Uma abordagem meta-heurística para um problema de rebalanceamento estático em sistemas de compartilhamento de bicicletas

Fábio Cruz, Anand Subramanian
Universidade Federal da Paraíba
Centro de Informática, Universidade Federal da Paraíba (UFPB) – R. dos Escoteiros, s/n – Mangabeira, João Pessoa – PB, CEP 58055-000
fabiocba@di.ufpb.br, anand@ct.ufpb.br

Bruno P. Bruck, Manuel Iori
University of Modena and Reggio Emilia
University of Modena and Reggio Emilia, 42122 Reggio Emilia, Italy.
bruno.p.bruck@gmail.com, manuel.iori@unimore.it

RESUMO

O problema do rebalanceamento estático de bicicletas sem preempção é motivado pela tarefa de reposicionar bicicletas entre estações em um sistema de compartilhamento de bicicletas. Este problema pode ser visto como uma variante do problema do caixeiro viajante com coleta e entrega de um único produto, onde apenas um veículo está disponível para realizar as operações e são permitidas múltiplas visitas a cada estação. Objetiva-se achar uma rota de custo mínimo satisfazendo as demandas das estações e sem violar os limites de cargas mínima (zero) e máxima (capacidade do veículo) ao longo da rota. Portanto, o número de bicicletas a serem coletadas ou entregues em cada estação deve respeitar tais restrições. Uma meta-heurística baseada no *iterated local search* (ILS) é proposta para resolver o problema. O algoritmo ILS foi testado em 450 instâncias da literatura, encontrando a maioria das soluções ótimas conhecidas e também melhorando resultados de várias instâncias abertas.

PALAVRAS CHAVE. Meta-heurísticas. *Bike-sharing*. Roteamento de veículos. Coleta e entrega com múltiplas visitas. *Iterated local search*.

Meta-heurísticas, Otimização Combinatória

ABSTRACT

The static bike rebalancing problem without preemption is motivated by the task of repositioning bikes among stations in a self-service bike-sharing systems. This problem may be seen as a variant of the one-commodity pickup and delivery traveling salesman problem, where only a single vehicle is available to perform the operations and multiple visits to each station are allowed. The objective is to find a least-cost route that meets the demand of all stations and does not violate the minimum (zero) and maximum (vehicle capacity) load limits along the tour. Therefore, the number of bikes to be collected or delivered at each station should respect such constraints. We propose an iterated local search (ILS) based heuristic to solve the problem. The ILS algorithm was tested on 450 benchmark instances from the literature, finding most of the known optimal solutions and also improving the results on several open instances.

KEYWORDS. Metaheuristics. Bike-sharing. Vehicle routing. Split pickup and delivery. Iterated local search.

Metaheuristics, Combinatorial Optimization

1. Introdução

A tarefa de reposicionar um produto de uma localização para outra é um problema bem conhecido que surge em diferentes contextos como logística, transporte e várias disciplinas, principalmente a Engenharia Industrial e a Pesquisa Operacional. Um problema particular está presente na transferência de um produto entre localizações com diferentes demandas, seja requisitando ou fornecendo unidades do produto para um veículo com capacidade limitada.

Uma aplicação prática pode ser encontrada em sistemas *self-service* de compartilhamento de bicicletas (*self-service bike-sharing systems*, BSS), que vem se tornando bastante popular nos últimos anos. Nesses sistemas, os usuários alugam bicicletas e as devolvem em estações distribuídas em uma região. Cada estação tem uma capacidade de armazenamento, um número de bicicletas disponíveis para aluguel e espaços (*slots*) livres para que os usuários possam retorná-las ao sistema.

Ao longo do dia, as estações podem não ter bicicletas disponíveis para aluguel ou *slots* livres para armazenar bicicletas devolvidas. Em vista disso e a fim de evitar esse cenário, que é indesejável para os usuários, é necessário decidir um número aceitável de bicicletas (e *slots* livres) para cada estação iniciar o serviço, e realizar o reposicionamento de bicicletas entre as estações, tarefa executada frequentemente e chamada de rebalanceamento. Segundo [DeMaio, 2009], essa melhoria na distribuição de bicicletas são fatores de impacto na eficiência dos sistemas *bike-sharing* atuais (4ª geração).

Periodicamente algumas bicicletas são movidas de uma estação a outra a fim de restaurar o número de bicicletas para a configuração inicial. Essa tarefa pode ser realizada enquanto o sistema está em funcionamento, denominado rebalanceamento dinâmico, ou quando nenhuma bicicleta está em uso, rebalanceamento estático, por exemplo, de acordo com [Chemla et al., 2013b] e [Dell'Amico et al., 2014], alguns sistemas são fechados durante a noite. Entre outros meios, uma frota de veículos com capacidade limitada é utilizada para transportar as bicicletas, logo, é possível otimizar essa tarefa, pois se trata de um problema particular de roteamento de veículos.

Alternativas para o tráfego urbano são importantes não só pelo seu impacto positivo no congestionamento, como também no meio-ambiente, na comunidade e em vários outros serviços. Os BSS que vem surgindo em todo o mundo tem se mostrado uma alternativa eficaz para mitigar os efeitos dos problemas do tráfego de grandes centros urbanos. De acordo com [DeMaio, 2009], os impactos são muitos: diminuição do tráfego, melhoria na saúde pública, contribuição na redução de emissão de CO_2 etc.

Até 2009, haviam cerca de 120 programas no mundo, e um dos sistemas mais conhecidos é o *Vélib'* em Paris, com 1800 estações e mais de 20000 bicicletas, funcionando 24 horas por dia, todos os dias da semana. No Brasil, podem ser citadas Fortaleza, Recife, Belo Horizonte e São Paulo entre as cidades que já contam com tais programas, algumas com até 60 estações, como o *BikeRio* no Rio de Janeiro. Um mapa interativo com detalhes de vários sistemas em todo o mundo está disponível em `http://bikesharingmap.com`.

O rebalanceamento de sistemas de compartilhamento de bicicletas pode ser modelado como um problema de otimização combinatória (OC), ou seja, em que as decisões são feitas sobre valores discretos, nesse caso, o número de bicicletas coletadas ou entregues durante a visita do veículo à uma estação. O problema do rebalanceamento estático de bicicletas (*static bike rebalancing problem*, SBRP) com um único veículo pode ser visto como uma variante de um outro problema de OC, o problema do caixeiro viajante com coleta e entrega de um único produto (1-PDTSP) [Hernández-Pérez e Salazar-González, 2004a], onde as demandas podem ser servidas em múltiplas visitas. Ambos são considerados problemas de entrega e coleta (*pickup and delivery problems, PDP*), de acordo com [Berbeglia et al., 2007]. É fácil observar que o SBRP pertence à classe de problemas $\mathcal{NP}$-Difícil, já que tem o problema do caixeiro viajante (PCV) como caso especial.

Métodos exatos propostos para variantes do SBRP fornecem soluções ótimas para instâncias de tamanho pequeno e médio, no entanto, nenhum trabalho até o momento resolve de maneira consistente instâncias com mais de 60 estações. Ainda, há um aumento considerável dos

tempos computacionais do método exato à medida que as instâncias se tornam maiores, em alguns casos até mesmo estourando o tempo limite imposto, e em média $43\times$ mais lento que os tempos da abordagem proposta. Portanto, o uso de métodos heurísticos pode ajudar na dificuldade envolvida em instâncias maiores, provendo de modo eficiente *upper bounds* para melhorar a convergência de métodos exatos ou até mesmo soluções ótimas. Este trabalho propõe uma abordagem heurística para o SBRP, em sua variante com um único veículo, que pode fracionar entregas e coletas, ou seja, as demandas das estação podem ser atendidas em múltiplas visitas. Ainda, visitas às estações inicialmente balanceadas (demanda zero) são opcionais.

Este trabalho propõe uma heurística híbrida baseada no *iterated local search* (ILS) para a variante do SBRP com um veículo sem operações temporárias, SBRPW, considerada em [Bruck et al., 2016]. A combinação do ILS com outros métodos, especialmente com a descida em vizinhança variável com escolha aleatória (*randomized variable neighborhood descent*, RVND), tem se mostrado eficaz na resolução de uma grande variedade de problemas de roteamento de veículos ([Penna et al., 2013; Silva et al., 2015; Vidal et al., 2015]), e com apenas um veículo ([Subramanian e Battarra, 2013]).

O algoritmo desenvolvido combina procedimentos eficazes de trabalhos anteriores com um método específico para o problema, baseado em [Bulhões et al., 2016], onde os autores propuseram uma estrutura de dados para verificação de viabilidade em tempo constante. Os resultados obtidos em 450 instâncias de referência da literatura mostram que nosso algoritmo foi capaz de encontrar a maioria dos ótimos conhecidos e também melhorar os resultados nas instâncias abertas.

O restante do trabalho está organizado da seguinte forma. A Seção 2 apresenta alguns trabalhos relacionados ao problema tratado. A Seção 3 detalha a definição do SBRPW. A Seção 4 descreve o algoritmo heurístico proposto. Na Seção 5 são apresentados os testes realizados e os resultados alcançados. Finalmente, a Seção 6 apresenta algumas considerações finais.

2. Trabalhos relacionados

Vários métodos exatos [Benchimol et al., 2011; Chemla et al., 2013a; Erdoğan et al., 2014, 2015; Salazar-Gonzalez e Santos-Hernandez, 2015; Bruck et al., 2016] e heurísticos [Ho e Szeto, 2014; Pal e Zhang, 2015] foram propostos para o SBRP com um único veículo. Com exceção do trabalho de [Bruck et al., 2016], os demais trabalhos assumiram características distintas daquelas lidadas no presente trabalho.

[Chemla et al., 2013b] apresentaram uma formulação matemática para o problema, considerando que as estações podem servir como depósitos temporários, seja para suprir a demanda ou armazenamento temporário de uma sub-rota, estratégia conhecida como *preempção*. [Erdoğan et al., 2015] propuseram o primeiro método exato efetivo, que consiste em um algoritmo *branch-and-cut* com decomposição *Benders*, e reportaram soluções ótimas para instâncias com até 60 estações, para os casos com e sem preempção. Todavia, na variante sem preempção, os autores assumiram que bicicletas pudessem ser coletadas e entregues em uma estação no máximo uma vez, em contraste à variante com preempção, onde essa restrição não existe. Por exemplo, uma estação inicialmente com quatro bicicletas, e que precisa ter dez (demanda de seis), pode ter suas quatro bicicletas temporariamente coletadas pelo veículo, e receber em visitas posteriores o total de dez, tendo na prática sua demanda alterada para mais após a primeira visita.

[Pal e Zhang, 2015] lidaram com o problema sem preempção da forma descrita por [Erdoğan et al., 2015] através da combinação do método da descida em vizinhança variável (*variable neighborhood descent*, VND) com o método de busca em vizinhança de grande porte (*large neighborhood search*, LNS). Os autores reportaram resultados obtidos e compararam com os trabalhos de [Chemla et al., 2013b; Erdoğan et al., 2015], superando, em alguns casos a busca tabu em [Chemla et al., 2013b] e o método exato de [Erdoğan et al., 2015].

No trabalho de [Bruck et al., 2016] foram propostos dois métodos exatos para o SBRP sem preempção (*static bike rebalancing problem without preemption*, SBRPW), onde, diferente da premissa de [Erdoğan et al., 2015], operações *temporárias* são proibidas, ou seja, uma estação

de entrega (de maneira análoga, uma estação de coleta) deve ter seu número de bicicletas sempre crescente (sempre decrescente), até o total de sua demanda. De acordo com os autores, este foi o primeiro trabalho a tratar esta variante especificamente. Ainda, foram reportados resultados teóricos e um procedimento eficiente para checagem de viabilidade de soluções.

3. Definição do problema

Seja n o número de estações, $V = \{1, ..., n\}$ o conjunto de vértices representando suas localizações; estação 0 representa o depósito, e A o conjunto de arcos em um grafo completo orientado, $G = (V \cup \{0\}, A)$. Para cada arco $a_{(i,j)} \in A$, tem-se um custo estritamente positivo c_a, satisfazendo a desigualdade triangular ($c_{(i,j)} + c_{(j,k)} \geq c_{(i,k)}, \forall i, j, k \in V$).

Para cada estação $i \in V$, seja $p_i \in \mathbb{Z}_+$ o número de bicicletas armazenadas em i, antes de o veículo iniciar o serviço, $p'_i \in \mathbb{Z}_+$ a quantidade requerida por i após o serviço ser executado; e seja $d_i = p'_i - p_i$ a demanda total de i, assumindo $d_i > 0$ para estações de entrega, i.e., a estação i tem bicicletas em falta (em contrapartida, estação de coleta quando $d_i < 0$, i.e., a estação i está em excesso e fornece bicicletas para o sistema); algumas estações podem não ter demanda ($p_i = p'_i$) e não serem visitadas pelo veículo (vértices de Steiner); Assume-se que o depósito não tem demandas nem bicicletas, i.e., $p_0 = p'_0 = 0$. Seja $Q \in \mathbb{Z}_{>0}$ a capacidade do veículo. Finalmente, seja uma sequência L representando uma sequência de estações visitas pelo veículo, onde o primeiro e o último vértices são o depósito, para cada visita $l \in L$ seja g_l o número de bicicletas entregues ($g_l < 0$) ou coletadas ($g_l > 0$) pelo veículo; para todas as visitas às estações cujo $d_i < 0$ (coletas), g é sempre positivo (de maneira análoga, g é sempre negativo para estações de entrega). Assim, o número de bicicletas em cada estação cresce (ou descresce) de maneira monótona.

O SBRPW tem por objetivo encontrar uma rota de custo mínimo de modo que (i) um veículo capacitado realize entregas e coletas, iniciando e terminando no depósito (vértice 0), (ii) as quantidades de entregas e coletas em cada visita respeitem as capacidades do veículo e das estações, ou seja, sem violar as restrições de limite de carga mínima (zero) e máxima (Q), e (iii) todas as estações sejam balanceadas, ou seja, as quantidades iniciais p_i são alteradas para as quantidades requeridas $p'_i, \forall i \in V$, visitando ao menos uma vez cada estação com demanda. Dessa forma, uma solução é formada por uma sequência de vértices, e o número de bicicletas coletadas/entregues em cada visita com respeito às restrições mencionadas.

4. Algoritmo proposto

Esta seção apresenta o algoritmo heurístico proposto, chamado $\text{ILS}_{\text{SBRPW}}$, para resolução do problema do rebalanceamento estático de bicicletas. O método é baseado no *framework* da meta-heurística *iterated local search* (ILS). Na fase de busca local, é utilizado o procedimento da descida em vizinhança variável (*variable neighborhood descent*, VND) [Mladenović e Hansen, 1997]. Dado um conjunto de vizinhanças, o VND escolhe de maneira sistemática uma vizinhança após a outra. No $\text{ILS}_{\text{SBRPW}}$, a escolha é aleatória, chamado *randomized variable neighborhood descent* (RVND), como em trabalhos anteriores [Penna et al., 2013; Silva et al., 2015].

A ideia central do $\text{ILS}_{\text{SBRPW}}$ reside em explorar um conjunto menor durante a busca local, com a ajuda de mecanismos de perturbação que ajudam a escapar de ótimos locais. Ou seja, iterativamente, o algoritmo alterna entre fases de busca local (intensificação na melhoria de uma solução) e perturbações (diversificação de soluções) com o objetivo de achar soluções de alta qualidade. Neste trabalho, diferente da maioria das implementações do ILS dos trabalhos citados na seção 1, soluções inviáveis produzidas pelos mecanismos de perturbação são temporariamente aceitas. Isto ajuda não só escapar de ótimos locais como diversificar a estrutura de soluções, encontrando por vezes soluções inalcançáveis usando as vizinhas propostas.

Ainda em relação à perturbação, o método proposto não considera apenas a melhor solução da iteração, mas também, de maneira análoga à técnica de recozimento simulado (*simulated annealing*, SA) [Kirkpatrick et al., 1983], soluções da busca local que não levem a melhoria. Ou seja, para uma maior diversificação, a perturbação pode ser feita sobre uma solução de pior custo,

com probabilidade em função da variável T, da heurística SA. Vale ressaltar que T é inicialmente ajustado para o parâmetro T_0, e reduzido a cada iteração em uma razão $\alpha \in (0, 1)$, diminuindo-se também a probabilidade de se aceitar soluções de pior custo.

O pseudocódigo presente no Algoritmo 1 enumera as fases do $\text{ILS}_{\text{SBRPW}}$. Para cada um dos I_R reinícios, T é ajustado para T_0 e uma solução inicial viável é construída por um procedimento aleatório e guloso (vide Seção 4.1). Em seguida, busca local e pertubações são aplicadas até que o critério de parada seja satisfeito, que é o número de soluções consecutivas do RVND que falham em escapar do ótimo local por I_{ILS} vezes. Por fim, $\text{ILS}_{\text{SBRPW}}$ retorna a melhor solução encontrada dentre todos as iterações.

Algoritmo 1 ILS-RVND

```
1: Procedimento ILS-RVND(I_R, I_ILS)
2: f* ← ∞
3: para i := 1 até I_R faça
4:     s ← GeraSolucaoInicial
5:     s' ← s
6:     currCost ← f(s)
7:     iterILS ← 0
8:     T ← T_0
9:     enquanto iterILS ≤ I_ILS faça
10:        s ← RVND(s, currCost)
11:        Δ ← f(s) − f(s')
12:        se Δ < 0 então
13:            s' ← s
14:            iterILS ← 0
15:        senão
16:            x ∈ [0, 1]
17:            se T > 0 e x < e^{−Δ/T} então
18:                s' ← s
19:        s ← Perturba(s')
20:        currCost ← f(s)
21:        iterILS ← iterILS + 1
22:        T ← T × α
23:    se f(s') < f* então
24:        s* ← s'
25:        f* ← f(s')
26: retorne s*
27: fim ILS_SBRPW.
```

4.1. Procedimento construtivo

O pseudocódigo do procedimento de construção de soluções iniciais é apresentado no Algoritmo 2. O algoritmo mantém uma lista de vértices abertos (OV) correspondentes às estações cujas demandas não estão totalmente satisfeitas. Algumas estações sem demanda escolhidas aleatoriamente são incluídas na lista, para fins de diversificação das soluções iniciais geradas. Ainda, a lista OV mantém uma ordem aleatória (linha 4).

O algoritmo segue um procedimento guloso, inserindo no final da rota (antes do depósito) o primeiro vértice cuja demanda seja totalmente atendida por uma única visita do veículo sem violar os limites ($[0, Q]$). Em seguida, a capacidade remanescente do veículo Q' é atualizada e a estação inserida da solução parcial é removida da lista OV (linhas 8-12).

No entanto, é possível que nenhuma estação possa ser atendida por completo em uma única visita, seja porque o veículo não possui bicicletas suficientes para entregar ou capacidade remanescente para coletá-las de uma só vez. Assim, é necessário fracionar a operação. A segunda parte do procedimento (linhas 13-18) itera sobre OV buscando uma estação cuja demanda maximiza o número de bicicletas a serem entregues ou coletadas. Em caso de empate, o critério de maior proximidade é aplicado. A demanda da estação e a capacidade remanescente do veículo são

atualizados. O algoritmo recomeça da linha 5 e o procedimento é repetido até que OV esteja vazia. Vale ressaltar que a solução inicial construída é sempre viável.

Algoritmo 2 Procedimento de construção de soluções iniciais

1: Procedimento GeraSolucaoInicial
2: $Q' \leftarrow Q$
3: $Solucao \leftarrow \varnothing$
4: $OV \leftarrow$ Lista com ordem aleatória com todas as estações cujo $d_i \neq 0$
+ estações escolhidas aleatoriamente cujo $di = 0$
5: **repita**
6: $inserido \leftarrow FALSE$
7: **para todo** $i \in OV$ **faça**
8: **se** $|d_i| \leq Q'$ **ou** $Q - Q' \geq d_i$ **então**
9: $Solucao \leftarrow Solucao \cup i$
10: atualize Q' e remova i de OV
11: $inserido \leftarrow TRUE$
12: $BREAK$
13: **se não** $inserido$ **então**
14: **para todo** $j \in OV$ **faça**
15: compute $troca_j$
16: $i \leftarrow \max\{troca_j \mid j \in troca\}$
17: $Solucao \leftarrow Solucao \cup i$
18: atualize d_i
19: atualize OV
20: atualize Q'
21: **até que** $OV \neq \varnothing$
22: **return** $Solucao$
23: **end** GeraSolucaoInicial.

4.2. Busca local

A fase de busca local do $\text{ILS}_{\text{SBRPW}}$ é realizada pela exaustão de sete vizinhanças, de acordo com o método RVND descrito no início desta seção 4. Como mencionado, uma solução que terá suas vizinhanças exploradas pode ser inviável (produzida por uma perturbação) e/ou mesmo soluções viáveis podem ter vizinhos que violem os limites de carga ($[0, Q]$), logo, a busca local é adaptada a fim de verificar se as novas sequências de vértices obtidas são soluções viáveis ou não.

Cada vizinhança (escolhida aleatoriamente) realiza o movimento que leve ao vizinho viável de menor custo. As estruturas das sete vizinhanças que compõe a lista do RVND, são descritas como segue:

- *Reinserção* — $N^{(1)}$: Uma estação é removida e inserida em uma outra posição da sequência.
- *Or-opt2* — $N^{(2)}$: Um bloco contíguo de dois vértices adjacentes é removido e então inserido em uma outra posição da sequência.
- *Or-opt3* — $N^{(3)}$: Similar à $N^{(2)}$, três vértices adjacentes são removidas e inseridas em uma outra posição.
- *Or-opt4* — $N^{(4)}$: Também similar à $N^{(2)}$, um bloco contíguo de quatro visitas é removido e inserido em uma outra posição da sequência.
- *2-opt* — $N^{(5)}$: Um movimento clássico do PCV; dois arcos não adjacentes são removidos da sequência, a subsequência resultante da remoção dos arcos é invertida e dois novos arcos são inseridos.

- *Split* — $N^{(6)}$: Uma estação cuja coleta/entrega seja maior que uma bicicleta é selecionada para ter sua demanda fracionada em uma visita adicional. Testa-se cada posição da sequência para se adicionar uma visita, considerando o número de bicicletas violando as carga do veículo e/ou as demandas das estações, causado pelo fracionamento da operação de entrega/coleta na visita original. Esse valor é adicionado aos bits mais à esquerda do custo (ou seja, multiplicado por uma constante de penalidade), dessa forma priorizando movimentos que minimizem a violação de cargas e a soma dos arcos adicionados.
- *Swap* — $N^{(7)}$: Permutação de duas estações.

4.3. Mecanismos de perturbação

Dois mecanismos de perturbação são utilizados. Soluções inviáveis produzidas nessa fase do algoritmo são aceitas. A cada iteração é escolhida aleatoriamente uma das perturbações a seguir:

- *Split* — $P^{(1)}$: De modo similar ao procedimento da busca local, no entanto, soluções de pior custo devido a penalização de violação de carga ou soma dos arcos da sequência podem ser aceitas.
- *Double-bridge* — $P^{(2)}$: Procedimento clássico do PCV, consiste na permutação de dois segmentos da sequência, ou seja, quatro arcos são removidos e quatro novos arcos são inseridos para formar uma nova sequência.

5. Resultados computacionais

O algoritmo foi implementado em linguagem C++ (g++ 4.6.4) e executado em um PC Intel® Core™ i7-3770 3,40 GHz com 16 GB de memória RAM e sistema operacional Linux 64 *bits* (*kernel* 3.11.0-26). As instâncias utilizadas são as descritas em [Hernández-Pérez e Salazar-González, 2004a,b], com diversos valores de n. Cada conjunto contém 10 instâncias com demandas entre -10 e 10 para cada estação i. Da mesma forma que em [Bruck et al., 2016], foram utilizadas as instâncias $n \in \{20, 30, 40, 50, 60\}$ e capacidades do veículo $Q \in \{10, 15, 20, 25, 30, 35, 40, 45, 1000\}$, bem como os seguintes procedimentos: $p_i = 10$, $p'_i = 10 + d_i$, $q_i = 20$, para cada estação i. Os valores de cada arco (distâncias) computados durante a criação dessas instâncias são arredondados para baixo, ou seja, o inteiro imediatamente anterior, seguindo a mesma convenção de trabalhos anteriores. Essa decisão, no entanto, não satisfaz por completo as desigualdades triangulares. Para tanto, e considerando valores simétricos, para quaisquer três vértices i, j, k, sempre que $c_{(i,k)} > c_{(i,j)} + c_{(j,k)}$, então assume-se $c_{(i,k)} = c_{(i,j)} + c_{(j,k)}$. Os resultados para cada instância são as médias de 10 execuções sequenciais do $\text{ILS}_{\text{SBRPW}}$.

Inicialmente, com objetivo de identificar as instâncias mais desafiadoras, um experimento foi conduzido com os seguintes parâmetros: o número de reinícios (I_R) foi ajustado para 100, enquanto definiu-se $10 \times n$ como critério de parada I_{ILS}, sem uso da técnica de SA. Do total de 450 instâncias, foram selecionadas 20 cuja qualidade geral das soluções foram aquém do esperado. Em seguida, foram realizados novos experimentos sobre essas instâncias, mantendo os mesmos parâmetros I_R e I_{ILS} mencionados, $T_0 = 1000$ e $\alpha \in \{0,25; 0,50; 0,75\}$. Na média, as soluções encontradas com $\alpha = 0,75$ foram as de melhor qualidade: *gap* entre as soluções e os *lower bounds* computados em Bruck et al. [2016] para as 20 instâncias foi em média $7,94\%$, sem comprometer o tempo computacional médio, semelhante aos demais valores de α. Por outro lado, $\alpha = 0,25$ e $\alpha = 0,50$ apresentaram maiores médias dos *gaps*, $8,19\%$ e $8,05\%$, respectivamente. Dessa forma, $\alpha = 0,75$ é o melhor custo-benefício dentre as opções testadas, com a qualidade geral das soluções melhores que aquelas em $\alpha \in \{0,25; 0,50\}$, e tempo computacional equivalente.

Como descrito na seção 4, o SA mostrou-se eficaz na melhoria das soluções, e os resultados experimentais foram superiores àqueles sem o uso da técnica. Todavia, testes com valores $T > 1000$ aumentaram o tempo computacional total, mas sem impacto significante nos *gaps*.

Experimentos posteriores consideraram os valores $I_R \in \{50, 100, 150\}$, enquanto foram mantidos I_{ILS}, T_0 e α. Os resultados quando I_R assume 50 obtiveram um *gap* médio de $8,46\%$ e tempo computacional de $18,42s$, ao passo que o *gap* e o tempo quando $I_R = 150$ foram $7,67\%$, $53s$, respectivamente. Por fim, os resultados indicaram um melhor custo benefício quando I_R foi ajustado para 100, com *gap* médio $7,89\%$ ($35,57s$ gastos em média), ou seja, soluções de qualidades equivalentes ou relativamente melhores ($0,5\%$ melhores que $I_R = 50$), e aproximadamente 67% mais rápido que $I_R = 150$. Portanto, os resultados dos experimentos computacionais descritos a seguir fizeram uso dos seguintes parâmetros: $I_R = 100$, $I_{ILS} = 10 \times n$, $T_0 = 1000$ e $\alpha = 0.75$.

As tabelas 1-5 apresenta os resultados para $n \in \{20, 30, 40, 50, 60\}$. Os resultados estão organizados como segue: grupo das instâncias agregados pela capacidade do veículo, seguido dessa, Q; as médias (por grupo de instância), em percentuais, dos *gaps* entre os *lower bounds* computados por Bruck et al. [2016] e tanto a solução média quanto a melhor solução dentre as dez execuções, seguidos pelas médias dos tempos de execução e o tempo médio até a melhor solução encontrada ou alcançar o *lower bound* conhecido, ambos em segundos.

Tabela 1: Resultados para n = 20

Grupo de Instâncias	Q	ILS$_{\text{SBRPW}}$			
		Médias das soluções (%)	Médias das melhores (%)	Médias dos tempos (s)	Médias dos tempos p/ *target* (s)
n20q10	10	0.04	0.00	0.64	0.24
n20q15	15	0.01	0.00	0.14	0.11
n20q20	20	0.00	0.00	0.04	0.04
n20q25	25	0.00	0.00	0.04	0.04
n20q30	30	0.00	0.00	0.03	0.03
n20q35	35	0.00	0.00	0.03	0.03
n20q40	40	0.00	0.00	0.04	0.04
n20q45	45	0.00	0.00	0.03	0.03
n20q1000	1000	0.00	0.00	0.03	0.03

Tabela 2: Resultados para n = 30

Grupo de Instâncias	Q	ILS$_{\text{SBRPW}}$			
		Médias das soluções (%)	Médias das melhores (%)	Médias dos tempos (s)	Médias dos tempos p/ *target* (s)
n30q10	10	1.12	0.58	5.56	3.17
n30q15	15	0.80	0.68	3.42	1.19
n30q20	20	0.25	0.17	2.56	0.72
n30q25	25	0.00	0.00	0.13	0.13
n30q30	30	0.01	0.00	0.41	0.26
n30q35	35	0.00	0.00	0.18	0.18
n30q40	40	0.00	0.00	0.12	0.12
n30q45	45	0.00	0.00	0.17	0.17
n30q1000	1000	0.00	0.00	0.14	0.14

A tabela 6 apresenta, para cada valor de n, o número total de instâncias testadas, seguido pelo número de ótimos conhecidos e o número de ótimos encontrados pelo ILS$_{\text{SBRPW}}$; as médias dos *gaps* entre as melhores soluções e os limites inferiores de [Bruck et al., 2016]; e as médias dos tempos, em segundos. Vale ressaltar que os limites inferiores para $n > 30$ são, por vezes, bem abaixo dos próprios limites superiores computados pelos autores, o que transparece o desafio envolvidos em instâncias média e grandes, i.e., com 40 estações ou mais.

6. Considerações finais

O presente trabalho lida com um problema relativamente recente na literatura, o problema do rebalanceamento estático de bicicletas. Uma abordagem heurística é proposta baseada na

Tabela 3: Resultados para n = 40

Grupo de Instâncias	Q	ILS_{SBRPW}			
		Médias das soluções (%)	Médias das melhores (%)	Médias dos tempos (s)	Médias dos tempos p/ *target* (s)
n40q10	10	4.08	3.24	17.54	8.47
n40q15	15	0.13	0.04	9.46	6.08
n40q20	20	0.01	0.00	2.78	1.65
n40q25	25	0.09	0.09	3.30	1.65
n40q30	30	0.00	0.00	0.72	0.72
n40q35	35	0.00	0.00	0.94	0.94
n40q40	40	0.00	0.00	1.00	0.91
n40q45	45	0.00	0.00	1.14	0.93
n40q1000	1000	0.00	0.00	1.04	0.81

Tabela 4: Resultados para n = 50

Grupo de Instâncias	Q	ILS_{SBRPW}			
		Médias das soluções (%)	Médias das melhores (%)	Médias dos tempos (s)	Médias dos tempos p/ *target* (s)
n50q10	10	6.49	4.73	36.93	21.06
n50q15	15	3.49	2.93	28.46	13.82
n50q20	20	1.01	0.92	25.44	11.71
n50q25	25	0.04	0.00	9.51	6.10
n50q30	30	0.12	0.12	10.86	3.24
n50q35	35	0.09	0.00	9.40	3.95
n50q40	40	0.00	0.00	2.57	2.29
n50q45	45	0.00	0.00	2.34	2.34
n50q1000	1000	0.00	0.00	1.57	1.57

Tabela 5: Resultados para n = 60

Grupo de Instâncias	Q	ILS_{SBRPW}			
		Médias das soluções (%)	Médias das melhores (%)	Médias dos tempos (s)	Médias dos tempos p/ *target* (s)
n60q10	10	16.78	14.38	66.29	31.26
n60q15	15	4.84	3.65	61.98	31.69
n60q20	20	0.94	0.86	39.82	24.06
n60q25	25	0.03	0.02	27.97	13.34
n60q30	30	0.21	0.21	18.18	12.53
n60q35	35	0.05	0.05	20.63	6.76
n60q40	40	0.00	0.00	13.63	6.08
n60q45	45	0.00	0.00	13.52	5.98
n60q1000	1000	0.00	0.00	10.24	7.93

meta-heurística *iterated local search* em conjunto com o procedimento *randomized variable neighborhood descent* como busca local e uma adaptação da técnica de *simulated annealing* para maior diversidade de soluções. Foram realizados testes em 45 grupos com 10 instâncias cada, sendo o número de estações entre 20 e 60, variando Q. Com exceção de 14 instâncias, o algoritmo proposto alcançou todas as melhores soluções conhecidas, bem como forneceu novos valores para 34, cujos ótimos não são conhecidos $(70,8\%)$, sendo 1 para uma instância com 30 estações, 4 para instâncias com 40 estações, 13 com 50 estações e 16 com 60 estações, que podem ser utilizadas como soluções iniciais de métodos exatos, potencialmente aumentando a convergência desses. Embora o algoritmo tenha encontrado um número de ótimos inferior aos conhecidos na literatura, vale ressaltar que o método busca encontrar soluções de boa qualidade em um tempo computacional aceitável, mesmo

Tabela 6: Sumário da performance das melhores solucões agregadas por n

		ILS_{SBRPW}		
n	#	**Ótimos**	**Médias gap (%)**	**Médias tempos (s)**
20	90	89	0.01	0.11
30	90	81	0.24	1.41
40	90	78	0.48	4.21
50	90	66	1.25	14.12
60	90	63	2.54	30.25
Total	450	377	0.90	10.02

para instâncias de médio e grande porte, sendo o maior tempo médio em torno de um minuto de execução, relativamente mais eficiente se comparado a métodos exatos, cujas execuções podem ultrapassar uma hora. Ainda, quando os *lower bounds* diferem das melhores soluções encontradas, a heurística converge eficientemente para soluções de boa qualidade (de acordo com os *gaps* descritos), visto que os tempos médios até as melhores soluções encontradas são relativamente menores.

Como trabalhos futuros estão o estudo de métodos de reparo de soluções inviáveis, outras estruturas de vizinhança para busca local e mecanismos de perturbação. Outros tipos de hibridização entre o ILS e outras heurísticas ou métodos exatos. Também podem ser considerados outros aspectos como visitar obrigatoriamente todas as estações (para verificação da integridade da estação, por exemplo), custos para carga e descarga, restrições de tempo ou de distância da rota, múltiplos veículos, etc.

Finalmente, com a introdução de tecnologias de rastreamento em tempo real, é possível levar em consideração o cenário dinâmico do problema, e utilizar a abordagem para instâncias baseadas em mecanismos de simulação e/ou previsão (Caggiani e Ottomanelli [2013]; Kloimüllner et al. [2014]) das demandas em cada estação, e à medida que se fizesse necessário computar as rotas adequadas e agendar um rebalanceamento mais responsivo.

Referências

Benchimol, M., Benchimol, P., Chappert, B., de la Taille, A., Laroche, F., Meunier, F., e Robinet, L. (2011). Balancing the stations of a self service "bike hire" system. *RAIRO Operations Research*, 45(1):33–61.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., e Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *TOP*, 15(1):1–31.

Bruck, B. P., Cruz, F., Iori, M., e Subramanian, A. (2016). On solving the static bike sharing rebalancing problem without temporary operations. Technical report, Università degli Studi di Modena e Reggio Emilia.

Bulhões, T., Erdoğan, G., Laporte, G., e Subramanian, A. (2016). The static bike relocation problem with multiple vehicles. Technical report, Universidade Federal da Paraíba.

Caggiani, L. e Ottomanelli, M. (2013). A dynamic simulation based model for optimal fleet repositioning in bike-sharing systems. *Procedia-Social and Behavioral Sciences*, 87:203–210.

Chemla, D., Meunier, F., Pradeau, T., Wolfler Calvo, R., e Yahiaoui, H. Self-service bike sharing systems: simulation, repositioning, pricing. URL `https://hal.archives-ouvertes.fr/hal-00824078`. Working paper, 2013a.

Chemla, D., Meunier, F., e Calvo, R. W. (2013b). Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization*, 10(2):120–146.

Dell'Amico, M., Hadjicostantinou, E., Iori, M., e Novellani, S. (2014). The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega*, 45(0):7–19. ISSN 0305-0483.

DeMaio, P. (2009). Bike-sharing: History, impacts, models of provision, and future. *Journal of Public Transportation*, 12(4):3.

Erdoğan, G., Battarra, M., e Calvo, R. W. (2015). An exact algorithm for the static rebalancing problem arising in bicycle sharing systems. *European Journal of Operational Research*, 245(3): 667–679.

Erdoğan, G., Laporte, G., e Calvo, R. W. (2014). The static bicycle relocation problem with demand intervals. *European Journal of Operational Research*, 238(2):451–457.

Hernández-Pérez, H. e Salazar-González, J.-J. (2004a). A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Applied Mathematics*, 145:126–139.

Hernández-Pérez, H. e Salazar-González, J.-J. (2004b). Heuristics for the one-commodity pickup-and-delivery traveling salesman problem. *Transportation Science*, 38:245–255.

Ho, S. C. e Szeto, W. (2014). Solving a static repositioning problem in bike-sharing systems using iterated tabu search. *Transportation Research Part E: Logistics and Transportation Review*, 69 (0):180–198. ISSN 1366-5545.

Kirkpatrick, S., Gelatt, C. D., e Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Kloimüllner, C., Papazek, P., Hu, B., e Raidl, G. R. (2014). Balancing bicycle sharing systems: An approach for the dynamic case. In *Evolutionary Computation in Combinatorial Optimisation*, volume 7832 of *Lecture Notes in Computer Science*, p. 73–84, Berlin, Heidelberg. Springer Berlin Heidelberg.

Mladenović, N. e Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Pal, A. e Zhang, Y. (2015). Free-floating bike sharing: Solving real-life large-scale static rebalancing problems. Technical report, University of South Florida.

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*, 19:201–232.

Salazar-Gonzalez, J.-J. e Santos-Hernandez, B. (2015). The split-demand one-commodity pickup-and-delivery travelling salesman problem. *Transportation Research Part B: Methodological*, 75: 58–73.

Silva, M. M., Subramanian, A., e Ochi, L. S. (2015). An iterated local search heuristic for the split delivery vehicle routing problem. *Computers & Operations Research*, 53:234–249.

Subramanian, A. e Battarra, M. (2013). An iterated local search algorithm for the travelling salesman problem with pickups and deliveries. *Journal of the Operational Research Society*, 64(3): 402–409.

Vidal, T., Battarra, M., Subramanian, A., e Erdoğan, G. (2015). Hybrid metaheuristics for the clustered vehicle routing problem. *Computers & Operations Research.*, 58:87 – 99. ISSN 0305-0548.