DESENVOLVIMENTO DE UM ALGORITMO DE BUSCA LOCAL ITERADA PARA O PROBLEMA DIAL-A-RIDE

Eduardo Motta de Oliveira
Universidade Federal do Espírito Santo
Av. Fernando Ferrari, 514, Goiabeiras — Vitória - ES
edumottaoliveira@gmail.com

André Renato Sales Amaral
Universidade Federal do Espírito Santo
Av. Fernando Ferrari, 514, Goiabeiras — Vitória - ES
amaral@inf.ufes.br

RESUMO

Este artigo considera o modelo matemático multiobjetivo de Mauri e Lorena (2009) para o problema *dial-a-ride*. Nesse modelo, a função de minimização multiobjetivo trata os custos de transporte e o desconforto dos clientes por meio de penalizações. Aqui, um algoritmo de Busca Local Iterada (ILS- *Iteraded Local Search*) é apresentado para resolver o modelo multiobjetivo. Resultados computacionais com base em um conjunto tradicional de instâncias da literatura comprovam a eficiência e o caráter competitivo do algoritmo ILS proposto.

PALAVRAS CHAVE. Dial-a-Ride, Roteamento de veículos, Busca Local Iterada.

Área Principal: Otimização Combinatória

ABSTRACT

This paper considers the multiobjective mathematical model of Mauri and Lorena (2009) for the dial-a-ride problem. In this model, the multiobjective minimization function deals with transportation costs and the discomfort of customers through penalties. Here, an Iterated Local Search (ILS- Iteraded Local Search) algorithm is presented to solve the multiobjective model. Computational results based on a traditional set of instances from the literature demonstrate the efficiency and the competitive nature of the proposed ILS algorithm.

KEYWORDS. Dial-a-Ride, Vehicle routing problem, Iterated Local Search.

Main Area: Combinatorial Optimization

1. Introdução

Há uma tendência global na adoção de políticas sociais de mobilidade urbana que busquem atender a parcela da população com debilidades físicas e que não possui condições de utilizar o transporte público tradicional. Há também a necessidade de estender essa preocupação às políticas na área da saúde, onde soluções específicas de transporte são necessárias para o atendimento de idosos e enfermos.

Os serviços nesse contexto que envolvem uma solicitação de transporte, geralmente por telefone, por parte do usuário, portador de necessidades especiais, são conhecidos como serviços *Dial-a-Ride* ou serviços de transporte Porta-a-Porta, em português, e o problema de otimização que surge na operação desses serviços é conhecido como *Dial-a-Ride Problem* (DARP).

O DARP consiste na elaboração de rotas e escalas de veículos para transportar diversos usuários, os quais fazem requisições de embarque e desembarque entre locais de origem e destino específicos. O objetivo do DARP é encontrar um conjunto de rotas para um número de veículos, com custo mínimo, capaz de acomodar a maior quantidade possível de usuários, sempre obedecendo a um conjunto de restrições (Cordeau e Laporte, 2007).

Em termos gerais, o DARP é uma generalização de vários problemas de roteamento de veículos como o *Pick-up and Delivery Vehicle Routing Problem* (PDVRP) e o *Vehicle Routing Problem with Time Windows* (VRPTW). O que faz o DARP diferente da maioria de tais problemas é a perspectiva humana, ou seja, a redução do desconforto dos usuários (qualidade do serviço), no transporte de passageiros, deve ser equilibrada com a minimização dos custos operacionais (Cordeau e Laporte, 2007).

A estrutura básica de um sistema *Dial-a-Ride* é descrita por Soldano e Valandro (2004):

- A *frota* é o grupo de veículos disponíveis, onde cada veículo está associado com um tempo de serviço, sua capacidade e o depósito a que pertence.

- As *solicitações* descrevem a demanda do usuário, incluindo pontos de origem e destino (embarque e desembarque), número de passageiros e o tempo mais cedo de partida (ou o mais tarde de chegada).

- As *restrições* consideram a capacidade máxima dos veículos, que não pode ser excedida, e a necessidade de visitar o ponto de embarque de cada solicitação antes de visitar seu ponto de desembarque. Além disso, algumas restrições asseguram aos consumidores uma qualidade mínima de serviço, evitando que esperem muito tempo antes de embarcarem e proporcionando a eles uma viagem com a menor duração possível.

- Quanto aos *objetivos*, o DARP envolve diferentes aspectos. Consumidores estão interessados em boa qualidade de serviço, a empresa que fornece o transporte espera baixo custo de operação, enquanto a administração pública está interessada na disponibilização do serviço em todos os lugares da região em que atua. Por essas razões, diferentes objetivos conflituosos podem ser considerados: a maximização da qualidade de serviço, a minimização do número de veículos usados, a redução das rotas e a maximização do número de consumidores servidos.

De acordo com Mauri e Lorena (2009), uma tendência comum nos modelos do DARP é deixar que os usuários determinem uma *janela de tempo* (isto é, o período de tempo antes e depois de um horário desejado) para sua partida e sua chegada.

Serviços *Dial-a-ride* podem operar de acordo com uma abordagem estática ou dinâmica. No primeiro caso, todas as solicitações de transporte são conhecidas previamente, enquanto no segundo caso, todas as solicitações de transporte são reveladas ao longo do dia e as rotas dos veículos são ajustadas em tempo real para ir de encontro à demanda. Na prática, DARPs puramente dinâmicos raramente existem, pois um conjunto de solicitações é geralmente conhecido com

antecedência. Além disso, Cordeau e Laporte (2003a) afirmam que, mesmo quando a abordagem dinâmica é utilizada, um problema estático geralmente é resolvido com um conjunto de solicitações iniciais para obter uma solução inicial que será modificada mais tarde quando novas solicitações forem recebidas.

Os algoritmos desenvolvidos para o DARP nas últimas décadas são heurísticos em sua maioria. Isso porque a complexidade do problema é muito alta para que soluções em programação linear inteira ou mista sejam usadas em contextos reais. Além disso, por se tratar de um problema *NP-Hard*, que generaliza o Problema do Caixeiro Viajante com Janelas de Tempo (*Traveling Salesman Problem with Time Windows - TSPTW)*, o uso de heurísticas se faz necessário para encontrar boas soluções, sob suas diversas variações. Heurísticas são úteis também porque as restrições de janelas de tempo fazem o problema extremamente não-convexo, criando dificuldades para o encontro de soluções viáveis (Cubillos *et al.*, 2009).

Ademais, o algoritmo deve ser rápido o suficiente para ser executado em tempo hábil. Cordeau e Laporte (2003b) citam dois importantes motivos para essa necessidade. Primeiro, há vários contextos onde o tamanho do problema é muito grande (o número de solicitações por dia em algumas cidades europeias e norte americanas chega ultrapassar 2000). Segundo, enquanto é sensato executar um algoritmo por algumas horas em um contexto estático, uma resposta muito mais rápida é necessária em ambientes dinâmicos. Na verdade, sempre que um usuário solicita um transporte por telefone ou internet, o operador do sistema deveria estar preparado para informar, em alguns segundos, se a solicitação pôde ser incluída. Além do mais, se a solicitação for aceita, ela deveria ser adicionada às rotas dos veículos em um tempo relativamente curto (não mais que dez minutos, por exemplo).

Devemos lembrar, também, que esse tipo de transporte é custoso e a administração dos veículos requer o máximo de eficiência possível, embora o número de solicitações incluídas no planejamento das rotas possa variar dependendo da solução usada (Deleplanque *et al.*, 2013).

O presente artigo assemelha-se ao trabalho de Mauri e Lorena (2009) pois aborda o DARP com as seguintes características: estático, com janelas de tempo, foco multiobjetivo (minimizar custos operacionais e o desconforto do cliente), frota homogênea (todos os veículos com as mesmas características) e garagem única.

Entretanto, a meta-heurística aqui utilizada para tratar o problema é a ILS, enquanto Mauri e Lorena utilizaram a meta-heurística simulated annealing. A abordagem aqui apresentada também difere da apresentada por Mauri e Lorena (2009) na forma de geração de solução inicial, na implementação da heurística de programação, e nas formas de geração de vizinhança.

Na próxima seção é apresentada uma formulação geral para o problema e na seção seguinte a metodologia de solução adotada. Por fim, são mostrados os resultados dos testes computacionais.

2. Formulação Matemática do DARP

A estrutura de uma solução do DARP é formada pelo conjunto de rotas atribuída aos veículos, onde cada veículo está vinculado a suas garagens de origem e destino, e também, a uma lista particular de pontos de embarque e desembarque de clientes.

Cordeau (2006; 2007) propôs um modelo matemático que é usado como referência em muitos trabalhos que tratam do DARP. Mauri e Lorena (2009) adaptaram o modelo para realizar uma abordagem multiobjetivo, a qual é descrita abaixo.

- n representa o número de usuários ou solicitações de viagem.
- $G = (N, A)$ é o grafo dirigido, com $N = P \cup D \cup \{0, 2n + 1\}$, onde:
  - $P = \{1, ..., n\}$ representa o conjunto dos nós de embarque ou coleta (*pick-up*);
  - $D = \{n + 1, ..., 2n\}$ representa o conjunto dos nós de desembarque ou entrega (*drop-off*);

  - $\{0, 2n+1\}$ representa as garagens de origem e destino.

- K é o conjunto que contém cada veículo k;
- A cada usuário/vértice i é associado(a):
  - um nó de embarque/origem $i \in P$;
  - um nó de desembarque/destino $n+i \in D$;
  - uma duração do serviço d_i (tempo gasto no embarque, com o veículo parado), sendo $d_0 = d_{2n+1} = 0$;
  - uma lotação q_i, com $q_0 = q_{2n+1} = 0$, e $q_i = -q_{n+i} (i = 1, ..., n)$;
  - uma *janela de tempo* $[e_i, l_i]$, onde e_i e l_i representam, respectivamente, o momento mais cedo (*earliest*) e mais tarde (*latest*) entre os quais o serviço deve começar em i.
- c_{ij}^k é o custo para percorrer o arco (i, j) com o veículo k, onde t_{ij} representa o tempo gasto nessa viagem;
- x_{ij}^k é a variável que assume 1 se, e somente se, o arco (i, j) é percorrido pelo veículo k. Caso contrário, seu valor é 0;
- Q_i^k é a carga (número de pessoas) dentro do veículo k após visitar i;
- L_i^k representa o tempo total de viagem (*ride time*) de i com o veículo k.

Com respeito à programação dos tempos de atendimento, uma solução deverá conter, para cada vértice, as seguintes informações:

- A_i^k é o horário que o veículo k chega ao vértice i;
- W_i é o tempo de espera entre a chegada ao local e o início do atendimento no vértice i;
- B_i^k representa o momento em que o veículo k começa a servir o usuário/vértice i;
- D_i é o horário de partida (*departure time*) do vértice i, que coincide com o momento do fim do serviço iniciado em B_i^k.

Para estabelecer métricas e parâmetros de avaliação da qualidade da solução, são definidos os limites máximos aceitáveis para alguns aspectos do serviço. Assumindo que a frota é homogênea, temos que:

- L'' representa o tempo máximo de viagem de um usuário;
- W'' é o tempo máximo que cada veículo pode esperar antes de começar a servir um usuário;
- Q'' é a capacidade máxima de cada veículo;
- T'' é a duração máxima da rota de cada veículo.

O custo da solução é calculado por uma função que atribui pesos, ou penalizações, aos requisitos do serviço que merecem ser enfatizados. Mauri e Lorena (2009) dividem esses requisitos em duas categorias: *não essenciais* e *essenciais*.

Os requisitos não essenciais são aqueles que não definem a viabilidade da solução mas a qualidade a ser buscada. Esses requisitos são usados para verificar o nível de qualidade de atendimento ao cliente e de custo do serviço e são relacionados à distância total percorrida pelos veículos,

ao número de veículos utilizados na solução do problema, à duração das rotas, ao tempo de viagem dos clientes e ao tempo de espera nos locais de embarque e desembarque.

Os requisitos essenciais, por outro lado, são aqueles que não podem ser violados, pois conflitam com as premissas básicas do problema e restrições encontradas nesse tipo de serviço. Portanto, requisitos essenciais envolvem as restrições de janela de tempo para o início do serviço em cada cliente, tempo máximo de cada rota, tempo máximo de viagem de cada cliente, tempo máximo de espera permitido e capacidade máxima dos veículos. Por serem críticos, às violações desses requisitos são atribuídos pesos muito maiores que os aplicados aos requisitos não essenciais.

O modelo apresentado por Mauri e Lorena (2009) é descrito a seguir.

Minimizar

$$\omega_0 \sum_{k\in K} max\{0,(B_0^k - D_{n+1}^k) - T''\} + \omega_1 \sum_{i\in P}\sum_{k\in K} max\{0, L_i^k - L''\} + \omega 2 \sum_{i\in\{P\cup D\}} max\{0, W_i - W''\} + \tag{1}$$

$$\omega_3 \sum_{k\in K} max\{0, \sum_{i\in P\cup D}(Q_i - Q'')\} \; + \; \omega_4 \sum_{i\in N}(max\{0, e_i - B_i\} \; + \; max\{0, B_i - l_i\}) + \tag{2}$$

$$\beta_0 \sum_{k\in K}\sum_{i\in N}\sum_{j\in N; j\neq i} c_{ij}^k x_{ij}^k + \beta_1 \sum_{k\in K}\sum_{i\in P} x_{0i}^k + \beta_2 \sum_{k\in K}(B_{n+1}^k - D_0^k) + \beta_3 \sum_{i\in P} L_i^k + \beta_4 \sum_{i\in\{P\cup D\}} W_i \tag{3}$$

sujeito a

$$\sum_{k\in K}\sum_{j\in N} x_{ij}^k = 1 \qquad (i \in P), \tag{4}$$

$$\sum_{i\in N} x_{0i}^k = \sum_{i\in N} x_{i,2n+1}^k = 1 \qquad (k \in K), \tag{5}$$

$$\sum_{j\in N} x_{ij}^k - \sum_{j\in N} x_{n+i,j}^k = 0 \qquad (i \in P,\ k \in K), \tag{6}$$

$$\sum_{j\in N} x_{ji}^k - \sum_{j\in N} x_{ij}^k = 0 \qquad (i \in P \cup D,\ k \in K), \tag{7}$$

$$B_j^k \geq (B_i^k + d_i + t_{ij}) x_{ij}^k \qquad (i \in N, j \in N, k \in K), \tag{8}$$

$$Q_j^k \geq (Q_i^k + q_j) x_{ij}^k \qquad (i \in N,\ j \in N,\ k \in K), \tag{9}$$

$$L_i^k \geq B_{n+1}^k - (B_i^k + d_i) \qquad (i \in P,\ k \in K), \tag{10}$$

$$B_{2n+1}^k - B_0^k \leq T_k \qquad (k \in K), \tag{11}$$

$$e_i \leq B_i^k \leq l_i \qquad (i \in N,\ k \in K), \tag{12}$$

$$t_{i,n+i} \leq L_i^k \leq L'' \qquad (i \in P,\ k \in K), \tag{13}$$

$$max\{0, q_i\} \leq Q_i^k \leq min\{Q'', Q'' + q_i\} \qquad (i \in N,\ k \in K), \tag{14}$$

$$x_{ij}^k \in \{0,1\} \qquad (i \in N,\ j \in N,\ k \in K), \tag{15}$$

A duas primeiras partes da função objetivo visam minimizar os requisitos essenciais do problema: a primeira parte (1) visa minimizar as violações dos tempos máximos de duração das rotas, de viagem dos clientes e de espera; enquanto a segunda parte (2) visa minimizar as violações de capacidade do veículo e das janelas de tempo. A terceira parte (3) busca minimizar os requisitos não essenciais: a distância total percorrida pelos veículos, o número de veículos usados, o

tempo total de duração das rotas, o tempo total de viagem dos clientes e o tempo total de espera dos veículos, respectivamente. Os pesos atribuídos aos requisitos essenciais e não essenciais são identificados por $\{\omega_0, \omega_1, \omega_2, \omega_3, \omega_4\}$ e $\{\beta_0, \beta_1, \beta_2, \beta_3, \beta_4\}$, respectivamente.

A Equação (4) garante que cada solicitação seja atendida somente uma vez e somente por um único veículo, enquanto a (5) garante que cada rota de um veículo comece numa garagem de origem e termine em uma garagem de destino. A Equação (6) faz com que os nós de origem e destino sejam visitados pelo mesmo veículo e a Equação (7), que o número de veículos que chega a um nó seja o mesmo que sai desse nó.

A Restrição (8) garante a consistência relacionada ao tempo enquanto a (9) garante a consistência relacionada à capacidade dos veículos. Neste caso, a quantidade de pessoas no veículo k, após sair do nó j, é maior ou igual ao número de pessoas que estavam no veículo antes de j mais o número de pessoas que embarcaram em j.

A Restrição (10) define o tempo de viagem de cada usuário, enquanto a (11) limita a duração da rota do veículo k. A Restrição (12) garante que o início do serviço ocorra dentro dos limites da janela de tempo definida. A Restrição (13) garante que a duração de uma viagem esteja entre o tempo de deslocamento entre os nós de origem e destino e o tempo máximo de viagem previamente definido; e a Restrição (14) limita os valores que a variável Q_i^k pode assumir. Finalmente, a Restrição (15) define os valores possíveis para a variável x_{ij}^k.

Cada solicitação de transporte é caracterizada por um número de identificação, uma localização (como coordenadas geográficas, por exemplo), um tempo de serviço, um valor de demanda (número de passageiros) e uma janela de tempo.

3. Metodologia de solução

Primeiramente, uma solução inicial incompleta é gerada, na qual ainda não há a atribuição dos tempos a cada ponto de parada da rota, portanto, uma *heurística de programação* adaptada de Mauri e Lorena (2009) é utilizada para definir esses valores. Com todos os atributos da solução definidos, é executada uma função de avaliação, que calcula o custo da solução levando em consideração os pesos para atributos que impactam negativamente na qualidade do resultado (e.g. tempo de espera, distância total percorrida, tempo total de viagem do cliente) e as penalizações para atributos que violem os requisitos básicos de viabilidade (e.g. excesso na capacidade dos veículos, total dos tempos que violam as janelas de tempo).

A solução inicial criada é usada como entrada para um algoritmo ILS, que realiza buscas locais seguidas de perturbações para evitar que a busca permaneça estagnada em mínimos locais. O critério de parada se baseia no número de iterações sem melhora. O retorno do algoritmo ILS é a melhor solução encontrada em seu espaço de busca.

É interessante destacar que o uso de meta-heurísticas torna a resolução do problema menos árdua, porém permite que soluções inválidas sejam obtidas. Soluções com um pequeno grau de inviabilidade podem ser aceitas em alguns casos reais onde problemas de maior porte não são resolvidos por métodos exatos (Cordeau, 2006).

3.1. Solução inicial

O método aqui aplicado para construir uma solução inicial incompleta considera a lista de solicitações de transporte incluída como entrada. Nessa lista, cada solicitação é formada pelo par embarque e desembarque, que deve ser sempre manipulado respeitando sua ordem de precedência. Inicialmente, para cada veículo do problema é criada uma rota contendo apenas sua garagem de origem e destino, que no nosso caso será sempre a mesma garagem em virtude do modelo adotado de garagem única. Em seguida, os nós de embarque e desembarque de cada solicitação são atribuídos e inseridos nas rotas respeitando a algum modelo específico de inserção. Por último, a fim de se obter uma solução inicial completa, uma heurística de programação é executada, estabelecendo os tempos de cada nó para o cumprimento da rotas.

Nesse trabalho, foram testados dois modelos de inserção das solicitações:

- Inserção Aleatória: São feitas as escolhas aleatórias de uma rota e de duas posições consecutivas dentro dessa rota e a seguir inserem-se os nós de embarque e desembarque nos espaços respectivos.

- Inserção Criteriosa: Ordena-se a lista de solicitações cronologicamente pelo tempo médio da janela de tempo. Em seguida, insere-se a primeira solicitação de embarque da lista de solicitações ao final da rota cujo último vértice é mais próximo, em termos de distância euclidiana. Ao final, o nó de desembarque associado à solicitação do cliente atual é inserido na sequência, proporcionando o menor tempo de viagem possível para esse cliente.

  A Inserção Criteriosa cria uma solução inicial com uma drástica redução no número de violações de uma solução, se comparado ao método aleatório, o que facilita a convergência para a melhor solução tende a otimizar o processo de busca.

Observa-se que soluções inválidas podem ser obtidas, contudo qualquer inviabilidade é penalizada na função objetivo e, portanto, tende a desaparecer ao longo das iterações do algoritmo ILS.

3.1.1. Heurística de programação

A solução do problema não se limita apenas à determinação da ordem de atendimento dos clientes e da distribuição deles nos veículos. A solução completa deve estabelecer também os intervalos de tempo de chegada, partida, início e fim do atendimento em cada nó, além da ocupação dos veículos em cada nó.

Dessa forma, uma heurística de programação é necessária para computar esses valores para a solução inicial e para recomputar esses valores a cada nova solução gerada no processo de busca. O cálculo desses tempos para cada rota baseia-a se no modelo matemático apresentado por Mauri e Lorena (2009), onde, para cada nó atendido i $(\forall i \in N)$ podem ser definidos:

- Horário de chegada dos veículos: $A_i = 0$ se $i \in \{0\}$ e $A_i = D_{i-1} + t_{i-1,i}$ se $i \in \{P \cup D \cup \{0\}\}$;

- Horário de partida dos veículos: $D_i = 0$ se $i \in \{2n+1\}$ e $D_i = B_i + d_i$ se $i \in \{P \cup D\}$ e $D_i = B_i$ se $i \in \{2n+1\}$;

- Horário de início do serviço: $B_i = D_i$ se $i \in \{0\}$ e $B_i = max\{e_i, A_i\}$ se $i \in \{P \cup D \cup \{2n+1\}\}$;

- Tempo de espera antes do início do serviço:$W_i = 0$ se $i \in \{0\}$ e $W_i = B_i - A_i$ se $i \in \{P \cup D \cup \{2n+1\}\}$;

- Carga (número de assentos ocupados) após o término do serviço: $Q_i = 0$ se $i \in \{0, 2n+1\}$ e $Q_i = Q_{i-1} + q_i$ se $i \in \{P \cup D\}$;

- Tempo de viagem do cliente é $L_i = B_{n+i} - D_i$.

É importante notar que a heurística de programação busca adiantar o atendimento nos nós o máximo possível ($B_i = max\{e_i, A_i\}$), deslocando o horário de início do serviço em cada nó para o mais próximo do início de sua janela de tempo. A justificativa está no fato de isso proporcionar uma folga de tempo que pode ser necessária para uma melhor disposição dos tempos de atendimento nos nós seguintes.

No entanto, adiantar o início dos serviços pode criar longos tempos de espera entre os atendimentos, o que prejudicaria o custo da solução. Por essa razão, após a programação inicial é necessário procurar atrasar o início do serviço nos pontos iniciais da rota de forma a minimizar essa espera. O conceito de *Forward Time Slack*, proposto inicialmente por Savelsbergh (1992), foi então

empregado para definir o máximo atraso possível do início do serviço em cada ponto de forma a não violar as restrições de sua janela de tempo. Assim, os atendimentos dos clientes são atrasados gradualmente a partir do início da rota sem geração de novas violações.

3.2. Função objetivo

A função objetivo foi apresentada na seção 2 e fornece um meio de avaliar a qualidade de cada nova solução gerada pela ILS.

É importante enfatizar que a avaliação completa de custo das soluções, salvo a da solução inicial, deve ser evitada. Como as soluções vizinhas geradas e analisadas diferem em poucos aspectos, faz sentido apenas executar a função de avaliação nas rotas que sofreram modificações e assim reduzir o número necessário de ciclos de processamento em cada iteração do processo de busca.

3.3. Estrutura de vizinhança

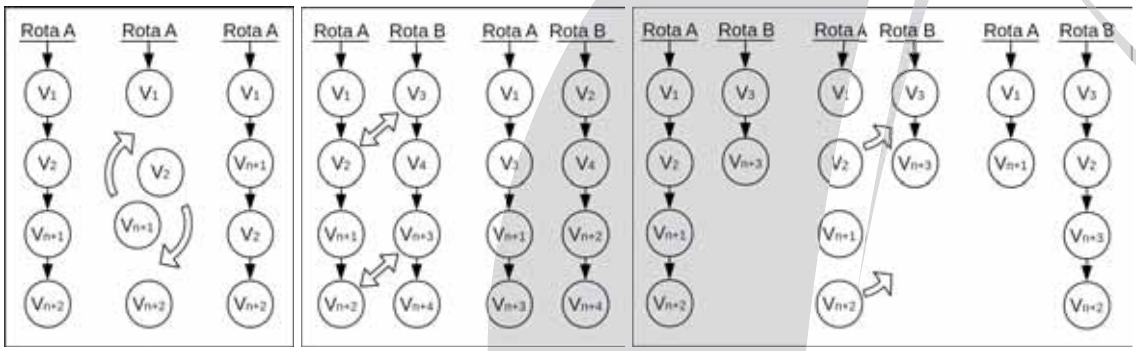
A procura por uma solução melhor no DARP envolve o teste de soluções vizinhas. Mauri e Lorena (2009) fazem uso de três movimentos para a geração de vizinhanças: reordenação, realocação e troca de nós entre rotas. Segundo eles, esses movimentos são baseados em outros encontrados frequentemente nos trabalhos referentes ao DARP (Cordeau e Laporte, 2003; Jorgensen, 2007; Savelbergh, 1992).

O movimento de *reordenação* consiste em recolocar um nó qualquer de uma rota qualquer em outra posição da mesma rota sem violar as relações de precedência entre embarque e desembarque.

O movimento de *troca* é aplicado aos nós de dois clientes atendidos por duas rotas diferentes. As rotas e os clientes são selecionados aleatoriamente e seus pares de nós são permutados fazendo com que os clientes troquem de posição e de rota.

O movimento de *realocação* é executado sobre duas rotas quaisquer removendo os nós de embarque e desembarque de algum cliente de uma das rotas e reinserindo-os em posições aleatórias na outra rota.

Esses movimentos podem gerar soluções inválidas. Nesse caso, essas soluções são penalizadas na função objetivo.



(a) (b) (c)

Figura 1: Exemplos dos movimentos. (a) reordenação. (b) troca. (c) realocação.

3.4. Busca Local

Também conhecida como método de descida de colina, a Busca Local se propõe a encontrar os ótimos locais gerando, a cada iteração, soluções vizinhas com base na melhor solução atual e aceitando somente as soluções melhores que a atual ou as que atendam a um determinado critério de aceitação. Ao fim do processo, a melhor solução encontrada é retornada.

Nesse artigo, foram implementadas duas versões de Busca Local, que se diferenciam no método de geração de vizinhanças. A primeira versão implementa a geração de vizinhanças de forma totalmente aleatória: a cada iteração, um movimento é selecionado aleatoriamente e aplicado

a um conjunto de rotas, posições e clientes que também são selecionados aleatoriamente. A segunda versão, no entanto, permite uma análise abrangente da vizinhança: a partir de escolhas aleatórias de movimento, clientes e rotas, são geradas todas as movimentações possíveis de clientes sobre as rotas, buscando cobrir uma área maior da vizinhança.

As duas funções de busca local são finalizadas quando é detectado um número preestabelecido de iterações sem melhoria da qualidade da solução.

3.5. Busca Local Iterada

A Busca Local Iterada, ou *Iterated Local Search* (ILS), é uma meta-heurística simples e eficiente. A partir de uma solução inicial ela gera um ótimo local e, em cada iteração, executa uma perturbação seguida de uma busca local na solução perturbada, produzindo um novo ótimo local. As perturbações são funções que alteram a solução de forma a gerar soluções vizinhas não tão próximas que possam cair no mesmo ótimo local nem tão distantes que possam se comportar como aleatórias, gerando uma busca local tradicional a partir de uma nova solução aleatória inicial. Na prática, O algoritmo ILS consiste em uma sequência de buscas locais intercaladas por perturbações. Seu pseudo-código é dado pelo Algoritmo 1.

Algoritmo 1: ITERATED LOCAL SEARCH

```
   Entrada: s_0                                          /* Solução inicial */
   Saída: s                                  /* Melhor solução encontrada */
1  início
2      BuscaLocal(s_0)
3      s ← s_0
4      custo ← funcaoCusto(s)
       /* Mantém loop se o número máximo de iterações sem melhora foi
       ultrapassado                                                      */
5      while numIter < numMaxIter do
6          s* ← s
7          perturbacao(s*)
8          BuscaLocal(s*)
9          custo* ← funcaoCusto(s*)
10         numIter ← numIter + 1
11         if custo* <= custo then
12             s ← s*
13             numIter ← 0
14         end
15     end
16     return s
17 fim
```

O critério de parada é definido pelo número de iterações sem melhora. Quando uma sequência grande de iterações não consegue reduzir o custo da solução o algoritmo é interrompido e retorna melhor solução obtida durante sua execução.

A linha 7 chama uma *função de perturbação* que realiza um movimento diferente daqueles usados na geração de vizinhanças. Semelhantemente a um movimento 2-opt, um nó é escolhido aleatoriamente e permutado com o nó seguinte de atendimento dentro da rota, criando uma pequena, porém distinta perturbação que tende a ser mais difícil de reverter durante a busca local e, portanto, tende a direcionar a busca para regiões alternativas. Os cuidados necessários para a eficácia desse movimento são a proibição de permutações que desrespeitem as relações de precedência entre embarque e desembarque e a aplicação de um nível de perturbação proporcional ao tamanho da solução, o que é conseguido com sua aplicação em quantidade proporcional ao número de clientes ou veículos do problema.

4. Resultados computacionais

Para os experimentos foram usadas as instâncias de Cordeau e Laporte (2003). Essas são diversificadas e apresentam casos com até 13 veículos e 144 clientes sendo, por isso, bastante

empregadas como base de comparação para vários trabalhos da literatura sobre o DARP.

Os testes computacionais foram executados em um computador Intel Core i5-2400 com 4GB de memória RAM. Todo o código foi produzido na linguagem C e o sistema Ubuntu 14.04 - 64 bits foi usado para compilação e execução.

A fim de realizar uma comparação com o trabalho de Mauri e Lorena (2009) os pesos na função objetivo foram definidos da mesma forma que a sugerida por esses autores (i.e. a cada cada um dos requisitos essenciais foi definido o peso de 1500 e para os requisitos não essenciais os pesos foram distribuídos da seguinte maneira: peso 8 para a distância total percorrida, peso 1 para a duração total das rotas e para o tempo total de espera e peso 3 para o tempo total de viagem dos clientes).

A condição de parada do algoritmo ILS foi definida pela detecção de 15 iterações sem melhora.

Foram implementadas quatro variações do algoritmo principal, cada uma das quais foi executada 15 vezes para cada instância. A Tabela 1 contém os custos das soluções obtidas e a descrição dos resultados em cada coluna, onde:

- M&L: Representa os melhores resultados conseguidos por Mauri e Lorena(2009), que utilizam o *Simulated Annealing* em conjunto com técnicas de distribuição e programação semelhantes ao deste artigo;
- A: Corresponde à execução do algoritmo com a geração da solução inicial por inserção criteriosa e a busca local com vizinhança aleatória;
- B: Corresponde à execução do algoritmo com solução inicial aleatória e a busca local com vizinhança aleatória;
- C: Corresponde à execução do algoritmo com a geração da solução inicial por inserção criteriosa e a busca local com análise abrangente da vizinhança;
- D: Corresponde à execução do algoritmo com solução inicial aleatória e a busca local com análise abrangente da vizinhança.

Observando os valores em negrito na Tabela 1 podemos considerar que a segunda variante do algoritmo ILS (B) produziu o maior número de melhores soluções. Também é possível identificar que o total na última linha favorece a primeira variante (A), com uma pequena redução de custos de 244.480,95 (dado por M&L) para 240.207,96.

As duas últimas variantes (C e D) também obtiveram resultados vantajosos com relação aos resultados do trabalho de Mauri e Lorena (2009), que já era capaz de superar os resultados de Cordeau e Laporte (2003) e Jorgensen *et al.* (2007). No entanto, elas apresentaram resultados inferiores, em termos gerais, às duas primeiras variantes.

Os tempos de execução variaram entre 0,49 segundos (pr01, variante C) a 108,02 segundos (pr16, variante B). As médias dos tempos de execução do conjunto de instâncias para cada variante foram, em segundos: 30,60 (A), 32,06 (B), 16,73 (C) e 17,55 (D). Os tempos de execução no trabalho de Mauri e Lorena (2009) para mesmas instâncias não podem ser comparados diretamente com os tempos do algoritmo ILS pois foram obtidos em um hardware diferente.

Adotando a segunda variante do algoritmo ILS (B) e comparando o seu resultado detalhadamente com os de Mauri e Lorena na Tabela 2 é perceptível que o método aqui proposto possibilitou a redução dos tempos de espera e dos tempos de viagem.

Observa-se da Tabela 2 que o total do tempo de viagem piorou de 11.302,23 para 11.745,71, porém isso equivale a uma diferença de apenas 3,92%. Vê-se também que a distância teve uma leve piora de 2,57%. Em compensação o total do tempo de espera foi reduzido de 2.082,41 (M&L) para

Instância	Número de veículos	Número de clientes	Melhor custo total obtido				
			M&L	A	B	C	D
pr01	3	24	**3.677,91**	3.683,06	3.683,06	3.683,06	3.683,06
pr02	5	48	7.017,34	**6.736,14**	7.026,23	6.848,75	6.764,55
pr03	7	72	11.873,76	**11.777,33**	11.872,40	11.782,09	11.848,83
pr04	9	96	13.725,92	13.440,90	**13.361,57**	13.443,61	13.506,28
pr05	11	120	15.736,66	15.424,75	**15.355,88**	15.842,20	15.735,05
pr06	13	144	20.465,39	19.755,09	**19.732,66**	20.215,30	20.200,94
pr07	4	36	5.610,05	5.538,53	**5.535,19**	5.555,80	5.557,23
pr08	6	72	11.343,19	11.178,02	11.233,52	**10.951,26**	11.010,31
pr09	8	108	**15.632,09**	16.360,27	17.023,64	17.272,91	16.770,04
pr10	10	144	22.430,00	**21.422,83**	21.429,14	21.989,03	22.140,44
pr11	3	24	3.379,74	**3.355,78**	3.339,69	3.356,98	**3.355,78**
pr12	5	48	5.889,56	5**.770,95**	5.774,73	5.810,89	5.772,90
pr13	7	72	11.006,12	10.821,40	**10.757,97**	10.878,19	10.831,36
pr14	9	96	12.807,87	**12.482,34**	12.552,40	12.650,70	12.640,24
pr15	11	120	14.544,13	14.207,21	**14.064,74**	14.488,02	14.285,25
pr16	13	144	18.518,82	18.291,03	**18.037,87**	18.393,58	18.506,09
pr17	4	36	5.136,37	5.150,72	5.159,55	5.077,09	**5.074,72**
pr18	6	72	10.703,17	10.406,32	**10.365,7**9	10.382,66	10.447,71
pr19	8	108	15.013,71	14.786,70	**14.727,11**	14.947,73	20.481,29
pr20	10	144	1**9.969,15**	19.618,59	20.255,43	20.356,89	20.204,22
Total:	-	-	244.480,95	**240.207,96**	241.288,57	247.334,31	243.165,83

Tabela 1: Resultado da execucão dos algoritmos de Mauri e Lorena (2009) e das quatro variantes do algoritmo ILS para as 20 instâncias de Cordeau e Laporte (2003). Melhores valores estão representados em negrito.

1.444,28, ou seja, uma redução de 30,64%, o que proporcionou também uma redução da duração das rotas de 2,13%.

Os resultados das instâncias pr04, pr06, pr07, pr08, pr13, pr14 e pr18 não foram apresentados no trabalho de Mauri e Lorena e, por isso, não puderam ser usados como base de comparação. Porém, na análise dos custos totais, que consideram todos os aspectos essenciais e não essenciais do problema, percebemos que o algoritmo apresentado obteve uma melhora em 15 das 20 instâncias e que houve uma redução no total dos custos de 244.480,95 (M&L) para 241.288,57, ou seja, uma redução de 1,3%.

5. Conclusão

Neste artigo, a Busca Local Iterada foi utilizada para lidar com o problema *dial-a-ride* (DARP) multiobjetivo de forma simples e eficiente.

Foram testadas variações de métodos de geração de soluções iniciais e geração de vizinhanças. Os sistemas diferentes de geração de solução inicial não interferiram de modo significativo nos resultados finais. Já as variantes de geração de vizinhança afetaram significativamente os resultados.

Os experimentos computacionais mostraram uma redução dos custos da grande maioria das instâncias, se comparados com os resultados de Mauri e Lorena (2009), graças à grande redução dos tempos de espera dos veículos, o que implicou também na diminuição da duração das rotas. Assim, os resultados comparativos demonstram que a abordagem adotada e o uso da Busca Local Iterada formam uma alternativa extremamente competitiva para a resolução do DARP.

Trabalhos futuros devem se concentrar em melhorar a eficiência do algoritmo apresentado, assim como, permitir o tratamento de instâncias mais complexas, com garagens múltiplas ou que envolvam casos extremamente inviáveis, onde nem todos os usuários podem ser atendidos sem violar as restrições básicas.

6. Agradecimentos

Eduardo Motta de Oliveira agradece a bolsa de Iniciação Científica da UFES 2014/2015. Este trabalho teve o apoio financeiro da FAPES (processo nº 56145659/11).

Instância	Custo total		Distância		Duração		Tempo de Espera		Tempo de Viagem	
	M&L	B	M&L	B	M&L	B	M&L	B	M&L	B
pr01	**3.677,91**	3.683,06	252,79	273,70	831,30	863,65	98,51	109,95	241,93	173,29
pr02	**7.017,34**	7.026,23	437,45	435,28	1.992,34	1.961,60	594,90	562,67	310,17	330,19
pr03	11.873,76	**11.872,40**	831,74	882,22	2.404,67	2.454,46	132,93	183,53	894,08	862,32
pr05	15.736,66	**15.355,88**	1.085,45	1.107,87	3.920,25	3.655,17	434,81	187,22	899,35	989,96
pr09	**15.632,09**	17.023,64	1.064,23	1.095,94	3.258,66	3.193,07	34,42	7,17	1.275,06	1.872,07
pr10	22.430,00	**21.429,14**	1.392,09	1.449,50	4.475,42	4.484,12	203,33	164,20	2.204,85	1.753,81
pr11	3.379,74	**3.339,69**	251,85	267,99	738,42	742,17	6,57	5,26	206,66	178,97
pr12	5.889,56	**5.774,73**	436,69	433,22	1.428,44	1.393,45	31,75	3,88	311,95	313,61
pr15	14.544,13	**14.064,74**	1.010,09	1.045,50	3.654,02	3.489,92	243,94	100,91	855,16	853,95
pr16	18.518,82	**18.037,87**	1.289,31	1.319,29	4.318,33	4.180,19	149,02	38,17	1.245,66	1.241,11
pr17	**5.136,37**	5.159,55	375,67	371,61	1.095,67	1.123,56	0,00	11,18	345,10	295,28
pr19	15.013,71	**14.727,11**	1.041,09	1.064,02	3.315,28	3.238,89	114,19	49,83	1.085,18	1.068,64
pr20	**19.969,15**	20.255,43	1.414,65	1.417,48	4.332,69	4.222,23	38,04	20,31	1.427,08	1.812,51
Total:	244.480,95	241.288,57	10.883,10	11.163,62	35.765,49	35.002,48	2.082,41	1.444,28	11.302,23	11.745,71

Tabela 2: Comparação dos resultados de Mauri e Lorena (2009) e da Busca Local Iterada com solução inicial aleatória e estrutura de vizinhança aleatória.

Referências

Cordeau, J.-F., A Branch-and-Cut Algorithm for the Dial-a-Ride Problem, *Operations Research*, v. 54, n. 3, p. 573-586, 2006.

Coudeau, J.-F; Laporte, G., A tabu search heuristic for the static multi-vehicle dial-a-ride problem, *Transportation Research Part B: Methodological*, v. 37, n. 6, p. 579-594, 2003.

Coudeau, J.-F; Laporte, G., The dial-a-ride problem: variants, modeling issues and algorithms, *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, n. 1, p. 89-101, 2003.

Coudeau, J.-F; Laporte, G., The dial-a-ride problem: models and algorithms, *Annals of Operations Research*, v.153, n. 1, p. 29-46, 2007.

Cubillos C.; Urra E.; Rodríguez N.,Application of Genetic Algorithms for the DARPTW Problem, *Int. J. of Computers, Communications and Control*, v. 4, n. 2, 127-136, 2009.

Deleplanque, S.; Derutim, J.-P; Quilliot, A., Anticipation in the Dial-a-Ride Problem: an introduction to the robustness, *Proceedings of the 2013 Federated Conference on Computer Science and Information Systems*, p. 299-305, 2013.

Jorgensen, R. M.; Larsen, J.-P; Bergvinsdottir, K. B, Solving the dial-a-ride problem using genetic algorithms, *Journal of the Operational Research Society*, v. 58, n. 10, p. 1321-1331, 2009.

Mauri G. R.; Lorena L. A. N., Uma nova abordagem para o problema dial-a-ride, *Produção*, v. 19, n. 1, 41-54, 2009.

Savelsbergh, M. W. P., The vehicle routing problem with time windows: minimizing route duration, *ORSA Journal on Computing*, v. 4, n. 2, p. 146-154, 1992.

Soldano, A.; Valandro, F., Online and offline algorithms for the Dial-a-Ride problem: design and implementation of operational research heuristic algorithms for transportation problems, Politecnico di Milano, 2004.