SUPRESSÃO DE CONFLITOS NO PROBLEMA DE PLANEJAMENTO E ROTEAMENTO DE TRENS UTILIZANDO A METAHEURÍSTICA VNS

Higor D'Talles Costa, Marcia Helena Moreira Paiva, Helder Roberto de Oliveira Rocha
Universidade Federal do Espírito Santo
Departamento de Engenharia Elétrica, Vitória, Espírito Santo, Brasil
higorcosta@gmail.com, marcia.paiva@ufes.br, helder.rocha@ufes.br

RESUMO

O transporte ferroviário tem sido muito demandado para o transporte eficiente de cargas e passageiros. Para torná-lo mais competitivo, destaca-se o problema de planejamento e roteamento de trens. Este estudo apresenta um modelo para resolver esse problema baseado em ferrovias de fluxo bidirecional. O cerne do problema está em reduzir os tempos de viagem detectando e eliminados os conflitos entre as rotas. Por se tratar de um problema *NP-Hard* foi proposta uma solução heurística. Foi implementada a metaheurística *Variable Neighborhood Search*. O algoritmo foi testado em quatro topologias planares aleatórias com n vértices e m arestas, sendo n/m dados por: 49/57, 94/112, 184/222 e 274/332. O número de trens foi de 20% do número de vértices de cada topologia. Nos experimentos foram gerados números elevados de conflitos para permitir testes em condições extremas. Os resultados foram ótimos em relação aos objetivos propostos e à complexidade das redes.

PALAVRAS CHAVE. Planejamento e roteamento de trens, Operação ferroviária, Variable neighborhood search.

L&T – Logística e Transportes, MH – Metaheuristicas, TAG – Teoria e Algoritmos em Grafos

ABSTRACT

The rail transport has been much in demand for the efficient transportation of cargo and passengers. To make it more competitive, the problem of train scheduling and routing is highlighted. This study presents a model to solve this problem based on two-way flow railways. The core of the problem lies in reducing travel times by detecting and eliminating conflicts between routes. As it is an NP-hard problem, a heuristic solution was proposed. The Variable Neighborhood Search metaheuristic was implemented. The algorithm was tested in four random planar topologies with n vertices and m edges, with n/m given by: 49/57, 94/112, 184/222 and 274/332. The number of trains was 20% of the number of vertices of each topology. In the experiments, high numbers of conflicts were generated to allow tests in extreme conditions. The results were excellent regarding the proposed objectives and the complexity of the networks.

KEYWORDS. Train scheduling and routing, Railway operations, Variable neighborhood search.

L&T – Logistics and Transport, MH – Metaheuristics, TAG – Theory and Graph Algorithms

https://proceedings.science/p/85446?lang=pt-br

1. Introdução

O planejamento do transporte ferroviário envolve várias etapas e, em geral, essas etapas são comuns a outros modais de transporte. Elas podem ser descritas por: análise da demanda, planejamento das rotas, planejamento da programação dos trens, planejamento do material rodante e da equipe de operadores [Li et al., 2012]. Dentre as etapas citadas, destacam-se os problemas de planejamento e de roteamento de trens.

O planejamento de trens é fundamental para evitar que eles realizem paradas indesejadas. Essas paradas são especialmente relevantes já que um dos maiores custos da ferrovia é referente ao consumo de combustível. Para se ter ideia desse consumo, considere uma composição com 3 locomotivas de 4.400 HP cada uma e 252 vagões carregados de minério. Essa composição tem uma carga total aproximada de 25.180 toneladas. Para retirar esse trem do repouso e levá-lo à velocidade que ele estaria se não tivesse parado seriam necessários, em média, 150 litros de diesel.

No melhor caso não deveria haver qualquer parada na rota de um trem porque assim seria praticado o menor tempo possível de viagem. Contudo, como não é viável aguardar que um trem chegue ao seu destino para então enviar o próximo, faz-se necessário parar alguns deles em momentos específicos para que outros, que sejam prioritários, possam continuar sua viagem. A lógica de cruzamento de trens pode tornar bastante complexa a etapa de planejamento da programação.

Dado esse contexto, a questão central do problema está em detectar e suprimir os conflitos de rotas entre os trens que circulam na ferrovia. A solução proposta foi baseada em métodos heurísticos já que trata-se de um problema *NP-Hard* [Caprara et al., 2002]. Foi implementada a metaheurística VNS *(Variable Neighborhood Search)* conforme descrito no trabalho publicado por [Hansen and Mladenović, 1999].

O restante deste trabalho é organizado como segue. A Seção 2 apresenta a revisão bibliográfica. A Seção 3 descreve como o problema foi modelado e apresenta um exemplo prático. Na Seção 4 é apresentada a heurística de otimização que foi utilizada, incluindo a busca local, as estruturas de vizinhança e a heurística de compactação dos tempos de viagem. Em seguida, na Seção 5, são mostrados todos os detalhes dos experimentos realizados bem como os resultados obtidos. E, por fim, na Seção 6 é feita a análise dos resultados e a conclusão do estudo como um todo.

2. Trabalhos Relacionados

O planejamento de trens envolve as etapas de planejamento de rotas e de planejamento da programação. Essas etapas são importantes para se tentar melhorar o nível de serviço e, consequentemente, reduzir os tempos de atraso de todos os trens em circulação na malha ferroviária [Li et al., 2012]. Segundo [Cai et al., 1998], os problemas de roteamento de trens podem ser modelados com ferramentas de otimização matemática bastante sofisticadas levando a resultados muito expressivos. Entretanto, essas ferramentas devem ser restritas à solução de problemas bem reduzidos e com poucos trens já que trata-se de problemas de difícil solução.

Frequentemente o problema de planejamento e roteamento de trens é modelado como um problema conhecido na literatura como *Job Shop Scheduling Problem (JSP)*. Muitos autores o implementam com base na modelagem por grafos disjuntivos [Mascis and Pacciarelli, 2002]. Esse tipo de grafo é bastante popular para resolver problemas dessa natureza [Blażewicz et al., 2000]. Em [Mascis and Pacciarelli, 2002] foi proposta uma generalização desses grafos pela verificação de aplicabilidade a alguns dos casos mais bem sucedidos da literatura referente ao problema de *JSP* aplicado a ferrovias. Alguns autores relatam, inclusive, excelentes resultados oriundos de casos reais de planejamento e roteamento de trens utilizando essa metodologia [D'ariano et al., 2007, Samà et al., 2017]. Além disso, há ferramentas em constante desenvolvimento que são utilizadas por alguns projetos europeus onde é necessário um sistema robusto de suporte às operações [D'Ariano et al., 2008].

Discutida a modelagem do problema, se faz necessário discutir o método de geração das menores rotas. [Yen, 1971] propôs um algoritmo para o cálculo dos *K Shortest Loopless Paths* de uma rede. Esse é um dos mais eficientes para este cálculo. Ele apresenta um alto desempenho, já

https://proceedings.science/p/85446?lang=pt-br

que, a complexidade computacional dos cálculos cresce de modo linear. Esse algoritmo se baseia no que foi proposto por [Dijkstra, 1959] onde, a partir de um determinado vértice, são calculados os caminhos mais curtos para qualquer outro vértice da rede.

Uma vez discutido o método de roteamento, é hora de pensar em estratégias para a detecção e eliminação dos conflitos entre as rotas. Essa é uma das partes mais importantes e difíceis de serem solucionadas em tempo real. [Hansen, 2010] propõe que haja um tempo mínimo entre os trens. Isso resulta em um bloco de tempo dedicado a cada um deles e em cada um dos trechos por onde eles passam. Desse modo, os conflitos podem ser detectados por qualquer evento de sobreposição desses blocos. O uso dos blocos de tempo também é citado por [Samà et al., 2017] como técnica de detecção de conflitos. Em outra pesquisa, [Nõu, 1997] propõe a utilização da metaheurística *Busca Tabu* para encontrar as melhores permutações entre os trens para que, deste modo, os conflitos possam ser eliminados. Outro método utilizado para resolver conflitos foi proposto por [Ingolotti et al., 2005]. Ele define uma série de restrições e uma em particular contribui para o tratamento dos conflitos. Quando essa restrição é violada para dois trens que participam de um conflito, é atribuído um atraso a um deles até que o conflito existente seja eliminado.

Discutida a questão dos conflitos, é preciso avaliar métodos para melhorar a solução corrente. Métodos de otimização são largamente utilizados para se obter soluções ótimas, ou muito próximas disso, para problemas de difícil solução. Em geral são utilizadas técnicas de programação matemática tais como a *Programação Linear Inteira Mista*. [Szpigel, 1973] foi o primeiro a propor o uso do algoritmo *branch-and-bound* para o planejamento de trens em linha singela. Por se tratar de problemas do tipo *NP-Hard*, frequentemente são utilizados métodos heurísticos para se conseguir bons resultados em um tempo computacional aceitável [Samà et al., 2017]. Como exemplo, há uma diversidade de estudos que utilizam a metaheurística *Colônia de Formigas* em problemas complexos de ferrovia [Tsuji et al., 2012, Huang, 2006]. Outros utilizam a metaheurística VNS sozinha ou acompanhada de outra, como a *Busca Tabu*, para lidar com o processo de busca local nas vizinhanças utilizadas pelo VNS [Samà et al., 2017]. Algumas outras abordagens utilizam esquemas de VNS adaptativo devido aos diferentes esquemas que podem ser utilizados para a obtenção de novos cenários [Hassannayebi and Zegordi, 2017]. Esses são apenas alguns dos exemplos encontrados na literatura em meio a tantos outros.

No processo de busca e eliminação de conflitos nem sempre serão encontrados os menores tempos de viagem. Por vezes as paradas dos trens poderão ser maiores do que o necessário para que ocorram os cruzamentos necessários entre eles. Isso ocorre porque o objetivo dessa etapa é, literalmente, eliminar os conflitos. Para os casos em que são utilizados blocos de tempo, conforme descrito anteriormente, [Landex et al., 2006] propõe um método de compactação desses tempos. O efeito prático é que os trens podem circular mais próximos uns dos outros.

3. Modelagem da Ferrovia

Ao contrário de muitas linhas de pesquisa, neste trabalho optou-se por não utilizar um modelo baseado no *JSP*. O desenvolvimento se deu segundo a teoria dos grafos por ela apresentar importantes características e métricas. As seções de bloqueio são representadas pelos vértices e as conexões entre elas são representadas pelas arestas. A seção de bloqueio em si é dada por um trecho onde só pode haver um trem por vez. O controle lógico de ocupação dessas seções é realizado pelo sistema de sinalização, que envia sinais pelos trilhos. Esses sinais são captados pelas locomotivas e veículos de manutenção e trazem, essencialmente, informações de velocidade e da situação de ocupação dos próximos trechos da rota. Dependendo dessas informações, o maquinista pode ser obrigado a parar para que uma composição prioritária possa passar. Embora não seja abordado neste trabalho, na prática existem algumas situações em que um trem é prioritário sobre o outro. Como exemplos reais de prioridade, pode-se citar: trem de socorro é prioritário sobre todos os outros, trem de passageiros é prioritário sobre trem de carga, entre outros casos a depender da legislação federal em vigor e das regras de cada ferrovia. Na Seção 3.2 será mostrado como é tratada a prioridade de passagem mediante um conflito.

https://proceedings.science/p/85446?lang=pt-br

A título de ilustração, na Figura 1 é mostrada uma topologia composta por 49 vértices e 57 arestas gerada a partir de um grafo planar com grau médio igual a *2,3*. O grau médio de um grafo é dado por *2m/n*, onde m é o número de arestas e n o número de vértices. Nesta rede foram programadas as rotas de 10 trens. Os vértices maiores são aqueles que apresentam maior medida de centralidade *betweenness*. Essa métrica é proporcional à probabilidade de um nó ser intermediário na comunicação entre outros nós da rede. Trata-se de uma métrica importante para o controle da comunicação entre os nós de uma rede [Freeman, 1977].

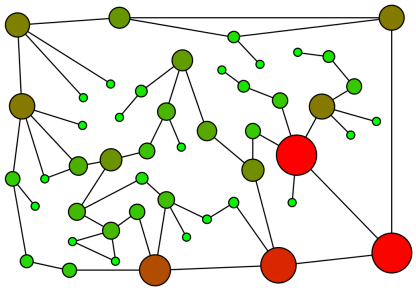


Figura 1: Modelagem da ferrovia com grafo planar de 49 vértices, 57 arestas e 10 trens. Quanto maior a *betweenness*, maior é o nó e mais vermelha é a sua cor na escala do vermelho para o verde.

3.1. Planejamento e Programação de Trens

O planejamento dos trens compreende tanto a definição do número de trens que devem circular em cada rede quanto a programação de cada um deles ao longo do tempo. Como resultado final é obtido o perfil da rota de cada um dos trens, com tempos de movimento e de parada, do início ao final de sua rota.

Em termos de planejamento, se o número de trens for muito grande em relação à rede, o número de conflitos também será, exigindo um tempo de processamento bastante elevado. Se os trens são pouco espaçados no tempo em relação aos horários de partida, também são gerados muitos conflitos. Para evitar sobrecargas é necessário que na fase de planejamento sejam consideradas as características da ferrovia, a carga que deve ser transportada e os prazos acordados.

Em relação à programação, neste trabalho ela foi realizada de modo aleatório segundo os vértices de cada uma das redes. A rota mais curta para cada um dos trens foi obtida pelo algoritmo utilizado para o cálculo dos *K Shortest Loopless Paths* proposto por [Yen, 1971]. Já que neste trabalho considera-se que há trens prioritários na utilização dos trechos da ferrovia, optou-se por utilizar somente a rota mais curta para cada trem.

Para cada trecho foi estabelecido um tempo médio de cruzamento, dado pela razão entre o comprimento dele, em *Km*, e a velocidade média permitida, em *Km/h*. Desse modo, a busca pela solução do problema se resume a encontrar os tempos de parada que devem ser atribuídos a cada um dos trechos que compõe a rota de cada um dos trens. Sendo assim, entre outros parâmetros, os trens terão as referências de tempo de chegada em cada trecho, parada e partida para o próximo.

3.2. Verificação e Supressão de Conflitos

A informação de conflito de rota é levantada pela comparação de todos os trechos de cada trem. Na primeira fase são observados os conflitos no espaço e, em seguida, é verificado se esses conflitos ocorrem em intervalos de tempo que se cruzam. Havendo qualquer conflito, é então verificado quais trens estão envolvidos e qual deles tem a prioridade de passagem. Neste trabalho a prioridade se dá pela ordem de chegada no trecho em que o conflito é percebido. O trem menos prioritário é parado no trecho anterior ao conflito até que o trem prioritário termine sua trajetória. Os conflitos são contabilizados por trecho e por pares de trens, de modo similar ao proposto por [Ingolotti et al., 2005].

https://proceedings.science/p/85446?lang=pt-br

A Figura 2 exemplifica um caso de prioridade de passagem na ocorrência de um conflito. A Figura 2a mostra o momento em que os trens T_1 e T_2 acessam o trecho x_1-x_2. É possível notar que ocorre um conflito no trecho mencionado e que o trem T_1 chega primeiro a x_1. Então, sabendo que T_1 tem prioridade de passagem, T_2 deve ficar parado até que T_1 conclua o seu trajeto. Na Figura 2b é mostrada a situação descrita.

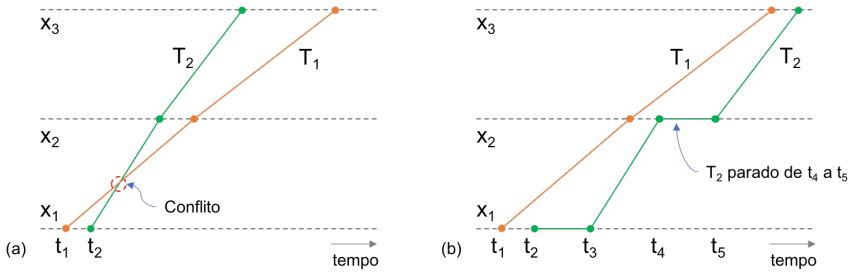


Figura 2: Detecção e eliminação de conflito mediante a prioridade de passagem de trens.

3.3. Exemplo Prático

No exemplo a seguir a Figura 2 é utilizada como base. Será feita uma análise complementar à anterior, onde o procedimento completo de detecção e eliminação de conflito é apresentado.

A figura supracitada mostra o momento em que ocorre um conflito no trecho x_1-x_2. Nesse caso, o VNS executa uma perturbação na solução corrente gerando uma solução em sua vizinhança. Uma vez identificado qualquer conflito na nova solução é executada a etapa de busca local. Nessa busca o tempo de conflito é utilizado parcial ou totalmente como referência para tentar eliminá-lo. Ele é adicionado como tempo de parada no trem menos prioritário para que este sofra um atraso para entrar no trecho em conflito. É então encontrado um tempo tal que T_2 deve ficar parado em x_1 até que T_1 termine de passar pelo trecho em questão. Essa parada ocorre no intervalo de tempo t_2-t_3. Novamente é verificado se há qualquer conflito decorrente do movimento realizado. Analisando a Figura 2b é possível notar que ocorreria outro conflito no trecho x_2-x_3. Sendo assim, T_2 deve sofrer uma nova parada no intervalo de tempo t_4-t_5. Se todos os conflitos forem eliminados o VNS move a solução encontrada para a solução corrente. O VNS repete as etapas de perturbação e busca local até que não haja mais conflitos e que o menor tempo total de parada de todos os trens seja encontrado.

3.4. Função Objetivo e Premissas

A função objetivo que se deseja minimizar é o somatório do tempo de viagem de todos os trens. Esse tempo é composto pelo somatório dos tempos de movimento e dos tempos de parada, do início ao final de cada rota. Sendo assim, quanto menor o tempo de parada melhor é a solução. Então, já que o melhor ajuste para uma rota se daria pela eliminação dos tempos de parada de um trem, foi assumida a premissa de que uma solução seria melhor do que outra se ela tivesse o menor tempo de viagem. Somente se isso ocorrer o VNS aceita a movimentação da solução corrente para a nova solução. E, como no procedimento de busca local podem ser atribuídos tempos de parada às rotas para que os conflitos sejam eliminados, foi considerada uma tolerância de 35% para que soluções piores pudessem ser aceitas. Desse modo evita-se que o VNS fique preso em ótimos locais. Após a etapa do VNS os tempos indesejados de parada são localizados e removidos.

Outra premissa assumida neste trabalho é em relação ao horário de partida. Nenhum trem pode partir do ponto inicial de sua rota antes do horário estabelecido para o início da sua viagem. Para o algoritmo desenvolvido isso seria como atribuir um tempo negativo ao horário de partida.

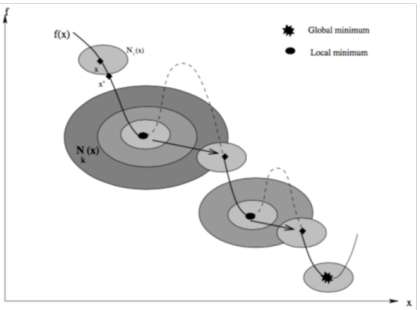
4. Heurística de Otimização

Muitas metaheurísticas são baseadas em heurísticas de busca local para encontrar soluções ótimas. O VNS, por outro lado, não segue esse caminho. Ele explora vizinhanças gradativamente

mais distantes da solução atual. Essa técnica permite utilizar somente uma rotina de busca local e ainda assim convergir para soluções ótimas. Isso porque características favoráveis da solução atual são mantidas e usadas para conseguir soluções promissoras em novas estruturas de vizinhança [Hansen and Mladenović, 2001]. Trata-se de um esquema bastante robusto que se adapta muito bem a outras metaheurísticas na formação de modelos híbridos, seja utilizando o VNS como heurística principal ou mesmo na etapa de busca local. Dentre suas principais características, destacam-se: mudança sistemática de vizinhança, busca local possivelmente aleatória e exploração de vizinhanças gradativamente maiores.

O VNS explora sistematicamente três fatos. O primeiro deles define que um mínimo local em uma estrutura de vizinhança não é necessariamente um mínimo local para outra. O segundo define que um mínimo global é um mínimo local para todas as estruturas de vizinhança possíveis. E o terceiro define que, para muitos problemas, ótimos locais para uma ou mais estruturas de vizinhança são relativamente próximos. O último fato é empírico e pressupõe que um ótimo local sempre fornece muitas informações acerca do ótimo global.

A Figura 3a exemplifica como são utilizadas as estruturas de vizinhança no VNS Básico, esquema utilizado neste estudo. São combinados métodos estocásticos e determinísticos para a mudança de vizinhança. É bastante frequente a utilização de vizinhanças aninhadas quando se utiliza esse esquema de VNS. E, na Figura 3b, é mostrado o algoritmo do VNS Básico, que por definição utiliza como busca local um algoritmo diferente do VND (*Variable Neighborhood Descent*). O VND foi proposto como heurística de busca local e seu uso configura o esquema do VNS Geral conforme descrito por [Hansen and Mladenovic, 2007].



1 Definir um conjunto de k_{max} vizinhanças.
2 Determinar uma solução inicial x;
3 Enquanto não é satisfeito um critério de parada,
3.1 $k \leftarrow 1$;
3.2 Enquanto ($k \leq k_{max}$),
3.2.1 Gerar, aleatoriamente, $x' \in N_k(x)$;
3.2.2 Usando um método de busca local, encontrar x'', mínimo local próximo de x'.
3.2.3 Se ($f(x'') < f(x)$),
3.2.3.1 Então $x \leftarrow x''$; $k \leftarrow 1$;
3.2.3.2 Senão $k \leftarrow k + 1$;
3.2.4 Fim-se;
3.3 Fim-enquanto;
4 Fim-enquanto;

(b)

Figura 3: (a) Busca de ótimos locais em diferentes estruturas de vizinhança, (b) algoritmo VNS Básico (*fonte: [Hansen and Mladenovic, 2007]*).

Do ponto de vista do número de execuções do VNS, foi estabelecido um número máximo que depende do número de vértices e de trens que circulam na malha. Entretanto, o algoritmo fica limitado também por um número mínimo de execuções. Além disso, um dos critérios de parada está relacionado ao número de vezes que o algoritmo roda sem obter qualquer melhoria em relação à solução corrente.

4.1. Busca Local

Para que os conflitos possam ser resolvidos é executada a etapa de busca local, que tem como principal objetivo avaliar os conflitos existentes e buscar os tempos de parada que devem ser atribuídos a cada trecho de cada um dos trens que participam de um dado conflito. A busca local é finalizada quando todos os conflitos são eliminados ou quando pelo menos um dos critérios de parada é atingido.

4.2. Estruturas de Vizinhança

Tendo atingido um ótimo local, é necessário provocar uma perturbação na solução corrente para que outra possa ser obtida e desse modo o VNS consiga escapar de um possível ótimo local. Com esse objetivo, foram estabelecidas seis estratégias de vizinhança, que podem ser vistas

https://proceedings.science/p/85446?lang=pt-br

na Tabela 1. A troca entre elas ocorre da primeira para a última, sempre executando buscas locais para tentar encontrar as melhores soluções dentro de cada uma delas.

Tabela 1: Vizinhanças utilizadas pelo VNS.

Vizinhança	Descrição da vizinhança
Viz. 1	Caso o trem fique parado em algum trecho, permuta dois dos seus tempos de parada mesmo que em um dos trechos o tempo seja nulo. Essa ação cria oportunidades de parada por onde outros trens possam passar.
Viz. 2	Onde tiver tempo de parada para um dado trem, escolhe um trecho e reduz o tempo em 10%. Em seguida, esse tempo é somado ao tempo de parada do trecho anterior. O efeito dessa perturbação faz com que o tempo de parada seja reduzido.
Viz. 3	Onde o trem tiver parada superior a 5 minutos, escolhe um trecho e reduz esse tempo em 10%. O efeito prático é a antecipação da partida do trem em direção ao próximo trecho.
Viz. 4	Para um trem e um trecho escolhidos ao acaso, insere um tempo padrão de parada. Esses tempos criam oportunidades para que outros trem possam efetuar cruzamentos.
Viz. 5	Para o trem que estiver envolvido no maior número de conflitos, insere um tempo padrão de parada no trecho inicial de sua rota. Isso faz com que o trem atrase sua partida.
Viz. 6	Para um trem escolhido ao acaso, insere um tempo padrão de parada no trecho inicial de sua rota. O objetivo é atrasar a partida do trem.

4.3. Compactação dos Tempos de Viagem

O principal objetivo do VNS é desembaralhar as rotas, encaixando os cruzamentos e as paradas para que os trens possam fluir da melhor forma possível sem qualquer possibilidade de colisão. Entretanto, ao final desse processo pode ocorrer que não tenham sido encontrados os menores tempos de parada para cada um dos trens.

Em vista disso, foi proposta uma heurística para identificar e eliminar as paradas indesejadas. Trata-se de um algoritmo de compactação de tempos inspirado em [Landex et al., 2006]. A Figura 4 mostra duas situações onde esses tempos podem ocorrer. Na Figura 4a, instância *yEd_94v112a*, são mostrados exemplos de tempos que podem ser removidos já que trata-se de paradas no trecho final e, portanto, sem relevância. A Figura 4b, instância *yEd_184v222a*, mostra que todos os trens ficam parados em um dado intervalo de tempo e por isso é possível remover esses tempos na rota de cada um deles. As marcações foram feitas em linhas tracejadas e são indicadas por setas. Os tempos de parada são indicados pelas linhas paralelas ao eixo do tempo. As características das instâncias citadas podem ser vistas na Tabela 3, Seção 5.1.

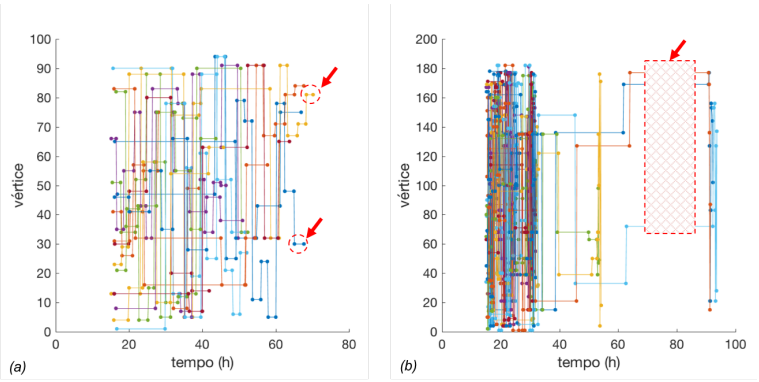


Figura 4: Paradas indesejadas: (a) tempo de parada em regiões onde não há outros trens (*yEd_94v112a*), (b) tempo de parada de todos os trens ao mesmo tempo (*yEd_184v222a*).

5. Experimentos

Diante do exposto nas seções anteriores, na Figura 5 é apresentado um fluxograma básico que resume todo o processo descrito neste trabalho.

Com base nesse processo foram realizados experimentos com 4 grafos distintos. Eles são planares, têm grau médio aproximado de $2,4$ e são formados por n vértices e m arestas, sendo n/m dados por: 49/57, 94/112, 184/222 e 274/332. Eles foram gerados aleatoriamente pela aplicação

https://proceedings.science/p/85446?lang=pt-br

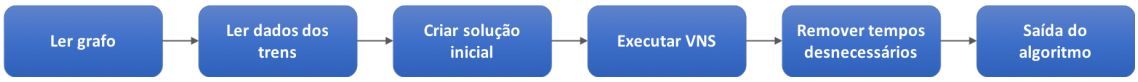


Figura 5: Fluxograma do processo de planejamento e roteamento de trens.

yEd Graph Editor [Jünger and Mutzel, 2012]. A cada vértice foi atribuído um peso que representa o tempo, em *horas*, necessário para que um trem o percorra por completo. Essas topologias foram testadas com um número de trens igual a 20% do número de vértices. Para cada um desses grafos o algoritmo foi executado 4 vezes.

Em cada uma das rotas e, para cada um dos seus trechos, foi estabelecida uma velocidade média de tráfego. Essa velocidade varia entre 30 e 65 Km/h. A velocidade mais baixa, em geral, está associada ao tráfego em regiões com restrições definidas pela operação ou manutenção. A mais alta, por sua vez, representa o limite seguro de tráfego para o perfil da ferrovia. Outro dado importante é o comprimento de cada trecho. Nas redes testadas ele varia entre 5 e 10 Km. Nos modelos criados tanto a velocidade quanto o comprimento dos trechos foram obtidos segundo uma distribuição aleatória.

A Tabela 2 resume os parâmetros calculados para cada um dos grafos. A menos que outra unidade seja indicada, todos os tempos são dados em horas.

Tabela 2: Definição dos parâmetros do problema.

Parâmetro	Descrição do parâmetro
Conf	Número médio de conflitos das 4 execuções
Temp	Tempo médio que o algoritmo leva para encontrar a solução final
TVI	Somatório do tempo de viagem da solução inicial considerando todos os trens
TVF	Somatório do tempo de viagem da solução final considerando todos os trens
TPI	Somatório do tempo de parada da solução inicial considerando todos os trens
TPF	Somatório do tempo de parada da solução final considerando todos os trens
Conf / Trechos	Relação entre o número de conflitos e o somatório dos trechos de todas as rotas
Conf / Trem	Número médio de conflitos por trem
Temp / Vértice	Relação entre o tempo de solução e o número de vértices
Temp / Trem	Relação entre o tempo de solução e o número de trens
Temp / Conf	Relação entre o tempo de solução e o número de conflitos
TVF / TVI	Relação entre o tempo de viagem final e inicial
TPF / TPI	Relação entre o tempo de parada final e inicial
DesvPadConf	Desvio padrão do número de conflitos das 4 execuções
DesvPadTemp	Desvio padrão de *Temp*
DesvPadTVI	Desvio padrão de *TVI*
DesvPadTVF	Desvio padrão de *TVF*
DesvPadTPI	Desvio padrão de *TPI*
DesvPadTPF	Desvio padrão de *TPF*

5.1. Resultados

Os resultados obtidos para cada uma das topologias são mostrados na Figura 6. A Figura 6a mostra o tempo de processamento até que o algoritmo convergisse para a solução final. É possível notar que até a instância com 94 vértices o tempo de convergência é bastante reduzido não chegando a 2 minutos. Daí para frente, entretanto, o tempo começa a aumentar de modo considerável, embora ele cresça de forma bastante linear. Em relação aos tempos de viagem, Figura 6b, pode-se notar também que a partir da instância com 184 vértices, embora o tempo final de viagem seja um pouco superior ao inicial, eles crescem quase que paralelamente. No que tange ao tempo de parada ao longo da rota, Figura 6c, o tempo final ficou bem superior ao inicial. Isso já era esperado já que os tempos de parada são essenciais para que ocorram os cruzamentos.

A Tabela 3, a seguir, mostra mais detalhes sobre os resultados obtidos. Nela são mostrados quatro grupos de dados, a saber: *Entradas*, *Tempos*, *Desvios Padrão* e *Indicadores*. As instâncias utilizadas estão nas colunas e são representadas por nomes iniciados por *yEd*.

O grupo de *Entradas* traz relações bastante importantes envolvendo o número de conflitos. Uma medida que chama atenção é a variável *DesvPadConf*. Ela decresce à medida em que aumenta

https://proceedings.science/p/85446?lang=pt-br

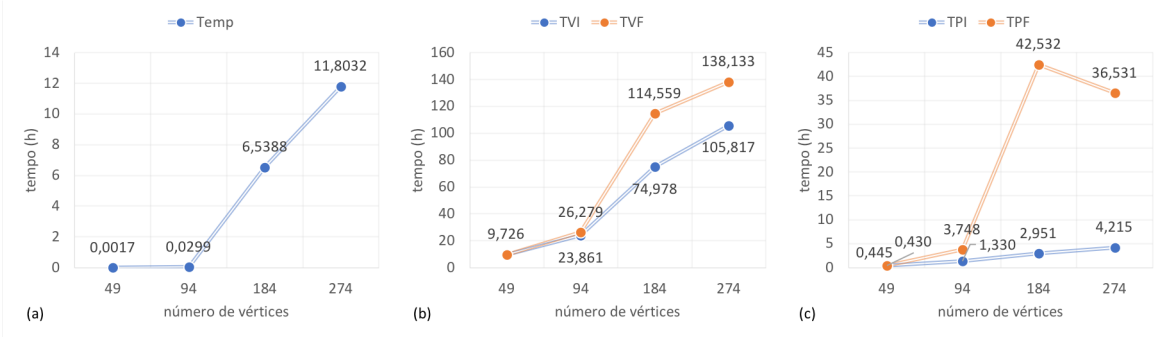


Figura 6: Evolução dos tempos de (a) processamento, (b) viagem e (c) parada.

o número de trechos da rede. Em outras palavras, à medida em que aumenta a rede, aumenta também a homogeneidade dos conflitos. Amostras com desvio padrão pequeno indicam que os dados estão mais condensados nas proximidades da média e, com desvio padrão grande, indicam que eles estão mais distribuídos em torno dela.

Outra análise que pode ser feita é em relação ao aumento do número de trechos ou do tempo para que os conflitos sejam eliminados. Para cada uma das instâncias foi levantada a quantidade total de trechos e o número total de trechos em conflito na solução inicial. Se a instância *yEd_184v222a* for analisada, por exemplo, pode-se concluir, em linhas gerais, que para eliminar os conflitos seria necessário aumentar em quase 25% o número de trechos para que esses conflitos pudessem ser removidos. Alternativamente, conforme proposto pelo algoritmo utilizado neste trabalho, com um aumento médio de 50% do tempo de viagem dos trens, foi possível eliminar todos os conflitos na programação realizada. Do ponto de vista econômico, na maior parte dos casos, esse aumento de tempo se justifica por ser mais viável do que construir e manter novos ramais ou linhas férreas. Isso sem contar os esforços para se conseguir as licenças ambientais e sociais necessárias para que essas linhas possam ser construídas. Por outro lado, por exemplo, é possível distribuir as cargas ao longo do tempo, negociando com os clientes ou aumentando os estoques.

Quanto ao grupo de *Desvios Padrão*, nele são apresentados os desvios padrão de todos os tempos do processo. Se for observado o *DesvPadTVF*, pode-se verificar que ele aumenta em relação à média à medida em que cresce o número de trechos da rede e, por outro lado, *DesvPadConf* diminui. Desse modo, à medida em que a rede cresce, *DesvPadTVF* se torna menos homogêneo e *DesvPadConf* se torna mais homogêneo. Em termos práticos, pode-se concluir que à medida em que a rede aumenta, aumenta também a dispersão do tempo de viagem ao passo em que a dispersão do número de conflitos é reduzida.

A Figura 7 mostra os gráficos referentes aos principais indicadores do grupo de *Indicadores*. Para os três apresentados é notável a tendência linear apresentada por *Temp/Conf*, Figura 7c, a partir da topologia com 94 vértices.

Nas Figuras 8 e 9 são mostrados os gráficos obtidos para a menor e a maior rede. Na primeira ainda é possível acompanhar as linhas referentes às rotas percorridas pelos trens. As Figuras 8a e 9a, 8b e 9b e 8c e 9c mostram, respectivamente, a solução inicial, a solução obtida pelo VNS e a solução final obtida após a compactação dos tempos. Observando os tempos de viagem nas Figuras 9b e 9c fica clara a importância da heurística de compactação de tempos descrita na Seção 4.3.

Das 16 execuções do algoritmo, em apenas duas delas os conflitos não foram completamente eliminados. Contudo, nos dois casos restou apenas um único conflito. Analisando esses casos foi observado que eles ocorreram nos trechos iniciais de uma das rotas que participam dos conflitos. Isso pode ocorrer quando, para eliminar um conflito, o algoritmo altera o horário de partida para um momento anterior àquele definido para a partida do trem. Entretanto, neste trabalho foi

https://proceedings.science/p/85446?lang=pt-br

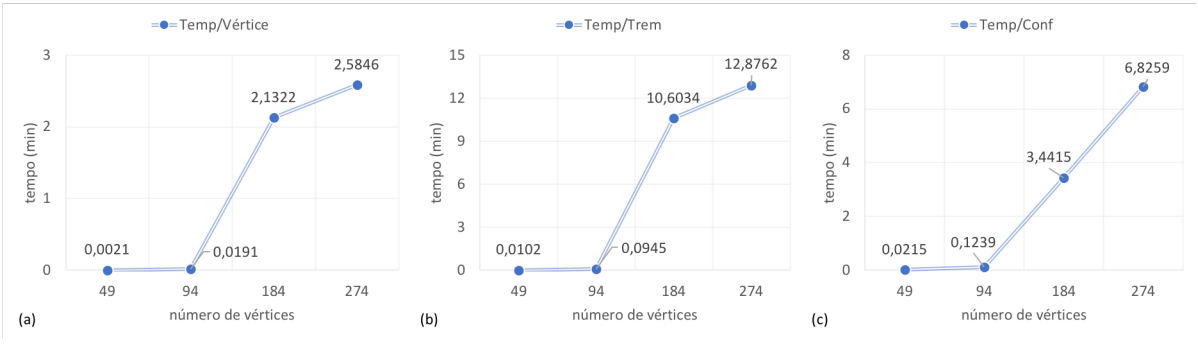


Figura 7: Indicadores (a) *Temp/Vértice*, (b) *Temp/Trem* e (c) *Temp/Conf*.

Tabela 3: Parâmetros calculados para cada topologia.

Grupo	Parâmetro	yEd_49v57a	yEd_94v112a	yEd_184v222a	yEd_274v332a
Entradas	Vértices	49	94	184	274
	Arestas	57	112	222	332
	Trens	10	19	37	55
	Total Trechos	66	138	472	650
	Conf	4,7500	14,5000	114,0000	103,7500
	Conf/Trem	0,4750	0,7632	3,0811	1,8864
	Conf/Trechos	0,0720	0,1051	0,2415	0,1596
Tempos	Temp	0,0017	0,0299	6,5388	11,8032
	TVI	9,7416	23,8614	74,9776	105,8173
	TVF	9,7263	26,2794	114,5588	138,1333
	TPI	0,4453	1,3304	2,9510	4,2153
	TPF	0,4299	3,7484	42,5323	36,5309
Desvios Padrão	DesvPadConf	2,2174	1,7321	7,8740	3,3040
	DesvPadTemp	0,0003	0,0080	1,0329	8,7482
	DesvPadTVI	0,1402	0,6016	0,4499	0,4872
	DesvPadTVF	0,3484	1,3681	6,0326	14,9225
	DesvPadTPI	0,1402	0,6016	0,4499	0,4873
	DesvPadTPF	0,3484	1,3681	6,0326	14,9226
Indicadores	Temp/Vértice (minutos)	0,0021	0,0191	2,1322	2,5846
	Temp/Trem (minutos)	0,0102	0,0945	10,6034	12,8762
	Temp/Conf (minutos)	0,0215	0,1239	3,4415	6,8259
	TVF/TVI	0,9984	1,1013	1,5279	1,3054
	TPF/TPI	0,9655	2,8176	14,4127	8,6663

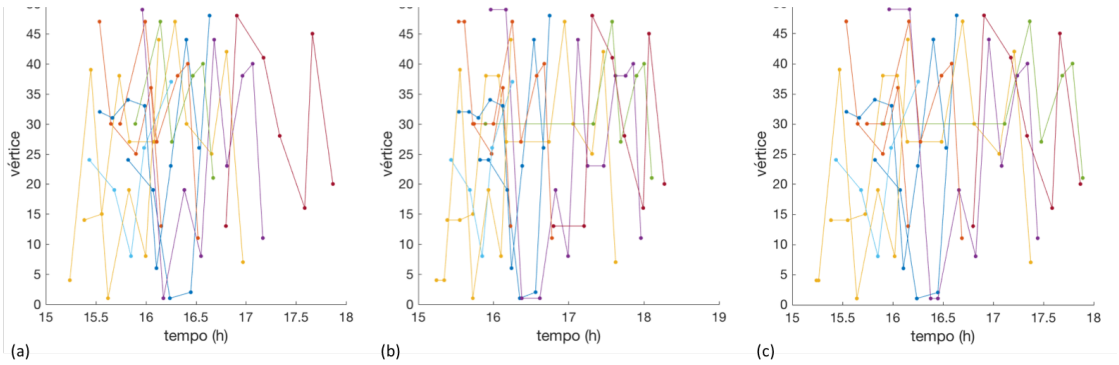


Figura 8: Supressão de conflitos para (a) solução inicial, (b) solução pelo VNS e (c) compressão de tempos para a topologia *yEd_49v57a*.

estabelecido que o horário de partida não poderia ser inferior ao adotado na programação inicial.

6. Conclusões e Trabalhos Futuros

O objetivo deste trabalho foi investigar o uso da metaheurística VNS aplicada ao problema de planejamento e roteamento de trens. Esse problema é conhecido por ser do tipo *NP-Hard* e,

https://proceedings.science/p/85446?lang=pt-br

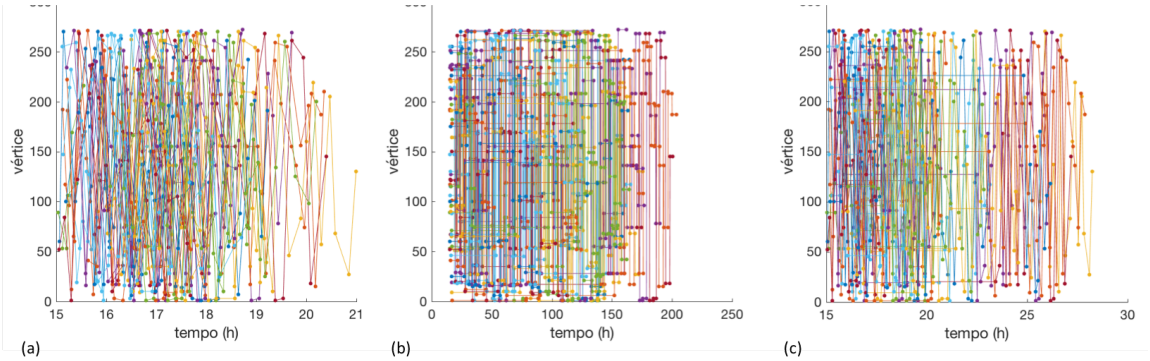


Figura 9: Supressão de conflitos para (a) solução inicial, (b) solução pelo VNS e (c) compressão de tempos para a topologia *yEd_274v332a*.

portanto, de difícil solução para abordagens mais tradicionais como a programação linear. Uma das partes que exige maior esforço computacional é aquela responsável por identificar e suprimir conflitos de rotas. Dado esse contexto, a ferrovia foi modelada por grafos onde, para cada trem, foi obtida a menor rota possível pelo algoritmo de *Yen*, segundo o comprimento de cada trecho.

Os resultados obtidos foram bastante satisfatórios diante do desafio proposto. Todas as instâncias iniciaram com muitos conflitos entre os trens. Isso porque a proposta principal foi testar a robustez do algoritmo na busca e eliminação dos conflitos existentes. Sabe-se que em ferrovias reais os trens são liberados para partida com um distanciamento mínimo segundo os padrões estabelecidos para a segurança operacional. Nesse caso, o algoritmo teria um desempenho melhor na busca da melhor solução.

O esquema utilizado para solucionar o problema pode ser beneficiado com a utilização de técnicas mais avançadas. Isso inclui etapas de otimização por programação linear e metaheurísticas híbridas que deem ao modelo uma garantia maior de convergência com tempos de processamento mais reduzidos. Além disso, é válido investigar propriedades de grafos que possam contribuir para o processo de otimização. Nessa linha de pesquisa a métrica *betweenness* poderia servir como entrada para o algoritmo de *Yen* de modo que, em alguns momentos, os vértices com maior *betweenness* pudessem ser evitados para guardá-los de sobrecargas, mesmo que uma rota mais longa tivesse que ser atribuída a um determinado trem. E, para melhorar ainda mais o algoritmo deste trabalho, pretende-se avaliar a perda de um ou mais trechos, por falha ou por qualquer outro evento.

Referências

Blażewicz, J., Pesch, E., and Sterna, M. (2000). The disjunctive graph machine representation of the job shop scheduling problem. *European Journal of Operational Research*, 127(2):317–331.

Cai, X., Goh, C., and Mees, A. I. (1998). Greedy heuristics for rapid scheduling of trains on a single track. *IIE transactions*, 30(5):481–493.

Caprara, A., Fischetti, M., and Toth, P. (2002). Modeling and solving the train timetabling problem. *Operations research*, 50(5):851–861.

D'Ariano, A., Corman, F., Pacciarelli, D., and Pranzo, M. (2008). Reordering and local rerouting strategies to manage train traffic in real time. *Transportation science*, 42(4):405–419.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

D'ariano, A., Pacciarelli, D., and Pranzo, M. (2007). A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research*, 183(2):643–657.

https://proceedings.science/p/85446?lang=pt-br

Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41.

Hansen, I. A. (2010). Railway network timetabling and dynamic traffic management. *International Journal of Civil Engineering*, 8(1):19–32.

Hansen, P. and Mladenović, N. (1999). An introduction to variable neighborhood search. In *Meta-heuristics*, pages 433–458. Springer.

Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European journal of operational research*, 130(3):449–467.

Hansen, P. and Mladenovic, N. (2007). Variable neighborhood search methods.

Hassannayebi, E. and Zegordi, S. H. (2017). Variable and adaptive neighbourhood search algorithms for rail rapid transit timetabling problem. *Computers & Operations Research*, 78:439–453.

Huang, J.-y. (2006). Using ant colony optimization to solve train timetabling problem of mass rapid transit. In *JCIS*.

Ingolotti, L., Barber, F., Tormos, P., Lova, A., Salido, M. A., and Abril, M. (2005). A scheduling order-based method to solve timetabling problems. In *Conference of the Spanish Association for Artificial Intelligence*, pages 52–61. Springer.

Jünger, M. and Mutzel, P. (2012). *Graph drawing software*. Springer Science & Business Media.

Landex, A., Kaas, A. H., Schittenhelm, B., and Schneider-Tilli, J. (2006). Practical use of the uic 406 capacity leaflet by including timetable tools in the investigations. *WIT Transactions on The Built Environment*, 88.

Li, F., Gao, Z., Li, K., and Wang, D. Z. (2012). Train routing model and algorithm combined with train scheduling. *Journal of Transportation Engineering*, 139(1):81–91.

Mascis, A. and Pacciarelli, D. (2002). Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, 143(3):498–517.

Nõu, A. (1997). *Railway timetabling-Lagrangian heuristics*. Department of Mathematics, Royal Institute of Technology.

Samà, M., Corman, F., Pacciarelli, D., et al. (2017). A variable neighbourhood search for fast train scheduling and routing during disturbed railway traffic situations. *Computers & Operations Research*, 78:480–499.

Szpigel, B. (1973). Optimal train scheduling on a single line railway.

Tsuji, Y., Kuroda, M., Kitagawa, Y., and Imoto, Y. (2012). Ant colony optimization approach for solving rolling stock planning for passenger trains. In *System Integration (SII), 2012 IEEE/SICE International Symposium on*, pages 716–721. IEEE.

Yen, J. Y. (1971). Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716.

https://proceedings.science/p/85446?lang=pt-br