UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Planejamento e Roteamento de Trens Utilizando a Meta-heurística VNS e Técnicas em Grafos

DISSERTAÇÃO DE MESTRADO

HIGOR D'TALLES COSTA

Vitória-ES

2019

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Planejamento e Roteamento de Trens Utilizando a Meta-heurística VNS e Técnicas em Grafos

HIGOR D'TALLES COSTA

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica.

Orientador: Marcia Helena Moreira Paiva
Coorientador: Helder Roberto de Oliveira Rocha

Vitória-ES
2019

Ficha catalográfica disponibilizada pelo Sistema Integrado de Bibliotecas - SIBI/UFES e elaborada pelo autor

---

Costa, Higor D'Talles, 1979-
C837p Planejamento e roteamento de trens utilizando a meta heurística VNS e técnicas em grafos / Higor D'Talles Costa. - 2019.
128 f. : il.

Orientadora: Marcia Helena Moreira Paiva.
Coorientador: Helder Roberto de Oliveira Rocha.
Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Transporte - Planejamento. 2. Ferrovias - Tabelas de horários. 3. Otimização combinatória. 4. Programação heurística. I. Paiva, Marcia Helena Moreira. II. Rocha, Helder Roberto de Oliveira. III. Universidade Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU: 621.3

HIGOR D'TALLES COSTA

Planejamento e Roteamento de Trens Utilizando a Meta-heurística VNS e Técnicas em Grafos

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica.

Trabalho aprovado. Vitória-ES, 06 de Maio de 2019.

M. Paiva

Profa. Dra. Marcia Helena Moreira Paiva
Universidade Federal do Espírito Santo
Orientadora

Helder Rocha

Prof. Dr. Helder Roberto de Oliveira Rocha
Universidade Federal do Espírito Santo
Coorientador

Prof. Dr. Renato Elias Nunes de Moraes
Universidade Federal do Espírito Santo
Examinador Externo

Prof. Dr. Rodrigo de Alvarenga Rosa
Universidade Federal do Espírito Santo
Examinador Externo

Vitória-ES
2019

À minha família, por todo o amor, apoio e paciência pelos momentos em que não pude estar presente.

Agradecimentos

Agradeço primeiramente a Deus por me guiar nessa caminhada cercada de desafios concorrentes entre si. E agradeço por colocar em minha vida pessoas que me querem bem e me impulsionam para frente.

Aos meus pais, David e Carolina, por me transmitirem a confiança de que um homem com estudos tem grande valor não só para si mesmo como também para a sociedade. E por terem me dado todas as oportunidades possíveis para que eu me tornasse um grande homem.

Agradeço ao professor Marcelo Segatto, que acreditou em meu potencial e me apoiou na fase inicial do mestrado. Agradeço também à minha orientadora Marcia Helena Moreira Paiva, pelas explicações simples e objetivas, e também pelas provocações sutis e precisas, que me ajudaram a caminhar por caminhos até então desconhecidos, mas de grande valor para o meu crescimento enquanto pesquisador. Agradeço ao meu coorientador Helder Roberto de Oliveira Rocha, que contribuiu com considerações bastante relevantes para a minha pesquisa. Ao departamento de Engenharia Elétrica e à Universidade Federal do Espírito Santo, pela estrutura de ensino fornecida ao longo dos anos.

Aos colegas que ingressaram no curso de mestrado comigo e que de algum modo contribuíram para o meu desenvolvimento. Obrigado pelas conversas e reflexões sobre a vida e sobre como seguir em frente diante das dificuldades pelas quais passamos na pós-graduação.

Aos amigos da turma de Especialização em Engenharia Ferroviária 2011, pelas diversas contribuições que foram prontamente cedidas quando solicitadas.

Aos meus gestores da Vale S.A., que compreenderam a importância do mestrado para o meu desenvolvimento pessoal e profissional, além dos ganhos possíveis para a própria Companhia. Obrigado pela confiança e por me darem a oportunidade de assistir as aulas em horários de trabalho.

"You must understand that there is more than one path to the top of the mountain".

Miyamoto Musashi

Resumo

O modo de transporte ferroviário é um dos mais eficientes para o transporte terrestre de grandes volumes de carga a granel e de *commodities* diversas. Para que sua eficiência seja alta é necessário que as composições realizem poucas paradas e trafeguem à velocidade ótima definida para cada trecho de ferrovia. Um bom planejamento de rotas e de tráfego de trens pode garantir isso. Ambos os problemas podem ser resolvidos pela minimização de dois objetivos: número de conflitos entre rotas e tempo de parada dos trens em circulação. Uma rota é definida por um conjunto de nós sequenciais que conectam um nó de origem a um nó de destino em uma rede. Os conflitos ocorrem quando dois ou mais trens tentam utilizar o mesmo trecho de ferrovia em intervalos de tempo concorrentes. Para realizar a tarefa de otimização, foi proposto um método de solução composto por três etapas: (a) criação de rotas viáveis, (b) minimização de conflitos entre rotas e (c) redução de tempos de viagem. Na primeira etapa as rotas foram obtidas por técnicas em grafos. Na segunda etapa os conflitos e os tempos de viagem foram minimizados por uma meta-heurística. Por fim, na terceira etapa, os tempos de parada foram minimizados pela eliminação de atrasos desnecessários. Para a otimização da segunda etapa decidiu-se pela implementação de um algoritmo *Variable Neighborhood Search* (VNS) multiobjetivo. De fato, a versão multiobjetivo foi baseada no esquema *Skewed VNS*. Este algoritmo é bastante apropriado para a busca de soluções em vizinhanças muito distantes da solução corrente. A solução por métodos heurísticos foi proposta devido à natureza do problema, que é do tipo *NP-Hard* segundo a teoria da complexidade computacional. O algoritmo foi testado em 5 grafos planares aleatórios com n vértices e m arestas, onde n/m é dado por: 49/57, 94/112, 184/222, 274/332 e 364/442. O número de trens em circulação em cada grafo foi ajustado como sendo 20% do número de vértices de cada um deles. No caso mais complexo foi planejada a circulação de 73 trens. Foram criados cenários utilizando 1 ou 3 rotas por trem, para os quais foram utilizados grafos ponderados ou com pesos unitários em seus vértices. Além disso, foi utilizada a medida de centralidade *betweenness*. É importante ressaltar que, propositalmente, foram gerados muitos conflitos entre as rotas dos trens com a finalidade de testar o algoritmo em condições extremas. Para verificar a qualidade do processo de otimização foi construída a Fronteira de Pareto, cuja qualidade foi checada por três métricas sugeridas na literatura e uma quarta proposta neste trabalho. Os resultados obtidos foram muito bons frente aos objetivos propostos e à complexidade das topologias testadas.

Palavras-chave: Planejamento e roteamento de trens. Operação ferroviária. Otimização multiobjetivo. *Variable Neighborhood Search*. Fronteira de Pareto.

Abstract

Railway transportation is one of the most efficient means for overland transportation of high volumes of bulk cargo and a range of commodities. To be highly efficient it is necessary for trains to make few stops and run at optimal speeds defined for each section of railway. A good route, and train traffic planning, can ensure this. Both objectives can be attained by minimizing the two problems: the number of conflicted schedules along routes and the stopping time accrued by trains in service. A route is defined by a set of sequential nodes that connect a source node with a destination node in a network. Conflicts occur when two or more trains try to use the same railway resource at the same time. To optimize performance a method was put forward to find a solution, a method composed of three stages: (a) create viable routes, (b) minimize conflicting schedules of all rail routes and (c) reduce travel times. In the first stage, routes were obtained using graph techniques. In the second stage, conflicts and travel times were minimized by the use of a metaheuristic. Finally, in the third stage, stop times were minimized by eliminating unnecessary delays. To perform the optimization task of the second stage, we decided to implement a multi-objective version of the Variable Neighborhood Search (VNS) algorithm. In fact, the multi-objective version was based on the Skewed VNS scheme. This algorithm, regarding the current objective, is much more suitable for finding solutions in very distant Neighborhoods. Solution proposed by heuristic method, due to the nature of the problem, was NP-Hard regarding computational complexity theory. The developed algorithm was tested in 5 random planar graphs with n vertices and m edges, where n/m is given by: 49/57, 94/112, 184/222, 274/332 and 364/442. The number of running trains in each graph was adjusted to 20% of the number of vertices to each one of them. It was planned as having 73 trains running in the most complex case. Scenarios were created using 1 or 3 routes for each train, for which we used weighted graphs or graphs with unitary weights on their vertices. Furthermore, we used the betweenness centrality measure. It is relevant to emphasize that many conflicts among train routes were purposefully generated, aiming to test the algorithm in extreme conditions. To verify the optimization process quality a Pareto Front was built and its quality was checked using three measures suggested by some papers and a fourth one proposed by this research. The results were very good when compared to proposed objectives and to the complexity of tested topologies.

Keywords: Train scheduling and routing. Railway operations. Multi-objective optimization. Variable neighborhood search. Pareto Front.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
AM	Amplitude
BVNS	Basic Variable Neighborhood Search
CV	Coeficiente de Variação
DRC	Detectar e Resolver Conflitos
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
GVNS	General Variable Neighborhood Search
HV	Hypervolume
JSS	Job Shop Scheduling
KSP	K Shortest Loopless Paths
LP	Linear Programming
MILP	Mixed Integer Linear Programming
MO-SVNS	Multi-objective Skewed Variable Neighborhood
MOACO	Multi-objective Ant Colony Optimization
MS	Maximum Spread
NSGA-II	Non-dominated Sorting Genetic Algorithm II
RVNS	Reduced Variable Neighborhood Search
SP	Spacing
SSD	Sistema de Suporte à Decisão
SVNS	Skewed Variable Neighborhood Search
TS	Tabu Search
VND	Variable Neighborhood Descent
VNS	Variable Neighborhood Search

Sumário

1 Introdução

Dada uma rede formada por nós e arestas que os conectam, o problema de roteamento pode ser definido como sendo a busca pelos nós intermediários que conectam os nós de origem e destino para que uma entidade possa trafegar ao longo da rede. As entidades podem ser pacotes de dados em redes de telecomunicações, veículos nas ruas de uma cidade, trens em uma rede ferroviária, etc. As rotas possíveis para uma dada entidade que trafega em uma rede podem ser calculadas a *priori* ou a *posteriori*. Quando calculadas a *priori*, as rotas são mantidas inalteradas ao longo do tempo para uma entidade que trafega entre os nós de origem e destino. Nessa situação, um algoritmo poderá ser utilizado para escolher a melhor rota disponível levando em conta a necessidade de uso de recursos da rede, o tempo de tráfego aceitável entre origem e destino, entre outras características inerentes ao tráfego de entidades em uma rede. Por outro lado, quando calculadas a *posteriori*, os nós intermediários que conectam a origem ao destino podem variar em função das condições de tráfego da rede. Desse modo, as opções de rota do nó atual ao nó de destino são reavaliadas sempre que a entidade atinge um novo nó em seu caminho até o destino. As condições de tráfego podem mudar, por exemplo, em função de trechos muito congestionados ou da perda de ligações. Esse método também pode ser classificado como roteamento adaptativo. O roteamento adaptativo não é foco desta pesquisa.

As redes podem ter diferentes topologias, isto é, elas podem ter diferentes configurações físicas. As topologias podem ter as mais variadas configurações como estrela ou malha, por exemplo. A transmissão de dados de um nó a outro pode assumir qualquer caminho viável e depende de uma requisição de conexão. No entanto, é necessário que o caminho a ser utilizado tenha recursos disponíveis para permitir o tráfego de dados, senão a conexão é bloqueada. Os recursos podem ser canais de comunicação, capacidade dos canais, ligação física entre os nós da rede, etc. O bloqueio tem maior probabilidade de ocorrência em roteamentos mais simples onde as rotas não sofrem alterações em seus nós intermediários. Em técnicas de roteamento adaptativas, por outro lado, a taxa de bloqueio é menor embora exija maior custo de processamento. Se for considerado um projeto de rede óptica, pode-se destacar duas etapas principais: o projeto da topologia lógica, definida como sendo o conjunto dos caminhos ópticos, e o roteamento e alocação de comprimentos de onda (PAIVA, 2008).

As redes ópticas transportam dados por um par *origem–destino* utilizando um determinado comprimento de onda. Esses dados são empacotados e enviados pela rede passando por uma cadeia de roteadores segundo um esquema de comutação. Além disso, um canal de comunicação pode ser melhor aproveitado utilizando-se técnicas de multiplexação de sinais. A cada rota é atribuído um comprimento de onda diferente que esteja disponível.

Em termos de definição de rotas, regras de tráfego e uso de recursos da rede, o roteamento de trens em uma ferrovia apresenta diversas similaridades em relação ao modo como essas etapas são tratadas nas redes de telecomunicações. Em uma ferrovia são transportadas cargas de uma origem a um destino. A velocidade de cada um dos trens deve ser ajustada de tal modo que não ocorra qualquer colisão entre eles, decorrente de conflitos entre suas rotas. Um conflito ocorre quando um trecho de ferrovia é utilizado por dois ou mais trens em um mesmo intervalo de tempo. E, diferentemente das redes de telecomunicações que podem trabalhar com a comutação de pacotes, por exemplo, não é prático dividir um trem em composições menores ao longo do seu trajeto. Por isso, nesta pesquisa, um trem é considerado como sendo uma entidade indivisível que trafega entre os nós da rede. A passagem de um trem por um nó, seja qual for o sentido de tráfego, é permitida sempre que o nó em questão não tiver sido reservado para outro trem no mesmo intervalo de tempo.

Quando comparada a uma rede de comunicação, a ferrovia apresenta muitas similaridades. As semelhanças mais notáveis remetem aos nós e às ligações entre eles, à definição de rotas viáveis entre origem e destino, ao tráfego de entidades ao longo da rede bem como às regras de controle de fluxo de entidades. Este último está intrinsecamente ligado ao planejamento e à programação dos trens ao longo da rota definida para cada um deles de modo que não ocorram colisões ao longo do tempo.

O planejamento do transporte ferroviário envolve várias etapas encontradas também em outros modos 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. Frequentemente essas atividades são consideradas bastante difíceis de serem executadas devido à complexidade referente à interação entre trens e também devido ao tamanho das redes. As possibilidades de tráfego são limitadas pela disponibilidade de linhas férreas, aparelhos de mudança de via, condições de sinalização, tipo de carga transportada, entre outros fatores, conforme pontuado por Törnquist (2006).

Tratar problemas de planejamento e roteamento de entidades em redes pode ser uma tarefa bastante complicada. Contudo, algumas análises podem ser simplificados quando a rede é modelada pela teoria dos grafos. De acordo com Netto (2012), um grafo G é uma estrutura formada por um conjunto discreto de vértices V e por um conjunto não vazio de arestas E, definidas em função de V, tal que $G = (V, E)$. Nos grafos utilizados nesta pesquisa os vértices representam os trechos de ferrovia e as arestas representam as conexões entre eles.

O uso da teoria dos grafos apresenta muitas vantagens para análises de redes. Na literatura há uma gama de algoritmos bastante consolidados que se prestam a calcular

diversas medidas de centralidade que revelam características importantes sobre os grafos. Além disso, há algoritmos muito eficientes para encontrar menores caminhos entre dois nós quaisquer de uma rede. Esses algoritmos associados a uma visão simplificada e intuitiva proporcionada pela teoria dos grafos formam um ferramental importante que será utilizado ao longo deste trabalho.

1.1 Motivação do Trabalho

Um dos maiores custos da ferrovia é referente ao consumo de combustível (BRASIL, 2018). Para se ter ideia quanto ao diesel consumido, considere uma composição com 3 locomotivas cuja potência individual é de 4.000 HP e uma frota 252 vagões carregados de minério de ferro. Essa composição tem uma carga total aproximada de 25.180 toneladas. Para retirar esse trem do repouso e levá-lo à velocidade nominal da via, considerada como sendo de 65 km/h, seriam necessários, em média, 150 litros de diesel. Portanto, o fator energético é extremamente importante para o planejamento de rotas e também no tocante ao meio ambiente.

No melhor dos casos nenhum trem deveria parar ao longo de sua rota. Se essa fosse uma condição exequível, seria praticado o menor tempo possível de viagem. No entanto, por ser bastante comum que as ferrovias operem com linhas de fluxo bidirecional, os trens acabam concorrendo pelos mesmos trechos de ferrovia durante intervalos de tempo conflitantes. Uma vez que não é viável aguardar que um deles chegue ao seu destino para que então o próximo comece sua viagem, faz-se necessário parar um ou mais trens em locais e momentos bem determinados para que outros, que sejam prioritários, possam continuar sua viagem evitando atrasos. A lógica para definir quais trens devem sofrer atrasos e quais são os melhores trechos em que cada um deles deve parar pode tornar bastante complexa a etapa de planejamento e programação de cada um dos trens.

Conforme mencionado anteriormente, para que um trem possa utilizar um determinado trecho é preciso que o trecho em questão esteja liberado no momento desejado. Para isso é necessário atribuir atrasos a alguns dos trens. Entretanto, é importante que esses tempos não sejam longos a ponto de impactar negativamente o acordo de nível de serviço estabelecido com o cliente. Além disso, se os tempos forem muito curtos há a possibilidade de variar a velocidade do trem ao longo de um trecho de modo que ele sequer precise efetuar uma parada onde estava previsto. Motivado por encontrar tempos reduzidos de parada, é necessário encontrar meios de reduzir ao máximo os atrasos sofridos pelos trens. Embora seja estudado um modelo de planejamento de trens em um contexto diferente do proposto nesta pesquisa, Landex et al. (2006) propõem um método de redução de tempos de viagem com o objetivo de aumentar a capacidade de transporte da ferrovia, isto é, aumentar o número de trens que podem trafegar em um dado intervalo de tempo.

Esse estudo será utilizado como base para a implementação de um algoritmo adequado às condições desta pesquisa.

As demandas dos clientes determinam a origem e o destino de uma dada carga. Levando em conta a condição da via, no que tange a disponibilidade de trechos para circulação, os operadores definem as rotas e o horário de partida de cada um dos trens. À medida em que cresce o número de trens, torna-se cada vez mais difícil evitar que ocorram atrasos ao longo de suas rotas. Nesse caso, é preciso escolher bem o local e o tempo de atraso de modo que possíveis colisões sejam evitadas. Nesse contexto surge a necessidade de Sistemas de Suporte à Decisão (SSD), conforme mencionado por Samà et al. (2017). É importante que esses sistemas tenham respostas rápidas a eventos que possam ocorrer ao longo do tempo. Esses eventos podem ser decorrentes de mudanças nas condições de tráfego ou da programação dos trens. Por isso, é de fundamental importância que o SSD forneça informações relevantes aos operadores de tráfego com o objetivo de garantir uma operação segura e rápida de todos os trens. Os SSD precisam fornecer soluções viáveis já que os operadores de tráfego poderão não ter tempo hábil para verificar e efetuar qualquer ajuste em um curto intervalo de tempo.

Quanto à atuação dos operadores de tráfego, pode ser necessário que eles intervenham para eliminar manualmente os últimos conflitos. Essa atuação pode ocorrer no caso do sistema SSD não conseguir resolver todos os conflitos em um tempo computacional aceitável. Se isso ocorrer o algoritmo de busca é interrompido para que os conflitos restantes sejam eliminados manualmente. Esse cenário requer que o número de conflitos entre as rotas seja tratado como variável de decisão e não como restrição na modelagem do algoritmo de otimização. Afinal, os conflitos precisam ser tratados em sua totalidade para impedir que ocorram colisões entre trens.

Dado esse contexto, a questão central do problema de planejamento e roteamento de trens está em Detectar e Resolver Conflitos (DRC) de rotas entre os trens que circulam na ferrovia, garantindo os menores tempos possíveis de viagem. No processo são considerados dados da malha ferroviária tais como a máxima velocidade permitida em cada trecho, o comprimento de cada um deles, o conjunto de rotas dos trens com os horários de passagem ou de parada em cada ponto relevante da rede, posição e velocidade de cada trem em um dado instante de tempo. Como premissa, nenhum trem deve partir de um nó antes do horário programado e o tempo de atraso em cada um dos nós da rede deve ser o menor possível. O problema de DRC é caracterizado pelo roteamento de trens e pelas decisões de programação. Ele é baseado em restrições fixas e alternativas. As fixas dizem respeito à viabilidade da programação de um trem em especial, onde ocorre a reserva de recursos para ele. No caso das restrições alternativas, elas modelam a viabilidade global da programação de todos os trens. Assim, dado um trecho de ferrovia a ser percorrido por dois trens, enquanto um dos trens o ocupa o outro não pode ocupá-lo (SAMÀ et al., 2017).

O problema de planejamento e roteamento de trens é bastante estudado na literatura e, conforme comprovado por Caprara, Fischetti e Toth (2002), trata-se de um problema do tipo *NP-Hard* segundo a teoria da complexidade computacional (GOLDREICH, 2008). Sendo assim, o método de solução proposto nesta pesquisa utiliza métodos heurísticos para resolver o problema em tempo computacional aceitável.

1.2 Objetivo Geral

Visando atender às motivações supramencionadas, o objetivo geral desta pesquisa está em propor um método para solucionar o problema de planejamento e roteamento de trens em redes ferroviárias modeladas via grafos planares aleatórios.

Logo, dado o objetivo citado, neste trabalho não serão utilizadas ferrovias descritas por topologias em linha singela ou em linha dupla como muitos estudos consideram. A escolha das redes planares aleatórias trata o problema em questão de modo mais genérico, dispensando regras de tráfego de trens aplicáveis a casos e condições especiais próprias de topologias utilizadas em ferrovias reais.

Como parte da solução é necessário definir rotas viáveis para cada um dos trens em circulação, tratando as questões relativas ao planejamento e à programação de cada um deles de modo a minimizar ou eliminar todos os conflitos entre suas rotas, executando todas as viagens nos menores tempos possíveis sem abrir mão da segurança operacional. Assim, o método proposto pode ser resumido em três etapas, a saber:

- **Etapa 1:** Gerar rotas viáveis e a programação inicial de cada um dos trens;
- **Etapa 2:** Detectar e minimizar o número de conflitos existentes entre as rotas dos trens em circulação, ao passo em que o tempo de viagem também é minimizado;
- **Etapa 3:** Reduzir os tempos de atraso ou de parada para que o tempo de viagem dos trens seja minimizado em relação à solução obtida na *Etapa 2*.

1.2.1 Cenários Analisados

Para testar o método proposto e verificar a qualidade da solução encontrada foram elaborados doze cenários de teste. Eles possibilitam analisar os impactos decorrentes do uso de uma ou mais rotas por trem, do uso de grafos ponderados ou com pesos unitários em seus vértices e do uso da medida de centralidade *betweenness* para o cálculo das rotas. Essa métrica será discutida no Capítulo 3, página 35.

Os cenários propostos são descritos a seguir:

- **Cenário 1:** É utilizado grafo ponderado e a rota de menor custo é calculada com base nesses pesos. O custo está associado ao peso dos vértices do grafo. No caso, os pesos são relativos ao comprimento de cada um dos trechos de ferrovia. É fornecida uma única rota por trem fazendo com que o foco do problema esteja em planejar e programar cada um deles de modo a evitar conflitos;
- **Cenário 2:** Esse cenário é similar ao anterior, mas nesse caso o grafo não é ponderado. Os pesos utilizados são todos iguais e recebem valor unitário;
- **Cenário 3:** Esse cenário é semelhante ao primeiro, embora sejam calculadas três opções de rota para cada um dos trens. Nesse caso ambos os algoritmos de roteamento e de planejamento são utilizados para que a solução do problema possa ser obtida pelo método proposto;
- **Cenário 4:** Esse cenário apresenta similaridades com o *Cenário 2*. A diferença é que neste caso são calculadas três opções de rota para cada um dos trens e os grafos não são ponderados;
- **Cenários 5 a 8:** Similares aos *Cenários 1 a 4*, respectivamente, mas nesses casos é calculado o valor da métrica *betweenness* para cada um dos vértices. Com base nesses valores é calculada a rota de menor custo;
- **Cenário 9 a 12:** Nesses cenários são calculadas três opções de rota para cada um dos trens. Nos quatro cenários são utilizados grafos ponderados e o tempo de processamento é limitado e propositalmente insuficiente para que todos os conflitos sejam eliminados. Nos *Cenários 9 e 10* o roteamento é realizado em função da ponderação dos vértices e nos *Cenários 11 e 12* o roteamento é realizado em função da métrica *betweenness ponderada*.

1.2.2 Organização da Dissertação

O Capítulo 2 apresenta a revisão bibliográfica. O Capítulo 3 descreve os detalhes sobre o método de solução proposto para o problema de planejamento e roteamento de trens. No Capítulo 4 são apresentados esquemas de otimização utilizando a meta-heurística *Variable Neighborhood Search* (VNS), incluindo a busca local, as estruturas de vizinhança e a heurística de redução dos tempos de viagem. Em seguida, Capítulo 5, é apresentada a teoria sobre a Fronteira de Pareto e algumas métricas para mensurar sua qualidade, incluindo uma métrica proposta pelo autor. No Capítulo 6 são mostrados todos os detalhes dos experimentos realizados bem como os resultados obtidos. E, por fim, no Capítulo 7 é feita a análise dos resultados, a conclusão do estudo e em seguida são apontadas algumas propostas para trabalhos futuros. Foram incluídos ainda dois apêndices, sendo que no Apêndice A é discutida a estrutura da aplicação desenvolvida e no Apêndice B são mostrados todos os resultados obtidos ao longo da pesquisa.

2 Trabalhos Relacionados

Nesse capítulo é feita uma breve revisão dos conceitos de planejamento e roteamento de trens. Em seguida é apresentado um problema clássico bastante utilizado como modelo para resolver o problema de planejamento e roteamento de trens. Na sequência, são abordados os métodos de otimização frequentemente encontrados na literatura e algumas de suas particularidades são apresentadas. Por fim, é feita a delimitação do tema, onde é apresentado o modo como o problema em questão será abordado nesta pesquisa.

2.1 Revisão da Literatura

O problema de planejamento e roteamento de trens tem ganhado cada vez mais atenção da comunidade científica. Segundo Pinedo (2005), o planejamento é um processo decisório amplamente utilizado na indústria, nos transportes, no processamento de informações e nas comunicações. Um bom planejamento depende de métodos matemáticos e heurísticos para alocar recursos limitados a atividades que precisam ser realizadas. Essa alocação deve ser feita da melhor maneira possível de tal modo que se possa garantir um tráfego seguro e eficiente das entidades que trafegam por uma determinada rede. No caso da ferrovia, os trens são as entidades que trafegam ao longo da rede ferroviária. Quanto ao roteamento, trata-se de definir rotas adequadas para que uma entidade trafegue entre dois nós da rede, denominados nós de *origem* e de *destino*. A rota pode ser definida como sendo o conjunto de nós que conectam a *origem* ao *destino*.

Quando um trem executa uma rota ele parte da *origem* para o *destino* passando sequencialmente por todos os trechos que compõem sua rota. Por se tratar de um processo sequencial, esse problema se torna semelhante ao problema de sequenciamento de máquinas em uma linha de produção. Devido a essa semelhança, não raramente é sugerido modelar o problema de planejamento e roteamento de trens como um caso particular do problema *Job Shop Scheduling* (JSS). Szpigel (1972) foi um dos primeiros a propor essa abordagem. Ele afirma que definir a ordem em que os trens devem ocupar trechos de uma linha férrea é similar a definir o sequenciamento de tarefas em um conjunto de máquinas.

Quanto à implementação do JSS, alguns autores o fazem utilizando grafos de conflito, que segundo Blażewicz, Pesch e Sterna (2000), é um tipo de grafo bastante utilizado para resolver problemas dessa natureza. Esses grafos são capazes de representar todas as possibilidades de planejamento de rotas de trens. Mascis e Pacciarelli (2002) realizaram uma pesquisa onde esses grafos foram utilizados. Eles propuseram uma generalização pela verificação de aplicabilidade a alguns dos casos mais bem sucedidos da literatura referente ao problema de JSS aplicado a ferrovias. Alguns autores relatam, inclusive,

excelentes resultados oriundos de casos reais de planejamento e roteamento de trens utilizando essa abordagem (D'ARIANO; PACCIARELLI; PRANZO, 2007; SAMÀ et al., 2017). Além disso, há ferramentas em constante desenvolvimento que são utilizadas por projetos europeus onde é necessário um sistema robusto de suporte à decisão no processo de operação ferroviária, conforme indicado no trabalho de D'Ariano et al. (2008). A robustez de um sistema de planejamento e roteamento de trens frequentemente é mensurada em função da capacidade de absorção de pequenos distúrbios que podem ocorrer durante a operação ferroviária, conforme descrito por Hansen e Pachl (2014).

Em geral os problemas de planejamento e roteamento são de difícil solução. Por isso, abordagens matemáticas mais tradicionais como algoritmos baseados em *Linear Programming* (LP) podem levar a soluções muito complexas ou mesmo insatisfatórias do ponto de vista do tempo computacional, ainda que a solução ótima possa ser encontrada. Segundo Cai, Goh e Mees (1998), os problemas de planejamento e roteamento podem ser modelados com ferramentas de otimização matemática bastante sofisticadas levando a resultados muito expressivos. Um dos algoritmos mais utilizados na solução de problemas utilizando LP é o *branch-and-bound*. Esse algoritmo foi proposto pela primeira vez por Szpigel (1973) para resolver o problema de planejamento de trens em linha singela, isto é, linha única para passagem de trens em ambos os sentidos. Entretanto, Cai, Goh e Mees (1998) defendem que essas ferramentas devem ser restringidas à solução de problemas bem reduzidos e com poucas entidades trafegando nas redes. Essa recomendação se dá pelo fato de que se trata de um problema de difícil solução com características estocásticas e não lineares. Devido a esses fatores, tem sido cada vez mais frequente encontrar abordagens por métodos heurísticos, onde as meta-heurísticas estão incluídas.

Apesar das limitações relativas ao uso de LP para a modelagem do problema de planejamento e roteamento de trens, frequentemente esse método tem sido utilizado. Contudo, ele costuma ser empregado na etapa de obtenção da solução inicial. Nesse caso, em geral o algoritmo LP é interrompido após um tempo pré-determinado e a solução obtida até o momento é utilizada como solução inicial para meta-heurísticas que fazem a otimização das variáveis de interesse. Esse tipo de abordagem pode ser visto nos trabalhos de Burdett e Kozan (2009), Li et al. (2012), Samà et al. (2016) e Samà et al. (2017).

Nas pesquisas de D'ariano, Pacciarelli e Pranzo (2007), D'Ariano et al. (2008), Burdett e Kozan (2009), Caimi et al. (2011), Pellegrini, Marlière e Rodriguez (2014), Samà et al. (2016) e Samà et al. (2017), boas soluções são obtidas pela utilização de um algoritmo *Mixed-integer Linear Programming* (MILP). Nesses trabalhos são utilizadas rotas fixas para cada um dos trens. De modo complementar, nos trabalhos de D'Ariano et al. (2008), Caimi et al. (2011), Samà et al. (2016) e Samà et al. (2017) a solução pelo MILP é interrompida depois de um tempo limite e daí por diante é executada uma etapa de busca local ou alguma meta-heurística com o objetivo de rerrotear alguns trens de modo que a

solução corrente possa ser melhorada. Na pesquisa de D'Ariano et al. (2008) é utilizada somente uma etapa de busca local para rerrotear os trens que não puderam ser roteados pelo algoritmo *branch-and-bound* implementado pelo MILP. A meta-heurística *Tabu Search* é utilizada na pesquisa desenvolvida por Samà et al. (2016). E no trabalho de Samà et al. (2017), é utilizada tanto a meta-heurística *Tabu Search* quanto a meta-heurística *Variable Neighborhood Search* para que boas soluções ou mesmo soluções ótimas pudessem ser obtidas.

Caimi et al. (2011) utilizaram uma abordagem bastante diferente das anteriores. Segundo eles, o grafo de conflito comumente utilizado em alguns trabalhos acaba não gerando uma solução muito boa pela relaxação adotada na implementação em LP. O modelo proposto por eles cria uma árvore com a sequência de recursos utilizados pela rota de cada um dos trens. Com essa árvore foram obtidos resultados em tempos computacionais bastante reduzidos quando comparados à abordagem que utilizava os grafos de conflito. Na prática, essa árvore é utilizada para variar as rotas dos trens em busca de melhores soluções para o problema de planejamento.

Dentre os trabalhos supramencionados, com exceção do trabalho de Sun, Cao e Wu (2014), todos fazem a minimização tanto do tempo de atraso quanto do número de conflitos entre os trens. No trabalho mencionado como exceção, é criado um modelo para otimizar três objetivos: tempo médio de viagem, consumo de energia dos trens e tempo total de atraso. Foi utilizada a meta-heurística *Genetic Algorithm*. Nos testes realizados foi verificado que o algoritmo proposto foi capaz de reagir rapidamente a distúrbios ocorridos ao longo da rota dos trens.

Quanto à detecção de conflitos entre as rotas dos trens em circulação, em todos os trabalhos citados anteriormente foi utilizado um método baseado em blocos de tempo. Nesse método é feita a atribuição de um bloco de tempo para cada um dos trechos que compõem a rota de um trem. Desse modo, os conflitos podem ser detectados por qualquer ocorrência de sobreposição entre esses blocos. Essa técnica garante que haja um tempo mínimo entre os trens em circulação, independentemente da distância física entre eles.

2.2 Delimitação do Tema e Contribuições

Nesta pesquisa o problema de planejamento e roteamento de trens será modelado via grafos. Todavia, diferentemente das abordagens apresentadas na Seção 2.1, não serão utilizados os grafos de conflito. Além disso, o problema não será modelado como um caso particular do JSS. Quanto ao método de solução, serão utilizados somente métodos heurísticos. Por fim, não serão abordadas nesta pesquisa as técnicas em LP bem como o algoritmo *branch-and-bound*.

O problema será modelado via grafos planares aleatórios. Nesses grafos não ocorrem

cruzamentos entre sua arestas. Isso implica dizer que não é permitido o cruzamento entre linhas férreas como ocorre em algumas ferrovias reais. Os grafos foram gerados de modo aleatório e têm grau médio igual a $2,4$. O grau médio de um grafo é dado por $2m/n$, onde m é o número de arestas e n o número de vértices.

Quanto ao cálculo das rotas dos trens, será utilizado um algoritmo proposto por Yen (1971) para o cálculo dos *K Shortest Loopless Paths* (KSP). Esse algoritmo é um dos mais eficientes para este cálculo. Com ele as rotas podem ser calculadas com base nos pesos atribuídos a cada um dos vértices do grafo. De acordo com Yen (1971), geralmente esse algoritmo é extremamente eficiente quando comparado a outros importantes algoritmos encontrados na literatura. E, como alternativa para a geração das rotas, será utilizada a métrica *betweenness*. Essa métrica é proporcional à probabilidade de um vértice ser intermediário na comunicação entre outros vértices do grafo. Então, se os pesos dos vértices forem substituídos pelo valor da métrica *betweenness* referente a cada um dos vértices, as referências passadas para o algoritmo KSP serão diferentes. Com isso, as regiões do grafo onde havia maior probabilidade de congestionamento poderão deixar de ser tão disputadas no cálculo das rotas. Logo, o emprego dessa métrica contribui para uma melhor distribuição das rotas ao longo dos trechos da ferrovia.

Para os casos em que são utilizados os blocos de tempo, conforme mencionado na Seção 2.1, a norma *UIC code 406* propõe um método de cálculo de capacidade para uma ferrovia (LEAFLET, 2013). Essa norma foi discutida por Landex et al. (2006) no contexto do transporte ferroviário da Dinamarca. Ele descreveu, em termos práticos, como o tempo ocioso da ferrovia poderia ser reduzido para que sua capacidade de transporte pudesse ser expandida. O efeito prático é que os trens possam circular mais próximos uns dos outros.

O conceito de redução de tempos de viagem será utilizado nesta pesquisa com o objetivo de minimizar os tempos de atraso ou de parada dos trens em circulação. Entretanto, a ideia em questão será adaptada para ser utilizada em ferrovias cuja topologia é baseada em circuitos de via. Esses circuitos têm a função de delimitar a extensão lógica de cada um dos trechos da ferrovia. Um dos pontos positivos dessa abordagem é em relação ao monitoramento da posição dos trens em circulação. Os trens podem ser monitorados ao longo do tempo à medida em que avançam pelos circuitos de via correspondentes aos trechos que compõem sua rota.

Esta pesquisa se propõe a solucionar o problema de planejamento e roteamento de trens propondo um método de solução que difere em vários aspectos daqueles encontrados na literatura. No melhor do nosso conhecimento, este é o primeiro trabalho que se propõe a resolver o problema em questão utilizando grafos planares aleatórios e a medida de centralidade *betweenness* para o cálculo de rotas. Além disso, neste trabalho esses conceitos são combinados. As principais contribuições desta pesquisa são descritas a seguir:

1. Modelagem da ferrovia via grafos planares aleatórios. O modelo proposto faz a conversão dos trechos da ferrovia como sendo os vértices do grafo. As arestas representam as conexões entre os trechos. Na literatura há modelos similares, mas não foi encontrado um modelo que utilizasse grafos dessa natureza. Essa abordagem contribui para que o método de solução proposto nesta pesquisa possa ser utilizado em diversas redes, extrapolando as redes ferroviárias.

2. Utilização da métrica *betweenness* no processo de roteamento. São realizados comparativos entre soluções onde as rotas dos trens são calculadas com base em grafos ponderados, grafos com pesos unitários e grafos cujos pesos são relativos ao valor da métrica *betweenness*, calculada para as duas situações anteriores. Em todos os casos os pesos são atribuídos aos vértices do grafo. Esses comparativos permitem avaliar a influência dos parâmetros da rede no cálculo dos menores caminhos e, consequentemente, permitem avaliar os impactos provocados no planejamento do tráfego de trens ao longo do tempo.

3. Algoritmo de redução dos tempos de viagem via eliminação dos tempos de atraso dos trens. Os tempos considerados desnecessários são mapeados e eliminados sem que qualquer novo conflito seja gerado entre as rotas dos trens. Esse algoritmo recebe como entrada a solução obtida na etapa de minimização de conflitos e de tempos de parada dos trens em circulação. O algoritmo de redução de tempos considera o uso de ferrovias modeladas por circuitos de via.

4. Proposta da métrica *Amplitude* para auxiliar na escolha da Fronteira de Pareto quando não houver uma referência na literatura. A referência em questão se dá tanto em função do problema quanto do grafo analisado.

5. Critério de seleção da Fronteira de Pareto. Dentre um conjunto de execuções do método de solução aplicado a uma determinada instância de teste, são geradas várias fronteiras com base nos objetivos otimizados. Essas fronteiras são avaliadas e a melhor entre elas é escolhida como sendo a Fronteira de Pareto para a instância testada. A métrica *Amplitude* é um dos parâmetros avaliados no processo de seleção.

3 Modelagem do Problema

Neste capítulo são apresentadas as partes que compõem o método proposto nesta pesquisa para solucionar o problema de planejamento e roteamento de trens. São apresentadas a modelagem da ferrovia via grafos, as características das instâncias de teste utilizadas e a aplicação dos conceitos de planejamento e programação de trens. Em seguida é apresentado um exemplo de tráfego de trens para que se possa familiarizar com os métodos e termos utilizados ao longo dessa pesquisa. Ao final são apresentadas as partes que compõem o método de solução proposto nesta pesquisa para lidar com o problema de planejamento e roteamento de trens em redes aleatórias planares modeladas via grafos.

3.1 Visão Geral do Método de Solução Proposto

De acordo com Talbi (2009), o termo *heurística* se refere à arte de descobrir estratégias para resolver problemas. As heurísticas são utilizadas para resolver problemas específicos de otimização. As meta-heurísticas, por sua vez, são métodos gerais mais avançados utilizados para solucionar problemas de otimização. As meta-heurísticas podem ser utilizadas como base para a criação de heurísticas ou métodos heurísticos desenvolvidos para a solução de problemas específicos.

Em termos práticos, as heurísticas desenvolvidas para problemas específicos correm o risco de ficarem presas em ótimos locais sem que soluções melhores sejam encontradas. Por outro lado, por serem mais genéricas, as meta-heurísticas são métodos heurísticos capazes de escapar de ótimos locais contribuindo para que a solução ótima global possa ser encontrada. Isso as torna mais robustas já que elas têm maior capacidade de diversificação e de intensificação ao longo do processo de busca por soluções.

Nesta pesquisa é proposto um método heurístico composto por três etapas para solucionar o problema de planejamento e roteamento de trens em redes aleatórias planares modeladas via grafos. Essas etapas são mostradas na Figura 1. A etapa de geração de rotas viáveis tem como entrada a ferrovia modelada via grafos, conforme descrito na Seção 3.2. Essa modelagem é uma das fases mais importantes e necessárias para garantir o correto funcionamento do método proposto. As rotas são calculadas em função dos vértices do grafo conforme explicado na Seção 3.3. Na etapa seguinte, todos os conflitos existentes entre as rotas dos trens em circulação são identificados e minimizados ou completamente eliminados. Os detalhes sobre esse procedimento são apresentados na Seção 3.4. Nessa etapa é utilizada a meta-heurística MO-SVNS com o objetivo de desembaralhar as rotas. Esse processo é executado com foco em encontrar os menores tempos de atraso que podem ser atribuídos aos trechos de rota de cada um dos trens. Por fim, na terceira etapa, foi

proposta uma heurística de busca de tempos de atraso que podem ser eliminados sem que haja qualquer prejuízo em relação aos resultados obtidos na etapa anterior. Ao reduzir os atrasos consequentemente são reduzidos os tempos de viagem dos trens. Os detalhes sobre essa etapa podem ser vistos na Seção 3.5.

Figura 1 – Fluxo básico do método de solução do problema de planejamento e roteamento de trens.

Fonte: O autor

3.2 Modelagem da Ferrovia via Grafos

Frequentemente o problema de planejamento e roteamento de trens é abordado como sendo um caso especial do problema JSS. Em geral essa abordagem é utilizada para ferrovias em linha singela e o problema costuma ser modelado por técnicas de programação linear. O modelo do JSS não é recomendado para solucionar problemas com muitos trens em circulação, conforme mencionado por Cai, Goh e Mees (1998).

Netto (2012) define que um grafo G é uma estrutura formada por um conjunto discreto de vértices V e por um conjunto não vazio de arestas E, definidas em função de V, tal que $G = (V, E)$. Em complemento a essa definição, de acordo com Benjamin, Chartrand e Zhang (2015), um grafo G é planar se ele pode ser desenhado em um plano de tal modo que quaisquer duas de suas arestas não se cruzem.

Como o foco deste estudo está nas redes aleatórias planares modeladas via grafos, isso impossibilita a modelagem do problema como um caso do JSS devido à complexidade que uma rede pode assumir. A rede em questão é entendida como sendo uma instância de ferrovia composta por n vértices e m arestas por onde circulam T trens. As características completas das redes testadas podem ser vistas na Tabela 1, página 34. Os grafos planares foram utilizados por não permitirem que ocorra qualquer cruzamento entre suas arestas. Com o uso de grafos aleatórios a ferrovia pode ser modelada de modo bastante genérico e independente de regras de tráfego muito específicas. Por outro lado, em uma rede aleatória pode haver muitos caminhos possíveis entre uma origem e um destino escolhidos, fato que pode acarretar em perda de desempenho das heurísticas utilizadas no método de solução proposto nesta pesquisa.

Muito embora a ferrovia em linha singela ou em linha dupla não seja foco deste trabalho, um trecho de ferrovia em linha dupla foi utilizado como referência para a elabo-

ração do modelo segundo a teoria dos grafos. Mais adiante esse modelo será extrapolado para que possam ser utilizados grafos aleatórios, conforme mencionado anteriormente. Neste caso, devido à generalização da rede ferroviária, as regras de tráfego comumente encontradas em ferrovias reais deixam de ser aplicáveis. Isso implica dizer que se for feita uma adaptação do método de planejamento e roteamento de trens desenvolvido neste trabalho para ser utilizado em uma ferrovia real, seria necessário realizar uma série de ajustes e adequações para que a solução encontrada estivesse em conformidade com as melhores práticas de operação e segurança operacional. Os detalhes sobre esses assuntos não serão abordados nesta pesquisa.

A Figura 2 mostra um trecho padrão da linha principal de uma ferrovia em linha dupla, cujas linhas são dadas por *L1* e *L2*. Na Figura 2a as seções numeradas se referem aos trechos de ferrovia por onde passam os trens. Aquelas numeradas de 15 a 20 se referem às chaves que possibilitam a troca de uma composição de uma linha para outra. As letras denotam os limites de cada um dos trechos de ferrovia e representam as conexões entre eles. Sempre que um trem passa por um trecho ele o ocupa completamente impossibilitando que outro trem acesse o mesmo trecho em intervalos de tempo conflitantes. Um trecho só é liberado para uso quando o trem que o ocupa atingir o próximo trecho de sua rota em direção ao destino planejado. Esse trecho de ferrovia foi convertido para um modelo em grafo, mostrado na Figura 2b. Os trechos numerados foram associados aos vértices do grafo, enquanto as letras foram associadas à ligação entre eles, isto é, às arestas do grafo. Por exemplo, conforme pode ser visto na Figura 2b, o vértice 2 simboliza o encontro entre os trechos 2, 15 e 16 da Figura 2a. Essa relação foi feita dessa forma uma vez que o encontro entre esses três vértices é representado por um único circuito de via. O circuito de via delimita um trecho lógico de ferrovia, onde só pode haver um único trem por vez, conforme mencionado anteriormente. Quanto às distâncias associadas a cada trecho de ferrovia, na conversão para o modelo em grafos essa distância foi associada ao peso dado aos vértices e não às arestas como é bastante comum encontrar em problemas envolvendo o uso da teoria dos grafos.

Tomando como referência a Figura 2b, é definido um par *origem–destino* dado por $r(3;13)$. Para este par, a rota possível seria $R(3;4;12;13)$, onde R denomina a rota. Para que a rota R possa ser realizada por um trem T, este deve passar em sequência pelos vértices $3 \rightarrow 4 \rightarrow 12 \rightarrow 13$. Isso implica dizer que para completar essa rota o trem deverá percorrer a distância completa referente a cada um dos quatro vértices, isto é, a distância será igual à soma dos pesos dos vértices que compõem a rota do trem em questão.

Desde os primeiros testes computacionais realizados nesta pesquisa foram avaliados diversos modelos de ferrovias modeladas por grafos. Entretanto, a conversão apresentada na Figura 2 foi a que trouxe mais vantagens para o método de solução proposto nesta pesquisa. Um modelo em grafo semelhante ao apresentado aqui foi também adotado

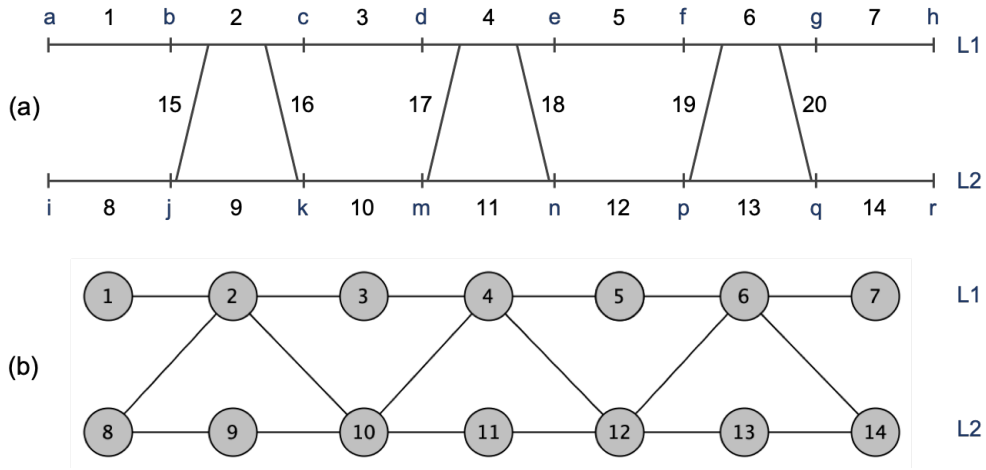

Figura 2 – Equivalência entre: (a) ferrovia real e (b) ferrovia modelada via grafos.

Fonte: O autor

por D'Ariano et al. (2008), onde os vértices correspondem aos trechos de ferrovia. No que tange à topologia adotada nesta pesquisa, a maior similaridade foi encontrada no trabalho de Burdett e Kozan (2009), que utilizou o modelo JSS para resolver o problema de planejamento de trens. A Figura 3 mostra o modelo utilizado por ele. Na Figura 3a é mostrada a linha de produção do JSS. Esse modelo é bastante similar ao apresentado na Figura 2a. E na Figura 3b, é mostrada a conversão da linha de produção para um modelo em grafo. O grafo resultante é bastante similar ao apresentado na Figura 2b.

O modelo apresentado por Burdett e Kozan (2009) foi avaliado nesta pesquisa, mas após alguns testes e motivado pelo uso dos grafos aleatórios, decidiu-se pela eliminação de alguns vértices. Em especial foram eliminados os vértices relativos aos rótulos $M7$ e $M8$, Figura 3b.

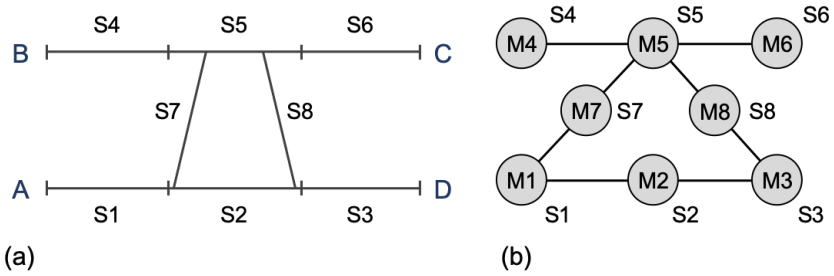

Figura 3 – Modelagem do problema JSS: (a) linhas de produção em paralelo com pontos de cruzamento e (b) leiaute de máquinas.

Fonte: Adaptado de Burdett e Kozan (2009)

Uma vez compreendida a lógica de criação do modelo ferroviário segundo a teoria dos grafos, o modelo apresentado pode ser abstraído para as redes aleatórias planares, foco

deste trabalho. A título de ilustração, na Figura 4 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. O tamanho de cada vértice é dado em função do seu grau, isto é, em função do número de arestas conectadas a ele.

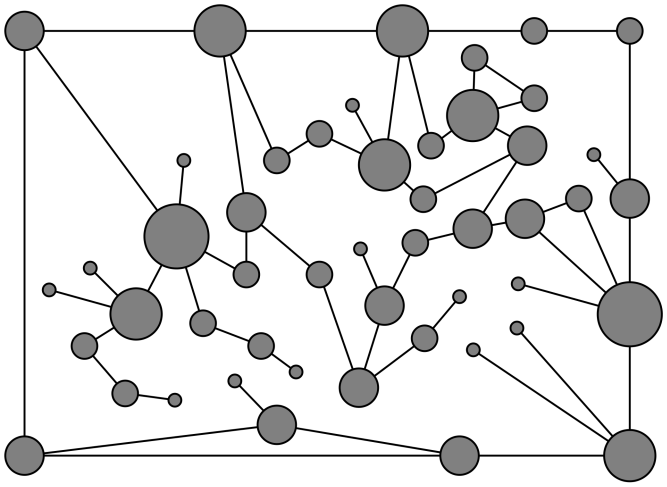

Figura 4 – Modelagem de uma ferrovia com grafo planar de 49 vértices e 57 arestas.

Fonte: O autor

3.2.1 Instâncias de Teste

Foram realizados experimentos com 5 grafos distintos. As características completas de cada um deles são mostradas na Tabela 1, onde o título de cada coluna é o nome que identifica a instância.

A geração dos grafos foi realizada de modo aleatório pela aplicação *yEd Graph Editor* (JÜNGER; MUTZEL, 2012). A única ressalva foi a proibição de laços. Conforme definido por Netto (2012), laços são arestas que envolvem apenas um vértice. Os grafos utilizados são planares e têm vértices e arestas tais que o grau médio seja de 2,4. O grau médio é mantido constante enquanto o tamanho dos grafos aumenta de uma instância para outra. Isso garante que as instâncias cresçam a taxas constantes tanto em relação ao número de vértices quanto ao número de arestas. Com exceção do primeiro grafo, a variação de uma rede para outra é de 90 vértices e 110 arestas. Essa variação é descrita por *VarVertices* e *VarArestas*, respectivamente para os vértices e arestas.

A cada vértice foi atribuído um peso que representa a distância do vértice atual até o próximo vértice da rota. As topologias foram testadas com um número de trens igual a 20% do número de vértices da rede em teste. Esse percentual foi escolhido com base na extensão da linha férrea e no número de trens que circularam na Ferrovia Vitória a Minas

no ano de 2011. Apesar desses dados serem antigos em relação à data da presente pesquisa, eles servem tão somente para validarem as premissas utilizadas neste trabalho quanto à densidade de trens que trafegam na rede. Os detalhes acerca dos dados mencionados são sigilosos e por isso não serão fornecidos neste trabalho.

O número de trechos para cada instância corresponde ao número total de trechos utilizados pelas rotas de todos os trens. Na Tabela 1 esse número corresponde aos casos em que foram utilizadas as rotas mais curtas entre a origem e o destino de cada trem. Essa condição também é válida para o número de conflitos mostrado na tabela.

Tabela 1 – Características das instâncias utilizadas.

Parâmetros	49v57a	94v112a	184v222a	274v332a	364v442a
Topologia	Planar	Planar	Planar	Planar	Planar
Vértices	49	94	184	274	364
Arestas	57	112	222	332	442
VarVertices	45	90	90	90	90
VarArestas	55	110	110	110	110
Trens	10	19	37	55	73
Trechos	66	138	472	650	976
Conflitos	8	23	152	201	319
Grau médio	2,3	2,4	2,4	2,4	2,4

3.3 Planejamento e Roteamento 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 atrasos sofridos, do início ao final da rota.

Em termos de planejamento, se o número de trens for muito grande em relação à rede, fica claro que o número de conflitos também será. Do ponto de vista da otimização combinatória isso poderia demandar tempo de processamento bastante elevado já que trata-se de um problema do tipo *NP-Hard* segundo a teoria da complexidade computacional. Essa classificação foi comprovada por Caprara, Fischetti e Toth (2002). Além disso, com um número elevado de conflitos poderia ocorrer a inserção de muitos eventos de atraso de trens ou ainda tempos de atraso muito longos. Esses tempos são necessários para possibilitar que os cruzamentos entre os trens ocorram conforme esperado. Com isso, é preciso que o número de conflitos seja minimizado ou idealmente igual a zero. Também podem ser gerados conflitos indesejados caso os trens saiam de estações muito próximas e com pouco espaçamento no tempo em relação ao horário de partida de cada um deles. Para evitar sobrecargas na rede ferroviária é necessário que na fase de planejamento sejam consideradas as características da ferrovia, a carga que deve ser transportada bem como os prazos acordados com os clientes.

Nesta pesquisa a programação de cada um dos trens foi realizada em função do horário de partida do vértice de origem e do tempo médio de cruzamento relativo a cada

um dos trechos, dado pela razão entre o comprimento, em km, e a velocidade média permitida, em km/h. A velocidade permitida em cada trecho varia entre 30 e 65 km/h. O limite de velocidade inferior, em geral, está associado ao tráfego em regiões com restrições definidas pela operação ou manutenção. O limite superior, por sua vez, representa o limite seguro de tráfego para o perfil da ferrovia. Essas referências podem ser diferentes para cada ferrovia a depender, principalmente, de suas características construtivas. Quanto ao comprimento de cada trecho, nas redes testadas ele varia entre 5 e 10 km. Tanto o comprimento quanto a velocidade foram definidos segundo uma distribuição aleatória.

As rotas para cada um dos trens foram obtidas com base no algoritmo *K Shortest Loopless Paths* (KSP) proposto por Yen (1971). Internamente é utilizado o algoritmo proposto por Dijkstra (1959) para o cálculo do caminho mais curto entre dois pontos de interesse em uma rede. Ambos os algoritmos são aplicáveis a grafos. Quanto ao número de rotas calculadas, nesta pesquisa optou-se por realizar experimentos considerando dois casos distintos: (a) cálculo de uma única rota por trem e (b) cálculo de três opções de rota por trem.

No primeiro caso foi considerado somente o cálculo da rota de menor custo para cada um dos trens. Nesse caso, o caminho de menor custo é aquele retornado diretamente pela função *Dijkstra* com base nos pesos dos vértices do grafo. Nesse cenário a busca pela solução do problema se resume a encontrar os tempos de atraso que devem ser atribuídos a cada um dos trechos que compõem a rota de cada um dos trens. Trata-se de resolver o problema de planejamento de trens. Desse modo, entre outros parâmetros, os trens carregam em si as referências de tempo de chegada, parada e partida de cada um dos trechos para o próximo.

Por outro lado, no segundo caso, foram calculadas as três rotas de menor custo para cada um dos trens e então foi escolhida aquela que acarretasse no menor número de conflitos ao inserir o trem em questão no contexto dos trens já inseridos nas fases anteriores do processo de escolha de rotas. Nesse cenário o foco da solução do problema está em calcular as três rotas de menor custo para cada um dos trens, escolher a mais adequada e então tratar o problema de planejamento de trens conforme o primeiro cenário.

3.3.1 Medida de Centralidade *Betweenness*

Segundo Freitas (2010), as medidas de centralidade são relações que tentam descrever a importância dos elementos de uma rede. Essas medidas levam em consideração o modo como cada elemento interage em uma rede, sendo que os elementos mais importantes são aqueles que exercem maior influência e por isso são mais estratégicos. Seu cálculo é dado em função de invariantes de um grafo. Há diversas medidas propostas na literatura, incluindo mas não se limitando às seguintes métricas: *degree centrality*, *closeness centrality* e *betweenness centrality*. Esta última é de especial interesse para esta pesquisa. Ela é utili-

zada no processo de roteamento como alternativa para evitar grandes congestionamentos entre as rotas dos trens.

A métrica *betweenness* é proporcional à probabilidade de um nó ser intermediário na comunicação entre outros nós de uma rede. Trata-se de uma métrica importante para o controle da comunicação entre os nós da rede (FREEMAN, 1977). No contexto deste trabalho o uso dessa métrica contribui para que os caminhos relativos às rotas dos trens sejam melhor distribuídos pelo grafo, evitando a sobrecarga de alguns vértices ao utilizar algoritmos para a busca de menores caminhos entre dois vértices quaisquer.

Nesta pesquisa o cálculo dessa métrica foi realizado com base no algoritmo de Brandes (2001). Trata-se de um algoritmo bastante rápido que é capaz de trabalhar com grafos ponderados ou não ponderados, podendo calcular os valores da métrica tanto para vértices quanto para arestas. Nesta pesquisa a métrica *betweenness* foi calculada para as arestas dos grafos. Utilizando o grafo ponderado para o cálculo da métrica obtém-se a *betweenness ponderada*. Um exemplo de aplicação desta última pode ser visto no trabalho de Cousineau et al. (2015).

A Figura 5 mostra um exemplo hipotético onde é calculada a menor rota para um par de vértices *origem–destino*. Esse cálculo é feito em função dos pesos das arestas do grafo ponderado, Figura 5a, ou em função dos pesos das arestas relativos à métrica *betweenness ponderada*, Figura 5b. Em ambos os casos a rota desejada tem o vértice 1 como *origem* e o vértice 4 como *destino*.

Quando somente os pesos do grafo ponderado são levados em conta, Figura 5a, há uma tendência de que muitas rotas passem pelos mesmos vértices e arestas, gerando a sobrecarga de alguns caminhos. Isso ocorre porque o algoritmo KSP considera os pesos das arestas para encontrar os caminhos de menor custo. Quando a métrica *betweenness ponderada* é utilizada, Figura 5b, são calculados novos pesos para cada uma das arestas. Com isso, os caminhos que seriam naturalmente mais congestionados poderão deixar de ser. Essa alternativa contribui para evitar congestionamentos excessivos de alguns trechos.

Na Figura 5a é destacada a menor rota entre o par *origem-destino*. A soma dos pesos dessa rota é igual a 21. Por outro lado, Figura 5b, é mostrada a menor rota para o caso em que é utilizada a métrica *betweenness ponderada*. Nesse caso, a soma dos pesos é igual a 7. Desse modo, fica claro o fato de que o uso da métrica *betweenness ponderada* pode ajudar a distribuir melhor as rotas ao longo do grafo contribuindo para a redução de possíveis congestionamentos.

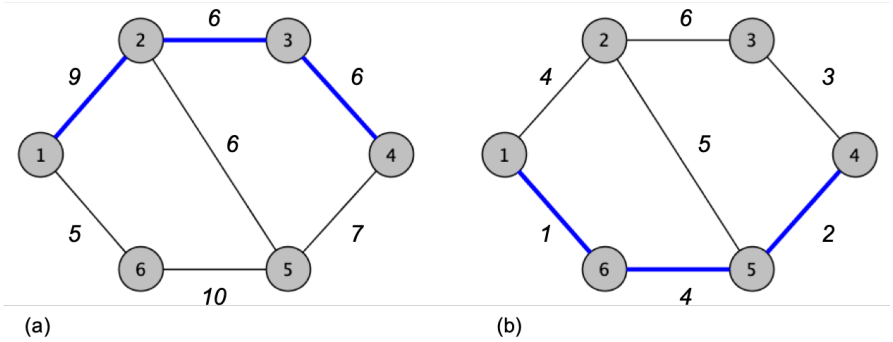

Figura 5 – Seleção de rotas por: (a) pesos e (b) *betweenness*.

Fonte: O autor

3.4 Verificação e Minimização de Conflitos

A informação de conflito de rota é levantada pela comparação de todos os trechos que compõem a rota de cada um dos trens. Esse procedimento é realizado em duas etapas distintas: (a) verificação de trechos iguais comparando todas as rotas e (b) verificação do intervalo de tempo em que cada trecho está programado para ser utilizado.

Os conflitos são contabilizados por trechos e por pares de trens, de modo similar ao que foi proposto por Ingolotti et al. (2005). Ele define uma série de restrições e uma em particular contribui para o tratamento dos conflitos. Quando uma ou mais restrições são violadas por dois trens que participam de um conflito, é atribuído um tempo atraso a um deles até que o conflito existente seja eliminado. A utilização de atrasos para resolver conflitos foi também utilizada por Şahin (1999), que estudou o planejamento de trens em tempo real, resolvendo os conflitos à medida em que eles ocorriam. Esse procedimento também foi abordado no trabalho de Costa, Paiva e Rocha (2018).

Os conflitos podem ser categorizados em função de características relativas ao modo como os trens trafegam na rede. Foram mapeadas seis diferentes condições de tráfego, conforme mostrado na Figura 6. Com base na situação identificada é possível determinar tanto o trem que acessa primeiro um dado trecho quanto seu sentido de deslocamento no trecho em questão. Essa informação possibilita classificar um dado trem como prioritário ou não prioritário, onde o prioritário é aquele que acessa primeiro o trecho em análise. A Figura 6 mostra também os pontos p_i, p_f, q_i e q_f. Eles correspondem aos vértices inicial e final dos trens T_1 e T_2, respectivamente.

É mostrada na Figura 6a uma situação em que os trens T_1 e T_2 trafegam no mesmo sentido, indo do vértice x_1 para o vértice x_2. O sinal $\cap$ mostra a interseção que ocorre entre suas rotas. A Figura 6b mostra uma situação similar, mas desta vez o trem T_2 se desloca do vértice x_2 para o vértice x_1. Na Figura 6c o trem T_2 se desloca de x_1 para x_2 em um intervalo de tempo completamente contido no intervalo em que T_1 trafega entre os mesmos

vértices. O mesmo comportamento pode ser observado na Figura 6d. No caso das Figuras 6e e 6f, de fato não ocorre qualquer conflito entre as rotas de T_1 e T_2. Esses cenários foram mostrados para complementar a compreensão relativa às situações de tráfego de trens que podem ser encontradas. Em todos os casos da Figura 6 é possível notar que o trem T_1 é prioritário em relação ao trem T_2. Isso ocorre porque em todos os casos apresentados T_1 inicia sua viagem no tempo t_1 e só após esse tempo T_2 inicia sua viagem.

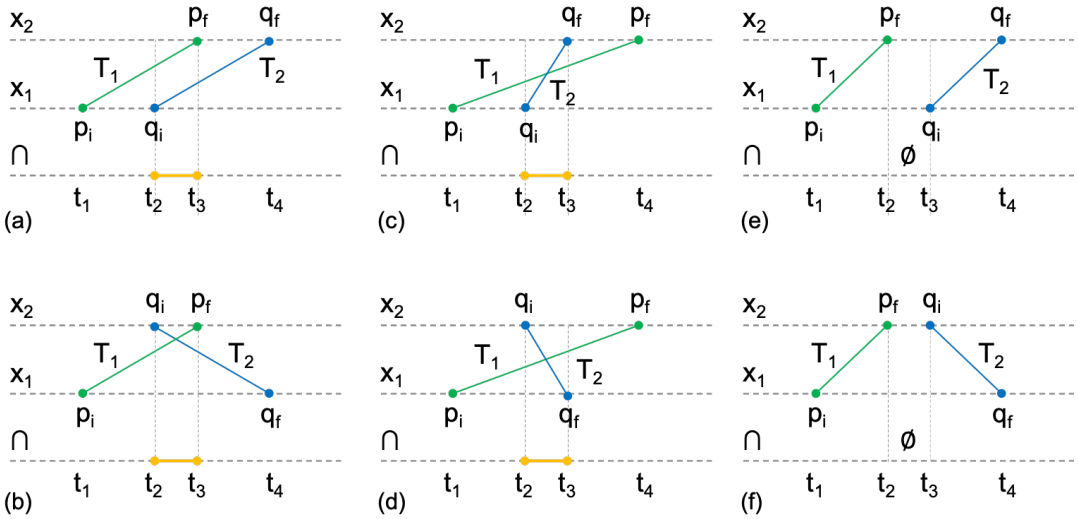

Figura 6 – Casos para detecção de conflito entre um par de trens T_1 e T_2.

Fonte: O autor

Quanto aos pontos p_i, p_f, q_i e q_f, as seguintes relações foram utilizadas como base para identificar o trem prioritário e o não prioritário.

$$dif_1 = p_i - q_i \tag{3.1}$$

$$dif_2 = p_i - q_f \tag{3.2}$$

$$dif_3 = p_f - q_i \tag{3.3}$$

$$dif_4 = p_f - q_f \tag{3.4}$$

Nas situações onde ocorrem conflitos a prioridade dos trens é definida segundo as Equações 3.1 a 3.4. Duas situações são mapeadas com base em seis grupos de inequações. Basta que um dos grupos seja atendido para que a situação de prioridade dos trens seja identificada. O símbolo $\wedge$ indica a operação E lógico entre as inequações, indicando que um grupo é atendido quando todas as suas inequações são satisfeitas. As *Situações 1* e *2* são apresentadas a seguir:

- **Situação 1**: T_1 prioritário e T_2 não prioritário:
  $Grupo\ 1 : (dif_1 \leq 0) \ \wedge \ (dif_2 \leq 0) \ \wedge \ (dif_3 \geq 0) \ \wedge \ (dif_4 \leq 0)$
  $Grupo\ 2 : (dif_1 \leq 0) \ \wedge \ (dif_2 \leq 0) \ \wedge \ (dif_3 \leq 0) \ \wedge \ (dif_4 \leq 0)$
  $Grupo\ 3 : (dif_1 \leq 0) \ \wedge \ (dif_2 \leq 0) \ \wedge \ (dif_3 \geq 0) \ \wedge \ (dif_4 \geq 0)$

- **Situação 2**: T_1 não prioritário e T_2 prioritário:
  $Grupo\ 4 : (dif_1 \geq 0) \ \wedge \ (dif_2 \leq 0) \ \wedge \ (dif_3 \geq 0) \ \wedge \ (dif_4 \geq 0)$
  $Grupo\ 5 : (dif_1 \geq 0) \ \wedge \ (dif_2 \geq 0) \ \wedge \ (dif_3 \geq 0) \ \wedge \ (dif_4 \geq 0)$
  $Grupo\ 6 : (dif_1 \geq 0) \ \wedge \ (dif_2 \leq 0) \ \wedge \ (dif_3 \geq 0) \ \wedge \ (dif_4 \leq 0)$

Uma vez que se sabe qual é o trem menos prioritário em um conflito, é possível definir o tempo de atraso que ele deve sofrer para permitir que o trem prioritário continue sua viagem. A Figura 7 apresenta uma situação em que são mostradas as rotas de três trens que trafegam em uma rede. Na Figura 7a são destacados dois conflitos que ocorrem entre esses trens. No trecho $x_3{-}x_4$ ocorre o cruzamento do trem T_3 com os trens T_1 e T_2. Nesse caso há dois possíveis conflitos já que o mesmo trecho está previsto para a rota dos três trens. Em seguida é verificado o conflito no tempo. O trem T_2 entra no trecho $x_3{-}x_4$ no instante de tempo t_2. Nesse mesmo instante o trem T_3 entra no trecho $x_4{-}x_3$. Uma análise similar pode ser feita em relação aos trens T_1 e T_3. Logo, pode-se concluir que os dois conflitos no espaço são também conflitos no tempo e por isso devem ser suprimidos. Escolhendo T_2 como prioritário em relação a T_3 faz com que este último sofra um atraso até que T_2 termine sua passagem pelo trecho em conflito. Todavia, ao eliminar esse conflito também é eliminado o conflito que existia entre T_1 e T_3. O atraso sofrido por T_3 faz com que seu horário de partida seja antecipado para o instante t_0. O planejamento final dos trens é mostrado na Figura 7b.

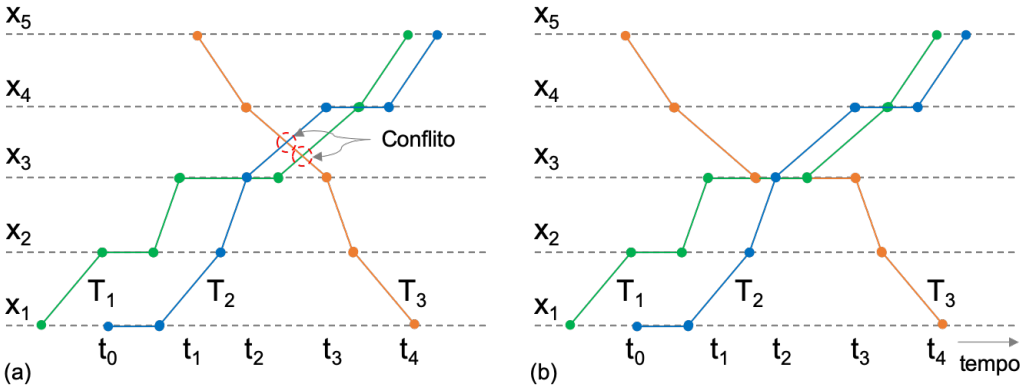

Figura 7 – Contagem de conflitos por pares de trens: (a) ocorrência de 2 conflitos envolvendo 3 trens e (b) eliminação dos conflitos existentes pela atribuição de tempo de atraso à rota do trem T_3.

Fonte: O autor

Na Figura 8 é mostrado outro caso de conflito entre trens, mas desta vez está em foco o tratamento da prioridade de passagem na ocorrência de conflitos entre trens. A Figura 8a mostra o momento em que os trens T_1 e T_2 trafegam pelo trecho $x_1{-}x_2$, ambos no mesmo sentido. Essa situação é a mesma daquela mostrada na Figura 6c. É possível notar que ocorre um conflito no trecho mencionado e que o trem T_1 chega primeiro a x_1, logo é dada a ele a prioridade de passagem pelo trecho em questão. Então, sabendo que T_1 tem prioridade de passagem, T_2 deve sofrer um atraso no trecho x_1, onde permanece parado pelo tempo necessário para que o conflito seja suprimido. A Figura 8b mostra a situação descrita, em que são atribuídos dois tempos de atraso a T_2. o primeiro ocorre no intervalo de tempo $t_2{-}t_3$ e o segundo, no intervalo $t_4{-}t_5$. Essa é apenas uma das diversas configurações possíveis para que se possa suprimir o conflito percebido na Figura 8a. Sempre que um conflito é eliminado ocorre uma nova verificação do número de conflitos para que o procedimento de supressão de conflitos possa ser repetido.

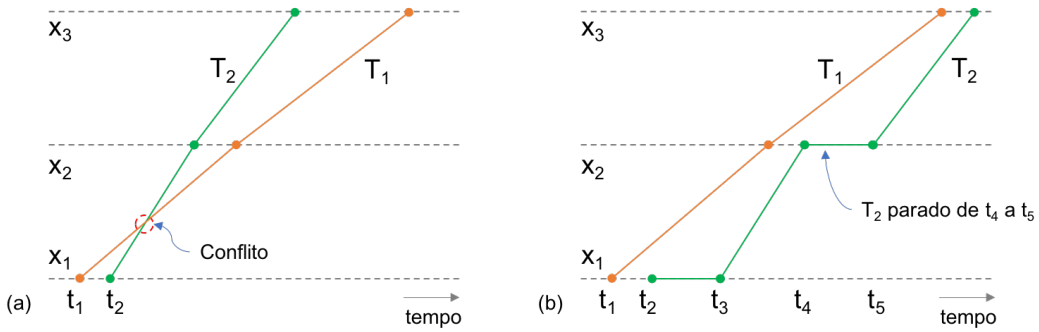

Figura 8 – Detecção e eliminação de conflito mediante a prioridade de passagem de trens: (a) ocorrência de conflito entre T_1 e T_2 e (b) eliminação do conflito pela inserção de atrasos na rota de T_2.

Fonte: O autor

Pode ser bastante complexa a tarefa de encontrar os conflitos entre as rotas e de definir os tempos de atraso que devem ser atribuídos a cada um dos trechos de rota de cada um dos trens. Por isso, nesta etapa foi utilizada a meta-heurística *Variable Neighborhood Search* (VNS), descrita no trabalho publicado por Hansen e Mladenović (1999). Para tratar algumas limitações do VNS, Hansen e Mladenović (2001) apresentaram um novo esquema denominado *Skewed VNS* (SVNS). Ele tem como objetivo evitar que o VNS encontre falsos ótimos globais. Tomando como base o SVNS, foi implementada nesta pesquisa uma versão *Multi-objective Skewed Variable Neighborhood Search* (MO-SVNS) para otimizar dois objetivos que se deseja minimizar: (a) número de conflitos entre trens (*Conf*) e (b) tempo médio de parada dos trens (*TMP*). Essas funções objetivo são descritas pelas

Equações 3.5 e 3.6, respectivamente.

$$Conf = \sum_{j=1}^{t} C_j \tag{3.5}$$

$$TMP = \frac{\sum_{i=1}^{\eta} TP_i}{\eta} \tag{3.6}$$

onde o índice i representa os η possíveis trens, o índice j representa os t possíveis trechos e C_j é o número de conflitos de rota ocorridos no trecho j.

Quanto menor o valor de TMP melhor é a solução em relação a esse objetivo. Em relação a *Conf*, a melhor solução é aquela que apresenta o menor número possível de conflitos ao final do processo de otimização.

Embora haja relação entre o tempo de viagem e o tempo de parada dos trens, este último foi escolhido como referência por dar ao controlador de tráfego o foco nos atrasos sofridos pelos trens. Conforme visto no Capítulo 1, uma parada pode representar um alto custo do ponto de vista da eficiência energética. Quanto ao número de conflitos, essa variável foi adotada como função objetivo porque nem sempre todos os conflitos serão eliminados pelo algoritmo de otimização. Caso isso ocorra, será necessário contar com a experiência dos controladores de tráfego para que os conflitos restantes sejam eliminados manualmente de modo que seja suprimida qualquer possibilidade de colisão entre trens.

Dado esse contexto, tem-se que a melhor solução é aquela em que ambos os objetivos são minimizados. Com base nessa premissa, ao longo da busca pela melhor solução são coletados os pontos que dão origem à Fronteira de Pareto. Para que os resultados obtidos possam ser facilmente analisados pelos operadores de tráfego, o gráfico que representa a Fronteira de Pareto relaciona o tempo médio de parada dos trens pelo número total de conflitos. Mais detalhes sobre a Fronteira de Pareto podem ser vistos no Capítulo 5, página 66.

3.5 Redução de Tempos de Viagem

A principal função do MO-SVNS, apresentado na Seção 3.4, é desembaralhar as rotas dos trens de modo que os conflitos sejam eliminados e que os tempos de paradas sejam minimizados. Entretanto, ao final do processo de busca pode ocorrer que não tenham sido encontrados os menores tempos de parada para cada um dos trens. Nesse caso, uma análise minuciosa pode revelar algumas oportunidades de eliminação de atrasos desnecessários.

A Figura 9 mostra duas situações onde alguns tempos de atraso poderiam ser eliminados. Para localizar e eliminar esses tempos foi desenvolvida uma heurística capaz de encontrar duas situações bem distintas classificadas como: (a) atraso *Tipo I* e (b) atraso

Tipo II. O *Tipo I* é aquele que quando removido para um determinado trem não gera nenhum conflito com outros trens que estejam trafegando na rede. O *Tipo II* é aquele onde todos os trens se encontram parados em um mesmo intervalo de tempo e por isso pode ser removida uma janela de tempo completa sem que qualquer conflito seja gerado. Na figura citada os tempos de parada que poderiam ser removidos foram marcados com linhas tracejadas, indicados por setas e o tipo foi colocado ao lado da seta. Os tempos de parada são representados pelas linhas paralelas ao eixo do tempo.

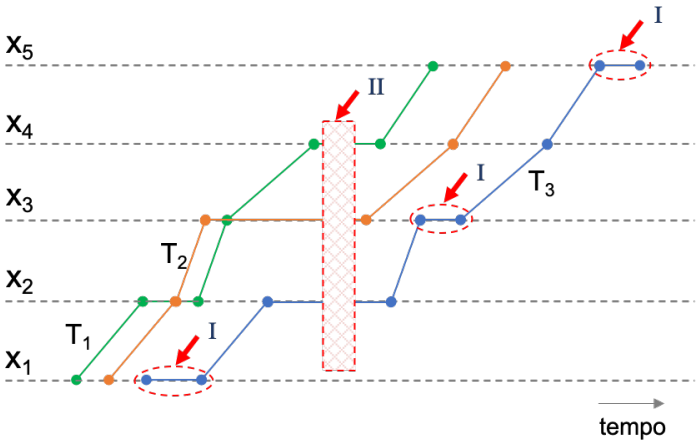

Figura 9 – Paradas indesejadas. *Tipo I*: tempos de parada analisados para cada trem individualmente e *Tipo II*: janela de tempo em que todos os trens encontram-se parados.

Fonte: O autor

3.6 Tráfego de Trens

O entendimento acerca do tráfego de trens é de fundamental importância para a compreensão do procedimento de verificação e minimização de conflitos explicado na Seção 3.4. Como essa pesquisa leva em conta o tráfego de trens em redes aleatórias, muitas das regras de tráfego empregadas em ferrovias reais deixam de ser aplicáveis. A utilização do método desenvolvido nessa pesquisa para ferrovias reais exigiria que algumas particularidades fossem implementadas.

Para a análise das regras utilizadas foi proposto um cenário hipotético em que são mostradas as rotas de seis trens. Esse cenário é mostrado na Figura 10. Nessa figura os trens são representados pelos rótulos enumerados de T_1 a T_6. As variáveis x_1 a x_5 indicam os trechos da ferrovia, isto é, os vértices do grafo que representa a rede por onde trafegam os trens. Logo, se um trem T trafega do vértice x_1 para x_2, o vértice x_1 permanece ocupado até que T atinja o vértice x_2. Nesse momento o vértice x_1 é liberado para outro trem e o vértice x_2 é ocupado. Quando é atribuído um tempo de atraso a um trem T em um trecho

x de sua rota, ele permanece em repouso em x para que outro trem possa trafegar por x. Mesmo que em um primeiro momento a interpretação seja de que o trem em repouso e o trem em movimento colidiriam por utilizarem o mesmo trecho x, a interpretação física é de que o trem que ficaria em repouso utilizaria um trecho paralelo x'. Para simplificar a modelagem da ferrovia pela teoria dos grafos o trecho x' não é representado. Entretanto, a comparação entre o modelo teórico e a ferrovia real é fundamental para o entendimento das premissas utilizadas na construção do modelo em grafo.

Na Figura 10 são destacados quatro eventos que ocorrem ao passo em que os trens trafegam na rede. É possível observar que ocorrem eventos similares que não são discutidos. É importante notar que neste momento não está em análise a qualidade do roteamento já que o exemplo tem como foco permitir a compreensão do processo de planejamento do tráfego de trens.

- **Ponto a:** Assume-se que T_1 e T_2 trafegam no mesmo sentido, $x_1 \rightarrow x_5$, utilizando a mesma linha férrea. Em x_2 o trem T_1 efetua uma parada para que T_2 prossiga sua viagem. Assim que T_2 chega a x_3 o trem T_1 continua sua viagem conforme programado;
- **Ponto b:** Assume-se que T_3 e T_5 trafegam na mesma linha, porém em sentidos opostos. T_3 segue no sentido $x_1 \rightarrow x_5$ enquanto T_5 segue no sentido $x_5 \rightarrow x_1$. Nesse caso, o ponto de cruzamento em destaque, trecho x_2-x_3, representa um conflito de rota e por isso não pode ser executado;
- **Ponto c:** Assume-se que T_4 e T_6 trafegam em linhas diferentes e que T_4 segue no sentido $x_1 \rightarrow x_5$ e T_6 segue no sentido $x_5 \rightarrow x_1$. Nesse caso, por circularem em linhas diferentes, o ponto de cruzamento em destaque, trecho x_3-x_4, não representa um conflito;
- **Ponto d:** O ponto em destaque mostra que embora o trem T_4 pudesse iniciar sua rota, ele permanece em repouso por um dado intervalo de tempo até que sua rota seja finalmente iniciada.

3.7 Implementação do Método Proposto

As seguintes etapas foram executadas para que o método proposto por esta pesquisa pudesse ser implementado com sucesso. Essas etapas compreendem desde a avaliação de técnicas de planejamento e roteamento à avaliação da qualidade da solução obtida. As etapas são:

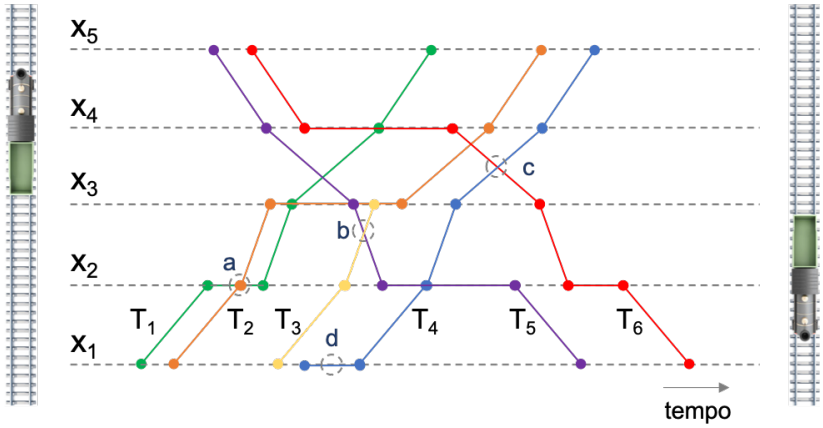

Figura 10 – Exemplo de tráfego de trens.

Fonte: O autor

- Avaliar técnicas de planejamento e roteamento aplicáveis a problemas de telecomunicações tendo em vista que assuntos dessa natureza são amplamente discutidos para tratar questões relacionadas ao tráfego de dados;
- Avaliar modelos de ferrovia segundo a teoria dos grafos;
- Avaliar técnicas de detecção e solução de conflitos de rota entre entidades que trafeguem em uma rede;
- Utilizar a métrica *betweenness* tanto com grafos ponderados quanto com grafos com pesos unitários em seus vértices para avaliar sua influência no processo de otimização;
- Definir regras para que os trens possam trafegar em redes aleatórias planares;
- Estudar o VNS e seus esquemas para definir as características que devem ser consideradas no algoritmo MO-SVNS;
- Implementar algoritmo para otimizar tanto o número de conflitos entre trens quanto o tempo de parada dos trens em circulação, ambos os objetivos que se deseja minimizar para resolver o problema de planejamento e roteamento de trens;
- Implementar algoritmo capaz de reduzir os tempos de parada dos trens evitando atrasos desnecessários;
- Construir a Fronteira de Pareto para avaliar as soluções ótimas segundo cada um dos dois objetivos que devem ser otimizados pelo MO-SVNS;
- Implementar métricas capazes de mensurar a qualidade da Fronteira de Pareto;

- Propor métrica para auxiliar na análise da Fronteira de Pareto para um conjunto de execuções realizadas em instâncias que não tenham correspondência direta na literatura;
- Estressar o algoritmo MO-SVNS iniciando a busca pelo pior cenário possível, onde os trens iniciam sua viagem sem qualquer tempo de atraso.

4 Implementação do Método Proposto

Neste capítulo são apresentados os algoritmos que compõem a implementação do método de otimização proposto para solucionar o problema de planejamento e roteamento de trens em redes modeladas via grafos. São apresentados os algoritmos básicos do VNS e as adaptações feitas em suas funções para a obtenção da meta-heurística MO-SVNS utilizada nesta pesquisa. Em seguida são explicadas as estruturas de vizinhança e o algoritmo desenvolvido com a finalidade de reduzir os tempos de viagem dos trens.

4.1 *Variable Neighborhood Search*

Muitas meta-heurísticas são baseadas em heurísticas de busca local para encontrar soluções ótimas. O VNS, em específico, explora vizinhanças gradativamente mais distantes da solução atual. Essa meta-heurística 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; MLADENOVIĆ, 2001). Trata-se de um esquema de otimização baseado em trajetória. Ele é bastante robusto e se adapta muito bem a outras meta-heurísticas na formação de modelos híbridos, seja utilizando o VNS como heurística principal ou mesmo como heurística de busca local. Dentre suas principais características, destacam-se: (a) mudança sistemática de vizinhança, (b) busca local possivelmente aleatória e (c) exploração de vizinhanças gradativamente maiores.

O VNS explora sistematicamente três fatos, a saber:

- Um mínimo local em uma estrutura de vizinhança não é necessariamente um mínimo local para outra;
- Um mínimo global é um mínimo local para todas as estruturas de vizinhança possíveis;
- Para muitos problemas, ótimos locais para uma ou mais estruturas de vizinhança são relativamente próximos. Esse fato é empírico e pressupõe que um ótimo local sempre traz em si muitas informações acerca do ótimo global.

O VNS pode ser utilizado com muitos esquemas diferentes tais como *Reduced VNS* (RVNS), *Basic VNS* (BVNS), *General VNS* (GVNS), *Skewed VNS* (SVNS), entre outros, conforme descrito por Hansen e Mladenovic (2007). A seguir serão descritos os esquemas BVNS e SVNS por serem aqueles de maior relevância para esta pesquisa.

4.1.1 *Basic VNS*

O *Basic VNS* (BVNS) utiliza os elementos básicos do VNS na busca das soluções ótimas. Estão incluídas as etapas onde a solução corrente é perturbada, depois é realizada uma etapa de busca local e então a troca da vizinhança caso a vizinhança atual não seja mais promissora.

De acordo com Hansen e Mladenovic (2007), a busca local é uma heurística que consiste em escolher uma solução inicial x, encontrar uma direção de descida a partir de x, dentro de uma vizinhança $N(x)$, e mover para o mínimo da função $f(x)$ dentro de $N(x)$ na direção de descida. Se nenhuma direção de descida for encontrada então o algoritmo de busca local é finalizado. Caso contrário, a busca é iterada para que o processo continue. A busca local pode ser do tipo *First Improvement* ou *Best Improvement*. No primeiro tipo o algoritmo de busca local é interrompido assim que é encontrada a direção de descida. Por outro lado, no segundo tipo, todas as soluções vizinhas da solução x são avaliadas e então é escolhida a melhor entre elas, isto é, a solução ótima local.

No esquema BVNS frequentemente é utilizada a busca local do tipo *First Improvement*, embora Hansen e Mladenovic (2007) afirmem que também poderia ser utilizada a busca local do tipo *Best Improvement*. Neste trabalho foi utilizado um algoritmo de busca local do tipo *First Improvement*, que executa um movimento assim que é encontrada a direção de descida em busca do ótimo global. Um movimento é uma transformação que dá origem a uma nova solução. Os movimentos são obtidos a partir das estruturas de vizinhança utilizadas pelo esquema de VNS escolhido. As estruturas de vizinhança utilizadas nesta pesquisa serão apresentadas na Seção 4.3.

A Figura 11 exemplifica como são utilizadas as estruturas de vizinhança no BVNS. São combinados métodos estocásticos e determinísticos para a mudança de vizinhança. Além disso, é bastante frequente a utilização de vizinhanças aninhadas quando se utiliza o esquema BVNS. De fato, esse comportamento também é observado em outros esquemas. Ainda nessa figura, $N_k(x)$ se refere ao conjunto de soluções na *k-ésima* estrutura de vizinhança da solução x, onde $k = 1, 2, ..., k_{max}$.

O Algoritmo 1 descreve o processo do BVNS com algumas pequenas adaptações para o contexto desta pesquisa. Ele recebe como entrada a programação dos trens ($progTrens$), a maior estrutura de vizinhança que deve ser utilizada (k_{max}) e o tempo máximo de execução do algoritmo (t_{max}).

A função *SolucaoInicial* gera uma solução x a partir da programação inicial dos trens. Nessa solução não é atribuído qualquer tempo de parada aos trens, fato que gera a melhor solução possível em relação ao tempo de viagem dos trens, embora possam existir muitos conflitos entre suas rotas. Uma solução é definida como sendo a programação de todos os trens, com ou sem conflitos entre suas rotas.

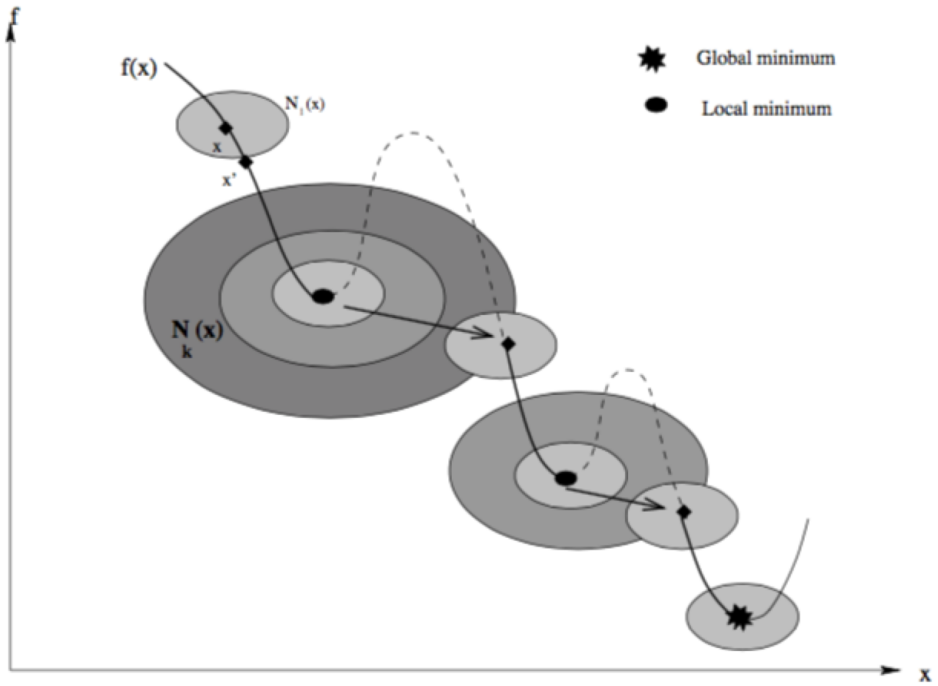

Figura 11 – Busca de ótimos locais em diferentes estruturas de vizinhança.

Fonte: Hansen e Mladenovic (2007)

Algoritmo 1: *BasicVNS() | Basic VNS*

Entrada: $progTrens, k_{max}, t_{max}$
Saída: Melhor solução encontrada

```
1  x ← SolucaoInicial(progTrens);
2  repita
3      k ← 1;
4      repita
5          x' ← PerturbarSolucao(x, k);
6          x" ← BuscaLocal(x');
7          x ← MudarVizinhanca(x, x", k);
8          t ← TempoProcessamento();
9      até k = k_max;
10 até t > t_max;
11 retorna x
```

No laço principal a função *PerturbarSolucao* promove movimentos aleatórios na solução x dentro da vizinhança k de tal modo que seja gerada uma nova solução x'. Um movimento é caracterizado por transformações que acarretam em diferentes soluções na vizinhança da solução corrente. Nesta pesquisa os movimentos são relacionados à atribuição de tempos de atraso em um ou mais trechos que compõem a rota de um dado trem. Esses movimentos se dão em função das estruturas de vizinhança, que serão vistas na Seção 4.3.

Em seguida é executada a função *BuscaLocal*, Algoritmo 2, que tem como entrada a solução x'. Essa função gera uma nova solução x" na vizinhança da solução recebida como entrada. A busca local utilizada é do tipo *First Improvement*. Nesse método sistematicamente são enumerados vetores $x^i \epsilon N(x)$ e um movimento é realizado assim que a direção de decida é identificada. Nesse movimento a solução corrente recebe a nova solução encontrada. De acordo com Gendreau, Potvin et al. (2010), o método *First Improvement* é útil em problemas mais complexos visto que a busca local do tipo *Best Improvement* pode consumir muito tempo para encontrar um ótimo local.

Algoritmo 2: *BuscaLocal()* | Busca Local tipo *First Improvement*

Entrada: x
Saída: *Solução First Improvement encontrada*

```
1  repita
2      x' ← x;
3      i ← 0;
4      repita
5          i ← i + 1;
6          x ← arg min{f(x), f(x^i)}, x^i ϵ N(x);
7      até (f(x) < f(x') ou i = |N(x)|);
8      x ← arg min_{x ϵ N(x)} f(x);
9  até f(x) ≥ f(x');
10 retorna x
```

Com base na estrutura de vizinhança atual k e nas soluções x e x" é executada a função $MudarVizinhanca$, Algoritmo 3. Essa função avalia se a função x" é melhor do que a solução corrente x. Caso seja, a solução x" é atribuída à solução corrente e então voltam a ser exploradas as soluções na primeira estrutura de vizinhança. Caso contrário, a estrutura de vizinhança é trocada para que novas soluções em vizinhanças mais distantes da solução corrente possam ser avaliadas.

O término do Algoritmo 1 se dá ao atingir o tempo de processamento predefinido. A condição de saída é avaliada pela função *TempoProcessamento*. Esse tempo, na verdade, pode ser definido segundo uma combinação de parâmetros tais como tempo de execução, número de iterações, tempo após a última melhoria, etc. De fato, neste trabalho foi definida uma combinação de critérios de parada.

Algoritmo 3: *MudarVizinhanca()* | Mudar estrutura de vizinhança para BVNS

Entrada: x, x', k
Saída: *Solução corrente e estrutura de vizinhança a ser explorada*

```
1 se f(x') < f(x) então
2 |  x ← x';
3 |  k ← 1;
4 senão
5 |  k ← k + 1;
6 fim
7 retorna x, k
```

4.1.2 *Skewed VNS*

Como o VNS explora vizinhanças da solução corrente, se a solução ótima estiver em um pico ou vale sendo explorado, ela poderá ser rapidamente encontrada pelo VNS. Contudo, pode ocorrer que a solução ótima esteja em um vale ainda mais baixo, mas em uma região muito distante da vizinhança sendo explorada. Ao tentar fazer saltos muito distantes pode ocorrer a degeneração do VNS fazendo com que ele passe a se comportar de modo semelhante à meta-heurística *multistart*, que não é tão eficiente. Nesse caso, pode ser que a solução ótima jamais seja encontrada. Há alguns modos de superar esse problema, mas Hansen e Mladenović (2001) propõem que simplesmente seja adicionado um fator α que permita a aceitação de resultados piores ao avaliar a função objetivo. Assim, mesmo que possam ser encontradas soluções piores do que aquelas já conhecidas até o momento, logo em seguida outras melhores poderão ser descobertas até que o ótimo global possa ser encontrado. Esse esquema foi chamado de *Skewed VNS* (SVNS).

O parâmetro α está relacionado à distância entre a solução corrente x e a solução x" gerada após a busca local. Esse fator deve ser escolhido de modo experimental, podendo variar em função da distância mencionada ou em função de algum método de aprendizagem.

Alguns testes realizados por Hansen e Mladenović (2001) tinham como objetivo resolver 25 problemas com o SVNS e também com o TS. Em 23 problemas a solução encontrada pelo SVNS foi a melhor conhecida. Quando os mesmos problemas foram resolvidos utilizando a meta-heurística TS, somente em 5 dos casos foram encontradas as melhores soluções conhecidas.

O Algoritmo 4 mostra os passos básicos para o SVNS. Este algoritmo difere do BVNS pela função *MudarVizinhanca*, em que o parâmetro α também é passado para compor o cálculo com a função ρ. Esta última função tem a finalidade de mensurar a distância entre a solução corrente x e a solução ótima local x". Além de α, são recebidos como entrada da função a programação dos trens *progTrens*, a maior vizinhança k_{max} para ser utilizada e o tempo máximo de execução t_{max}.

No Algoritmo 5 é mostrado como se dá a mudança de vizinhança para o SVNS.

Algoritmo 4: *SkewedVNS() | Skewed VNS*

Entrada: $progTrens, k_{max}, t_{max}, \alpha$
Saída: *Melhor solução encontrada*
$x \leftarrow SolucaoInicial(progTrens)$;
repita
  $k \leftarrow 1$;
  repita
    $x' \leftarrow PerturbarSolucao(x, k)$;
    $x" \leftarrow BuscaLocal(x')$;
    $[x, k] \leftarrow MudarVizinhanca(x, x", k, \alpha)$;
    $t \leftarrow TempoProcessamento()$;
  até $k = k_{max}$;
até $t > t_{max}$;
retorna x

Esse algoritmo recebe como entrada a solução corrente x, a melhor solução x" obtida em sua vizinhança, a vizinhança atual k e o fator α. A diferença marcante em relação ao algoritmo utilizado pelo BVNS está no termo adicional $\alpha\rho(x, x")$. Essa desigualdade define a condição para aceitar a solução x" como sendo a nova solução corrente.

Algoritmo 5: *MudarVizinhanca()* | Mudar estrutura de vizinhança para SVNS

Entrada: $x, x", k, \alpha$
Saída: *Solução corrente e estrutura de vizinhança a ser explorada*
se $f(x") - \alpha\rho(x, x") < f(x)$ **então**
  $x \leftarrow x"$;
  $k \leftarrow 1$;
senão
  $k \leftarrow k + 1$;
fim
retorna x, k

4.2 Algoritmos Adaptados

Após vistas as funções básicas que compõem os algoritmos do BVNS e do SVNS, em seguida serão apresentadas as funções adaptadas à realidade desta pesquisa. Logo, os algoritmos trarão mais detalhes daqui para frente.

4.2.1 Algoritmo do *Multi-objective Skewed VNS*

Como o foco desta pesquisa está na solução de um problema multiobjetivo, foi desenvolvido um algoritmo *Multi-objective Skewed Variable Neighborhood Search* (MO-SVNS) baseado no trabalho de Rubio-Largo et al. (2010). Em adição ao SVNS, nesse esquema além de ser feita a otimização das duas funções objetivo que se deseja minimizar

neste trabalho, também é realizada a coleta dos pontos que dão origem à Fronteira de Pareto, que será vista na Seção 5.1. Na Seção 5.2 serão discutidos alguns indicadores utilizados para avaliar a qualidade da Fronteira de Pareto.

O Algoritmo 6 mostra o procedimento do MO-SVNS. Essa função tem como entrada a programação inicial dos trens (*progTrens*), o maior valor de estrutura de vizinhança que pode ser assumido (k_{max}), o tempo máximo que o algoritmo terá para encontrar uma solução (t_{max}) e o valor de α, que funciona como uma tolerância para aceitar soluções piores do que a solução corrente.

Algoritmo 6: *MoSVNS() | Multi-objective Skewed VNS*

Entrada: $progTrens, k_{max}, t_{max}, \alpha$
Saída: *Melhor solução encontrada*

```
1  x ← SolucaoInicial(progTrens);
2  conflitos ← VerificarConflitos(x);
3  tempoMedioParada ← CalcularTempoMedioParada(x);
4  pareto ← (conflitos, tempoMedioParada);
5  AjustarParametros();
6  controle = 0;
7  repita
8      k ← 1;
9      repita
10         x' ← PerturbarSolucao(x, k);
11         x" ← BuscaLocal(x', n);
12         [x, k] ← MudarVizinhanca(x, x", k, α);
13         se x for melhorada então
14             x ← ReduzirTempos(x);
15             conflitos ← VerificarConflitos(x);
16             tempoMedioParada ← CalcularTempoMedioParada(x);
17             pareto ← (conflitos, tempoMedioParada);
18         fim
19         t ← TempoProcessamento();
20         se t > t_max então
21             controle = 1;
22         fim
23     até (k = k_max) OU (controle = 1);
24 até controle = 1;
25 retorna x
```

No início do procedimento é calculada uma solução inicial x com base na programação dos trens. A obtenção da solução inicial será vista na Seção 4.2.2. Em seguida, baseado na solução x, é realizada uma busca para identificar todos os conflitos entre as rotas dos trens. Os conflitos são mapeados para cada par de trens e pode ocorrer mais de um conflito por par. O algoritmo de verificação de conflitos será visto na Seção 4.2.3. No passo seguinte é calculado o tempo médio de parada dos trens. A função *CalcularTempoMedioParada*

soma todos os tempos de atraso dos trens em circulação e então calcula o valor médio. Tanto o número de conflitos quanto o tempo médio de parada são armazenados na variável que contém o repositório de Pareto (*pareto*). Essas são as duas variáveis de decisão que devem ser otimizadas ao longo do procedimento executado pelo MO-SVNS. Na sequência é invocada a função *AjustarParametros*, que carrega todos os parâmetros de definição do MO-SVNS. Esses parâmetros são listados a seguir:

- α: Ajustado em 25%, com base em experimentos. Esse fator permite aceitar soluções piores do que a solução corrente;
- **maxZeroConf**: Ajustado em 2. Caso ocorram duas iterações seguidas em que o número de conflitos permaneça em zero o algoritmo MO-SVNS é finalizado;
- **tempoEntreMelhorias**: Ajustado em 60 segundos. Se o tempo médio entre as melhorias for superior ao tempo definido, o algoritmo MO-SVNS é finalizado;
- **maxIter**: Ajustado em T x n, onde T é o número de trens em circulação e n é o número de vértices.

Uma vez realizados os ajustes iniciais, o laço principal do Algoritmo 6 é iniciado. A busca começa explorando a primeira estrutura de vizinhança. A partir da solução corrente x e da estrutura de vizinhança k é realizada uma perturbação gerando uma nova solução x' na vizinhança de x. Essa nova solução é enviada para a função *BuscaLocal* que será vista em detalhes na Seção 4.2.4. A saída dessa função gera a solução x". Com base em x, x", k e α é verificado pela função *MudarVizinhanca*, Algoritmo 3, se a vizinhança atual continuará sendo explorada ou se serão procuradas soluções em outras estruturas de vizinhança. Essa função tem como saída a nova solução corrente e a estrutura de vizinhança que deverá ser explorada. Caso a solução corrente tenha melhorado, é aplicada a ela a função *ReduzirTempos*, que recebe como entrada a solução corrente x. Essa função será vista na Seção 4.4. Em seguida os conflitos são avaliados, o tempo médio de parada é calculado e ambos os dados são armazenados no repositório de Pareto. O tempo de processamento é atualizado e o processo de busca continua até que a condição de parada seja atingida.

O tempo de processamento t é calculado pela função *TempoProcessamento*. Se uma das condições a seguir for atingida o algoritmo MO-SVNS é finalizado. As condições são:

- **Iterações**: Caso o número de iterações atinja o máximo definido por *maxIter*;
- **Zero conflitos**: Caso seja atingido o número máximo de zero conflitos, definido por *maxZeroConf*;

- **Conflitos**: Caso o número de conflitos seja igual a zero. Contudo, esse parâmetro só é considerado após um número mínimo de iterações. Esse limite é imposto já que o tempo médio de parada também deve ser minimizado. É adotada uma taxa de 105% em relação à melhor iteração registrada até o momento. O valor da melhor iteração é atualizado sempre que a solução corrente é melhorada. Outro limite é o número mínimo de iterações que devem ser executadas, estipulado em 1% do número máximo de iterações;

- **Tempo entre melhorias**: Caso o tempo entre melhorias seja superior ao dobro do tempo definido por *tempoEntreMelhorias*. Esse tempo é calculado pela média entre o valor corrente de *tempoEntreMelhorias* e o tempo decorrido desde a última melhoria da solução corrente.

4.2.2 Algoritmo da Solução Inicial

O Algoritmo 7 descreve o procedimento de construção da solução inicial. Ele recebe como entrada a matriz de adjacências (*matrizAdj*), a programação dos trens (*progTrens*), o número de rotas que devem ser calculadas (K), e os parâmetros básicos relativos a cada um dos trens (*paramRota*). Quanto ao número de rotas calculadas, nos experimentos realizados nesta pesquisa podem ser escolhidas uma ou três rotas por trem, dependendo do experimento a ser realizado.

A matriz de adjacências, segundo Netto (2012), tem ordem n assim como o grafo G representado por ela. Cada linha e coluna da matriz é associada a um vértice. Os dados da matriz recebem valores nulos quando não há uma ligação e valores não nulos quando há uma ligação entre dois vértices quaisquer do grafo. O valor em questão se refere ao peso de cada uma das arestas. O peso de cada aresta pode ser igual a 1 para grafos unitários ou igual a qualquer outro valor para grafos ponderados.

Inicialmente são calculadas e armazenadas as rotas de todos os trens. Em seguida, para cada uma das rotas armazenadas são calculados os parâmetros adicionais que compreendem todas as informações relativas aos tempos de entrada e saída dos trens em cada um dos trechos que formam sua rota. Após essa etapa, com base nas rotas calculadas são verificados todos os conflitos existentes entre os trens. Com esses dados o número de conflitos ocorridos entre as rotas de cada um dos trens é comparado e, então, dentre as rotas disponíveis é escolhida aquela que provoca o menor número de conflitos no contexto das rotas já selecionadas. É escolhida somente uma rota para cada trem. Quando a escolha da rota de todos os trens é realizada o procedimento é finalizado.

A função *RotasKSP* utiliza o algoritmo de *Yen* para o cálculo dos K menores caminhos. Esse algoritmo, por sua vez, utiliza o algoritmo de *Dijkstra* para o cálculo do menor caminho entre dois vértices quaisquer de uma rede. O menor caminho é definido como

Algoritmo 7: *SolucaoInicial()* | Gerar solução inicial

Entrada: $matrizAdj, progTrens, K, paramRota$
Saída: *Solução inicial com a programação dos trens*

```
1  para cada progTrens faça
2  |  origem ← progTrens.origem;
3  |  destino ← progTrens.destino;
4  |  progTrens.rotaCalculada ← RotasKSP(matrizAdj, origem, destino, K);
5  fim
6  para cada progTrens.rotaCalculada faça
7  |  progTrens.parametrosAdicionais ← CalcParam(progTrens, paramRota);
8  fim
9  para cada progTrens faça
10 |  para m ← 1 até K faça
11 |  |  progTrens.rota(m) ← progTrens.rotaCalculada(m);
12 |  |  progTrens.rota(m).conflitos ← VerificarConflitos(progTrens);
13 |  fim
14 |  para m ← 1 até K − 1 faça
15 |  |  se progTrens.rota(m).conflitos < progTrens.rota(m + 1).conflitos então
16 |  |  |  progTrens.melhorRota ← progTrens.rota(m);
17 |  |  fim
18 |  fim
19 fim
20 x ← progTrens;
21 retorna x
```

sendo aquele que apresenta a menor soma de pesos. Nesta pesquisa os pesos correspondem ao comprimento dos trechos e são atribuídos aos vértices. Outra função utilizada é a *CalcParam*. Nessa função são atualizados os campos *Tempos do processo* e *Relação tempo processo*, apresentados na Figura 12. Nessa figura é mostrada a estrutura de dados calculada para cada um dos trens. Os campos apresentados são descritos a seguir:

- **Rótulo do trem**: Os trens são identificados pelo campo *nome* e são representados por números inteiros positivos;
- **Vértices origem e destino**: Trata-se dos vértices de origem e destino do trem na rede por onde ele circulará;
- **Início desejado**: A hora de início desejado apresenta o horário em que o trem deverá partir do vértice de origem;
- **Rota do trem**: Na rota são apresentados todos os vértices por onde o trem deverá passar desde a origem até o destino final;
- **Parâmetros adicionais**: Para cada um dos vértices da rota são armazenados os dados de distância (km), velocidade (km/h) e tempo de travessia (h);

- **Tempos do processo**: Nesse campo é apresentado o tempo mínimo da rota, o tempo total de viagem, o tempo total parado, o horário de entrada e saída de cada vértice e o tempo de parada ou atraso em cada um dos vértices. No exemplo mostrado não há qualquer atraso atribuído a nenhum dos trechos da rota do trem em questão. Os dados de tempo são representados por números decimais, onde a parte inteira é referente à hora do dia e a parte decimal é referente aos minutos;
- **Relação tempo processo**: Esse campo apresenta a relação entre o tempo total e o tempo mínimo de viagem.

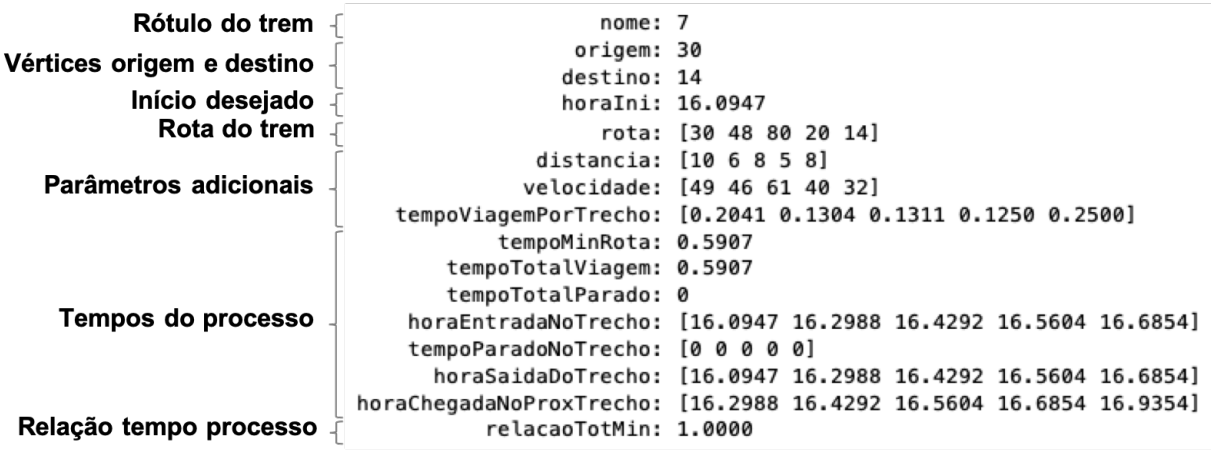

Figura 12 – Estrutura de dados calculada para cada trem.

Fonte: O autor

4.2.3 Algoritmo da Verificação e Minimização de Conflitos

A função *VerificarConflitos* é descrita no Algoritmo 8. Ela recebe como entrada a programação dos trens. Cada par de trens tem seus trechos comparados e quando são encontrados trechos em comum são armazenados alguns dados de referência. São armazenados tanto o nome dos trens quanto os intervalos de tempo em que cada um deles inicia e termina sua travessia pelo trecho em questão. Esses dados são submetidos à função *ConflitoTempo*, que avalia a ocorrência de conflito no tempo e identifica qual dos dois é o prioritário para seguir viagem sem sofrer atrasos. Essa função lança mão dos conceitos explicados na Seção 3.4. Todos os conflitos encontrados para cada par de trens são armazenados na matriz *conflitos*, que é a variável retornada pela função.

A Figura 13 mostra a estrutura de dados utilizada para armazenar os dados de cada um dos conflitos. Os campos de dados são explicados a seguir:

- **Vértice do conflito**: Esse campo traz o número do vértice do grafo onde ocorre o conflito;

Algoritmo 8: *VerificarConflitos()* | Verificar onde ocorrem conflitos de rota

Entrada: $progTrens$
Saída: *Matriz com o mapeamento dos conflitos encontrados*

```
1  numTrens ← progTrens.numTrens;
2  para i ← 1 até numTrens faça
3      k ← i + 1;
4      para j ← k até numTrens faça
5          c ← 0;
6          para cada progTrens(i).trecho faça
7              para cada progTrens(j).trecho faça
8                  se progTrens(i).trecho = progTrens(j).trecho então
9                      c ← c + 1;
10                     T1 ← progTrens(i);
11                     T2 ← progTrens(j);
12                     confEspaco(c).T1 ← T1.nome;
13                     confEspaco(c).T2 ← T2.nome;
14                     confEspaco(c).dt1 ← [T1.tempIni, T1.tempFim];
15                     confEspaco(c).dt2 ← [T2.tempIni, T2.tempFim];
16                 fim
17             fim
18         fim
19         para cada confEspaco faça
20             T1 ← confEspaco.T1;
21             T2 ← confEspaco.T2;
22             dt1 ← confEspaco.dt1;
23             dt2 ← confEspaco.dt2;
24             confTempo ← ConflitoTempo(T1, dt1, T2, dt2);
25         fim
26         conf ← conf + confTempo;
27     fim
28 fim
29 conflitos ← conf;
30 retorna conflitos
```

- **Trens**: Nesse campo é identificado o par de trens envolvido no conflito bem como a classificação dos dois como prioritário ou menos prioritário;
- **Tempo de conflito**: Aqui é calculado o tempo de conflito em relação aos intervalos de tempo em que cada um dos trens utiliza o trecho em conflito. Esse tempo serve de base para o cálculo do menor tempo de atraso necessário que deve ser atribuído ao trem menos prioritário para que o conflito possa ser eliminado;
- **Tempo no vértice**: Campo que traz o intervalo de tempo exato em que cada trem inicia e termina sua viagem ao longo do trecho em conflito.

Figura 13 – Estrutura de dados calculada para cada conflito encontrado.

Fonte: O autor

4.2.4 Algoritmo da Busca Local

O procedimento de busca local tem papel fundamental na redução dos conflitos. A cada iteração do algoritmo MO-SVNS a busca local é executada à procura da primeira melhoria que indique o caminho de descida em direção à solução ótima para o problema. Portanto, o algoritmo utilizado no procedimento de busca local é do tipo *First Improvement*. No Algoritmo 9 são mostrados os passos executados. Esse algoritmo recebe como entrada a solução corrente x e o número de vértices n do grafo.

Primeiramente é definido o número máximo de iterações que deverão ser realizadas ($iter_{max}$). Esse número é dado pela relação $n/10$, onde n é o número de vértices do grafo utilizado. Em seguida são levantados todos os conflitos da solução dada como entrada para a função *BuscaLocal*. Com esses dados calculados, um conflito é escolhido ao acaso. A partir do conflito escolhido os dados do trem menos prioritário são armazenados na variável *tremMP*. Em seguida é executada a função *PosicaoParada*, que tem como entrada a rota do trem escolhido. Essa função avalia os dados de conflito e escolhe o trecho mais adequado para que o trem menos prioritário efetue uma parada. Essa escolha é feita de tal modo que sejam evitadas grandes perturbações na programação corrente dos trens.

A função *DefinirParametros* tem como entrada a solução corrente. Nessa função é definido o tempo de atraso que o trem menos prioritário deverá sofrer. O tempo definido é então adicionado ao tempo previamente existente no trecho em questão. O objetivo

é escolher o menor tempo de atraso possível para que o conflito em questão possa ser eliminado sem onerar muito o tempo de viagem do trem menos prioritário. Em adição, na função *DefinirParametros* são atualizados os tempos de todas as rotas em função do atraso sofrido pelo trem menos prioritário. Essa atualização de parâmetros gera uma solução temporária x'. Nesse momento são calculados os conflitos da solução corrente x e também da solução temporária x'. Se os conflitos forem reduzidos ou permanecerem os mesmos então a nova solução é aceita e x' é movido para x. Senão, um novo conflito é escolhido ao acaso. Em ambos os casos o processo de busca continua enquanto existir um conflito, enquanto o número de iterações não atingir o número máximo de iterações e enquanto o número de conflitos não sofrer qualquer redução.

Algoritmo 9: *BuscaLocal()* | Busca Local tipo *First Improvement*

Entrada: x, n
Saída: Solução *First Improvement* encontrada

$iter_{max} \leftarrow Arredondar(n/10)$;
$conflitos_{atual} \leftarrow VerificarConflitos(x)$;
$conflitos_{ant} \leftarrow conflitos_{atual}$;
enquanto $conflitos_{atual} > 0$ *e* $iter \leq iter_{max}$ *e* $conflitos_{atual} \geq conflitos_{ant}$ **faça**
    $conflito = Aleatorio(conflitos_{atual})$;
    $tremMP = conflito.tremMenosPrioritario$;
    $trechoAtraso = PosicaoParada(tremMP.rota)$;
    $p = DefinirParametros(x)$;
    $x' \leftarrow AtribuirTempoTremMP(x, p, tremMP, trechoAtraso)$;
    $conflitos_x \leftarrow VerificarConflitos(x)$;
    $conflitos_{x'} \leftarrow VerificarConflitos(x')$;
    se $conflitos_{x'} \leq conflitos_x$ **então**
        $x \leftarrow x'$;
        $conflitos_{ant} \leftarrow conflitos_{atual}$;
        $conflitos_{atual} \leftarrow conflitos_{x'}$;
    senão
        $conflitos_{atual} \leftarrow conflitos_x$;
    fim
    $iter = iter + 1$;
fim
retorna x

4.3 Estruturas de Vizinhança

As estruturas de vizinhança são utilizadas para promover perturbações na solução corrente x de modo que outra solução x' possa ser obtida na vizinhança de x. Essa pertubação é essencial para que o VNS consiga escapar de um possível ótimo local em busca do ótimo global. Quanto às estruturas de vizinhança em si, elas devem ser

estrategicamente elaboradas com base nas situações que poderiam levar o VNS a ficar preso em regiões distantes do ótimo global.

Nesta pesquisa a vizinhança de uma solução é caracterizada por mudanças nos tempos de atraso sofridos pelos trens em cada um dos trechos que compõem sua rota. É importante notar, portanto, que as perturbações sofridas pela solução corrente não resultam em alterações na topologia física do grafo que representa a ferrovia em avaliação em cada um dos testes que serão realizados.

Dado esse contexto, foram estabelecidas seis estruturas de vizinhança para serem utilizadas com o MO-SVNS, que podem ser vistas na Tabela 2. Elas são executadas da primeira para a última, à medida da necessidade. Em cada uma dessas estruturas, a busca local é executada para intensificar a procura pelo ótimo local, que pode vir a ser o ótimo global que se deseja encontrar ao final do processo de otimização. Se o processo de busca ficar estagnado, uma nova vizinhança é utilizada para diversificar a região de busca, tal como ilustrado na Figura 11, página 48.

Tabela 2 – Vizinhanças utilizadas pelo MO-SVNS.

Vizinhança	Descrição da vizinhança
Viz–1	Caso o trem fique parado em algum trecho, permutar dois dos seus tempos de parada mesmo que em um dos trechos o tempo seja zero.
Viz–2	Onde tiver tempo de parada para um dado trem, escolher um trecho e reduzir o tempo em 10%. Em seguida, o tempo referente à redução é somado ao tempo de parada do trecho anterior.
Viz–3	Onde o trem tiver uma parada superior a 5 minutos, escolher um desses trechos e reduzir o tempo em 10%.
Viz–4	Para um trem e um trecho, ambos escolhidos aleatoriamente, inserir um tempo de parada padrão para criar oportunidades de ultrapassagem de trens.
Viz–5	Para o trem que estiver envolvido no maior número de conflitos, inserir um tempo de parada padrão no trecho inicial de sua rota para atrasar sua partida.
Viz–6	Para um trem escolhido aleatoriamente, inserir um tempo de parada padrão no trecho inicial de sua rota para atrasar sua partida.

A seguir as estruturas de vizinhança serão ilustradas e explicadas em mais detalhes. Foram construídos trechos hipotéticos para ilustrar a rota resultante da aplicação de uma dada estrutura de vizinhança. Nas figuras abaixo, a rota original é indicada por (a) e a rota resultante é indicada por (b). A variável δ indica o tempo de duração de uma parada e δ' indica a nova duração da parada após aplicada uma perturbação em uma estrutura de vizinhança.

A Figura 14 mostra a aplicação da primeira estrutura de vizinhança (*Viz-1*). Ao permutar dois tempos de parada distintos há a possibilidade de que sejam criadas oportunidades de cruzamento entre os trens em circulação.

Na segunda estrutura de vizinhança (*Viz-2*), Figura 15, o objetivo é fazer reduções gradativas dos tempos de parada nos trechos onde elas ocorrem. As reduções começam pelos tempos de atraso ocorridos ao final da rota de um trem, avançando para os trechos mais iniciais. Essas reduções são compensadas em trechos anteriores da rota, que recebem o intervalo de tempo removido do trecho onde ocorreu a redução. Em um primeiro momento

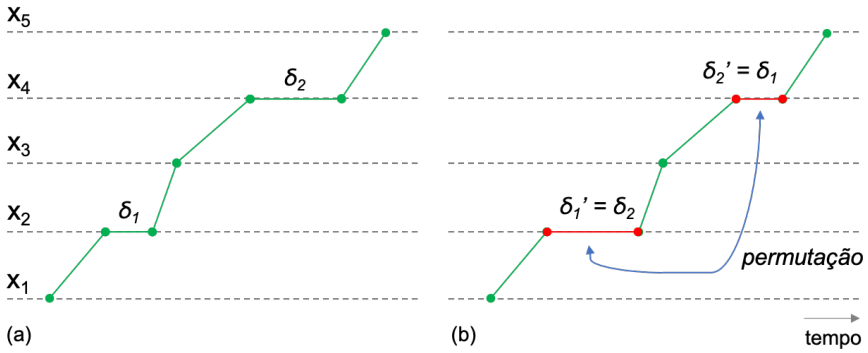

Figura 14 – Vizinhança 1: (a) rota de referência, (b) rota com o movimento promovido pela estrutura de vizinhança.

Fonte: O autor

não ocorre um ganho efetivo já que o tempo total de parada não é reduzido. Contudo, à medida em que essa vizinhança é explorada os tempos de parada são deslocados para o trecho inicial da rota do trem, onde podem ser ignorados já que o efeito prático é que a partida do trem seja atrasada em relação ao planejamento inicial.

É importante notar que do ponto de vista operacional não é bom que ocorra qualquer atraso na partida ou na chegada dos trens. Entretanto, a solução dada pelo método proposto nesta pesquisa tem exatamente o intuito de melhorar o planejamento do tráfego de trens. Desse modo, podem ser evitados quaisquer atrasos não previstos e, ainda, podem ser identificadas oportunidades de ganho ao longo do tempo. Esses ganhos são relativos à redução do tempo médio de viagem dos trens.

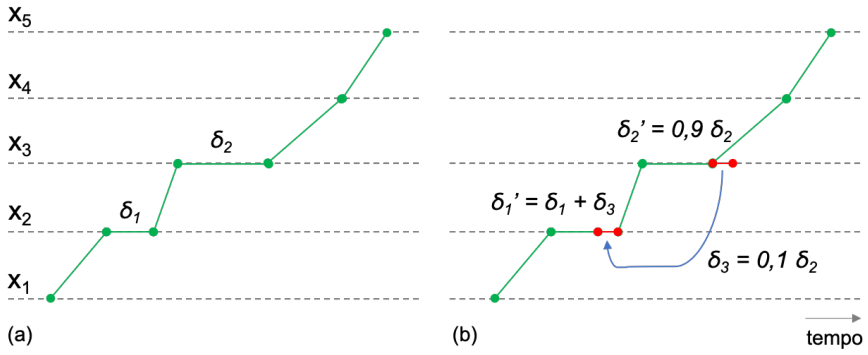

Figura 15 – Vizinhança 2: (a) rota de referência, (b) rota com o movimento promovido pela estrutura de vizinhança.

Fonte: O autor

Diferentemente dos dois casos anteriores, na terceira estrutura de vizinhança (*Viz-3*) os tempos de parada superiores a 5 minutos são localizados e reduzidos na medida do possível. A redução é de 10% do tempo de atraso atribuído ao trecho escolhido. Nesse caso,

a parcela de tempo reduzida não migra para outro trecho, resultando em uma redução imediata do tempo de viagem. A Figura 16 ilustra o procedimento.

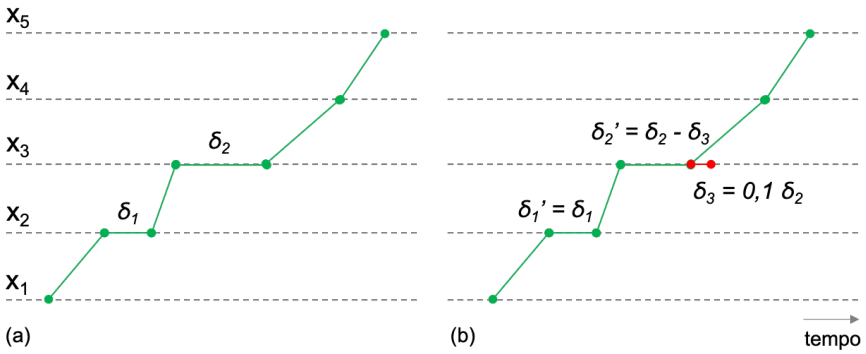

Figura 16 – Vizinhança 3: (a) rota de referência, (b) rota com o movimento promovido pela estrutura de vizinhança.

Fonte: O autor

Na Figura 17 é ilustrado o procedimento referente à quarta estrutura de vizinhança (*Viz-4*). Aqui são criadas oportunidades de cruzamento de trens mesmo que o tempo médio de viagem seja aumentado. Essa medida atribui tempos de atraso a trechos aleatórios de um trem escolhido ao acaso. É importante notar que um trem só pode passar por um trecho se este estiver desocupado no intervalo de tempo em que se pretende utilizá-lo. Por isso, é preciso encontrar o equilíbrio entre os tempos de atraso e os tempos de viagem.

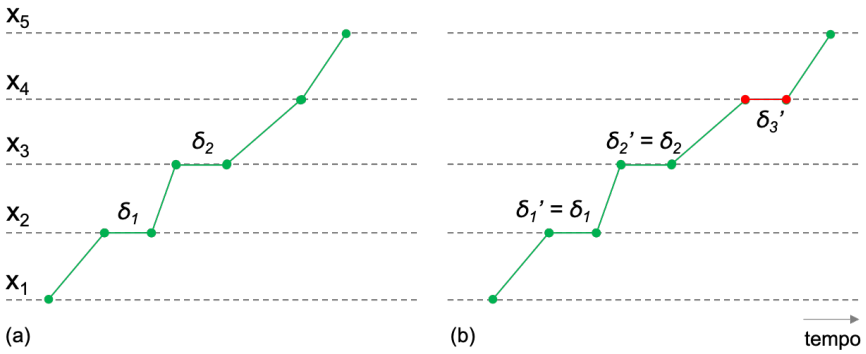

Figura 17 – Vizinhança 4: (a) rota de referência, (b) rota com o movimento promovido pela estrutura de vizinhança.

Fonte: O autor

Já na Figura 18 é mostrado um exemplo que se aplica tanto à quinta (*Viz-5*) quanto à sexta (*Viz-6*) estrutura de vizinhança. Na quinta estrutura é escolhido o trem que está envolvido no maior número de conflitos e ao seu horário de partida é adicionado um tempo de atraso. O efeito prático é a postergação da partida do trem, mudando também as referências de todos os conflitos nos quais o trem em questão estaria envolvido. Na

sexta estrutura de vizinhança o procedimento é o mesmo, porém com a particularidade de que o trem que sofrerá o atraso na partida será escolhido ao acaso. A aleatoriedade é aplicada para que sejam evitadas soluções determinísticas. Além disso, uma perturbação na rota de um trem pode acarretar em um grande número de conflitos de rota entre os trens que trafegam pela rede.

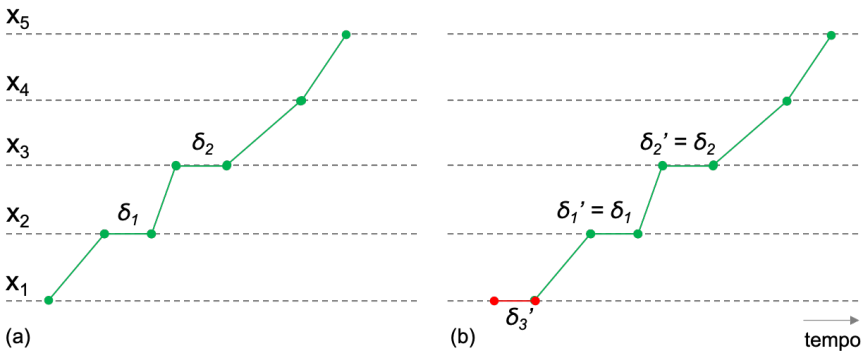

Figura 18 – Vizinhanças 5 e 6: (a) rota de referência, (b) rota com o movimento promovido pela estrutura de vizinhança.

Fonte: O autor

4.4 Redução de Tempos de Viagem

Tomando como base os tipos de atraso vistos na Seção 3.5, foi proposta uma heurística para identificar e eliminar as paradas indesejadas. Trata-se de um algoritmo de redução de tempos inspirado por Landex et al. (2006). Os Algoritmos 10 e 11 mostram como é feito o processo de redução dos tempos de viagem.

No Algoritmo 10 busca-se os tempos de parada superiores a 1 segundo, que quando eliminados não geram qualquer conflito. Esse tempo foi escolhido de modo que se pudesse reduzir ao máximo os tempos de parada. O efeito prático esperado é que esses tempos sejam tão pequenos que o maquinista possa facilmente variar a velocidade do trem ao longo de um trecho de modo que uma determinada parada sequer seja necessária. Desse modo os trens poderiam trafegar com maior fluidez.

O algoritmo supracitado tem como entrada somente a solução corrente x. O procedimento inicia com a ordenação dos horários em que cada trem sai de cada um dos trechos por onde passa. Para cada um dos trens é verificada a existência de trechos com atrasos superiores ao tempo mínimo estabelecido como alvo, ajustado em 1 segundo. Então, para cada um dos trechos encontrados é feita uma varredura de 0% a 100% do tempo que o trem fica parado com o objetivo de verificar o menor tempo possível que poderia ser utilizado sem que qualquer novo conflito fosse gerado. Quando esse tempo é encontrado

a solução é efetivada e então o processo de busca continua com os próximos trechos que compõem a rota do trem em análise.

Algoritmo 10: *ReduzirParadaTipoI()* | Redução de tempos de parada *Tipo I*

Entrada: x
Saída: *Solução com redução de tempos de parada Tipo I*

```
1  TremOrd ← OrdenarTrens(x);
2  TempoMin ← 1 segundo;
3  para cada TremOrd faça
4  |  tr ← TrechosComTempoDeParada(x, TempoMin);
5  |  para cada tr faça
6  |  |  para tempo ← 0 até 1 passo 0.05 faça
7  |  |  |  tempoTeste ← tempo * TempoParada(tr);
8  |  |  |  tempoIdeal ← TestarTempos(x, tr, tempoTeste);
9  |  |  |  se tempoIdealEncontrado então
10 |  |  |  |  AceitarSolucao(x, tempoIdeal);
11 |  |  |  fim
12 |  |  fim
13 |  fim
14 fim
15 retorna x
```

Algoritmo 11: *ReduzirParadaTipoII()* | Redução de tempos de parada *Tipo II*

Entrada: x
Saída: *Solução com redução de tempos de parada Tipo II*

```
1  tempoMin ← 30min;
2  tr ← TrechosComTempoDeParada(x, tempoMin);
3  janelaTempo ← LocalizarJanelasDeTempo(x, tr, tempoMin);
4  horaIni ← primeiro horário de partida de um trem;
5  horaFim ← último horário de chegada de um trem;
6  para cada janelaTempo faça
7  |  numConflitos_antes ← VerificarConflitos(x);
8  |  para tempo ← horaIni até horaFim passo tempoMin faça
9  |  |  x' ← RemoverTempoJanela(x, tempo);
10 |  |  numConflitos_depois ← VerificarConflitos(x');
11 |  |  se numConflitos_antes = numConflitos_depois então
12 |  |  |  AceitarSolucao(x, x');
13 |  |  senão
14 |  |  |  Sair e verificar próxima JanelaTempo
15 |  |  fim
16 |  fim
17 fim
18 retorna x
```

Há situações em que a remoção de um intervalo de tempo de parada de qualquer um dos trens acarreta na geração de novos conflitos. Todavia, a remoção de uma janela de

tempo comum a todos os trens ainda é possível. No Algoritmo 11 é realizada uma busca pelas janelas de tempo de parada que sejam comuns a todos os trens em circulação. Neste caso, o objetivo específico é encontrar janelas de tempo superiores a 30 minutos. Esse tempo foi escolhido como gatilho para que o algoritmo de busca e eliminação de janelas de tempo pudesse ser acionado. Todavia, podem ser removidos tempos superiores à janela definida como gatilho.

Para a busca da janela de tempo somente a solução corrente x é utilizada como entrada. Para cada um dos trens são levantados os trechos em que ocorram paradas iguais ou superiores à janela de tempo que se deseja eliminar, ajustada em 30 minutos. Feito isso, os trechos encontrados são armazenados levando consigo as informações da localização e do intervalo de tempo da parada. Após todo o levantamento os tempos de parada armazenados para cada um dos trens são comparados para verificar a ocorrência de intervalos concorrentes. Assim que uma janela de tempo em comum é localizada, é feito um teste para certificar que não haja qualquer conflito entre os trens em circulação no intervalo de tempo em avaliação. Se nenhum conflito for encontrado, a janela de tempo é eliminada para todos os trens gerando uma nova solução. Os conflitos são novamente verificados e se qualquer novo conflito for gerado pela eliminação da janela de tempo a solução não é efetivada e uma nova busca é iniciada utilizando novos intervalos de tempo.

5 Método de Avaliação de Soluções

Em problemas com mais de um objetivo é bastante frequente o uso da Fronteira de Pareto para avaliar a melhor relação entre os objetivos a serem otimizados. E para verificar a qualidade da Fronteira de Pareto, algumas métricas podem ser calculadas e analisadas. Nesta pesquisa serão empregadas três métricas bastante utilizadas na literatura e uma quarta proposta por esta pesquisa. Esses assuntos serão abordados neste capítulo e o fechamento se dará com a apresentação do critério utilizado para eleger a Fronteira de Pareto teórica para problemas sem referência exata na literatura, como ocorre nesta pesquisa.

5.1 Fronteira de Pareto

Os problemas multiobjetivo são mais difíceis de serem resolvidos do que aqueles com um único objetivo. Isso se deve ao fato de que o processo de otimização deve encontrar a melhor relação entre os objetivos do problema. Esses pontos onde ocorrem essas relações ótimas dão origem à Fronteira de Pareto (BAGCHI, 1999). Essa teoria representa o início das pesquisas acerca da otimização multiobjetivo. A determinação da Fronteira de Pareto baseia-se no conceito de dominância, que no século XIX foi generalizado por Pareto (1896). Na época ele aplicou esse conceito no contexto da economia política, mas logo começaram a surgir diversas aplicações nos mais variados campos de pesquisa.

Para problemas onde a Fronteira de Pareto já tenha sido mapeada, seja por métodos de otimização exata ou qualquer outro método, é possível utilizar essa referência para a comparação de diferentes algoritmos de otimização. Em geral esses algoritmos obtêm um conjunto de soluções que formam uma fronteira e com base em algumas métricas é possível estimar a qualidade da fronteira obtida e, inclusive, avaliar se a Fronteira de Pareto conhecida foi superada pela solução em avaliação.

De acordo com Antoniucci (2016), em um problema multiobjetivo em geral a Fronteira de Pareto PF^* gera um conjunto incontável de soluções que não pode ser completamente representado por um computador e por isso é feita uma aproximação discreta denominada PF_{known}. E o conjunto de soluções ótimas de Pareto P^* é referenciado por P_{known}. De modo similar a Fronteira de Pareto e o conjunto de soluções ótimas de Pareto são representadas computacionalmente por aproximações discretas tratáveis denominadas por PF_{true} e P_{true}, respectivamente.

Uma solução de um problema é dita Pareto Ótimo ou solução ótima de Pareto x_i^* se não houver nenhuma solução que melhore um critério sem provocar qualquer perturbação

em outro, isto é, que melhore a solução em uma dimensão sem piorar as soluções já obtidas para outras dimensões. Em geral não há somente uma única solução para o problema, mas sim um conjunto de soluções chamadas de não dominadas. Qualquer solução fora desse conjunto não é uma solução ótima e portanto é chamada de solução dominada. A fronteira que separa as soluções não dominadas das soluções dominadas é definida como sendo a Fronteira de Pareto (COELLO et al., 1999).

A Figura 19 mostra a Fronteira de Pareto para um problema com dois objetivos, $f(x)$ e $g(x)$. Ambos os objetivos devem ser minimizados. Na figura, as soluções de um problema são representadas pelos pontos. Os pontos mais claros referem-se às soluções dominadas, enquanto os mais escuros referem-se às soluções ótimas de Pareto. Segundo Bagchi (1999), para um problema biobjetivo as soluções ótimas de Pareto são tais que não existe nenhuma outra solução x possível tal que $f(x) \leq f(x_i^*)$ e $g(x) \leq g(x_i^*)$.

Na Figura 19a é mostrado o espaço de decisão das soluções e na Figura 19b é mostrado o espaço de objetivos. As setas que conectam ambas figuras exemplificam a relação entre as soluções nesses espaços.

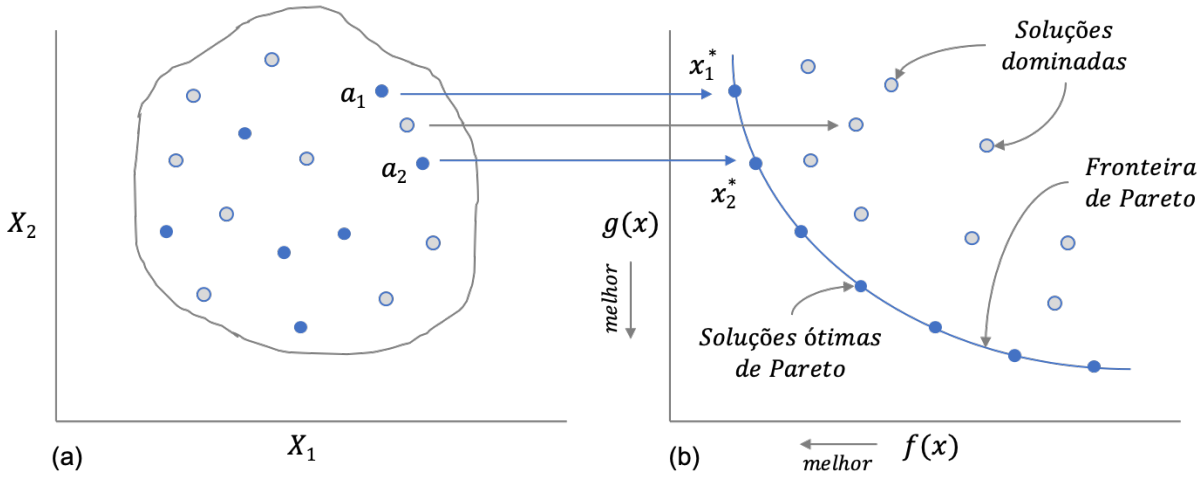

Figura 19 – Fronteira de Pareto para um problema com dois objetivos a serem minimizados: (a) espaço de decisão e (b) espaço de objetivos.

Fonte: O autor

No algoritmo desenvolvido nesta pesquisa os dados referentes às soluções são obtidos durante o procedimento do MO-SVNS. Quando esse procedimento é finalizado a Fronteira de Pareto é obtida. As soluções armazenadas ao longo do processo são avaliadas quanto à dominância para que a região de fronteira seja definida.

5.2 Métricas para Avaliação da Fronteira de Pareto

O uso da otimização combinatória na solução de problemas discretos visa encontrar os melhores resultados para a função objetivo. A solução pode ser finita ou contável infinita. Ela pode ser inteira, um conjunto, uma permutação ou um grafo (PAPADIMITRIOU; STEIGLITZ, 1998).

Conforme discutido na Seção 5.1, nos problemas multiobjetivo as soluções ótimas delimitam a Fronteira de Pareto. Uma vez que a análise desta pode não ser tão simples e direta, faz-se necessária a utilização de métricas que permitam avaliar a qualidade das soluções encontradas. Essas métricas também viabilizam a comparação entre diferentes algoritmos utilizados para solucionar um dado problema.

De acordo com Antoniucci (2016), as métricas podem ser classificadas como *unárias* ou *binárias*. As unárias mapeiam um único conjunto de soluções em um número real, enquanto as binárias também utilizam números reais mas visam a comparação entre diferentes conjuntos de soluções. As métricas unárias são mais simples e muito utilizadas na literatura, além de fornecerem resultados suficientemente bons quando comparadas às métricas *binárias*. Neste trabalho serão utilizadas métricas unárias.

Nas próximas seções serão explicadas as métricas utilizadas neste trabalho, a saber: *hypervolume*, *maximum spread*, *spacing* e *amplitude*. Esta última é uma proposta do autor.

5.2.1 *Hypervolume*

A métrica *Hypervolume* (HV) foi proposta por Zitzler e Thiele (1998) para avaliação da Fronteira de Pareto. No caso da otimização biobjetivo, cada solução não dominada da Fronteira de Pareto cobre uma área retangular. A união dessas áreas constitui a área total de cobertura da Fronteira de Pareto. Essa área define a métrica HV e seu cálculo pode ser realizado de modo similar para problemas com mais de dois objetivos. A vantagem dessa métrica é que ela independe da meta-heurística utilizada. Além disso, ela fornece informações sobre a diversidade e, quando comparada à melhor Fronteira de Pareto, PF_{true}, também fornece informações sobre a convergência das soluções. Valores mais elevados de HV indicam melhor desempenho.

A Figura 20 mostra um exemplo em que o HV é calculado para um problema de otimização biobjetivo onde ambos objetivos devem ser minimizados. Nesse caso, deve ser escolhido um ponto p de referência para que as áreas possam ser calculadas. Esse ponto deve ser escolhido de tal modo que não tenha as coordenadas x e y muito maiores do que aquelas dos pontos x_1^* e x_n^* para evitar erros de interpretação dos resultados. Nesta pesquisa o ponto $p(x, y)$ foi escolhido como tendo uma unidade a mais em relação às coordenadas dos pontos x_1^* e x_n^*. Na Figura 20a é mostrada a região de cobertura de cada solução dominante e na Figura 20b é mostrada a área que representa o HV para o conjunto

de soluções.

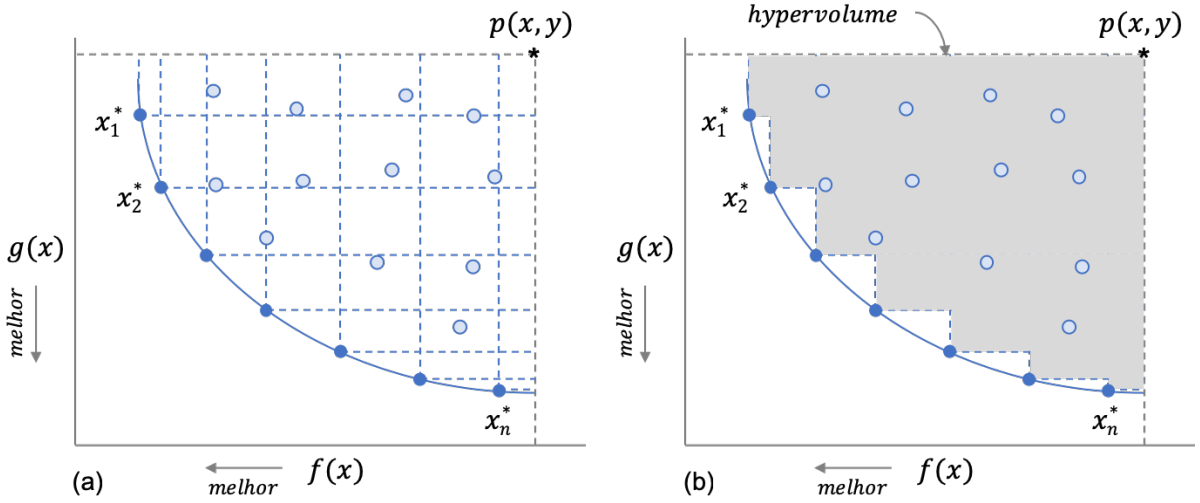

Figura 20 – Métrica *Hypervolume* para a Fronteira de Pareto de um problema de otimização biobjetivo onde ambos objetivos devem ser minimizados: (a) região de cobertura de cada solução dominante e (b) *hypervolume* para a Fronteira de Pareto.

Fonte: O autor

5.2.2 *Maximum Spread*

A dispersão dos valores da Fronteira de Pareto deve ser maximizada de modo que as soluções não dominadas cubram um amplo conjunto de soluções dominadas. Esse é um requisito destacado por Zitzler (1999) ao propor a métrica *Maximum Spread* (MS). Sendo assim, é utilizada a maior extensão em cada dimensão para estimar o alcance das soluções do conjunto não dominado. No caso da otimização biobjetivo essa métrica é calculada pela distância euclidiana entre os pontos mais extremos da Fronteira de Pareto, no espaço de objetivos. A Equação 5.1 define a expressão utilizada para esse cálculo, onde γ é o número de soluções da Fronteira de Pareto e M é o número de objetivos do problema em questão. Assim como no HV, valores mais elevados de MS indicam melhor desempenho.

$$MS = \sqrt{\sum_{j=1}^{M} (max_{i=1}^{\gamma} f_j^i - min_{i=1}^{\gamma} f_j^i)^2} \tag{5.1}$$

5.2.3 *Spacing*

Proposta por Schott (1995), a métrica *Spacing* (SP) é obtida pelo cálculo do desvio padrão da distância de Manhattan entre as soluções vizinhas que compõem a Fronteira de Pareto. Se o valor de SP for igual a zero significa que todas as soluções estão espaçadas de

modo equidistante na Fronteira de Pareto. Seu cálculo é descrito pela Equação 5.2, onde γ corresponde ao número de soluções da Fronteira de Pareto, PF_{known}, $\bar{d}$ se refere à média de todos os d_i, onde d_i é descrito pela Equação 5.3 e tanto i quanto j variam de 1 a γ.

Essa métrica deve ser avaliada juntamente com outras uma vez que ela apenas fornece informação acerca do espalhamento das soluções da Fronteira de Pareto.

$$SP = \sqrt{\frac{1}{\gamma - 1} \sum_{i=1}^{\gamma} (\bar{d} - d_i)^2} \tag{5.2}$$

$$d_i = min_j(|f_1^i(\vec{x}) - f_1^j(\vec{x})| + |f_2^i(\vec{x}) - f_2^j(\vec{x})|) \tag{5.3}$$

5.2.4 Métrica Proposta: *Amplitude*

A métrica *Amplitude* (AM) foi proposta nesta pesquisa como uma alternativa para que se possa estimar uma boa Fronteira de Pareto para um dado problema. Cada vez que se executa um método de solução para um problema multiobjetivo, são obtidas soluções que definem a melhor relação entre os objetivos de otimização. No caso desta pesquisa os objetivos são: (a) número de conflitos e (b) tempo médio de parada dos trens em circulação.

As soluções obtidas podem ser influenciadas pela topologia da rede, número de trens em circulação, velocidades praticadas por cada trem em cada trecho da rede, entre outros fatores. Desse modo, quanto mais particular é o problema maior é a chance de que ele não tenha uma referência estabelecida para a Fronteira de Pareto. Essa referência diz respeito às melhores soluções conhecidas.

Sendo assim, a concepção da métrica AM vem da necessidade de criar uma boa referência para a Fronteira de Pareto, quando ela ainda não foi definida para o problema em estudo nas condições consideradas. Essa métrica usa como base os valores obtidos para o *hypervolume*, *maximum spread* e *spacing* de um problema que se deseja otimizar. O valor da métrica AM pode ser obtido segundo as Equações 5.4 e 5.5, descritas a seguir.

$$AM = ref_{max} - min_{SP} \tag{5.4}$$

$$ref_{max} = max_{HV} + max_{MS} \tag{5.5}$$

onde, para cada execução do algoritmo de otimização, são coletadas as referências do valor máximo da métrica HV (max_{HV}), do valor máximo da métrica MS (max_{MS}) e do valor mínimo da métrica SP (min_{SP}).

A Figura 21 mostra um gráfico em que o valor da métrica AM é calculado à medida em que as soluções encontradas pelo MO-SVNS vão sendo adicionadas ao repositório

Pareto. As variáveis que compõem as Equações 5.4 e 5.5 são destacadas no gráfico. O valor da métrica AM é a medida da amplitude entre ref_{max} e min_{SP}.

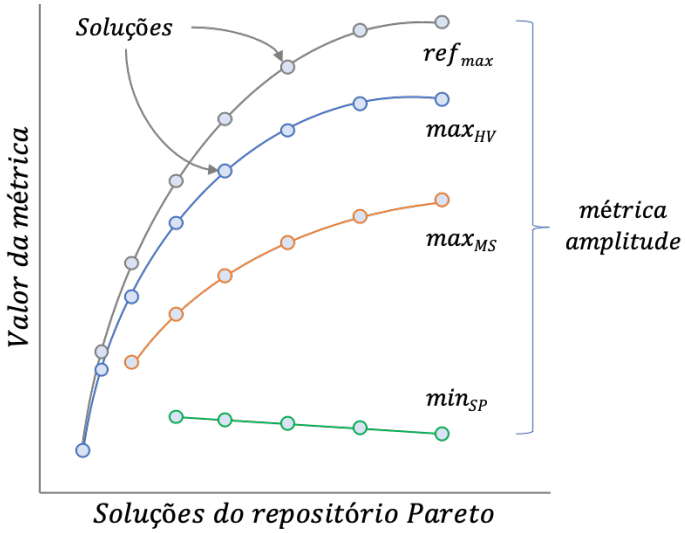

Figura 21 – Métrica *Amplitude* para escolha da melhor Fronteira de Pareto em problemas sem referências para a Fronteira de Pareto.

Fonte: O autor

5.3 Escolha da Melhor Fronteira

Cada instância deve ser executada um número j de vezes. Em cada uma dessas execuções são coletadas as soluções que dão origem a uma fronteira candidata a Fronteira de Pareto. A escolha da melhor fronteira que será tida como Fronteira de Pareto para o problema de planejamento e roteamento de trens estudado nesta pesquisa é realizada em três etapas, a saber:

- **Etapa 1**: Seleção das fronteiras que apresentem o menor número de conflitos restantes. O ideal é que ao final do processo de otimização não reste nenhum conflito entre as rotas dos trens em circulação. Contudo, caso haja algum conflito, este deverá ser solucionado manualmente por um operador do centro de controle de trens;
- **Etapa 2**: A fronteira é resultante de um gráfico que relaciona o número de conflitos pelo tempo médio de parada dos trens. Para excluir valores muito discrepantes, foi calculada a média e o desvio padrão do maior tempo médio de parada dos trens para as fronteiras selecionadas na etapa anterior. Todas as fronteiras que têm tempo de parada superior a $\beta = \bar{x} + s$ são eliminadas, onde β é o fator calculado, $\bar{x}$ é a média do tempo e s é o desvio padrão do tempo;

- **Etapa 3**: A partir das fronteiras selecionadas na etapa anterior é escolhida como sendo a Fronteira de Pareto aquela que tiver o maior valor de AM.

O procedimento acima busca garantir que cada um dos objetivos da otimização biobjetivo tenha o melhor valor possível, ao passo em que valoriza a fronteira que apresenta valores maiores de HV e MS, além de valores menores de SP garantindo uma boa uniformidade na distribuição das soluções que formam a fronteira.

6 Experimentos e Resultados

No Capítulo 1 foi feita a introdução da pesquisa, onde foram apresentados os cenários utilizados para testar o método proposto nesta pesquisa. Diversas fontes de inspiração foram apresentadas no Capítulo 2. Os trabalhos citados trouxeram importantes pontos de vista e orientações sobre como os problemas de planejamento e de roteamento de trens poderiam ser solucionados, em especial utilizando métodos heurísticos. No Capítulo 3 foram apresentados alguns detalhes sobre o modo como o problema foi modelado e como os conflitos seriam encontrados e minimizados. Em seguida, no Capitulo 4, foi apresentada a meta-heurística VNS e alguns dos seus esquemas incluindo o MO-SVNS, utilizado nesta pesquisa. No Capítulo 5 foi apresentada a teoria sobre a Fronteira de Pareto e sobre algumas métricas utilizadas para avaliação da qualidade da fronteira obtida.

Neste capítulo são apresentados os experimentos realizados com foco tanto na avaliação do problema de planejamento quanto no problema de roteamento de trens. Também é apresentado o fluxo básico do processo de otimização, com um breve resumo de cada uma das etapas. Na sequência são mostrados os parâmetros considerados em cada um dos cenários testados. São comentados os resultados obtidos para todas as instâncias no cenário cujo foco é no planejamento de trens e em seguida é escolhida uma das instâncias para que todos os cenários possam ser verificados. O cenário identificado como sendo o mais adequado é novamente utilizado para um novo teste com todas as instâncias. Neste último teste, são utilizadas novas estruturas de vizinhança focadas na obtenção de soluções em tempos computacionais reduzidos. Ao final os testes são comparados com base na análise de algumas variáveis de processo.

6.1 Fluxograma do Método de Solução

Com base em todo o conteúdo apresentado nos capítulos anteriores é apresentado na Figura 22 o fluxo básico do método de solução proposto nesta pesquisa. Detalhes adicionais podem ser vistos no Apêndice A.

Os principais parâmetros citados ao longo do texto e importantes para os experimentos serão retomados a seguir com base no fluxograma apresentado:

- **Parâmetros iniciais**: Compreende as questões que devem ser respondidas ao iniciar a execução da aplicação. Os principais parâmetros dizem respeito ao uso da métrica *betweenness*, número de rotas que devem ser calculadas para cada trem e número de repetições do método de solução para cada uma das instâncias;

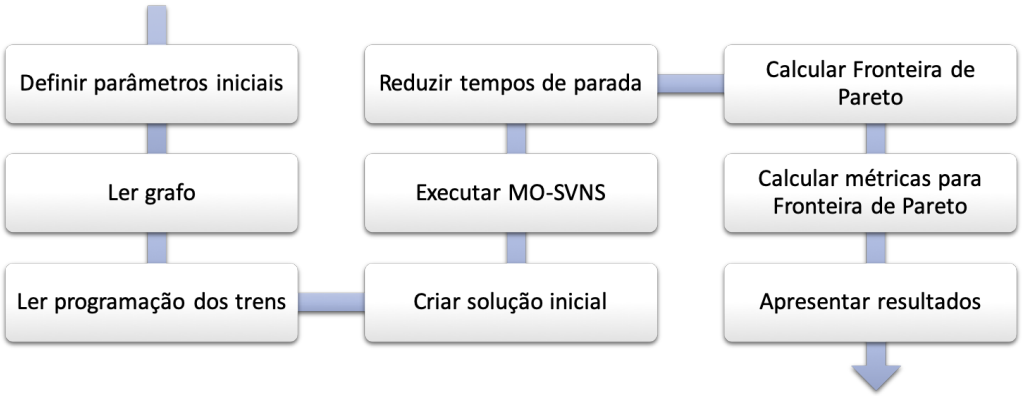

Figura 22 – Fluxograma do processo de planejamento e roteamento de trens.

Fonte: O autor

- **Grafos**: Os grafos disponíveis são listados e disponibilizados para teste. É possível escolher mais de um grafo para que os testes sejam executados com base nos parâmetros previamente selecionados;
- **Programação dos trens**: Contém os vértices de origem e destino de cada um dos trens, bem como o horário de partida desejado;
- **Solução inicial**: É calculado o número de rotas solicitadas para cada um dos trens, conforme a escolha realizada nos parâmetros iniciais. Se for calculada mais de uma rota a escolha da melhor entre elas é feita em função do número de conflitos gerados ao selecionar cada uma delas. Na solução inicial não são inseridos tempos de atraso em nenhum dos trechos que compõem as rotas dos trens;
- **MO-SVNS**: Essa meta-heurística recebe como entrada a solução inicial. Ela tem como objetivo minimizar o número de conflitos entre as rotas dos trens e também o tempo médio de parada dos trens, que por sua vez tem influência direta no tempo médio de viagem dos trens;
- **Redução de tempos**: A saída do MO-SVNS gera uma nova solução que traz em si algumas oportunidades de melhoria. Nessa etapa são removidos os tempos de atraso considerados desnecessários. É importante notar que a exclusão desses tempos não gera qualquer conflito adicional em relação à solução entregue pelo MO-SVNS;
- **Fronteira de Pareto**: Ao longo do tempo em que a meta-heurística MO-SVNS trabalha na otimização das duas funções objetivo, número de conflitos e tempo de parada, são coletados dados referentes à Fronteira de Pareto. Esses dados integram o repositório Pareto para serem utilizados mais tarde. Por não haver uma referência da Fronteira de Pareto para o problema estudado nesta pesquisa, foi proposto um método para eleger a melhor fronteira dentre aquelas obtidas em todas as repetições

de uma instância. Para isso foram utilizadas três métricas conhecidas e uma quarta métrica proposta pelo autor.

Nesta pesquisa primeiro foi utilizado o BVNS, depois o SVNS e então o MO-SVNS. Os resultados obtidos com o BVNS não foram satisfatórios uma vez que o algoritmo nem sempre conseguia eliminar todos os conflitos existentes. Além disso, quando os eliminava, por vezes o tempo total de viagem dos trens acabava sendo muito elevado. Com o SVNS foi possível explorar soluções mais distantes na vizinhança da solução corrente. Isso elevou bastante a garantia de eliminação de todos os conflitos de rotas além de reduzir muito os tempos de viagem quando comparados aos resultados obtidos pelo BVNS. Ainda assim, os tempos de viagem obtidos eram muito elevados para as pretensões desta pesquisa. Com o intuito de reduzir os tempos médios de parada dos trens e, consequentemente, os tempos totais de viagem, optou-se por utilizar o MO-SVNS. Desse modo, tanto o número de conflitos quanto o tempo de parada se tornaram os objetivos que deveriam ser minimizados para que o problema de planejamento e roteamento de trens fosse resolvido. Não obstante, embora o desempenho do MO-SVNS seja melhor, é importante mencionar que eventualmente qualquer um dos três métodos citados pode não conseguir eliminar todos os conflitos. Se isso ocorrer, é necessário que um operador de tráfego de trens atue para eliminar os conflitos restantes.

No desenvolvimento do algoritmo de otimização foi assumida a premissa de que nenhum trem pode partir do ponto inicial de sua rota antes do horário programado para o início da sua viagem. Para o algoritmo desenvolvido isso seria como atribuir um tempo negativo ao horário de partida. É importante notar que são esperados ajustes no horário de partida dos trens. Isso ocorre porque na solução inicial foi gerado um grande número de congestionamentos por não ter sido atribuído qualquer tempo de atraso às rotas dos trens.

6.2 Cenários de Teste

Para testar o método de solução proposto foram elaborados doze diferentes cenários. Os detalhes sobre esses cenários foram resumidos na Tabela 3. Ela resume o que foi visto no Capítulo 1, Seção 1.2.1.

Ao longo dos experimentos serão realizados três diferentes testes. No primeiro deles, todas as instâncias serão executadas no cenário *Cen–1*. Em seguida, no segundo teste, a instância *94v112a* será testada em todos os cenários. Então, o cenário que se destacar como sendo o melhor será utilizado para executar o terceiro teste. Neste caso, todas as instâncias serão novamente executadas. Em todos os casos será feita uma série de análises sobre os resultados obtidos.

A seguir são explicados os parâmetros utilizados na Tabela 3:

Tabela 3 – Parâmetros para os cenários analisados.

Cenário	Execuções	Rotas	Pond/Unit	Pond/Btw	Tempo Limite	Manter Trens
Cen–1	5	1	Pond	Pond	Não	Sim
Cen–2	5	1	Unit	Pond	Não	Sim
Cen–3	5	3	Pond	Pond	Não	Sim
Cen–4	5	3	Unit	Pond	Não	Sim
Cen–5	5	1	Pond	Btw	Não	Sim
Cen–6	5	1	Unit	Btw	Não	Sim
Cen–7	5	3	Pond	Btw	Não	Sim
Cen–8	5	3	Unit	Btw	Não	Sim
Cen–9	5	3	Pond	Pond	Sim	Sim
Cen–10	5	3	Pond	Pond	Sim	Não
Cen–11	5	3	Pond	Btw	Sim	Sim
Cen–12	5	3	Pond	Btw	Sim	Não

- **Execuções**: Cada uma das instâncias foi executada cinco vezes passando pelo procedimento completo do método de solução proposto. Esse número de execuções foi adotado para que as estatísticas geradas fossem mais consistentes. Embora fosse ideal realizar mais repetições para cada instância, isso não foi feito devido ao tempo muito longo exigido por algumas delas;

- **Rotas**: O número de rotas utilizadas em cada um dos cenários pode ser uma ou três. Quando somente uma rota é utilizada a solução do problema é focada no planejamento dos trens, onde precisam ser encontrados horários de entrada e saída dos trens em cada um dos trechos da ferrovia. Por outro lado, quando são utilizadas três rotas, a fase inicial é focada em escolher uma dentre três opções de rota e a fase seguinte é focada no planejamento, tal qual comentado anteriormente;

- **Pont/Unit**: Os pesos dos vértices indicam o comprimento de cada um dos trechos de ferrovia. Os vértices podem ser ponderados ou podem ter peso unitário;

- **Pond/Btw**: Essa opção informa se as rotas devem ser calculadas em função da ponderação dos vértices do grafo ou em função da métrica *betweenness* calculada para cada um dos vértices do grafo. Se no item anterior for escolhido grafo ponderado, será obtida a *betweenness ponderada*;

- **Tempo Limite**: Se um tempo limite é dado ao MO-SVNS, quando o tempo é atingido o procedimento MO-SVNS é finalizado. Essa opção é utilizada para casos em que soluções que apresentem um determinado número de conflitos seja aceitável. Neste caso, é assumida a premissa de que operadores de tráfego farão o tratamento manual dos conflitos restantes;

- **Manter Trens**: Caso reste qualquer conflito na solução final, estes podem ser suprimidos pela eliminação de alguns trens. Se a opção escolhida for de não manter os trens, serão eliminados tantos trens quantos forem necessários para que o número de conflitos seja igual a zero.

Na Figura 23 é mostrada a representação em árvore dos parâmetros relativos aos doze cenários de teste elaborados. Essa árvore resume as explicações anteriores acerca dos cenários.

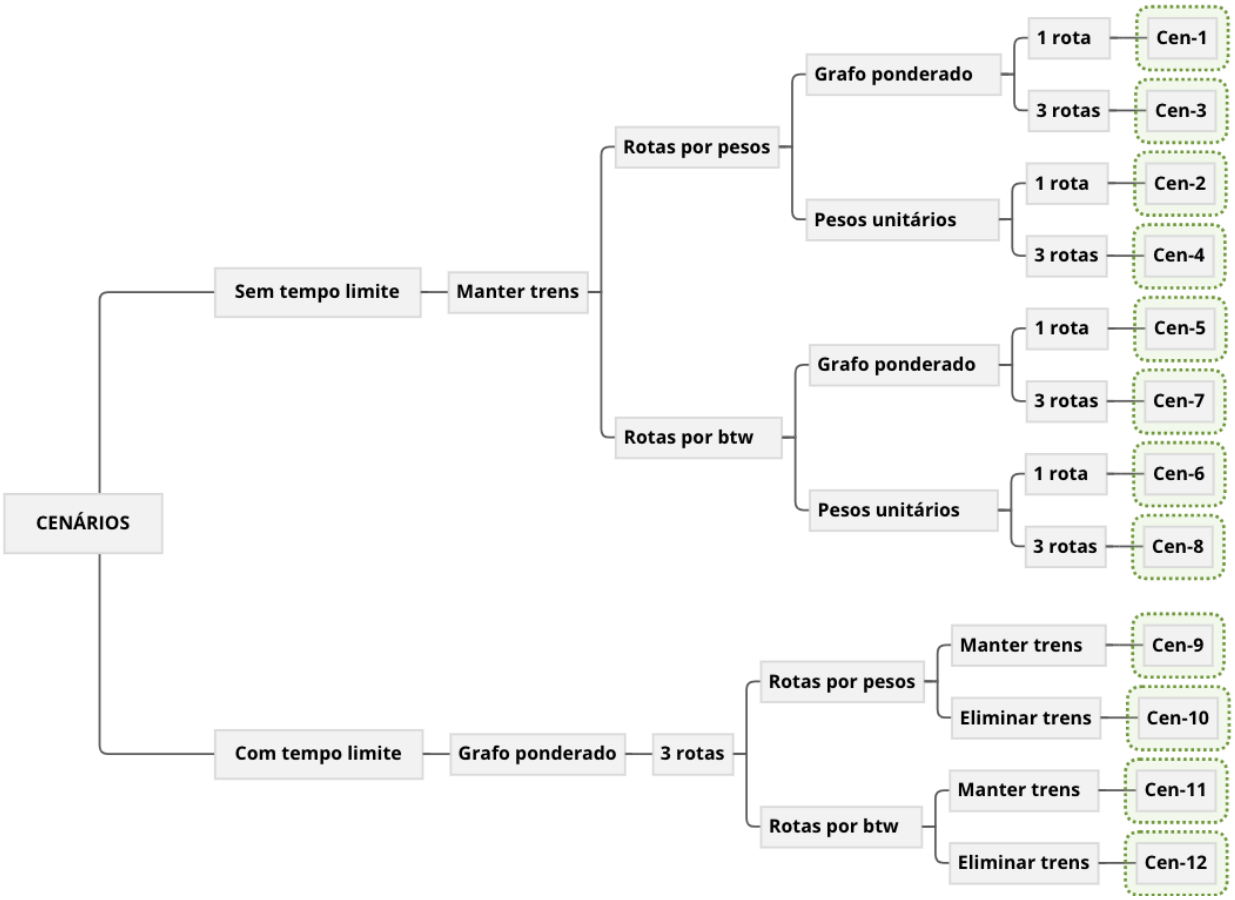

Figura 23 – Representação em árvore dos 12 cenários de teste.

Fonte: O autor

6.3 Definição de Parâmetros

Ao longo dos testes diversos parâmetros foram calculados para suportar as análises dos resultados obtidos. A Tabela 4 resume esses parâmetros e a descrição de cada um deles.

Em todos os cenários os resultados obtidos para os parâmetros calculados foram separados em seis diferentes grupos, a saber:

- **Entrada**: Traz os parâmetros básicos das topologias utilizadas, incluindo dados sobre os grafos, o número de trens em circulação e o número total de trechos utilizados pelas rotas de todos os trens;

Tabela 4 – Definição dos parâmetros do problema.

Parâmetro	Descrição
Rotas	Número de rotas calculadas para cada um dos trens
Conf	Número médio de conflitos das 5 execuções
ConfRest	Número médio de conflitos restantes após as 5 execuções
Trens	Número de trens que trafega na rede
Trechos	Número total de trechos
TempoLimite (s)	Tempo limite para que a rotina do VNS seja interrompida
Temp (h)	Tempo médio que o algoritmo leva para encontrar a solução final
TVI (h)	Tempo de viagem da solução inicial considerando todos os trens
TVV (h)	Tempo de viagem da solução obtida pelo MO-SVNS considerando todos os trens
TVF (h)	Tempo de viagem da solução final considerando todos os trens
TPI (h)	Tempo de parada da solução inicial considerando todos os trens
TPV (h)	Tempo de parada da solução obtida pelo MO-SVNS considerando todos os trens
TPF (h)	Tempo de parada da solução final considerando todos os trens
CFI	Número inicial de conflitos considerando todos os trens
CFV	Número de conflitos da solução obtida pelo MO-SVNS considerando todos os trens
CFF	Número de conflitos da solução final considerando todos os trens
DesvPadConf	Desvio padrão do número de conflitos das 5 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*
MelhorParetoAM	Valor da *amplitude* para a melhor Fronteira de Pareto encontrada
MelhorParetoHV	Valor do *hypervolume* para a melhor Fronteira de Pareto encontrada
MelhorParetoMS	Valor do *maximum spread* para a melhor Fronteira de Pareto encontrada
MelhorParetoSP	Valor do *spacing* para a melhor Fronteira de Pareto encontrada
MediaAM	Média aritmética da métrica *amplitude*
MediaHV	Média aritmética da métrica *hypervolume*
MediaMS	Média aritmética da métrica *maximum spreading*
MediaSP	Média aritmética da métrica *spacing*
DesvPadAM	Desvio padrão da métrica *amplitude*
DesvPadHV	Desvio padrão da métrica *hypervolume*
DesvPadMS	Desvio padrão da métrica *maximum spreading*
DesvPadSP	Desvio padrão da métrica *spacing*
CVAM	Coeficiente de variação da *amplitude*
CVHV	Coeficiente de variação do *hypervolume*
CVSP	Coeficiente de variação do *spacing*
CVMS	Coeficiente de variação do *maximum spread*

- **Densidade Conflitos**: Traz o número de conflitos iniciais que ocorrem entre as rotas dos trens em circulação. Além disso traz a relação do número de conflitos com o número de trens e com o número de trechos. Esses dados contribuem para que redes similares possam ser criadas e utilizadas em estudos futuros;

- **Tempo**: São apresentados dados relativos ao tempo de processamento das soluções e aos tempos de viagem inicial, intermediário e final. O tempo intermediário é aquele obtido pelo MO-SVNS, enquanto o tempo final é aquele obtido após a redução dos tempos de viagem;

- **Desvio Padrão**: Para entender como variam os dados foi calculado o desvio padrão do número de conflitos e dos tempos do processo, incluindo tempo de processamento e tempos de viagem;

- **Indicador**: Nesse grupo são apresentados diversos indicadores do processo de otimização. Os tempos de viagem são relacionados entre si e é também calculado o coeficiente de variação (CV). Esse coeficiente é dado pela relação σ/μ, onde σ é o

desvio padrão e μ é a média da variável em análise. Ele mensura a variabilidade de uma série numérica independentemente da unidade. O CV é principalmente utilizado em estatística descritiva, mas pode também ser utilizado em casos de análise estatística inferencial (ABDI, 2010). O CV é intrínseco a cada processo. Isso implica dizer que classificar valores segundo faixas de dispersão baixas, médias ou altas depende de estudos comparativos para que essas referências sejam estabelecidas a contento. Embora Gomes (1987) tenha realizado experimentos com problemas cujo foco não era a ferrovia ou os meios de comunicação, ele sugere algumas faixas de valores para auxiliar na avaliação do CV. Ele classifica como baixa dispersão valores abaixo de 10%, média dispersão valores compreendidos entre 10% e 20%, alta dispersão entre 20% e 30% e muito alta dispersão valores superiores a 30%. Essas faixas serão utilizadas como referência para as análises realizadas nesta pesquisa;

- **Métrica Pareto**: São apresentados os valores calculados para as quatro métricas utilizadas para avaliação da Fronteira de Pareto. Os parâmetros referentes ao *Melhor Pareto* trazem o valor da métrica para a Fronteira de Pareto escolhida como sendo a referência. Por outro lado, os valores para *Média*, *Desvio Padrão* e *CV* são calculados em função das cinco execuções realizadas para cada uma das instâncias de teste.

6.4 Planejamento de Trens

Para analisar o planejamento de trens inicialmente todas as instâncias foram testadas no cenário *Cen–1*. Os principais resultados obtidos para os parâmetros calculados são mostrados na Tabela 5. A tabela completa com os resultados relativos à melhor execução de cada instância em cada um dos cenários pode ser vista no Apêndice B.

Quando se analisa a instância *184v222a*, onde há 472 trechos e 152 conflitos e for considerado que para eliminar cada um desses conflitos bastaria adicionar um novo trecho, ao final seriam adicionados 152 novos trechos. Essa adição corresponde a um aumento de 32% em relação ao número de trechos utilizados por todas as rotas. Esse incremento no número de trechos é apenas uma estimativa já que, na realidade, a mudança de rota de um trem pode acarretar na eliminação de diversos conflitos ao mesmo tempo. Alternativamente, conforme proposto pelo método de solução utilizado neste trabalho, com um aumento médio de 96% do tempo de viagem dos trens, relação entre *TVF* e *TVI*, foi possível eliminar todos os conflitos da programação dos trens. Essa adição de tempo pode parecer elevada, mas é importante lembrar que o tempo *TVI* considera que nenhum dos trens efetua qualquer parada do início ao final de sua rota. Essa não é uma condição de tráfego realista e foi utilizada somente como referência inicial. Do ponto de vista econômico, na maior parte dos casos é mais viável transportar em mais tempo do que construir e manter novos ramais ou linhas férreas. Além do mais, com um bom

Tabela 5 – Parâmetros calculados para cada topologia utilizando o *Cen–1*.

Grupo	Parâmetro	49v57a	94v112a	184v222a	274v332a	364v442a
Entrada	Vértices	49	94	184	274	364
	Arestas	57	112	222	332	442
	Trens	10	19	37	55	73
	Trechos	66	138	472	650	976
Densidade Conflitos	Conf	8	23	152	201	319
	Conf/Trem	0,800	1,211	4,108	3,655	4,370
	Conf/Trechos	0,121	0,167	0,322	0,309	0,327
Tempo	Temp (h)	0,000	0,013	9,329	20,073	156,916
	TVI (h)	9,296	22,531	72,027	101,602	149,927
	TVV (h)	10,436	32,112	196,175	230,960	528,051
	TVF (h)	9,725	27,641	140,934	171,518	307,150
	TPV (h)	1,133	9,581	124,148	129,358	338,609
	TPF (h)	0,429	5,110	68,907	69,922	157,655
Desvio Padrão	DesvPadTemp (h)	0,000	0,006	3,464	3,749	25,828
	DesvPadTPV (h)	0,446	2,324	31,789	25,195	53,972
	DesvPadTPF (h)	0,341	0,551	10,707	6,745	3,650
Indicador	ConfRest	0	0	0	0	0
	Temp/Vértice (min)	0,000	0,009	3,042	4,395	25,865
	Temp/Trem (min)	0,002	0,042	15,129	21,897	128,972
	Temp/Conf (min)	0,003	0,035	3,683	5,992	29,514
	TVF/TVI	1,046	1,227	1,957	1,688	2,049
	TVF/TVV	0,932	0,861	0,718	0,743	0,582
	TPF/TPV	0,378	0,533	0,555	0,541	0,466
	CVTemp	0,522	0,437	0,371	0,187	0,165
	CVTPV	0,393	0,243	0,256	0,195	0,159
	CVTPF	0,794	0,108	0,155	0,096	0,023
Métrica Pareto	MediaAM	16,405	49,027	521,750	591,943	1074,800
	MediaHV	9,247	28,004	373,216	401,272	763,379
	MediaMS	8,038	23,228	153,745	202,166	320,681
	MediaSP	8,038	2,205	5,210	11,495	9,220
	DesvPadAM	0,382	1,464	33,626	16,086	22,177
	DesvPadHV	0,184	1,459	33,553	15,262	21,324
	DesvPadMS	0,026	0,056	0,218	0,072	0,164
	DesvPadSP	0,288	0,551	0,509	1,284	1,576
	CVAM	0,023	0,030	0,064	0,027	0,021
	CVHV	0,020	0,052	0,090	0,038	0,028
	CVMS	0,003	0,002	0,001	0,000	0,001
	CVSP	0,036	0,250	0,098	0,112	0,171

planejamento é possível contornar os desafios relativos ao tempo. Como exemplos, pode-se criar estoques intermediários, adequar acordos de nível de serviço, otimizar processos de manutenção e também de operação. Além disso, um aspecto importante em relação à construção de novas linhas é o esforço necessário para se conseguir licenças ambientais e sociais para construção e operação. Por vezes é necessário fazer o traçado por áreas de empréstimo, áreas de proteção ambiental, entre outros casos. Esses fatores podem levar a custos proibitivos ou a condições sociais e ambientais que inviabilizem o projeto.

Para o entendimento dos indicadores será tomada como referência a instância *274v332a*. Na relação entre *TVF/TVI* é possível verificar o quanto foi necessário expandir o tempo de viagem para que todos os conflitos entre as rotas pudessem ser suprimidos. Foi necessário um aumento de 69% do tempo de viagem em relação ao melhor tempo possível, *TVI*. E, para verificar a eficiência do algoritmo de redução dos tempos de viagem, foi avaliada a relação *TVF/TVV*. Pode ser verificado que ocorreu uma redução de 26% de *TVF* em relação a *TVV*. Além dessa redução, foi observado que *TPF/TPV* revela uma redução de 46% de *TPF* quando comparado a *TPV*. Com esses resultados fica explícita a

eficácia do processo de redução de tempos de parada. Análises similares podem ser feitas para as outras instâncias.

Quando se analisa os coeficientes de variação referentes ao tempo de processamento das instâncias, descritos por *CVTemp*, pode-se observar que para a primeira instância ele tem um valor muito elevado, mas esse valor reduz bastante para as próximas instâncias. Esse comportamento revela que para as instâncias maiores o processo se torna muito mais estável já que o valor do desvio padrão reduz bastante em relação ao tempo de processamento da solução. O mesmo ocorre para o parâmetro *CVTPV*. E, quando se compara *CVTPV* com *CVTPF*, observa-se que para o último todos os resultados são bem melhores. A única exceção ocorre para a primeira instância, *49v57a*, em que o valor de *CVTPF* é mais que o dobro do valor de *CVTPV*, demonstrando uma dispersão muito elevada em relação aos tempos de solução. O restante dos coeficientes de variação apresentam valores considerados bons.

Em relação ao valor de CV, de acordo com as faixas sugeridas por Gomes (1987), pode ser observado no grupo *Métrica Pareto* da Tabela 5 que 85% dos coeficientes de variação têm dispersão baixa, 10% têm dispersão média e apenas 5% têm dispersão alta. Na tabela pode ser verificado que os CV com dispersão média e alta são relativos à métrica *spacing* das instâncias *94v112a* ($CVSP = 25\%$), *274v332a* ($CVSP = 11\%$) e *364v442a* ($CVSP = 17\%$). Os efeitos dessa métrica podem ser verificados na Figura 28a, página 86. Em algumas regiões da Fronteira de Pareto mostrada nessas figuras é possível observar regiões de grande concentração de soluções, formando alguns agrupamentos bem distintos. Em contraste, outras regiões apresentam soluções bem afastadas umas das outras.

A Figura 24 mostra alguns indicadores para as instâncias analisadas. É importante ressaltar que as instâncias foram ordenadas da menor para a maior em relação ao número de vértices. Dentre os indicadores são mostrados: tempo de processamento *Temp*, *Temp/Vértice*, *Temp/Trem* e *Temp/Conf*.

Na Figura 24a é mostrado o tempo de processamento até a convergência do algoritmo para a solução final. O gráfico de barras mostra o tempo médio de processamento para cada uma das cinco instâncias testadas. A linha pontilhada representa a tendência que responde segundo a expressão $y = 0,0002x^{8,3989}$, onde y é o tempo em horas. A Equação 6.1 descreve como é encontrado o valor da variável x. Ela é utilizada em todas as curvas de tendência. A exceção de uso da Equação 6.1 ocorre para a instância *49v57a*, onde $x = 1$.

$$x = 2 + \frac{(n_i - n_2)}{90} \tag{6.1}$$

onde n é o número de vértices do grafo e o valor de i corresponde ao número da instância, onde as instâncias crescem à taxa definida pelo grau médio do grafo. Logo, i é inteiro e maior do que 1. A instância *94v112a* é definida como sendo a de número 2.

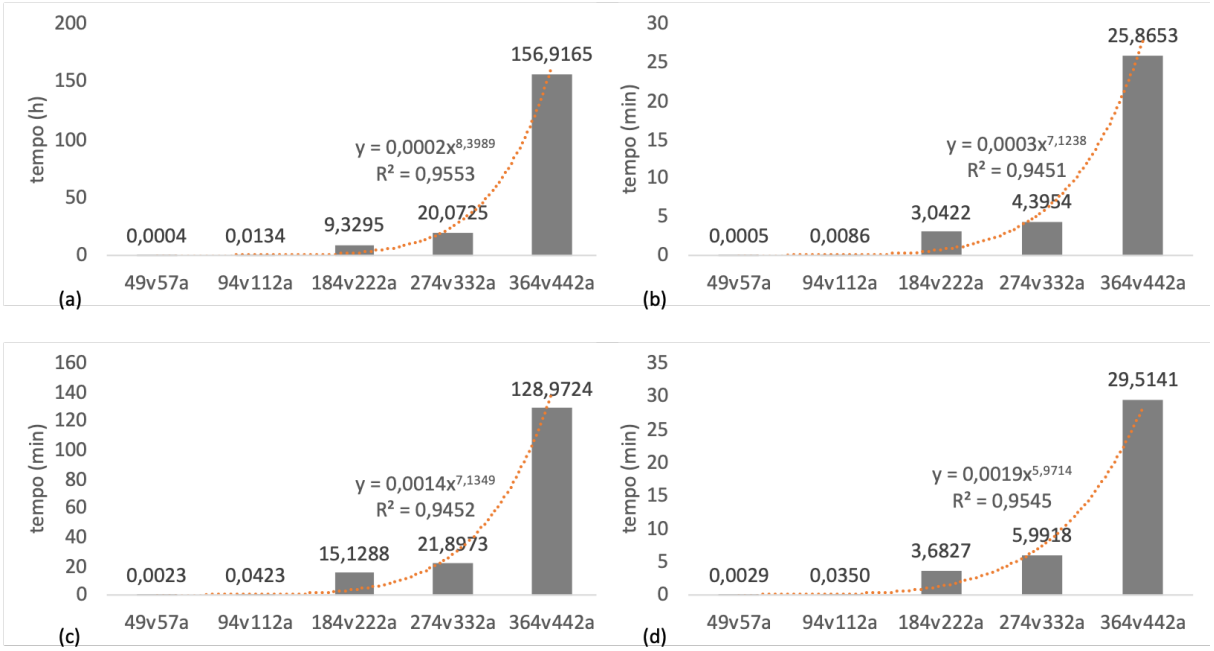

Figura 24 – Tempo de processamento do MO-SVNS e indicadores para as 5 instâncias de teste: (a) *Temp*, (b) *Temp/Vértice*, (c) *Temp/Trem* e (d) *Temp/Conf.*

Fonte: O autor

Como pode ser observado pelo valor do coeficiente de determinação, também chamado de R^2, o ajuste obtido pela curva de tendência foi de aproximadamente 96%. Logo, tomando como base a curva de tendência obtida, é possível estimar o tempo de processamento para se chegar à solução do problema em outras redes geradas segundo os mesmos critérios. Esses números são importantes para que trabalhos futuros possam ser realizados com o objetivo de melhorar os resultados apresentados nesta pesquisa. Em relação aos outros indicadores, Figuras 24b, 24c e 24d, embora a expressão que descreva cada uma das linhas de tendência seja diferente, o ajuste à curva em relação ao valor de R^2 foi de 95% em todos os casos. A expressão referente a cada uma das curvas de tendência pode ser vista diretamente em seus respectivos gráficos.

Como pode ser observado na Figura 24a, o tempo de processamento aumenta razoavelmente para a instância *364v442a*. Em alguns experimentos esse resultado chegou a 177 horas para essa mesma instância. Após o ajuste de alguns parâmetros do MO-SVNS o tempo obtido foi de 157 horas, isto é, foi obtida uma melhoria da ordem de 11%. Ainda assim, trata-se de um tempo muito elevado. Esse é um sinal claro de que outras técnicas de otimização precisam ser combinadas na busca de melhores resultados em um tempo computacional reduzido. Essas considerações são válidas para o ambiente de testes utilizado nesta pesquisa, conforme mencionado no Apêndice A. É importante lembrar que resultados ainda melhores podem ser obtidos se for utilizada uma linguagem de programação como C/C++ e além disso for utilizado um computador com mais recursos de processamento e

de memória RAM.

As Figuras 25 e 26 serão tomadas como base para o entendimento dos gráficos de saída dados pelo método de solução proposto nesta pesquisa. Cada uma dessas figuras se refere a somente uma rodada. No Apêndice B podem ser encontrados todos os resultados gerados ao longo dos experimentos. Eles não foram colocados no texto principal para que as análises fossem mais direcionadas.

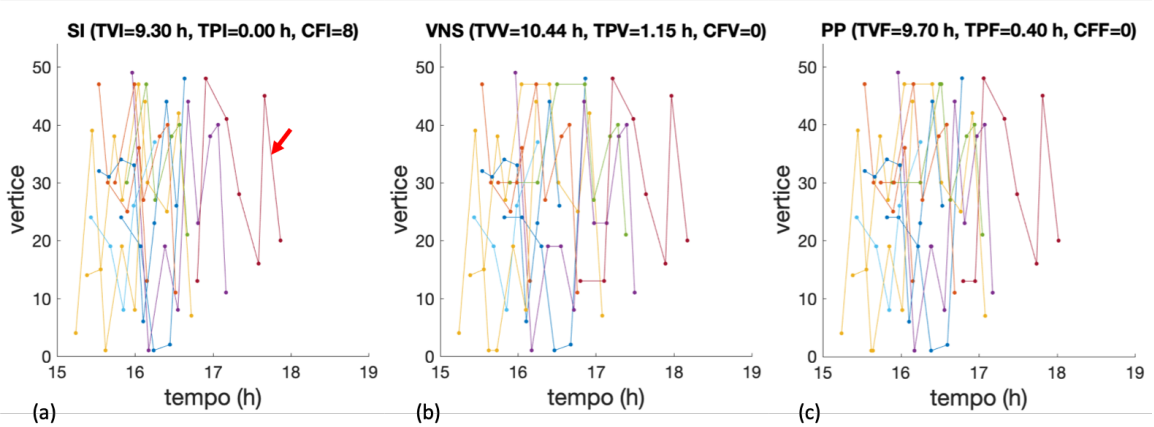

Figura 25 – Supressão de conflitos para a instância *49v57a*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

Fonte: O autor

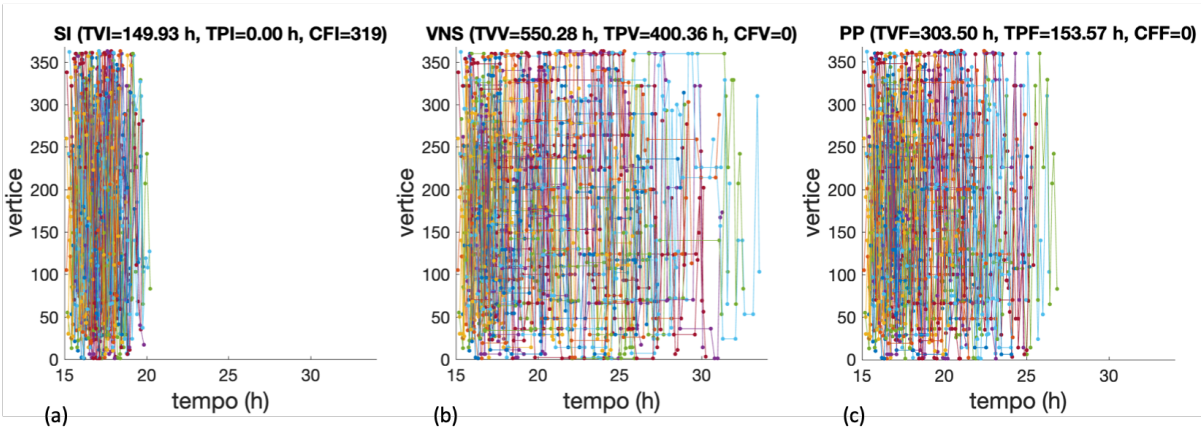

Figura 26 – Supressão de conflitos para a instância *364v442a*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

Fonte: O autor

Na Figura 25 são mostradas as informações relativas ao tempo de viagem, tempo de parada e número de conflitos. A Figura 25a mostra a solução inicial. Na Figura 25b é mostrada a solução dada pelo MO-SVNS. Por fim, Figura 25c, é mostrada a solução final dada pela redução dos tempos de viagem. Os eixos foram padronizados entre elas para

facilitar a análise dos efeitos sofridos pela programação dos trens em cada uma das etapas. O eixo horizontal mostra o tempo, dado em horas, e o eixo vertical mostra os vértices pelos quais passa a rota de cada um dos trens. O eixo do tempo evolui de modo contínuo, isto é, ele não é segmentado a cada 24 horas para separar os dias. Os vértices são representados por números inteiros. Por se tratar de grafos aleatórios, a rota de um trem pode passar em vértices que não estejam em uma sequência numérica necessariamente crescente ou decrescente. As mesmas considerações são aplicáveis à Figura 26.

Tomando como exemplo a Figura 25, o trem de número 7 da instância *49v57a* tem rota $R(13; 48; 41; 28; 16; 45; 20)$, onde o vértice de origem é o de número 13 e o de destino é o de número 20. Na Figura 25a há uma seta que destaca a rota em questão. Quanto ao tempo de viagem, ele é calculado pela soma dos tempos de deslocamento e de parada de cada um dos trens. Os tempos de parada também somam todas as paradas realizadas de todos os trens em circulação. Em ambos os casos são ignorados os tempos de parada no início e no final das rotas. O número de conflitos, por sua vez, é igual à soma de todos os conflitos que ocorrem entre os trens em circulação.

Na Figura 25a o tempo de viagem mostrado é o melhor que se pode ter já que nenhuma atraso é atribuído a qualquer um dos trens, isto é, os trens trafegam do início ao fim de sua rota às velocidades máximas atribuídas a cada um dos trechos. Na Figura 25b, com o intuito de eliminar todos os conflitos, são atribuídos tempos de parada aos trens com base nas estruturas de vizinhança utilizadas pelo algoritmo MO-SVNS. Quando o processo de otimização é finalizado o que se espera é que o tempo de viagem tenha sofrido um aumento, o tempo de parada tenha um valor maior do que zero e que todos os conflitos tenham sido eliminados. Na figura em questão, os tempos de parada são identificados pelas linhas paralelas ao eixo do tempo. Já na Figura 25c, são mostrados os resultados obtidos após a conclusão da rotina de redução dos tempos de parada.

Quando os resultados apresentados nas Figuras 25a e 25b são comparados, pode ser observado que em relação ao tempo de viagem ocorre um aumento de 12% após a otimização pelo MO-SVNS. Contudo, quando os resultados da Figura 25c são comparados aos da Figura 25b, pode ser verificado que ocorre uma redução de 7% em relação ao tempo de viagem mostrado na Figura 25b. Ao final do processo de otimização, Figura 25c, ocorre um aumento de apenas 4% em relação ao tempo de viagem inicial. Em relação ao tempo de parada, quando se compara os resultados apresentados nas Figuras 25b e 25c, é possível verificar que o algoritmo de redução de tempos de atraso foi capaz de promover uma redução de 65%, conseguindo um ótimo resultado frente a um dos objetivos que se deseja minimizar nesta pesquisa. As mesmas análises realizadas para a instância *49v57a* podem ser feitas para as outras instâncias.

Quanto à atuação do algoritmo MO-SVNS e do algoritmo de redução de tempos, uma análise rápida das Figuras 26a, 26b e 26c deixa clara a atuação de cada um deles.

Como na Figura 26a não há qualquer atraso para os trens em circulação, é possível perceber que as rotas começam aproximadamente às 15h e são finalizadas por volta das 20h. Quando a solução inicial, Figura 26a, é passada como entrada para o algoritmo MO-SVNS, Figura 26b, ocorre o acréscimo de muitos atrasos para todos os trens para possibilitar que as rotas sejam desembaralhadas e os trens consigam trafegar sem que haja qualquer conflito. É possível notar que os últimos trens concluem sua rota aproximadamente às 33h. Conforme já explicado anteriormente nesta pesquisa, é sabido que muitos dos atrasos podem ser eliminados ou minimizados. Então, quando a solução dada pelo MO-SVNS, Figura 26b, é dada como entrada para o algoritmo de redução de tempos é notável que a rota do último trem termina por volta das 27h.

A Figura 27 mostra o percentual de redução dos tempos de viagem para todas as instâncias. Esses percentuais foram calculados pela relação dos tempos médios de viagem após a etapa de redução de tempos de parada e a etapa de processamento do MO-SVNS. É notável que a eficiência do algoritmo de redução de tempos de viagem aumenta significativamente à medida em que as instâncias crescem. Essa eficiência no processo de redução de tempos tem como um dos principais motivos o fato de que quanto maiores as redes e o número de trens, maior é a necessidade de criar oportunidades de ultrapassagem. Essas oportunidades são expressas pelos tempos de parada. Se esses tempos sofrem qualquer aumento, isso reflete também em aumentos em relação aos tempos de viagem. Contudo, assim que o processo do MO-SVNS termina, muitas dessas paradas podem ser eliminadas sem qualquer prejuízo para a qualidade do planejamento dos trens.

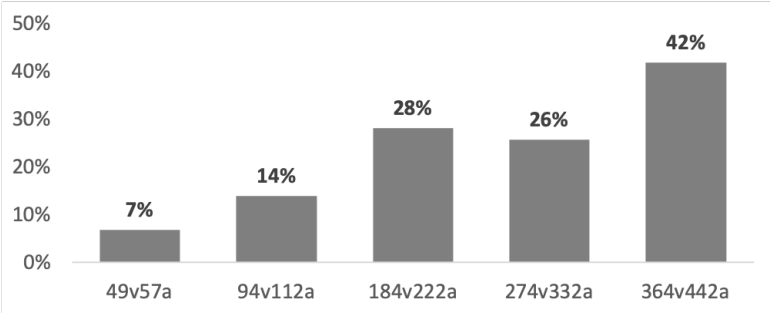

Figura 27 – Resultados atingidos com o algoritmo de redução de tempos de viagem em relação à solução encontrada pelo MO-SVNS.

Fonte: O autor

Na Figura 28 é mostrada tanto a Fronteira de Pareto quanto a evolução das métricas para avaliação da sua qualidade. Os gráficos são referentes à instância *364v442a*. É possível notar na Figura 28a que nas regiões onde a fronteira é mais próxima de um dos eixos o espaçamento entre as soluções tende a não ser tão uniforme. Esse fato normalmente ocorre porque na primeira perturbação do MO-SVNS muitos conflitos tendem a ser eliminados. Depois dessa etapa começa a busca pelos movimentos não tão óbvios, o que

faz com que o tempo de busca seja mais uniforme entre as eliminações de conflitos. E, por fim, para eliminar os últimos conflitos existentes o MO-SVNS entra em uma etapa de análise combinatória mais complexa já que pequenos movimentos podem trazer grandes perturbações para os resultados obtidos até o momento. Em meio à busca pela eliminação dos conflitos, o tempo médio de parada dos trens também é verificado para que essa dimensão também seja otimizada a contento.

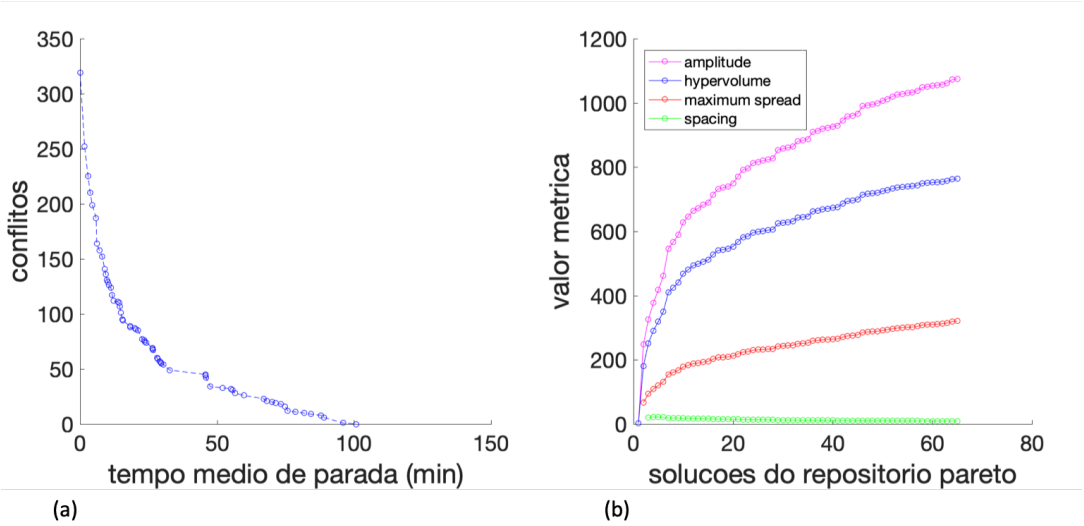

Figura 28 – Resultados para *364v442a*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

Fonte: O autor

Como o processo de otimização pode tomar muito tempo para eliminar os últimos conflitos, na prática ele poderia ser interrompido para que um operador de tráfego de trens pudesse fazer manualmente a eliminação dos últimos conflitos. Essa seria uma ação de sobreposição ao sistema computadorizado de suporte à decisão, comentado no Capítulo 1, página 20. Se as ações do operador forem assistidas e armazenadas em um banco de dados, elas podem ser utilizadas para a construção de um modelo baseado em aprendizado de máquina. Posteriormente esse modelo poderia ser incorporado ao algoritmo do MO-SVNS. Desse modo, sempre que o tempo entre a obtenção de ótimos locais se tornasse maior que um limite pré-estabelecido, o algoritmo de aprendizado de máquina seria utilizado para definir um movimento adequado a fim de solucionar os últimos conflitos.

6.5 Avaliação do Melhor Cenário

Tendo verificado o desempenho do planejamento de trens realizado pelo método de otimização implementado, foi escolhida a instância *94v112a* para avaliar todos os doze cenários propostos. O principal objetivo é verificar os resultados obtidos em cenários que utilizam grafos com vértices ponderados, grafos com peso unitário nos vértices e a

aplicação da métrica *betweenness* a ambos os casos anteriores. Além desses cenários, será verificada a solução obtida quando é definido um tempo limite para que a solução seja encontrada. Foram elaboradas duas versões para o caso em que é estabelecido um tempo limite. Na primeira delas todos os trens são mantidos mesmo havendo conflitos ao final do tempo estabelecido e na segunda é removido o número de trens necessário para que todos os conflitos sejam eliminados. A instância escolhida tem tempo médio de solução de 20 segundos. Quanto aos pesos dos vértices dos grafos ponderados, eles foram obtidos conforme mencionado no Capítulo 3.

Um ponto importante sobre a análise dos cenários *Cen–1* a *Cen–12* é o fato de que em alguns deles é utilizada a métrica *betweenness*. Seu uso possibilita uma melhor distribuição das rotas ao longo da rede, evitando que muitos trens passem pelos mesmos trechos gerando grandes congestionamentos. Com o uso dessa métrica podem ser utilizados tanto os grafos ponderados quanto aqueles cujos vértices têm peso unitário. A Tabela 3, página 76, contém os parâmetros utilizados em cada um desses cenários.

A Tabela 6 mostra os resultados obtidos para os cenários *Cen–1* a *Cen–8*. Nesses cenários todos os conflitos foram eliminados com sucesso. Na Tabela 7 são apresentados os resultados para os cenários *Cen–9* a *Cen–12*.

Tabela 6 – Parâmetros calculados para os cenários *Cen–1* a *Cen–8*.

Grupo	Parâmetro	Cen–1	Cen–2	Cen–3	Cen–4	Cen–5	Cen–6	Cen–7	Cen–8
Entrada	Rotas	1	1	3	3	1	1	3	3
	Pond/Btw	Pond	Pond	Pond	Pond	Btw	Btw	Btw	Btw
	Pond/Unit	Pond	Unit	Pond	Unit	Pond	Unit	Pond	Unit
	Vértices	94	94	94	94	94	94	94	94
	Arestas	112	112	112	112	112	112	112	112
	Trens	19	19	19	19	19	19	19	19
	Trechos	138	137	149	152	171	156	174	169
Densidade Conflitos	Conf	23	23	21	15	26	24	18	18
	Conf/Trem	1,211	1,211	1,105	0,789	1,368	1,263	0,947	0,947
	Conf/Trechos	0,167	0,168	0,141	0,099	0,152	0,154	0,103	0,107
Tempo	Temp (h)	31,405	29,908	23,671	27,162	45,787	20,339	43,307	23,502
	TVI (h)	22,530	22,250	24,790	25,010	28,820	25,940	29,450	28,270
	TVF (h)	26,972	27,070	30,426	29,920	34,264	28,434	34,782	31,088
	TPF (h)	4,440	4,814	5,636	4,908	5,442	2,498	5,214	2,810
Desvio Padrão	DesvPadTemp (h)	11,777	12,584	10,470	9,409	18,325	5,772	26,681	20,776
	DesvPadTVF (h)	1,492	0,726	0,545	1,316	1,481	0,706	0,949	0,885
	DesvPadTPF (h)	1,495	0,721	0,545	1,315	1,481	0,704	0,748	0,884
Indicador	ConfRest	0	0	0	0	0	0	0	0
	Temp/Vértice (min)	0,334	0,318	0,252	0,289	0,487	0,216	0,461	0,250
	Temp/Trem (min)	1,653	1,574	1,246	1,430	2,410	1,070	2,279	1,237
	Temp/Conf (min)	1,365	1,300	1,127	1,811	1,761	0,847	2,406	1,306
	TVF/TVI	1,197	1,217	1,227	1,196	1,189	1,096	1,181	1,100

6.5.1 Comparativo dos cenários *Cen–1* e *Cen–2*

Nos cenários *Cen–1* e *Cen–2* o cálculo das rotas levou em conta somente uma rota por trem. Trata-se de um problema predominantemente de planejamento de trens. No cenário *Cen–1* foi utilizado grafo ponderado e no cenário *Cen–2* foi utilizado grafo com peso unitário para todos os vértices. Para o cenário *Cen–2* foi observada uma redução

Tabela 7 – Parâmetros calculados para os cenários *Cen–9* a *Cen–12*.

Grupo	Parâmetro	Cen–9	Cen–10	Cen–11	Cen–12
Entrada	Rotas	3	3	3	3
	Pond/Btw	Pond	Pond	Btw	Btw
	Pond/Unit	Pond	Pond	Pond	Pond
	TempoLimite (s)	10	10	10	10
	Manter Trens	Sim	Não	Sim	Não
	Vértices	94	94	94	94
	Arestas	112	112	112	112
	Trens Início	19	19	19	19
	Trens Final	19	16	19	13
Densidade Conflitos	Conf	21	21	18	18
	Conf/Trem	1,105	1,105	0,947	0,947
	Conf/Trechos	0,141	0,164	0,103	0,133
Tempo	Temp (h)	12,571	12,377	13,345	11,886
	TVI (h)	24,790	19,936	29,450	21,474
	TVF (h)	27,270	21,894	32,502	23,038
	TPF (h)	2,480	1,956	3,056	1,566
Desvio Padrão	DesvPadTemp (h)	0,465	0,785	0,690	0,904
	DesvPadTVF (h)	1,042	2,796	0,671	2,406
	DesvPadTPF (h)	1,042	0,676	0,668	0,703
Indicador	ConfRest	5	0	5	0
	Temp/Vértice (min)	0,134	0,132	0,142	0,126
	Temp/Trem (min)	0,662	0,651	0,702	0,626
	Temp/Conf (min)	0,599	0,589	0,741	0,660
	TVF/TVI	1,100	1,098	1,104	1,073

de 5% em relação ao tempo médio de cálculo da solução. Essa redução se dá pelo fato de que quando os pesos são iguais o algoritmo KSP calcula a menor rota pelo menor número de saltos da origem ao destino para cada um dos trens. Com menos saltos há também menores possibilidades de ocorrerem muitas paradas de trens, fato que contribui para que o tempo de busca seja menor. Nesse caso, podem ocorrer menos intervalos de parada, embora os tempos dessas paradas possam ser maiores. Quanto ao parâmetro *TVF*, em ambos os cenários o valor foi bastante similar, não chegando a uma diferença de 1%. Já em relação ao parâmetro *TPF*, o valor encontrado no cenário *Cen–2* foi 8% maior. Quanto ao desvio padrão, para ambas as variáveis, *TVF* e *TPF*, foi observada uma redução média de 52% no cenário *Cen–2*.

6.5.2 Comparativo dos cenários *Cen–3* e *Cen–4*

Nos testes seguintes, cenários *Cen–3* e *Cen–4*, foram feitos ajustes similares àqueles realizados nos cenários *Cen–1* e *Cen–2*, respectivamente. A diferença aqui é que foram calculadas três rotas para cada um dos trens e então escolhida a melhor delas para prosseguir no processo de busca da solução. Nesse caso foram mais explorados os aspectos do roteamento dos trens. As rotas foram adicionadas uma a uma para cada um dos trens. Em todos os caso foi escolhida como melhor rota aquela que gerava o menor número de conflitos dentre os trens cujas rotas já haviam sido selecionadas nas etapas anteriores. Na comparação com o cenário *Cen–3*, o tempo de solução do cenário *Cen–4* aumentou em 15%. Esse acréscimo é decorrente do tempo que se leva para escolher a melhor rota para cada um dos trens. E quando analisados os parâmetros *TVF* e *TPF*, foi observado

que, para o cenário *Cen–4*, eles sofreram uma redução de 2% e de 13%, respectivamente. Essa redução indica uma eficiência maior do algoritmo de busca já que a solução inicial pode começar com rotas que acarretem em um número menor de conflitos. Em relação aos parâmetros *DesvPadTVF* e *DesvPadTPF*, foi observado que no cenário *Cen–4* ocorreu um aumento de 141% para ambas as variáveis.

6.5.3 Comparativo dos cenários *Cen–1* e *Cen–3*

Quando se compara os cenários *Cen–1* e *Cen–3* observa-se que no cenário com mais rotas, *Cen–3*, o tempo de processamento sofre uma redução de 7% e o desvio padrão reduz em 11%, tornando esse cenário mais rápido e mais estável. O tempo de processamento depende mais da escolha da melhor rota para cada um dos trens do que do tempo utilizado pelo algoritmo KSP. Em seguida, para o *Cen–3*, foi observado um aumento de 13% em relação a *TVF* e de 27% em relação a *TPF*. De modo similar, na comparação dos cenários *Cen–2* e *Cen–4*, foi observado que o tempo de solução para o cenário *Cen–4* sofreu uma redução de 9% e o desvio padrão sofreu uma redução de 25%. Contudo, *TVF* e *TPF* sofreram um aumento respectivo de 11% e de 2%. Sendo assim, pode-se concluir que embora o tempo de cálculo da solução seja menor e o processo seja mais estável com o uso de mais opções de rota, por fim as soluções encontradas não são tão boas quando comparadas àquelas obtidas com apenas uma opção de rota para o planejamento de cada um dos trens.

6.5.4 Comparativo dos cenários *Cen–5* e *Cen–6*

Os cenários *Cen–5* e *Cen–6* são similares aos cenários *Cen–1* e *Cen–2*, respectivamente. A diferença é que nesses novos cenários é utilizada a métrica *betweenness*. Algo notável é o número de trechos nesses cenários, que já era esperado que fosse maior ao utilizar essa métrica. Ao longo dos experimentos foi observado que ao calcular rotas utilizando essa métrica frequentemente ocorrem mais saltos. Na comparação dos cenários *Cen–5* e *Cen–6* com os cenários *Cen–1* e *Cen–2* o número médio de trechos aumentou 24% e 14%, respectivamente.

Comparando os cenários *Cen–5* e *Cen–6* observa-se que todos os tempos e desvios padrão tiveram seus valores reduzidos no cenário *Cen–6*. O tempo de cálculo da solução sofreu uma redução de 56%, *TVF* sofreu uma redução de 17% e *TPF* sofreu uma redução de 54%. Quanto ao desvio padrão, *desvPadTemp* sofreu uma redução de 69% enquanto *desvPadTVF* e *desvPadTPF* sofreram uma redução de 52%. Para o caso de apenas uma rota disponível para cada trem, fica claro que o uso da métrica *betweenness ponderada*, cenário *Cen–5*, não traz vantagens.

Outra comparação que pode ser feita é entre os cenários *Cen–1* e *Cen–5*, com a

diferença de que no cenário *Cen–1* não foi utilizada a métrica *betweenness*. Mais uma vez o uso dessa métrica não foi vantajoso. Por outro lado, quando comparados os cenários *Cen–2* e *Cen–6*, que utilizam grafos com peso unitário em seus vértices, o uso da métrica *betweenness* mostra-se bastante vantajoso. No cenário *Cen–6* foi observada uma redução de 32% em relação ao tempo de cálculo da solução com uma redução de 54% em relação ao desvio padrão. Embora *TVF* tenha aumentado em 5% no cenário *Cen–6*, o valor de *TPF* apresentou uma redução de 48%, que denota uma condição operacional mais favorável já que a redução de *TPF* implica em uma redução do tempo de viagem dos trens.

Além dos resultados supracitados, foi observado que utilizando a métrica *betweenness* em geral o número de conflitos eliminados nas primeiras iterações é maior do que o número de eliminações realizadas sem sua utilização. Esses resultados foram verificados na Fronteira de Pareto dos quatro cenários comparados. Além disso, foi observado que no cenário *Cen–6* o tempo médio de parada foi menor do que nos outros três cenários, demonstrando uma condição melhor para a aplicação do método de solução proposto. Esses resultados podem ser observados nas Fronteiras de Pareto mostradas na Figura 29.

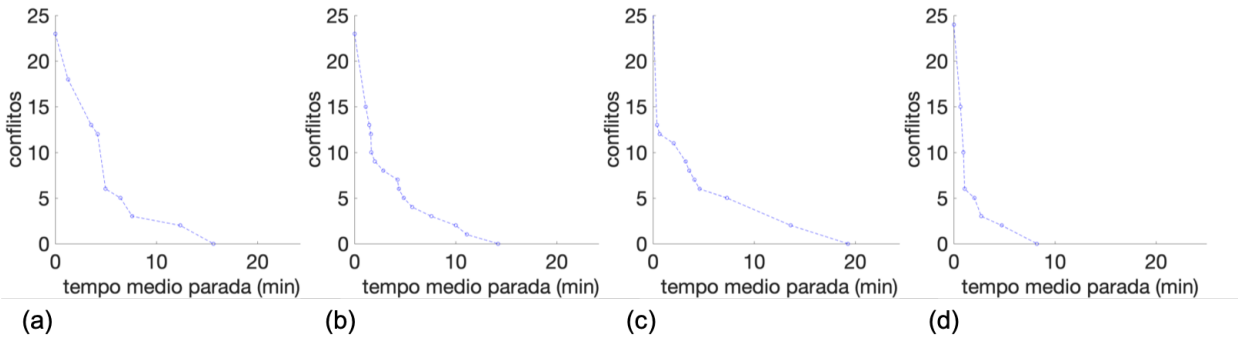

Figura 29 – Comparativo da Fronteira de Pareto utilizando ou não a métrica *betweenness*: (a) grafo ponderado (*Cen–1*), (b) grafo com pesos unitários (*Cen–2*), (c) grafo ponderado e *betweenness* (*Cen–5*) e (d) grafo com pesos unitários e *betweenness* (*Cen–6*).

Fonte: O autor

6.5.5 Comparativo dos cenários *Cen–7* e *Cen–8*

Os próximos cenários, *Cen–7* e *Cen–8*, são similares aos cenários *Cen–3* e *Cen–4*, respectivamente. Nesses cenários também foi utilizada a métrica *betweenness* e três opções de rota para cada trem. Foi observado que ocorreu um aumento do número de trechos nos cenários que utilizaram a métrica *betweenness*, assim como ocorreu nos cenários *Cen–5* e *Cen–6* vistos anteriormente.

Comparando os cenários *Cen–7* e *Cen–8* pode-se verificar que os resultados foram parecidos com aqueles obtidos na comparação entre os cenários *Cen–5* e *Cen–6*, isto

é, o uso da métrica *betweenness* no grafo cujos pesos dos vértices são unitários, *Cen–8*, apresentou diversas reduções nos parâmetros de tempo. O tempo de solução reduziu 46% e seu desvio padrão reduziu 22%. Os valores de *TVF* e *TPF* sofreram uma redução de 11% e de 46%, respectivamente. Mais uma vez o uso da métrica *betweenness ponderada* não trouxe vantagens, assim como foi observado na comparação entre os cenários *Cen–5* e *Cen–6*.

Outra comparação importante ocorre entre os cenários *Cen–3* e *Cen–7*, sendo que no cenário *Cen–3* não foi utilizada a métrica *betweenness*. Observa-se que o uso da métrica *betweenness ponderada* não traz muitas vantagens no cenário *Cen–7*, acarretando no aumento de 49% do tempo de solução e no aumento de 14% do parâmetro *TVF*, embora o valor de *TPF* tenha sofrido uma redução de 7%. Em contrapartida, ao comparar os cenários *Cen–4* e *Cen–8* verifica-se que no *Cen–8* o tempo de solução sofreu uma redução de 13% e *TPF* sofreu uma redução bastante significativa de 43%. Foi observado também que *TVF* do cenário *Cen–8* sofreu um aumento de 4% em relação ao cenário *Cen–4*. Em linhas gerais isso indica que os resultados obtidos com o uso da métrica *betweenness ponderada* foi melhor no cenário *Cen–8*.

6.5.6 Comparativo dos cenários *Cen–9* a *Cen–12*

Os últimos cenários, *Cen–9* a *Cen–12*, utilizam a opção de limitar o tempo de processamento da solução. O tempo foi ajustado em 10 segundos para interromper a rotina do MO-SVNS. É importante ressaltar que o tempo real de execução poderá ser maior do que a referência de 10 segundos. Isso se deve ao fato de que a avaliação do tempo ocorre ao final do algoritmo MO-SVNS, isto é, após a execução de todos os procedimentos intermediários. Para mais detalhes consulte o Algoritmo 6, página 52. Vale lembrar que o tempo de solução dessa instância é de aproximadamente 20 segundos.

Nos cenários *Cen–9* e *Cen–10* é utilizado o grafo ponderado e nos cenários *Cen–11* e *Cen–12* é utilizado o grafo com peso unitário nos vértices. Nos cenários *Cen–9* e *Cen–11* são mantidos todos os trens mesmo que haja algum conflito após o tempo estipulado. Por outro lado, nos cenários *Cen–10* e *Cen–12*, são eliminados quantos trens forem necessários até que todos os conflitos sejam eliminados ao término do tempo.

Quando os cenários *Cen–9* e *Cen–10* são comparados, observa-se que no *Cen–10* ocorre uma redução de 2% do tempo de cálculo da solução. Quando se compara os parâmetros *TVF* e *TPF* observa-se que no cenário *Cen–10* ocorre uma redução média de 20% para ambas as variáveis quando os valores são comparados àqueles obtidos no cenário *9*. É importante ressaltar que no cenário *Cen–10* foram eliminados 16% dos trens em relação ao cenário *Cen–9*. Quando alguns trens são eliminados os parâmetros *TVF* e *TPF* são diretamente impactados.

Na comparação dos cenários *Cen–11* e *Cen–12* observa-se que o tempo de busca pela solução sofre uma redução de 11% no cenário *Cen–12*. Em relação aos parâmetros *TVF* e *TPF*, o cenário *Cen–12* apresenta uma redução de 29% e de 49%, respectivamente. No cenário *Cen–12* foram eliminados 30% dos trens em relação ao cenário *Cen–11*.

Em relação ao número de conflitos restantes ao final do tempo, tanto no cenário *Cen–9* quanto no cenário *Cen–11*, na média, restaram cinco conflitos, que para o cenário *Cen–9* corresponde a 24% do número inicial de conflitos e para o cenário *Cen–11* corresponde a 28%. Ainda na comparação desses dois cenários, verifica-se que é necessário um tempo 6% maior para se chegar à solução no cenário *Cen–11*. Além disso, o valor dos parâmetros *TVF* e *TPF* para o cenário *Cen–11* são também 19% e 23% maiores.

Ao comparar os cenários *Cen–10* e *Cen–12* pode-se verificar que o tempo de solução é 4% menor para o cenário *Cen–12*. O valor de *TVF* é apenas 5% maior no cenário *Cen–12* e o valor de *TPF* é 20% menor. Esses valores indicam uma solução aparentemente melhor ao utilizar a métrica *betweenness ponderada*, já que ao término do tempo o valor de *TPF* é menor para o cenário *Cen–12*. Entretanto, no cenário *Cen–12* foi necessário eliminar quase o dobro do número de trens quando comparado ao cenário *Cen–10*. Além do mais, o cenário *Cen–12* já começou com 14% menos conflitos do que o cenário *Cen–10*.

Quando se leva em conta as análises anteriores pode-se concluir que o uso da métrica *betweenness ponderada* não apresentou vantagens na maior parte dos casos.

6.6 Adaptações do Método

Quando se compara as Fronteiras de Pareto obtidas nos cenários *Cen–1* a *Cen–8* pode ser observado que no cenário *Cen–8* foi obtida a melhor solução, tanto em relação à minimização do número de conflitos quanto em relação à minimização do tempo médio de parada dos trens em circulação. Motivado pelos resultados obtidos no *Cen–8*, serão realizados novos experimentos com todas as instâncias de teste. É válido destacar que nesse cenário são explorados tanto o planejamento quanto o roteamento de trens.

Ao longo dos testes realizados também foram observadas as estatísticas de uso das estruturas de vizinhança definidas na Tabela 2, página 60. Foi verificado que as vizinhanças mais relevantes ao longo do processo de otimização do MO-SVNS eram aquelas que adicionavam tempos de atraso ou faziam a permutação dos atrasos de diferentes trechos de uma rota. A relevância em questão diz respeito ao tempo de processamento da solução. Com base nessa análise foram elaboradas cinco novas estruturas de vizinhança para serem utilizadas no lugar das seis anteriores. Nessas vizinhanças tanto os trechos com atraso quanto os trens impactados foram escolhidos aleatoriamente. As novas estruturas de vizinhança podem ser vistas na Tabela 8.

Tabela 8 – Novas vizinhanças utilizadas pelo MO-SVNS.

Vizinhança	Descrição da vizinhança
Viz–1n	Adiciona tempo de parada a um dos trechos da rota de 10% dos trens em circulação.
Viz–2n	Adiciona tempo de parada a dois dos trechos da rota de 10% dos trens em circulação.
Viz–3n	Adiciona tempo de parada a três dos trechos da rota de 10% dos trens em circulação.
Viz–4n	Para a rota de 10% dos trens em circulação, escolhe um tempo de atraso, calcula 15% desse tempo e o soma à parada anterior ou ao trecho inicial da rota, caso seja permitido parar.
Viz–5n	Permuta atrasos entre dois trechos da rota de 10% dos trens em circulação.

Os resultados obtidos a partir do cenário *Cen–8* e das novas vizinhanças podem ser vistos na Tabela 9, que para fins de comparação traz os mesmos campos mostrados na Tabela 5, página 80. Assim como mencionado nas seções anteriores, os resultados completos podem ser vistos no Apêndice B.

Tabela 9 – Parâmetros calculados para cada topologia utilizando o *Cen–8*.

Grupo	Parâmetro	49v57a	94v112a	184v222a	274v332a	364v442a
Entrada	Vértices	49	94	184	274	364
	Arestas	57	112	222	332	442
	Trens	10	19	37	55	73
	Trechos	79	169	724	950	1373
Densidade Conflitos	Conf	5	18	234	88	337
	Conf/Trem	0,500	0,947	6,324	1,600	4,616
	Conf/Trechos	0,063	0,107	0,323	0,093	0,245
Tempo	Temp (h)	0,000	0,006	6,223	3,664	77,351
	TVI (h)	11,530	28,270	116,590	148,190	211,670
	TVV (h)	12,202	36,634	292,008	224,536	571,358
	TVF (h)	11,772	31,882	219,710	191,356	403,578
	TPV (h)	0,668	8,362	175,416	76,344	359,688
	TPF (h)	0,242	3,610	103,118	43,166	191,908
Desvio Padrão	DesvPadTemp (h)	0,000	0,002	0,448	0,251	17,751
	DesvPadTPV (h)	0,215	4,669	20,652	5,752	99,898
	DesvPadTPF (h)	0,317	0,970	13,063	4,062	34,166
Indicador	ConfRest	0	0	0	0	0
	Temp/Vértice (min)	0,000	0,004	2,029	0,802	12,750
	Temp/Trem (min)	0,001	0,018	10,091	3,997	63,576
	Temp/Conf (min)	0,002	0,019	1,596	2,498	13,772
	TVF/TVI	1,021	1,128	1,884	1,291	1,907
	TVF/TVV	0,965	0,870	0,752	0,852	0,706
	TPF/TPV	0,362	0,432	0,588	0,565	0,534
	CVTemp	0,260	0,354	0,072	0,069	0,229
	CVTPV	0,322	0,558	0,118	0,075	0,278
	CVTPF	1,311	0,269	0,127	0,094	0,178
Métrica Pareto	MediaAM	10,275	37,695	1023,900	227,470	1427,900
	MediaHV	6,043	21,449	794,857	141,964	1096,100
	MediaMS	5,022	18,189	236,807	88,780	339,624
	MediaSP	0,790	1,943	7,754	3,273	7,841
	DesvPadAM	0,479	0,806	82,501	6,584	148,776
	DesvPadHV	0,036	0,905	82,375	6,353	147,854
	DesvPadMS	0,026	0,051	0,355	0,076	0,467
	DesvPadSP	0,443	0,704	0,855	0,315	1,179
	CVAM	0,047	0,021	0,081	0,029	0,104
	CVHV	0,006	0,042	0,104	0,045	0,135
	CVMS	0,005	0,003	0,001	0,001	0,001
	CVSP	0,561	0,362	0,110	0,096	0,150

As análises comparativas entre os dados da Tabela 5, cenário *Cen–1*, e da Tabela 9, cenário *Cen–8*, serão feitas para alguns parâmetros considerados mais relevantes, a saber:

- **Trechos**: Com o uso da métrica *betweenness* a primeira mudança que pode ser

observada é em relação ao número de trechos utilizados para cada uma das instâncias. Foi verificado um aumento de 53% para a instância *184v222a*;

- **Conf**: Como são calculadas três opções de rota para cada trem em circulação, cenário *Cen–8*, via de regra há uma tendência de que a rota escolhida para cada um dos trens acarrete em um número menor de conflitos. Embora tenha sido percebido um aumento de 54% do número de conflitos para instância *184v222a*, para a instância *274v332a* foi verificada uma redução de 56%. E, quando todas as instâncias são consideradas, é verificado um aumento médio de 11%;

- **Temp**: Em relação ao tempo de solução foram atingidos ganhos bastante expressivos utilizando o cenário *Cen–8* e as novas estruturas de vizinhança. A maior redução foi observada na instância *274v332a*, sendo de 82%. Para as demais instâncias ocorreu uma redução média de 50% do tempo em relação ao cenário *Cen–1*;

- **TVF**: Comparando os valores obtidos para *TVF* foi observado que no cenário *Cen–8* as soluções não foram tão boas quanto aquelas obtidas no cenário *Cen–1*. Para a instância *184v222a* foi verificado um aumento de 56%. Para as outras instâncias foi obtido um aumento médio de 20%;

- **TPF**: Os valores obtidos para *TPF* apresentaram algumas melhorias no cenário *Cen–8*. Para quatro instâncias ocorreu uma redução média de 22%. Por outro lado, para a instância *184v222a* foi verificado um aumento de 50%;

- **CV**: Quanto ao coeficiente de variação das métricas utilizadas para avaliação da Fronteira de Pareto, foram verificadas algumas melhorias. No cenário *Cen–8* ocorreram dois casos em que o CV foi classificado como alto, cinco em que foi classificado como médio e o restante foi classificado como baixo. Sendo assim, foi observada uma piora para o CV em relação ao cenário *Cen–1*.

De acordo com as análises realizadas, o maior ganho obtido com a utilização do cenário *Cen–8* e as novas estruturas de vizinhança foi em relação ao tempo de solução das instâncias, que na média reduziu 56%. Outra redução importante ocorreu em relação ao número de conflitos. Por apresentarem uma redução relevante, média de 11%, por fim essa redução contribui para que os conflitos restantes possam ser suprimidos manualmente pelos operadores de tráfego de trens no caso do algoritmo ser interrompido por tempo de execução. No que tange ao parâmetro *TPF*, nos novos experimentos ocorreu uma redução média de 8%. Algumas dessas conclusões já haviam sido identificadas na Seção 6.5.5.

Com relação às Fronteiras de Pareto obtidas para o cenário *Cen–8*, foi observado que das cinco instâncias testadas somente três delas apresentaram resultados melhores. Os melhores resultados foram observados para as instâncias *49v57a*, *94v112a* e *274v332a*.

Para essas três instâncias um fato relevante é que o número inicial de conflitos foi inferior àquele observado ao utilizar o cenário *Cen–1*.

Com base nos resultados obtidos, há indícios de que quando é utilizada a métrica *betweenness* e é calculada mais de uma rota por trem, o algoritmo de seleção de rotas faz boas escolhas somente para as rotas dos primeiros trens. Como o número de trechos aumenta consideravelmente utilizando a métrica *betweenness*, as rotas dos últimos trens inseridos na programação acabam sendo penalizadas com um número muito alto de conflitos. Essa hipótese é sugerida pelo aumento médio de 11% em relação ao número de conflitos de todas as instâncias testadas, conforme visto anteriormente.

7 Conclusões e Trabalhos Futuros

Este trabalho teve como objetivo propor um método de solução para o problema de planejamento e roteamento de trens em ferrovias modeladas por redes aleatórias planares modeladas via grafos. Para cumprir esse objetivo foi criado um método de conversão de um trecho padrão de ferrovia para um modelo em grafo. Esse é um dos principais alicerces para o método proposto. Em seguida foram apresentadas informações sobre o planejamento de trens e sobre como seriam calculadas as rotas utilizadas para cada um dos trens, onde foram escolhidos os algoritmos de *Yen* e *Dijkstra* para encontrar os menores caminhos entre dois vértices do grafo. E, com o intuito de evitar grandes congestionamentos em alguns caminhos do grafo, foi testada a medida de centralidade *betweenness*. Essa métrica foi utilizada tanto com grafos com vértices ponderados quanto com pesos unitários. Uma vez calculadas as rotas dos trens, é necessário verificar e suprimir os conflitos existentes entre elas. Nesta etapa foi utilizada uma versão adaptada da meta-heurística *Skewed Variable Neighborhood Search*. A adaptação realizada permitiu trabalhar na minimização de dois objetivos: número de conflitos e tempo médio de parada dos trens em circulação. Inspirado em uma das referências bibliográficas, foi desenvolvida uma heurística para reduzir tempos de atraso dos trens a partir da solução dada pelo MO-SVNS. Por fim, os resultados da otimização biobjetivo foram avaliados pela Fronteira de Pareto, que apresenta um gráfico onde os dois objetivos são relacionados. A curva em questão separa as soluções dominadas das não dominadas, sendo que as soluções não dominadas correspondem às soluções ótimas de Pareto, isto é, às melhores relações entre os objetivos otimizados.

Como foram utilizados muitos cenários e instâncias ao longo dos testes, os resultados obtidos foram bastante interessantes. Foi possível avaliar a contribuição das etapas do método heurístico construído para resolver o problema proposto. Além disso, foi verificado o comportamento do MO-SVNS, etapa mais demorada do método, ao lidar com os diferentes cenários e estruturas de vizinhança disponíveis. Foi verificado pela Fronteira de Pareto que em alguns casos as soluções obtidas eram muito boas embora o tempo computacional fosse um pouco longo. Por outro lado, nos testes realizados com uma nova estrutura de vizinhança elaborada para ganhar velocidade no tempo de solução, foi observado que os resultados não foram tão bons para o objetivo relacionado ao tempo médio de parada dos trens. Desse modo, fica clara a necessidade de um ajuste mais fino tanto dos cenários quanto das estruturas de vizinhança utilizadas pelo MO-SVNS. Esses parâmetros devem ser avaliados e ajustados em função do que se deseja priorizar ao longo da solução: tempo de solução ou qualidade dos resultados obtidos.

Como o método proposto utilizou redes aleatórias, foram abstraídas particularidades físicas de ferrovias reais, fato que leva também à simplificação das regras comumente

aplicáveis ao tráfego de trens. Além disso, o uso de redes aleatórias permite o emprego do método desenvolvido em redes das mais diversas naturezas, incluindo, mas não se limitando às redes de telecomunicações, malhas rodoviárias e aéreas. Trata-se de um método heurístico que pode ser utilizado como ponto de partida para heurísticas aplicáveis a problemas com regras mais específicas de tráfego de entidades ao longo de uma rede.

O uso do MO-SVNS mostrou-se bastante adequado na solução do problema de otimização multiobjetivo envolvendo o planejamento e o roteamento de trens. Trata-se de uma meta-heurística relativamente simples para a obtenção de resultados bastante satisfatórios. E no que tange à Fronteira de Pareto, aqui cabe um destaque especial à métrica *Amplitude*, que foi proposta pelo autor para ajudar na escolha da melhor Fronteira de Pareto em problemas que não tenham correspondência exata na literatura, como é o caso deste estudo. A falta de correspondência também se deve ao fato de que as topologias das redes utilizadas não são conhecidas e estudadas pela comunidade científica.

Quanto aos resultados obtidos, é válido lembrar que ainda há muitas possibilidades de melhorá-los. É importante notar que propositalmente foram geradas condições de tráfego de trens que resultavam em muitos conflitos entre suas rotas. Isso foi feito porque este estudo propõe um método de solução para o problema de planejamento e roteamento de trens e por isso fazia parte do escopo da pesquisa realizar testes em condições mais extremas. Todavia, em ferrovias reais é comum encontrar regras bem definidas relativas ao espaçamento mínimo que deve haver entre os trens, fato este que pode tornar o processo de otimização bem mais rápido. A liberdade adotada para as condições gerais de tráfego de trens e dos cenários elaborados se valeu do fato desta pesquisa não ter sido focada em estudos comparativos com pesquisas similares.

Um dos pontos altos da solução do problema foi a implementação do algoritmo para reduzir os tempos de viagem. Seu uso pode ser replicado em todos os casos de roteamento onde exista a necessidade de que uma entidade permaneça parada para que outra utilize um determinado recurso da rede. Isso é possível porque os tempos desnecessários são eliminados somente após a garantia de que nenhum conflito será gerado ao remover qualquer atraso atribuído à rota de uma entidade que trafegue pela rede. Quanto ao recurso, ele pode ser entendido como sendo um trecho por onde só uma entidade poderia trafegar em um determinado intervalo de tempo.

Como trabalhos futuros são sugeridas algumas linhas de pesquisa que se propõem a aumentar o alcance dos resultados obtidos neste trabalho, a saber:

- Considerar outras funções objetivo como, por exemplo, o número de paradas;
- Avaliar o uso de diferentes estratégias de roteamento e também o uso de técnicas de roteamento adaptativo;

- Verificar a utilização de outras medidas de centralidade que possibilitem tanto evitar grandes congestionamentos quanto otimizar o uso da rede utilizada;
- Realizar estudo comparativo das instâncias utilizando meta-heurísticas amplamente encontradas na literatura, tais como *Non-dominated Sorting Genetic Algorithm II* (NSGA-II), *Multi-objective Ant Colony Optimization* (MOACO), *Artificial Bee Colony* (ABC), entre diversos outros algoritmos evolucionários aplicados a problemas multiobjetivo;
- Utilizar técnicas de aprendizado de máquina para auxiliar o algoritmo do MO-SVNS na eliminação mais rápida dos últimos conflitos já que nessa fase do processo de otimização ele se torna mais sensível a perturbações;
- Implementar uma solução para o problema em questão com base no algoritmo do *Job Shop Scheduling Problem*. Esse seria um grande desafio uma vez que algumas implementações encontradas na literatura se baseiam em casos muito particulares de planejamento e roteamento de trens;
- Eleger um trem prioritário e com rotas bem definidas, como é o caso do trem de passageiros. Dado esse cenário, realizar o roteamento e a programação de todos os outros trens levando em conta os dados do trem prioritário;
- Avaliar a ocorrência de interferências na programação dos trens ou mesmo na perda de conexões de rotas. Propor métodos para que o roteamento e a programação dos trens sejam atualizados a contento;
- Comparar resultados obtidos por programação linear e por outras meta-heurísticas com os resultados obtidos nesta pesquisa. Avaliar também o desempenho da métrica *amplitude* proposta neste trabalho e do método utilizado para a escolha da melhor Fronteira de Pareto.

Publicações

- Apresentação de pôster:

COSTA, H. D.; PAIVA, M. H. M.; ROCHA, H. R. d. O. Variable Neighborhood Search Aplicado ao Escalonamento e Roteamento de Trens. In: *Anais do* 49º *Simpósio Brasileiro de Pesquisa Operacional*. Blumenau, SC: SOBRAPO, 2017.

- Trabalho publicado:

COSTA, H. D.; PAIVA, M. H. M.; ROCHA, H. R. d. O. Supressão de Conflitos no Problema de Planejamento e Roteamento de Trens Utilizando a Metaheurística VNS. In: *Anais do* 50º *Simpósio Brasileiro de Pesquisa Operacional*. Rio de Janeiro, RJ: SOBRAPO, 2018.

- Trabalho aceito para publicação:

COSTA, H. D.; PAIVA, M. H. M.; ROCHA, H. R. d. O. Planejamento e Roteamento de Trens Utilizando a Meta-heurística VNS e Técnicas em Grafos. In: *Anais do* 51º *Simpósio Brasileiro de Pesquisa Operacional*. Limeira, SP: SOBRAPO, 2019.

Referências

ABDI, H. Coefficient of variation. *Encyclopedia of research design*, SAGE Publications, Inc.: Thousand Oaks, CA, USA, v. 1, p. 169–171, 2010. Citado na página 79.

ANTONIUCCI, G. A. *Analysis of the distribution of Pareto optimal solutions on various multi-objective evolutionary algorithms*. Dissertação (B.S. thesis) — Universitat Politècnica de Catalunya, 2016. Citado 2 vezes nas páginas 66 e 68.

BAGCHI, T. P. *Multiobjective scheduling by genetic algorithms*. [S.l.]: Springer Science & Business Media, 1999. Citado 2 vezes nas páginas 66 e 67.

BENJAMIN, A.; CHARTRAND, G.; ZHANG, P. *The fascinating world of graph theory*. [S.l.]: Princeton University Press, 2015. Citado na página 30.

BLAŻEWICZ, J.; PESCH, E.; STERNA, M. The disjunctive graph machine representation of the job shop scheduling problem. *European Journal of Operational Research*, Elsevier, v. 127, n. 2, p. 317–331, 2000. Citado na página 24.

BRANDES, U. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, Taylor & Francis, v. 25, n. 2, p. 163–177, 2001. Citado na página 36.

BRASIL. *Projeto Custos Ferroviários: Relatório Final Objeto 1*. Brasılia, DF: Agência Nacional de Transportes Terrestres (ANTT), 2018. Citado na página 20.

BURDETT, R. L.; KOZAN, E. Scheduling trains on parallel lines with crossover points. *Journal of Intelligent Transportation Systems*, Taylor & Francis, v. 13, n. 4, p. 171–187, 2009. Citado 2 vezes nas páginas 25 e 32.

CAI, X.; GOH, C.; MEES, A. I. Greedy heuristics for rapid scheduling of trains on a single track. *IIE transactions*, Taylor & Francis, v. 30, n. 5, p. 481–493, 1998. Citado 2 vezes nas páginas 25 e 30.

CAIMI, G. et al. A new resource-constrained multicommodity flow model for conflict-free train routing and scheduling. *Transportation science*, INFORMS, v. 45, n. 2, p. 212–227, 2011. Citado 2 vezes nas páginas 25 e 26.

CAPRARA, A.; FISCHETTI, M.; TOTH, P. Modeling and solving the train timetabling problem. *Operations research*, Informs, v. 50, n. 5, p. 851–861, 2002. Citado 2 vezes nas páginas 22 e 34.

COELLO, C. C. et al. An updated survey of evolutionary multiobjective optimization techniques: State of the art and future trends. In: IEEE PRESS PISCATAWAY, NJ. *Proceedings of the Congress on Evolutionary Computation*. [S.l.], 1999. v. 1, p. 3–13. Citado na página 67.

COSTA, H. D.; PAIVA, M. H. M.; ROCHA, H. R. d. O. Supressão de conflitos no problema de planejamento e roteamento de trens utilizando a metaheurística VNS. In: *Anais do 50º Simposio Brasileiro de Pesquisa Operacional*. Rio de Janeiro, RJ: SOBRAPO, 2018. Citado na página 37.

COUSINEAU, M. et al. RWA problem with geodesics in realistic OTN topologies. *Optical Switching and Networking*, Elsevier, v. 15, p. 18–28, 2015. Citado na página 36.

D'ARIANO, A. et al. Reordering and local rerouting strategies to manage train traffic in real time. *Transportation science*, INFORMS, v. 42, n. 4, p. 405–419, 2008. Citado 3 vezes nas páginas 25, 26 e 32.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. *Numerische mathematik*, Springer, v. 1, n. 1, p. 269–271, 1959. Citado na página 35.

D'ARIANO, A.; PACCIARELLI, D.; PRANZO, M. A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research*, Elsevier, v. 183, n. 2, p. 643–657, 2007. Citado na página 25.

FREEMAN, L. C. A set of measures of centrality based on betweenness. *Sociometry*, JSTOR, p. 35–41, 1977. Citado na página 36.

FREITAS, L. Q. de. *Medidas de centralidade em grafos*. Tese (Doutorado) — dissertação de mestrado, Universidade Federal do Rio de Janeiro, 2010. Citado na página 35.

GENDREAU, M.; POTVIN, J.-Y. et al. *Handbook of metaheuristics*. [S.l.]: Springer, 2010. v. 2. Citado na página 49.

GOLDREICH, O. Computational complexity: a conceptual perspective. *ACM Sigact News*, ACM, v. 39, n. 3, p. 35–39, 2008. Citado na página 22.

GOMES, F. P. *Curso de estatística experimental*. [S.l.]: Nobel, 1987. Citado 2 vezes nas páginas 79 e 81.

HANSEN, I. A.; PACHL, J. Railway timetabling & operations. *Eurailpress, Hamburg*, 2014. Citado na página 25.

HANSEN, P.; MLADENOVIĆ, N. An introduction to variable neighborhood search. In: *Meta-heuristics*. [S.l.]: Springer, 1999. p. 433–458. Citado na página 40.

HANSEN, P.; MLADENOVIĆ, N. Variable neighborhood search: Principles and applications. *European journal of operational research*, Elsevier, v. 130, n. 3, p. 449–467, 2001. Citado 3 vezes nas páginas 40, 46 e 50.

HANSEN, P.; MLADENOVIC, N. Variable neighborhood search methods. 2007. Citado 3 vezes nas páginas 46, 47 e 48.

INGOLOTTI, L. et al. A scheduling order-based method to solve timetabling problems. In: SPRINGER. *Conference of the Spanish Association for Artificial Intelligence*. [S.l.], 2005. p. 52–61. Citado na página 37.

JÜNGER, M.; MUTZEL, P. *Graph drawing software*. [S.l.]: Springer Science & Business Media, 2012. Citado na página 33.

LANDEX, A. et al. Practical use of the UIC 406 capacity leaflet by including timetable tools in the investigations. *WIT Transactions on The Built Environment*, WIT Press, v. 88, 2006. Citado 3 vezes nas páginas 20, 27 e 63.

LEAFLET, U. *Code 406–Capacity*. [S.l.]: Paris: International Union of Railways (UIC), 2013. Citado na página 27.

LI, F. et al. Train routing model and algorithm combined with train scheduling. *Journal of Transportation Engineering*, American Society of Civil Engineers, v. 139, n. 1, p. 81–91, 2012. Citado 2 vezes nas páginas 19 e 25.

MASCIS, A.; PACCIARELLI, D. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, Elsevier, v. 143, n. 3, p. 498–517, 2002. Citado na página 24.

NETTO, P. O. B. Grafos: teoria, modelos, algoritmos. rev. e ampl. Edgard Blücher, 2012. Citado 5 vezes nas páginas 19, 30, 33, 54 e 107.

PAIVA, M. *Aplicação de sistemas baseados em regras fuzzy para o roteamento em redes ópticas*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, 2008. Citado na página 18.

PAPADIMITRIOU, C. H.; STEIGLITZ, K. *Combinatorial optimization: algorithms and complexity*. [S.l.]: Courier Corporation, 1998. Citado na página 68.

PARETO, V. Cours d'économie politique, vol. I-II. *F. Rouge, Éditeur, Lausanne*, 1896. Citado na página 66.

PELLEGRINI, P.; MARLIÈRE, G.; RODRIGUEZ, J. Optimal train routing and scheduling for managing traffic perturbations in complex junctions. *Transportation Research Part B: Methodological*, Elsevier, v. 59, p. 58–80, 2014. Citado na página 25.

PINEDO, M. *Planning and scheduling in manufacturing and services*. [S.l.]: Springer, 2005. Citado na página 24.

RUBIO-LARGO, A. et al. Solving the routing and wavelength assignment problem in WDM networks by using a multiobjective variable neighborhood search algorithm. In: SPRINGER. *Soft computing models in industrial and environmental applications, 5th international workshop (SOCO 2010)*. [S.l.], 2010. p. 47–54. Citado na página 51.

ŞAHIN, İ. Railway traffic control and train scheduling based oninter-train conflict management. *Transportation Research Part B: Methodological*, Elsevier, v. 33, n. 7, p. 511–534, 1999. Citado na página 37.

SAMÀ, M. et al. A variable neighbourhood search for fast train scheduling and routing during disturbed railway traffic situations. *Computers & Operations Research*, Elsevier, v. 78, p. 480–499, 2017. Citado 3 vezes nas páginas 21, 25 e 26.

SAMÀ, M. et al. Lower and upper bound algorithms for the real-time train scheduling and routing problem in a railway network. *IFAC-PapersOnLine*, Elsevier, v. 49, n. 3, p. 215–220, 2016. Citado 2 vezes nas páginas 25 e 26.

SCHOTT, J. R. *Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization*. [S.l.], 1995. Citado na página 69.

SUN, Y.; CAO, C.; WU, C. Multi-objective optimization of train routing problem combined with train scheduling on a high-speed railway network. *Transportation Research Part C: Emerging Technologies*, Elsevier, v. 44, p. 1–20, 2014. Citado na página 26.

SZPIGEL, B. *Sequenciamento de trens*. Tese (Doutorado) — Dissertação de Mestrado. Pontifícia Universidade Católica do Rio de Janeiro: Rio de Janeiro, 1972. Citado na página 24.

SZPIGEL, B. Optimal train scheduling on a single line railway. JOURNAL OF OPERATIONS RESEARCH, 1973. Citado na página 25.

TALBI, E.-G. *Metaheuristics: from design to implementation*. [S.l.]: John Wiley & Sons, 2009. v. 74. Citado na página 29.

TÖRNQUIST, J. Computer-based decision support for railway traffic scheduling and dispatching: A review of models and algorithms. In: SCHLOSS DAGSTUHL-LEIBNIZ-ZENTRUM FÜR INFORMATIK. *OASIcs-OpenAccess Series in Informatics*. [S.l.], 2006. v. 2. Citado na página 19.

YEN, J. Y. Finding the k shortest loopless paths in a network. *Management Science*, INFORMS, v. 17, n. 11, p. 712–716, 1971. Citado 2 vezes nas páginas 27 e 35.

ZITZLER, E. *Evolutionary algorithms for multiobjective optimization: Methods and applications*. [S.l.]: Citeseer, 1999. v. 63. Citado na página 69.

ZITZLER, E.; THIELE, L. Multiobjective optimization using evolutionary algorithms—a comparative case study. In: SPRINGER. *International conference on parallel problem solving from nature*. [S.l.], 1998. p. 292–301. Citado na página 68.

Apêndices

APÊNDICE A – Estrutura da Aplicação

Neste capítulo é apresentada a estrutura completa da aplicação referente ao método de solução proposto por esta pesquisa para resolver o problema de planejamento e roteamento de trens em redes aleatórias planares modeladas via grafos.

A.1 Ambiente de Teste

O algoritmo do método de solução proposto nesta pesquisa foi desenvolvido e testado em um computador *Apple MacBook Air*, com processador *1,4 GHz Intel Core i5*, memória RAM *4 GB 1600 MHz DDR3* e sistema operacional *macOS Mojave versão 10.14.1*.

A.2 Fluxograma da Aplicação

O fluxograma apresentado na Figura 30 mostra as principais etapas do método proposto. Essas etapas serão brevemente descritas a seguir e nas seções subsequentes suas partes serão descritas em mais detalhes.

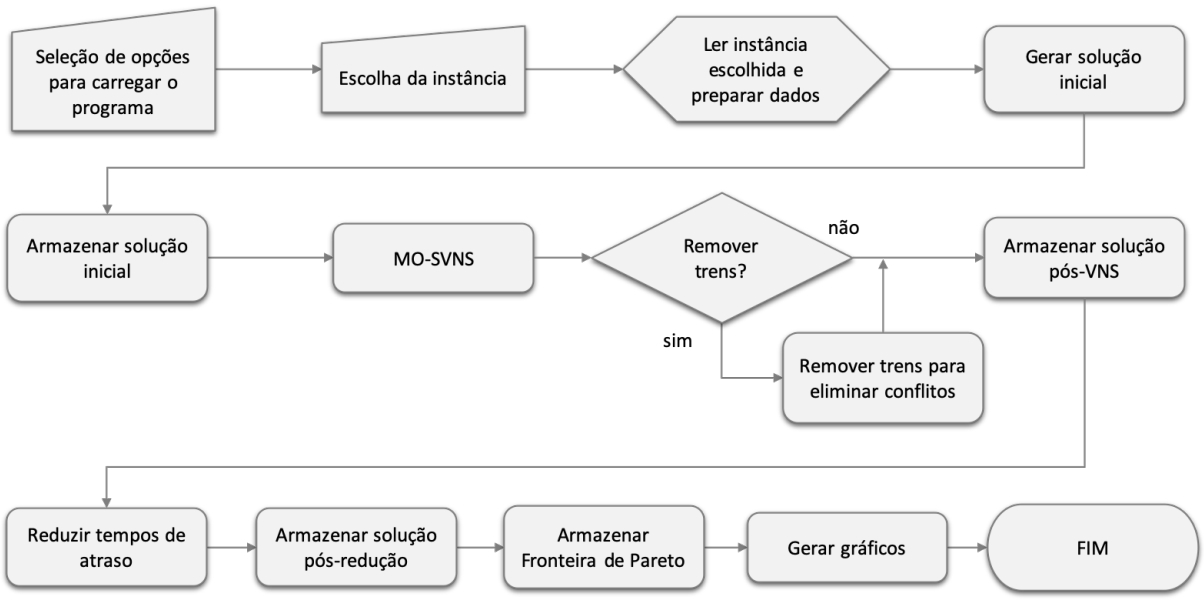

Figura 30 – Fluxograma da aplicação desenvolvida.

Fonte: O autor

A.2.1 Seleção de Opções

O programa inicia dando ao usuário algumas opções que serão consideradas ao longo da execução. São realizadas seis perguntas, a saber:

1. Calcular menores caminhos pelos Pesos ou pela métrica Betweenness?
2. Usar pesos diferentes para as arestas? Se não, o usuário informa o peso padrão.
3. Definir tempo limite para a busca da solução?
4. Remover trens se restarem conflitos?
5. Quantas rotas devem ser calculadas por trem?
6. Quantas vezes o algoritmo deve ser executado por instância?

A.2.2 Seleção de Instância

Definidas as opções acima, o usuário deve escolher a instância que deverá ser carregada. As instâncias são aquelas cujas propriedades foram apresentadas na Tabela 1, página 34.

Na etapa seguinte é realizada a leitura da instância escolhida. O formato dos dados é apresentado na Figura 31, que mostra apenas as primeiras linhas de um arquivo de dados. Os detalhes sobre cada campo são apresentados na própria figura.

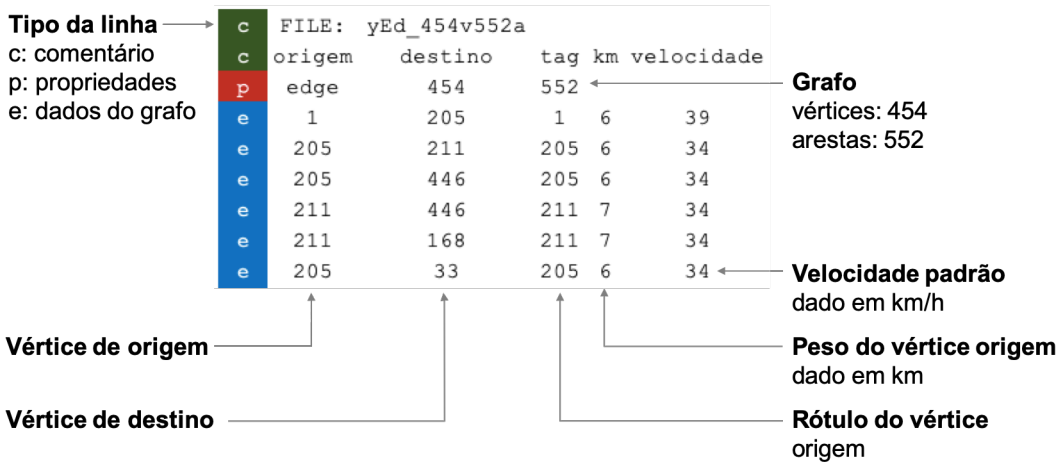

Figura 31 – Formato dos dados para a leitura dos grafos.

Fonte: O autor

A.2.3 Dados Iniciais

Após realizada a leitura do grafo, os dados iniciais são gerados dando origem às seguintes estruturas de dados:

- **Matriz de adjacências**: Matriz n x n, onde n é o número de vértices do grafo. Nessa matriz são mostradas as conexões de cada vértice com seus vértices adjacentes e o peso de cada uma dessas conexões. Segundo Netto (2012), um vértice adjacente é todo aquele que participa de uma ligação com um dado vértice de referência;
- **Matriz esparsa**: Matriz em que a maioria dos elementos é igual a zero. Trata-se de uma matriz com três colunas onde é apresentada a conexão entre pares de vértices e o peso de cada uma dessas conexões;
- **Vértices**: Número de vértices do grafo;
- **Arestas**: Número de arestas do grafo;
- **Trechos**: Matriz 1 x n, onde n é o número de vértices do grafo. Esses vértices correspondem aos trechos da rede por onde os trens podem trafegar;
- **Distâncias**: Matriz 1 x n, onde n é o número de vértices do grafo. Nessa matriz são apresentados os pesos para cada um dos trechos;
- **Velocidades**: Matriz de 1 x n, onde n é o número de vértices do grafo. Nessa matriz são apresentadas as velocidades permitidas para cada um dos trechos.

A.2.4 Solução Inicial

Uma vez gerados os dados iniciais é possível obter uma solução inicial para o problema de planejamento e roteamento de trens. Para isso é utilizada a matriz de adjacências, a programação inicial dos trens, o número de rotas que devem ser calculadas para cada um deles e os parâmetros adicionais referentes às rotas. Na programação inicial consta o trecho de partida, o trecho de destino e o horário de partida de cada um dos trens. Para cada um dos trens o trecho de origem e de destino foi escolhido aleatoriamente com base nos trechos disponíveis em cada uma das redes. E, quanto ao horário de partida de cada trem, a escolha foi feita de modo aleatório no intervalo compreendido entre as 14h e as 17h, independentemente da rede utilizada. Os parâmetros adicionais trazem essencialmente as informações relativas aos trechos, distâncias e velocidades permitidas em cada um dos trechos. Com base nesses dados são calculados os K menores caminhos para o par *origem–destino* de cada um dos trens. Os menores caminhos ou rotas são aquelas que apresentam a menor soma de pesos dos vértices selecionados. Quanto ao número de rotas, quando há mais de uma é escolhida aquela que gera o menor número de conflitos no

contexto das rotas já escolhidas até o momento. A solução inicial é armazenada e também utilizada como ponto de partida no processo de otimização. O procedimento completo de geração dessa solução é descrito na Seção 4.2.2.

A.2.5 *Multi-objective Skewed Variable Neighborhood Search*

A solução inicial é submetida ao algoritmo da meta-heurística MO-SVNS juntamente com outros dados utilizados. Nessa etapa o MO-SVNS tem como principal objetivo desembaralhar as rotas dos trens ao passo em que trabalha para minimizar tanto o número de conflitos quanto o tempo de parada dos trens. Ao final do procedimento é gerada uma nova solução para o problema. Esse procedimento é explicado na Seção 4.2.1.

A.2.6 Remover Trens para Eliminar Conflitos

Caso seja definido que deverão ser removidos tantos trens quantos forem necessários no caso de restar qualquer conflito após a etapa de otimização do MO-SVNS, a solução obtida será submetida a um procedimento de análise e remoção de trens. Cada conflito é analisado e o trem menos prioritário é então removido da solução corrente. Essa remoção é propagada também para a solução inicial. Após essa etapa a solução obtida pelo MO-SVNS é armazenada.

A.2.7 Reduzir Tempos de Atraso

A próxima etapa do método de otimização é aquela onde é feita a redução dos tempos de atraso. Nessa etapa todo e qualquer tempo dispensável é removido da rota de cada um dos trens. Essa etapa é explicada na Seção 3.5.

A.2.8 Etapas Finais da Aplicação

Nas etapas finais os dados da Fronteira de Pareto são armazenados e os gráficos são gerados. Além disso, são geradas algumas estatísticas relativas às métricas utilizadas para avaliação da qualidade da fronteira obtida. Os detalhes sobre essas métricas podem ser vistos na Seção 5.2.

APÊNDICE B – Resultados Completos

Neste capítulo são apresentados os resultados completos obtidos ao executar todas as instâncias segundo os parâmetros definidos para o cenário *Cen–1*. Nesse cenário foi avaliado o planejamento de trens. Em seguida são apresentados os resultados para os cenários *Cen–1* a *Cen-12*, utilizados para definir o melhor cenário segundo o propósito desta pesquisa. Nos testes com os cenários *Cen–1* a *Cen-12* foi utilizada a instância *94v112a*. Assim que o melhor cenário foi definido, cenário *Cen–8*, todas as instâncias foram testadas novamente. A Tabela 3, página 76, mostra os parâmetros utilizados para cada um dos cenários. É importante notar que neste capítulo os resultados não são comentados, mas as análises dos gráficos são análogas àquelas apresentadas no Capítulo 6.

B.1 Parâmetros Calculados

A seguir são apresentadas as tabelas com os resultados completos calculados para todos os testes realizados. De fato, são mostrados os resultados relativos à melhor execução de cada instância em cada um dos cenários. A melhor execução foi considerada como sendo aquela que deu origem à fronteira escolhida como sendo a Fronteira de Pareto.

B.1.1 Parâmetros para o Planejamento de Trens

As Tabelas 10 e 11 mostram todos os parâmetros calculados para as instâncias testadas nas condições do cenário *Cen–1*.

Tabela 10 – Parâmetros calculados para cada topologia no cenário *Cen–1* - Parte 1.

Grupo	Parâmetro	49v57a	94v112a	184v222a	274v332a	364v442a
Entrada	Vértices	49	94	184	274	364
	Arestas	57	112	222	332	442
	Trens	10	19	37	55	73
	Trechos	66	138	472	650	976
Densidade Conflitos	Conf	8	23	152	201	319
	Conf/Trem	0,800	1,211	4,108	3,655	4,370
	Conf/Trechos	0,121	0,167	0,322	0,309	0,327
Tempo	Temp (h)	0,000	0,013	9,329	20,073	156,916
	TVI (h)	9,296	22,531	72,027	101,602	149,927
	TVV (h)	10,436	32,112	196,175	230,960	528,051
	TVF (h)	9,725	27,641	140,934	171,518	307,150
	TPI (h)	0,000	0,000	0,000	0,000	0,000
	TPV (h)	1,133	9,581	124,148	129,358	338,609
	TPF (h)	0,429	5,110	68,907	69,922	157,655

Tabela 11 – Parâmetros calculados para cada topologia no cenário *Cen–1* - Parte 2.

Grupo	Parâmetro	49v57a	94v112a	184v222a	274v332a	364v442a
Desvio Padrão	DesvPadConf	0,000	0,000	0,000	0,000	0,000
	DesvPadTemp (h)	0,000	0,006	3,464	3,749	25,828
	DesvPadTVI (h)	0,000	0,000	0,000	0,000	0,000
	DesvPadTVV (h)	0,435	2,324	31,789	25,195	28,659
	DesvPadTVF (h)	0,341	0,551	10,707	6,747	10,452
	DesvPadTPI (h)	0,000	0,000	0,000	0,000	0,000
	DesvPadTPV (h)	0,446	2,324	31,789	25,195	53,972
	DesvPadTPF (h)	0,341	0,551	10,707	6,745	3,650
Indicador	ConfRest	0	0	0	0	0
	Temp/Vértice (min)	0,000	0,009	3,042	4,395	25,865
	Temp/Trem (min)	0,002	0,042	15,129	21,897	128,972
	Temp/Conf (min)	0,003	0,035	3,683	5,992	29,514
	TVF/TVI	1,046	1,227	1,957	1,688	2,049
	TVV/TVI	1,123	1,425	2,724	2,273	3,522
	TVF/TVV	0,932	0,861	0,718	0,743	0,582
	TPF/TPV	0,378	0,533	0,555	0,541	0,466
	CVConf	0,000	0,000	0,000	0,000	0,000
	CVTemp	0,522	0,437	0,371	0,187	0,165
	CVTVI	0,000	0,000	0,000	0,000	0,000
	CVTVV	0,042	0,072	0,162	0,109	0,054
	CVTVF	0,035	0,020	0,076	0,039	0,034
	CVTPI	0,000	0,000	0,000	0,000	0,000
	CVTPV	0,393	0,243	0,256	0,195	0,159
	CVTPF	0,794	0,108	0,155	0,096	0,023
Métrica Pareto	MelhorParetoAM	16,860	50,528	493,391	609,783	1074,800
	MelhorParetoHV	9,274	29,751	345,546	419,212	763,379
	MelhorParetoMS	8,040	23,307	153,562	202,243	320,681
	MelhorParetoSP	0,454	2,529	5,717	11,672	9,220
	MediaAM	16,405	49,027	521,750	591,943	1074,800
	MediaHV	9,247	28,004	373,216	401,272	763,379
	MediaMS	8,038	23,228	153,745	202,166	320,681
	MediaSP	8,038	2,205	5,210	11,495	9,220
	DesvPadAM	0,382	1,464	33,626	16,086	22,177
	DesvPadHV	0,184	1,459	33,553	15,262	21,324
	DesvPadMS	0,026	0,056	0,218	0,072	0,164
	DesvPadSP	0,288	0,551	0,509	1,284	1,576
	CVAM	0,023	0,030	0,064	0,027	0,021
	CVHV	0,020	0,052	0,090	0,038	0,028
	CVMS	0,003	0,002	0,001	0,000	0,001
	CVSP	0,036	0,250	0,098	0,112	0,171

B.1.2 Parâmetros para a Avaliação do Melhor Cenário

As Tabelas 12 e 13 mostram os parâmetros calculados para os cenários *Cen–1* a *Cen–8*. Na Tabela 14 são mostrados os parâmetros calculados para os cenários *Cen–9* a *Cen–12*.

Tabela 12 – Parâmetros calculados para os cenários *Cen–1* a *Cen–8* - Parte 1.

Grupo	Parâmetro	Cen–1	Cen–2	Cen–3	Cen–4	Cen–5	Cen–6	Cen–7	Cen–8
Entrada	Rotas	1	1	3	3	1	1	3	3
	Pond/Btw	Pond	Pond	Pond	Pond	Btw	Btw	Btw	Btw
	Pond/Unit	Pond	Unit	Pond	Unit	Pond	Unit	Pond	Unit
	Vértices	94	94	94	94	94	94	94	94
	Arestas	112	112	112	112	112	112	112	112
	Trens	19	19	19	19	19	19	19	19
	Trechos	138	137	149	152	171	156	174	169
Densidade Conflitos	Conf	23	23	21	15	26	24	18	18
	Conf/Trem	1,211	1,211	1,105	0,789	1,368	1,263	0,947	0,947
	Conf/Trechos	0,167	0,168	0,141	0,099	0,152	0,154	0,103	0,107

Tabela 13 – Parâmetros calculados para os cenários *Cen–1* a *Cen–8* - Parte 2.

Grupo	Parâmetro	Cen–1	Cen–2	Cen–3	Cen–4	Cen–5	Cen–6	Cen–7	Cen–8
Tempo	Temp (h)	31,405	29,908	23,671	27,162	45,787	20,339	43,307	23,502
	TVI (h)	22,530	22,250	24,790	25,010	28,820	25,940	29,450	28,270
	TVF (h)	26,972	27,070	30,426	29,920	34,264	28,434	34,782	31,088
	TPF (h)	4,440	4,814	5,636	4,908	5,442	2,498	5,214	2,810
Desvio Padrão	DesvPadTemp (h)	11,777	12,584	10,470	9,409	18,325	5,772	26,681	20,776
	DesvPadTVF (h)	1,492	0,726	0,545	1,316	1,481	0,706	0,949	0,885
	DesvPadTPF (h)	1,495	0,721	0,545	1,315	1,481	0,704	0,748	0,884
Indicador	ConfRest	0	0	0	0	0	0	0	0
	Temp/Vértice (min)	0,334	0,318	0,252	0,289	0,487	0,216	0,461	0,250
	Temp/Trem (min)	1,653	1,574	1,246	1,430	2,410	1,070	2,279	1,237
	Temp/Conf (min)	1,365	1,300	1,127	1,811	1,761	0,847	2,406	1,306
	TVF/TVI	1,197	1,217	1,227	1,196	1,189	1,096	1,181	1,100

Tabela 14 – Parâmetros calculados para os cenários *Cen–9* a *Cen–12*.

Grupo	Parâmetro	Cen–9	Cen–10	Cen–11	Cen–12
Entrada	Rotas	3	3	3	3
	Pond/Btw	Pond	Pond	Btw	Btw
	Pond/Unit	Pond	Pond	Pond	Pond
	TempoLimite (s)	10	10	10	10
	Manter Trens	Sim	Não	Sim	Não
	Vértices	94	94	94	94
	Arestas	112	112	112	112
	Trens Início	19	19	19	19
	Trens Final	19	16	19	13
Densidade Conflitos	Conf	21	21	18	18
	Conf/Trem	1,105	1,105	0,947	0,947
	Conf/Trechos	0,141	0,164	0,103	0,133
Tempo	Temp (h)	12,571	12,377	13,345	11,886
	TVI (h)	24,790	19,936	29,450	21,474
	TVF (h)	27,270	21,894	32,502	23,038
	TPF (h)	2,480	1,956	3,056	1,566
Desvio Padrão	DesvPadTemp (h)	0,465	0,785	0,690	0,904
	DesvPadTVF (h)	1,042	2,796	0,671	2,406
	DesvPadTPF (h)	1,042	0,676	0,668	0,703
Indicador	ConfRest	5	0	5	0
	Temp/Vértice (min)	0,134	0,132	0,142	0,126
	Temp/Trem (min)	0,662	0,651	0,702	0,626
	Temp/Conf (min)	0,599	0,589	0,741	0,660
	TVF/TVI	1,100	1,098	1,104	1,073

B.1.3 Parâmetros para o Método Adaptado

As Tabelas 15 e 16 mostram todos os parâmetros calculados para as instâncias testadas no cenário *Cen–8* em conjunto com as novas estruturas de vizinhança descritas na Tabela 8.

Tabela 15 – Parâmetros calculados para cada topologia no cenário *Cen–8* - Parte 1.

Grupo	Parâmetro	49v57a	94v112a	184v222a	274v332a	364v442a
Entrada	Vértices	49	94	184	274	364
	Arestas	57	112	222	332	442
	Trens	10	19	37	55	73
	Trechos	79	169	724	950	1373
Densidade Conflitos	Conf	5	18	234	88	337
	Conf/Trem	0,500	0,947	6,324	1,600	4,616
	Conf/Trechos	0,063	0,107	0,323	0,093	0,245

Tabela 16 – Parâmetros calculados para cada topologia no cenário *Cen–8* - Parte 2.

Grupo	Parâmetro	49v57a	94v112a	184v222a	274v332a	364v442a
Tempo	Temp (h)	0,000	0,006	6,223	3,664	77,351
	TVI (h)	11,530	28,270	116,590	148,190	211,670
	TVV (h)	12,202	36,634	292,008	224,536	571,358
	TVF (h)	11,772	31,882	219,710	191,356	403,578
	TPI (h)	0,000	0,000	0,000	0,000	0,000
	TPV (h)	0,668	8,362	175,416	76,344	359,688
	TPF (h)	0,242	3,610	103,118	43,166	191,908
Desvio Padrão	DesvPadConf	0,000	0,000	0,000	0,000	0,000
	DesvPadTemp (h)	0,000	0,002	0,448	0,251	17,751
	DesvPadTVI (h)	0,000	0,000	0,000	0,000	0,000
	DesvPadTVV (h)	0,213	4,670	20,649	5,752	99,898
	DesvPadTVF (h)	0,317	0,969	13,063	4,062	34,166
	DesvPadTPI (h)	0,000	0,000	0,000	0,000	0,000
	DesvPadTPV (h)	0,215	4,669	20,652	5,752	99,898
	DesvPadTPF (h)	0,317	0,970	13,063	4,062	34,166
Indicador	ConfRest	0	0	0	0	0
	Temp/Vértice (min)	0,000	0,004	2,029	0,802	12,750
	Temp/Trem (min)	0,001	0,018	10,091	3,997	63,576
	Temp/Conf (min)	0,002	0,019	1,596	2,498	13,772
	TVF/TVI	1,021	1,128	1,884	1,291	1,907
	TVV/TVI	1,058	1,296	2,505	1,515	2,699
	TVF/TVV	0,965	0,870	0,752	0,852	0,706
	TPF/TPV	0,362	0,432	0,588	0,565	0,534
	CVConf	0,000	0,000	0,000	0,000	0,000
	CVTemp	0,260	0,354	0,072	0,069	0,229
	CVTVI	0,000	0,000	0,000	0,000	0,000
	CVTVV	0,017	0,127	0,071	0,026	0,175
	CVTVF	0,027	0,030	0,059	0,021	0,085
	CVTPI	0,000	0,000	0,000	0,000	0,000
	CVTPV	0,322	0,558	0,118	0,075	0,278
	CVTPF	1,311	0,269	0,127	0,094	0,178
Métrica Pareto	MelhorParetoAM	10,566	38,649	1034,300	226,722	1472,800
	MelhorParetoHV	6,052	21,410	806,277	140,909	1140,500
	MelhorParetoMS	5,020	18,208	236,848	88,769	339,741
	MelhorParetoSP	0,506	0,969	8,782	2,956	7,445
	MediaAM	10,275	37,695	1023,900	227,470	1427,900
	MediaHV	6,043	21,449	794,857	141,964	1096,100
	MediaMS	5,022	18,189	236,807	88,780	339,624
	MediaSP	0,790	1,943	7,754	3,273	7,841
	DesvPadAM	0,479	0,806	82,501	6,584	148,776
	DesvPadHV	0,036	0,905	82,375	6,353	147,854
	DesvPadMS	0,026	0,051	0,355	0,076	0,467
	DesvPadSP	0,443	0,704	0,855	0,315	1,179
	CVAM	0,047	0,021	0,081	0,029	0,104
	CVHV	0,006	0,042	0,104	0,045	0,135
	CVMS	0,005	0,003	0,001	0,001	0,001
	CVSP	0,561	0,362	0,110	0,096	0,150

B.2 Resultados Obtidos pelo Método Implementado

Os gráficos a seguir mostram a solução inicial, a solução dada pelo MO-SVNS e a solução obtida após a redução dos tempos de atraso.

B.2.1 Resultados para o Planejamento de Trens

A seguir, Figuras 32 a 36, são apresentados os resultados dados pelo método de solução proposto ao executar todas as instâncias no cenário *Cen–1*.

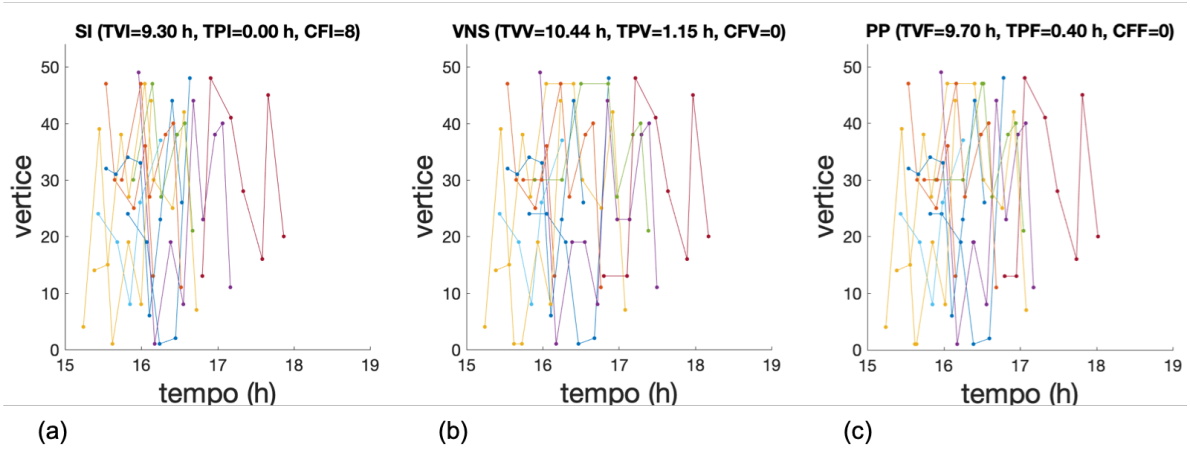

Figura 32 – Supressão de conflitos para a instância *49v57a*, *Cen–1*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

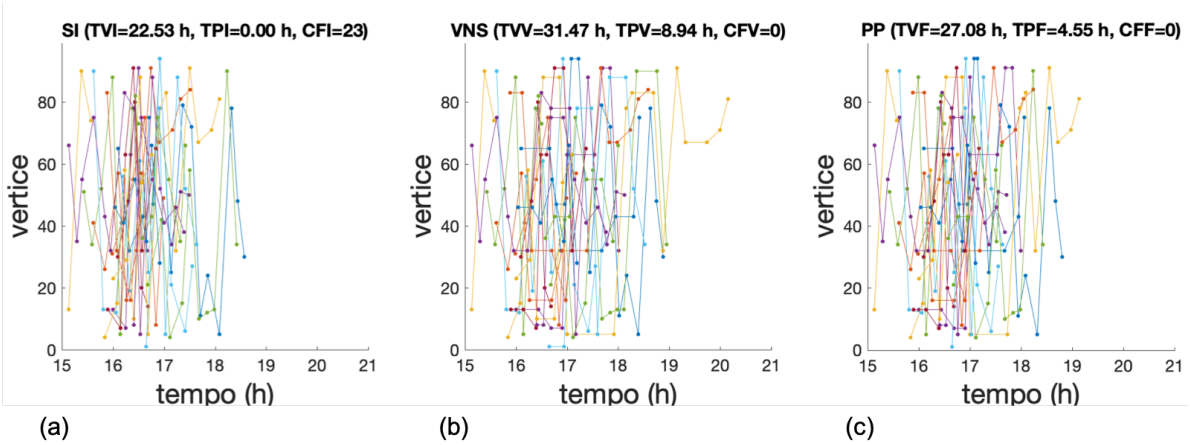

Figura 33 – Supressão de conflitos para a instância *94v112a*, *Cen–1*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

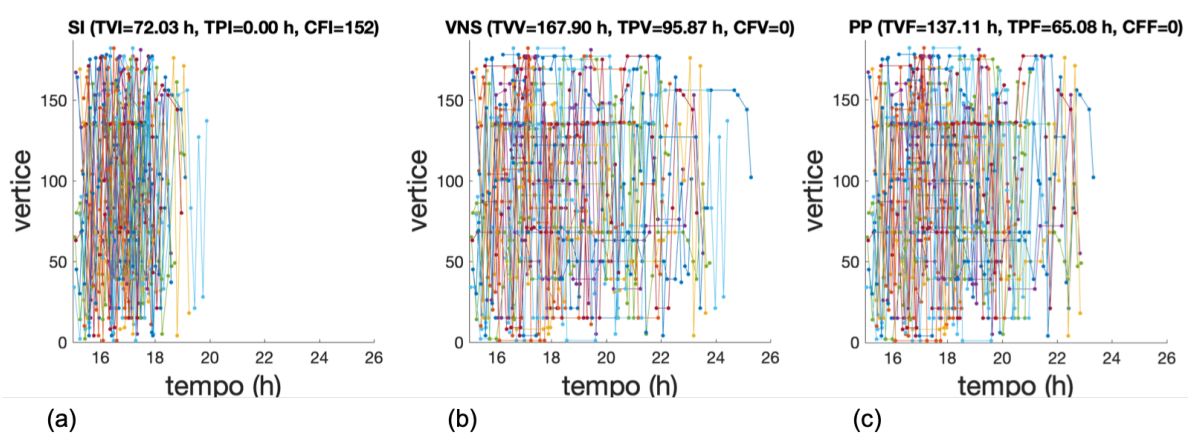

Figura 34 – Supressão de conflitos para a instância *184v222a*, *Cen–1*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

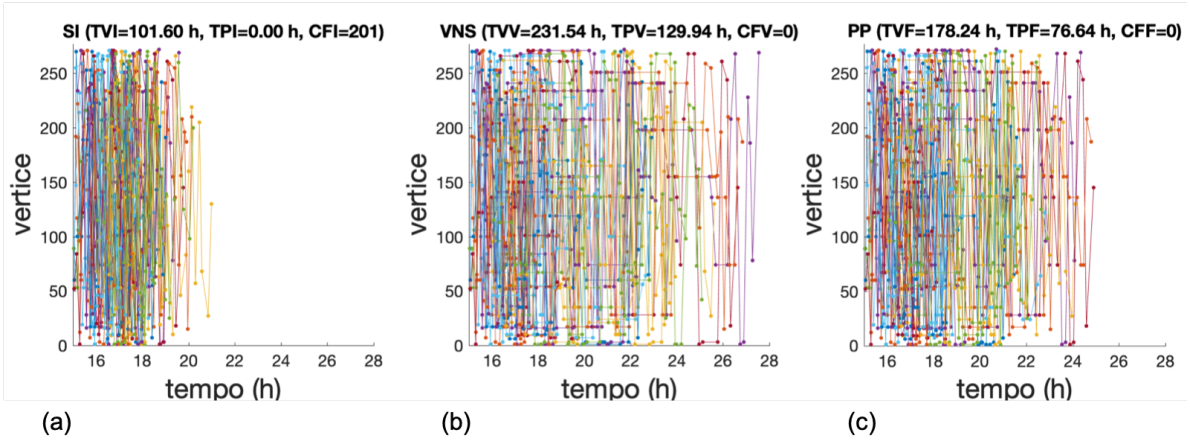

Figura 35 – Supressão de conflitos para a instância *274v332a*, *Cen–1*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

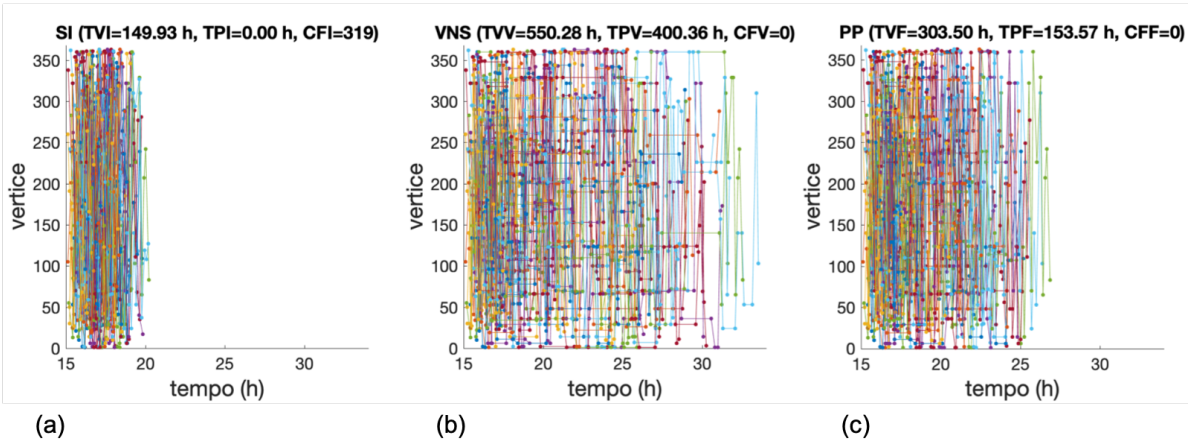

Figura 36 – Supressão de conflitos para a instância *364v442a*, *Cen–1*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

B.2.2 Resultados para a Avaliação do Melhor Cenário

A seguir, Figuras 37 a 48, são apresentados os resultados dados pelo método de solução proposto para a instância *94v112a*, executada nos cenários *Cen–1* a *Cen–12*.

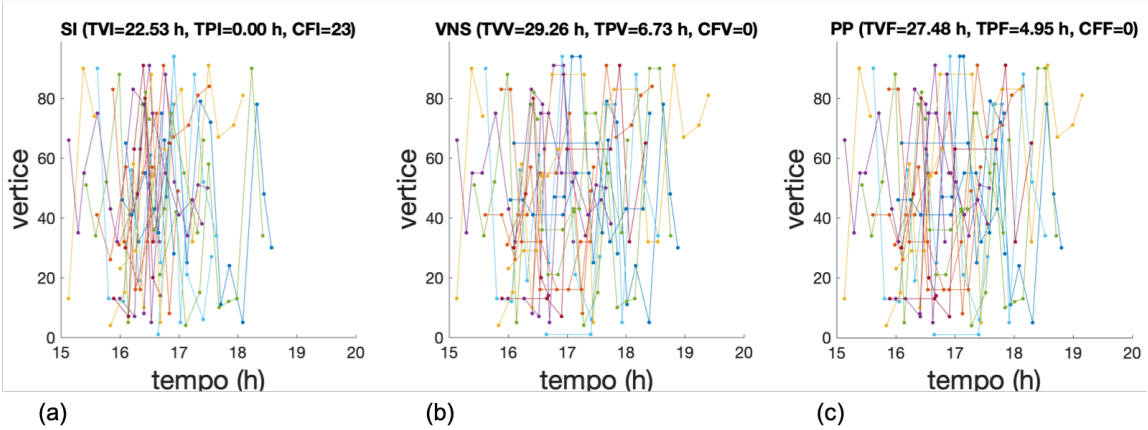

Figura 37 – Supressão de conflitos para a instância *94v112a*, *Cen–1*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

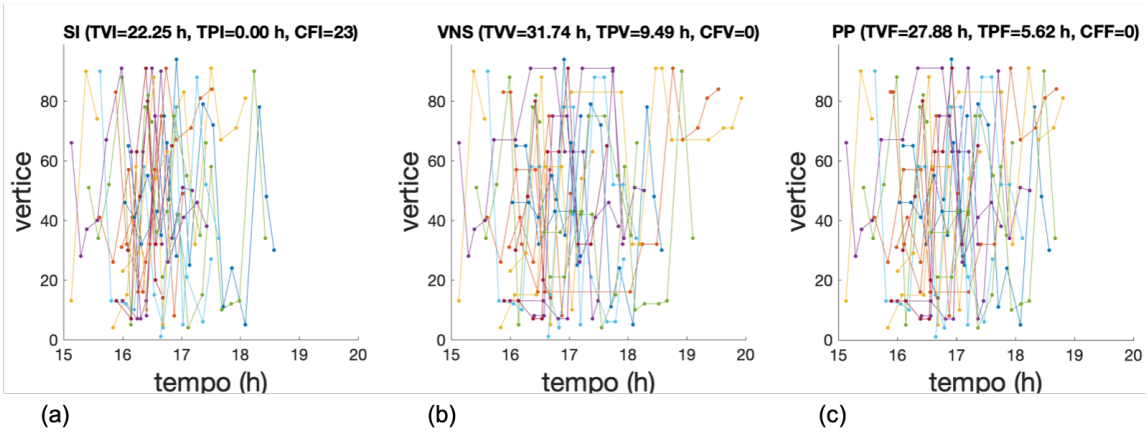

Figura 38 – Supressão de conflitos para a instância *94v112a*, *Cen–2*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

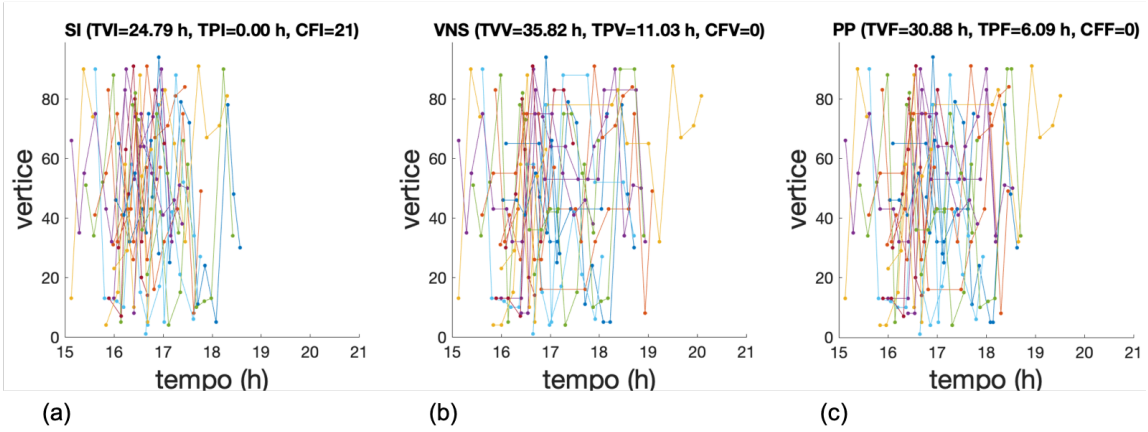

Figura 39 – Supressão de conflitos para a instância *94v112a*, *Cen–3*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

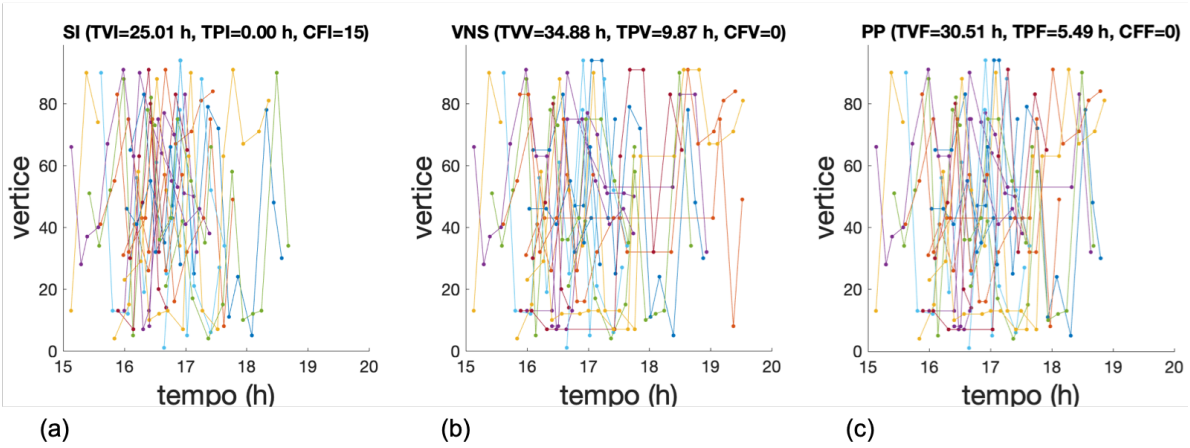

Figura 40 – Supressão de conflitos para a instância *94v112a*, *Cen–4*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

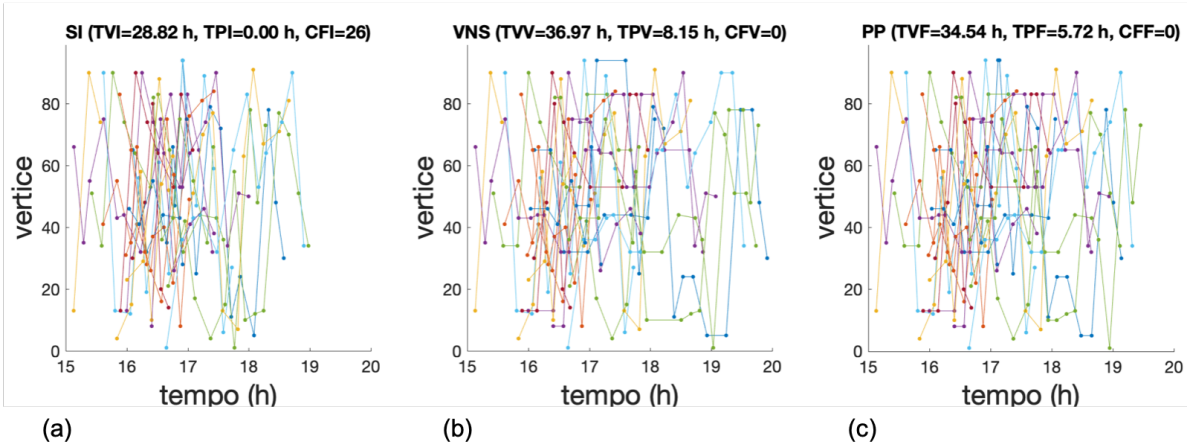

Figura 41 – Supressão de conflitos para a instância *94v112a*, *Cen–5*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

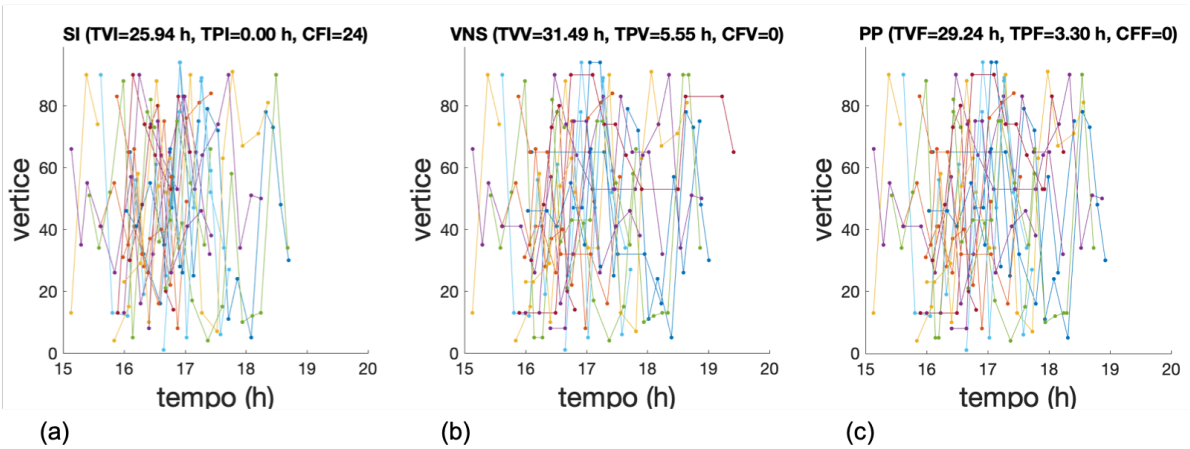

Figura 42 – Supressão de conflitos para a instância *94v112a*, *Cen–6*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

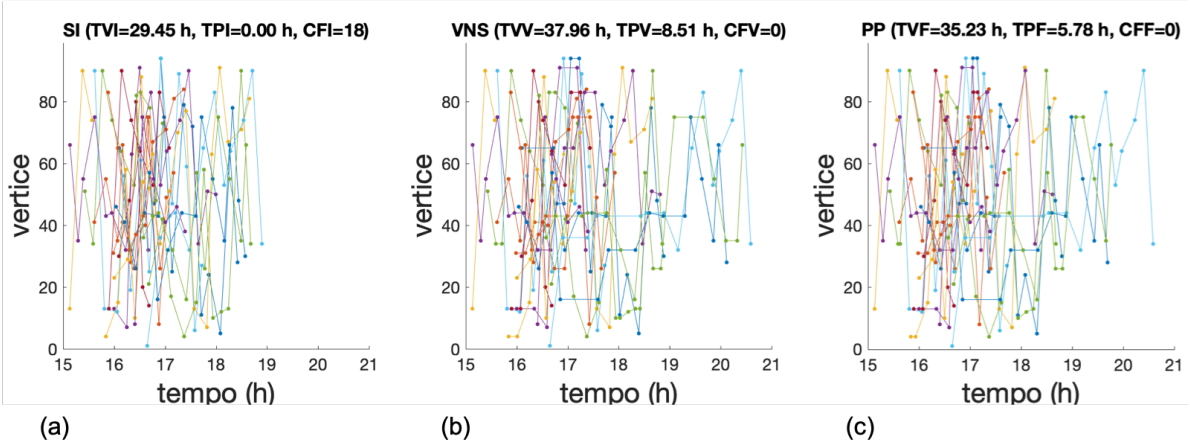

Figura 43 – Supressão de conflitos para a instância *94v112a*, *Cen–7*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

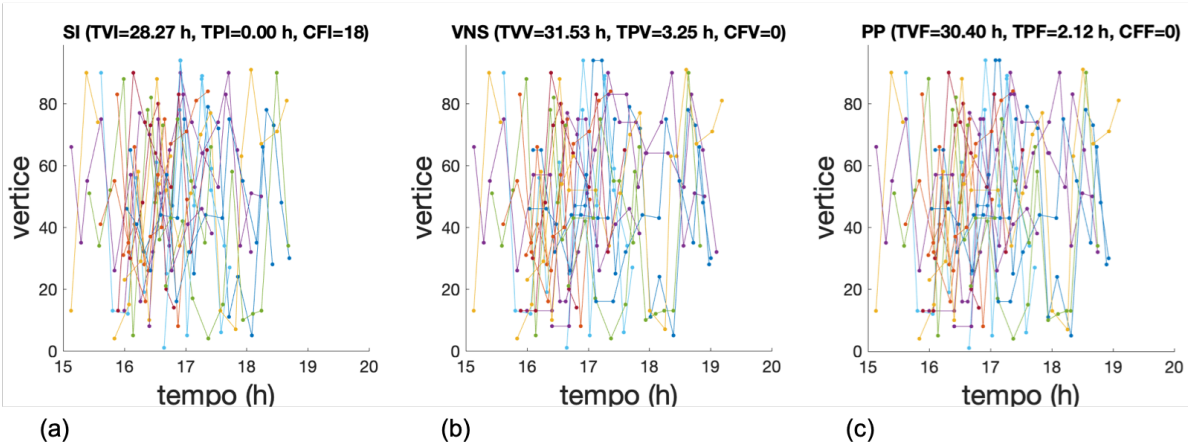

Figura 44 – Supressão de conflitos para a instância *94v112a*, *Cen–8*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

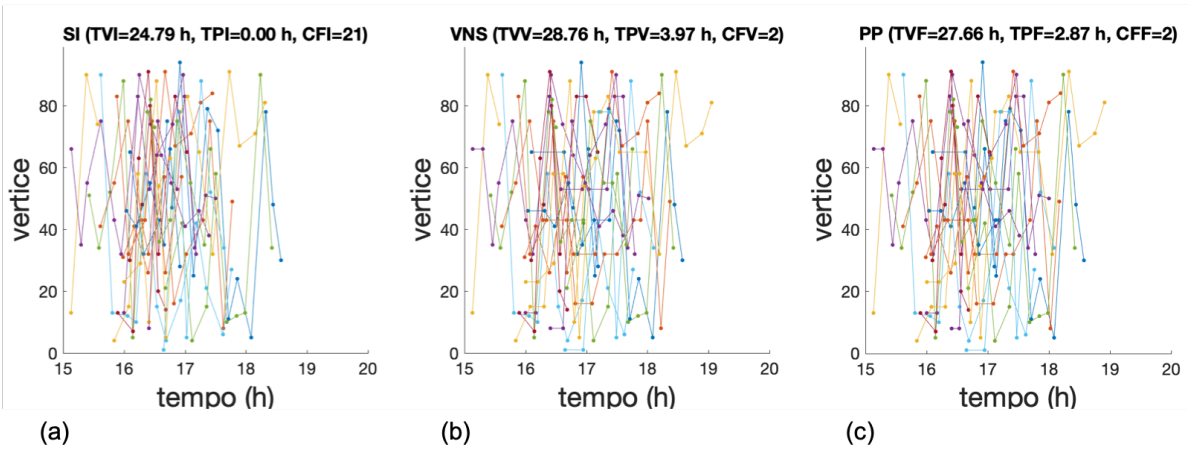

Figura 45 – Supressão de conflitos para a instância *94v112a*, *Cen–9*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

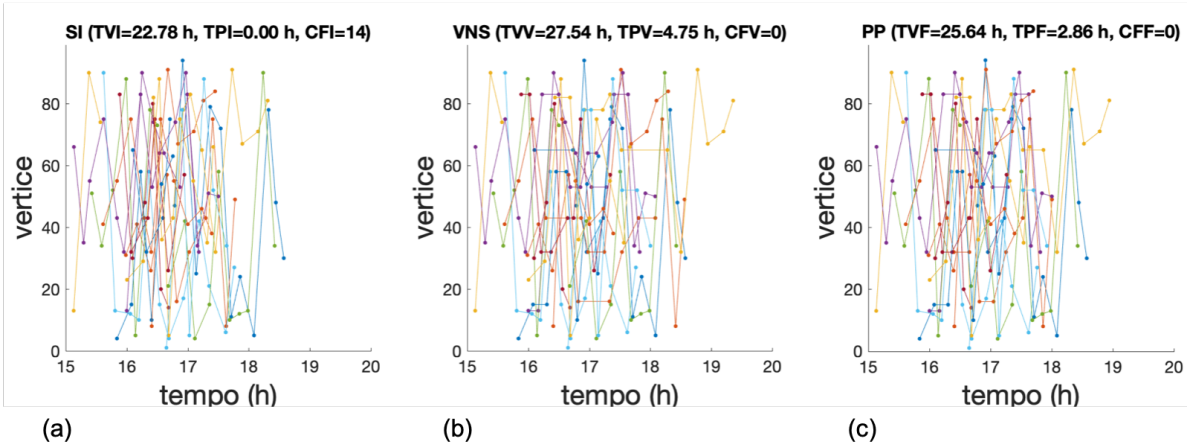

Figura 46 – Supressão de conflitos para a instância *94v112a*, *Cen–10*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

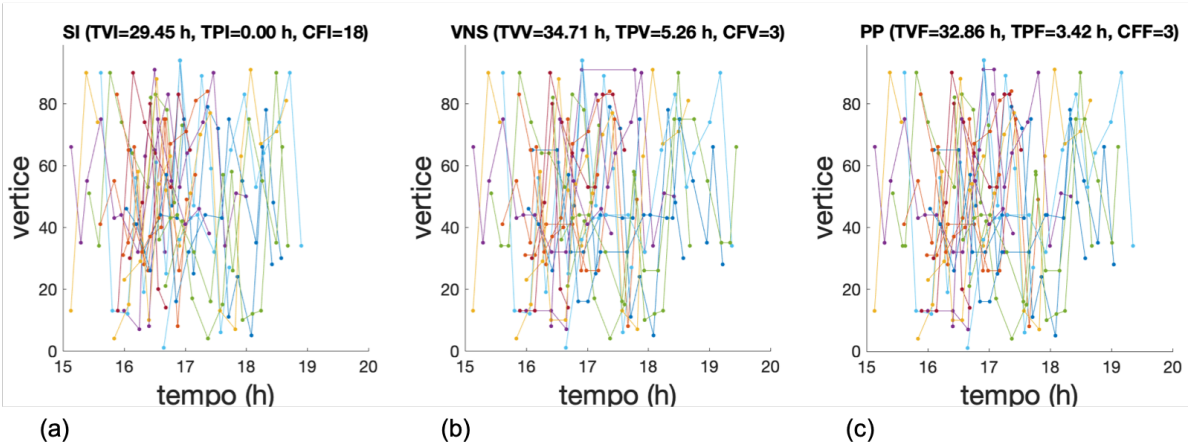

Figura 47 – Supressão de conflitos para a instância *94v112a*, *Cen–11*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

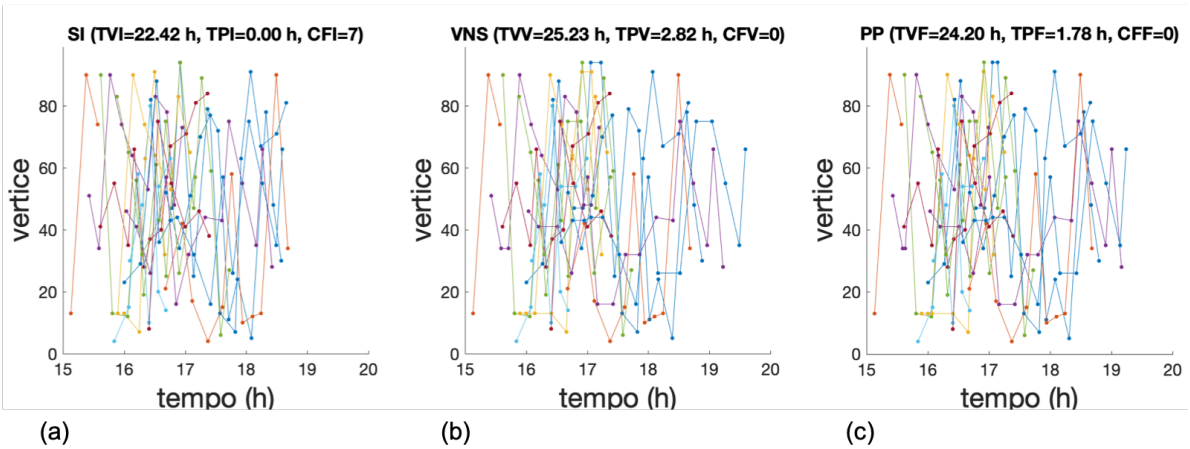

Figura 48 – Supressão de conflitos para a instância *94v112a*, *Cen–12*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

B.2.3 Resultados para o Método Adaptado

A seguir, Figuras 49 a 53, são apresentados os resultados dados pelo método de solução proposto ao executar todas as instâncias no cenário *Cen–8*.

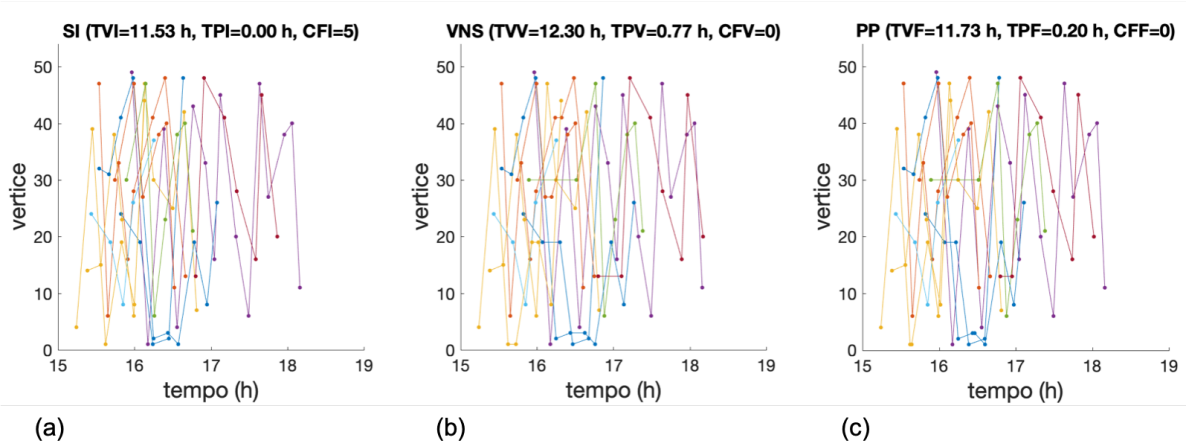

Figura 49 – Supressão de conflitos para a instância *49v57a*, *Cen–8*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

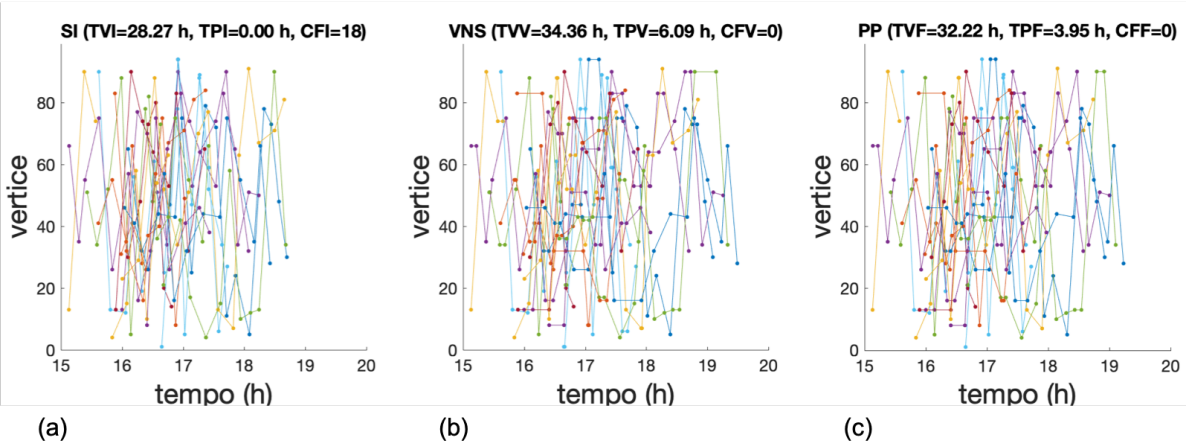

Figura 50 – Supressão de conflitos para a instância *94v112a*, *Cen–8*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

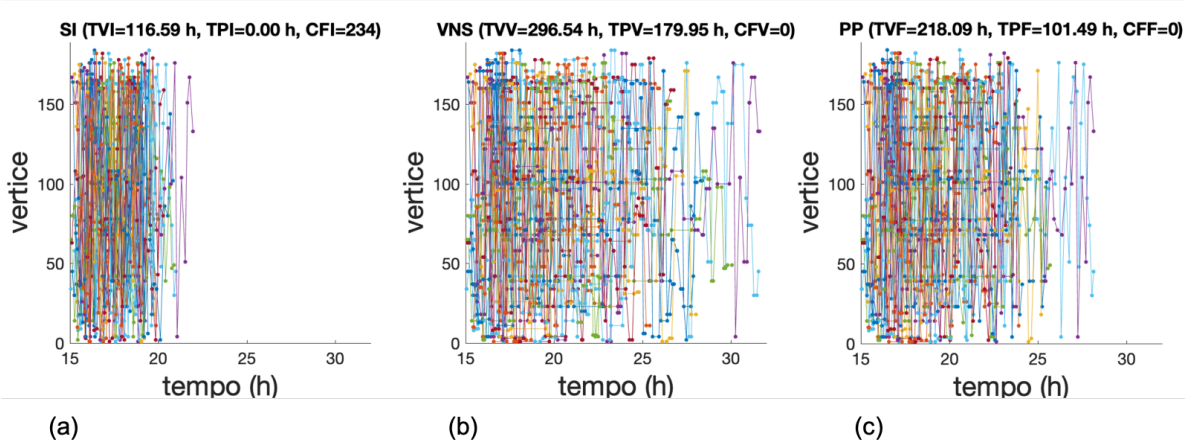

Figura 51 – Supressão de conflitos para a instância *184v222a*, *Cen–8*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

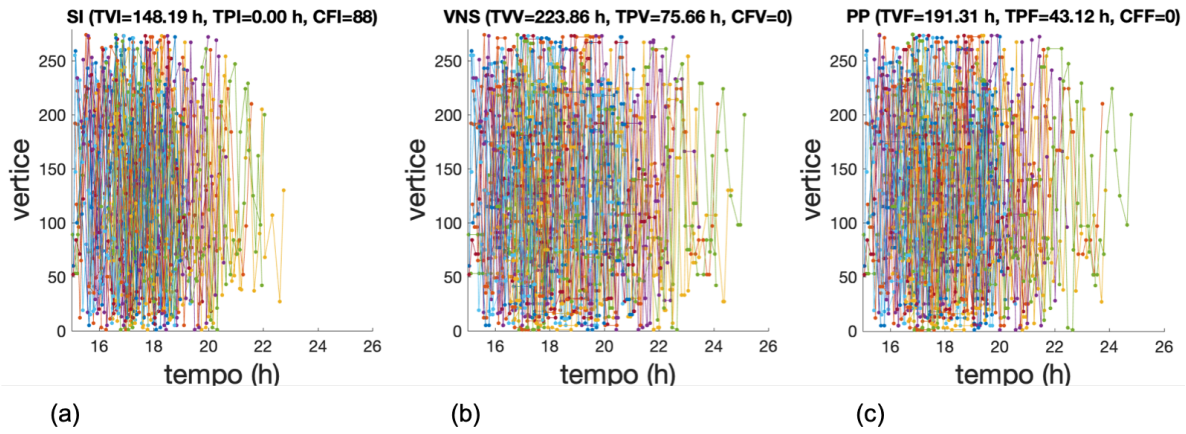

Figura 52 – Supressão de conflitos para a instância *274v332a*, *Cen–8*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

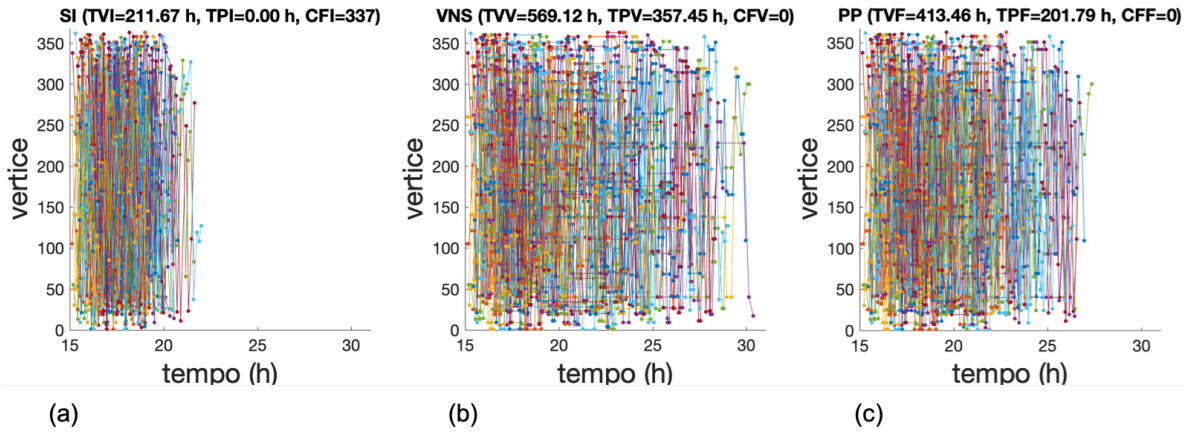

Figura 53 – Supressão de conflitos para a instância *364v442a*, *Cen–8*: (a) solução inicial, (b) solução pelo MO-SVNS e (c) redução de tempos.

B.3 Fronteira de Pareto e Métricas

Nas próximas sessões são apresentadas tanto a melhor Fronteira de Pareto quanto os gráficos das métricas relativas a cada uma das delas.

B.3.1 Fronteira de Pareto e Métricas para o Cenário *Cen–1*

As Fronteiras de Pareto e as métricas apresentadas a seguir, Figuras 54 a 58, são referentes à melhor fronteira obtida para cada uma das instâncias testadas no cenário *Cen–1*.

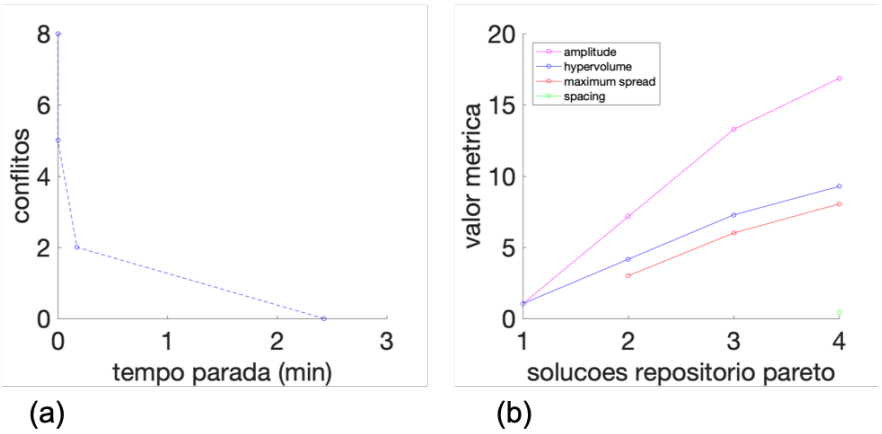

Figura 54 – Resultados para *49v57a*, *Cen–1*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

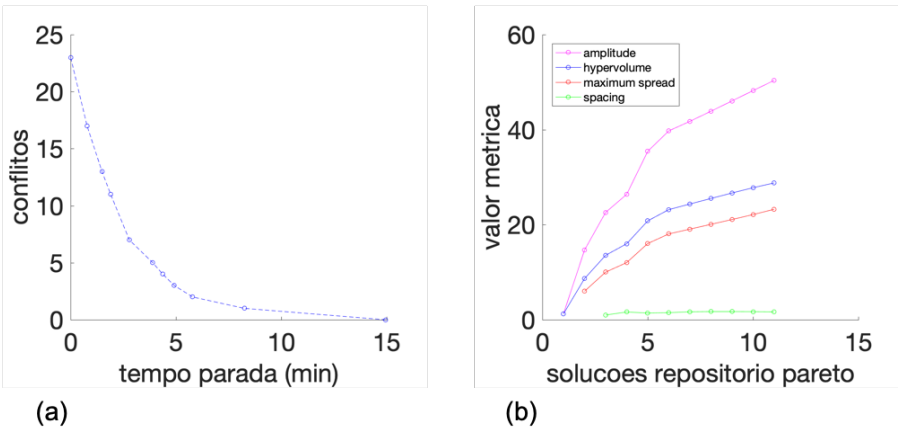

Figura 55 – Resultados para *94v112a*, *Cen–1*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

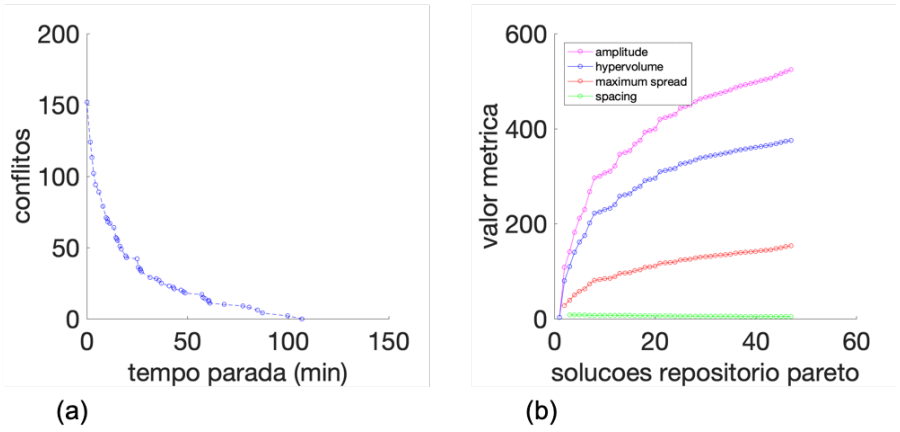

Figura 56 – Resultados para *184v222a*, *Cen–1*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

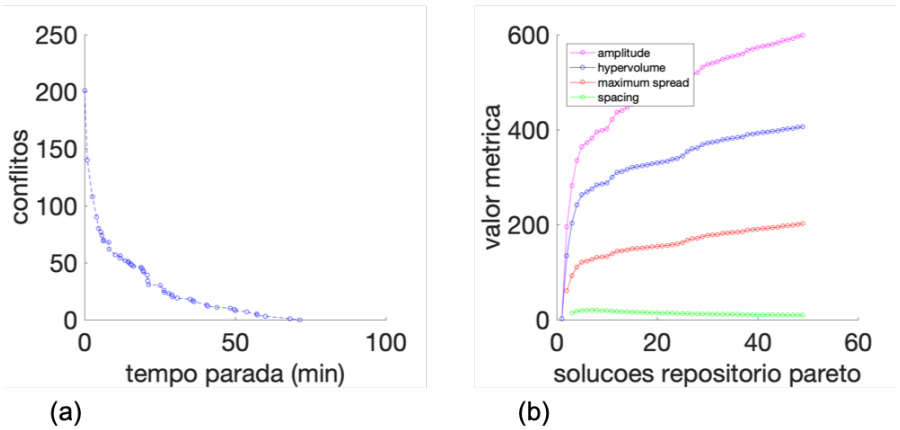

Figura 57 – Resultados para *274v332a*, *Cen–1*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

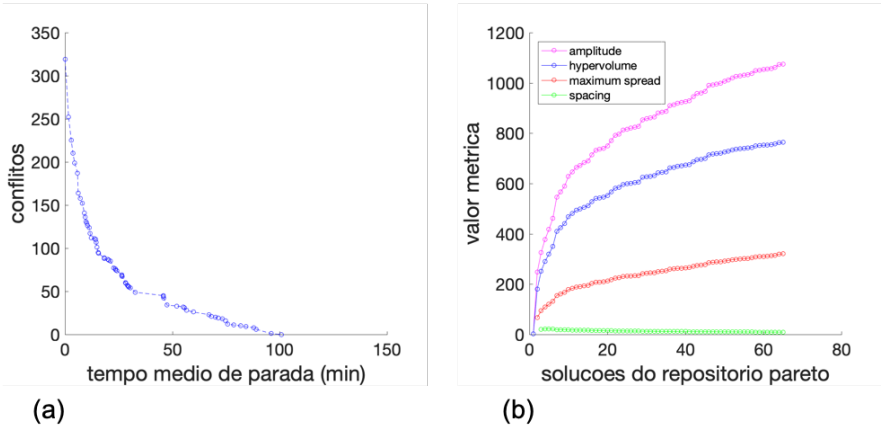

Figura 58 – Resultados para *364v442a*, *Cen–1*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

B.3.2 Fronteira de Pareto e Métricas para a Instância *94v112a*

As Fronteiras de Pareto e as métricas apresentadas a seguir, Figuras 59 a 70, são referentes à melhor fronteira obtida para a instância *94v112a*, executadas nos cenários *Cen–1* a *Cen–12*.

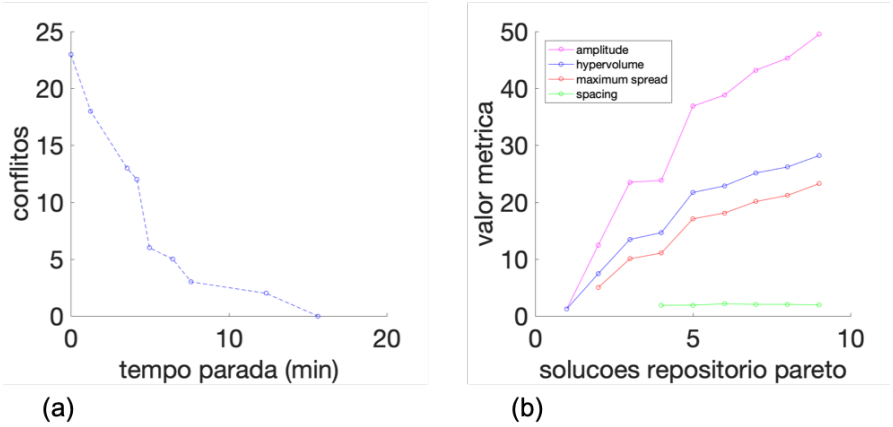

Figura 59 – Resultados para *94v112a*, *Cen–1*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

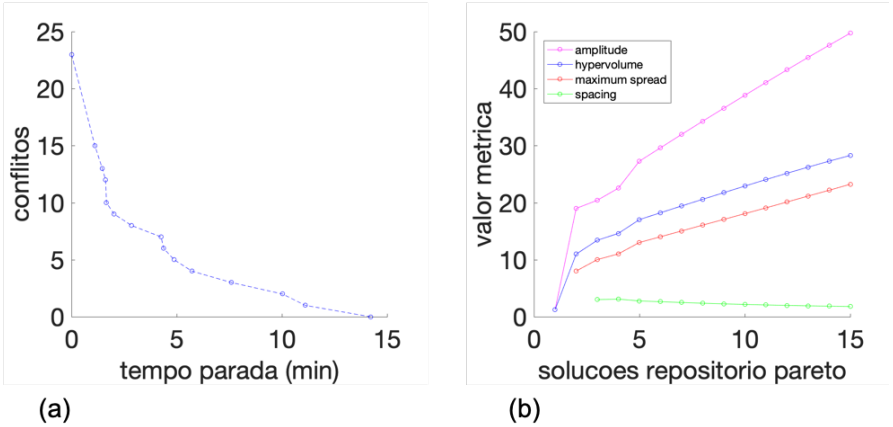

Figura 60 – Resultados para *94v112a*, *Cen–2*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

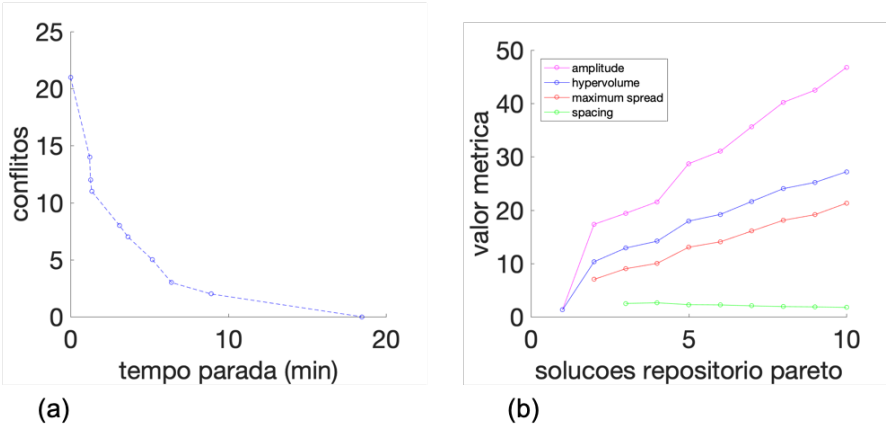

Figura 61 – Resultados para *94v112a*, *Cen–3*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

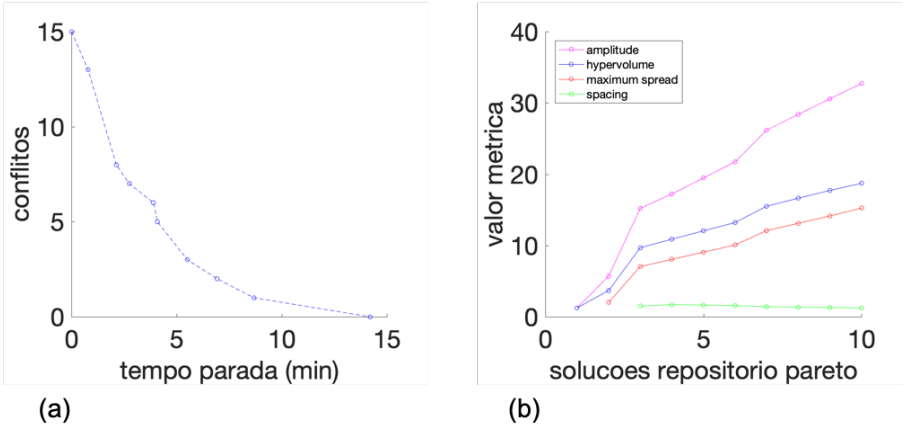

Figura 62 – Resultados para *94v112a*, *Cen–4*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

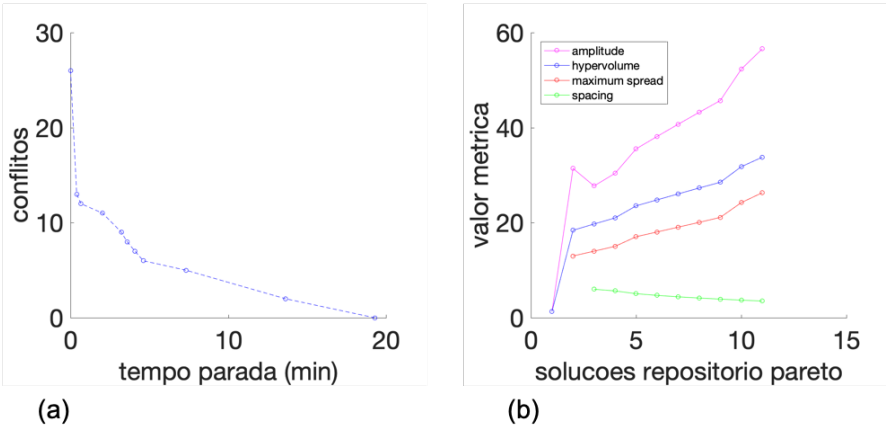

Figura 63 – Resultados para *94v112a*, *Cen–5*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

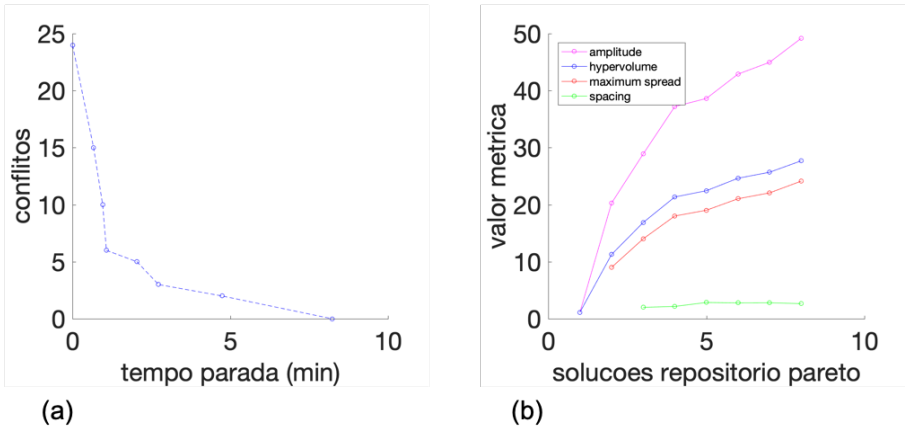

Figura 64 – Resultados para *94v112a*, *Cen–6*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

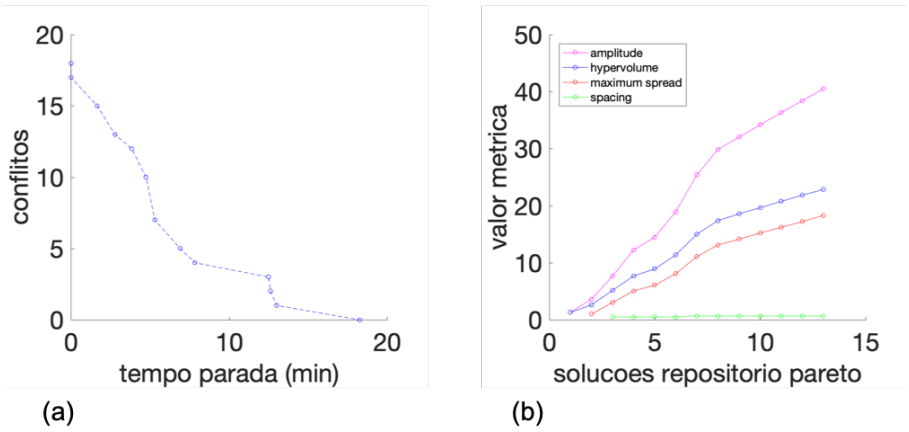

Figura 65 – Resultados para *94v112a*, *Cen–7*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

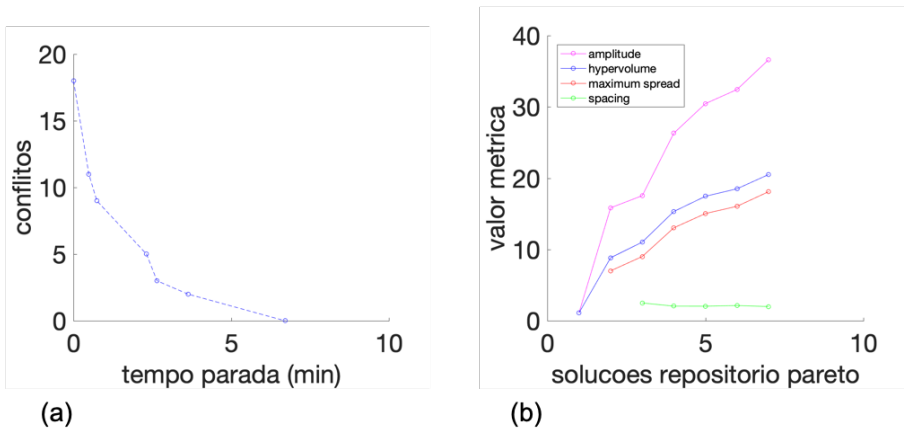

Figura 66 – Resultados para *94v112a*, *Cen–8*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

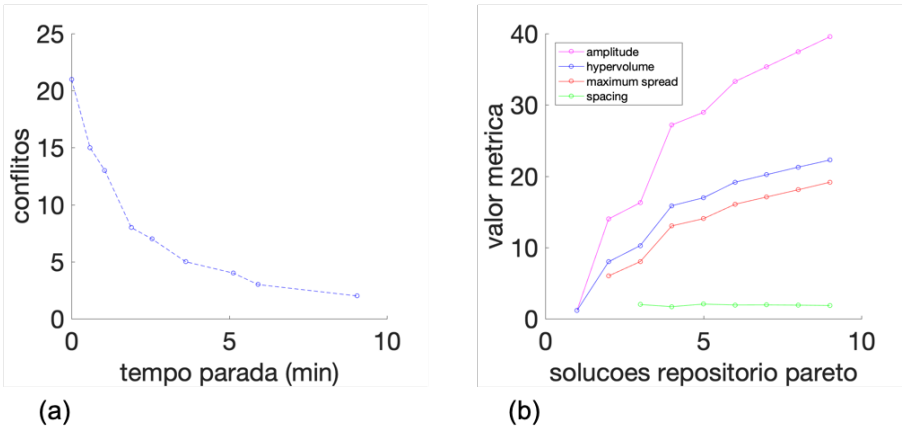

Figura 67 – Resultados para *94v112a*, *Cen–9*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

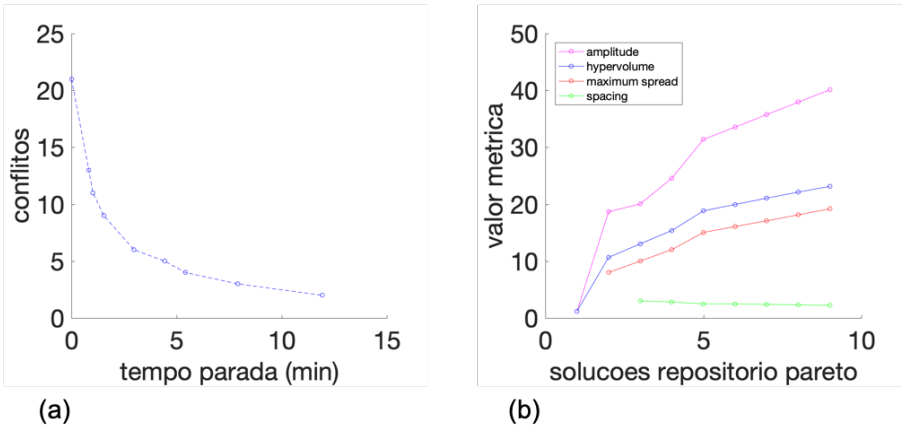

Figura 68 – Resultados para *94v112a*, *Cen–10*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

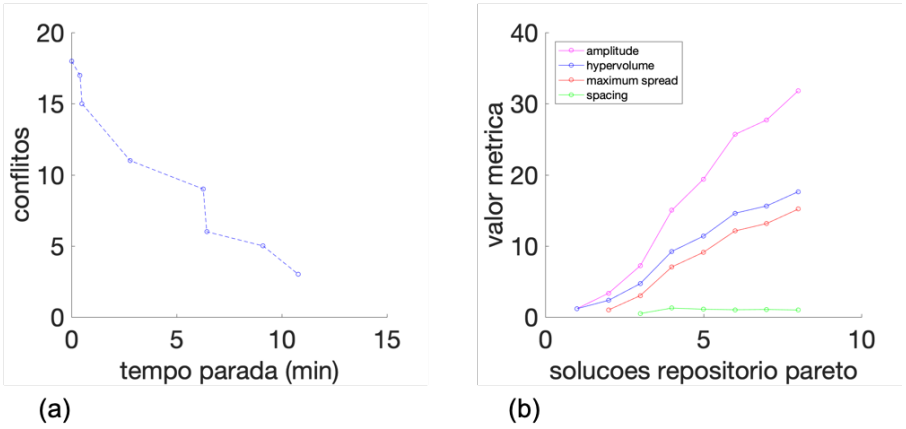

Figura 69 – Resultados para *94v112a*, *Cen–11*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

B.3.3 Fronteira de Pareto e Métricas para o Cenário *Cen–8*

As Fronteiras de Pareto apresentadas a seguir, Figuras 71 a 75, são referentes à melhor fronteira obtida para cada uma das instâncias testadas no cenário *Cen–8*.

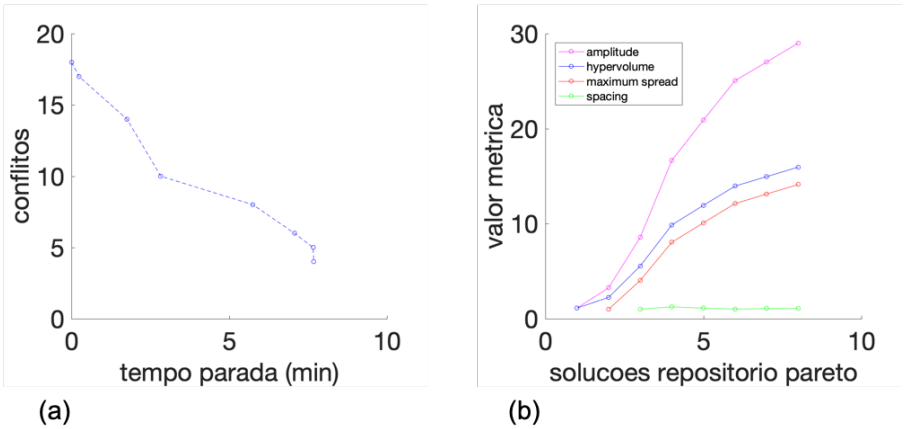

Figura 70 – Resultados para *94v112a*, *Cen–12*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

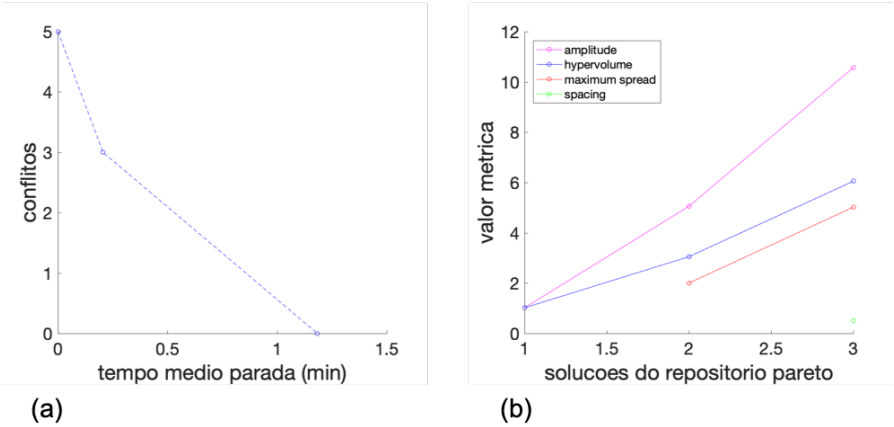

Figura 71 – Resultados para *49v57a*, *Cen–8*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

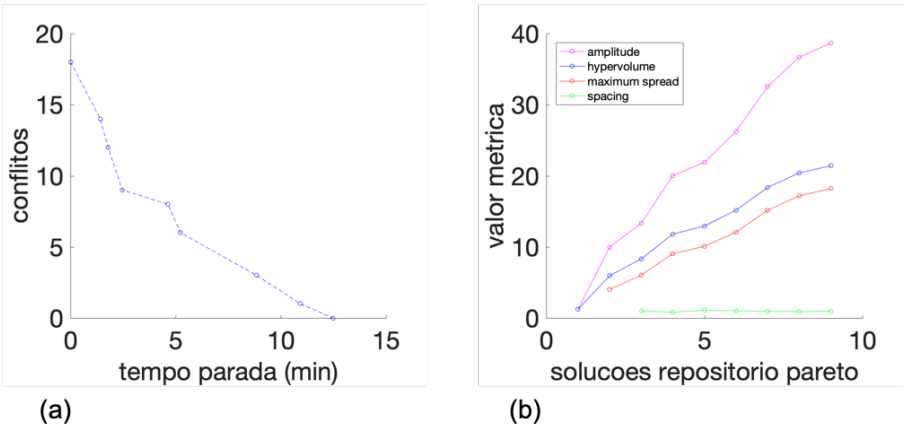

Figura 72 – Resultados para *94v112a*, *Cen–8*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

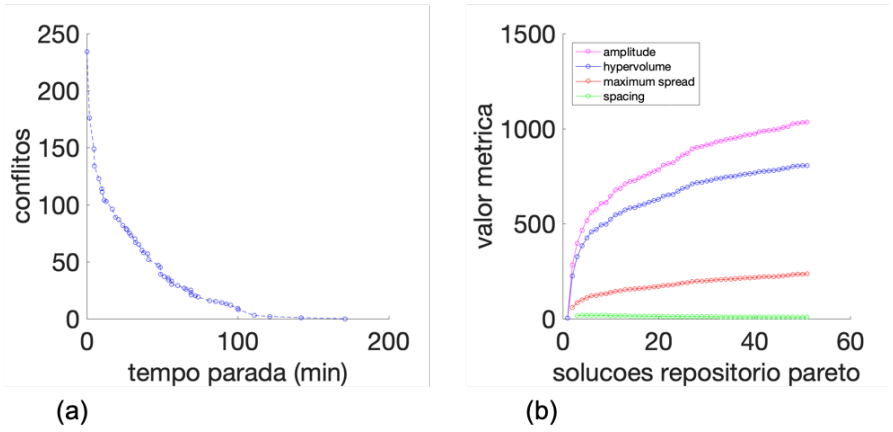

Figura 73 – Resultados para *184v222a*, *Cen–8*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

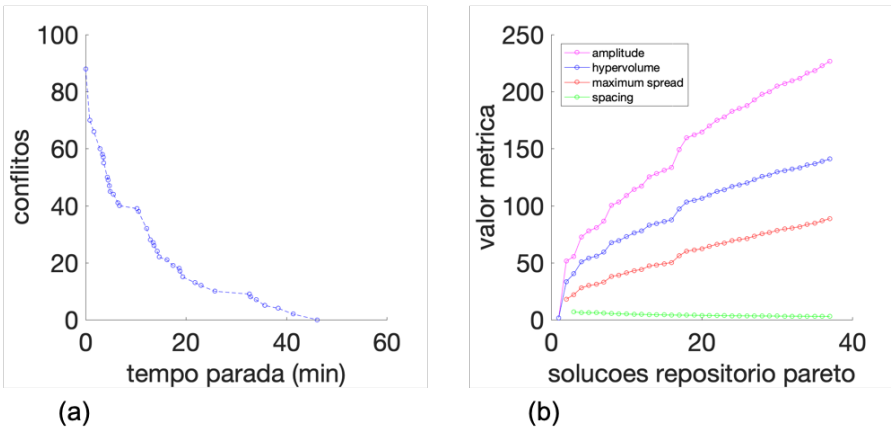

Figura 74 – Resultados para *274v332a*, *Cen–8*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.

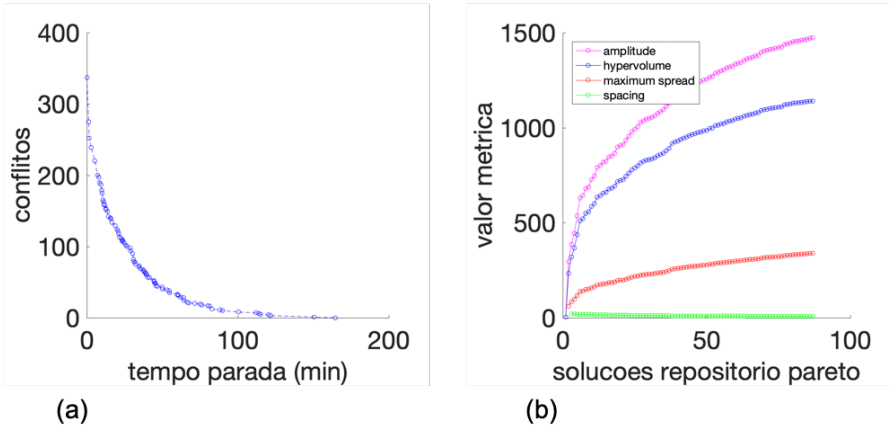

Figura 75 – Resultados para *364v442a*, *Cen–8*: (a) Fronteira de Pareto, (b) métricas para avaliação da Fronteira de Pareto.