LII Simpósio Brasileiro de Pesquisa Operacional
João Pessoa - PB, 3 a 5 de novembro de 2020

Um algoritmo heurístico para o problema de roteamento de ônibus escolares

Iderval da Costa e Silva Neto, Teobaldo Bulhões, Anand Subramanian
Universidade Federal da Paraíba
Centro de Informática, Rua dos Escoteiros s/n, Mangabeira, 58055-000, João Pessoa, Brasil
idervalneto@cc.ci.ufpb.br, tbulhoes@ci.ufpb.br, anand@ci.ufpb.br

Ewerton Teixeira
Universidade Federal da Paraíba
Centro de Tecnologia, Cidade Universitária, 58051-900, João Pessoa, Brasil
ewertonvppt@gmail.com

Raphael Kramer
Universidade Federal de Pernambuco
Departamento de Engenharia de Produção, Av. da Arquitetura, s/n – Cidade Universitária, 50740-550, Recife, PE
raphael.kramer@ufpe.br

RESUMO

Este artigo aborda uma variante do Problema de Roteamento de Ônibus Escolares, que envolve decisões relacionadas à localização de paradas de ônibus, atribuição de estudantes às paradas e roteamento dos ônibus. O objetivo consiste em minimizar o custo total de roteamento para pegar estudantes e entregá-los a uma escola, satisfazendo a capacidade do veículo e a distâncias máxima que estudantes podem caminhar de sua casa até as paradas de ônibus. Para resolver o problema, foi proposto um algoritmo híbrido que combina uma metaheurística *Iterated Local Search* (ILS) com uma formulação de programação linear inteira (PLI). Em particular, o ILS faz uso de diversas vizinhanças e estruturas de dados auxiliares para explorar eficientemente o espaço de busca de solução. O algoritmo proposto foi avaliado ao resolver 95 instâncias de *benchmark*, e mostrou ser capaz de obter soluções de alta qualidade, 10 das quais são melhores do que as melhores encontradas na literatura.

PALAVRAS CHAVE. Roteamento de veículos; Algoritmos; Iterated Local Search.
MH - Metaheurísticas, L&T - Logística e Transportes

ABSTRACT

This paper addresses a variant of the School Bus Routing Problem, which involves decisions related to the location of bus stops, the assignment of students to stops, and the routing of buses. The objective consists in minimizing the total routing cost for picking up students at the bus stops and delivering them to a single school, satisfying the vehicle capacity and the maximum distance the students can walk from their homes to the bus stops. To solve the problem, we propose a hybrid algorithm that combines an iterated local search (ILS) metaheuristc with an integer linear programming formulation. In particular, the ILS makes use of several neighborhoods and auxiliary data structures to efficiently explore the solution search space. The proposed algorithm was evaluated by solving 95 benchmark instances and shown to be capable of providing high-quality solutions, 10 of which were better than the best ones from the literature.

KEYWORDS. School Bus routing; Iterated Local Search; Set Partitioning.
MH - Metaheuristics, L&T - Logistics and Transportation

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

1. Introdução

O Problema de Roteamento de Ônibus Escolares (SBRP, do inglês *School Bus Routing Problem*) é uma extensão do Problema de Roteameto de Veículos (PRV) que vem ganhando bastante relevância e atenção por parte de gestores e pesquisadores nos últimos anos, tanto devido a sua complexidade quanto a sua importância na redução de custos operacionais e ambientais [Ellegood et al., 2019; Fleming, 2019]. Como o próprio nome sugere, o problema está relacionado ao desenvolvimento de um sistema de transporte de ônibus escolar que seja seguro, confiável e economicamente efetivo para transportar estudantes para (e/ou de) suas escolas diariamente.

O SBRP foi proposto no final da década de 60 por Newton e Thomas [1969] e é considerado uma extensão do PRV por incluir decisões relacionadas a localização de paradas de ônibus e alocação de estudantes às paradas. Uma vez escolhidas as paradas de ônibus e definidas as quantidades de estudantes alocados a cada parada, o problema consiste em definir a sequência de paradas a serem visitadas por cada ônibus escolar (i.e., rotas) de modo que a quantidade total de estudantes transportados em cada rota não exceda a capacidade total do veículo e o custo (e.g., distância ou tempo) total das viagens seja minimizado. Além das restrições de capacidade, podem ser considerados os tempos de viagem de modo que todos os estudantes possam chegar na(s) escola(s) antes do início das aulas [Park e Kim, 2010]. Por ser uma extensão do PRV, o SBRP é considerado um problema da classe NP-Difícil.

Dependendo do contexto em que o problema se encontra, um SBRP pode possuir características e/ou restrições adicionais às apresentadas anteriormente. Tomando como base um levantamento dos trabalhos publicados até 2019, Ellegood et al. [2019] classificaram essas características em sete categorias: i) tipo de subproblema; ii) ambiente de aplicação; iii) tipo da frota; iv) número de escolas; v) tipo da carga; vi) objetivos; e vii) restrições. Deste modo, um SBRP pode possuir uma ou mais características relacionadas a cada categoria. Os SBRPs podem incluir decisões relacionadas a escolha das paradas de ônibus, a geração das rotas e/ou aos horários de chegada em cada parada de ônibus e escola(s), podendo ser aplicados tanto em ambientes urbanos ou rurais (implicando, e.g., na densidade das redes rodoviárias e na segurança) envolvendo frotas homogêneas ou heterogêneas (e.g., veículos com capacidades distintas). Quanto ao número de escolas, podem envolver apenas uma ou múltiplas escolas. No caso de múltiplas escolas, existem casos em que são permitidos o transporte de estudantes de diferentes escolas em um mesmo ônibus (carga mista) e casos que não são permitidos (carga homogênea). A respeito dos objetivos, pode-se buscar, de maneira exclusiva ou combinada, a minimização do número de ônibus necessários para realizar o serviço, a minimização da distância total percorrida, a minimização da duração da rota de maior duração, o balanceamento das distâncias (ou duração) do conjunto de rotas, a maximização da utilização da capacidade dos ônibus, etc. Por fim, as restrições dos problemas podem considerar, além das mencionadas no parágrafo anterior, um limite máximo para a distância caminhada pelos estudantes até as paradas de ônibus escolhidas, o tempo máximo de permanência dentro do ônibus, penalizações por atrasos (dos motoristas e/ou estudantes), intervalos de tempo para o ônibus visitar cada parada, etc.

O foco deste artigo está no estudo do SBRP envolvendo decisões relacionadas à localização das paradas de ônibus, à alocação de estudantes às paradas e ao roteamento dos ônibus escolares, com o objetivo de minimizar a distância total percorrida pelos ônibus, garantindo que todos os estudantes sejam atendidos pelo sistema. Além disso, o problema estudado envolve uma única escola, frota homogênea e restrições associadas à capacidade dos veículos e à distância máxima que os estudantes podem caminhar até as paradas que foram alocados. Para solucionar o problema, uma algoritmo baseado na meta-heurística *Iterated Local Search* é proposto e avaliado por meio de experimentos computacionais. Os resultados obtidos mostram-se bastante promissores:

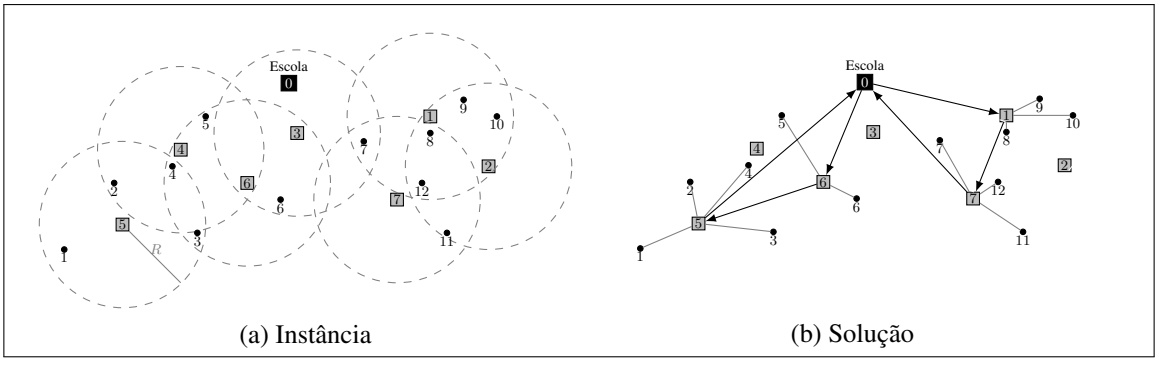
Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

de um total de 95 instâncias resolvidas, o algoritmo foi capaz de encontrar soluções melhores que as da literatura em 10 casos e de igual qualidade em 83 outras.

O restante do trabalho está organizado da seguinte maneira. Uma descrição formal do problema é apresentada na Seção 2, juntamente com uma breve revisão da literatura. A Seção 3 descreve os algoritmos propostos para a resolução do problema, tais como as estruturas de vizinhanças e as perturbações adotadas. Os experimentos computacionais e os resultados obtidos são reportados na Seção 4. Na Seção 5 são apresentadas a conclusão e as sugestões de trabalhos futuros.

2. Descrição do Problema e Revisão da Literatura

O SBRP estudado neste artigo pode ser definido da seguinte maneira. Seja $G = (V, E)$ um grafo não orientado, onde $V = V_P \cup V_S \cup \{0\}$ é o conjunto de vértices e $E = E_P \cup E_S$ é o conjunto de arestas. O conjunto $V_P = \{1, \ldots, m\}$ representa os locais de possíveis paradas de ônibus, enquanto que $V_S = \{1, \ldots, n\}$ representa o conjunto de (residências dos) estudantes e o vértice 0 representa a escola. Além disso, seja $E_P = \{(j, k),\ j, k \in V_P \cup \{0\}, j < k\}$ e $E_S = \{(i, j) :\ i \in V_S, j \in V_P, d_{ij} \leq D\}$, onde d_{ij} é a distância entre os locais i e j, e D é a distância máxima que um estudante pode caminhar (de sua residência) até uma parada de ônibus. O conjunto das paradas $p \in V_P$ que estão dentro do raio permitido para um estudante $i \in V_S$ é denotado por P_i. O SBRP consiste em alocar cada estudante $i \in V_S$ a uma parada de ônibus $j \in P_i$, e em determinar um conjunto de rotas (cada uma associada a um ônibus escolar) que minimizem a distância total percorrida pelos ônibus e que satisfaçam as seguintes restrições: cada parada de ônibus contendo ao menos um estudante alocado deve ser visitada uma única vez; e a quantidade de estudantes alocados às paradas de uma certa rota não pode exceder a capacidade Q do ônibus. A Figura 1 mostra um exemplo de uma instância do SBRP apresentado, com $n = 12$, $m = 7$ e $Q = 6$, e uma possível solução viável com duas rotas.



(a) Instância (b) Solução

Figura 1: Representação gráfica de (a) uma instância e (b) uma solução viável para o SBRP. Pontos na cor preta representam o conjunto de estudantes, quadrados na cor cinza representam os locais de possíveis paradas de ônibus, e o quadrado na cor preta representa a escola.

Conforme mencionado anteriormente, o SBRP foi formalmente proposto no final da década de 60 por Newton e Thomas [1969]. Nesta primeira versão, os locais de parada dos ônibus e os estudantes alocados à cada parada eram dados de entrada, e o problema consistia em determinar as rotas e os horários de cada ônibus, cada rota iniciando e terminando em uma única escola, respeitando-se restrições de capacidade e tempo máximo de viagem. Para solucionar o problema, os autores propuseram uma heurística baseada na geração de um *giant tour* seguido de um procedimento para particioná-lo em rotas menores satisfazendo as restrições do problema. Um problema bastante similar, com rotas iniciando de uma garagem e terminando em uma escola (ou vice-versa), foi resolvido por Bennett e Gazis [1972] por meio de uma heurística baseada no algoritmo de *savings* de Clarke e Wright [1964]. A partir de então, uma série de artigos neste tema foi publicada,

como reportado nos *surveys* de Park e Kim [2010] e, mais recentemente, de Ellegood et al. [2019]. Nestes *surveys*, mais de 90 artigos são analisados, incluindo problemas teóricos e com aplicações reais, envolvendo múltiplas escolas [Ezquerro Eguizábal et al., 2018; Shafahi et al., 2018], frota heterogência [Sales et al., 2018], carga mista [Bögl et al., 2015; Miranda et al., 2018], entre outras características. A seguir, são apresentados os principais e mais recentes artigos relacionados ao SBRP estudado neste artigo.

De acordo com o estudo realizado por Park e Kim [2010], em 2010 poucos algoritmos baseados em meta-heurísticas tinham sido propostos para solucionar o SBRP. Até então, a maioria dos algoritmos adotavam uma abordagem sequencial, que resolviam os subproblemas (e.g., de localização das paradas de ônibus, de alocação de estudantes e de roteamento) um após o outro de maneira independente. Observando essa lacuna na literatura, bem como os bons resultados obtidos ao utilizar meta-heurísticas para solucionar problemas de roteamento, Schittekat et al. [2013] propuseram um algoritmo baseado nas meta-heurísticas GRASP (*Greedy Randomized Adaptive Search Procedure*) e VND (*Variable Neighborhood Descent*) para resolver os subproblemas de forma integrada. A heurística dos *savings* de Clarke e Wright [1964] foi integrada ao GRASP para obter a solução inicial, e quatro estruturas de vizinhanças (reinserção intra e inter-rotas, troca e remoção) foram utilizadas pelo VND para encontrar o ótimo local. Para solucionar o problema de alocação de estudantes às paradas de ônibus (equivalente à um problema de transporte), adotou-se o algoritmo de Ford e Fulkerson [1962].

Kinable et al. [2014] propuseram um algoritmo *branch-and-price* baseado em uma formulação de *set covering*. No procedimento de geração de colunas, o problema de *pricing* consiste em resolver um problema de caminho elementar mais curto com restrições de recursos (ESPPRC, do inglês *elementary shortest path problem with resource constraints*). Por se tratar de um problema NP-difícil, o ESPPRC é resolvido por meio de dois procedimentos: uma heurística baseada em busca local e um algoritmo exato de rotulagem (*labeling algorithm*) baseado em programação dinâmica. Inicialmente, executa-se a heurística. Quando a heurística não encontra colunas com custo reduzido negativo, executa-se o algoritmo de *labeling*. Além disso, diversas técnicas foram adotadas para melhorar a performance do algoritmo, como técnicas de estabilização, de gerenciamento do conjunto de colunas, de ramificação, etc.

3. Algoritmo Proposto

O algoritmo proposto é similar ao ILS (*Iterated Local Search*), apresentado em Subramanian [2012]. Nesse algoritmo são realizadas buscas locais de forma iterativa e perturba-se as soluções encontradas quando um ótimo local é atingido. Durante o procedimento, as rotas geradas são armazenadas e ao final do algoritmo é resolvido um problema de programação linear inteira (PLI) para obter a combinação de rotas de menor custo total.

O algoritmo 1 descreve o algoritmo proposto, chamado de HILS (*Hybrid Iterated Local Search*). Tal algoritmo tem dois parâmetros principais, que são o número de reinicializações(I_{Max}) e o número máximo de iterações da busca local (I_{ILS}). As primeiras linhas tem a função de inicializar uma solução vazia e o valor de sua função objetivo (linhas 2 e 3), assim como a o *pool* de rotas (linha 4) que será utilizado no problema de PLI. Na linha 5, inicia-se o laço de *restarts* do algoritmo, em que a cada iteração uma nova solução inicial é gerada. Nas linhas 9-16 temos o laço principal do algoritmo. A cada iteração é realizada uma busca local (linha 10) e novas rotas são adicionadas ao *pool* de rotas (linha 11). Caso a busca local traga melhora à função objetivo, a nova solução é armazenada como melhor solução da busca local (linhas 12-13). A melhor solução armazenada é então perturbada e atribuída a solução corrente (linha 15) e o processo continua a partir da linha 9. Caso a melhor solução ao fim da busca local seja melhor do que a solução corrente, a solução corrente é atualizada (linhas 18-19). Ao fim das reinicializações, o modelo de PLI é resolvido (linha

20) e a solução associada a melhor composição de rotas com base no *pool* de rotas é retornada (linha 21).

Algoritmo 1 HILS

```
1:  procedimento HILS(I_Max, I_ILS, v, T_0, α)
2:      s* ← ∅
3:      f* ← ∞
4:      PoolRotas ← ∅
5:      para i ← 1, ..., I_Max faça
6:          s ← geraSoluçãoInicial()
7:          s' ← s
8:          IterILS ← 0
9:          enquanto IterILS ≤ I_ILS faça
10:             s ← RVND(s)
11:             PoolRotas ← AtualizaPoolRotas(s, PoolRotas)
12:             se f(s) < f(s') então
13:                 s' ← s
14:                 IterILS ← 0
15:             s ← Perturba(s')
16:             IterILS ← IterILS + 1
17:         se f(s') < f(s*) então
18:             s* ← s'
19:             f* ← f(s')
20:     s* ← PLI(PoolRotas, s*)
21:     retorne s*
```

3.1. Procedimento construtivo

O procedimento construtivo é realizado de forma semelhante àquele proposto por Subramanian [2012]. Nesse procedimento é criado uma lista de candidatos (*LC*) que contém paradas não pertecentes a solução. Então, paradas são inseridas em rotas baseado em um de dois critérios escolhido aleatoriamente no início do procedimento, sendo eles o critério do vizinho mais próximo, e o critério da inserção mais barata. Esses critérios buscam minimizar uma função *g(k)* para calcular o custo de inserção da parada k.

O critério da inserção mais barata calcula o custo baseado na inserção de uma parada $k \in LC$ entre um par de paradas adjacentes *i* e *j* pertencentes à solução.

$$g(k) = c_{ik} + c_{kj} - c_{ij} \tag{1}$$

No critério do vizinho mais próximo, é calculada a menor distância entre $k \in LC$ e uma parada *i* pertencente à solução.

$$g(k) = c_{ik} \tag{2}$$

Paradas são iterativamente inseridas em rotas através desse processo. Como verificação da viabilidade de inserção, um processo de alocação guloso de estudantes como o mostrado na seção 3.4.1 é utilizado. O procedimento termina quando uma solução viável é encontrada. Isso é, uma solução em que cada estudante está alocado a exatamente uma parada.

3.2. Busca local

A busca local é baseada no procedimento RVND (*Random Variable Neighborhood Descent* como visto em Subramanian [2012]. Esse procedimento utiliza uma lista de estruturas de vizinhança que são exploradas aleatoriamente. Caso ao explorar uma dessas vizinhanças seja obtido melhora na função objetivo, todas as estruturas de vizinhança ficam disponíveis na próxima iteração. Do contrário, a estrutura é removida da lista. Se todas as vizinhanças forem exploradas e nenhuma melhora for alcançada, deixando a lista de vizinhanças vazia, o procedimento é finalizado.

O algoritmo aqui proposto faz uso de 12 estruturas de vizinhança. São usadas 5 estruturas inter-rotas (subseção 3.2.1) e 7 estruturas intra-rota (subseção 3.2.2).

3.2.1. Estruturas de vizinhança inter-rotas

Foram utilizadas 5 estruturas inter-rotas, que são analisadas exaustivamente. Uma vez que o problema possui restrições de capacidade nos veículos, é sempre necessário verificar se um determinado movimento irá exceder a carga máxima do veículo. Isso pode ser feito verificando a quantidade de estudantes que as paradas em questão estão carregando, e analisando se o movimento fará com que as cargas dos veículos excedam a capacidade máxima. Caso isso torne o movimento inviável, outro procedimento é utilizado em que todos os estudantes são realocados (visto na seção 3.4). Caso nenhum dos dois métodos faça com que o movimento seja viável, ele não é executado. As 5 vizinhanças utilizadas foram as seguintes:

Shift(1,0) — $N^{(1)}$: Uma parada k é transferida de uma rota r_1 a uma rota r_2.
Shift(2,0) — $N^{(2)}$: Um bloco de duas paradas consecutivas k e l é transferido de uma rota r_1 para outra rota r_2.
Swap(1,1) — $N^{(3)}$: Uma parada k de r_1 é permutada por uma parada l de r_2.
Swap(2,1) — $N^{(4)}$: Um bloco de duas paradas consecutivas k e l de r_1 é permutado por uma parada k' de r_2.
Swap(2,2) — $N^{(5)}$: Um bloco de duas paradas consecutivas k e l de r_1 é permutado por outro bloco de duas paradas consecutivas k' e l' de r_2.

3.2.2. Estruturas de vizinhança intra-rota

Sempre que um movimento inter-rota traz melhora à função objetivo, é feita a busca inter-rota. Duas das estruturas utilizadas, Remove e Replace, são específicas ao SBRP, devido à natureza do problema.
Swap — $N^{(1)}$: Duas paradas de uma mesma rota são trocadas.
2-opt — $N^{(2)}$: Dois arcos de uma rota são removidos e ela é reconstruída de forma invertida.
Reinsertion — $N^{(3)}$: Uma parada é retirada de sua posição e inserida em outra da mesma rota.
Or-opt2 — $N^{(4)}$: Semelhante à $N^{(3)}$, mas realocando-se um par de paradas adjacentes.
Or-opt3 — $N^{(5)}$: Semelhante à $N^{(3)}$, mas realocando-se três paradas adjacentes.
Remove — $N^{(6)}$: Uma parada é removida de uma rota. Para verificar a viabilidade da solução após a operação, é necessário realizar o procedimento de alocação de estudantes como visto na seção 3.4.
Replace — $N^{(7)}$: Uma parada de uma rota é permutada com uma parada que não está sendo utilizada. Assim como a vizinhança $N^{(6)}$, é necessário realizar o procedimento de alocação de estudantes para verificar se a viabilidade do movimento.

3.3. Perturbação

Para que a solução não fique presa em um ótimo local, se faz necessário o uso de perturbações. A cada iteração do ILS uma das duas perturbações utilizadas é aleatoriamente escolhida:

Multiple-Swap(1,1) — $P^{(1)}$: São realizados múltiplos movimentos Swap(1,1) aleatoriamente.

Multiple-Shift(1,1) — $P^{(2)}$: São realizados múltiplos movimentos Shift(1,1) de forma aleatória. Nesse movimento, uma parada k é transferida de uma rota r_1 para uma rota r_2, e uma parada l é transferido de uma rota r_2 para uma rota r_1, sendo inseridas em posições aleatórias.

3.4. Alocação de estudantes

Para que uma solução seja considerada viável é necessário que exista uma disposição de estudantes nas diferentes paradas de tal forma que sejam atendidas as restrições de que todo estudante deve estar alocado a exatamente uma parada e de que nenhum veículo tenha sua carga excedida. Para esse fim, foi elaborado um procedimento híbrido que utiliza um algoritmo guloso e um algoritmo exato baseado no problema de fluxo máximo. Tais algoritmos são executados de maneira sequencial: se o algoritmo guloso não encontrar uma solução viável, executa-se o algoritmo exato. A descrição destes algoritmos é apresentada seguir.

3.4.1. Algoritmo guloso

Foi elaborado um algoritmo semelhante ao utilizado por Niasar et al. [2017]. Nesse procedimento, estudantes que possuem uma menor quantidade de paradas no seu raio de alcance tem maior prioridade e são alocados primeiro. Em casos que o estudante possua mais de uma parada em seu raio de alcance, o estudante é alocado para a parada mais próxima.

O algoritmo tem como parâmetros de entrada o conjunto de rotas da solução, representado por R, e seu funcionamento está descrito no Algoritmo 2. Primeiramente, inicializam-se: i) a coleta de cada rota com zero (linha 2); ii) um conjunto P contendo as paradas utilizadas no conjunto de rotas da solução (linha 3); e iii) um conjunto vazio S para armazenar os estudantes alocados (linha 4). Nas linhas 5-11, é mostrado o laço principal do algoritmo, iterando por todos os estudantes, e por todas as paradas que tal estudante consegue alcançar. Se a rota r que visita a parada p possui capacidade para que um estudante seja alocado, a coleta daquela parada é incrementada e o estudante é adicionado ao conjunto S (linhas 7-11). Caso a quantidade de estudantes alocados seja igual ao número total de estudantes, a alocação é um sucesso, e o procedimento retorna verdadeiro. Do contrário, o procedimento retorna falso (linhas 13-15).

Algoritmo 2 AlocaçãoGulosaDeEstudantes

```
1:  procedimento AlocaçãoGulosaDeEstudantes(R)
2:      c_r ← 0, ∀r ∈ R
3:      P ← {p ∈ V_P : p é visitada por alguma rota r ∈ R}
4:      S ← ∅
5:      para i ∈ V_S em ordem não decrescente de |P_i ∩ P| faça
6:          para p ∈ P_i ∩ P em ordem não decrescente de distância faça
7:              r ← rota de R que passa pela parada p
8:              se c_r < Q então
9:                  c_r ← c_r + 1
10:                 S ← S ∪ {i}
11:                 break
12:     se S = V_S então
13:         retorne verdadeiro
14:     senão
15:         retorne falso
```

3.4.2. Algoritmo exato

No algoritmo guloso apresentado, a ordem de alocação de estudantes é predeterminada baseado em um sistema de prioridades. Por consequência, em instâncias mais complexas, pode ser

que uma alocação viável não seja encontrada, mesmo nos casos em que ela exista. Foi observado, então, que a interação entre estudantes, paradas e rotas pode ser modelada como um problema de fluxo máximo na rede G' definida a seguir.

Seja $G' = (V', A)$ um grafo direcionado, onde $V' = V_S \cup V_P \cup V_R \cup \{s, t\}$ é o conjunto de vértices e $A = A_1 \cup A_2 \cup A_3 \cup A_4$ é o conjunto de arcos. O conjunto V_S representa o conjunto de estudantes; V_p, o conjunto de paradas; V_R, o conjunto de rotas da solução; s, o vértice de origem; e t, o vértice de destino. Já o conjunto A_1 contém os arcos que conectam o vértice s aos vértices de V_S; A_2, os arcos que conectam os vértices de V_S aos vértices de V_P; A_3, os arcos que conectam os vértices de V_P aos vértices de V_R; e A_4, os arcos que conectam os vértices de V_R ao vértice t. Formalmente, os conjuntos A_1, A_2, A_3 e A_4 podem ser definidos como: $A_1 = \{(s, i), i \in V_S\}$; $A_2 = \{(i, p), i \in V_S, p \in V_p : d_{ip} \leq D\}$; $A_3 = \{(p, r), p \in V_p, r \in V_R : p \text{ seja visitada por } r\}$ e $A_4 = \{(r, t), r \in V_R\}$. Além disso, as capacidades dos arcos são definidas como $c(i, j) = 1$, para todo arco $(i, j) \in A_1 \cup A_2$, e $c(i, j) = Q$, para todo arco $(i, j) \in A_3 \cup A_4$. Assim, a alocação será viável se, e somente se, o fluxo máximo de s a t for igual a $|V_S|$.

Para resolução do problema modelado, foi usada uma implementação do algoritmo *push-relabel* proposto por Goldberg e Tarjan [1988]. Contudo, a execução de tal algoritmo é mais custosa quando comparada ao algoritmo guloso, o que afeta negativamente o tempo de execução. Para amenizar tal situação, foi utilizado um procedimento de "aprendizado" que ajuda a evitar a execução do algoritmo exato tomando como base a correlação entre a quantidade de estudantes não alocados à uma rota pelo algoritmo guloso e a possibilidade de que o algoritmo exato encontre uma alocação viável. Para identificar tal situação, durante a primeira iteração do HILS foi calculada a média da quantidade de estudantes que não foram alocados pelo algoritmo guloso e que resultaram em uma solução viável obtida pelo algoritmo exato. Tal valor foi chamado de *Corte*. Então, para as iterações posteriores do HILS, foi adicionada uma condição que impede a execução do algoritmo exato quando o número de estudantes não alocados pelo algoritmo guloso supere o valor *Corte*.

3.5. Formulação de Programação Linear Inteira

Sejam R o conjunto de rotas, $R_j \subseteq R$ o subconjunto de rotas que contêm a parada $j \in V_P$ e $R_i \subseteq R$ o subconjunto de rotas que atendem ao estudante $i \in V_S$. Defina y_r como a variável de decisão associada a rota $r \in R$ e c_r seu custo. Considere a seguinte formulação:

$$\text{Min} \sum_{r \in R} c_r y_r \tag{3}$$

sujeito a

$$\sum_{r \in R_i} y_r \leq 1 \qquad \forall j \in V_P, \tag{4}$$

$$\sum_{r \in R_i} y_r \geq 1 \qquad \forall i \in V_S, \tag{5}$$

$$y_r \in \{0, 1\} \qquad \forall r \in R. \tag{6}$$

Uma vez que temos as rotas adicionadas ao *pool* de rotas, preenchido no procedimento HILS, podemos utilizar tal formulação para minimizar o custo da solução. A função objetivo (3) minimiza a soma dos custos das rotas. As restrições (4) garantem que cada parada aparece no máximo uma vez na composição de rotas. As restrições (5) garantem que todo estudante será coberto por pelo menos uma rota. Por fim, as restrições (6) tornam y_r em uma variável binária.

4. Experimentos Computacionais

Com o intuito de avaliar os algoritmos e modelos apresentados, experimentos computacionais foram executados em um computador com processador Intel®Core™i7 de 3.4 GHz e com 16 GB de memória RAM, em um sistema operacional Ubuntu 16.04. Os algoritmos foram implementados em C++, e a seguinte parametrização foi adotada: I_{Max} = 50 e I_{ILS} = 50, para instâncias com $m < 20$; e I_{Max} = 50 e I_{ILS} = 125, para instâncias com $m \geq 20$.

Para conduzir os experimentos foram consideradas 95 das 112 instâncias utilizadas por Schittekat et al. [2013]. Em tais instâncias, o número de paradas varia de 5 à 40 e o número de estudantes varia de 25 à 800. Devido a existência de alguns componentes aleatórios no HILS, cada instância foi resolvida 10 vezes e os resultados obtidos foram comparados com os reportados por Schittekat et al. [2013] e Kinable et al. [2014] que, por sua vez, foram obtidos por uma metaheurística GRASP-VND e por um algoritmo branch-and-price, respectivamente.

Na Tabela 1, as 4 primeiras colunas mostram, respectivamente: a ID da instância, a quantidade de paradas (m), a quantidade de estudantes (n) e a capacidade dos veículos (Q). As colunas *MH* e *B&P*, por sua vez, mostram as melhores soluções obtidas por Schittekat et al. [2013] e Kinable et al. [2014], respectivamente. Nas duas colunas seguintes são reportadas a média e a melhor das soluções obtidas pelo HILS, após as dez execuções. Por fim, as três últimas colunas reportam o tempo médio de execução do HILS, em segundos, e os *gaps* da melhor solução obtida pelo HILS para *MH* e *B&P*. Os valores destacados em negrito mostram resultados em que as soluções do HILS foram melhores do que as obtidas por Schittekat et al. [2013] ou Kinable et al. [2014].

Quando comparado com a metaheurística GRASP-VND de Schittekat et al. [2013], verfica-se a superioridade do HILS proposto pois o mesmo foi capaz de encontrar soluções melhores em 18 das 95 instâncias e de igualar os resultados obtidos em 75 instâncias, obtendo um *gap* médio de -0.42%. Quando comparado com o método exato proposto por Kinable et al. [2014], que possui as melhores soluções conhecidas na literatura, o algoritmo proposto obteve soluções melhores em 10 das 95 instâncias e de mesmo custo em outras 83, obtendo um *gap* médio de -0.24%.

Tabela 1: Resultados computacionais

ID	m	n	Q	MH	B&P	HILS média	HILS melhor	Tempo(s)	$Gap_{MH}(\%)$	$Gap_{B\&P}(\%)$
1	5	25	25	141,01	141,01	141,01	141,01	0,1	0	0
2	5	25	50	161,62	161,62	161,62	161,62	0,03	0	0
3	5	25	25	182,14	182,14	182,14	182,14	0,01	0	0
4	5	25	50	195,8	195,8	195,8	195,8	0,02	0	0
5	5	25	25	111,65	111,65	111,65	111,65	0,01	0	0
6	5	25	50	103,18	103,18	103,18	103,18	0,1	0	0
7	5	25	25	7,63	7,63	7,63	7,63	0,1	0	0
8	5	25	50	25,64	25,64	25,64	25,64	0,12	0	0
9	5	50	25	286,68	286,68	286,68	286,68	0,28	0	0
10	5	50	50	197,2	197,2	197,2	197,2	0,22	0	0
11	5	50	25	193,55	193,55	193,55	193,55	0,53	0	0
12	5	50	50	215,86	215,86	215,86	215,86	0,23	0	0
13	5	50	25	130,53	130,53	130,53	130,53	0,71	0	0
14	5	50	50	96,26	96,26	96,26	96,26	0,25	0	0
15	5	50	25	12,89	12,89	12,89	12,89	0,33	0	0
16	5	50	50	30,24	30,24	30,24	30,24	0,26	0	0
17	5	100	25	360,35	360,35	360,35	360,35	0,53	0	0
18	5	100	50	304,23	304,23	304,23	304,23	0,39	0	0
19	5	100	25	294,21	294,21	294,21	294,21	1,41	0	0
20	5	100	50	229,41	229,41	229,41	229,41	0,32	0	0
21	5	100	25	134,95	134,95	134,95	134,95	1,17	0	0
22	5	100	50	144,41	144,41	144,41	144,41	0,75	0	0
23	5	100	25	58,95	58,95	58,95	58,95	1	0	0
24	5	100	50	39,44	39,44	39,44	39,44	1,06	0	0
25	10	50	25	242,85	242,85	242,85	242,85	1,35	0	0

Tabela 1: Resultados computacionais (continuação)

ID	m	n	Q	MH	B&P	HILS média	HILS melhor	Tempo(s)	$Gap_{MH}(\%)$	$Gap_{B\&P}(\%)$
26	10	50	50	282,12	282,12	282,12	282,12	0,32	0	0
27	10	50	25	244,54	244,54	244,54	244,54	1,26	0	0
28	10	50	50	288,33	288,33	288,33	288,33	0,32	0	0
29	10	50	25	108,98	108,98	108,98	108,98	1,35	0	0
30	10	50	50	157,48	157,48	157,48	157,48	0,37	0	0
31	10	50	25	32,25	32,25	32,25	32,25	0,52	0	0
32	10	50	50	36,66	36,66	36,66	36,66	0,41	0	0
33	10	100	25	403,18	403,18	403,18	403,18	4,59	0	0
34	10	100	50	296,53	296,53	296,53	296,53	2,3	0	0
35	10	100	25	388,87	388,87	388,87	388,87	4,19	0	0
36	10	100	50	294,8	294,8	294,8	294,8	2,12	0	0
37	10	100	25	178,28	178,28	190,39	178,28	1,96	0	0
38	10	100	50	175,96	175,96	175,96	175,96	2,48	0	0
39	10	100	25	57,5	57,5	57,5	57,5	1,45	0	0
40	10	100	50	31,89	31,89	31,89	31,89	1,3	0	0
41	10	200	25	735,27	735,27	735,27	735,27	4,06	0	0
42	10	200	50	512,16	506,06	506,06	**506,06**	2,62	-1,21	0
43	10	200	25	513	513	513	513	8,93	0	0
44	10	200	50	475,21	475,21	475,21	475,21	7,89	0	0
45	10	200	25	347,29	347,29	347,29	347,29	8,48	0	0
46	10	200	50	217,46	217,46	217,46	217,46	5,81	0	0
47	10	200	25	102,93	102,93	102,93	102,93	6,34	0	0
48	10	200	50	55,05	55,05	55,05	55,05	3,17	0	0
49	20	100	25	520,24	507,81	507,81	**507,81**	13,32	-2,45	0
50	20	100	50	420,64	406,65	406,65	**406,65**	11,25	-3,44	0
51	20	100	25	422,21	419,17	419,17	**419,17**	31,01	-0,73	0
52	20	100	50	360,86	360,86	360,86	360,86	27,07	0	0
53	20	100	25	245,17	245,17	246,22	245,17	22,15	0	0
54	20	100	50	185,06	185,06	185,06	185,06	20,6	0	0
55	20	100	25	52,52	52,52	52,59	52,52	5,78	0	0
56	20	100	50	19,05	19,05	19,05	19,05	3,61	0	0
57	20	200	25	903,84	875,46	875,46	**875,46**	58,38	-3,24	0
58	20	200	50	485,65	476,05	476,05	**476,05**	38,14	-2,02	0
59	20	200	25	616,93	606,8	610,28	**606,8**	57,29	-1,67	0
60	20	200	50	462,31	462,31	462,31	462,31	31,87	0	0
61	20	200	25	373,21	373,21	375,29	373,21	31,1	0	0
62	20	200	50	250,75	250,75	253,04	250,75	50,77	0	0
63	20	200	25	93,01	93,01	93,01	93,01	24,04	0	0
64	20	200	50	45,4	45,4	45,4	45,4	10,3	0	0
65	20	400	25	1323,35	1323,35	1323,35	1323,35	362,85	0	0
66	20	400	50	733,54	720,83	720,83	**720,83**	99,7	-1,76	0
67	20	400	25	975,12	975,12	980,87	975,12	285,06	0	0
68	20	400	50	614,67	614,67	615,41	614,67	224,87	0	0
69	20	400	25	763,76	763,76	768,69	763,76	121,13	0	0
70	20	400	50	298,47	298,47	301,93	298,47	128,4	0	0
71	20	400	25	239,58	239,58	239,58	239,58	151	0	0
72	20	400	50	84,49	84,49	84,49	84,49	49,07	0	0
73	40	200	25	831,94	831,94	808,69	**808,69**	89,03	-2,88	-2,88
74	40	200	50	593,35	593,35	556,04	**555,982**	119,118	-6,72	-6,72
75	40	200	25	728,44	728,44	727,66	**715,77**	201,08	-1,77	-1,77
76	40	200	50	481,05	481,05	475,65	**475,64**	200,37	-1,14	-1,14
77	40	200	25	339,75	339,75	341,2	339,75	103,7	0	0
78	40	200	50	273,88	273,88	275,91	273,88	218,77	0	0
79	40	200	25	76,77	76,77	76,77	76,77	29,61	0	0
80	40	200	50	58,46	58,46	58,46	58,46	25,24	0	0
81	40	400	25	1407,05	1394,23	1347,56	**1339,33**	455,37	-5,06	-4,1
82	40	400	50	858,8	858,8	834,44	**834,44**	166,83	-2,92	-2,92
83	40	400	25	891,02	891,02	895,97	**890,92**	497,29	-0,01	-0,01
84	40	400	50	757,42	757,42	746,67	**740,48**	439,9	-2,29	-2,29
85	40	400	25	586,29	586,29	593,18	586,29	430,54	0	0
86	40	400	50	395,95	395,95	401,93	401,22	197,72	1,31	1,31
87	40	400	25	195,33	195,33	195,33	195,33	188,34	0	0
88	40	400	50	70,77	70,77	70,77	70,77	72,95	0	0
89	40	800	25	2900,14	2900,14	2900,14	2900,14	1392,68	0	0
90	40	800	50	1345,7	1345,7	1309,76	**1309,76**	916,75	-2,74	-2,74

Tabela 1: Resultados computacionais (continuação)

ID	m	n	Q	MH	B&P	HILS média	HILS melhor	Tempo(s)	$Gap_{MH}(\%)$	$Gap_{B\&P}(\%)$
91	40	800	25	2200,57	2200,57	2221,32	2200,57	1293,71	0	0
92	40	800	50	1025,16	1025,16	1025,92	**1024,71**	979,83	-0,04	-0,04
93	40	800	25	1404,16	1404,16	1407,35	1404,16	1508,82	0	0
94	40	800	50	616,58	616,58	629,24	622,32	716,03	0,92	0,92
95	40	800	25	396,92	396,92	396,92	396,92	1686	0	0

5. Conclusões e Trabalhos Futuros

Nesse trabalho foi proposto um algoritmo para a resolução do problema de roteamento de ônibus escolares, que abrange a seleção de paradas, alocação de estudantes para tais paradas, e roteamento dos veículos visando a minimização dos custos. Foi elaborado um algoritmo baseado na metaheurística ILS e utilizando um modelo de programação linear inteira. Para a alocação de estudantes foi utilizado um algoritmo heurístico guloso aliado a um algoritmo exato baseado no problema de fluxo máximo.

Experimentos computacionais realizados em 95 das instâncias utilizadas por Schittekat et al. [2013] mostraram que o método proposto é bastante competitivo, sendo capaz de encontrar soluções melhores em 18 das instâncias, e tendo um *gap* médio de -0.42% em relação às melhores reportadas na literatura atual. Como trabalhos futuros, podem ser introduzidos novos elementos ao problema de modo à torná-lo mais realista. Tais elementos incluem, e.g., múltiplas escolas, janelas de tempo e tempo máximo de viagem.

Referências

Bennett, B. T. e Gazis, D. C. (1972). School bus routing by computer. *Transportation Research*, 6 (4):317–325.

Bögl, M., Doerner, K. F., e Parragh, S. N. (2015). The school bus routing and scheduling problem with transfers. *Networks*, 65(2):180–203.

Clarke, G. e Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581.

Ellegood, W. A., Solomon, S., North, J., e Campbell, J. F. (2019). School bus routing problem: Contemporary trends and research directions. *Omega*. Forthcoming.

Ezquerro Eguizábal, S., Moura Berodia, J. L., Ibeas Portilla, A., e Benavente Ponce, J. (2018). Optimization model for school transportation design based on economic and social efficiency. *Transport Policy*, 67:93–101.

Fleming, S. (2019). This US city put an algorithm in charge of its school bus routes and saved $5 million. World Economic Forum. `https://www.weforum.org/agenda/2019/08/this-us-city-put-an-algorithm-in-charge-of-its-school-bus-routes-and-saved-5-million/`. Online; accessed 26 April 2020.

Ford, L. R. e Fulkerson, D. R., editors (1962). *Flows in Networks*. Princeton University Press, Princeton, NJ.

Goldberg, A. V. e Tarjan, R. E. (1988). A new approach to the maximum-flow problem. *JACM: Journal of the Association for Computer Machinery*, 35(4):921–940.

Kinable, J., Spieksma, F. C. R., e Vanden Berghe, G. (2014). School bus routing—a column generation approach. *International Transactions in Operational Research*, 21(3):453–478.

Miranda, D. M., de Camargo, R. S., Conceição, S. V., Porto, M. F., e Nunes, N. T. (2018). A multi-loading school bus routing problem. *Expert Systems with Applications*, 101:228–242.

Newton, R. M. e Thomas, W. H. (1969). Design of school bus routes by computer. *Socio-Economic Planning Sciences*, 3(1):75–85.

Niasar, M. S. F., Talarico, L., Sajadifar, M., e Tayebi, A. H. (2017). Iterated local search algorithm with strategic oscillation for school bus routing problem with bus stop selection. *IJSOM: International Journal of Supply and Operations Management*, 4(1):1–14.

Park, J. e Kim, B.-I. (2010). The school bus routing problem: A review. *European Journal of Operational Research*, 202(2):311–319.

Sales, L. P. A., Melo, C. S., Bonates, T. O., e Prata, B. A. (2018). Memetic algorithm for the heterogeneous fleet school bus routing problem. *Journal of Urban Planning and Development*, 144(2):1–12.

Schittekat, P., Kinable, J., Sörensen, K., Sevaux, M., Spieksma, F., e Springael, J. (2013). A metaheuristic for the school bus routing problem with bus stop selection. *European Journal of Operational Research*, 229(2):518–528.

Shafahi, A., Wang, Z., e Haghani, A. (2018). Speedroute: Fast, efficient solutions for school bus routing problems. *Transportation Research Part B: Methodological*, 117:473–493.

Subramanian, A. (2012). *Heuristic, Exact and Hybrid Approaches for Vehicle Routing Problems*. PhD thesis, Universidade Federal Fluminense.