combinações e seus custos. Cada elemento a_{ij} da matriz de custos $A_{|J|+|K|}$ (onde $|J|$ representa a quantidade de jornadas e $|K|$ a quantidade de tarefas), indica o custo de se atribuir a tarefa do índice j para a jornada i. Esta matriz é dividida nos seguintes blocos:

- Bloco 1: trata as atribuições de tarefas reais à jornadas reais. Caso haja viabilidade da atribuição, a posição $a_{i,j}$ assume o valor da função $g(i, j)$. Esta é a função objetivo da nova jornada gerada ao sequenciar a tarefa j à jornada i. Caso haja inviabilidade da atribuição, então a posição a_{ij} recebe ∞.

- Bloco 2: contém a atribuição de tarefas fictícias à jornadas reais. Estas tarefas representam possíveis tempos de parada na jornada do motorista. Caso haja inviabilidade de atribuição, a_{ij} recebe ∞.

- Bloco 3: trata a atribuição de tarefas reais à jornadas fictícias. Ou seja: $a_{ij} = c_{nj}$, onde c_{nj} é o valor dado ao se criar uma jornada não vazia.

- Bloco 4: é a atribuição de tarefas inexistentes para jornadas inexistentes; portanto, com custo zero.

A matriz de custos M é designada para o algoritmo que resolve o PA, para selecionar a solução com menor custo associado a qual será a entrada da fase melhorativa.

3.2. A Fase Melhorativa

A Fase Melhorativa é compreendida pela meta-heurística VNS (Hansen et al. [2008]) em conjunto com dois operadores que modificam a solução. Os operadores utilizados neste trabalho foram o Procedimento de cortes e recombinações (PCR) Constantino et al. [2014a] e k-swap. O PCR visa gerar soluções a partir de uma solução inicial S, garantindo que a solução gerada nunca será pior que a solução S (Constantino et al. [2014b]). O k-swap gera estruturas de vizinhanças, por meio de dois cortes na solução.

Ambos os operadores possuem duas formas de percorrer a vizinhança. A primeira, *foward*, percorre a vizinhança gerada na ordem crescente de tempo. A segunda, *backward*, percorre a vizinhança gerada na ordem decrescente de tempo. Os critérios adotados para a parada do algoritmo são: *first improvement*, *best improvement* e *continuous improvement* (Hansen et al. [2008], Sakayma et al. [2014]).

A Tabela 1 apresenta as possíveis combinações para percorrer a vizinhança com os critérios de parada dos operadores. A coluna *variações* indica se existe variação do operador. Como pode ser observado, k-swap possui 5 variações considerando de 1 até 5 cortes de distância na linha do tempo. A coluna *exploração de vizinhos* indica como a vizinhança será percorrida. Por fim, *critério de parada* indica qual dos critério será utilizado. Portanto, o VNSP possui 36 buscas locais diferentes, onde cada uma utiliza uma das variações dos operadores.

Tabela 1: Variações dos operadores de vizinhança utilizados por VNSP.

Nome	Variações	Exploração de Vizinhos	Critério Parada	Total
PCR	0	Forward (F) Backward (B)	First (FI) Best (BS) Continuous (CN)	6
Kswap	1,2,3,4,5	Forward (F) Backward (B)	First (FI) Best (BS) Continuous (CN)	30

A estratégia de paralelismo usada neste trabalho considera um VNS sequencial com a busca local paralelizada. Um processador se torna o mestre e executa as etapas básicas do VNS sequencial até a etapa da busca local. Nesta etapa, a busca local é executada em paralelo por processadores escravos.

O Algoritmo 1 apresenta o VNSP.

Algorithm 1 Algoritmo do VNSP

```
1:  function VNSP(S, k_max)
2:      k ← 1
3:      while k ≤ k_max do
4:          solucaoCompartilhada ← VAZIO
5:          S' ← Shake(S, k)
6:          for all operadorParalelo in operadoresParalelos do
7:              while melhora S' do
8:                  S'' ← operadorParalelo(S')
9:                  acessoMemoriaCompartilhada(S'')
10:                 if f(S'') < f(S') then
11:                     S' ← S''
12:         if f(solucaoCompartilhada) < f(S) then
13:             S ← solucaoCompartilhada
14:             k ← 1
15:         else
16:             k ← k + 1
        return S
```

O VNSP recebe como parâmetro uma solução S e um k_{max}, que é o número máximo de vizinhanças que serão exploradas (o VNSP utilizada 4 vizinhanças). A função $Shake(S, k)$ é a mesma utilizada em Dos Santos et al. [2016]. Na linha 4 é inicializada a solução que irá armazenar a melhor solução gerada pelas buscas locais.

A linha 6 do Algoritmo 1 mostra um laço de repetição, que constitui a busca local e é executado em paralelo. Ou seja, cada busca local é executada por uma instância paralela (*thread*). Essas execuções em paralelo geram soluções e compartilham as melhores soluções em uma memória compartilhada chamada $solucaoCompartilhada$.

As buscas locais são executadas em paralelo a cada iteração do VNS e depositam suas soluções na memória compartilhada, se a solução gerada pela busca for melhor que a solução existente na memória compartilhada, então a memória compartilhada é atualizada, caso contrário nada é feito. Portanto, VNSP aplica em paralelo 36 buscas locais no espaço de solução aumentando as chances de obter melhores soluções.

4. Resultados Experimentais

A avaliação experimental tem por objetivo avaliar o desempenho do algoritmo VNSP e investigar as seguintes questões: (1) qual o desempenho do algoritmo com relação a melhora da

solução inicial; (2) quais buscas locais são mais eficientes para a melhora da solução; e (3) qual o desempenho do VNSP quando comparado com outros algoritmos da literatura.

4.1. Ambiente Experimental

Platarforma de Hardware Os experimentos foram executados em um servidor com sistema operacional Ubuntu 16.04.3 LTS, 31 GB de memória e processador Intel(R) Xeon(R) CPU E7-4860 v2 @ 2.60GHz com 60 núcleos.

Platarforma de Software O algoritmo proposto foi desenvolvido na linguagem C e o compilador usado foi GNU Compiler Conection (GCC) 5.4. Para implementar o paralelismo foi utilizado a biblioteca POSIX Threads Library (Pthread).

Dados Experimentais O VNSP foi executado 30 vezes para cada instância e os resultados apresentados representam a média entre as 30 execuções.

Instâncias Foram utilizadas 6 instâncias do PEM para a realização dos experimentos. Destas 6 instâncias, quatro são sintéticas e duas reais (412 e 2313). A Tabela 2 apresenta o valor da função objetivo (F(x) onde x representa uma solução) e a quantidade de jornadas das soluções iniciais, as quais foram obtidas pela fase construtiva.

Tabela 2: Soluções iniciais.

Nome	Tarefas	F(x)	$\|J\|$
Inst130	130	11480	21
Inst412	412	38453	79
Inst761	761	67290	132
Inst1000	1000	87443	173
Inst1517	1517	133302	265
Inst2313	2313	194867	389

4.2. Visão Geral dos Resultados

A Tabela 3 apresenta os resultados obtidos pelo VNSP de função objetivo, número de jornadas e tempo de execução. Para cada um é apresentado o melhor (min), pior (max) e média (med) dos valores alcançados. Como pode ser observado, VNSP foi capaz de melhorar a qualidade da solução inicial para todas as instâncias.

Tabela 3: Resultados obtidos pela fase melhorativa.

Nome		F(x)	$\|J\|$	Tempo (seg)	Nome		F(x)	$\|J\|$	Tempo (seg)
Inst130	min	9240	21	38	Inst1000	min	69532	158	13903
	med	9439	21	42		med	71687	162	9138
	max	10120	23	80		max	78320	178	8935
Inst412	min	34320	78	496	Inst1517	min	106068	241	48313
	med	34628	78	486		med	110662	251	32592
	max	35640	81	425		max	117480	267	37211
Inst761	min	52800	120	6118	Inst2313	min	159720	363	89878
	med	53915	122	4168		med	163108	370	109031
	max	55440	126	2698		max	175560	399	110917

Os resultados indicam que existe uma relação direta entre a função objetivo e a quantidade de jornadas. Para cinco instâncias a diferença entre o valor obtido na melhor solução e na pior solução segue a mesma proporção, tanto para a função objetivo quanto para a quantidade de

jornadas. Isso indica que melhorar a função objetivo, significa diminuir a quantidade das jornadas. Nestas instâncias tal diferença variou entre 5% (Inst761) e 12,64% (Inst1000). A exceção ocorre para a instância Inst412, para a qual a variação no valor da melhor solução foi de 2,92%; enquanto a variação da quantidade de jornadas foi de 3,85%.

Um ponto interessante a ser observado é o fato de para apenas duas instâncias (Inst130 e Inst2313), o tempo de execução obtido para a pior solução ser menor do que o tempo de execução obtido pela execução que obteve a melhor solução. A tendência em um algoritmo heurístico é que as melhores soluções sejam encontradas em tempo de execução maior. Contudo, na maioria dos casos (4 instâncias) VNSP foi capaz de encontrar a melhor solução em um tempo até 55% menor do que aquele obtido pela execução que encontrou a pior solução. Isto demonstra que a estratégia de paralelismo utilizada por VNSP é uma boa estratégia para reduzir o tempo de resposta e ainda fornecer boas soluções.

4.3. Detalhando os Resultados

A Figura 2 apresenta a relação de melhora da solução inicial (diminuição do valor da função objetivo) em cada uma das 30 execuções.

É possível perceber que para as instâncias Inst130, Inst761, Inst1000, Inst1517 e Inst2313, o VNSP obteve uma melhora de até 26% para a solução inicial. Apenas para a instância Inst412 o percentual de melhoria foi menor do que 20% (13%). Um fato importante a ser observado é que a maioria das execuções foi capaz de melhorar a qualidade da solução inicial. De fato, existem mais execuções que produzem bons resultados do que execuções que produzem maus resultados. É importante perceber que existe uma variação na qualidade da solução final, entre cada execução. Contudo, a maioria das execuções é capaz de fornecer uma solução final acima de um limiar. Isto é um excelente resultado, indicando que VNSP é capaz de explorar de forma eficiente o espaço de busca. Outro ponto a ser destacado, é o fato de não ser necessário mais que 15 execução (na média) para que VNSP encontre a melhor solução possível. Isto também é um excelente resultado, indicando que não existe a necessidade de um tempo de resposta considerável para que VNSP encontre uma boa solução. De fato, os resultados demonstram que apenas para 2 instâncias (Inst412, Int761) foram necessárias 25 rodadas, para que a melhor solução fosse encontrada.

A Figura 3 apresenta a evolução da função objetivo e do número de jornadas, da melhor (azul) e da pior solução (vermelha) para as instâncias Inst412, Inst1000 e Inst2313 ao longo do tempo. A unidade de tempo apresentada representa iterações do VNSP.

Os resultados detalham o que foi mencionado anteriormente, a quantidade de jornadas esta diretamente relacionada ao valor da função objetivo. Quanto menor a quantidade de jornadas menor tende a ser o valor da função objetivo. A quantidade de jornadas tende a se estabilizar antes do valor da função objetivo, ou seja, embora a quantidade de jornadas se estabilize, VNSP ainda consegue fazer diferentes combinações que tendem a diminuir o valor da função objetivo. É possível perceber também, que o valor da função objetivo da melhor solução, tente a ser mais estável ao longo das iterações, enquanto a pior solução tende a ser mais instável, variando drasticamente de iteração para iteração. Nota-se que a melhor solução passa várias iterações com o valor da função objetivo igual, ou seja, passa muitas iterações sem ter melhora na solução. No entanto apesar de serem valores de função objetivo iguais, podem ser soluções diferentes, que em determinado momento do algoritmo acabam encontrando um valor de função objetivo melhor.

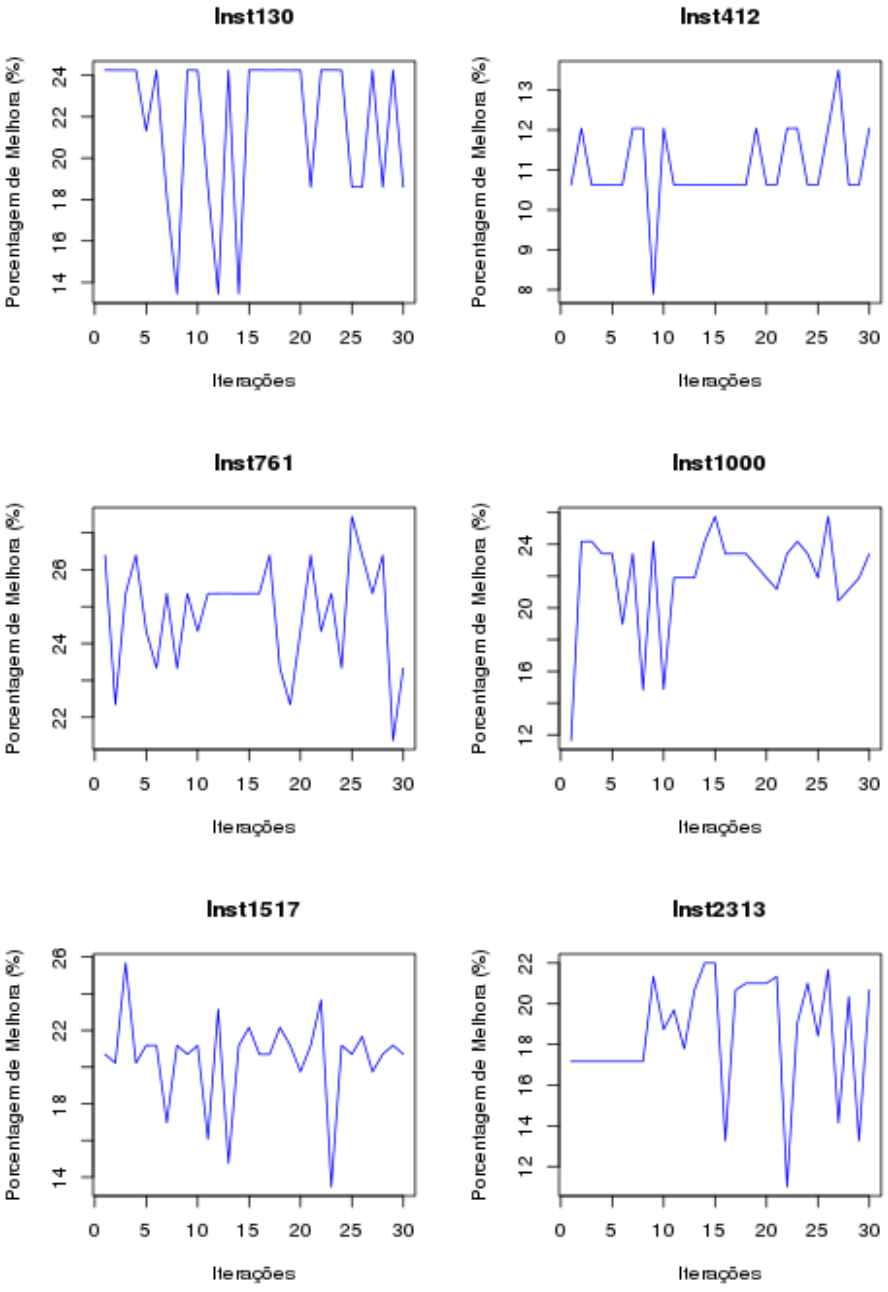

Figura 2: Melhoria da função objetivo obtida, em relação a solução inicial, para cada execução do VNSP.

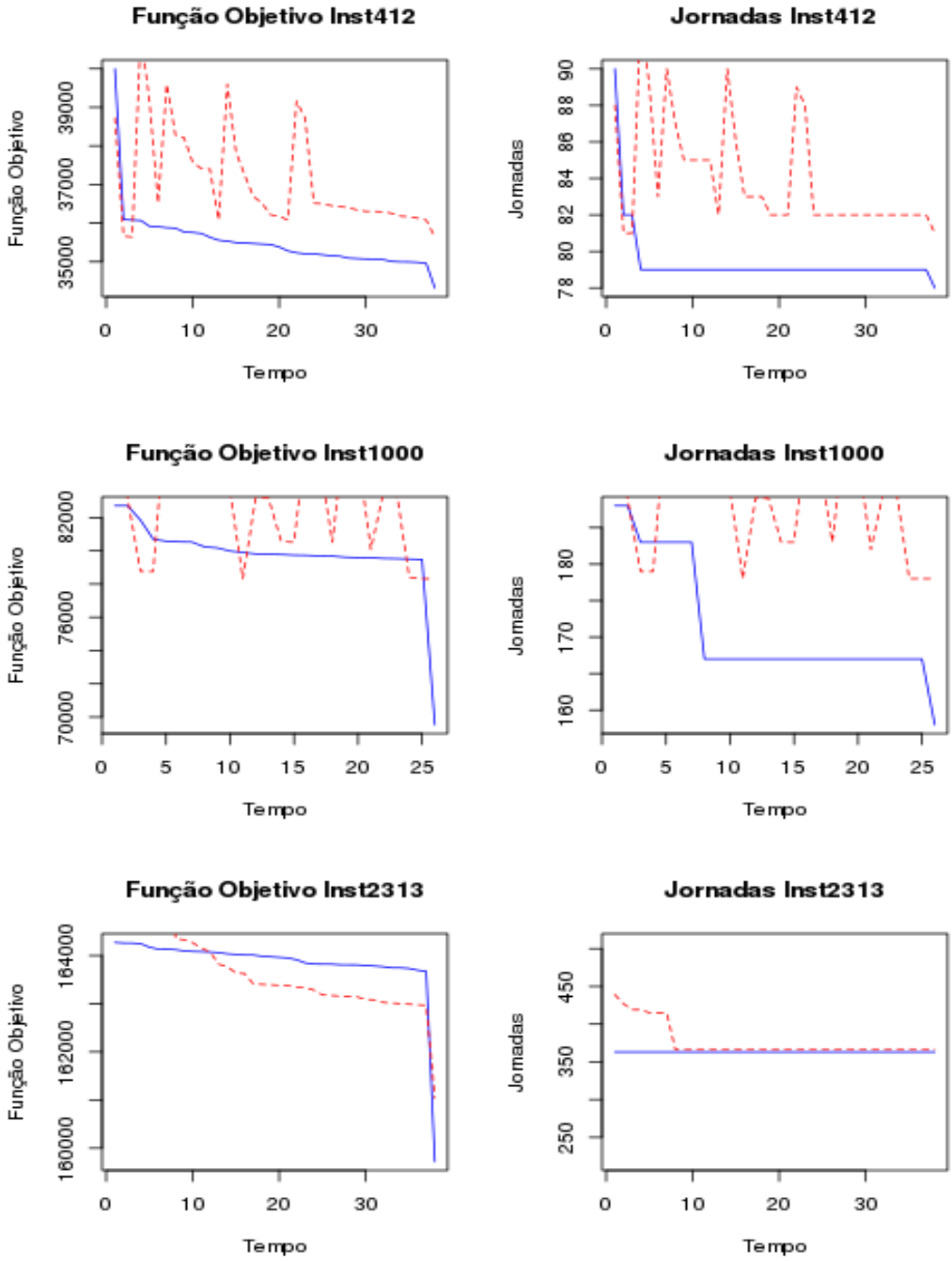

Figura 3: Comparação da evolução da função objetivo e jornadas, da melhor e pior solução gerada pelo VNSP para Ins412, Inst1000 e Inst2313.

A Figura 4 apresenta a taxa de uso das buscas locais, em uma escala logarítmica. Das 36 buscas locais implementadas apenas 14 se mostraram efetivas na melhora da solução inicial. Sendo assim, conclui-se que VNSP acaba por perder tempo de processamento com 22 buscas que não auxiliam na melhora da solução. Estas buscas poderiam então ser desativadas no VNSP, reduzindo assim o tempo de processamento e mantendo a qualidade das soluções obtidas.

A busca local mais utilizada, é a que implementa o operador PCR.B.FI. Seguida pelas buscas que utilizam os operadores PCR.F.CN e Kswap.5.B.FI. Observa-se que as buscas locais que utilizam os operadores mais simples, os quais fazem menos modificações nas soluções, foram as que

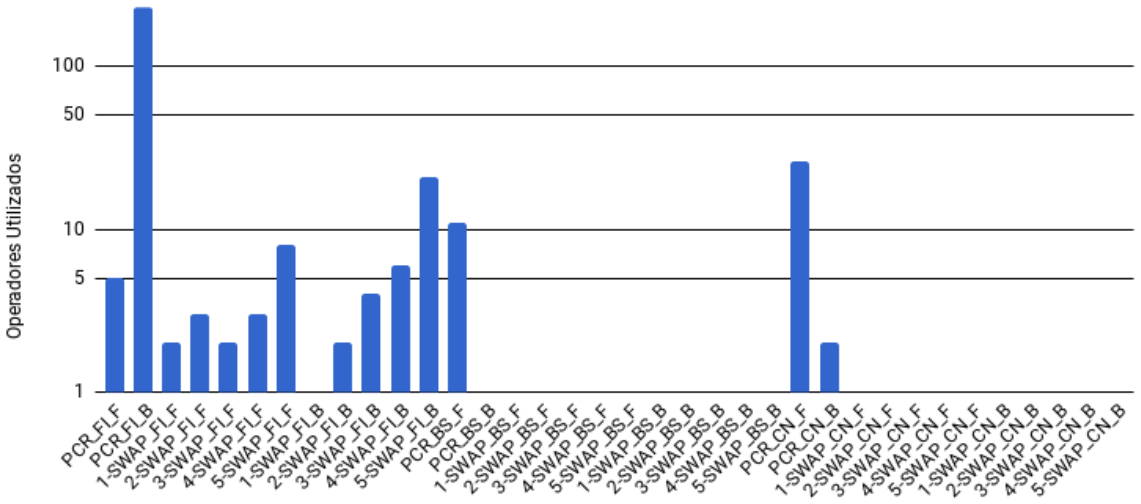

Figura 4: Buscas locais que otimizam as soluções.

mais contribuíram para melhora da solução, com exceção da busca local que usa o Kswap.5.B.FI.

4.4. Comparação com Outras Abordagens

É interessante comparar VNSP com outras abordagens apresentadas na literatura. Para alcançar tal objetivo, o VNSP é comparado com as abordagens de Dos Santos et al. [2016] e Sakayma et al. [2014].

A Tabela 4 apresenta os valores de função objetivo, tempo de execução e as jornadas das soluções obtidas por VNSP e das abordagens de Dos Santos et al. [2016] e Sakayma et al. [2014].

Tabela 4: Comparação função objetivo e número de jornadas.

Instância	VNSP			Dos Santos et al. [2016]			Sakayma et al. [2014]		
	F(x)	Tempo (m)	$\|J\|$	F(x)	Tempo	$\|J\|$	F(x)	Tempo	$\|J\|$
Inst1517	110662	543	251	99871	9753	225	113230	256	1275
Inst2313	163108	1817	370	149960	19610	341	167116	378	18743

Observa-se que os valores de função objetivo obtidos pelo VNSP são piores, em relação a função objetivo e ao número de jornadas, que as solução atingidas por Dos Santos et al. [2016]. No entanto em comparação com os resultados de Sakayma et al. [2014] o VNSP se mostra superior. É possível observar que as propostas de Dos Santos et al. [2016] e Sakayma et al. [2014] levaram um tempo de processamento parecidos, o qual é superior ao tempo de execução do VNSP. Como esperado a abordagem paralela leva um menor tempo de execução do que as abordagens sequenciais.

A Tabela 5 apresenta a diferença [1] entre a função objetivo e o tempo de execução entre os três trabalhos. O VNSP em relação a função objetivo, encontrou soluções cerca de 9% pior que as soluções obtidas por Dos Santos et al. [2016]. Em comparação com Sakayma et al. [2014], as soluções do VNSP se mostraram 1% melhor. Considerando o tempo de execução, a abordagem paralela ganha das duas propostas, justificando assim sua utilização.

[1] Diferença = ((Proposta Comparada / Proposta Nova -1)* 100)

Tabela 5: Diferença dos valores de função objetivo e tempo de processamento.

Instância	Diferença(%) VNSP - Dos Santos et al. [2016]		Diferença(%) VNSP -Sakayma et al. [2014]	
	F(x)	Tempo	F(x)	Tempo
Inst1517	-9.7	511	1.8	134
Inst2313	-8.2	1009	2.87	931

5. Considerações Finais

Este trabalho investigou a proposta de um VNS paralelo para a solução do problema de escalonamento de motoristas de transporte público. Os resultados experimentais obtidos mostraram que a abordagem é válida, pois o VNSP otimizou em até 26% as soluções iniciais das instâncias de testes utilizadas.

Quando comparado com duas abordagens apresentadas na literatura, VNSP perde em relação ao valor de função objetivo. Porém, como esperado, ele possui um tempo de execução inferior as outras abordagens. Esta característica estimula seu uso, considerando que a diferença de tempo de execução é consideravelmente alto.

Como as buscas locais são executadas de forma independente, foi possível identificar quais efetivamente contribuem para a melhoria da solução. Notou-se que das 36 buscas locais, 14 são efetivas na melhoria da solução. Isto indica que VNSP utiliza processamento desnecessário em buscas que não melhoram a solução. Esta descoberta abre caminho para novas pesquisas.

Como trabalhos futuros são enumerados: (1) aumento da quantidade de instâncias de teste; (2) comparação com mais abordagens da literatura; (3) realizar experimentos utilizando apenas as buscas locais que se mostraram eficientes; (4) desenvolver novas estratégias que proporcionem uma melhora na qualidade da solução final; e (5) desenvolver uma estratégia que dinamicamente desligue os operadores que não contribuem na melhora da solução.

6. Agradecimentos

Agradecemos a CAPES, ao CNPq (processo 306754/2015-0) e a Fundação Araucária pelo suporte financeiro para o desenvolvimento deste trabalho.

Referências

Constantino, A. A., de Mendonca Neto, C. F. X., de Araujo, S. A., Landa-Silva, D., Calvi, R., e dos Santos, A. F. (2017). Solving a large real-world bus driver scheduling problem with a multi-assignment based heuristic algorithm. *j-jucs*, 23(5):479–504.

Constantino, A. A., Landa-Silva, D., de Melo, E. L., de Mendonça, C. F. X., Rizzato, D. B., e Romão, W. (2014a). A heuristic algorithm based on multi-assignment procedures for nurse scheduling. *Annals of Operations Research*, 218(1):165–183.

Constantino, A. A., Landa-Silva, D., de Melo, E. L., de Mendonça, C. F. X., Rizzato, D. B., e Romão, W. (2014b). A heuristic algorithm based on multi-assignment procedures for nurse scheduling. *Annals of Operations Research*, 218(1):165–183.

Crainic, T. G., Gendreau, M., Hansen, P., e Mladenović, N. (2004). Cooperative parallel variable neighborhood search for the p-median. *Journal of Heuristics*, 10(3):293–314.

Dos Santos, A., Flausino, Constantino, A., e Romão, W. (2016). Algoritmos baseados na metaheurística VNS aplicados ao problema de escalonamento de motoristas de ônibus. *Simpósio Brasileiro de Pesquisa Operacional*, p. 1388–1399.

Garca-Lopez, F., Melian-Batista, B., Moreno-Perez, J. A., e Moreno-Vega, J. M. (2002). The parallel variable neighborhood search for the p-median problem. *Journal of Heuristics*, 8(3): 375–388.

Hansen, P., Mladenović, N., e Moreno Pérez, J. A. (2008). Variable neighbourhood search: methods and applications. *4OR*, 6(4):319–360.

Hillier, F. S. e Lieberman, G. J. (2013). *Introdução a pesquisa operacional*. McGraw Hill Brasil.

Knausz, M. (2008). *Parallel variable neighbourhood search for the car sequencing problem*. Fakultat fur Informatik der Technischen Universitat Wien.

Marinho, E. H., Ochi, L. S., Drummond, L. M., Souza, M. J. F., e Silva, G. P. (2004). Busca tabu aplicada ao problema de programação de tripulações de ônibus urbano. *Simpósio Brasileiro de Pesquisa Operacional, XXXVI*, p. 1471–1482.

Sakayma, R. Z., Constantino, A. A., e Romão, W. (2014). Meta-heuristica vns aplicada a um problema de planejamento operacional de transporte publico. In *XLVI Simpósio Brasileiro de Pesquisa Operacional*, p. 1632–1643.

Sevkli, M. e Aydin, M. E. (2007). Parallel variable neighbourhood search algorithms for job shop scheduling problems. *IMA Journal of Management Mathematics*, 18(2):117–133.

Silva, G. P. e da Cunha, C. B. (2010). Uso da técnica de busca em vizinhança de grande porte para a programação da escala de motoristas de ônibus urbano. *Transportes*, 18(2).

Silva, G. P., Souza, M. J. F., e Reis, J. v. A. d. (2004). Um método exato para otimizar a escala de motoristas e cobradores do sistema de transporte público. *XVIII ANPET - Congresso de Pesquisa e Ensino em Transporte*.

Yunes, T. H., Moura, A. V., e De Souza, C. C. (2005). Hybrid column generation approaches for urban transit crew management problems. *Transportation Science*, 39(2):273–288.