O *POLLUTION-ROUTING PROBLEM* COM SAÍDAS TARDIAS DO DEPÓSITO: UMA NOVA ABORDAGEM PARA REDUÇÃO DOS CUSTOS

Raphael Kramer, Nelson Maculan
Programa de Engenharia de Sistemas e Computação, COPPE, UFRJ
Centro de Tecnologia, Bloco H, 21941-972, Rio de Janeiro, RJ
{raphaelkramer, maculan}@cos.ufrj.br
Anand Subramanian
Departamento de Engenharia de Produção, Centro de Tecnologia, UFPB
Campus I, Bloco G, Cidade Universitária, 58051-970, João Pessoa, PB
anand@ct.ufpb.br
Thibaut Vidal
Laboratory for Information and Decision Systems — MIT
77 Massachusetts Avenue, Cambridge, MA 02139, U.S.
vidalt@mit.edu

RESUMO

Este artigo trata de duas variantes do Problema de Roteamento de Veículos (PRV) com considerações ambientais. O primeiro se refere ao *Pollution-Routing Problem* (PRP), recentemente introduzido na literatura, cujo objetivo consiste na minimização dos custos operacionais e ambientais, respeitando as restrições de capacidade e janelas de tempo. O custo operacional é baseado no salário dos motoristas e o custo ambiental é baseado na quantidade de gases poluentes emitidos, proporcional ao consumo de combustível. O segundo problema, proposto neste artigo, é uma extensão do PRP que considera saídas tardias do depósito, denominada de PRP *with late departures* (PRPLD). No PRPLD o tempo de saída do depósito é uma variável de decisão que possibilita a redução do tempo da jornada de trabalho dos motoristas e, consequentemente, o custo total. Uma abordagem híbrida que combina uma metaheurística baseada em busca local com uma abordagem exata e um algoritmo recursivo para otimizar as velocidades é proposta para solucionar o problema. Por fim, uma comparação com a versão original do problema é realizada.

PALAVRAS CHAVE. Roteamento de Veículos, Otimização de Velocidades, Logística Verde.

Áreas Principais: Logística e Transportes. Metaheurísticas. Otimização Combinatória.

ABSTRACT

This paper deals with two variants of the Vehicle Routing Problem (VRP) with environmental considerations. The first one refers to the Pollution-Routing Problem (PRP), recently introduced in the literature. The PRP aims to minimize operational and environmental costs while respecting capacity constraints and service time windows. Operational costs are based on driver wages and environmental costs are based on the amount of pollution emitted, which is proportional to fuel consumption. The second problem, proposed in this work, is an extension of the PRP which considers late departures from the depot and it is called PRP with late departures (PRPLD). In the PRPLD the departure time from the depot is a decision variable. This allows workload, and thus, costs reduction. A local search-based metaheuristic with an integer programming approach over a set partitioning formulation and a recursive speed-optimization algorithm is proposed to solve both problems. Finally, we investigate the impact of late departure times on total cost in a variety of settings.

KEY WORDS. Vehicle Routing, Speed Optimization, Green Logistics.

Main areas: Logistics and Transport. Metaheuristics. Combinatorial Optimization.

1 Introdução

O *Problema de Roteamento de Veículos* (PRV) e suas variantes foram objeto de estudo de um elevado número de pesquisas nos anos passados, a maioria destas lidando com objetivos e restrições encontrados em problemas reais (Vidal *et al*., 2013a). Dada a crescente preocupação mundial com relação às questões ambientais, os PRVs começaram a incorporar questões como poluição, utilização de combustíveis alternativos, entre outras (ver e.g., a recente revisão de Lin *et al*. 2014 para PRVs com questões ambientais).

Este trabalho esta focado em uma variante recente do PRV com abordagem ambiental, denominada *Pollution-Routing Problem* (PRP), proposta por Bektaş e Laporte (2011), onde a velocidade do veículo entre pares de clientes é uma variável de decisão que tem impacto direto nas restrições de janela de tempo (JT). O PRP é considerado um problema $\mathcal{NP}$-difícil dado que este generaliza o PRV com Janelas de Tempo (PRVJT). Além disto, uma extensão ao PRP permitindo saídas tardias do depósito, denominada de PRP *with late departures* (PRPLD), é apresentada. Tal abordagem contribui para uma redução significativa da jornada de trabalho dos motoristas e, consequentemente, para a redução dos custos totais.

Um método híbrido que integra um algoritmo de otimização de velocidades (*Speed Optimization Algorithm* - SOA) e uma abordagem de Particionamento de Conjuntos (*Set Partitioning* - SP) em uma metaheurística *iterated local search* (ILS) *multi-start* é utilizado para solucionar o PRP e o PRPLD. A avaliação dos movimentos durante a busca local é efetuada em tempo $\mathcal{O}(1)$ amortizado através da extensão das estruturas de dados auxiliares (EDAs) desenvolvidas por Vidal *et al*. (2013b).

O bom desempenho do método é demonstrado através da resolução de um conjunto de instâncias para o PRP, disponíveis na literatura, caracterizadas por possuir intervalos de JT muito grandes, bem como pela resolução de um conjunto de instâncias geradas com intervalos de JT menores.

O restante do artigo está organizado como segue. A Seção 2 apresenta alguns trabalhos que relacionam PRVs com questões ambientais. A Seção 3 define formalmente o PRP e o PRPLD e a Seção 4 descreve o algoritmo proposto para solucionar os problemas. Os resultados computacionais são apresentados na Seção 5 e a Seção 6 conclui o trabalho.

2 Trabalhos Relacionados

Palmer (2007) foi o primeiro a relacionar roteamento de veículos com questões ambientais. Neste trabalho o autor investigou a variação da quantidade de gás carbônico emitida ao ambiente em função da velocidade do veículo e do consumo de combustível. Considerando diversos cenários de congestão e de janelas de tempo, o autor mostrou que com a modificação da função objetivo, de redução de tempo para redução da emissão de gás carbônico, foi possível obter uma redução média de $4,8\%$ na quantidade de CO_2 emitida à custas de um aumento médio de apenas $0,5\%$ nos custos totais.

Kara *et al*. (2007) propuseram uma formulação matemática para o chamado *Energy Minimizing Vehicle Routing Problem* (EMVRP), que tem como objetivo a minimização da soma do produto entre carga e distância pra cada arco das rotas. Abordagens similares, i.e, aquelas que fazem uso da carga do veículo para minimizar o consumo de combustível e/ou emissões de CO_2, foram apresentadas por Peng e Wang (2009), Scott *et al*. (2010), Ubeda *et al*. (2011) e Xiao *et al*. (2012). Kopfer *et al*. (2013) também consideraram a carga para estimar as emissões de CO_2, porém considerando uma frota heterogênea de veículos.

Entretanto, minimizar o consumo de combustível considerando apenas a carga e a distância não é suficiente. Outro importante fator relacionado ao consumo de combustível é a velocidade dos veículos. Diante do exposto, Kuo (2010) propôs um modelo para a variante do PRV *time-dependent* tratando a duração da viagem e a velocidade dos veículos como dependentes do instante de tempo em que a viagem é realizada e os custos associados ao consumo de combustível

como uma função da quantidade de carga transportada. Para solucionar o problema o autor utilizou a meta-heurística *Simulated Annealing*. Posteriormente, Kuo e Wang (2011) propuseram um algoritmo baseado em Busca Tabu para o mesmo problema. Outros trabalhos baseados na variante *time-dependent* do PRV objetivando a minimização das emissões podem ser encontrados em Figliozzi (2011), Saberi e Verbas (2012) e Jabali *et al*. (2012). Outras referências relacionadas à logística verde são mencionadas nos *surveys* de Dekker *et al*. (2012), Salimifard *et al*. (2012), Lin *et al*. (2014) e Demir *et al*. (2014b).

Bektaş e Laporte (2011) apresentaram uma extensão ao PRV clássico de forma a contabilizar na função objetivo os custos advindos da quantidade emitida de gases poluentes, do consumo de combustível e do tempo total de viagem, além da distância, denominando-a de *Pollution-Routing Problem* (PRP). Além da formulação do PRP, foram propostas três diferentes variantes com objetivos distintos: a) minimização da distância; b) minimização da carga ponderada; e c) minimização da energia. Neste trabalho os autores ainda realizaram uma análise experimental de modo a capturar o *trade-off* entre cada variante, bem como o efeito na distância, energia e nos custos ao incorporar restrições de janelas de tempo e variáveis de velocidade ao modelo.

Demir *et al*. (2012) abordaram o PRP de maneira heurística em dois estágios. Inicialmente resolveram o PRVJT através da meta-heurística *Adaptative Large Neighborhood Search* (ALNS) utilizando 5 operadores de inserção e 12 de remoção. No segundo estágio, utilizaram um algoritmo recursivo para otimizar a velocidade em cada arco de uma dada solução, de modo a minimizar o consumo de combustível e os custos com motoristas. Experimentos computacionais foram realizados em instâncias de até 200 clientes. Outras variantes baseadas no PRP foram propostas por Demir *et al*. (2014a) e Franceschetti *et al*. (2013), onde são apresentados o PRP bi-objetivo e o PRP *time-dependent*, respectivamente.

3 Descrição do Problema

O PRP pode ser definido da seguinte maneira. Seja $G = (\mathcal{V}, \mathcal{A})$ um grafo orientado completo, com $\mathcal{V} = \{0, 1, 2, \ldots, n\}$ sendo o conjunto de vértices e $\mathcal{A} = \{(i, j) : i, j \in V, i \neq j\}$ o conjunto de arcos. O vértice $\{0\}$ representa o depósito, onde estão localizados inicialmente uma frota homogênea de m veículos com capacidade Q. Os vértices $\mathcal{V} - \{0\}$ representam os clientes, caracterizados por uma demanda não-negativa q_i, um tempo de atendimento τ_i e um intervalo de JT $[a_i, b_i]$ para o início do atendimento. Assume-se que a demanda e o tempo de atendimento do depósito são nulos, $q_0 = 0$ e $\tau_0 = 0$, respectivamente. Cada arco $(i, j) \in \mathcal{A}$ representa uma possível viagem de i para j a uma distância d_{ij}.

Uma particularidade do PRP reside no fato de que a velocidade v_{ij} em cada arco (i, j) é uma variável de decisão contínua, definida no intervalo $[v_{min}, v_{max}]$. Além disso, cada veículo emite uma certa quantidade de poluentes que depende de um conjunto de fatores, tais como peso de carga e velocidade. O PRP objetiva encontrar uma matriz de velocidades $(\mathbf{v})_{ij}$ para os arcos, e um conjunto de rotas $\mathbf{R}$ (tal que $|\mathbf{R}| \leq m$) par atender todos os clientes minimizando os custos ambientais e operacionais. Cada rota $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_{|\sigma|})$ deve iniciar e terminar no depósito, i.e., $\sigma_1 = 0$ e $\sigma_{|\sigma|} = 0$, a demanda total de cada rota não pode exceder a capacidade do veículo, e cada cliente deve ser visitado durante sua respectiva JT.

A Equação (1) define, para um rota $\boldsymbol{\sigma}$, a carga do veículo $f_{\sigma_i\sigma_{i+1}}$ no arco (σ_i, σ_{i+1}), e a Equação (2) define recursivamente o tempo t_{σ_i} de chegada nos clientes, considerando que cada rota inicia no tempo zero ($t_{\sigma_1} = 0$) e que é permitido chegar mais cedo no cliente i e esperar pelo início de sua JT (a_i). Chegadas tardias, porém, não são permitidas.

$$f_{\sigma_i\sigma_{i+1}} = \sum_{k=i+1}^{|\sigma|} q_{\sigma_k}, \qquad i = 1, \ldots, |\sigma| - 1 \tag{1}$$

$$\begin{cases} t_{\sigma_1} = 0 \\ t_{\sigma_i} = \max\left\{a_{\sigma_{i-1}}, t_{\sigma_{i-1}}\right\} + \tau_{\sigma_{i-1}} + \dfrac{d_{\sigma_{i-1}\sigma_i}}{v_{\sigma_{i-1}\sigma_i}}, \quad i = 2, \ldots, |\sigma| \end{cases} \tag{2}$$

O objetivo do PRP é baseado na suposição de que as emissões de CO_2 são proporcionais ao consumo de combustível. O consumo de combustível no arco (σ_i, σ_{i+1}) para uma velocidade $v_{\sigma_i\sigma_{i+1}}$ pode ser calculado pela Equação (3), onde os parâmetros w_1, w_2, w_3, w_4 são baseados em propriedades do combustível, do veículo e nas características da rede (Bektaş e Laporte, 2011). Por fim, o objetivo geral do PRP, dado pela minimização da Equação (4), considera tanto o consumo de combustível, a um custo de ω_{FC} por litro, quanto o custo com motoristas, a uma taxa de ω_{FD} por unidade de tempo.

$$F^{\text{F}}_{\sigma_i\sigma_{i+1}}(v_{\sigma_i\sigma_{i+1}}) = d_{\sigma_i\sigma_{i+1}}\left(\frac{w_1}{v_{\sigma_i\sigma_{i+1}}} + w_2 + w_3 f_{\sigma_i\sigma_{i+1}} + w_4 v^2_{\sigma_i\sigma_{i+1}}\right) \tag{3}$$

$$Z_{\text{PRP}}(\mathbf{R}, \mathbf{v}) = \sum_{\boldsymbol{\sigma}\in\mathbf{R}}\left(\omega_{\text{FC}} \sum_{i=1}^{|\sigma|-1} F^{\text{F}}_{\sigma_i\sigma_{i+1}}(v_{\sigma_i\sigma_{i+1}}) + \omega_{\text{FD}}\, t_{\sigma_{|\sigma|}}\right) \tag{4}$$

Dado que a função $F^{\text{F}}_{\sigma_i\sigma_{i+1}}(v_{\sigma_i\sigma_{i+1}})$ é convexa, a velocidade v^*_{F} que minimiza o consumo de combustível pode ser obtida derivando a Equação (3) e igualando à zero. De modo que:

$$v^*_{\text{F}} = \left(\frac{w_1}{2w_4}\right)^{1/3} \tag{5}$$

De maneira similar, para cada arco (σ_i, σ_{i+1}), assumindo que não tenha tempo de espera na rota após σ_i, a velocidade v^*_{FD} que minimiza o custo com combustível e com motorista é dada pela Equação (6). Ambos os valores, v^*_{F} e v^*_{FD}, independem do arco em consideração.

$$v^*_{\text{FD}} = \left(\frac{\frac{\omega_{\text{FD}}}{\omega_{\text{FC}}} + w_1}{2w_4}\right)^{1/3} \tag{6}$$

Ao permitir saídas tardias do depósito, o termo t_{σ_1} da Equação (2) passa a ser uma variável de decisão e o novo problema, denominado PRP *with late departures* (PRPLD), torna-se significativamente mais difícil de ser resolvido. A função objetivo do PRPLD é apresentada na Equação (7).

$$Z_{\text{PRPLD}}(\mathbf{R}, \mathbf{v}) = \sum_{\boldsymbol{\sigma}\in\mathbf{R}}\left(\omega_{\text{FC}} \sum_{i=1}^{|\sigma|-1} F^{\text{F}}_{\sigma_i\sigma_{i+1}}(v_{\sigma_i\sigma_{i+1}}) + \omega_{\text{FD}}\,(t_{\sigma_{|\sigma|}} - t_{\sigma_1})\right) \tag{7}$$

Observa-se que a Equação (7) calcula o custo com motoristas de acordo com o tempo real de trabalho, ao contrário da Equação (4) que calcula o custo de motoristas em função do instante de tempo em que o motorista retorna ao depósito.

4 Algoritmo Proposto

O algoritmo proposto, chamado ILS-SP-SOA, combina uma metaheurística *iterated local search* (ILS) *multi-start* com um procedimento de otimização de velocidades (SOA) e um procedimento de programação inteira baseado em uma formulação de particionamento de conjuntos (SP). Métodos que combinam metaheurísticas com procedimentos exatos são comumente chamados de *matheuristica* (Maniezzo *et al*., 2009).

Conforme mostrado recentemente por Subramanian *et al*. (2013), o ILS em conjunto com o SP é capaz de gerar resultados competitivos para diversas variantes de PRVs que não

consideram restrições de JT. O algoritmo apresentado nesta Seção generaliza esse método através da manipulação eficiente das restrições de JT e aceitando soluções inviáveis (penalizando-as).

A construção da solução inicial e a busca local, bem como os procedimentos de perturbação objetivam minimizar os custos considerando apenas decisões de roteamento, sem modificar as velocidades nos arcos. Esse sub-problema pode ser visto como um PRVJT objetivando a minimização da Equação (4) para o caso do PRP, ou da Equação (7) para o caso do PRPLD. Neste caso, as avaliações dos *movimentos* durante a busca local podem ser realizadas em tempo $O(1)$, conforme apresentado na Seção 4.1. A busca é complementada por um procedimento recursivo de otimização de velocidades, descrito na Seção 4.3, que é aplicada sobre um ótimo local do sub-problema de roteamento (PRVJT). Após a otimização das velocidades, a matriz de tempo de viagem entre clientes (utilizada no PRVJT) é atualizada de acordo com as velocidades obtidas. Por fim as rotas associadas aos mínimos locais são armazenadas em um *pool*, e usadas por um procedimento de otimização inteira baseado em uma formulação de particionamento de conjuntos objetivando a geração de soluções melhores compostas pela recombinação de rotas.

O ILS-SP-SOA é apresentado no Algoritmo 1. O método executa n_{R} reinicializações (Linhas 3-23) de um procedimento que combina ILS, SOA e SP. A cada reinicialização, a matriz de velocidades $\mathbf{v}$ é inicializada com a velocidade máxima permitida (dado da instância), $v_{ij} = v_{\text{MAX}}$ $\forall\ (i,j) \in \mathcal{A}$, de modo a aumentar o número clientes possíveis de serem visitados sem violar as restrições de JT. Uma solução S é obtida após aplicar os procedimentos de busca local e SOA sobre uma solução inicial (Linha 8). Essa solução inicial é gerada utilizando a heurística de inserção mais barata de Penna *et al.* (2013) considerando o objetivo do PRP e relaxando as restrições de JT, conforme apresentado na Seção 4.1. A matriz de velocidades é então atualizada de acordo com as velocidades dos arcos associados à solução S (Linha 9).

Algoritmo 1 ILS-SP-SOA(n_{R}, n_{ILS}, n_{SP}, n_{POOL}, T_{MIP}, semente)

1: $S_{\text{BEST-ALL}} \leftarrow \emptyset$; $f(S_{\text{BEST-ALL}}) \leftarrow \infty$; /*$S_{\text{BEST-ALL}}$ *é a melhor solução geral encontrada**/
2: $P_{\text{PERM}} \leftarrow \emptyset$; $i_{\text{R}} = 0$; /*P_{PERM} *é o pool permanente de rotas no SP**/
3: **enquanto** $i_{\text{R}} < n_{\text{R}}$ **faça**
4: $\quad i_{\text{R}} \leftarrow i_{\text{R}} + 1$;
5: $\quad S_{\text{BEST}} \leftarrow \emptyset$; $f(S_{\text{BEST}}) \leftarrow \infty$; /*$S_{\text{BEST}}$ *é a melhor solução de cada fase de reinicialização**/
6: $\quad P_{\text{TEMP}} \leftarrow \emptyset$; $i_{\text{ILS}} \leftarrow 0$; /*$P_{\text{PERM}}$ *é o pool temporário de rotas no SP**/
7: $\quad \mathbf{v} \leftarrow$ InicializaMatrizDeVelocidades(v_{MAX});
8: $\quad S \leftarrow$ OtimizaVelocidades(BuscaLocal(GeraSolucaoInicial(semente)));
9: $\quad \mathbf{v} \leftarrow$ AtualizaMatrizDeVelocidades(S);
10: $\quad$ **enquanto** $i_{\text{ILS}} < n_{\text{ILS}}$ **faça**
11: $\quad\quad i_{\text{ILS}} \leftarrow i_{\text{ILS}} + 1$;
12: $\quad\quad S \leftarrow$ OtimizaVelocidades(BuscaLocal(Perturba(S_{BEST}, semente)));
13: $\quad\quad \mathbf{v} \leftarrow$ AtualizaMatrizDeVelocidades(S);
14: $\quad\quad P_{\text{TEMP}} \leftarrow P_{\text{TEMP}} \cup$ (Rotas viáveis de S);
15: $\quad\quad$ **se** $f(S) < f(S_{\text{BEST}})$ **então**
16: $\quad\quad\quad S_{\text{BEST}} \leftarrow S$; $i_{\text{ILS}} \leftarrow 0$;
17: $\quad\quad$ **se** $i_{\text{ILS}} \geq n_{\text{ILS}}/2$ **então**
18: $\quad\quad\quad \mathbf{v} \leftarrow$ ReinicializaMatrizDeVelocidades(v_{MAX}, S_{BEST});
19: $\quad$ **se** ($n \leq n_{\text{SP}}$ **e** $i_{\text{R}} = n_{\text{R}} - 1$) **ou** $n > n_{\text{SP}}$ **então**
20: $\quad\quad S_{\text{BEST}} \leftarrow$ MIPSolver(S_{BEST}, P_{TEMP}, P_{PERM}, n_{ILS}, T_{MIP});
21: $\quad$ **se** $f(S_{\text{BEST}}) < f(S_{\text{BEST-ALL}})$ **então**
22: $\quad\quad S_{\text{BEST-ALL}} \leftarrow S_{\text{BEST}}$;
23: $\quad P_{\text{PERM}} \leftarrow P_{\text{PERM}} \cup$ (Rotas viáveis de S_{BEST});
24: $\quad$ **se** número de reinicializações consecutivas $\geq n_{\text{POOL}}$ **então**
25: $\quad\quad P_{\text{TEMP}} \leftarrow \emptyset$;
26: **retorne** $S_{\text{BEST-ALL}}$

O laço ILS é executado por n_{ILS} iterações consecutivas de perturbação e busca local sem melhora. A cada iteração (Linhas 10-18) a solução ótima local é modificada por um mecanismo de perturbação, selecionado aleatoriamente, como descrito na Seção 4.2. As restrições de JT são

relaxadas durante a perturbação. Essa solução modificada é possivelmente melhorada ao aplicar a busca local e o SOA (Linha 12), e a matriz de velocidades é atualizada (Linha 13). Após o SOA, a matriz de tempo de viagem entre clientes é atualizada de acordo com as velocidades ótimas obtidas, a qual será utilizada na próxima iteração da busca local. Um *pool* temporário de rotas é atualizado durante o laço ILS adicionando rotas associadas às soluções ótimas locais (Linha 14), que serão utilizadas como dados de entrada do modelo de particionamento de conjuntos. Por fim, se o número de iterações consecutivas sem melhora do ILS for maior ou igual à $n_{\text{ILS}}/2$, a matriz de velocidades entre clientes é reinicializada com a máxima velocidade permitida, com exceção dos arcos associados à melhor solução corrente (Linhas 17-18).

Se o tamanho da instancia é menor que n_{SP}, então o procedimento SP é chamado uma única vez após a última fase de reinicialização; caso contrário, o procedimento SP é chamado após cada fase de reinicialização. O procedimento SP procura uma nova solução formada por rotas de um *pool* temporário de rotas P_{TEMP} composto por rotas derivadas de ótimos locais obtidas pela busca local, que é esvaziado a cada n_{POOL} reinicializações (Linhas 24-25), e de um *pool* permanente de rotas P_{PERM} que contém rotas associadas às melhores soluções S_{BEST} de cada fase de reinicialização (Linha 23). Os problemas SP são resolvidos utilizando um resolvedor de Programação Inteira Mista (PIM) que chama o ILS-SOA toda vez que uma nova solução incumbente é encontrada (Linha 20). Essa abordagem colaborativa, descrita em Subramanian *et al*. (2013), não só reduz o tempo de execução do resolvedor, como também contribui para encontrar soluções melhores que a melhor combinação possível de rotas do *pool* de rotas. Se isto ocorrer, o *pool* P_{TEMP} é atualizado. O resolvedor PIM é executado diversas vezes até que nenhuma melhora seja encontrada sobre S_{BEST}. Um tempo limite T_{MIP} é imposto à cada execução do resolvedor PIM para evitar elevados tempos de processamento. Por fim, o algoritmo retorna a melhor solução encontrada dentre todas as fases (Linha 26).

4.1 Busca Local e Estruturas de Dados Auxiliares

O procedimento de busca local baseia-se no método RVND (*Randomized Variable Neighborhood Descent*) de Subramanian *et al*. (2010), contando com cinco vizinhanças entre rotas: *Shift(1,0)*, *Shift(2,0)*, *Swap(1,1)*, *Swap(2,2)*, *2-opt**; e cinco vizinhanças intra rotas: *Reinserção*, *Or-opt2*, *Or-opt3*, *Troca* e *2-opt*. Uma descrição detalhada desta vizinhanças, bem como das estruturas de dados auxiliares utilizadas para aumentar o desempenho da busca local, podem ser encontrada em Penna *et al*. (2013) e Vidal *et al*. (2013b).

A busca local para o PRVJT pode ter dificuldades na geração de soluções viáveis, possivelmente comprometendo a convergência para boas soluções. Uma alternativa bastante eficiente para contornar tal situação, tem sido aceitar soluções inviáveis (por atraso no atendimento da JT), penalizando-as (Vidal *et al*., 2013b).

Para calcular o custo das novas rotas geradas durante a busca local em tempo constante amortizado, outras EDAs baseadas em concatenação de subsequências foram implementadas. Para cada subsequência σ o algoritmo armazena:

- duração mínima $T(\sigma)$,
- mínimo uso de *time-warp*[1] $TW(\sigma)$,
- tempo de chegada mais cedo $E(\sigma)$ e mais tarde $L(\sigma)$ para o primeiro cliente gerando a mínima duração e o mínimo uso de *time-warp*,
- carga acumulada $Q(\sigma)$,
- distância $D(\sigma)$,
- tempo de viagem $TT(\sigma)$,
- carga $\times$ distância $QD(\sigma)$,
- e velocidade^2 $\times$ distância $SSD(\sigma)$.

[1] *time-warp* é uma expressão utilizada para designar a quantidade de tempo retornada para viabilizar uma solução.

Para uma subsequência $\bar{\sigma}$ envolvendo um único cliente, i, as EDAs são calculadas como segue: $T(\bar{\sigma}) = \tau_i$; $TW(\bar{\sigma}) = 0$; $E(\bar{\sigma}) = a_i$; $L(\bar{\sigma}) = b_i$; $Q(\bar{\sigma}) = q_i$; $D(\bar{\sigma}) = 0$; $TT(\bar{\sigma}) = 0$; $QD(\bar{\sigma}) = 0$; $SSD(\bar{\sigma}) = 0$. Se o primeiro nó da subsequência é um depósito e se não for permitida saídas tardias do depósito, então $E(\bar{\sigma}) = L(\bar{\sigma}) = 0$. Para subsequências maiores, as EDAs podem ser calculadas, por concatenação $\oplus$, da seguinte maneira.

$$\Delta = T(\sigma) - TW(\sigma) + \delta_{\sigma_{|\sigma|}\sigma'_1} \tag{8}$$
$$\Delta WT = \max\{E(\sigma') - \Delta - L(\sigma), 0\} \tag{9}$$
$$\Delta TW = \max\{E(\sigma) + \Delta - L(\sigma'), 0\} \tag{10}$$
$$T(\sigma \oplus \sigma') = T(\sigma) + T(\sigma') + \delta_{\sigma_{|\sigma|}\sigma'_1} + \Delta WT \tag{11}$$
$$TW(\sigma \oplus \sigma') = TW(\sigma) + TW(\sigma') + \Delta TW \tag{12}$$
$$E(\sigma \oplus \sigma') = \max\{E(\sigma') - \Delta, E(\sigma)\} - \Delta WT \tag{13}$$
$$L(\sigma \oplus \sigma') = \min\{L(\sigma') - \Delta, L(\sigma)\} + \Delta TW \tag{14}$$
$$Q(\sigma \oplus \sigma') = Q(\sigma) + Q(\sigma') \tag{15}$$
$$D(\sigma \oplus \sigma') = D(\sigma) + D(\sigma') + d_{\sigma_{|\sigma|}\sigma'_1} \tag{16}$$
$$TT(\sigma \oplus \sigma') = TT(\sigma) + TT(\sigma') + \delta_{\sigma_{|\sigma|}\sigma'_1} \tag{17}$$
$$QD(\sigma \oplus \sigma') = QD(\sigma) + QD(\sigma') + Q(\sigma')(D(\sigma) + d_{\sigma_{|\sigma|}\sigma'_1}) \tag{18}$$
$$SSD(\sigma \oplus \sigma') = SSD(\sigma) + SSD(\sigma') + v^2_{\sigma_{|\sigma|}\sigma'_1} d_{\sigma_{|\sigma|}\sigma'_1} \tag{19}$$

Logo, o custo penalizado de uma rota σ pode ser calculado como mostrado na Equação (20), onde ω_{TW} é o custo unitário por uso de *time-warp*.

$$\begin{aligned} Z(\sigma) = {} & \omega_{\text{FC}} \left(w_1 TT(\sigma) + w_2 D(\sigma) + w_3 QD(\sigma) + w_4 SSD(\sigma)\right) \\ & + \omega_{\text{FD}} T(\sigma) + \omega_{\text{TW}} TW(\sigma) \end{aligned} \tag{20}$$

4.2 Mecanismos de Perturbação

Os três mecanismos aplicados durante a fase de perturbação são descritos a seguir:

- *Shift to End* — Transferência de um cliente de uma rota para o final de outra rota, selecionados aleatoriamente.
- *Merge Routes* — As duas rotas com menor carga acumulada são mescladas, caso a capacidade do veículo não seja excedida.
- *Change Speeds* — Substitui as velocidades associadas aos arcos de uma rota $r \in S_{\text{BEST}}$ por uma velocidade do conjunto $\{v^*_{\text{F}}, v^*_{\text{FD}}, v_{\text{MAX}}\}$, ambos selecionados aleatoriamente.

Durante a fase de perturbação, *Shift to End* e *Merge Routes* são selecionados aleatoriamente, com diferentes probabilidades: 90% e 10%, respectivamente. Uma perturbação que leve a uma solução inviável, com respeito às restrições de capacidade, é desfeita e uma nova perturbação é selecionada. A perturbação *Change Speeds*, por sua vez, é aplicada apenas após n iterações consecutivas sem melhora do ILS.

4.3 Problema de Otimização de Velocidades

Quando as rotas já estão definidas, o PRP se reduz a um Problema de Otimização de Velocidades, que consiste em encontrar as velocidades ótimas em cada arco (minimizando o custo total da rota), respeitando as restrições de JT. Este problema é resolvido através de um algoritmo recursivo (Algoritmo 2) de complexidade $\mathcal{O}(n^2)$, baseado no algoritmo de Norstad *et al.* (2011) e

Hvattum *et al.* (2013), que, diferentemente dos citados, considera os custos com motoristas, tempos de espera para o atendimento, e o tempo de chegada no último cliente não é conhecido *a priori*.

Seja t'_{σ_i} o tempo de chegada ao cliente σ_i sem aceitar atrasos e esperas (ou seja, o tempo de início do atendimento do cliente σ_i). O Algoritmo 2 é aplicado sobre uma rota completa, definindo $s = 1$ e $e = |\sigma|$, bem como $t'_{\sigma_1} = 0$.

Inicialmente, calcula-se o tempo de chegada $t'_{\sigma_{|\sigma|}}$ ao último cliente utilizando a velocidade v^*_{FD} (minimizando custos com combustível e motorista) partindo do depósito no instante t'_{σ_1}. Se $t'_{\sigma_{|\sigma|}}$ for maior ou menor que os limites de JT, o tempo é atualizado para o limite de JT mais próximo (Linha 6). Em seguida, a velocidade necessária v_{REF} para chegar em t'_{σ_e} é calculada (Linha 7), e o algoritmo encontra o cliente σ_p com maior violação de JT ao trafegar com a velocidade v_{REF} (Linhas 10-13). Se nenhuma violação for encontrada, retorna-se a solução. Caso contrário, o tempo de chegada t'_{σ_p} é corrigido para o limite de JT mais próximo (Linha 15) e o SOA é chamado recursivamente para dois subproblemas: de s à p (Linha 16), e de p à e (Linha 17).

Após o cálculo dos tempos $t'_{\sigma_i}, \forall i = 1, \ldots, |\sigma|$, as velocidades associadas à cada arco são revisadas de modo que todas as velocidades inferiores à velocidade ótima v^*_{F} que minimiza o consumo de combustível seja substituída por v^*_{F} e, então, calcula-se o tempo de chegada em cada cliente t_{σ_i} de modo a aceitar esperas para o atendimento (Linhas 18-22).

Algoritmo 2 Algoritmo de Otimização das Velocidades — SOA

1: Procedimento $SOA(\sigma, s, e)$
2: $maxViolation \leftarrow 0$
3: $D \leftarrow \sum_{i=s}^{e-1} d_{\sigma_i,\sigma_{i+1}}$
4: $T \leftarrow \sum_{i=s}^{e-1} \tau_{\sigma_i}$
5: **se** $e = |\sigma|$ **então**
6: $\quad t'_{\sigma_e} = \min\{\max\{a_{\sigma_e}, t'_{\sigma_s} + D/v^*_{\text{FD}} + T\}, b_{\sigma_e}\}$
7: $v_{\text{REF}} \leftarrow D/(t'_{\sigma_e} - t'_{\sigma_s} - T)$
8: **para** $i = s + 1 \ldots e$ **faça**
9: $\quad t'_{\sigma_i} = t'_{\sigma_{i-1}} + \tau_{\sigma_{i-1}} + d_{\sigma_{i-1},\sigma_i}/v_{\text{REF}}$
10: $\quad violation = \max\{0, t'_{\sigma_i} - b_{\sigma_i}, a_{\sigma_i} - t'_{\sigma_i}\}$
11: $\quad$ **se** $violation > maxViolation$ **então**
12: $\quad\quad maxViolation = violation$
13: $\quad\quad p = i$
14: **se** $maxViolation > 0$ **então**
15: $\quad t'_{\sigma_p} = \min\{\max\{a_{\sigma_p}, t'_{\sigma_p}\}, b_{\sigma_p}\}$
16: $\quad SOA(\sigma, s, p)$ */*substituir por $SOALD(\sigma, s, p)$ para o PRPLD*/*
17: $\quad SOA(\sigma, p, e)$
18: **se** $s = 1$ **e** $e = |\sigma|$ **então**
19: $\quad t_{\sigma_1} = t'_{\sigma_1}$
20: $\quad$ **para** $i = 2 \ldots |\sigma|$ **faça**
21: $\quad\quad v_{\sigma_{i-1},\sigma_i} = max\{d_{\sigma_{i-1},\sigma_i}/(t'_{\sigma_i} - t'_{\sigma_{i-1}} - \tau_{\sigma_{i-1}})), v^*_{\text{F}}\}$
22: $\quad\quad t_{\sigma_i} = max\{a_{\sigma_{i-1}}, t_{\sigma_{i-1}}\} + \tau_{\sigma_{i-1}} + d_{\sigma_{i-1},\sigma_i}/v_{\sigma_{i-1},\sigma_i}$

4.3.1 Problema de Otimização de Velocidades com saídas tardias do depósito

Quando o tempo de saída do depósito não é definido, além de determinar as velocidades ótimas de cada arco, torna-se necessário calcular o tempo ótimo de saída do depósito. Tal objetivo pode ser resolvido através de uma adaptação do SOA (Algoritmo 2), substituindo o método utilizado para resolver o primeiro subproblema da recursão (Linha 16) pelo Algoritmo 3 (SOALD).

O SOALD, apresentado no Algoritmo 3, funciona de maneira similar ao SOA (com mesma complexidade), resolvido no sentido inverso. Se o primeiro nó do subproblema for o depósito inicial, calcula-se o tempo de saída do depósito que permita chegar em σ_e no instante de tempo t'_{σ_e} utilizando a velocidade v^*_{FD}, corrigindo-o para o limite de JT mais próximo, no caso de violação (Linha 6). Então, calculam-se v_{REF} (Linha 7), como no Alg. 2, e os tempos de chegada de cada cliente visitado anteriormente à σ_e, identificando o cliente σ_p com maior violação (Linhas

8-13). Assim como no algoritmo SOA, o tempo t'_{σ_p} é corrigido para o limite de JT mais próximo, e dois subproblemas são resolvidos: $SOALD(\sigma, s, p)$ e $SOA(\sigma, p, e)$.

Algoritmo 3 Algoritmo de Otimização das Velocidades com saídas tardias do depósito — SOALD

1: Procedimento $SOALD(\sigma, s, e)$
2: $maxViolation \leftarrow 0$
3: $D \leftarrow \sum_{i=s}^{e-1} d_{\sigma_i,\sigma_{i+1}}$
4: $T \leftarrow \sum_{i=s}^{e-1} \tau_{\sigma_i}$
5: **se** $s = 1$ **então**
6: $\quad t'_{\sigma_s} = \min\{\max\{a_{\sigma_s}, t'_{\sigma_e} - D/v^*_{\text{FD}} - T\}, b_{\sigma_s}\}$
7: $v_{\text{REF}} \leftarrow D/(t'_{\sigma_e} - t'_{\sigma_s} - T)$
8: **para** $i = e \ldots s+1$ **faça**
9: $\quad t'_{\sigma_{i-1}} = t'_{\sigma_i} - d_{\sigma_{i-1},\sigma_i}/v_{\text{REF}} - \tau_{\sigma_{i-1}}$
10: $\quad violation = \max\{0, t'_{\sigma_{i-1}} - b_{\sigma_{i-1}}, a_{\sigma_{i-1}} - t'_{\sigma_{i-1}}\}$
11: $\quad$ **se** $violation > maxViolation$ **então**
12: $\quad\quad maxViolation = violation$
13: $\quad\quad p = i - 1$
14: **se** $maxViolation > 0$ **então**
15: $\quad t'_{\sigma_p} = \min\{\max\{a_{\sigma_p}, t'_{\sigma_p}\}, b_{\sigma_p}\}$
16: $\quad SOALD(\sigma, s, p)$
17: $\quad SOA(\sigma, p, e)$
18: **se** $s = 1$ **e** $e = |\sigma|$ **então**
19: $\quad t_{\sigma_1} = t'_{\sigma_1}$
20: $\quad$ **para** $i = 2 \ldots |\sigma|$ **faça**
21: $\quad\quad v_{\sigma_{i-1},\sigma_i} = max\{d_{\sigma_{i-1},\sigma_i}/(t'_{\sigma_i} - t'_{\sigma_{i-1}} - \tau_{\sigma_{i-1}})), v^*_{\text{F}}\}$
22: $\quad\quad t_{\sigma_i} = max\{a_{\sigma_{i-1}}, t_{\sigma_{i-1}}\} + \tau_{\sigma_{i-1}} + d_{\sigma_{i-1},\sigma_i}/v_{\sigma_{i-1},\sigma_i}$

5 Resultados Computacionais

Nesta seção são reportados os resultados dos experimentos computacionais obtidos pela utilização do algoritmo ILS-SP-SOA para o PRP proposto por Bektaş e Laporte (2011), bem como para o PRP considerando saídas tardias do depósito. As instâncias resolvidas são apresentadas na Subseção 5.1 e os resultados são apresentados na Subseção 5.2.

5.1 Instâncias

O primeiro conjunto de instâncias (Conj. A) se referem às instâncias da PRPLIB, propostas por Demir *et al.* (2012)[2], que contém 9 diferentes grupos, cada um contendo 20 problemas-teste, variando de 10 à 200 clientes.

No entanto, tais instâncias caracterizam-se por possuir grandes intervalos de JT, de modo que o atendimento destas restrições não é um complicador para a definição das velocidades ótimas. Deste modo, dois novos conjuntos de instâncias foram criados, através da modificação das originais da PRPLIB. As novas instâncias possuem o mesmo horizonte de tempo das originais (i.e. 32400 segundos), porém com intervalos de JT menores. No primeiro conjunto (Conj. B) os clientes possuem intervalos de janelas de tempo variando de 2000 à 5000 segundos. Já no segundo conjunto (Conj. C), os clientes possuem intervalos de JT variando de 2000 à 15000 segundos.

5.2 Resultados para o PRP

O algoritmo proposto foi implementado em linguagem de programação C++ e executado em um PC com processador Intel Core i7-2600 3.4 GHz com 16 GB de memória RAM, sistema operacional Linux Mint Release 13. O resolvedor CPLEX 12.4 foi utilizado para resolver os problemas de particionamento de conjuntos. Apenas uma única *thread* foi utilizada e cada instância foi resolvida 10 vezes, com sementes aleatórias.

[2]Disponíveis em `http://www.apollo.management.soton.ac.uk/prplib.htm`

Os seguintes parâmetros foram adotados: $n_{\text{R}} = 20$, $n_{\text{ILS}} = n + 5m$, $n_{\text{SP}} = 150$, $n_{\text{POOL}} = 2$, $T_{\text{MIP}} = 360$ s, e $\omega_{\text{TW}} = 10^8$. Os quatro primeiros parâmetros foram adotados conforme Subramanian *et al*. (2013), enquanto o último se refere a um número grande para evitar soluções inviáveis por chegadas atrasadas aos clientes. Os valores dos parâmetros da função objetivos foram os mesmos utilizados por Demir *et al*. (2012), i.e.: $w_1 = 1.01763908 \times 10^{-3}$; $w_2 = 5.33605218 \times 10^{-5}$; $w_3 = 8.40323178 \times 10^{-9}$; $w_4 = 1.41223439 \times 10^{-7}$; $\omega_{\text{FC}} = 1.4£/l$ e $\omega_{\text{FD}} = 2.22222222 \times 10^{-3}£/s$.

Os resultados obtidos pelo ILS-SP-SOA estão apresentados na Tabela 1. Cada linha corresponde às médias das soluções obtidas para cada conjunto de 20 instâncias. Cada coluna reporta, para cada método e/ou abordagem, o *Gap* médio (%) e o tempo de CPU. O *Gap* para cada instância é calculado como $100(Z - Z_{\text{BKS}})/Z_{\text{BKS}}$, onde Z é o custo da solução e Z_{BKS} é o valor da melhor solução conhecida para o PRP original (Kramer *et al*., 2014).

Tabela 1: Resultados para o PRP e para o PRPLD

Instance Set	PRP				PRPLD	
	ALNS		ILS-SP-SOA		ILS-SP-SOA	
	Avg. Gap (%)	CPU Time (s)*	Avg. Gap (%)	CPU Time (s)	Avg. Gap (%)	CPU Time (s)
10-A	0.03	2.34	**0.01**	0.04	**-1.40**	0.04
10-B	-	-	**0.11**	0.05	**-11.59**	0.04
10-C	-	-	**0.07**	0.04	**-12.23**	0.04
50-A	0.57	35.40	**0.09**	3.14	**-1.44**	3.16
50-B	-	-	**0.12**	4.55	**-9.43**	4.76
50-C	-	-	**0.32**	4.16	**-11.05**	4.77
100-A	1.99	145.27	**0.28**	32.41	**-1.28**	36.20
100-B	-	-	**0.26**	92.03	**-9.76**	69.57
100-C	-	-	**0.32**	62.35	**-12.16**	61.13
200-A	4.24	625.73	**0.59**	297.38	**-0.68**	340.71
200-B	-	-	**0.63**	1184.03	**-12.80**	782.76
200-C	-	-	**0.92**	545.03	**-14.13**	482.29
Avg.	**1.71**	202.18	**0.31**	185.43	**-8.16**	148.79

* 3 GHz CPU with 1 GB of RAM

Verifica-se que o método proposto foi capaz de melhorar os resultados de Demir *et al*. (2012) de maneira significativa, em termos de qualidade de solução. Embora executados em ambientes computacionais diferentes, o algoritmo ILS-SP-SOA aparenta ser mais rápido que a metaheurística ALNS de Demir *et al*. (2012), dada que a diferença de computadores é moderada.

Comparando os resultados obtidos para os novos conjuntos de instâncias, é possível perceber um aumento no tempo de execução do algoritmo, confirmando uma maior dificuldade nas instâncias propostas. Ao se comparar os resultados do algoritmo ILS-SP-SOA para as duas versões do PRP, verifica-se que ao permitir a saída atrasada do depósito é possível obter uma redução média nos custos de aproximadamente 8.16%. Comparando os tempos de execução, observa-se que o algoritmo se comportou de maneira similar, com exceção dos conjuntos de instâncias B e C de 200 clientes, onde se obteve uma convergência relativamente mais rápida para o PRPLD.

6 Conclusões e Trabalhos Futuros

O artigo apresentado tratou do *Pollution-Routing Problem* (PRP), uma variante recente do Problema de Roteamento de Veículos (PRV) que considera as aspectos ambientais em sua formulação. Novas instâncias foram geradas, tornando o problema mais difícil, e uma modificação no problema original, considerando saídas tardias do depósito, possibilitou a obtenção de soluções com custos menores.

Para resolver os problemas, foi proposto uma *matheurística*, denominada ILS-SP-SOA, que integra eficientemente uma metaheurística de *iterated local search* (ILS) com uma formulação de particionamento de conjuntos (SP) e um algoritmo de otimização de velocidades (SOA). Os resultados apresentados mostram que o ILS-SP-SOA é capaz de gerar soluções de qualidade elevada, de maneira consistente, melhorando os resultados da literatura.

Como trabalhos futuros, o algoritmo ILS-SP-SOA pode ser adaptado para considerar o efeito do congestionamento das ruas nas emissões de CO_2, bem como o efeito do uso de uma

frota de veículos com características distintas, que emitem diferentes quantidades de poluentes ao ambiente. Dado que a função de custo do PRP é formada por duas parcelas (custo com combustível e custo com motoristas) de objetivos conflitantes, o algoritmo ILS-SP-SOA ainda pode ser adaptado para lidar com a versão bi-objetivo do PRP.

Além da adaptação do algoritmo proposto para outras variantes do PRP, trabalhos futuros podem ser elaborados no desenvolvimento de novos mecanismos para calcular as velocidades ótimas de uma rota de maneira eficiente durante a busca local.

Referências

Bektaş, T. e Laporte, G. (2011), The pollution-routing problem. *Transportation Research Part B: Methodological*, v. 45, n. 8, p. 1232–1250.

Dekker, R., Bloemhof, J. e Mallidis, I. (2012), Operations research for green logistics — an overview of aspects, issues, contributions and challenges. *European Journal of Operational Research*, v. 219, n. 3, p. 671–679.

Demir, E., Bektaş, T. e Laporte, G. (2012), An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, v. 223, n. 2, p. 346–359.

Demir, E., Bektaş, T. e Laporte, G. (2014a), The bi-objective pollution-routing problem. *European Journal of Operational Research*, v. 232, n. 3, p. 464 – 478.

Demir, E., Bektaş, T. e Laporte, G. (2014b), A review of recent research on green road freight transportation. *European Journal of Operational Research*, v. 237, n. 3, p. 775 – 793.

Figliozzi, M. A. (2011), The impacts of congestion on time-definitive urban freight distribution networks CO_2 emission levels: Results from a case study in Portland, Oregon. *Transportation Research Part C: Emerging Technologies*, v. 19, n. 5, p. 766 – 778.

Franceschetti, A., Honhon, D., van Woensel, T., Bektaş, T. e Laporte, G. (2013), The time-dependent pollution-routing problem. *Transportation Research Part B: Methodological*, v. 56, p. 265 – 293.

Hvattum, L. M., Norstad, I., Fagerholt, K. e Laporte, G. (2013), Analysis of an exact algorithm for the vessel speed optimization problem. *Networks*, v. 62, n. 2, p. 132–135.

Jabali, O., Van Woensel, T. e de Kok, A. (2012), Analysis of travel times and CO_2 emissions in time-dependent vehicle routing. *Production and Operations Management*, v. 21, n. 6, p. 1060–1074.

Kara, I., Kara, B. Y. e Yetis, M. K. Energy minimizing vehicle routing problem. Dress, A., Xu, Y. e Zhu, B. (Eds.), *Combinatorial Optimization and Applications, Proceedings*, volume 4616 of *Lecture Notes in Computer Science*, p. 62–71, 2007.

Kopfer, H., Schönberger, J. e Kopfer, H. (2013), Reducing greenhouse gas emissions of a heterogeneous vehicle fleet. *Flexible Services and Manufacturing Journal*, p. 1–28.

Kramer, R., Subramanian, A., Vidal, T. e Cabral, L. A. F. (2014), A matheuristic approach for the pollution-routing problem. *Computing Research Repository*, v. abs/1404.4895. Available at http://arxiv.org/abs/1404.4895.

Kuo, Y. (2010), Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem. *Computers & Industrial Engineering*, v. 59, n. 1, p. 157–165.

Kuo, Y. e Wang, C.-C. (2011), Optimizing the VRP by minimizing fuel consumption. *Management of Environmental Quality: An International Journal*, v. 22, n. 4, p. 440–450.

Lin, C., Choy, K., Ho, G., Chung, S. e Lam, H. (2014), Survey of green vehicle routing problem: Past and future trends. *Expert Systems with Applications*, v. 41, n. 4, Part 1, p. 1118 – 1138.

Maniezzo, V., Stützle, T. e Voß, S. (Eds.). *Matheuristics: hybridizing metaheuristics and mathematical programming*. Springer, New York, 2009.

Norstad, I., Fagerholt, K. e Laporte, G. (2011), Tramp ship routing and scheduling with speed optimization. *Transportation Research Part C: Emerging Technologies*, v. 19, n. 5, p. 853 – 865.

Palmer, A. *The Development of an Integrated Routing and Carbon Dioxide Emissions Model for Goods Vehicles*. Tese de doutorado, Cranfield University. School of Management, United Kingdom, 2007.

Peng, Y. e Wang, X. Research on a vehicle routing schedule to reduce fuel consumption. *Proceedings of the 2009 International Conference on Measuring Technology and Mechatronics Automation - Volume 03*, ICMTMA '09, p. 825–827, Washington, DC, USA. IEEE Computer Society, 2009.

Penna, P. H. V., Subramanian, A. e Ochi, L. S. (2013), An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, v. 19, p. 201–232.

Saberi, M. e Verbas, I. O. (2012), Continuous approximation model for the vehicle routing problem for emissions minimization at the strategic level. *Journal of Transportation Engineering-ASCE*, v. 138, n. 11, p. 1368–1376.

Salimifard, K., Shahbandarzadeh, H. e Raeesi, R. Green transportation and the role of operation research. *Proceedings of 2012 International Conference on Traffic and Transportation Engineering*, volume 26, Singapore. IACSIT Press, 2012.

Scott, C., Urquhart, N. e Hart, E. Influence of topology and payload on CO_2 optimised vehicle routing. Chio *et al.* (Eds.), *Applications of Evolutionary Computation*, volume 6025 of *Lecture Notes in Computer Science*, p. 141–150. Springer Berlin Heidelberg, 2010.

Subramanian, A., Drummond, L. M. A., Bentes, C., Ochi, L. S. e Farias, R. (2010), A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, v. 37, n. 11, p. 1899–1911.

Subramanian, A., Uchoa, E. e Ochi, L. S. (2013), A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, v. 40, n. 10, p. 2519–2531.

Ubeda, S., Arcelus, F. e Faulin, J. (2011), Green logistics at Eroski: A case study. *International Journal of Production Economics*, v. 131, n. 1, p. 44 – 51.

Vidal, T., Crainic, T., Gendreau, M. e Prins, C. (2013a), Heuristics for multi-attribute vehicle routing problems: a survey and synthesis. *European Journal of Operational Research*, v. 231, n. 1, p. 1–21.

Vidal, T., Crainic, T. G., Gendreau, M. e Prins, C. (2013b), A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, v. 40, n. 1, p. 475 – 489.

Xiao, Y., Zhao, Q., Kaku, I. e Xu, Y. (2012), Development of a fuel consumption optimization model for the capacitated vehicle routing problem. *Computers & Operations Research*, v. 39, n. 7, p. 1419 – 1431.