UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Sincronização de Semáforos como um Problema de Otimização com Muitos Objetivos

Dissertação de Mestrado

Saulo Antonio de Lima Matos

São Cristóvão – Sergipe

2017

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Saulo Antonio de Lima Matos

Sincronização de Semáforos como um Problema de Otimização com Muitos Objetivos

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): Prof. Dr. André Britto de Carvalho

São Cristóvão – Sergipe

2017

Saulo Antonio de Lima Matos
Sincronização de Semáforos como um Problema de Otimização com Muitos Objetivos/ Saulo Antonio de Lima Matos. – São Cristóvão – Sergipe, 2017-
95 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. André Britto de Carvalho

Dissertação de Mestrado – UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO, 2017.

1. Otimização com Muitos Objetivos. 2. Sistemas de Transporte Inteligente. 3. Otimização Multiobjetivo. 4. Sincronização de Semáforos. 5. Redução de Dimensionalidade. I. Orientador Prof. Dr. André Britto de Carvalho. II. Universidade Federal de Sergipe. III. Programa de Pós-gradução em Ciências da Computação IV. Sincronização de Semáforos como um Problema de Otimização com Muitos Objetivos

CDU 02:141:005.7

Saulo Antonio de Lima Matos

Sincronização de Semáforos como um Problema de Otimização com Muitos Objetivos

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Prof. Dr. André Britto de Carvalho
Orientador

Prof. Dr. Hendrik Teixeira Macedo
Membro Interno

Prof. Dra. Aurora Trinidad Ramirez Pozo
Membro Externo

São Cristóvão – Sergipe
2017

Agradecimentos

Gostaria de agradecer a Universidade Federal de Sergipe, em especial ao Programa de Pós-Graduação em Ciência da Computação, por disponibilizar a estrutura de laboratório que tornou possível realizar os experimentos para a construção deste trabalho.

Agradeço a todos os docentes pelo conhecimento transmitido e aos colegas de curso.

E por último, agradeço ao meu orientador pela dedicação e apoio.

Resumo

Os Sistemas de Transporte Inteligente (ITS) têm como objetivo otimizar a eficiência do transporte e melhorar a sua segurança através do uso de tecnologia avançada. Nesse contexto de ITS uma das áreas importantes é a gestão de tráfego, que utiliza novos conceitos de organização e manutenção do tráfego, buscando, entre outros aspectos, manter um fluxo de tráfego de qualidade. Na gestão de tráfego está inserida a sincronização de semáforos, que é uma das abordagens que lida com a redução de congestionamento do tráfego. Uma sincronização é atingida quando mais de um semáforo está executando o mesmo tipo de plano semafórico de modo que permita um veículo passar pelos semáforos sem paradas. Como consequência da sincronização, é possível otimizar alguma qualidade relacionada ao tráfego, normalmente o fluxo de veículos. Porém, obter a sincronização de semáforos é um problema complexo e é necessária a busca automática por soluções. Com um simulador de tráfego, é possível construir uma representação computacional de uma combinação de semáforos e obter medidas (tempo de atraso, tempo de viagem, tempo parado, velocidade média global, entre outras) de qualidades do tráfego calculadas pelo próprio simulador. Assim, através da representação computacional e de um conjunto de medidas, podemos modelar o problema da sincronização de semáforos como um Problema de Otimização Multiobjetivo (MOP), que é a classe de problemas que possuem mais de uma função objetivo a ser otimizada. Dentro dessa classe de problemas, recentemente foi definida a Otimização com Muitos Objetivos, que busca resolver um MOP que possui um grande número de funções objetivo, geralmente com mais de três funções. No contexto da sincronização de semáforos, apesar de o problema ser modelado com um grande número de funções objetivo, trabalhos da literatura buscam otimizar apenas um pequeno subconjunto envolvendo no máximo duas funções. Assim, este trabalho propõe modelar e resolver o problema da sincronização de semáforos como um Problema de Otimização com Muitos Objetivos (MaOP). Na modelagem, o problema foi representado computacionalmente e foram escolhidas seis funções objetivo; para resolução do problema, foram aplicadas técnicas de otimização com muitos objetivos. Para modelar o MaOP, foi desenvolvido um sistema e utilizado o simulador *Simulation of Urban Mobility* (SUMO). A finalidade do sistema é realizar a comunicação entre diversas tarefas que estão incorporadas em módulos, sendo assim possível efetuar a comunicação entre um algoritmo de busca e o SUMO. Para resolver o MaOP, foi aplicado o algoritmo *Nondominated Sorting Genetic Algorithm III* (NSGA-III) e técnicas de redução de dimensionalidade, tornando possível modelar o problema com um número de objetivos reduzido. Neste trabalho, foi realizado um conjunto de experimentos, buscando analisar a performance dos algoritmos NSGA-II e NSGA-III em diferentes cenários com muitos objetivos. Os resultados mostraram que NSGA-II superou NSGA-III para o problema na maioria dos cenários e que as técnicas de redução de dimensionalidade foram eficazes.

Palavras-chave: Otimização com Muitos Objetivos, Sistemas de Transporte Inteligente, Otimização Multiobjetivo, Sincronização de Semáforos, Redução de Dimensionalidade.

Abstract

Intelligent Transportation Systems (ITS) aim to optimize transportation efficiency and improve safety through the use of advanced technology, encompassing different spaces of application. One of the primary ITS-related areas is traffic management, which uses new concepts in the organization and maintenance of traffic, while seeking to keep good-quality traffic flow, among other aspects. Included in traffic management is traffic light synchronization, which is one of the approaches dealing with the reduction of traffic congestion. Synchronization is achieved when two or more traffic lights are running the same types of traffic patterns so that a vehicle can pass through the synchronized lights without stopping. As a consequence of synchronization, it is possible to optimize a given traffic-related quality, usually the flow of vehicles. However, synchronizing a set of traffic lights across the road networks of a city is a complex problem that requires solutions in an automatic way. With the aid of a traffic simulator, it is possible to build a computational representation of a set of lights and obtain measures (delay time, travel time, stopped time, average global speed, among others) of traffic-related qualities calculated by the simulator itself. Thus, through the computational representation and a set of quality measures, the problem of traffic light synchronization can be modeled as a Multi-objective Optimization Problem (MOP). Optimization problems that have more than one objective function that must be optimized are called MOPs. Within this class of problems, there has recently been established the Many-Objective Optimization. This area seeks to solve an MOP that has a large number of objective functions, usually problems with more than three functions. In the context of traffic light synchronization, even though the problem is modeled with a large number of objective functions, other studies in the literature seek to optimize only a small subset involving a maximum of two functions. Thus, this study proposes to model and solve the problem of traffic light synchronization as a Many-Objective Optimization Problem (MaOP). In the modeling, the problem was computationally represented and six objective functions were chosen; to solve the problem, many-objective optimization techniques were applied. To model the MaOP, a system was developed and the simulator Simulation of Urban Mobility (SUMO) was used. The purpose of the system is to establish communication between several tasks that are incorporated in modules, so it is possible to communicate between the search algorithm and SUMO. To solve the MaOP, the Nondominated Sorting Genetic Algorithm III (NSGA-III) algorithm and dimensionality reduction techniques were applied, making it possible to model the problem with a reduced number of objectives. In this study, a set of experiments was carried out, aiming to analyze the performance of the NSGA-II and NSGA-III algorithms in different scenarios with many objectives. The results showed that NSGA-II surpassed NSGA-III for the problem in most scenarios, and that dimensionality reduction techniques were effective.

Keywords: Many-Objective Optimization, Intelligent Transportation Systems, Multi-objective Optimization, Traffic Light Synchronization, Dimensionality Reduction.

Lista de ilustrações

Lista de tabelas

Lista de códigos

Lista de abreviaturas e siglas

ITS	*Intelligent Transportation System*
MAB	Módulo Algoritmo de Busca
MaOP	*Many-Objective Optimization Problem*
MBD	Módulo Base de Dados
MJM	Módulo jMetal
MOEAs	*Multi-Objective Evolutionary Algorithms*
MOP	*Multi-Objective Optimization Problem*
MRD	Módulo Redução de Dimensionalidade
MSI	Módulo Simulador
NSGA-II	*Non-dominated Sorting Genetic Algorithm-II*
NSGA-III	*Non-dominated Sorting Genetic Algorithm-III*
SUMO	*Simulation of Urban Mobility*
XML	*Extensible Markup Language*

Sumário

1 Introdução

Com o crescente aumento populacional, sobretudo em áreas urbanas, surge a necessidade de criar infraestruturas inteligentes. Sem tais estruturas, a quantidade de tempo e esforço humano que seria gasto com manutenções e operações com os sistemas seria impraticável, sobretudo devido ao número de eventos de diferentes naturezas, que podem ocorrer simultaneamente em uma cidade. Nesse contexto surgem as cidades inteligentes, onde diversas tecnologias funcionam de forma sustentável e inteligente, integrando diferentes infraestruturas, serviços e dispositivos inteligentes, responsáveis por realizar monitoramento, coleta e controle (GIFFINGER et al., 2007).

Um dos grandes problemas que as cidades enfrentam é relacionado ao transporte e mobilidade, ambos fortemente ligados ao padrão de qualidade de vida. Os Sistemas de Transporte Inteligente (ITS, do inglês *Intelligent Transportation Systems*) têm como objetivo otimizar a eficiência do transporte e melhorar a sua segurança através do uso de tecnologia avançada para fornecer informações das estradas e serviços (SEREDYNSKI; MAZURCZYK; KHADRAOUI, 2013).

As tecnologias em ITS englobam diferentes espaços de aplicação. Na gestão de tráfego, novos conceitos de organização e manutenção do tráfego são utilizados. Como base para o processo de análise do tráfego, informações adequadas sobre a situação atual do tráfego têm de ser recolhidas. Outro aspecto importante na gestão de tráfego é manter um fluxo de tráfego de qualidade. Isto significa, em outras palavras, reduzir o congestionamento e o tempo de viagem dos veículos. Um dos caminhos para obter um fluxo de tráfego de qualidade é através da sincronização de semáforos.

Uma sincronização é atingida quando dois ou mais semáforos estão executando os mesmos tipos de planos semafóricos de modo que permita um veículo passar pelos semáforos sincronizados sem paradas, garantindo um tempo de espera mínimo, normalmente utilizando programas de otimização. Porém, obter a sincronização de um cojunto de semáforos de malhas

viárias em uma cidade é um problema complexo e é necessária a busca por soluções de forma automática, pois até mesmo para um único cruzamento pode não existir uma solução óbvia.

É possível utilizar simuladores de tráfego para representar uma malha viária e obter, através de um computador, possíveis medidas (tempo de atraso, tempo de viagem, tempo parado, velocidade média global, entre outras) de qualidade do tráfego. Com o auxílio do simulador, é possível construir uma representação computacional de uma combinação de semáforos e obter medidas de qualidades do tráfego calculadas pelo próprio simulador. Assim, através da representação computacional e de um conjunto de medidas de qualidade, é possível modelar o problema da sincronização de semáforos como um problema de otimização.

Vários trabalhos na literatura lidam com o problema da sincronização de semáforos utilizando algoritmos de otimização e simulação. Em (SHEN; WANG; Y., 2013), foi utilizado o algoritmo NSGA-II (do inglês *Non-dominated Sorting Genetic Algorithm* II) para, a partir do tempo dos semáforos, otimizar o tempo médio de atraso e o tempo médio de parada. No trabalho (KWASNICKA; STANEK, 2006) é proposto um método de otimização de semáforos com base em um algoritmo genético visando minimizar perdas de tempo durante a viagem e maximizar a velocidade média dos veículos. Em (GARCÍA-NIETO; OLIVEIRA; ALBA, 2013), através de uma abordagem de otimização, um PSO (do inglês *Particle Swarm Optimization*) obteve ciclos de semáforo que otimizaram o número de veículos que atingem o seu destino e o tempo total de viagem.

No problema da sincronização de semáforos há várias medidas para se otimizar; as mais utilizadas na literatura são: o tempo de atraso (médio e total), o tempo parado (médio e total), o tempo de viagem, o número de veículos que chegam ao seu destino, a velocidade média global e o fluxo de veículos. Problemas com mais de um objetivo são classificados como Problemas de Otimização Multiobjetivo. Algoritmos de otimização evolucionária multiobjetivo geralmente funcionam muito bem nesses problemas, no entanto, a sua performance de busca é severamente deteriorada pelo aumento do número de objetivos. Problemas Multiobjetivo com quatro ou mais objetivos são chamados de Problemas com Muitos Objetivos (ISHIBUCHI; TSUKAMOTO; NOJIMA, 2008).

Apesar dos trabalhos encontrados na literatura terem resolvido o problema da sincronização de semáforos como um problema de otimização multiobjetivo, nenhum deles tratou como um problema com muitos objetivos. Como há diversas medidas obtidas na modelagem do problema é possível tentar otimizá-las ao mesmo tempo caracterizando um problema com muitos objetivos.

1.1 Objetivos

O presente trabalho tem como objetivos principais modelar e resolver o problema da sincronização de semáforos como um Problema de Otimização com Muitos Objetivos. Na mode-

lagem, o problema será representado computacionalmente e serão utilizadas seis funções objetivo: $Depart\ delay$, $Trip\ duration$, $Wait\ steps$, $Time\ loss$, $CO_2\ abs$ e $fuel\ abs$; para resolução do problema, serão aplicadas técnicas de otimização com muitos objetivos.

Para modelar o MaOP, será desenvolvido um sistema e utilizado o simulador SUMO. A finalidade do sistema é realizar a comunicação entre diversas tarefas que estão incorporadas em módulos, sendo assim possível efetuar a comunicação entre um algoritmo de busca e o SUMO. Para resolver o MaOP, serão aplicados o algoritmo NSGA-III e técnicas de redução de dimensionalidade, tornando possível modelar o problema com um número de objetivos reduzido.

Para alcançar esse objetivo, alguns objetivos específicos foram definidos:

- Modelar o problema da sincronização de semáforos como um problema de otimização;
- Desenvolver um sistema que possibilite a comunicação entre um algoritmo de busca e o SUMO, além de comunicar-se com os demais módulos existentes;
- Aplicar técnicas de Otimização com Muitos Objetivos;
- Efetuar a análise de desempenho dos algoritmos e das técnicas de Otimização com Muitos Objetivos através de um *benchmark* envolvendo cenários com diferentes números de objetivos.

1.2 Metodologia

Para atingir esses objetivos, primeiro foi necessário fazer uma revisão bibliográfica de trabalhos que modelam o problema da sincronização de semáforos através de técnicas computacionais e de trabalhos que aplicam técnicas de redução de dimensionalidade a problemas com muitos objetivos. Em paralelo, foi feito um levantamento dos principais simuladores de tráfego existentes. Então, foi feita uma revisão bibliográfica apenas dos trabalhos que utilizam algoritmos de otimização para resolver o problema.

No segundo momento, foi necessário modelar o problema dos semáforos como um problema de otimização. Então, a partir da revisão bibliográfica, foi definido o uso do simulador de tráfego SUMO e desenvolvido um sistema. O sistema funciona em conjunto com cinco módulos principais, onde diversas tarefas se comunicam. Os algoritmos de busca utilizados foram incorporados através do *framework* jMetal (JUAN; ANTONIO, 2011) e, então, foi desenvolvido um módulo intermediário que integrasse o jMetal ao processo. Para reduzir o número de simulações, foi utilizado o sistema de gerenciamento de banco de dados (SGBD) MySQL (MYSQL, 2017).

Para a produção da análise do desempenho dos algoritmos, foi definido um *benchmark* contendo dois cenários, onde cada cenário é representado por uma rede e três fluxos de veículos: baixo, médio e alto. Por fim, executou-se um conjunto de experimentos, buscando analisar a

performance dos algoritmos NSGA-II e NSGA-III em diferentes cenários com muitos objetivos e buscando identificar se a aplicação de técnicas de redução de dimensionalidade melhora o desempenho desses algoritmos.

1.3 Estrutura do Documento

Este documento está estruturado em capítulos e seções, que são:

- Capítulo 2 - *Intelligent Transportation System* (ITS): apresenta os conceitos principais relacionados aos ITS, a gestão de tráfego com foco na sicronização de semáforos e as tecnologias utilizadas com foco nos simuladores;
- Capítulo 3 - Otimização com Muitos Objetivos: apresenta os principais conceitos a respeito da otimização multiobjetivo e com muitos objetivos, juntamente a dois algoritmos bastante utilizados na literatura. Depois apresenta os principais conceitos a respeito da redução de dimensionalide e trabalhos relacionados ao tema. Por fim apresenta os trabalhos correlatos para a solução do problema da sincronização de semáforos;
- Capítulo 4 - Sincronização de Semáforos como um Problema de Otimização com Muitos Objetivos: apresenta o sistema proposto para resolução do problema;
- Capítulo 5 - Experimentos: descreve o *benchmark* utilizado nos experimentos, mais detalhes do simulador, os parâmetros dos algoritmos e os resultados dos experimentos;
- Capítulo 6 - Conclusão: E finalmente, são apresentadas as conclusões, as limitações do trabalho e os possíveis trabalhos futuros.

2

Intelligent Transportation System (ITS)

Neste capítulo são abordados os conceitos principais relacionados aos sistemas de transporte inteligente (seção 2.1). Depois, é abordada a gestão de tráfego (seção 2.2) com foco na sicronização de semáforos. Por fim, será apresentado os principais conceitos de simuladores (seção 2.3) e os principais softwares existentes, com foco no simulador utilizado neste trabalho (subseção 2.3.1).

2.1 Definição

O sistema de transporte é um componente fundamental da infraestrutura das cidades, oferecendo oportunidades para a circulação de pessoas e bens, e das suas ligações entre locais de vida, de trabalho e lazer. O Sistema de Transporte Inteligente (mais conhecido pela sigla ITS) tem como objetivo otimizar a eficiência do transporte e melhorar a sua segurança através do uso de tecnologia avançada, fornecendo informações das estradas e serviços (SEREDYNSKI; MAZURCZYK; KHADRAOUI, 2013). Tais sistemas permitem que os usuários estejam bem informados e tomem melhores decisões relacionadas a viagem. Segundo An, Lee e Shin (2011), ITS é um sistema que integra informação, comunicações, computadores e outras tecnologias e os aplica no campo do transporte para construir um sistema integrado de pessoas, estradas e veículos, utilizando tecnologias de comunicação de dados avançadas.

Devido à grande integração e quantidade de serviços, não é incomum confundir serviços de ITS com os de cidade inteligente (do inglês *Smart City*), onde o ITS é um componente fundamental. Na cidade inteligente, vários sistemas inteligentes estão trabalhando em conjunto para criar um novo contexto dos serviços que podem ser utilizados ou consumidos por muitas pessoas, entidades governamentais ou organizações privadas (HERRERA-QUINTERO et al., 2015). As soluções para estes problemas compreendem melhorias para a maioria dos componentes da dinâmica urbana (ilustrado na Figura 1), tais como saúde, serviços, transporte (de pessoas, cargas,

etc), mobilidade, distribuição eficiente e econômica de água e energia, que são fundamentais para obtenção de um alto padrão de vida (HANCKE; SILVA; JR, 2013).

Figura 1 – Detecção em cidades inteligentes

.

Fonte: Adaptado de Gerhard et al. (2013)

Hoje, há ampla gama de invenções e tecnologia de ponta no setor, no entanto, existem vários problemas para resolver, como congestionamento de tráfego, abastecimento de demanda, mobilidade, entre outros (CAÕN-LOZANO et al., 2013). No que tange às diversas soluções e tecnologias utilizadas em ITS, mais detalhes podem ser vistos, respectivamente, nos Apêndices A e B. Basicamente, a arquitetura de ITS está fundamentada na interação de três camadas de infraestrutura:

- Camada de Transportes: é composta pela infraestrutura física de ITS, contendo os usuários, veículos, centros de controle e equipamentos viários;
- Camada de Comunicações: é composta pela infraestrutura de informações que conecta todos os elementos da camada de transportes. É a camada que dá a característica de "sistema", propiciando coordenação e compartilhamento de informações entre sistemas e pessoas. A arquitetura descreve cuidadosamente quais os tipos de informação e de comunicação são necessários aos vários serviços de ITS, como os dados devem ser compartilhados (e usados) por entidades físicas (subsistemas) e quais as entidades irão participar. Descreve, também, quais os tipos de padrão são necessários para facilitar este compartilhamento;

- Camada Institucional: é composta pelas organizações e regras sociais que definem as fronteiras institucionais e os papéis dos organismos governamentais, empresas privadas, associações de usuários e outros participantes no contexto dos serviços de ITS. As atividades deste nível incluem o desenvolvimento de uma política local, o financiamento de ITS e a criação de parcerias que direcionem o desenvolvimento de ITS. Neste caso, a arquitetura recomenda quem deve estar conectado com quem e quais os tipos de informação que devem ser trocadas.

Quanto aos domínios de serviços em ITS, podemos exemplificar alguns:

- Gestão do tráfego e operações;
- Serviços de veículos;
- Transporte de mercadorias;
- Transporte público;
- Serviço de emergência;
- Sistemas avançados de segurança veicular;
- Transporte rodoviário relacionados com segurança pessoal;
- Monitoramento do tempo e de condições ambientais;
- Gestão e coordenação de resposta a desastres;
- Segurança nacional;
- Informações do "viajante" (carro, pessoa, ônibus, trem, avião, etc.);
- Serviços de pagamento eletrônico (transporte público, estacionamento, pedágio, etc);
- Gerenciamento de dados;
- Gestão de desempenho em ITS.

2.2 Gestão de Tráfego

A gestão de tráfego utiliza novos conceitos de organização e manutenção do tráfego. Como base para o processo de influenciar o tráfego, informações adequadas sobre a situação atual do tráfego têm de ser recolhidas. As tarefas podem ser divididas em diferentes partes: o desenvolvimento de métodos de monitoramento de tráfego, simulação e previsão de tráfego, e desenvolvimento de métodos para influenciar as operações e a qualidade de tráfego no transporte.

Outro aspecto importante na gestão de tráfego é manter um fluxo de tráfego de qualidade. Isto significa, em outras palavras, reduzir o congestionamento e o tempo de viagem dos veículos. Um dos caminhos para obter um fluxo de tráfego de qualidade é através da sincronização de semáforos, garantindo um tempo de espera mínimo, normalmente utilizando programas de otimização. Por outro lado, otimização semafórica, em geral, é um problema muito complexo e até mesmo para um único cruzamento pode não existir uma solução ótima óbvia.

2.2.1 Sincronização de Semáforos

Um semáforo pode ser classificado como sendo toda a instalação de controle existente em um cruzamento, incluindo os sinais luminosos, os fios, os instrumentos de controle, etc (LEITE, 1980). As grandezas temporais mais importantes dos semáforos são o intervalo, o ciclo e a fase. O intervalo é o período de tempo em que todas as indicações do semáforo permanecem estáticas, o ciclo é a sucessão de indicações no semáforo que se repetem periodicamente, e a fase é uma parte do ciclo dedicada a um conjunto de movimentos que recebem o direito de passagem simultaneamente. Em cada fase do semáforo é assegurado aos veículos um conjunto de movimentos dentro da via. Esses movimentos poderão ser o direito de passar pelo cruzamento no mesmo sentido, ou a realização de conversão.

A sincronização de semáforos é uma das abordagens que lida com o problema de redução de congestionamento de tráfego. E consequentemente minimiza outros fatores como a emissão de gases (por exemplo o dióxido de carbono e o hidrocarboneto) e o consumo de combustível. A sincronização é atingida quando dois ou mais semáforos estão executando os mesmos tipos de planos semafóricos de modo que permita um veículo passar pelos semáforos sincronizados sem paradas. Para que a sincronização seja possível, é necessário que haja uma defasagem (do inglês *offset*) relativa ao semáforo anterior (OLIVEIRA, 2005).

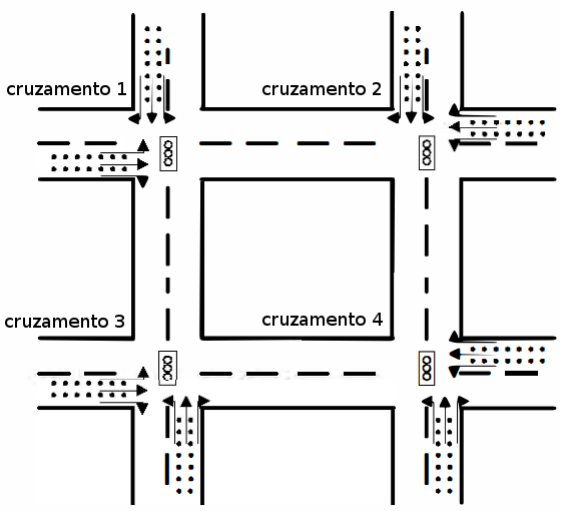
Porém, nem sempre a sincronização vai garantir que os veículos passem pelos semáforos abertos, mesmo quando há poucos semáforos. Isso ocorre devido a diversos fatores, o principal é a quantidade de veículos que se aproximam dos cruzamentos por diversas direções. Bons planos semafóricos coordenados são muito trabalhosos de serem criados. De acordo com (ROBERTSON; BRETHERTON, 1991), uma rede com 30 a 40 semáforos demanda um ano de trabalho-homem. Para uma melhor compreensão do processo da sincronização de semáforos é importante definir alguns conceitos a respeito do funcionamento dos semáforos:

- Tempo de fase: um intervalo de tempo de ciclo (tempo relativo de verde) reservado para qualquer movimento de tráfego (estágio);
- Estágio: é a sequência de diferentes configurações de direções em um período de tempo dentro do ciclo onde não há mudança de cores;

- Plano semafórico: é o conjunto de elementos que caracteriza a programação da sinalização semafórica para um cruzamento ou seção de via, em um determinado período do dia. O plano semáforico é um conjunto único de fases;
- Defasagem: tempo necessário para percorrer o espaço entre os dois semáforos adjacentes, que estejam sincronizados, a uma velocidade média;
- Tempo de ciclo: tempo levado para uma sequência completa das indicações do sinal, sendo que pode ter um tempo variável ou fixo. No tráfego lotado, ciclos mais longos levam a um melhor desempenho;
- Ciclo do cruzamento: é definido como o período de tempo levado até que todos os grupos semafóricos já tenham, ao menos uma vez, obtido o sinal verde (estado "aberto") em algum ponto;
- Defasegem do ciclo: define o tempo de início de um ciclo em relação a outros semáforos. A defasagem pode ser ajustada para que alguns semáforos "cooperem" para, por exemplo, geraram uma "Onda Verde".

A Figura 2, mostra um exemplo com quatro semáforos e quatro cruzamentos. Existe seis direções possíveis a seguir, porém nem todas direções são permitidas ao mesmo tempo. Essas direções são controladas pela sequência de fases ilustradas na Figura 3.

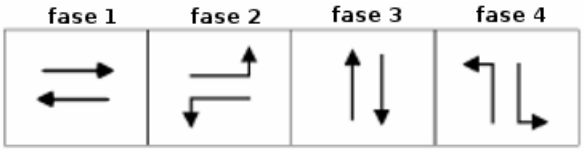
Figura 2 – Uma rede de tráfego com quatro cruzamentos



Fonte: Adaptado de Shen et al. (2013)

Basicamente existem dois tipos de temporização em sistemas de controle de tráfego urbano: fixo ou adaptativo. A sincronização com temporização fixa é a mais utilizada e a mais simples, já que não requer a instalação de detectores de tráfego. Os detectores são circuitos

Figura 3 – Sequência de fases dos semáforos



Fonte: Adaptado de Shen et al. (2013)

eletrônicos digitais que quando conectados a uma bobina, laço detector ou loop, compõem um conjunto capaz de acusar (detectar), eletricamente, a presença de uma massa metálica (veículo) no campo de influência da antena. A temporização que percebe o fluxo de tráfego e adapta-se a ele exige a instalação de detectores e de um sistema de controle computadorizado mais complexo. O meio termo entre esses dois tipos de sistema também é utilizado. O Quadro 1 resume as estratégias da sincronização, mostrando as diferenças na estrutura necessária, custo e desempenho de hardware de cada uma delas.

Quadro 1: Estratégias de controle de semáforos

Abordagem	Estrutura	Custo	Desempenho
Sincronização com temporização fixa	Não computadorizada	Baixo	Baixo a médio
Sincronização com temporização fixa	Computadorizada e Centralizada	Médio	Baixo a médio
Sincronização em resposta ao tráfego com seleção ou modificação de planos semafóricos	Computadorizada e centralizada	Médio a alto	Médio a alto
Sincronização com controle completamente distribuído	Distribuída	Médio a alto	Alto

Fonte: Oliveira (2005)

2.2.2 Complexidade na Sincronização de Semáforos

No trânsito é normal que ocorram situações previsíveis e imprevisíveis, onde ambas podem modificar o cenário do tráfego. Padrões de tráfego podem ser afetados por acidentes, inundações, neve, etc. Determinar a priori os planos semafóricos apropriados para os diferentes momentos de um dia é uma tarefa complexa que exige uma grande quantidade de conhecimento sobre o fluxo de tráfego dinâmico (OLIVEIRA; BAZZAN, 2006). Este fluxo, pode por muitas vezes variar de forma aleatória. Por conta disso, os planos semafóricos deverão se adequar não só aos outros semáforos, mas a diferentes fluxos de tráfego em diferentes horários do dia e situações. Essa volatilidade torna a sincronização de semáforos ainda mais complexa. Além disso, de acordo com Diakaki (DIAKAKI; PAPAGEORGIU; Y., 2002), existem quatro possibilidades de influenciar as condições do tráfego utilizando semáforos:

- Especificação do estágio: para cruzamentos complexos, a especificação do número de estágios pode ter impacto na eficiência do cruzamento;
- Tempo de fase: o tempo de verde de cada estágio deve ser dimensionado de acordo com a demanda das faixas envolvidas;
- Tempo ciclo: tempos de ciclo maiores geralmente aumentam a capacidade do cruzamento, por outro lado, tempos de ciclo maiores aumentam os tempos de espera em cruzamentos subsaturados;
- Defasagem: a especificação da defasagem ótima deve levar em conta a existência de possíveis filas e a velocidade média dos veículos.

Definir qual o melhor plano a depender do fluxo de veículos é a uma estratégia interessante e faz sentido ser a primeira a se preocupar. Porém, essa visão não é a única e não necessariamente a mais importante. Dentro do problema da sincronização, há de se pensar como definir que qualidade fará uma sincronização (leia-se temporização dos semáforos envolvidos) melhor do que outra. Se muitos veículos estão tendo que parar por diversos momentos (seja por engarrafamentos ou por que quando chegam aos cruzamentos encontram o sinal vermelho) então a sincronização utilizada não é a melhor. Assim, as medidas de qualidade de uma sincronização de semáforos a priori são: fluxo de veículos e o numéro de veículos parados.

Somadas as situações (previsíveis e imprevíveis) que podem ocorrer, as possibilidades que podem influenciar as condições de tráfego (os tempo de fase, de ciclo, etc) e as duas medidas de qualidade citadas acima (fluxo e número de veículos parados) nos leva a perceber que o processo de otimização pode ser muito complexo, mesmo para um único cruzamento, e cresce exponencialmente com o número de cruzamentos controlados devido ao número de variáveis. A otimização de tempo dos semáforos é um problema NP-Completo e nenhuma técnica de otimização clássica pode resolvê-lo em uma quantidade razoável de tempo (WÜNSCH; GREGOR, 2008 apud HAJBABAIE; BENEKOHAL, 2015), (HAJBABAIE, 2012 apud HAJBABAIE; BENEKOHAL, 2015).

Posto isso, é necessário definir quais medidas de qualidade serão utilizadas. Mas, antes, é necessário definir como obtê-las e compará-las. Para estimar diversas medidas relacionadas ao tráfego e obter medidas relacionadas aos veículos e então compará-las, utiliza-se simuladores de tráfego. Com os simuladores é possível medir não só fluxo de veículos e o número de veículos parados, mas também outros dados, tais como:

- Total de veículos que entraram na rede; total de veículos que deixaram a rede; total de veículos que não entraram na rede;
- Total de veículos que passaram por um detector; a frequência com que veículos passaram por um detector;

- Total de viagens concluídas;
- Tempo de viagem ideal. Obtem-se esse valor "relaxando" as restrições de fluxo no trajeto da viagem, ou seja, sem tráfego concorrente e sem paradas;
- Tempo de viagem do veículo; tempo médio de viagem;
- Velocidade média de um veículo; velocidade média global;
- Velocidade média em uma via;
- Tempo de atraso de um veículo. Subtrai-se: o tempo de viagem, do tempo de viagem ideal;
- Velocidade técnica média de um veículo. Obtem-se excluindo o tempo de atraso;
- Velocidade técnica média global;
- Tempo de atraso global. Soma-se todos os tempos de atraso;
- Tempo que um veículo ficou parado na via; total de paradas do veículo na via; total de paradas de todos os veículos na via;
- Tempo que um veículo ficou parado na rede; total de paradas do veículo na rede; total de paradas de todos os veículos na rede;
- Tempo médio de atraso global; tempo médio de atraso dos veículos com a mesma rota (viagens idênticas);
- Total de veículos parados em uma via, em toda a rede e em uma via;
- Comprimento da (s) fila (s) numa via; Soma-se o total de veículos parados na via;
- Comprimento da (s) fila (s) em um cruzamento. Soma-se o total de veículos parados nas vias direcionadas a este cruzamento;
- Número de filas (lotadas ou não);
- Tempo de verde ideal. Subtrai-se: o tempo de atraso, do ciclo total, em um cruzamento, em um determinado intervalo de tempo.
- Consumo de combustível de um veículo, global e em uma via;
- Emissão de ruído de um veículo, global e em uma via;
- Emissão de monóxido de carbono (CO) de um veículo, global e em uma via;
- Emissão de dióxido de carbono (CO_2) de um veículo, global e em uma via;
- Emissão de hidrocarboneto (HC) de um veículo, global e em uma via;

- Emissão de óxidos de nitrogênio (NO_x) de um veículo, global e em uma via;
- Emissão de partículas (PM_X) de um veículo, global e em uma via.

Na prática, utilizam-se simuladores para que a partir dessas várias medidas defina-se o quão boa é a sincronização. Com as diversas medidas em mãos e a possibilidade de simular por diversas vezes, é possível repetir o processo até que se chegue a uma sincronização tão boa quanto se deseje. A partir da otimização dessas medidas é possível, por exemplo: reduzir o tempo de viagem, reduzir o tempo espera, aumentar a velocidade média, aumentar o fluxo de veículos, reduzir a emissão de poluentes, reduzir o número de paradas, etc.

2.3 Simuladores

Os simuladores foram desenvolvidos devido à crescente demanda por melhoria do tráfego viário, principalmente por conta do aumento da frota de veículos, sobretudo nas grandes cidades. Simulações de tráfego são importantes para os pesquisadores, bem como para os profissionais da área de transporte. Com os simuladores é possível predizer como diversos pontos do tráfego em uma cidade se comportariam. Para isso devem ser definidos de forma fiel os valores (e o funcionamento) referentes aos agentes envolvidos no tráfego, que normalmente são: os veículos (sua velocidade e rotas), a rede viária (suas vias e direções) e os semáforos com seus respectivos tempos de duração para cada direção.

Segundo Machado (2009), os resultados gerados por simulação podem ser úteis a entidades governamentais na análise de risco da construção de redes viárias, de modo a garantir o máximo conforto e segurança aos seus futuros utilizadores. A previsão e o impacto que as novas redes viárias terão no tráfego, assim como os níveis de poluição ambiental gerados pelos veículos, são outras das vantagens da realização de simulações. Além disso, devido as características dos simuladores variarem muito entre os diferentes projetos, diversos simuladores podem ser utilizados.

O TRANSYT (*Traffic Network Study Tool*) é um dos programas de temporização de tempos de fase mais utilizado e mais antigo. É executado de maneira offline para determinar o tempo fixo ótimo de semáforos coordenados em qualquer rede de vias na qual o fluxo médio de veículos seja conhecido, utilizando modelos macroscópicos e determinísticos de simulação. Os critérios de otimização utilizados são: tamanho da fila, maximização do tamanho da "banda" da onda verde e quantidade de paradas. O programa otimiza as fases e as defasagens relativas dado um conjunto de tempos de ciclo realizando diversas interações entre o módulo de simulação de tráfego e o módulo de otimização de semáforos (OLIVEIRA, 2005).

O SCOOT (*Split Cycle and Offset Optimization Technique*) é um modelo centralizado de controle de tráfego desenvolvido pelo Transportation Road Research Laboratory no Reino Unido. O SCOOT usa detectores instalados nas vias para medir perfis do fluxo de tráfego em tempo

real e, juntamente com tempos de percurso e graus de saturação (ocupação relativa à capacidade nominal da via) pré-determinados, prediz filas em intersecções (ROBERTSON; BRETHERTON, 1991).

O SCATS (*Sydney Coordinated Adaptive Traffic System*) foi inicialmente desenvolvido na Austrália para aplicação em Sydney e em outras cidades australianas. Atualmente está instalado em mais de 50 cidades no mundo. É um sistema dinâmico de controle de semáforos com uma arquitetura descentralizada. A otimização do sistema se dá através de mudanças no tamanho do ciclo da fase e tempo de defasagem (LOWRIE, 1982).

O VISSIM (*Verkehr In Stadten - SIMulationsmodell*), desenvolvido pela *PTV Planung Transport Verkehr AG* (PTV GROUP, 2009), permite a simulação de vários tipos de agentes, tais como veículos, transportes públicos, bicicletas e pedestres. Possui opções de apresentação de simulações em três dimensões e um modelo realista do comportamento dos pedestres, tanto em transportes públicos como nas redes viárias.

O *TransModeler* (C. CORPORATION. TRANSMODELER, 2009), desenvolvido pela *Caliper Corporation*, se destaca pela sua versatilidade e fidelidade na análise em detalhe de diferentes redes viárias, desde zonas urbanas a autoestradas. Possui um ambiente GIS (do inglês *Geographic Information System*) em duas dimensões e três dimensões, útil para ilustrar dinâmicas de fluxo de tráfego, mudança do sinal dos semáforos, etc. O *TransModeler* permite a importação de dados de outros simuladores de tráfego, tais como o CORSIM (*CORredor SIMulation*) (MCTRANS CENTER, 2009) e *Sim Traffic* (TRAFFICWARE, 2009), e possui vários estudos de caso para situações de emergência.

Desenvolvido pela *McTrans Moving Technology*, o CORSIM possibilita a simulação de redes (*NETSIM*) e simulação de autoestradas (*FRESIM*). Dentro do pacote de software do CORSIM encontra-se um ambiente de desenvolvimento que permite a análise e gestão de operações de tráfego, a ferramenta TSIS (do inglês, *Traffic Software Integrated System*).

O *Sim Traffic*, implementado pela *TrafficWare*, possui cenários de redes urbanas em três dimensões, com tabelas de emissões de gases e acelerações atualizadas em tempo real durante a simulação. Foi desenhado para modelar redes com cruzamentos controlados, ou não, por semáforos. Gera relatórios relativos a um único cruzamento, faixa, zona ou trajetórias de veículos, exibindo vários indicadores, tais como densidade e ocupação de determinada via.

Possuindo poderosa plataforma integrada, o *Paramics* (Q. PARAMICS, 2009) produzido pela *Quadstone Paramics*, contém módulos que colocam este software como um dos melhores simuladores no mercado. *Paramics Modeller* fornece, através de uma GUI (do inglês *Graphical User Interface*), a possibilidade de criar, simular e visualizar modelos de tráfego. *Paramics Monitor* calcula os níveis de poluição ambiental gerados pelos veículos. Também é possível desenvolver novas funções com o *Paramics Programmer*. E ainda possui as ferramentas *Paramics Analyser*, *Paramics Processor*, *Designer* e os módulos *Paramics Estimator* e *Converter*.

2.3.1 *Simulation of Urban MObility*

O SUMO (do inglês *Simulation of Urban Mobility*) (D.L.R, 2017), é desenvolvido principalmente por funcionários do Instituto de Sistemas de Transporte do Centro Aeroespacial da Alemanha e teve início no ano de 2000. De acordo com o instituto e com a documentação, o SUMO oferece muitos recursos,

- Simulação microscópica: veículos, pedestres e transporte público são modeladas explicitamente;
- Simulação de tráfego multimodal, por exemplo, veículos, transporte público e pedestres;
- Horários de semáforos podem ser importados ou gerado automaticamente;
- Sem limitações artificiais em tamanho de rede e número de veículos simulados;
- Formatos de importação suportados: OSM (do inglês *Open Street Map*) (OPENSTREETMAP FOUNDATION, 2017), VISUM, VISSIM, NavTeq;
- É implementado em C ++ e usa apenas bibliotecas portáteis;
- Interação online: controlar a simulação com TraCI (do inglês *Traffic Control Interface*), dando a possibilidade de interação com programas externos. Assim, é possível implementar um sistema de controle adaptativo de semáforos através, por exemplo, de um *script Python* que inicia, controla e encerra a simulação através da interface TraCI.

e os seguintes módulos:

- *NETCONVERT*: criar (ou importar) redes viárias;
- *GUISIM*: simulação com interface gráfica para o usuário;
- *JTRROUTER*: geração de rotas com base em taxas de conversão à direita ou à esquerda em cada junção;
- *DUAROUTER*: geração de rotas usando dynamic user assignment. Técnica de melhor aproveitamento da rede, simulando que alguns motoristas conheçam atalhos;
- *DFROUTER*: geração de rotas usando fluxos dos detectores;
- *NETGEN*: criação de redes viárias simples a um nível menos detalhado e mais abstrato;
- *POLYCONVERT*: importação de polígonos (mapa viário) de outros formatos;
- *OD2TRIPS*: conversor de matrizes Origem-Destino (O/D) para viagens.

O SUMO requer vários arquivos de entrada que contêm informações sobre o tráfego e a rede viária a ser simulado. A rede viária pode ser importada a partir de mapas digitais populares tais como OSM e posteriormente convertidos para uma rede SUMO válida por meio de uma série de *scripts* fornecidos no pacote SUMO (KRAJZEWICZ; BONERT; WAGNER, 2006). O arquivo que representa a rede (*file.net.xml*) é um arquivo XML (do inglês *eXtensible Markup Language*). Para gerá-lo é necessário configurar três outros arquivos XML que irão formar a rede. Esses arquivos representam os nós (*file.nod.xml*), as arestas (*file.edg.xml*) e as conexões (*file.con.xml*) entre as arestas. Por fim, através do módulo *NETCONVERT*, é possível gerar a rede viária.

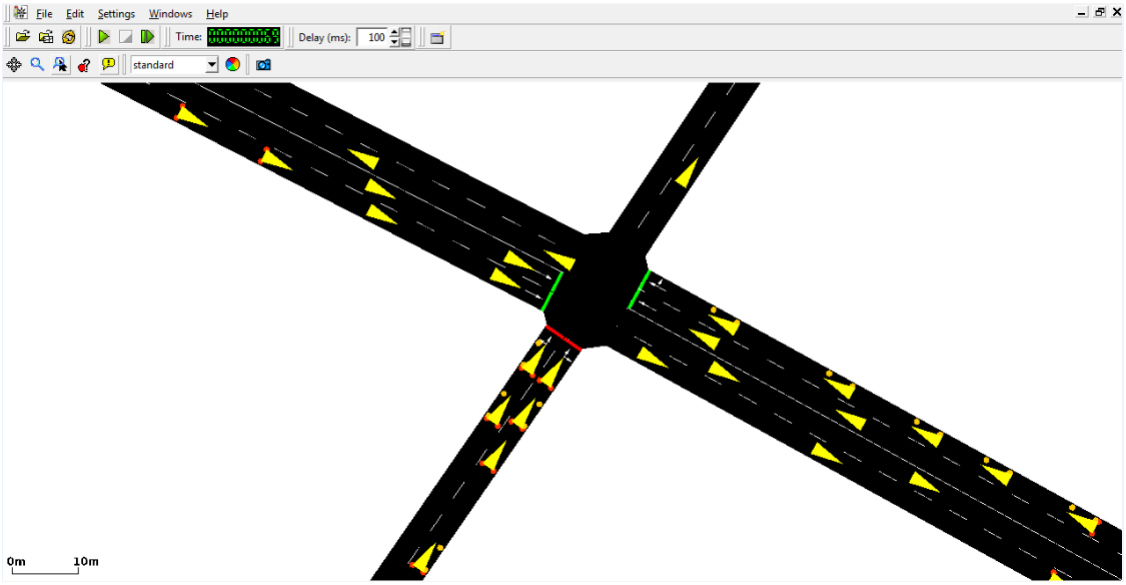
Em seguida ao processo de gerar a rede viária é feito o processo de atribuição de veículos que percorrerão as vias. Como o SUMO foi projetado para simular uma população hipotética que consiste de veículos com distintas rotas e explícitos tempos de saída, o modelo de demanda nativo do SUMO é uma lista de veículos, com tempos de saída e dois pontos no mapa para origem e destino, chamado de *trip* (viagem). Uma viagem (*trip*) é um movimento de veículo de um local para outro definido por: a aresta (via) de partida, a aresta de destino e a hora de partida. Uma rota é uma "longa" viagem; isso significa que a rota contém não só a primeira e última arestas, mas também todas as arestas que o veículo vai passar. Estas rotas são armazenadas em um arquivo de demanda (*file.rou.xml*) e podem ser geradas através de um gerador de rotas do SUMO, ou por rotas importadas de outros software, ou por rotas definidas pelo usuário (GARCÍA-NIETO; OLIVEIRA; ALBA, 2013).

Dentre as formas que o usuário pode atribuir essas viagens, estão a atribuição utilizando *flows* e a atribuição utilzando *trips*. Ao atribuir uma demanda de veículos, se faz necessário que um ou mais tipos de veículos sejam criados e com isso algumas características podem ser informadas, tais como: tipo, velocidade, taxa de aceleração e desaceleração, tamanho, cor, entre outras. Essas definições podem ser informadas em um arquivo separado, no arquivo de *flows* (*file.flows.xml*) ou no arquivo de *trips* (*file.trips.xml*).

A diferença na forma como os veículos são emitidos na simulação é que, na *trip*, será informada para cada veículo o momento da simulação em que ele deve partir e a sua rota. Já no *flow*, será informado quantos veículos serão emitidos em um dado intervalo de tempo. Por exemplo, dado um intervalo entre o início (cinquenta segundos) e o fim (cento e cinquenta segundos), se o número de veículos for cinquenta, então será emitido um veículo a cada dois segundos. A Figura 4 mostra um dado momento da simulação (na GUI do SUMO) em um cruzamento simples.

Os dados referentes às informações das viagens são obitidos como saída de uma simulação SUMO e ficam armazenados em um arquivo de informações de viagem (*tripinfo.xml*) que contém informações sobre horário de partida de cada veículo, o tempo que o veículo esperou para começar (*offset*), o tempo que o veículo chegou ao destino, a duração da sua jornada e o número de etapas em que a velocidade do veículo era inferior a $0.1m/s$ (paradas temporais na

Figura 4 – GUI do SUMO em dado momento da simulação.

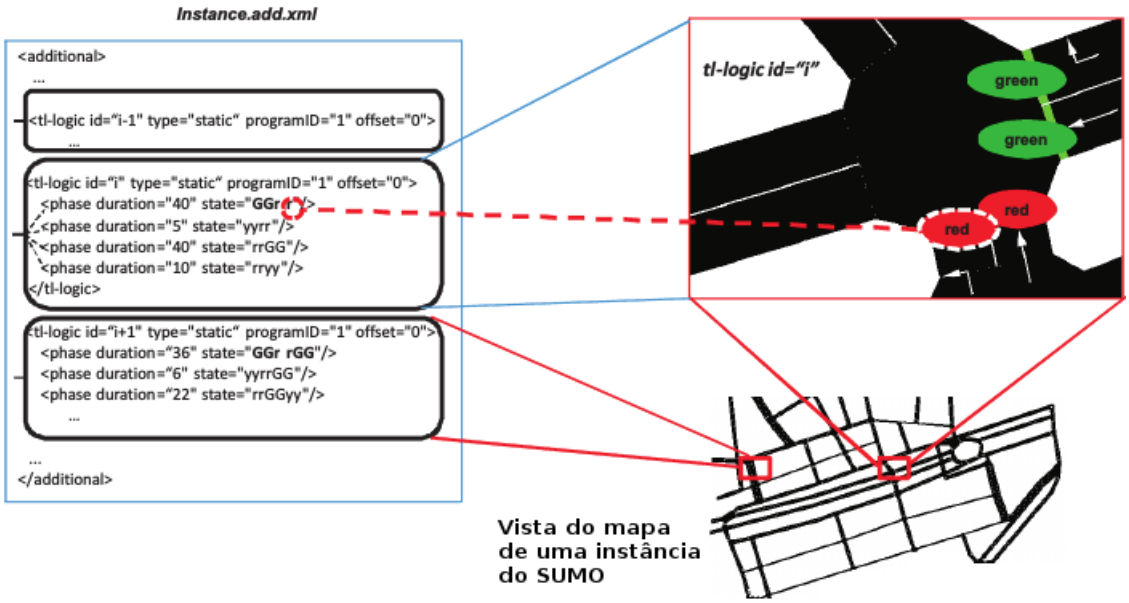


condução). Esta informação é usada para avaliar os programas de ciclo de semáforos (GARCÍA-NIETO; OLIVEIRA; ALBA, 2013). O SUMO também possibilita obter relatórios referentes aos veículos, as rotas, aos semáforos, informações da rede, nós e detectores especificados. Como saída referente aos veículos, podemos citar: o identificador, a rota, momento no qual entrou e saiu da simulação, quantidade de combustível utilizada, emissão de ruído e de poluentes (CO, CO_2, HC, NO_x, PM_X).

Através de arquivos adicionais (*.add.xml*) podem ser adicionados ao SUMO informações sobre o mapa ou sobre os semáforos. O SUMO permite substituir e editar informações sobre os ciclos de semáforos através da manipulação de um arquivo com extensão *.add.xml* (Figura 5). É importante notar que, por padrão, o simulador provê uma combinação válida de estados para o controlador de semáforo, localizada no arquivo *.net.xml*, e uma aproximação de tempos de intervalo para estes estados (GARCÍA-NIETO; OLIVEIRA; ALBA, 2013), (KRAJZEWICZ; BONERT; WAGNER, 2006).

Um exemplo SUMO para um cenário de tráfego urbano é composto basicamente por: cruzamentos, semáforos, pistas e veículos que se deslocam ao longo de suas rotas previamente especificadas. Os semáforos estão localizados nos cruzamentos (junções em SUMO) e controlam o fluxo de veículos, seguindo seus programas de fases e duração do ciclo. Neste contexto, todos os semáforos localizados no mesmo cruzamento são regidos por um programa comum, uma vez que eles têm de ser necessariamente sincronizados para a segurança do tráfego (GARCÍA-NIETO; OLIVEIRA; ALBA, 2013). Além disso, para todos os semáforos em um cruzamento, a combinação de fases durante um período de ciclo é sempre mantida válida (LEUNG, 2004) e deve seguir as regras de trânsito específicas de cruzamentos, a fim de evitar colisões de veículos e acidentes.

Figura 5 – Um cruzamento com quatro semáforos selecionados a partir do mapa de uma instância do SUMO. As durações de fase (ciclos) são especificadas no arquivo *Instance.add.xml*



Fonte: Adaptado de García-Nieto et al. (2013)

A cor do sinal de cada semáforo está disposta no parâmetro "*state*" e a duração dos sinais naquele estado é definido pelo parâmetro "*duration*". A duração é um valor inteiro que define o tempo de duração da fase. Já a fase é a sequência de cores que o semáforo assumirá durante a duração de tempo. A fase é configurada a partir de uma *string* contendo as letras "r", "y", "g" e "G" sendo o significado de cada uma explicado a seguir:

- r: define a cor vermelha (*red*) para o sinal;
- y: define a cor amarela (*yellow*) para o sinal;
- g: define a cor verde (*green*) para o sinal, sendo que a via com esse sinal não tem prioridade e deverão esperar os veículos (se houver) de um sinal com prioridade passar, para depois poder prosseguir;
- G: define a cor verde, mas com prioridade.

O número de letras que compõem a *string* do estado depende da quantidade de sinais que precisam ser dados. Considerando o cruzamento (em destaque) da Figura 5, quatro sinais precisam ser programados e por isso a *string* irá possuir quatro letras. Observe que os dois semáforos possuem dois sinais cada. Isso porque o semáforo mais à direita permite que o veículo siga em frente ou vire à sua direita, e o outro semáforo permite que o veículo siga em frente ou vire à sua esquerda. Esse estado (*state*) é representado pela *string* "GGrr" e terá duração (*duration*) de 40 segundos.

3

Otimização com Muitos Objetivos

Os mecanismos de otimização tratam da questão de determinar a melhor solução de problemas abstratos para os quais é possível quantificar o grau de adequação de cada solução à necessidade em causa. Dada a definição, em termos quantitativos, do que seriam os "objetivos de projeto", é possível comparar de maneira inequívoca duas possíveis soluções, sendo possível determinar de maneira não-ambígua qual delas é melhor. Com tal maneira de ordenar as soluções em uma escala de piores e melhores, torna-se possível fazer a própria máquina começar a gerar automaticamente soluções e "caminhar" automaticamente para soluções cada vez melhores (TAKAHASHI, 2007).

É possível tratar o problema da sincronização de semáforos como um problema de otimização, representando computacionalmente uma possível combinação de semáforos através de um vetor. O problema da sincronização de semáforos é definido como um problema de otimização através do uso de simuladores e suas medições de qualidade e algoritmos de otimização. Como há várias medidas possíveis de se otimizar, esse problema se torna um problema de otimização com muitos objetivos (ISHIBUCHI; TSUKAMOTO; NOJIMA, 2008). A sincronização de semáforos como um problema de otimização com muitos objetivos será aprofundada no capítulo 4.

3.1 Otimização Multiobjetivo

Os Problemas de Otimização Multiobjetivo, (MOP, do inglês *Multi-Objective Optimization Problems*) possuem mais de uma função objetivo. Na literatura, O MOP também é chamado de otimização multicritério ou otimização vetorial. Em geral, um MOP sem restrições é definido como:

$$f(x) = (f_1(x), \ldots, f_m(x)), x \in \Omega$$

Uma solução de MOP minimiza (ou maximiza) os componentes de um vetor $f(x) \in \Lambda$, onde x é um vetor composto por uma variável de decisão n-dimensional, $x = (x_1, \ldots, x_n)$, a partir de um universo Ω. O universo Ω contém todo x possível que pode ser utilizado para satisfazer uma avaliação de $f(x)$.

Nestes problemas, as funções objetivo que são otimizadas são conflitantes, logo não há somente uma melhor solução, mas um conjunto com as melhores soluções. Para obter esse conjunto de soluções é utilizada a Teoria da Otimalidade de Pareto (COELLO; LAMONT; VELDHUIZEN, 2006). Em problemas multiobjetivo, o ótimo é definido através dos termos a seguir (minimização):

- Dominância de Pareto: dadas duas soluções x e y, representadas pelos seus vetores de variáveis no espaço de objetivos, $u = (u_1, \ldots, u_m)$ e $v = (v_1, \ldots, v_m)$, respectivamente, dizemos que u domina v (denotado por $u \preceq v$) se, e somente se, u é parcialmente inferior a v, ou seja, $\forall i \in \{1, \ldots, m\}, u_i \leq v_i \wedge \exists i \in \{1, \ldots, m\} : u_i < v_i$. Ou seja, uma solução domina outra, se ela é pelo menos igual em todas as funções objetivo e é obrigatoriamente melhor em pelo menos uma função objetivo;

- Ótimo de Pareto: A solução x é Ótimo de Pareto se não existir vetor v que consiga diminuir algum objetivo sem causar um aumento simultâneo em pelo menos um outro objetivo (assumindo minimização). A solução u é dito Ótimo de Pareto se o seu vetor de funções objetivo é não dominado. Uma solução que é Ótimo de Pareto possui um vetor de funções objetivo que não pode ser simultaneamente melhorado (CARVALHO, 2013);

- Conjunto Ótimo de Pareto: para um MOP, o Conjunto Ótimo de Pareto, P^*, é o conjunto das melhores soluções em Ω, ou seja, é o conjunto de soluções do problema que são Ótimo de Pareto (CARVALHO, 2013);

- Fronteira de Pareto: cada solução em P^* possui uma imagem de um ponto não dominado em Λ. O conjunto de todos os pontos não dominados no espaço das funções objetivo é chamado de fronteira de Pareto (CARVALHO, 2013).

Os MOPs são solucionados por diferentes áreas de pesquisa, dentre as quais destacam-se os Algoritmos Evolucionários. Em geral, os algoritmos evolucionários utilizam uma população de soluções na busca e assim permitem a geração de diversos elementos da fronteira de Pareto em uma só execução. A área da Otimização Evolucionária Multiobjetivo (EMO, do inglês *Evolutionary Multi-Objective Optimization*) visa a aplicação de Algoritmos Evolucionários Multiobjetivo (MOEA, do inglês *Multi-Objective Evolutionary Optimization Algorithms*) para a solução de MOPs (CARVALHO, 2013).

Os MOEAs modificam os algoritmos evolucionários de duas maneiras: incorporam um mecanismo de seleção, geralmente baseado nos conceitos da Otimalidade de Pareto, e adotam

mecanismos para a preservação da diversidade, para evitar a convergência para uma só solução. Como a maioria dos MOPs são problemas complexos, os MOEAs focam em determinar um conjunto de soluções mais próximo possível do Conjunto Ótimo de Pareto, denominado de conjunto de aproximação. Um MOEA deve convergir para as melhores soluções do problema, porém deve cobrir de melhor forma as diferentes regiões para possibilitar a geração de melhor conjunto para o tomador de decisão (CARVALHO, 2013). Dentre os MOEAs existentes, podemos citar: o NSGA-II, MOPSO (do inglês *Multi-objective Particle Swarm Optimization*) e MOACO (do inglês *Multi-Objective Ant Colony Optimization*). Este trabalho irá explorar, dentre os diversos MOEAs, o NSGA-II que será apresentado na próxima seção.

3.1.1 *Nondominated Sorting Genetic Algorithm II*

Como já exposto nesta seção, os MOEAs são largamente reconhecidos como métodos apropriados e bem sucedidos para resolver problemas multiobjetivo. Dentre os diversos MOEAs, o NSGA-II proposto por (DEB et al., 2002) foi encontrado em diversos trabalhos (seção 3.3) que lidam com o problema da sincronização de semáforos como um problema multiobjetivo.

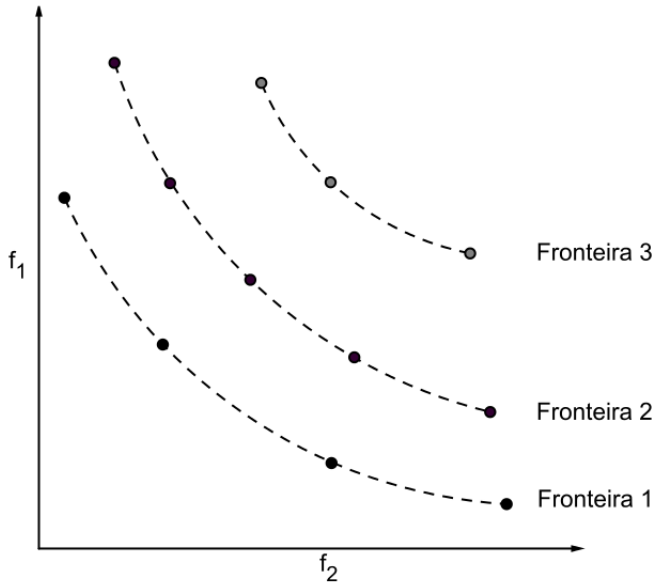
O NSGA-II divide os indivíduos da população em fronteiras e assim, as soluções não dominadas formam primeira fronteira, ou seja, as soluções dessa fronteira são as que possuem os melhores valores de aptidão. A segunda fronteira possui as soluções com o segundo melhor valor de aptidão, e assim por diante. Deste modo, seleciona-se soluções das primeiras fronteiras para fazer parte da próxima população. Caso a última fronteira a ser inserida na população ultrapasse o número máximo de soluções permitidas, o algoritmo irá selecionar as soluções mais distantes de seus vizinhos.

Este processo garante maior diversidade das soluções na fronteira de Pareto, pois qualifica e divide os indivíduos da população em fronteiras. Processo é dado utilizando a definição de dominância, e é denominado *fast non-dominated sorting* (FNDS). A Figura 6 ilustra um problema de minimização com duas funções objetivo e três fronteiras.

Assim, na Fronteira 1 estão contidas as soluções não dominadas, ou seja, todas as soluções que não são dominadas por nenhuma outra solução do conjunto. A Fronteira 2 é composta por soluções que são dominadas apenas pelas soluções da Fronteira 1. Por sua vez, a Fronteira 3 é formada por soluções que são dominadas pelas soluções da Fronteira 1 e pelas soluções da Fronteira 2, e assim sucessivamente.

Para garantir a diversidade das soluções ao longo de uma fronteira, o NSGA-II utiliza um operador denominado *Crowding Distance* (DEB et al., 2002). Este operador calcula a entre um ponto central i e dois pontos localizados nas extremidades deste ponto central, $(i-1)$ e $(i+1)$. Esta distância é normalizada pela diferença entre o maior e o menor valor que existe para cada objetivo. Assim, a *Crowding Distance* de uma solução i (d_i) representa uma estimativa do perímetro formado pelo cuboide cujos vértices são os seus vizinhos mais próximos. Os

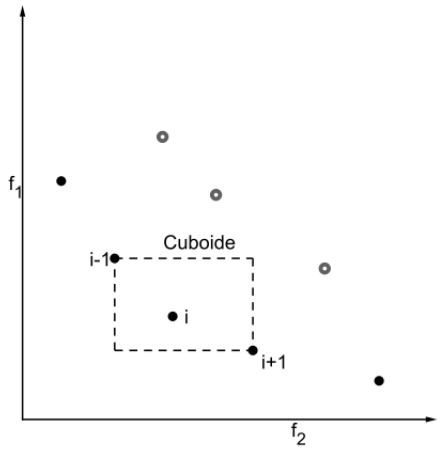
Figura 6 – Ordenação por dominância.



Fonte: Matsueda (2015)

pontos contidos nas extremidades da fronteira recebem um valor arbitrariamente grande, sendo priorizados durante a seleção. A Figura 7 ilustra o cálculo da *Crowding Distance*.

Figura 7 – *Crowding distance*



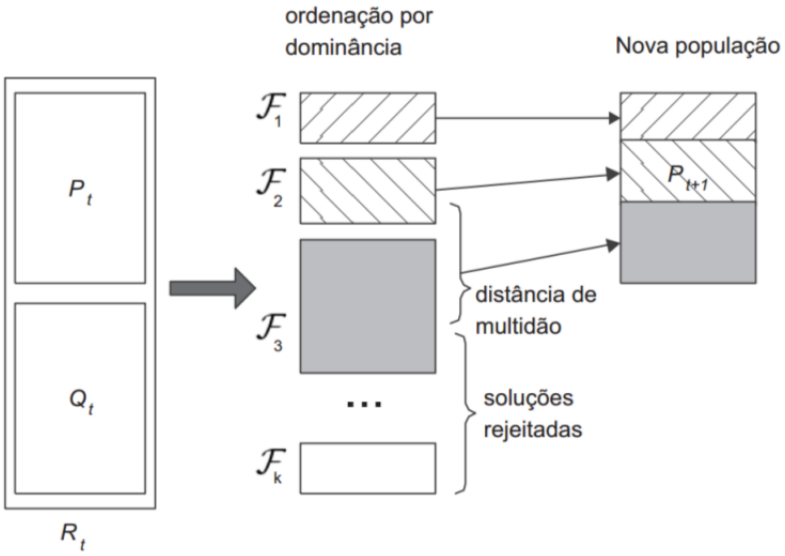
Fonte: Matsueda (2015)

Dados os procedimentos básicos do NSGA-II, o algoritmo primeiramente gera uma população inicial (pais) P_t de tamanho N. Posteriormente os operadores de cruzamento, mutação e seleção são aplicados nesta população, resultando em um uma nova população Q_t (filhos), também de tamanho N. Uma vez aplicados os operadores genéticos, o NSGA-II une as populações P_t e Q_t resultando em uma população R_t de tamanho $2N$. O seguinte passo faz uso do FNDS

sobre a população R_t, selecionando as melhores fronteiras, até que a população $P_t + 1$ volte a ter tamanho igual a N.

Caso a última fronteira a ser inserida na população $P_t + 1$ ultrapasse o número de indivíduos máximos da população (N), utiliza-se o algoritmo *Crowding Distance* para julgar quais serão os indivíduos, pertencentes a esta fronteira, que farão parte da nova população. Este ciclo se repete até que um critério de parada seja atingido. A Figura 8 ilustra o procedimento de inserção das soluções da população R na população P. A aplicação desse algoritmo ao problema da sincronização de semáforos será abordada na seção 4.1.3.

Figura 8 – Procedimento do NSGA-II



Fonte: Adaptado de Deb et al. (2002)

3.2 Problemas com Muitos Objetivos

Algoritmos MOEA geralmente funcionam muito bem em problemas com dois objetivos. No entanto, a sua performance de busca é severamente deteriorada pelo aumento do número de objetivos. Problemas multiobjetivo com quatro ou mais objetivos são chamados de problemas com muitos objetivos (ISHIBUCHI; TSUKAMOTO; NOJIMA, 2008). Quando o número de funções objetivo cresce, os algoritmos evolucionários multiobjetivo baseados em dominância de Pareto encontram algumas dificuldades em definir quais são as melhores soluções e não efetuam uma busca que converge para as soluções ótimas do problema.

Há uma necessidade crescente de desenvolver MOEAs para lidar com problemas de otimização com muitos objetivos (MaOP, do inglês *Many-Objective Optimization Problem*). Esses problemas representam uma série de desafios para qualquer algoritmo de otimização,

incluindo um MOEA. Em primeiro lugar, a proporção de soluções não dominadas em um conjunto de vetores objetivos, torna-se exponencialmente grande com o aumento do número de objetivos (DEB; JAIN, 2014). Os algoritmos MOEA que trabalham sob o princípio da dominação podem enfrentar as seguintes dificuldades (DEB; JAIN, 2014), (CARVALHO, 2013):

- Grande parte da população é não dominada: com um aumento no número de objetivos, uma parte cada vez maior de uma população gerada se torna não dominada. Assim, os MOEAs sofrem uma deterioração na busca quando o número de objetivos cresce;
- Avaliação da medida diversidade torna-se computacionalmente cara: para determinar a extensão da aglomeração das soluções numa população, a identificação de vizinhos torna-se computacionalmente cara em um espaço com muitas dimensões;
- Operação de recombinação pode ser ineficiente: em um problema com muitos objetivos, se apenas uma das soluções são encontradas em um espaço com muitas dimensões, as soluções tendem a ser muito distantes umas das outras;
- As métricas de desempenho são computacionalmente caras para calcular: como os conjuntos de pontos em muitas dimensões devem ser comparados uns com os outros para comparar o desempenho de um algoritmo em relação a outro, é necessário um maior esforço computacional;
- Visualização da fronteira de Pareto: a visualização dos pontos na fronteira de Pareto é importante para a tomada de decisão. Porém, com mais de três dimensões não é possível plotar os pontos da fronteira de Pareto com alguma forma convencional.

Para a atenuação dessas dificuldades, recentemente várias novas propostas têm sido apresentadas na literatura (ISHIBUCHI et al., 2011) e elas podem ser categorizadas em:

- Modificar a dominância de Pareto: obtem-se uma melhora na performance dos MOEAs para problema com muitos objetivos. (SATO; AGUIRRE; TANAKA, 2007) demonstraram uma clara melhora na performance do NSGA-II para problemas com muitos objetivos;
- Estratégias de decomposição: essa abordagem usa métodos de decomposição, os quais têm sido estudados na comunidade da programação matemática (Q.; LI, 2007) aplicada no contexto do EMO. Esse conjunto de métodos decompõe o problema multiobjetivo em vários problemas escalares, e então, um algoritmo evolucionário é aplicado para otimizar esses subproblemas (Q.; LI, 2007), (AGUIRRE; TANAKA, 2009);
- Diferentes esquemas de avaliação de *fitness*: essa abordagem não utiliza a dominância de Pareto como função de *fitness*, mas sim um conjunto diferente de medidas de avalição. Esses algoritmos utilizam medidas de desempenho do conjunto de soluções, como por

exemplo um conjunto de indicadores, tais como o hipervolume, epsilon e R2 (HUGHES, 2007), (ZITZLER; KÜNZLI, 2004);

- Utilizando informações de preferência: uma vez que o número de soluções necessárias para uma boa aproximação aumenta exponencialmente com o número de objetivos, é uma boa ideia se concentrar em uma região específica da fronteira de Pareto usando a preferência do tomador de decisão (PURSHOUSE; JALBa; FLEMING, 2011), (DEB; SUNDAR, 2006);

- Reduzindo o número de objetivos: utilizando apenas alguns objetivos importantes, pois assim quase todas as dificuldades encontradas na otimização evolucionária de muitos objetivos são eliminadas (DEB; SAXENA, 2006), (BROCKHOFF; ZITZLER, 2006a), (SINGH; ISAACS; RAY, 2011). A ideia geral é identificar os objetivos menos conflitantes (os quais possam ser removidos sem que haja uma alteração no conjunto Ótimo de Pareto) e descartá-los. Essa técnica será utilizada neste trabalho e será abordada na subseção 4.1.4.

Dentre as técnicas de Otimização com Muitos Objetivos, podemos destacar alguns trabalhos relacionados. Em (DEB; JAIN, 2014), propõe-se um algoritmo evolutivo de muitos objetivos, o NSGA-III, que enfatiza os membros da população que não são dominados e que estão próximos a um conjunto de pontos de referência fornecidos. O algoritmo proposto tem como base o NSGA-II, porém utiliza um conjunto estruturado de pontos de referência para garantir a dispersão das soluções na população. A proposta NSGA-III é aplicada a problemas de três a quinze objetivos e comparado com duas versões de um algoritmo MOEA recentemente sugerido, o MOEA/D. Embora os dois MOEA/D funcionem bem em diferentes classes de problemas, o NSGA-III proposto produz resultados satisfatórios em todos os problemas considerados neste trabalho. O NSGA-III foi capaz, por diversas vezes (em várias execuções), de encontrar com êxito um conjunto de pontos com boa diversidade e convergência. O desempenho para 15 objetivos é alcançado principalmente devido ao auxílio na preservação da diversidade através do fornecimento de um conjunto de pontos de referência bem distribuídos. Essa técnica será melhor abordada na próxima subseção 3.2.1

Em (BRITTO; POZO, 2012), o principal objetivo foi explorar vários métodos de arquivamento da literatura utilizados por MOPSO para armazenar líderes selecionados em MaOPs. Neste trabalho, buscou-se observar como cada método de arquivamento influencia o algoritmo MOPSO em termos de convergência e diversidade ao longo da fronteira de Pareto. Além disso, alguns novos métodos de arquivamento, com características específicas de convergência ou diversidade, foram propostos: *Ideal Archiver*, *Distributed Archiver* e *Distance to Reference Points Archiver*. Os melhores resultados foram obtidos através dos métodos de arquivamento que não limitam o tamanho do arquivo. A hipótese do trabalho era que métodos de arquivamento são capazes de reduzir a deterioração do MOPSO em cenários com muitos objetivos. Os resultados indicaram que é possível obter bons resultados utilizando métodos de arquivamento. As

abordagens *Unbound Archiver* e ε-Pareto obtiveram boa convergência e diversidade ao longo da fronteira de Pareto, já a *Ideal Archiver* apresentou a melhor convergência para a fronteira de Pareto.

Uma outra estratégia que explora MOPSO foi apresentada em (BRITTO; MOSTAGHIM; POZO, 2013). Neste trabalho, os autores apresentaram um novo algoritmo PSO, chamado *Iterated Multi-Swarm* (*I-Multi*), que explora as características específicas do PSO para enfrentar os problemas com muitos objetivos. *I-Multi* tem a característica de combinar diferentes métodos de arquivamento que melhoraram a busca tanto em termos de convergência para a fronteira de Pareto, quanto na diversidade das soluções não-dominadas. O algoritmo inicialmente realiza uma pesquisa diversificada usando o método de arquivamento MGA, encontrando uma grande área no espaço de soluções. Em seguida, utiliza sub-enxames em torno das soluções encontradas pela MGA. Esta abordagem permite uma busca que abrange diferentes áreas do espaço de soluções. É avaliado através de uma análise empírica usando um conjunto de problemas de otimização com muitos objetivos da família DTLZ. Por outro lado, *I-Multi* é comparado com os algoritmos MGA-SMPSO e Ideal-SMPSO. Em geral, os métodos de *I-Multi* obtiveram resultados promissores. As três variantes de *I-Multi* que foram propostas também obtiveram bons resultados, cada uma destacando-se em diferentes cenários. *I-Multi Random* foi mais eficaz no problema DTLZ2 e DTLZ4. *I-Multi Centroid* obteve um resultado semelhante ao de *I-Multi Random* no problema DTLZ4, e se destacou no problema DTLZ7. *I-Multi Extremes*, em geral, obteve resultados muito bons. No entanto, esta abordagem sofre uma grande deterioração quando o número de objetivos é similar ao número de enxames, uma vez que apenas os extremos são endereçados nesta situação.

3.2.1 *Nondominated Sorting Genetic Algorithm III*

Como dito anteriomente nesta seção, o algoritmo NSGA-III proposto por (DEB; JAIN, 2014), é uma versão do NSGA-II que utiliza pontos de referência para guiar a busca quando todos os indivíduos são não dominados. O procedimento realizado pelo NSGA-III é semelhante ao do NSGA-II. Tal como o seu antecessor, utiliza dominância de Pareto para classificar as soluções de uma população. Porém, quando existem soluções que pertencem ao mesmo nível de dominância o algoritmo utiliza pontos de referência no espaço de objetivos, de tal forma que as soluções mais próximas destes pontos são melhores classificadas.

A estrutura básica proposta pelo NSGA-III permanece semelhante ao algoritmo NSGA-II original com mudanças significativas em seu mecanismo de seleção. Ao contrário do NSGA-II que utiliza o cálculo do perímetro formado pelo cuboide dos vizinhos mais próximos (*Crowding Distance*), a manutenção da diversidade entre os membros da população no NSGA-III é auxiliado pelo fornecimento de uma série de pontos de referência bem espalhados. Estes pontos podem ser predefinidos de forma estruturada ou fornecido preferencialmente pelo usuário. Na ausência de qualquer informação de preferência, qualquer colocação estruturada de pontos de referência pode ser adotada (DEB; JAIN, 2014).

Devido ao fato do NSGA-III utilizar um conjunto pré-definido de pontos de referência para assegurar a diversidade em soluções obtidas, o algoritmo além de priorizar soluções não dominadas, também enfatiza os membros da população que são de algum modo associados com cada ponto de referência. Uma vez que os pontos de referência são criados, estes são amplamente distribuídos em todo o hiperplano normalizado. Sendo assim, o NSGA-III enfatiza os membros da população que são de algum modo associados com cada um destes pontos de referência, o algoritmo tende a encontrar soluções aproximadas de Pareto-Ótimo correspondentes aos pontos de referência fornecidos.

Algoritmo 1: NSGA-III - Pseudocódigo

Entrada : H pontos de referência estruturados Z^s ou Z^a pontos candidatos fornecidos pela população dos pais P_t
Saída : P_{t+1}

$S_t = \emptyset, i = 1$;
$Q_t =$ Recombinação+Mutação(P_t);
$R_t = P_t \cup Q_t$;
$(F_1, F_2, \ldots) =$ Ordenação não dominados(R_t);
repita
  $S_t = S_t \cup F_i$ and $i = i + 1$;
até $\mid S_t \mid \geq N$;
Última fronteira a ser incluída: $F_l = F_i$;
se $\mid S_t \mid = N$ **então**
  $P_{t+1} = S_t$, break;
senão
  $P_{t+1} = U_{j=1}^{l-1} F_j$;
  Pontos a serem escolhidos de F_l: $K = N - \mid P_{t+1} \mid$;
  $Normalizao(f^n, S_t, Z^r, Z^s, Z^a)$;
  $Associao(S_t, Z^r)$;
  $p_j = \sum_{s \in S_t / F_l} ((\pi(s) = j) \ ? \ 1 : 0)$;
  Selecionar K membros, um de cada vez, de F_l para construir $P_{t+1} : Niching(K, p_j, \pi, d, Z^r, F_l, P_{t+1})$;

Assim, o NSGA-III, do ponto de vista de uma aplicação, combina tomada de decisões e otimização de muitos objetivos. A estratégia proposta por (DAS; DENNIS, 1998) apresenta um comportamento eficiente a respeito das soluções. Isto porque, as soluções da aproximação de Pareto-Ótimo tendem a ser bem diversificadas nesta abordagem. O NSGA-III não faz uso de nenhum método de seleção para reprodução explícita para gerar a população filha. Neste algoritmo, os indivíduos são escolhidos de forma aleatória. Esta população é, portanto, construída através da aplicação de operadores de cruzamento e de mutação habituais, escolhendo aleatoriamente os indivíduos da população pai.

O procedimento para identificar as frentes não-dominadas (FNDS), apresentado na Subseção 3.1.1, também é usado no NSGA-III. Sendo que todos os indivíduos da população não-dominada da fronteira 1 até a fronteira l são incluídos inicialmente na população S_t. Se S_t

contiver exatamente o número máximo N de indivíduos, nenhuma outra operação é necessária e a próxima geração é iniciada com $P_{t+1} = S_t$. Caso $\mid S_t \mid > N$, os membros da fronteira $(l-1)$ são selecionados, ou seja, $P_{t+1} = U_{j=1}^{l-1} F_j$. E os $K = N - \mid P_{t+1} \mid$ membros restantes da população são escolhidos da última fronteira F_l. O processo de seleção então é formado por outros três procedimentos, a normalização , a associação e a preservação do nicho.

No entanto, após a formação da população P_{t+1}, ela é então usada para criar uma nova população de descendentes Q_{t+1} aplicando os operadores genéticos usuais. O NSGA-III já realiza uma cuidadosa seleção elitista de soluções e tenta manter a diversidade entre soluções enfatizando as soluções mais próximas da linha de referência de cada ponto de referência. Além disso, o número de indivíduos N de uma população do NSGA-III se aproxima do número de pontos de referência H que foram gerados para este mesmo algoritmo, ou seja, com $N \approx H$. Isto é, o tamanho da população do NSGA-III é definido como o menor múltiplo de quatro maior que o número de pontos de referência H gerados. Deste modo, a cada membro da população do NSGA-III é dada uma igual importância.

Por estas razões, como dito anteriormente, o NSGA-III não emprega qualquer tipo de operação de seleção explícita para gerar a população filha Q_{t+1}. Esta população é portanto construída através da aplicação de operadores de cruzamento e de mutação habituais, escolhendo aleatoriamente os pais de P_{t+1}. O pseudocódigo do NSGA-III pode ser visto em Algoritmo 1. A aplicação desse algoritmo ao problema da sincronização de semáforos será abordada na subseção 4.1.3.

3.2.2 Redução de Dimensionalidade

Redução de dimensionalidade é uma técnica utilizada em muitas áreas, como estatística, otimização de objetivos e mineração de dados. E pode ser distinguida entre duas principais abordagens: (i) extração de características e (ii) seleção de características. A tarefa de extração de características é determinar um conjunto (pequeno) de elementos arbitrários, enquanto a tarefa de seleção de características é encontrar o menor subconjunto dos recursos dados, representando os dados fornecidos de uma melhor forma.

As técnicas de redução da dimensionalidade podem ser aplicadas na otimização com muitos objetivos, com o intuito de reduzir o número de funções objetivo a serem otimizadas, onde a ideia geral é identificar os objetivos menos conflitantes (os quais possam ser removidos sem que haja uma alteração no conjunto Ótimo de Pareto) e descartá-los.

Os métodos para redução de objetivos podem ser benéficos tanto para a tomada de decisão, quanto para a busca. Segundo Jaimes, Coello e Barrientos (2009) o sucesso de um método de redução de objetivos durante a busca, depende principalmente do saldo entre os custos indiretos incorridos pelo próprio método de redução e o tempo economizado ao omitir algumas funções objetivo. Realizar redução de objetivos antes de interagir com o tomador de decisão tem

duas vantagens. Em primeiro lugar, a revelação de que certos objetivos são redundantes, reduz significativamente a complexidade do problema de otimização, implicando um menor custo computacional e uma maior eficiência da busca. Em segundo lugar, é bem sabido que os seres humanos não são eficientes no tratamento de vários fatores (objetivos no contexto atual) de cada vez. Assim, a simplificação do problema, a priori, aborda a questão fundamental da sobrecarga cognitiva para o tomador de decisão, o que pode ajudar a evitar preferências inconsistentes durante as diferentes fases do processo interativo.

A redução da dimensionalidade aplicada em algoritmos evolucionários multiobjetivo pode ser formalizada através de alguns conceitos (SAXENA et al., 2013). A ideia geral é que a partir da execução de um MOEA, encontre-se o conjunto de objetivos essenciais onde há o menor conjunto de objetivos conflitantes, chamado de F_T. A quantidade de objetivos do problema inicial é M, a quantidade de objetivos reduzida é m e o conjunto de objetivos redundantes é F_{Redn}. No trabalho, destaca-se os benefícios potenciais da redução de objetivo da seguinte forma:

- Se o problema poderia ser reduzido para $m \leq 3$: um problema que anteriormente não tinha solução, agora poderia ser resolvido utilizando qualquer MOEAs existente;
- Mesmo se $4 \leq m < M$: haveriam benefícios devido a redução de objetivo, aumentando a eficiência da busca, diminuindo o custo computacional, facilitando a visualização e tomada de decisão;
- A redução de objetivo poderia desempenhar um papel complementar para as abordagens preferência-ordenação. Se este último fosse aplicado à representação reduzida do problema original, ganhos maiores podem ser esperados.

O mesmo trabalho ainda afirma que a redução de objetivo refere-se a encontrar um conjunto objetivo essencial para um determinado problema. Nisso, foi definido que:

- Um conjunto objetivo essencial é definido como o menor conjunto de objetivos conflitantes (F_T, $|F_T| = m$) que pode gerar a mesma POF (do inglês *Pareto-optimal front*) que a gerada no problema original, dado por $F_0 = \{f_1, f_2, \ldots, f_M\}$;
- A dimensionalidade do problema refere-se ao número de objetivos essenciais (m, $m \leq M$);
- Um conjunto objetivo redundante ($F_{Redn} = F_0 \setminus F_T$) refere-se ao conjunto de objetivos, onde a eliminação não afeta a POF do problema dado. Percebe-se que um objetivo poderia ser redundante se for não-conflitante (ou correlacionado) com outros objetivos.

Destaca ainda que algumas questões importantes em torno da redução de objetivo, devem ser expostas antes de introduzir as abordagens existentes. Em primeiro lugar, todas estas abordagens operam nos vetores objetivos das soluções obtidas a partir de um MOEA com

soluções não dominadas. Outra questão importante é que a redução de objetivo pode ser realizada durante a execução do MOEA (redução *online*) para simplificar a busca, ou pode ser realizada após a execução do MOEA (redução *offline*) para auxiliar na tomada de decisão.

Em terceiro lugar, um conjunto objetivo essencial pode ser expresso como um subconjunto do problema original, ou os seus elementos podem ser expressos como uma combinação linear dos objetivos originais. A primeira abordagem é referida como seleção de características. E a última é referida como uma extração de funcionalidade. A maioria das abordagens de redução de objetivos existentes utilizam seleção de características para facilitar na tomada de decisão posteriormente. No caminho do exposto acima, segundo Saxena et al. (2013), as abordagens de redução de objetivos existentes são as seguintes:

- ***Dominance relation preservation (DRP)***: a abordagem de redução de objetivo proposta no trabalho de (BROCKHOFF; ZITZLER, 2006a) é baseada em preservar as relações de dominância das soluções não dominadas. Para um dado conjunto de objetivo F, se as relações de dominância entre os vetores objetivos permanecem inalteradas quando um objetivo $f \in F$ é removido, então f é considerado não-conflitante com os outros objetivos em F. Com base neste critério, um algoritmo exato e um guloso são propostos para resolver os problemas δ-*MOSS* (encontrar o subconjunto mínimo de objetivos correspondente a um determinado erro δ) e k-*EMOSS* (encontrar um subconjunto de objetivo de tamanho k com um mínimo erro possível);

- ***Unsupervised feature selection***: (JAIMES; COELLO; CHAKRABORTY, 2008) esta abordagem utiliza a correlação entre os objetivos e trata os objetivos mais distantes como os mais conflitantes. Neste: **a)** o conjunto de objetivos é dividido em vizinhanças de tamanho q em torno de cada um dos objetivos; e **b)** a "vizinhança" mais compacta é selecionada, cujo centro é mantido e os vizinhos são eliminados. Este processo é repetido até que um critério pré-determinado é atingido;

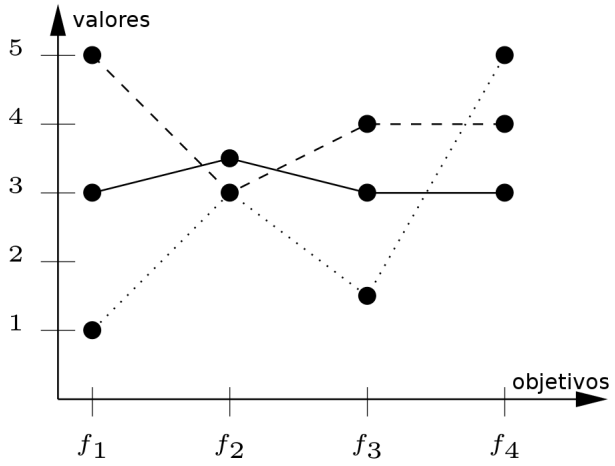
- ***Pareto corner search***: o algoritmo *Pareto corner search evolutionary* foi proposto no trabalho (SINGH; ISAACS; RAY, 2011), neste trabalho, ao invés de apontar para a completa POF, procura apenas os cantos ("*corner*") da POF. As soluções obtidas são assumidas para capturar adequadamente a dependência da POF em objetivos diferentes. Em seguida, a redução de objetivos baseia-se na premissa de que omitindo um objetivo redundante e essencial, trará um efeito, negligenciável e substancial, respectivamente, sobre o número de soluções não dominadas na população;

- ***Machine learning based objective reduction***: esta abordagem (DEB; SAXENA, 2006), (DEB; SAXENA, 2007), utiliza técnicas de aprendizado de máquina, como Análise de Componentes Principais (PCA, do inglês *Principal Component Analysis*) e Desdobramento de Variância Máxima (MVU, do inglês *Maximum Variance Unfolding*) para remover

as dependências de segunda-ordem e maior-ordem, respectivamente, nas soluções não dominadas.

Para ilustrar o conceito de redução de objetivos dos trabalhos (BROCKHOFF; ZITZLER, 2006a), (BROCKHOFF; ZITZLER, 2009), alguns exemplos são ilustrados a seguir. O gráfico da Figura 9 ilustra um problema de minimização com quatro objetivos (f_1, f_2, f_3 e f_4) e três soluções não dominadas: x_1 (linha cheia), x_2 (linha tracejada) e x_3 (linha pontilhada). Em uma análise mais profunda é possível notar que os objetivos f_1 e f_3 indicam redundância na formulação do problema, pois as relações são as mesmas, ou seja, para os dois objetivos, a solução x_2 tem valor maior que x_1 que por sua vez tem valor maior que x_3. Sendo assim é possível eliminar o objetivo f_1 ou f_3, mantendo a mesma relação entre as três soluções.

Figura 9 – Coordenadas paralelas para três soluções e quatro objetivos

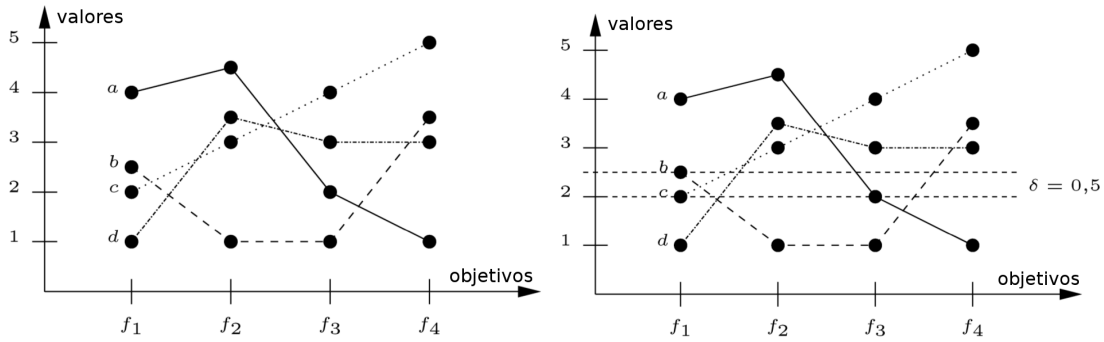


Fonte: Adaptado de Brockhoff; Zitzler (2006)

Considerando o problema de minimização representado na Figura 10, existem quatro objetivos (f_1, f_2, f_3 e f_4) e quatro soluções não dominadas: a (linha cheia), b (linha tracejada), c (linha pontilhada), e d (linha tracejada-pontilhada). No gráfico da esquerda (Figura 10(a)), se objetivo f_3 for omitido, a comparação entre todas as soluções permanece inconclusiva no que diz respeito ao sub-conjunto de objetivos $\{f_1, f_2, f_4\}$. Assim, classifica-se os conjuntos de objetivos $\{f_1, f_2, f_4\}$ e $\{f_1, f_2, f_3, f_4\}$ como não conflitantes.

No gráfico da direita (Figura 10(b)) a definição de δ-erro é ilustrada. Ao contrário de quando f_3 é omitido, se qualquer um dos outros três objetivos for omitido, haverá uma mudança na relação de dominância entre as soluções, ou seja, haverá ao menos uma solução que dominará outra. Ao examinar o que acontece se, por exemplo f_1 for omitido, observa-se que como resultado, a solução c é fracamente dominada pela solução b, ao contrário de quando as duas soluções são comparadas no cenário com os quatro objetivos. Todavia, se na solução c, o objetivo f_1 for acrescido em $\delta = 0.5$, então a relação entre a solução c e b não será alterada

Figura 10 – Coordenadas paralelas para um dado exemplo com quatro soluções e quatro objetivos (à esquerda). À direita, a definição do erro é ilustrada



Fonte: Adaptado de Brockhoff; Zitzler (2009)

quando f_1 for omitido, pois o objetivo f_1 terá o mesmo valor nas duas soluções, c e b. Assim, assumindo um δ-erro = 0.5, pode-se reduzir a um sub-conjunto conjunto de objetivos dado por $\{f_2, f_4\}$. O valor $\delta = 0.5$ pode ser usado como uma medida para quantificar a diferença na estrutura de dominância induzida por $\{f_2, f_4\}$ e $\{f_1, f_2, f_3, f_4\}$.

3.2.2.1 Trabalhos Relacionados

Dentre os trabalhos encontrados na literatura durante a pesquisa, no que tange redução de dimensionalidade, podemos citar os seguintes.

Deb e Saxena (2006) propuseram um método para reduzir o numero de objetivos por meio de uma abordagem baseada em PCA juntamente com algoritmos evolutivos multiobjetivo. O procedimento proposto de forma iterativa identifica os objetivos redundantes de soluções obtidas por um NSGA-II e os elimina. A complexidade do tempo de cada iteração deste algoritmo é $O(tM^2 + M^3) + O(gt^2M)$, onde o segundo termo corresponde à complexidade do NSGA-II, M é o número de objetivos, t é o tamanho do conjunto não dominado, e g o número de gerações para cada execução do NSGA-II. Outros trabalhos também apresentam técnicas baseadas em PCA para redução de objetivos ((DEB; SAXENA, 2007), (SAXENA et al., 2013))

Brockhoff e Zitzler (2006a) propuseram dois algoritmos gulosos para reduzir o número de objetivos. Um deles encontra um subconjunto objetivo mínimo que produz um dado erro δ (grau de mudança da relação de dominância). O outro algoritmo encontra um subconjunto objetivo de tamanho k com o erro mínimo possível. Ambos os algoritmos usam a relação ε-dominância para medir a mudança da relação de dominância quando os objetivos são descartados. A remoção é considerada viável pois o objetivo é considerado um objetivo sem conflito. A complexidade destes algoritmos são $O(min\{t^2M^3, t^4M^2\})$ e $O(t^2M^3)$, respectivamente. De forma similar em (BROCKHOFF; ZITZLER, 2006b), propuseram um método para redução de objetivos que procura pelo subconjunto mínimo de objetivos com erro mínimo.

Semelhante a Brockhoff e Zitzler, (JAIMES; COELLO; CHAKRABORTY, 2008), propuseram dois algoritmos para reduzir o número de objetivos. O primeiro algoritmo destina-se a determinar um subconjunto mínimo de objetivos que produz o erro mínimo possível, enquanto o segundo encontra um subconjunto de objetivos de um dado tamanho que produz o erro mínimo. Estes algoritmos baseiam-se numa técnica de seleção de características que utiliza a correlação entre vetores não dominados para estimar o conflito entre cada par de objetivos. A complexidade de ambos os algoritmos é $O(tM^2)$. Uma vez que esses algoritmos possuem baixa complexidade, eles são adequados para serem integrados em um MOEA, visto que são maiores as chances deles terem economia de tempo computacional maior que os custos dos outros métodos descritos aqui. Jaimes e Coello (2009) desenvolveram uma técnica de redução de objetivos que pode ser utilizada durante a busca ou no processo de tomada de decisão.

O trabalho (SINGH; ISAACS; RAY, 2011) propõe uma busca de soluções extremas da frente Pareto aproximada e que desprezam as demais soluções. Uma vez que as soluções extremas são identificadas, uma técnica heurística verifica o número de soluções não dominadas resultantes de uma redução nas soluções alcançadas, indicando se os objetivos são importantes ou se eles podem ser reduzidos.

Matsueda (2015) propôs um problema de roteamento de veículos com muitos objetivos e janelas de tempo flexíveis (MOPRV), através de uma abordagem baseada nos algoritmos NSGA-II e NSGA-III, e de um método para a redução e visualização de objetivos, chamado de Árvores de Agregação. Através de um estudo sobre a harmonia e conflito entre os objetivos do problema, foi observada a possibilidade de agregação entre os mesmos. Os resultados demonstram que é mais vantajoso visualizar a relação entre os objetivos do MOPRV e em seguida otimizar o problema com menos objetivos do que tentar otimizar diretamente o problema considerando todos os objetivos do MOPRV.

Brockhoff e Zitzler (2009) mostraram experimentalmente que a redução de objetivos *online* pode melhorar drasticamente o tempo de execução de um algoritmo de busca baseado em hipervolume. Neste trabalho foi proposta uma abordagem de redução de objetivos que é baseada em uma noção geral de conflito objetivo. Essa abordagem permite identificar conjuntos de objetivos de tamanho mínimo, garantindo que a relação de dominação Pareto seja preservada ou apenas ligeiramente alterada de acordo com um certo erro pré-definido. Para este fim, um algoritmo exato, bem como várias heurísticas foram propostas e implementações correspondentes foram disponibilizadas gratuitamente para download [1]. As implementações permitem de forma prática e simples obter várias informações a respeito dos objetivos. Além disso, são disponibilizadas todas as classes de implementação, bem como um arquivo de ajuda. Os principais algoritmos disponibilizados são:

[1] <http://www.tik.ee.ethz.ch/sop/download/supplementary/objectiveReduction/>

- *AllMinimalSets*: Calcula todos os conjuntos mínimos para um determinado conjunto de indivíduos com o algoritmo exato;
- *ConflictCalculator*: Calcula o tamanho de um conjunto mínimo não-redundante para um dado conjunto de indivíduos;
- *DeltaCalculator*: Calcula o erro *delta* para todas as soluções de acordo com o conjunto fornecido;
- *FrontCalculator*: Calcula os pontos não dominados ou dominados em um conjunto de indivíduos;
- *GreedyDeltaMOSS*: Executa o algoritmo guloso *delta-MOSS* para um determinado delta e os indivíduos fornecidos;
- *GreedyKEMOSS*: Executa o algoritmo guloso *k-EMOSS* para um determinado k e os indivíduos fornecidos.

3.3 Algoritmos Multiobjetivo Aplicados à Sincronização de Semáforos

Existe atualmente uma série de trabalhos na literatura que utilizam algoritmos de otimização e simulação para resolver o problema da sincronização de semáforos. Em (SHEN; WANG; Y., 2013), utilizou-se o algoritmo NSGA-II para resolver o problema de otimização de semáforos com duas funções objetivo: tempo médio de atraso e tempo médio de parada. Além disso, um simulador de tráfego baseado em CA (do inglês *Cellular Automata*) e uma programação paralela, através de GPU (do inglês *Graphics Processing Unit*), foram utilizados no sistema. Para os experimentos, o NSGA-II foi testado em dois casos. No primeiro caso, a malha viária possui quatro cruzamentos e no segundo, possui dezoito cruzamentos. As malhas representam uma área de *Zhongguancun* na cidade de *Beijing* na China. Em cada experimento obteve-se um fator de aceleração de 21.46 e 27.64, respectivamente.

No trabalho (KWASNICKA; STANEK, 2006), é proposto um método de otimização de semáfotos com base em um algoritmo genético suportado pelo programa de simulação microscópica. Seu objetivo é minimizar o tempo perdido durante a viagem e maximizar a quantidade de veículos que circulam pela rede (área otimizada). O trabalho utiliza duas configurações para realizar os experimentos, uma com e outra sem detectores dentro da malha viária. Ao utilizar detectores, encontram-se resultados mais precisos sem um aumento significativo do tempo de computação. Em ambas as configurações testadas, boas condições de tráfego foram alcançadas, com carros em movimento constante. As filas em cada entrada (injetores) foram esvaziadas rapidamente e não existiram situações onde os veículos não foram capazes de entrar na área de estudo.

Em (MONIREH; NASSER; ANA, 2011), foram utilizados sistemas multiagentes e aprendizado de máquina para desenvolver um mecanismo de controle de semáforo. Para a simulação, foi utilizado o simulador AIMSUN (do inglês *Advanced Interactive Microscopic Simulator for Urban and Non-Urban Networks*). O trabalho busca reduzir o tempo médio de atraso, numa abordagem de aprendizagem por reforço que é *model-free*, chamada de "*Q-learning*". Cada agente é responsável por controlar os semáforos em um cruzamento, utilizando apenas informações locais. Para estimar o estado, os agentes utilizam o comprimento médio das filas que se aproximam dos cruzamentos em um ciclo fixo. Em seguida, os agentes selecionam uma ação e recebem uma recompensa. Os resultados das simulações mostram que *Q-learning* superou o tempo fixo em diferentes demandas de tráfego.

No trabalho (SÁNCHEZ-MEDINA; GALÁN-MORENO; RUBIO-ROYO, 2010) foi desenvolvido um modelo de otimização semafórica com base na combinação de três técnicas principais: algoritmo genético como uma técnica de otimização, um simulador de tráfego baseado em CA dentro da avaliação de aptidão do algoritmo genético e um *cluster Beowulf*. No trabalho foram testadas quatro funções de aptidão: o total de veículos que chegaram ao seu destino, o tempo médio de viagem, a velocidade média global e o TOC/SOC (do inglês *time of occupancy* e *state of occupancy*). No entanto, o problema não foi modelado como multiobjetivo, e cada medida foi otimizada separadamente. Independente de qual fosse o objetivo a ser otimizado, os autores plotaram em um gráfico os valores obtidos para os outros três objetivos e com isso foi possível notar uma relação entre pares de objetivos, algumas dessas relações mostraram uma forte dependência entre o par de objetivos. Além disso o trabalho obteve êxito em otimizar todas as quatro funções.

O trabalho (HAJBABAIE; BENEKOHAL, 2015) propõe encontrar uma solução otimizada para os tempos dos semáforos ao mesmo tempo que define qual o fluxo de veículos ideal para o tráfego na região. Através de um algoritmo genético e do simulador de tráfego CORSIM, os autores desenvolveram uma meta-heurística nomeada IDSTOP (do inglês *Intelligent Dynamic Signal Timing Optimization Program*). Para alcançar os objetivos é necessário minimizar o tempo de viagem, tentando alcançar o limite superior. Para calcular este limite superior, é feito um processo de relaxamento das restrições do fluxo. Ou seja, dar condições para que um cruzamento não influencie outros. Feito isso, é garantido que toda a demanda de tráfego chegue ao seu destino ao final da simulação. E então, quanto mais próximo os veículos estejam do tempo ideal, melhores são os tempos dos semáforos. A meta-heurística é composta por cinco módulos: *Initialization*, *Signal timing*, *Traffic assignment*, *Constraint satisfaction* e *Solution quality evaluation*.

No processo inicial, o sistema de atribuição de tráfego ideal é iniciado assumindo uma divisão de tempo de $50\%/50\%$ para as diferentes direções. Usando estes parâmetros de tempo semafórico, o tráfego é atribuído e volumes de link são encontrados. Em seguida, uma geração para os tempos dos semáforos é criada, esse é um conjunto de soluções candidatas, e a qualidade

cada uma delas é avaliada através da execução do simulador CORSIM. A melhor solução (em termos de número de viagens completas) é selecionada para a atribuição de tráfego e sua qualidade é mais uma vez avaliada através do CORSIM. Se sua qualidade não melhorar, utilizasse as atribuições iniciais. Se melhorar, ou seja, se um maior número de viagens concluídas é obtido, os volumes de link são atualizadas para a próxima geração.

A estratégia utilizada no trabalho é atacar o problema de forma discreta, em curtos intervalos de tempo, dividindo todo o problema em problemas menores, onde as condições iniciais de um novo intervalo de tempo são as condições finais do intervalo anterior. Ao se discretizar o período de tempo, reduz-se significativamente o tamanho do espaço de busca e, consequentemente, a complexidade do problema. É assegurado ainda que um maior número de viagens são completadas em cada intervalo de tempo, e os véiculos em questão processados. Caso as viagens não terminem dentro do intervalo, os veículos são mantidos para o próximo intervalo.

O estudo de caso foi adotado a partir do centro de *Springfield*, em *Illinois*. A rede de estudo de caso inclui vinte cruzamentos e uma combinação de ruas de mão única e de mão dupla com diferentes número de pistas. O número máximo de fases varia na rede. Quatro padrões de demanda são usados para cobrir padrões simétricos e assimétricos em condições subsaturadas e sobressaturadas. Os resultados (viagens concluídas, o atraso total, a média de atraso, o tempo médio de viagem e a velocidade média) alcançados por do IDSTOP foram melhores quando comparado a abordagem que utiliza apenas o CORSIM. Em contrapartida, os tempos das execuções (para 20 gerações) foram muito maiores e poderiam ser ainda maiores se os autores não tivessem utilizado o módulo *Constraint satisfaction*, que descarta soluções não desejáveis.

A combinação entre simulação e algoritmos de otimização para resolver o problema da sincronização de semáforos não é nova, existem diversos trabalhos na literatura a respeito. Apesar disso, nenhum dos cinco trabalhos relacionados acima otimizam mais do que dois objetivos por vez. Apesar de listarem que existem várias medidas possíveis de otimizar, não as otimizam de uma só vez. Por consequência não exploram técnicas de muitos objetivos. No próximo capítulo será apresentado como modelar o problema da sincronização de semáforos como um problema de otimização com muitos objetivos.

4

Sincronização de Semáforos como um Problema de Otimização com Muitos Objetivos

O problema sincronização de semáforos pode ser modelado como um problema de otimização. Com o auxílio de um simulador de tráfego, dada uma combinação de semáforos, é possível construir uma representação computacional e obter medidas de qualidade oriundas do próprio simulador. Como o simulador disponibiliza diversas medidas de qualidade, tem-se um problema de otimização com muitos objetivos.

Como apresentado até aqui os objetivos deste trabalho são modelar e resolver o problema da sincronização de semáforos como um problema de otimização com muitos objetivos. Diferentemente dos trabalhos encontrados na literatura (seção 3.3), aqui são utilizadas seis funções objetivo: $Depart\ delay$, $Trip\ duration$, $Wait\ steps$, $Time\ loss$, $CO_2\ abs$ e $fuel\ abs$. O uso de várias funções objetivo e simuladores aproxima a modelagem proposta de situações reais, já que é importante considerar diferentes medidas de qualidade ao definir ciclos de semáforo. Para modelar o problema, foi desenvolvido um sistema e utilizado o SUMO. O sistema funciona em conjunto com cinco módulos principais, onde diversas tarefas se comunicam. A Figura 11 ilustra a integração entre o sistema e os módulos. As intereseções entre o círculo "Sistema" e os demais círculos representam a comunicação entre o sistema e esses módulos. Sendo assim, o sistema não se comunica diretamente com o módulo algoritmo de busca, e sim com o módulo jMetal.

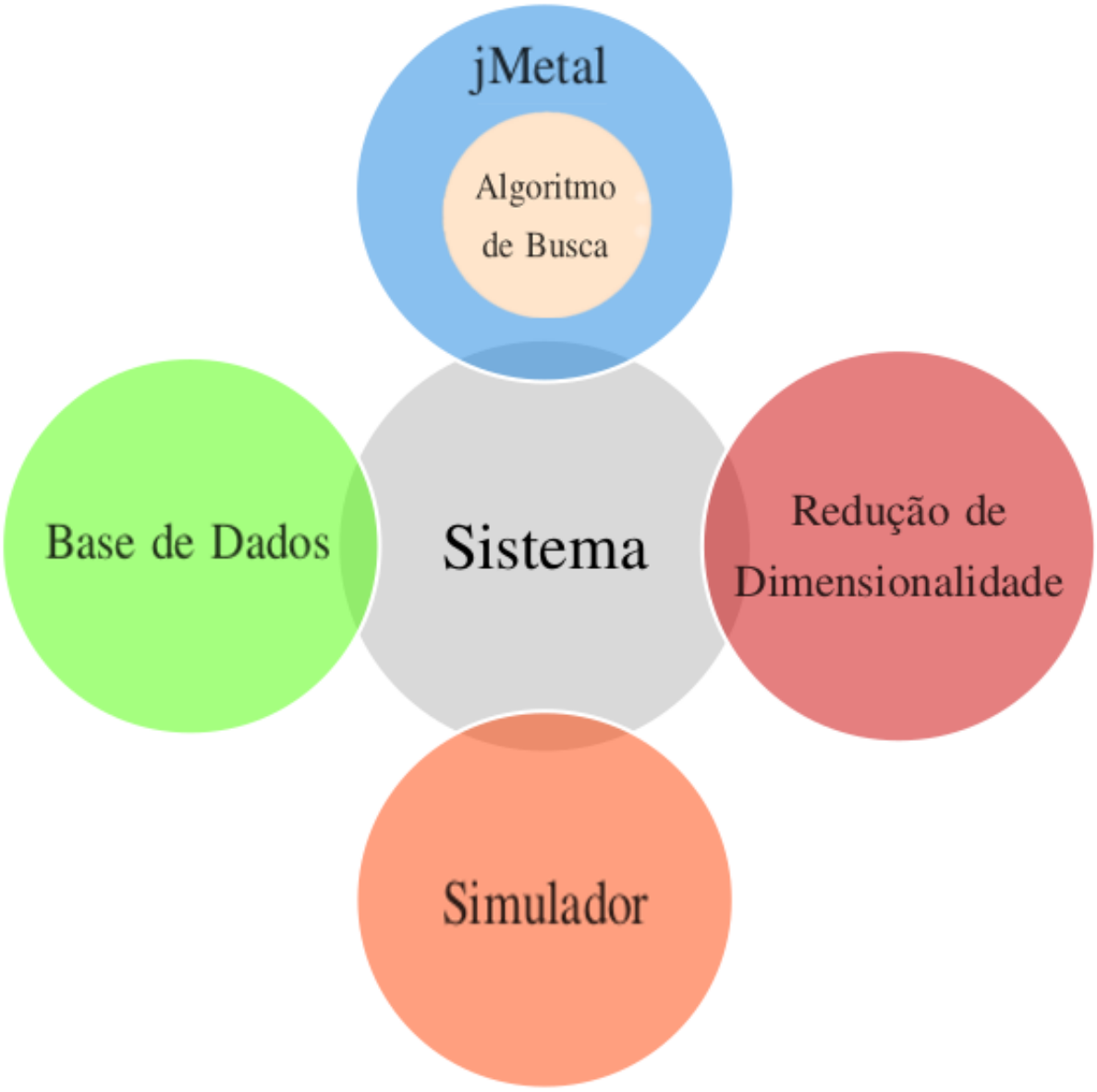
Quatro módulos intermediários foram desenvolvidos para garantir a comunicação entre o sistema e os demais módulos. Assim, foi possível efetuar a comunicação entre um algoritmo de busca e o simulador, e reduzir o número de simulações quando utilizada uma base de dados. Além disso, também foi possível incorporar um mecanismo de redução de dimensionalidade ao processo, onde foi possível executar experimentos com um número reduzido de objetivos.

Na Figura 11, os módulos representam: o *framework* jMetal, os algoritmos de busca, o SGBD, o *framework* responsável pela redução de dimensionalidade e o simulador de tráfego

SUMO, onde, para simplificar a leitura, cada módulo será representado, respectivamente, pelas siglas: MJM, MAB, MBD, MRD e MSI.

Os detalhes do sistema serão apresentados na seção seguinte. O simulador, os algoritmos de busca e a técnica de redução de dimensionalidade aplicados ao problema e a base de dados serão abordados, respectivamente, nas subseções 4.1.1, 4.1.3, 4.1.4 e 4.1.5. Já o *benchmark*, os experimentos e os resultados, serão abordados no próximo capítulo.

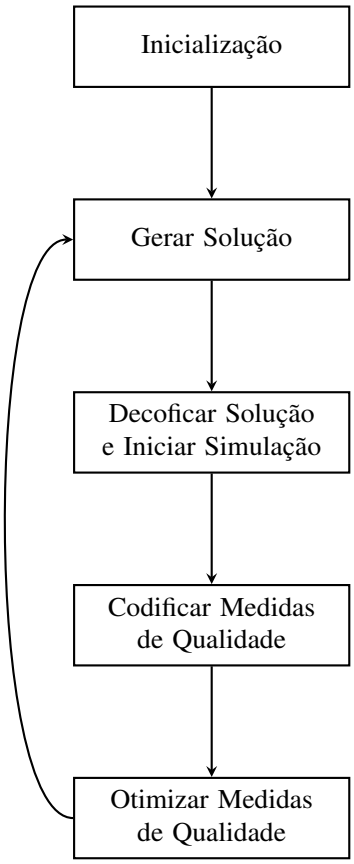
Figura 11 – Integração entre o sistema desenvolvido e os módulos principais



4.1 Sistema Desenvolvido

Várias tarefas compõe os módulos e o sistema desenvolvido; as principais tarefas são: inicialização, gerar uma solução, decodificar a solução e iniciar simulação, codificar as medidas de qualidade e otimizar as medidas de qualidade (como ilustrado na Figura 12). A redução de dimensionalidade é aplicada de forma *offline*, e por isso será abordada apenas na subseção 4.1.4.

Figura 12 – Representação esquemática da otimização semafórica



Na primeira tarefa (inicialização), é necessário configurar um cenário de tráfego que é composto basicamente pelo mapa viário de uma região, semáforos e fluxo de veículos, e definir quais medidas de qualidade serão utilizadas. Para configurar o cenário de tráfego o sistema se comunica com o MSI onde toda a configuração de um cenário é feita (isso será explorado no próximo capítulo (subseção 5.1.2)). Então, o sistema extrai do MSI a quantidade de fases dos semáforos a serem otimizadas e informa ao MJM que, por sua vez, informa a quantidade ao MAB. O sistema também informa ao MJM que, por sua vez, informa ao MAB, a restrição de quais valores de duração cada fase pode assumir e a quantidade de medidas de qualidade que serão utilizadas.

Na segunda tarefa, o MAB irá gerar uma solução para ser utilizada por ele e pelo MSI. O tamanho dessa solução é baseada no cenário, mais especificamente na quantidade de fases extraídas anteriormente. Para garantir que o MAB e o MSI utilizem a mesma solução, o sistema

possui módulos de comunicação com o MSI e o MJM. Assim, é possível realizar a comunicação entre o MSI, o MJM e o MAB. A tarefa gerar solução será descrita nas subseções 4.1.2 e 4.1.3.

Na terceira tarefa, a solução gerada pelo MAB é enviada ao MJM e então enviada ao sistema, onde terá de ser decodificada antes de ser utilizada pelo MSI. Ou seja, antes de ser utilizada para definir os novos tempos das fases dos semáforos e então iniciar uma simulação. A decoficação é necessária por que o simulador trabalha apenas com a linguagem XML e a solução gerada possui codificação inteira ou binária. Após a tarefa de decodificar uma solução, retoma-se comunicação com o MSI e a simulação tem início. Ao término da simulação, o MSI irá dispor como saída diversas medidas de qualidade em um arquivo XML. Detalhes da tarefa de decodificação são descritos na subseção 4.1.2. Já os arquivos do MSI onde os tempos das fases dos semáforos são informados e onde são disponibilizados as medidas de qualidade, serão descritos na subseção 4.1.1.

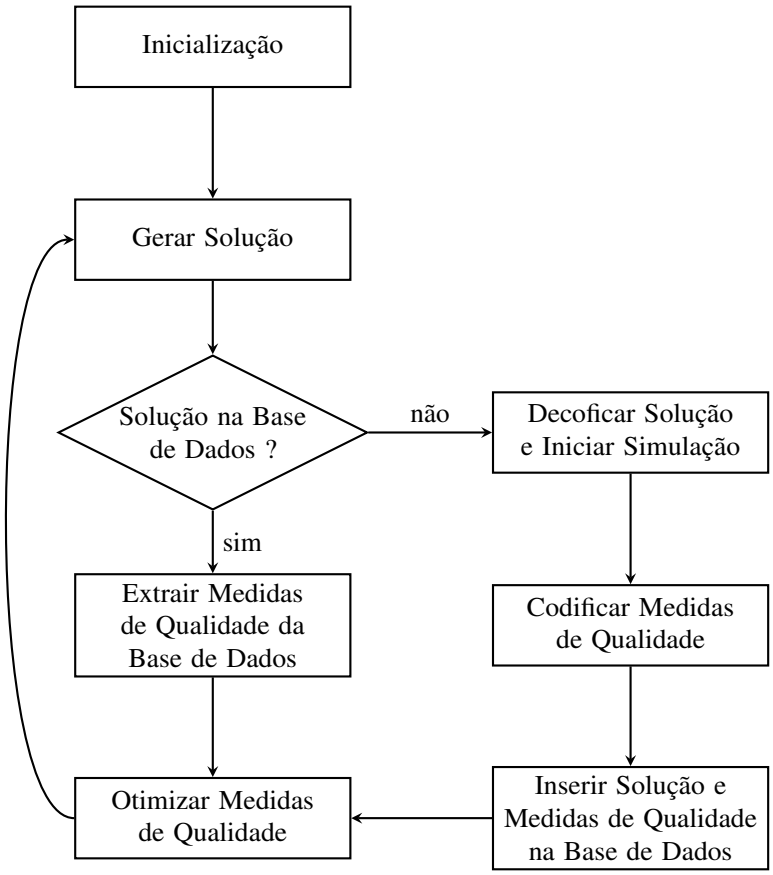
Na quarta tarefa, será efetuado o cálculo da funções objetivo. Para tal fim, o sistema irá extrair do MSI as medidas de qualidade (informadas pelo sistema na tarefa de inicialização), em seguida estas serão codificadas (de XML para *double*), normalizadas, passarão por um processo somatório (4.1), inseridas em um vetor e devolvidas para o MJM, e então enviadas ao MAB (os detalhes do processo de normalização serão descritos no próximo capítulo (subseção 5.3.3)). Na quinta tarefa, o MAB irá avaliar e otimizar essas medidas. Por fim, o MAB irá gerar uma nova solução. Esse ciclo se repetirá até que encontre o critério de parada. Ao encontrar o critério de parada, o MAB irá enviar ao MJM as soluções não dominadas que, por sua vez, irá armazená-las em um arquivo texto.

O sistema tem a opção de utilizar uma base de dados (subseção 4.1.5) que, para cada cenário de tráfego, armazena, em cada registro, uma solução e as medidas de qualidade correspondentes a essa solução (como ilustrado na Figura 13).

Caso opte por utilizar uma base de dados, durante a inicialização (primeira tarefa), o sistema irá acessar o MBD informando o mapa viário e o fluxo de veículos do cenário de tráfego. Então, após uma solução ser gerada (segunda tarefa) pelo MAB e posteriormente enviada MJM, o sistema irá solicitar essa solução ao MBD (terceira tarefa). Uma vez que obtenha êxito, o sistema irá devolver (fluxo “sim” - primeira tarefa) as medidas de qualidades obtidas no MBD para o MJM e então para para MAB, que irá avaliar e otimizar essas medidas (quarta tarefa). Por fim, o MAB irá gerar uma nova solução. Esse ciclo se repetirá até que encontre o critério de parada. Ao encontrar o critério de parada, o MAB irá enviar ao MJM as soluções não dominadas que, por sua vez, irá armazená-las em um arquivo texto.

Caso a solução não seja encontrada na base dados, a solução terá de ser decodificada (fluxo “não” - primeira tarefa) pelo sistema antes de ser utilizada pelo MSI, tal como descrito anteriormente, até a tarefa em que o sistema extrai, codifica, normaliza e aplica o processo somatório as medidas (fluxo “não” - segunda tarefa). Após essa tarefa, o sistema irá enviar as medidas de qualidade e a solução para o MBD, para que estas sejam inseridas, em um mesmo

Figura 13 – Representação esquemática da otimização semafórica com base de dados



registro, na base de dados (fluxo "não" - terceira tarefa). Então, o sistema devolverá as medidas de qualidades ao MJM, e então para MAB que irá avaliar e otimizar essas medidas (quarta tarefa). Por fim, o MAB irá gerar uma nova solução. Esse ciclo se repetirá até que encontre o critério de parada. Ao encontrar o critério de parada, o MAB irá enviar ao MJM as soluções não dominadas que, por sua vez, irá armazená-las em um arquivo texto.

4.1.1 Simulador

De modo geral, para descrever um cenário de tráfego se faz necessário o uso de uma ferramenta de simulação. Para isto, é necessário informar o mapa viário da região, a localização e o tempo das fases dos semáforos, e os dados característicos dos veículos. A partir daí, toma-se como base as saídas oriundas do simulador para definir as funções objetivo dos algoritmos de busca. Utilizando o SUMO (seção 2.2) como ferramenta de simulação, a descrição do cenário de tráfego, a definição dos tempos dos semáforos e a coleta das medidas, se dá através da leitura e escrita de arquivos XML.

Em um arquivo (*net.xml*) é possível editar informações a respeito dos semáforos, tais como o estado (representado por uma *string*) e a duração (em segundos) de casa fase. Sendo assim, sempre que se fizer necessário alterar a duração de uma ou mais fases de um ou mais semáforos, basta acessar o arquivo e alterar o valor correspondente a cada fase de cada semáforo.

No exemplo contido em Código 1, o tempo de cada fase é o valor do atributo *duration*. Aqui, apenas a primeira e a terceira *tags phase* seriam acessadas pois os estados (atributo *state*) das outras duas *tags* são formados apenas por vermelho e amarelo, e por isso tem o tempo de duração fixado em quatro segundos.

Código 1 – Fases de um semáforo

```
<tlLogic id="1" type="static" programID="0" offset="0">
    <phase duration="50" state="GGGrrrGGGrrr"/>
    <phase duration="4" state="yyyrrryyyrrr"/>
    <phase duration="35" state="rrrGGGrrrGGG"/>
    <phase duration="4" state="rrryyyrrryyy"/>
</tlLogic>
```

Quanto à coleta das medidas de qualidade, é necessário definir quais medidas se deseja obter. A depender de quais forem, o SUMO permite gerar como saída arquivos XML contendo diferentes medidas. Dentre as medidas disponíveis, o SUMO permite discriminadar em um único arquivo (*trips.xml*), informações a respeito de todas as viagens dos veículos que participaram da simulação. No exemplo contido em Código 2, cada atributo dentro das *tags tripinfo* e *emissions* é uma medida de qualidade referente a uma viagem realizada por um veículo. No arquivo *trips.xml*, o número de *tags tripinfo* e *emissions* é igual ao número total de viagens realizadas durante uma simulação.

Código 2 – Viagem de um veículo

```
<tripinfo departDelay="60.27" arrival="130.00" arrivalSpeed="58.42" duration="56.00"
 ↪  waitSteps="20" timeLoss="45.53">
    <emissions CO_abs="17221.73" CO2_abs="777675.11" HC_abs="97.89" PMx_abs="20.34"
     ↪  NOx_abs="356.07" fuel_abs="334.28"/>
</tripinfo>
```

Assumindo que, para cada simulação, cada veículo realizará apenas uma viagem (nó de origem e nó de destino), cada medida (m) será o somatório desta, obtida em cada veículo (v).

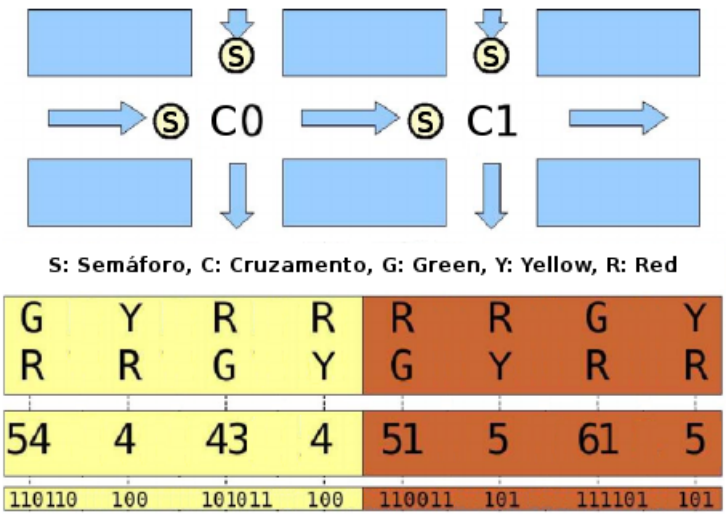
$$Medida(m) = \sum_{i=1}^{num\_v} v_i \tag{4.1}$$

O sistema é responsável não apenas pela comunicação com o simulador, alterar os tempos das fases, iniciar simulação e extrair as medidas oriundas do simulador; como também é responsável por codificar os dados que serão utilizados pelo algoritmo de busca, além de decodificar os dados oriundos do algoritmo de busca antes de serem utilizados pelo simulador. Isso será detalhado na próxima subseção.

4.1.2 Representação do Problema da Sincronização de Semafóros como um Problema de Otimização com Muitos Objetivos

Para modelar o problema da sincronização de semáforos como um problema de otimização, é necessário que dada uma combinação de semáforos, se construa uma representação computacional do tempo de duração de cada fase para cada semáforo. O valor que a duração pode assumir é um conjunto finito, geralmente com poucas opções (variando entre quatro e cem). A Figura 14 destaca a representação de um vetor para uma rede de tráfego formada por dois cruzamentos com dois semáforos em cada um. Nesse cenário, com apenas quatro semáforos, é possível visualizar as fases de cada semáforo em cada um dos dois cruzamento. As duas cores (amarela e marrom) separam os dois cruzamentos em duas regiões, onde, cada coluna representa uma fase de um cruzamento. No primeiro momento, as fases são representadas pelo estados de cada semáforo durante a fase: verde (G), amarelo (Y) ou vermelho (R). Depois, em um segundo momento, há uma reprentação do tempo de duração, em inteiro, de cada uma das fases. Por fim, há uma representação binária do tempo de duração de cada fase, onde, tanto a representação inteira, quanto a binária, podem ser utilizadas em um algoritmo de busca.

Figura 14 – Codificação cromossômica



Fonte: Adaptado de Javier et al (2010)

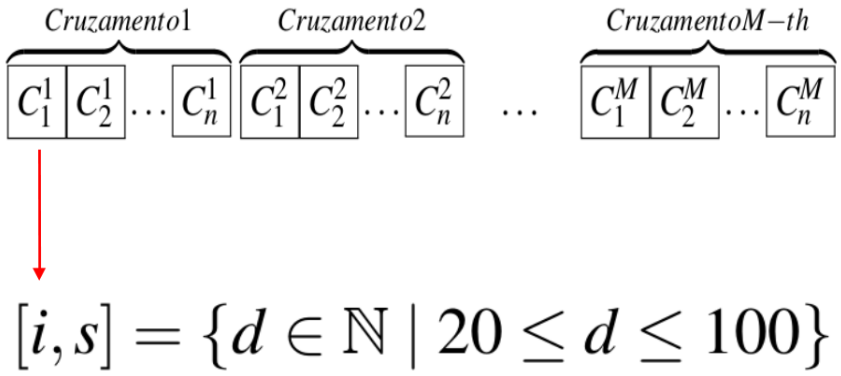
A representação computacional do tempo de duração das fases dos semáforos, pode assumir uma representação matemática por meio de um vetor solução, x, de variáveis inteiras, tal que:

$x = (C_1^1, C_2^1, \ldots, C_n^1, C_1^2, \ldots, C_n^2, C_1^M, \ldots, C_n^M)$, onde C_j^k representa a duração, d, da j-ésima fase (otimizável) do k-ésimo cruzamento e $20 \leq d \leq 100$, $d \in \mathbb{N}$.

A partir de cada solução x, por meio de uma simulação, o SUMO calcula os valores das m-funções objetivo selecionadas para formar o vetor objetivo $f(x) = (f_1(x), f_2(x), \ldots, f_m(x))$ que, posteriormente, será avaliado pelo algoritmo de busca.

Sendo assim, é possível tratar o problema da sincronização de semáforos como um problema de otimização, representando computacionalmente uma possível combinação dos tempos de fase dos semáforos através de um vetor. Neste trabalho (Figura 15) todos os elementos do vetor devem estar dentro do intervalo contido nos limites inferior (i) e superior (s) da duração (d) das "fases verdes". Para todo cruzamento (variando de 1 até M), o tamanho do vetor a ser otimizado será formado pelas fases (variando de 1 até N) onde em ao menos uma direção tenha-se um estado verde, uma "fase verde". Isto implica que a fase em que as direções são formadas apenas por estados vermelhos, ou apenas por estados amarelos e vermelhos, terá seu tempo ajustado em 4 segundos e não será usada no vetor, como ilustrado no Código 1 na segunda e na quarta *tags* "phases".

Figura 15 – Vetor das fases de todos cruzamentos



Além da representação vetorial, é necessário um conjunto de medidas de qualidade a serem utilizadas como funções objetivo. Sempre que uma solução tiver que ser avaliada, sua representação vetorial deve ser decodificada e utilizada na simulação. Com isso obtem-se os valores das medidas desejadas que, como mostrado no capítulo 2 (seção 2.2), são diversas disponíveis. Por se tratar de várias medidas disponíveis e uma representação vetorial, o problema pode ser classificado como Problema com Muitos Objetivos.

Cada solução deve executar uma simulação completa em SUMO: uma solução representa o tempo de cada fase que será otimizada em cada cruzamento. Essa solução, após ser decodificada

(para XML), é utilizada para atualizar o arquivo *net.xml*. Após a simulação terminar, todas medidas de qualidade obtidas pelo vetor solução são apresentadas no arquivo *trips.xml*. Essas medidas de qualidades serão extraídas, codificadas (de XML para *double*), normalizadas, passarão por um processo somatório (4.1) e então inseridas no vetor objetivo. Os detalhes do processo de normalização serão descritos no próximo capítulo (subseção 5.3.3).

4.1.3 Algoritmos Aplicados à Sincronização de Semáforos

Os algoritmos NSGA-II e NSGA-III foram apresentados no capítulo 3 (subseções 3.1.1 e 3.2.1). No atual capítulo mostrou-se que o problema da sincronização de semáforos pode ser tratado como um problema de otimização, representando computacionalmente uma possível combinação dos tempos de fase dos semáforos através de um vetor. Uma vez com o vetor em mãos, podemos usar um algoritmo de busca para definir os tempos das fases.

Aqui, o problema de otimização é um problema com muitos objetivos, onde: um indivíduo representa um plano de tempo (fases "verdes") dos semáforos e as funções de *fitness* de cada indivíduo são obtidas pelo simulador, como discutido anteriormente. Aqui, um indivíduo é definido como um vetor de números inteiros, assumindo uma representação binária.

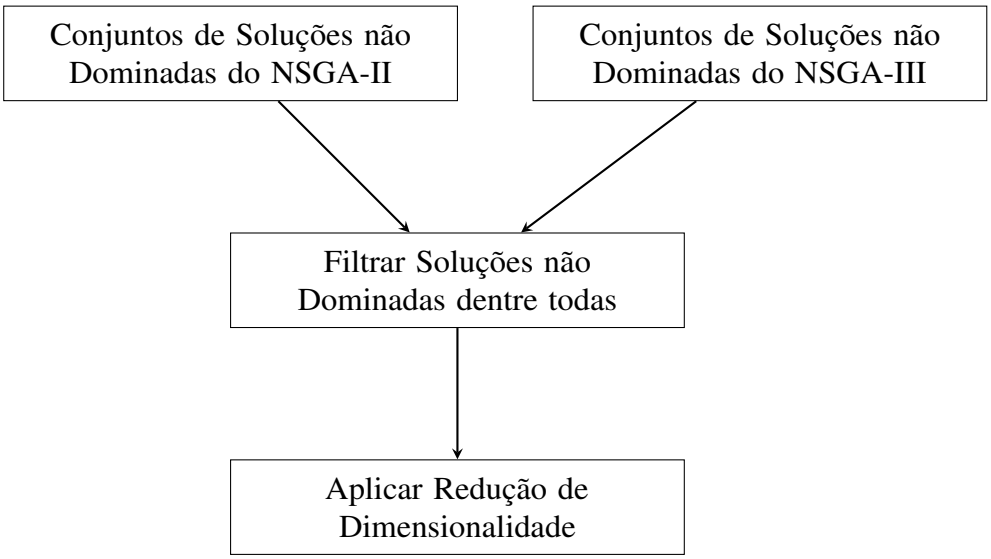
Os principais passos dos algoritmos de busca junto ao sistema proposto são: primeiro, a população é inicializada; Cada nova solução deve ser avaliada através da execução do simulador e após o ciclo evolutivo começar. Neste *loop*, primeiro é realizado o procedimento Recombinação + Mutação para gerar o descendente. Uma vez que as soluções são vetores binários, qualquer operador evolutivo básico pode ser aplicado (tarefa *Gerar Solução* das Figuras 12, 13 e 17). Cada nova solução deve ser avaliada (novamente executando uma simulação, como na tarefa *Decodificar Solução e Iniciar Simulação* das Figuras 12, 13 e 17). Finalmente, o algoritmo deve definir a nova população com base nos pais e descendentes.

Assim como o NSGA-II, no NSGA-III o algoritmo *fast-non-dominated-sort* é executado para definir a nova população, sendo que o NSGA-III difere do NSGA-II pelo operador de *crowding*. No NSGA-III, o processo de seleção baseado no conjunto de pontos de referência é formado por três procedimentos: normalização, associação e preservação do nicho (DEB; JAIN, 2014). Este processo ocorre na tarefa *Otimizar Medidas de Qualidade* das Figuras 12, 13 e 17. Por fim, após a seleção, o algoritmo continua até atingir um critério de parada.

4.1.4 Redução de Dimensionalidade Aplicada ao Problema

Existem diversas técnicas de redução de dimensionalidade, algumas delas foram mencionadas no capítulo anterior (seção 3.2.2.1). Aqui, utilizou-se a técnica DRP através dos algoritmos presentes no *framework* desenvolvido por Dimo Brockhoff (BROCKHOFF; ZITZLER, 2009). Neste trabalho, todas as tarefas do MRD são executadas de forma *offline*, depois das execuções dos algoritmos de busca. O objetivo do MRD é encontrar o subconjunto mínimo de objetivos

Figura 16 – Integração inicial entre o sistema e o módulo redução de dimensionalidade



que produza um erro $\delta = 0$, ou seja, sem alterar a relação de dominância das soluções não dominadas. A integração inicial entre o sistema e o MRD (e suas principais tarefas) resumem-se na Figura 16.

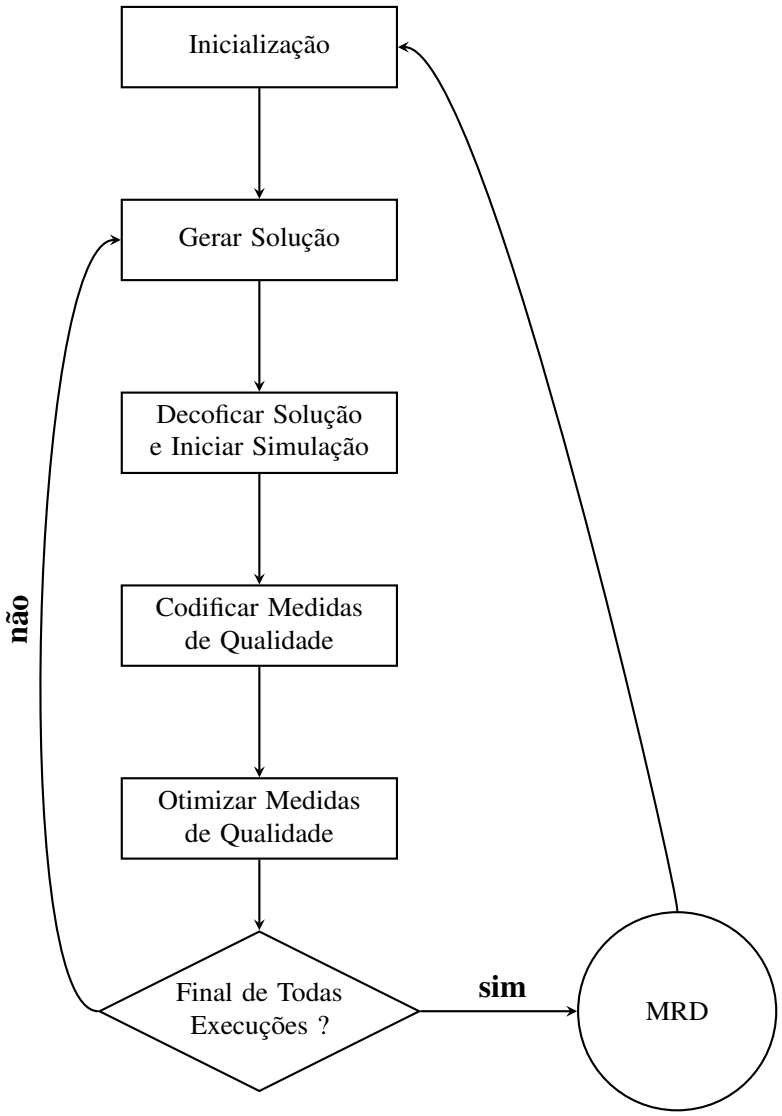
Na primeira tarefa, o sistema reúne em um único arquivo texto cada um dos arquivos gerados pelo MJM, após cada execução dos algoritmos (NSGA-II e NSGA-III). Cada um dos arquivos contém um conjunto de soluções (indivíduos) não dominadas, que são, de fato, as melhores soluções de cada execução em cada algoritmo. Ou seja, em cada conjunto não há uma única melhor solução, e sim um conjunto de melhores soluções, que é igual ao tamanho da população em cada algoritmo (o tamanho da população, dentre outros parâmetros, serão abordados no próximo capítulo). Então, o sistema insere, nas duas primeiras linhas desse arquivo, a quantidade de soluções e de objetivos (parâmetros essenciais para o *framework*), e entrega esse arquivo ao MRD. As próximas tarefas são efetuadas pelo MRD através dos algoritmos presentes no *framework*.

A segunda tarefa tem como objetivo filtrar, dentre todas as soluções, as não dominadas. Para tal, utiliza-se o algortimo *frontCalculator* passando como parâmetro o arquivo gerado na tarefa anterior. Esse algoritmo calcula os pontos não dominados no conjunto das soluções contidas no arquivo. Se houver pontos não dominados, o arquivo é atualizado com apenas as soluções não dominadas entre todas. Caso contrário, é mantido o arquivo original que foi passado como parâmetro para o *frontCalculator*.

Na terceira tarefa, é feita a tentativa de redução de dimensionalidade, através do algoritmo *conflictCalculator*. Nessa tarefa, optou-se por utilizar um algoritmo exato e não aceitando nenhum erro ($\delta = 0$). Essa escolha (algoritmo exato e $\delta = 0$) se dá através da passagem de parâmetros para o *conflictCalculator*. Por fim, o *framework* informa como saída se é possível reduzir o número de objetivos e, uma vez que seja possível, quais objetivos formam o subconjunto.

A Figura 17 representa a intregação do MRD ao processo da otimização semafórica. Dentro dessa representação, a redução de dimensionalidade é alocada ao término do processo (fluxo "sim" - única tarefa), após todas as execuções dos algoritmos de busca. Ao final dessa tarefa, o MRD informa ao sistema a saída do *framework*. Por fim, utilizando essa saída, o sistema define quais medidas de qualidade serão utilizadas na tarefa de inicialização.

Figura 17 – Representação esquemática da otimização semafórica com redução de dimensionalidade



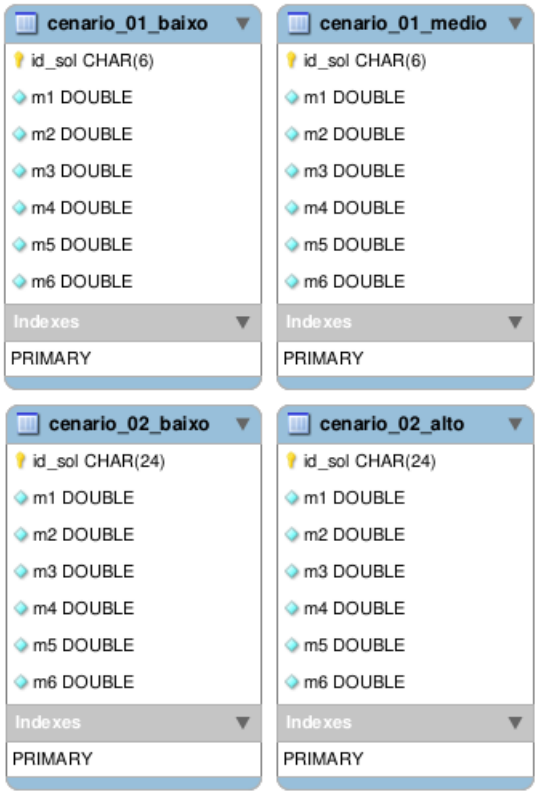
4.1.5 Base de Dados

O MBD foi desenvolvido com intuíto de não efetuar uma simulação que já tenha sido efetuada em outra ocasião, ou seja, uma simulação desnecessária, repetida. Isso só é possível por que no SUMO, dado um mapa, uma configuração de tempo de fases e o fluxo de veículos distribuído em cada rota, o valor de cada medida de qualidade obtido ao término da simulação será sempre o mesmo.

A base de dados é gerenciada pelo SGBD MYSQL. Cada tabela representa um cenário de tráfego (mapa viário e o fluxo de veículos), onde será armazenado, em cada registro, uma solução (tempo das fases) e as medidas de qualidade correspondentes a essa solução. Em outras palavras, serão armazenados os dados de entrada e de saída do simulador referentes a cada simulação efetuada.

A Figura 18 ilustra quatro tabelas da base de dados, onde é possível notar que, independente do fluxo, a estrutura de cada tabela, para cada cenário, será a mesma. Isso ocorre por que em cada mapa viário, a quantidade de fases é um valor fixo e a quantidade de medidas que serão armezadas é a mesma em qualquer cenário. Detalhes da configuração dos cenários serão abordados no próximo capítulo.

Figura 18 – Tabelas da base de dados



O MBD apenas efetua as rotinas de inserção e consulta de registros. Cada registro é composto por uma chave primária (do tipo *char*) que armazena o valor da solução e por seis campos (do tipo *double*) que armazenam cada uma das seis medidas de qualidade. A chave primária irá armazenar os tempos de duração das fases "verdes", onde cada tempo é formado por três algarismos (por exemplo: 020, 085, 100). No Cenário 01 existem 2 fases "verdes", sendo assim, o tamanho do campo *char* é 6 e irá armazenar soluções tais como "045080" e "100100". Já no Cenário 02 existem 8 fases "verdes", sendo assim, o tamanho do campo *char* é 24 e irá armazenar soluções tais como "082100023100027100064063" e "025065051095026042085025"

5 Experimentos

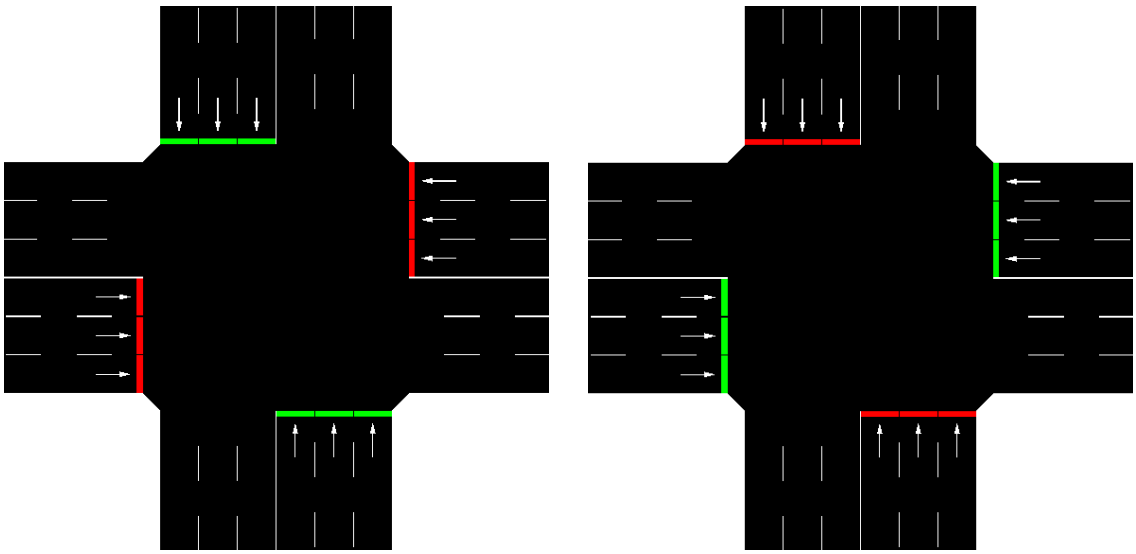
Este capítulo irá apresentar um conjunto de experimentos para avaliação do sistema desenvolvido. Neste estudo, para realizar os experimentos, foram definidos dois cenários, onde cada cenário é representado por uma rede e três fluxos de veículos: baixo, médio e alto. Em cada cenário, através do *framework* jMetal, dois algoritmos foram utilizados, o NSGA-II e o NSGA-III. Para esses algoritmos foram feitas análises de suas performances quando aplicada uma técnica de redução de dimensionalidade, como também quando não aplicou-se a técnica. O NSGA-III é voltado para problemas com muitos objetivos, já o NSGA-II é voltado para problemas multiobjetivo e foi previamente utilizado em trabalhos relacionados. Assim, foi possível avaliar diferentes algoritmos em um problema com muitos objetivos e identificar se a aplicação de técnicas de redução de dimensionalidade melhora o desempenho desses algoritmos. Este capítulo descreve o *benchmark* utilizado nos experimentos (seção 5.1) e apresenta mais detalhes do simulador (subseção 5.1.2). Também descreve os dois experimentos (seção 5.2), os parâmetros dos algoritmos (subseções 5.3.1 e 5.3.2) e o processo de normalização (subseção 5.3.3). Por fim, na seção 5.4 são detalhados os dois experimentos e discutidos os resultados.

5.1 *Benchmark*

Para medir a performance dos algoritmos, seis configurações de *benchmark* foram utilizadas. Essas configurações tem como principais parâmetros a rede de tráfego (número de cruzamentos) e o fluxo de tráfego (número de carros que efetuam uma viagem durante a simulação). Aqui, definem-se duas redes: Cenário 01 e Cenário 02. Também foram definidos três fluxos de tráfego diferentes: fluxo baixo, fluxo médio e fluxo alto. Estes parâmetros formam seis configurações diferentes: Cenário 01 com fluxo baixo, Cenário 01 com fluxo médio, Cenário 01 com fluxo alto, Cenário 02 com fluxo baixo, Cenário 02 com fluxo médio e Cenário 02 com fluxo alto.

Em cada uma das configurações de *benchmark*, além do fluxo de veículos e do número de cruzamentos, devem ser definidos o número de pistas, bem como o número de fases e a duração de cada fase. A Figura 19 representa as duas fases a serem otimizadas em qualquer cruzamento da rede, sendo as fases e o número de pistas os mesmos em todos os cruzamentos. A mesma figura mostra que em cada cruzamento passam 2 pistas de mão dupla, onde cada pista tem 6 faixas, ou seja, 3 faixas para cada mão (direção).

Figura 19 – *Benchmark* dos cenários. Fases dos cruzamentos.



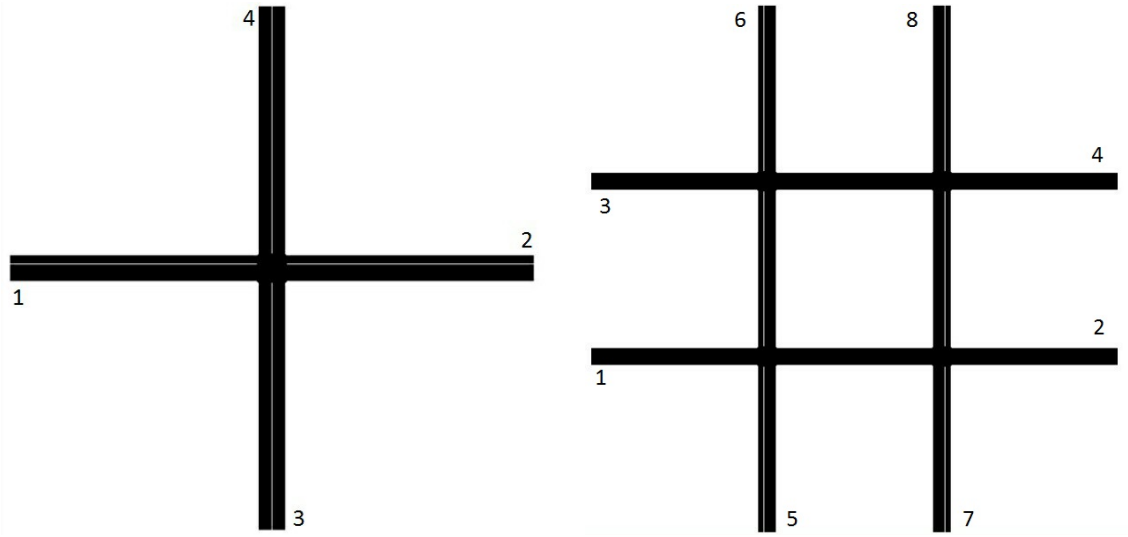
(a) Fase 01 (b) Fase 02

Cada mão da pista é representada por uma aresta no SUMO que possui 188 metros de comprimento, onde cada aresta é formada por um nó origem e um nó destino. Cada nó está localizado nas extremidades do mapa ou em um cruzamento, onde cada cruzamento possui 12 metros de comprimento. No primeiro cenário temos 8 arestas e 5 nós, sendo que em um desses nós encontram-se os semáforos. No segundo cenário temos 24 arestas e 12 nós, sendo que em quatro desses nós encontram-se os semáforos. Por fim, cada faixa é representada por conexões entre as arestas.

Os veículos são inseridos na simulação através dos injetores. Neste trabalho os injetores são na verdade todos os nós que estejam localizados nas extremidades do mapa (os injetores estão numerados na Figura 20(a) e na Figura 20(b)). Em todos os cenários o número de injetores é o dobro do número de pistas.

De forma aleatória foi definida a distribuição dos três fluxos de veículos nos injetores existentes em cada cenário. Essa definição foi feita previamente, antes de iniciar qualquer simulação e assim mantida em todos experimentos (subseções 5.4.1 e 5.4.2). Caso a distrubição entre os injetores não fosse mantida para os diferentes experimentos, os resultados fatalmente seriam diferentes. Isso ocorreria mesmo que os demais parâmetros da configuração de *benchmark* fossem os mesmos em um dado experimento.

Figura 20 – *Benchmark* dos cenários. Rede e injetores



(a) Rede do Cenário 01 (b) Rede do Cenário 02

Baseado no fluxo definido para cada rota (viagem), o SUMO insere os veículos no mapa de forma proporcional, assim a rota com maior número de veículos vai inserir mais veículos em um intervalo de tempo. Para respeitar essa proporcionalidade, o SUMO calcula a quantidade total de veículos atribuídos a cada rota, e divide essa quantidade pela duração do tempo total a ser simulado, que é de 01 hora. Neste trabalho a quantidade de rotas é igual ao número de injetores presentes na rede, pois cada rota tem como ponto (nó) de partida um injetor e como ponto (nó) de chegada o injetor que está exatamente no nó do lado oposto, seguindo as arestas em linha reta. Ou seja, cada veículo passa por apenas dois injetores (nó origem e nó destino).

Outra característica importante do *benchmark* é a função objetivo. Aqui foram utilizadas seis medidas diferentes, que são:

- *Depart delay*: O tempo que o veículo teve de esperar antes de iniciar sua viagem;
- *Trip duration*: O tempo que o veículo precisou para realizar sua viagem;
- *Wait steps*: O número de passos em que a velocidade do veículo foi inferior a $0.1m/s$;
- *Time loss*: O tempo perdido devido à condução abaixo da velocidade ideal ($60km/h$);
- CO_2 *abs*: O total de CO_2 emitido pelo veículo durante a viagem;
- *Fuel abs*: O total de combustível consumido pelo veículo durante a viagem.

5.1.1 Cenários

O Cenário 01 (Figura 20(a)) é composto por 01 cruzamento e 04 injetores. Para o fluxo baixo, 3000 veículos foram usados. Para o fluxo médio, 10500. Para o fluxo alto, 21000. Para o

Tabela 1 – Quantidade de veículos por injetor para cada fluxo no Cenário 01

Injetor	**Fluxo baixo**	**Fluxo médio**	**Fluxo alto**
01	1000	5000	10000
02	500	2000	4000
03	850	2500	5000
04	650	1000	2000

Tabela 2 – Quantidade de veículos por injetor para cada fluxo no Cenário 02

Injetor	**Fluxo baixo**	**Fluxo médio**	**Fluxo alto**
01	520	3500	7000
02	370	4000	8000
03	1500	1000	2000
04	650	5500	11000
05	800	2500	5000
06	710	1500	3000
07	450	1750	3500
08	1000	1250	2500

fluxo alto, cada simulação levou cerca de 7 segundos. A quantidade de veículos inseridos por cada injetor, para cada fluxo, é ilustrada na Tabela 1. Com essa configuração, a solução vai ser codificada em um vetor de duas posições ($n = 2$) pois o único cruzamento existente possui duas fases "verdes".

O Cenário 02 (Figura 20(b)) é composto por 04 cruzamentos e 08 injetores. Para o fluxo baixo, 6000 veículos foram utilizados. Para o fluxo médio, 21000. Para o fluxo alto, 42000. Para o fluxo alto, cada simulação levou cerca de 20 segundos. A quantidade de veículos inseridos por cada injetor, para cada fluxo, é ilustrada na Tabela 2. Com essa configuração, a solução vai ser codificada em um vetor de oito posições ($n = 8$) pois cada um dos 4 cruzamentos possui duas fases "verdes".

5.1.2 Arquivos de Configuração do SUMO

Como exposto na subseção 2.3.1, para gerar um arquivo contendo um mapa (vias e semáforos) é necessário configurar três arquivos e executar o módulo *NETCONVERT* do SUMO, passando esses três arquivos como parâmetro. Para ilustrar essa configuração são exibidos em Código 3, Código 4 e Código 5, arquivos semelhantes aos utilizados para gerar o mapa do Cenário 01. Por sua vez, para gerar um arquivo contendo o fluxo de veículos, uma das formas (a que foi utilizada neste trabalho) é configurar um arquivo contendo a quantidade de veículos que percorrerão cada rota (nó origem e nó destino) e executar o módulo *DUAROUTER* do SUMO,

passando como parâmetro tanto esse arquivo quanto o arquivo que contém o mapa. Para ilustrar essa configuração, é exibido em Código 6 um arquivo contendo o fluxo de veículos, semelhante ao utilizado no Cenário 01 com fluxo baixo.

Assim, tem-se os dois arquivos necessários para executar a simulação do Cenário 01, ou seja, o arquivo contendo o mapa (*net.xml*) e o arquivo contendo o fluxo (*flow.rou.xml*). Para simplificar, é ilustrado em Código 7 um arquivo que contém como entrada o mapa e o fluxo, e que tem como saída um arquivo (*trips.xml*) com as medidas oriundas de cada veículo ao término da simulação. Neste trabalho, um arquivo semelhante a Código 7 foi utilizado para dar início a cada simulação em todos os cenários.

Código 3 – Nós do Cenário 01

```
<nodes>
    <node id= "1" x= "0.00"    y= "200.00"  type="priority"/>
    <node id= "2" x= "0.00"    y= "0.00"    type="traffic_light"/>
    <node id= "3" x= "200.00"  y= "0.00"    type="priority"/>
    <node id= "4" x= "-200.00" y= "0.00"    type="priority"/>
    <node id= "5" x= "0.00"    y= "-200.00" type="priority"/>
</nodes>
```

Código 4 – Arestas do Cenário 01

```
<edges>
    <edge id="E1" from="4" to="2" name="" type="principal"/>
    <edge id="E7" from="2" to="4" name="" type="principal"/>
    <edge id="E5" from="3" to="2" name="" type="principal"/>
    <edge id="E2" from="2" to="3" name="" type="principal"/>
    <edge id="E3" from="5" to="2" name="" type="principal"/>
    <edge id="E9" from="2" to="5" name="" type="principal"/>
    <edge id="E6" from="2" to="1" name="" type="principal"/>
    <edge id="E8" from="1" to="2" name="" type="principal"/>
</edges>
```

Código 5 – Conexões entre as arestas do Cenário 01

```
<connections>
    <connection from="E1" to="E2" fromLane="0" toLane="0"/>
    <connection from="E1" to="E2" fromLane="1" toLane="1"/>
    <connection from="E1" to="E2" fromLane="2" toLane="2"/>
    <connection from="E8" to="E9" fromLane="0" toLane="0"/>
    <connection from="E8" to="E9" fromLane="1" toLane="1"/>
    <connection from="E8" to="E9" fromLane="2" toLane="2"/>
    <connection from="E3" to="E6" fromLane="2" toLane="2"/>
    <connection from="E3" to="E6" fromLane="0" toLane="0"/>
    <connection from="E3" to="E6" fromLane="1" toLane="1"/>
    <connection from="E5" to="E7" fromLane="2" toLane="2"/>
    <connection from="E5" to="E7" fromLane="0" toLane="0"/>
    <connection from="E5" to="E7" fromLane="1" toLane="1"/>
</connections>
```

Código 6 – Fluxo de veículos do Cenário 01 com fluxo baixo

```
<flows>
   <flow id="E1_2" from="E1" to="E2" begin="0" end="3600" number="1000" type="CarA"/>
   <flow id="E5_7" from="E5" to="E7" begin="0" end="3600" number="500"  type="CarA"/>
   <flow id="E3_6" from="E3" to="E6" begin="0" end="3600" number="850"  type="CarA"/>
   <flow id="E8_9" from="E8" to="E9" begin="0" end="3600" number="650"  type="CarA"/>
</flows>
```

Código 7 – Arquivo de configuração

```
<configuration>
    <input>
        <net-file value="net.xml" />
        <route-files value="flow.rou.xml" />
    </input>
    <output>
        <tripinfo-output value="output/tripinfos.xml" />
    </output>
</configuration>
```

5.2 Descrição dos Experimentos

No presente trabalho foram realizados dois experimentos. Cada experimento foi realizado utilizando os dois cenários citados na seção 5.1, além dos algoritmos NSGA-II e NSGA-III. Os experimentos buscaram identificar qual algoritmo teria melhor perfomance. Em cada experimento, para cada configuração de *benchmark*, cada um dos algoritmos efetuou 10 execuções. Sendo que, em cada execução foram realizadas 25200 avaliações da função objetivo, totalizando 252000 avaliações realizadas por cada algoritmo.

No Experimento 01, para representar o vetor solução, utilizou-se uma codificação inteira, os dados não foram normalizados e não foi utilizada nenhuma técnica de redução de dimensionalidade. Neste experimento foram utilizadas cinco das seis configurações de *benchmark* citadas na seção 5.1, são elas: Cenário 01 com fluxo baixo, médio e alto e Cenário 02 com fluxo baixo e alto. Para cada configuração de *benchmark* foram realizadas 20 execuções: 10 execuções do NSGA-II e 10 execuções do NSGA-III.

Após efetuar uma análise dos resultados do Experimento 01, algumas hipóteses foram levantadas buscando entender por que o NSGA-II superou NSGA-III para o problema em todas configuraçes de *benchmark*. As hipóteses foram: os pontos de referência (que são muito importantes pro NSGA-III) podem ter sido afetados de alguma forma devido a grande diferença entre valores, escalas e quantidade de medidas? O fato de usar uma construção de soluções com codificação inteira, dentro de um pequeno intervalo, em um vetor também pequeno, resultando em um espaço de busca muito pequeno, pode ter afetado o desempenho do NSGA-III? As medidas utilizadas seriam correlacionadas e assim estariam influenciando no desempenho dos algoritmos?

Para o problema dos pontos de referência a estratégia adotada foi normalizar os valores das medidas. Para o problema do espaço de busca, a estratégia foi utilizar uma codificação binária, aumentando o tamanho do intervalo e, portanto, aumentando o espaço de busca. E, para a questão das medidas correlacionadas, a estratégia foi incorporar técnicas da redução de dimensionalidade. Quanto ao tipo de codificação, este apenas influencia nos algoritmos, não há qualquer restrição em relação ao funcionamento simulador e ao tempo de simulação. Bem como, a quantidade de cruzamentos em um dado cenário, não afeta a codificação, seja ela inteira ou binária.

Com essas estratégias traçadas, decidiu-se projetar um segundo experimento. O Experimento 02 incorporaria técnicas de redução de dimensionalidade, realizaria normalização dos dados e, para representar o vetor solução, utilizaria codificação binária. Este experimento utilizou cinco das seis configurações de *benchmark* citadas na seção 5.1, são elas: Cenário 01 com fluxo baixo, médio e alto e Cenário 02 com fluxo baixo e médio. Para cada configuração de *benchmark* foram realizadas 40 execuções: 10 execuções do NSGA-II com redução de dimensionalidade e 10 execuções sem redução de dimensionalidade, 10 execuções do NSGA-III com redução de

dimensionalidade e 10 execuções sem redução de dimensionalidade.

Ainda no Experimento 02, para o Cenário 01 foi utilizada uma base de dados (abordada no Capítulo 4, seção 4.1 e subseção 4.1.5) que até então não havia sido utilizada. A base de dados foi utilizada por que, ao término do Experimento 01, notou-se que em experimentos nos quais o número de fases a serem otimizadas seja pequeno, é possível resolver o problema com um algoritmo exato, já que é possível gerar todas as soluções. Isso pode ser notado no Cenário 01, onde a quantidade de possíveis soluções é bastante limitada, pois a dimensão do vetor solução é de apenas duas posições, diferentemente do Cenário 02 onde o vetor possui oito posições. Essa diferença entre as dimensões dos vetores foi o que tornou inviável aplicar essa estratégia ao Cenário 02.

Somando o fato de que o vetor solução do Cenário 01 possui apenas duas posições, e que o valor que cada variável pode assumir em cada posição pertence ao intervalo fechado entre 20 e 100, isso implica em 81^2 (6561) possíveis soluções para cada fluxo do Cenário 01 e 81^8 possíveis soluções para cada fluxo do Cenário 02. Sendo assim, para cada fluxo do Cenário 01, após a base de dados atingir exatos 6561 registros, não foi necessário realizar nenhuma outra simulação, de um total de 1008000 avaliações (simulações) necessárias (cada um dos quatro algoritmos teria de realizar 252000 simulações). Como consequência dessa estratégia, o tempo total das execuções no Cenário 01 diminuiram drasticamente, em média 99%.

5.3 Algoritmos e Parâmetros

Nesta seção serão apresentados os parâmetros dos algoritmos NSGA-II e NSGA-III (5.3.1) e do algoritmo de redução de dimensionalidade (5.3.2) utilizados nos experimentos. Também será abordado o processo de normalização (5.3.3) utilizado no trabalho. Aqui, a letra n vai indicar o número de variáveis.

5.3.1 NSGA-II e NSGA-III

Tal como no NSGA-II, o algoritmo NSGA-III não requer a definição de qualquer novo parâmetro além dos parâmetros dos AG habituais, tais como o tamanho da população, critério de parada, as probabilidades de cruzamento e de mutação e os seus parâmetros associados. O número de pontos de referência H não é um parâmetro algorítmico, pois está diretamente relacionado com ao número de pontos de troca. Neste trabalho, H foi previamente definido de acordo com (DEB; JAIN, 2014). O tamanho N da população é dependente de H, com $N \approx H$. A localização dos pontos de referência é igualmente dependente da informação de preferência que o utilizador está interessado em obter nas soluções obtidas.

Os parâmetros utilizados na implementação do NSGA-II e NSGA-III foram definidos por padrão pelo *framework* jMetal e estão presentes na Tabela 3 e na Tabela 4. Para os dois algoritmos, o valor de n é 2 no Cenário 01 e 8 no Cenário 02. Quando os algoritmos possuem

Tabela 3 – Parâmetros dos NSGAs (codificação binária)

Parâmetro	**NSGA-II**	**NSGA-III**
Tamanho do indivíduo	n	n
Tamanho da população	100	56
Número de gerações	25200	25200
Probabilidade de cruzamento	0.9	0.9
Probabilidade de mutação	1/7	1/7
Índice de distribuição de cruzamento	20	30
Índice de distribuição de mutação	20	20
Método de cruzamento	*Single point*	*Single point*
Método de mutação	*Bit flip*	*Bit flip*
Operador de seleção	*Binary tournament*	*Binary tournament*

Tabela 4 – Parâmetros dos NSGAs (codificação inteira)

Parâmetro	**NSGA-II**	**NSGA-III**
Tamanho do indivíduo	n	n
Tamanho da população	100	56
Número de gerações	25200	25200
Probabilidade de cruzamento	0.9	0.9
Probabilidade de mutação	$1/n$	$1/n$
Índice de distribuição de cruzamento	20	30
Índice de distribuição de mutação	20	20
Método de cruzamento	*Integer SBX*	*Integer SBX*
Método de mutação	Polinomial	Polinomial
Operador de seleção	*Binary tournament*	*Binary tournament*

codificação binária (Tabela 3), a probabilidade de uma variável sofrer mutação é de $1/7$. O valor 7 é definido a partir do número de *bits* necessários para formar uma variável, onde, neste trabalho, o maior valor que uma variável pode assumir é 100 (Figura 15) que, em sua representação binária (1100100), precisa de 7 *bits*.

Para avaliar a performance de algoritmos multiobjetivo utilizam-se indicadores de qualidade. Estes indicadores são definidos como funções que mapeiam conjuntos de soluções para um número real. O hipervolume (ZITZLER et al., 2003) é uma métrica de maximização que determina a área coberta pela fronteira de Pareto aproximada, que é criada combinando os melhores valores encontrados para cada objetivo. É uma medida comum usada em trabalhos onde algoritmos multiobjetivo são comparados. Para comparar a performance dos algoritmos calculamos o hipervolume, considerando o ponto nadir. Nos experimentos deste trabalho, o ponto nadir foi obtido através dos valores de todas as execuções.

Para medir a diferença em cada comparação, o teste de *Wilcoxon* é aplicado aos hipervolumes a nível de significância de 5%. O teste indica se há alguma diferença estatística entre cada

conjunto de dados analisados e, em seguida, os valores médios são usados para identificar qual algoritmo tem os melhores valores.

5.3.2 Redução de Dimensionalidade

A redução de dimensionalidade foi abordada no capítulo anterior (subseção 4.1.4). Como já explicado anteriormente, a redução é aplicada de forma *offline*, depois das execuções dos algoritmos NSGA-II e NSGA-III. O pré-requisito inicial é reunir as soluções (indivíduos) não dominadas geradas em cada uma das 10 execuções do NSGA-II e do NSGA-III, com tamanho populacional de 100 e 56, respectivamente, gerando no máximo um total de 1560 soluções não dominadas. O responsável por reunir as soluções é o sistema, sendo as demais tarefas do processo realizadas pelo módulo MRD.

Neste trabalho, para cada uma das configurações de *benchmark*, sempre é necessário executar as 20 execuções com todas as medidas, aqui, seis ao total. Ao término do processo, o *framework* apontou que seria possível fazer uma redução de dimensionalidade, de seis medidas para quatro medidas. Isso ocorreu devido à redundância entre os pares de medidas: CO_2 *abs* e $Fuel$ *abs*, e *Wait steps* e $Time\ loss$. Quanto ao primeiro par de medidas, isso é possível por que a emissão de CO_2 (CO_2 *abs*) está linearmente relacionada com o consumo de combustível ($Fuel$ *abs*) (ELVIK, 2009) e (FREY; UNAL; COLYAR, 2003). Há ainda uma terceira relação sabida na literatura que aponta que ao reduzir a quantidade de paradas, se minimiza a aceleração e a frenagem, melhorando significativamente a economia de combustível. Porém, essa relação não é suficiente para eliminar uma dessas medidas.

Ao aplicar a redução de dimensionalidade partindo de um total de quatro objetivos, não foi possível reduzir o número de medidas ao término do processo. Ou seja, para realizar a redução seria necessário alterar a relação de dominância das soluções não dominadas, admitindo um erro $\delta > 0$. Sendo assim, como resultado final da redução de dimensionalidade, para todas as configurações de *benchmark*, foi possível executar os dois algoritmos (NSGA-II e NSGA-III) utilizando apenas quatro medidas, sendo elas:

- $Depart\ delay$: O tempo que o veículo teve de esperar antes de iniciar sua viagem;
- $Trip\ duration$: O tempo que o veículo precisou para realizar sua viagem;
- *Wait steps*: O número de passos em que a velocidade do veículo foi inferior a $0.1m/s$;
- CO_2 *abs*: O total de CO_2 emitido pelo veículo durante a viagem;

5.3.3 Normalização

O propósito da normalização foi minimizar os problemas oriundos do uso de medidas que possuem diferentes unidades e escalas, como pode ser visualizado na Tabela 6 referente ao

Experimento 01, onde os valores não foram normalizados. Essa diferença também é ilustrada no Código 2 do capítulo anterior. Após a realização do Experimento 5.4.1 e antes de iniciar Experimento 5.4.2, foram calculados os parâmetros a serem utilizados na normalização. Para o presente trabalho, foi escolhida a Normalização pelo Valor Máximo dos Elementos, que é um processo de normalização que não necessita do valor mínimo, apenas do valor máximo. Essa escolha se deu devido a dificuldade encontrada ao tentar mensurar o valor mínimo que cada medida pode assumir.

Com esse propósito, para cada configuração de *benchmark*, tomou-se como base os valores das medidas de todas as soluções geradas pelo Experimento 5.4.1. A partir desses valores foi possível calcular qual o valor máximo (MAX) que cada medida (m) alcançou em cada configuração de *benchmark*. Sendo o valor dessa medida, na verdade, o pior valor encontrado, pois se trata de um problema de minimização. A constante MAX é o pré-requisito necessário para realizar a normalização das medidas no Experimento 5.4.2.

A função de normalização utilizada neste trabalho pode ser definida da seguinte forma: $f(m) = m/MAX$, onde m é a medida a ser normalizada e a constante MAX é o valor máximo que a medida consegue assumir na configuração de *benchmark* que está sendo executada. Essa função é aplicada ao término de cada simulação, antes de ser feita a avaliação de *fitness*.

5.4 Experimentos e Resultados

Nesta seção, serão detalhados os dois experimentos realizados e discutidos os resultados. Cada experimento foi realizado utilizando os dois dos cenários citados na seção 5.1, além dos algoritmos NSGA-II e NSGA-III. Para cada uma das 10 execuções de cada algoritmo, efetuou-se 25.200 avaliações. Em ambos experimentos, duas medidas tiveram sua escala alterada para melhor visualização dos dados: o $depart\ delay$ calculado em horas (anteriormente em segundos) e o $CO_2\ abs$ calculado em quilogramas (anteriormente em miligramas). As outras medidas têm as suas grandezas expressas da seguinte forma: $duration$ (segundos), $time\ loss$ (segundos), $fuel\ abs$ (mililitros) e $wait\ steps$ (valor inteiro que contabiliza quantas vezes $velocidade < 0.1m/s$).

5.4.1 Experimento 01

Aqui, utilizamos cinco das seis configurações de *benchmark* citadas na seção 5.1, são elas: Cenário 01 com fluxo baixo, médio e alto e Cenário 02 com fluxo baixo e alto. Como dito anteriormente, nesse experimento os dados não foram normalizados e nenhuma técnica de redução de dimensionalidade foi aplicada. Sendo assim, para cada configuração de *benchmark* foram realizadas 20 execuções: 10 execuções do NSGA-II e 10 execuções do NSGA-III.

5.4.1.1 Hipervolume e P-value

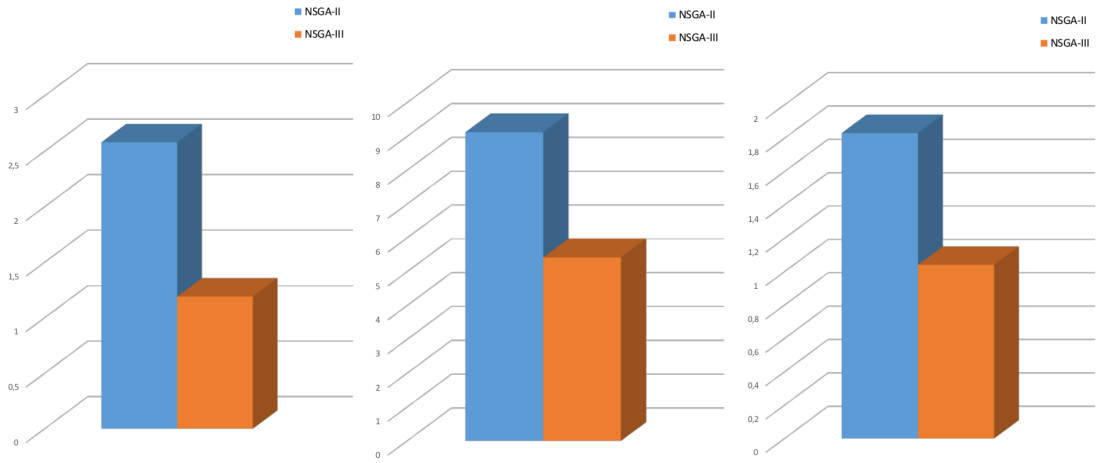
Observando a Tabela 5, cada linha representa um cenário e um fluxo de tráfego. Na segunda e terceira colunas são exibidos, para cada algoritmo, os valores da média e do desvio padrão (entre parênteses) do hipervolume. E na quarta coluna é exibido o valor do $p-value$ obtido pelo teste de *Wilcoxon*. Em todos os cenários NSGA-II teve os melhores resultados, porque os valores da média do hipervolume foram mais altos.

Tabela 5 – Experimento 01. Valores médios do hipervolume, desvio padrão (entre parênteses) e p-value

Cenário (Fluxo)	NSGA-II	NSGA-III	P-value
01 (baixo)	$2.58x10^{14}$ ($3.23x10^{13}$)	$1.19x10^{14}$ ($4.31x10^{13}$)	$1.016x10^{-10}$
01 (médio)	$9.12x10^{25}$ ($1.02x10^{24}$)	$5.42x10^{25}$ ($4.66x10^{24}$)	$1.083x10^{-5}$
01 (alto)	$1.83x10^{28}$ ($4.06x10^{26}$)	$1.04x10^{28}$ ($1.34x10^{27}$)	$1.083x10^{-5}$
02 (baixo)	$6.73x10^{17}$($1.97x10^{17}$)	$1.24x10^{17}$ ($1.98x10^{17}$)	0.0001299
02 (alto)	$4.95x10^{33}$($9.27x10^{32}$)	$4.06x10^{33}$($1.04x10^{33}$)	0.05243

No Cenário 01, ao dividir o valor das médias do hipervolume do NSGA-II pelo do NSGA-III, obtém-se que: para o fluxo baixo o NSGA-II é 2.16 maior que NSGA-III, para o fluxo alto é 1.75 maior, porém, para o fluxo médio é 1.68 maior, uma melhora em relação ao fluxo alto. Já para o Cenário 02, quando efetuado o mesmo procedimento, obtém-se que: para o fluxo baixo o NSGA-II é 5.42 maior que NSGA-III e para o fluxo alto é 1.22 maior. A partir desses valores, é possível afirmar que, em ambos cenários, o ganho de performance do NSGA-III foi maior que o ganho de performance do NSGA-II, quando o fluxo passou de baixo para alto. Porém, não sendo suficiente para que a média do hipervolume do NSGA-III ultrapasse a média do NSGA-II.

Figura 21 – Experimento 01. Cenário 01. Valores médios do hipervolume

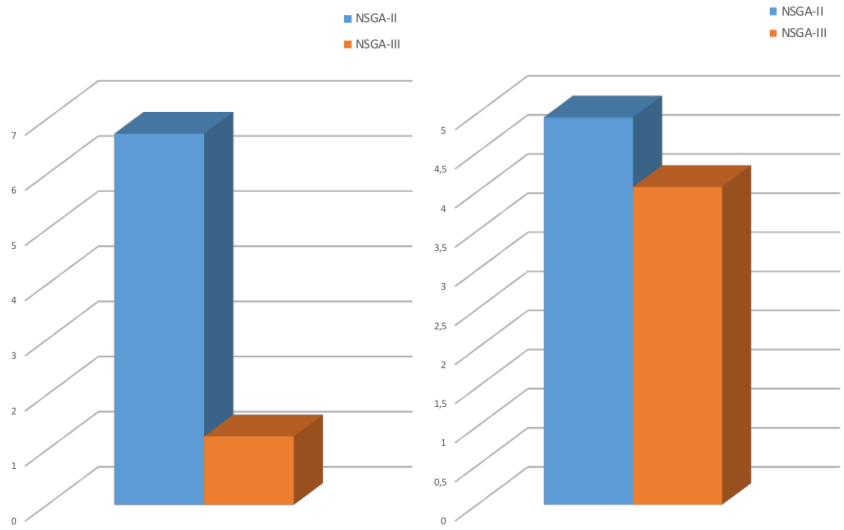


(a) Fluxo baixo (b) Fluxo médio (c) Fluxo alto

É possível notar (na última coluna) através do $p-value$, que há diferença estatística nos

quatro primeiros experimentos, porque $p - value < 0.05$. No entanto, no último experimento (Cenário 02 com fluxo alto), não há diferença estatística, pois $p - value > 0.05$. A Figura 21 ilustra o valor médio do hipervolume encontrado em cada um dos dois algoritmos para cada fluxo do Cenário 01, ou seja, a segunda e terceira colunas nas três primeiras linhas da Tabela 5 sem os valores do desvio padrão. A Figura 22, por sua vez, ilustra o valor médio do hipervolume encontrado em cada um dos dois algoritmos para o fluxo baixo e alto do Cenário 02, ou seja, a segunda e terceira colunas nas duas últimas linhas da Tabela 5 sem os valores do desvio padrão.

Figura 22 – Experimento 01. Cenário 02. Valores médios do hipervolume



(a) Fluxo baixo (b) Fluxo alto

5.4.1.2 Média das Medidas

Além do hipervolume e do teste de *Wilcoxon*, é feito o cálculo dos valores médios de todas as funções objetivo, de todas as soluções não dominadas encontradas para ambos algoritmos, ilustrados na Tabela 6. Para encontrar esses valores, é feito o cálculo da média aritmética, que se deu da seguinte forma: para todas as soluções não dominadas de cada algoritmo, em cada configuração de *benchmark*, para cada medida, efetuo-se a soma de todos os valores encontrados para essa medida e, posteriormente, efetuo-se a divisão do total de soluções não dominadas.

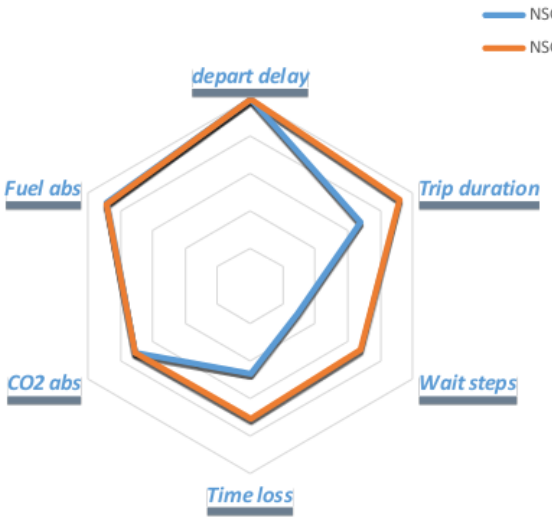
Tabela 6 – Experimento 01. Média das medidas

Medidas	**NSGA-II**					**NSGA-III**				
	Cenário 01			Cenário 02		Cenário 01			Cenário 02	
	Baixo	Médio	Alto	Baixo	Alto	Baixo	Médio	Alto	Baixo	Alto
depart delay	0.000305	3467.41	23130	0.00087	46286	0.000305	3482.7	23429	0.00087	46241
duration	68225.3	346016	706448	179731	2171638	92126	417830	880918	269466	2256486
wait steps	17602.9	142696.9	294097	51589.1	880583	40807.3	211474	463116	128920	1065175
time loss	46984.9	271637.9	557717	117205	1733454	70864	343404.9	732080	206939	1818322
CO_2 abs	3561.82	11814	23511.2	7829.68	52499	3529.04	11703.6	23267	7822.1	50989
fuel abs	1420044	4710144	9373561	3121578	20930589	1406974	4666072	9276201	3118553	20328895

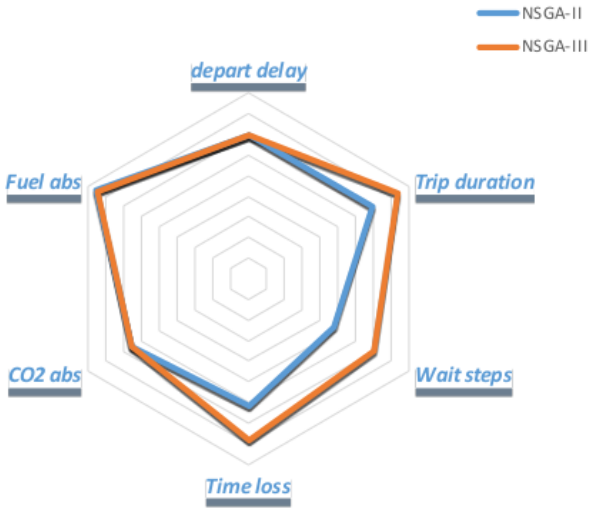
Algumas medidas tiveram resultados muito semelhantes para ambos algoritmos NSGA, mas para alguns o resultado foi muito diferente (com o NSGA-III derrotado) e isso foi decisivo

no cálculo do hipervolume. Para duas medidas, CO_2 *abs* e *fuel abs*, o valor médio foi semelhante para ambos NSGAs, nestes casos NSGA-III sempre teve os melhores valores para estas medidas. No entanto, quando o NSGA-II venceu, a diferença entre os valores médios quando comparados ao NSGA-III foi grande, isso ocorreu para duas medidas: *duration* e *wait steps*. Finalmente, o NSGA-II também ganhou (ou empatou), mas com uma pequena margem quando a medida comparada foi o *depart delay*, exceto para o último experimento, onde o NSGA-III tinha uma pequena vantagem.

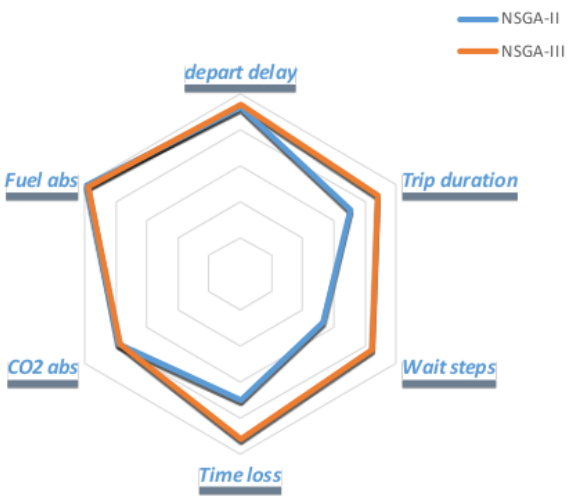
Figura 23 – Experimento 01. Cenário 01. Média das medidas



(a) Fluxo baixo



(b) Fluxo médio

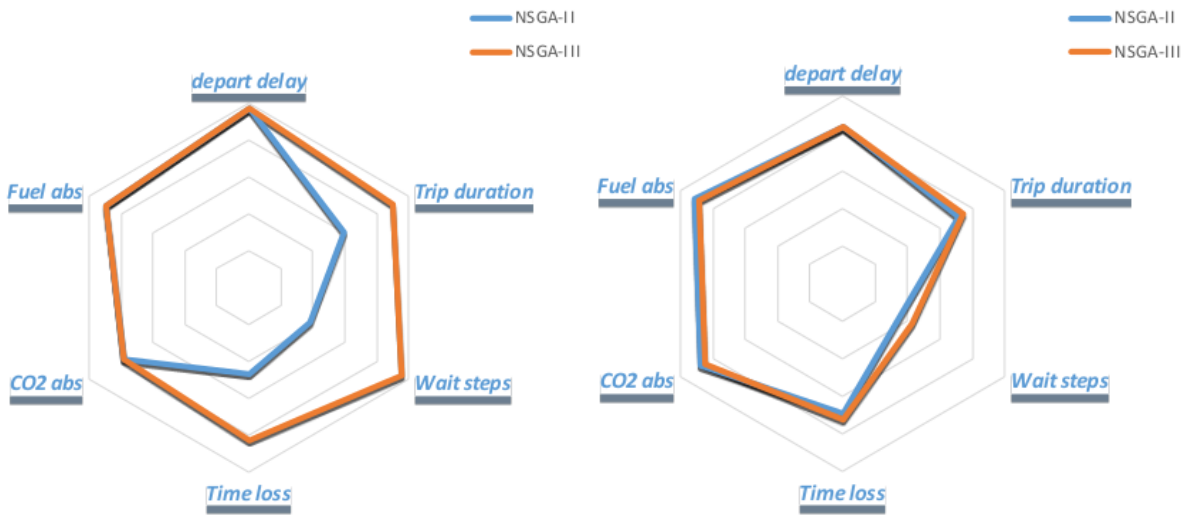


(c) Fluxo alto

Para cenários com fluxo baixo, o *depart delay* é próximo de zero, porque poucos veículos que tiveram esse atraso. Em relação aos valores mais baixos encontrados para as medidas em todas as viagens, vale ressaltar alguns cenários onde NSGA-III teve tais valores, são eles: Cenário 01 (fluxo baixo) obteve valores menores para todas as medidas, exceto *depart delay*. O Cenário 01 (fluxo médio) e o Cenário 02 (fluxo baixo) apresentaram menores valores de CO_2 *abs* e

fuel abs. Finalmente, o Cenário 02 (fluxo alto) apresentou menores valores para *depart delay*, CO_2 *abs* e *fuel abs*. As Figuras 23 e 24 fazem um comparativo entre as médias das medidas alcançadas pelos dois algoritmos, respectivamente, no Cenário 01 e Cenário 02. Para uma melhor vizualização dos gráficos, os valores encontrados na Tabela 6 foram normalizados.

Figura 24 – Experimento 01. Cenário 02. Média das medidas



(a) Fluxo baixo (b) Fluxo alto

5.4.2 Experimento 02

Aqui, utilizamos cinco das seis configurações de *benchmark* citadas na seção 5.1, são elas: Cenário 01 com fluxo baixo, médio e alto e Cenário 02 com fluxo baixo e médio. Como dito anteriormente, nesse experimento os dados foram normalizados e uma técnica de redução de dimensionalidade foi aplicada.

Sendo assim, para cada configuração de *benchmark* foram realizadas 40 execuções: 10 execuções do NSGA-II com redução de dimensionalidade e 10 execuções sem redução de dimensionalidade, 10 execuções do NSGA-III com redução de dimensionalidade e 10 execuções sem redução de dimensionalidade. Aqui, a letra r entre parênteses vai indicar a versão do algoritmo com redução de dimensionalidade.

5.4.2.1 Hipervolume e P-value

Observando a Tabela 7, cada linha representa um cenário e um fluxo de tráfego. Da segunda à quinta coluna são exibidos, para cada algoritmo, os valores da média e do desvio padrão (entre parênteses) do hipervolume. Neste experimento, os valores do hipervolume foram menores devido à normalização.

No Cenário 01, o NSGA-II com redução obteve os melhores resultados, porque os valores da média do hipervolume foram mais altos. Também é possível notar que tanto o NSGA-II

quanto o NSGA-III tiveram melhores resultados ao aplicar a redução de dimensionalidade. Em especial com o fluxo médio, onde o NSGA-III com redução teve uma melhora na perfomance bastante elevada quando comparada ao NSGA-III sem redução. No Cenário 02 com fluxo baixo, o NSGA-III com redução obteve o melhor resultado e o NSGA-III sem redução foi melhor que os dois NSGA-II.

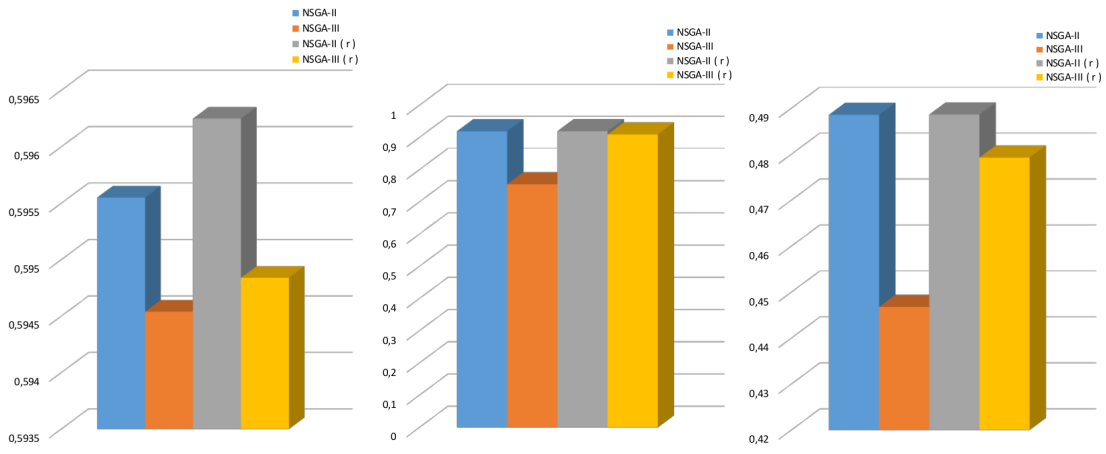
No Cenário 02 com o fluxo médio, o NSGA-III sem redução foi quem obteve o melhor resultado. Sendo essa a única configuração de *benchmark* em que um algoritmo sem redução de dimensionalidade obteve melhor resultado dentre todos. Sendo assim, ao contrário do Experimento 01 (onde NSGA-II sempre teve melhor performance), não houve um algoritmo que tivesse melhor perfomance em todos os cenários.

Tabela 7 – Experimento 02. Valores médios do hipervolume e desvio padrão (entre parênteses)

Cenário (Fluxo)	**NSGA-II**	**NSGA-III**	**NSGA-II (r)**	**NSGA-III (r)**
01 (baixo)	0.59555 (0.00187)	0.59454 (0.00065)	0.59625 (0.00062)	0.59484 (0.00120)
01 (médio)	0.91926 (0.00073)	0.75477 (0.10344)	0.91939 (0.00037)	0.90954 (0.00155)
01 (alto)	0.48857 (0.00024)	0.44678 (0.02645)	0.48861 (0.00013)	0.47926 (0.00698)
02 (baixo)	0.59458 (0.02160)	0.60131 (0.01677)	0.59479 (0.01653)	0.60418 (0.01628)
02 (médio)	0.94391 (0.02371)	0.95839 (0.03607)	0.94574 (0.03059)	0.95814 (0.02978)

A Figura 25 ilustra o valor médio do hipervolume encontrado em cada um dos quatro algoritmos para cada fluxo do Cenário 01, ou seja, da segunda à quinta coluna nas três primeiras linhas da Tabela 7 sem os valores do desvio padrão.

Figura 25 – Experimento 02. Cenário 01. Valores médios do hipervolume



(a) Fluxo baixo (b) Fluxo médio (c) Fluxo alto

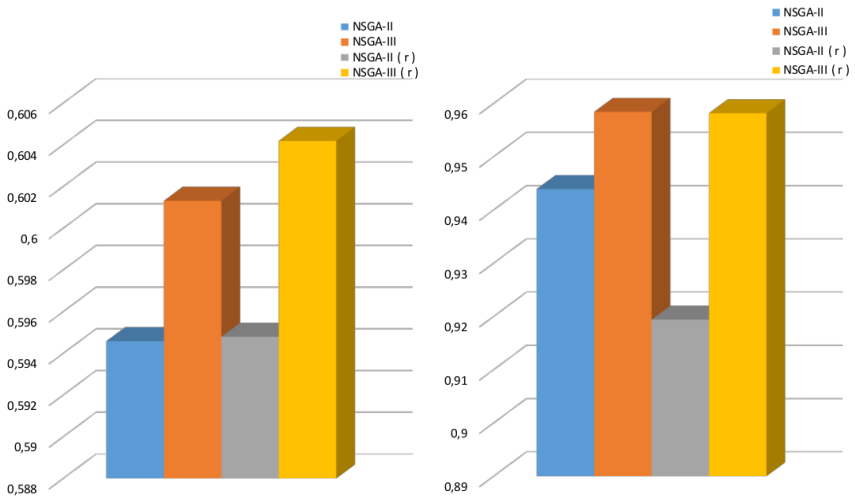
A Figura 26, por sua vez, ilustra o valor médio do hipervolume encontrado em cada um dos quatro algoritmos para o fluxo baixo e médio do Cenário 02, ou seja, da segunda à quinta coluna nas duas últimas linhas da Tabela 7 sem os valores do desvio padrão.

Em cada configuração de *benchmark* é feita uma comparação, através do $p - value$ obtido pelo teste de *Wilcoxon*. A comparação é feita para saber se há diferença estatística entre

a versão reduzida e a versão não reduzida de cada NSGA. Os resultados mostram que apenas há diferença estatística quando o algoritmo analisado é o NSGA-III, e somente no Cenário 01 com fluxo médio e com fluxo intenso, onde $p - value$ assume, respectivamente, $1.083x10^{-5}$ e 0.0001299.

Para os demais casos, onde não há diferença estatística, para cada configuração de *benchmark*, os valores de $p - value$ encontrados para NSGA-II e NSGA-III, foram: Cenário 01 com fluxo baixo, o NSGA-II obteve 0.2475 e o NSGA-III obteve 0.1051; Cenário 01 com fluxo médio, o NSGA-II obteve 0.7394; Cenário 01 com fluxo intenso, o NSGA-II obteve 0.6305; Cenário 02 com fluxo baixo, o NSGA-II obteve 0.7959 e o NSGA-III obteve 0.5787; Cenário 02 com fluxo médio, o NSGA-II obteve 0.7394 e o NSGA-III obteve 0.9705.

Figura 26 – Experimento 02. Cenário 02. Valores médios do hipervolume



(a) Fluxo baixo (b) Fluxo médio

Também é feita uma comparação, através do $p - value$, entre o melhor NSGA-II e melhor NSGA-III encontrados em cada configuração de *benchmark*. Os valores encontrados em cada configuração são ilustrados na Tabela 8. Nota-se que há diferença estatística para todos os fluxos do Cenário 01, pois $p - value < 0.05$. No entanto, para o Cenário 02, não há diferença estatística, pois $p - value > 0.05$ em todos os fluxos.

Tabela 8 – Experimento 02. Valores do p-value entre os melhores algoritmos

Cenário (Fluxo)	**Algoritmos**	**P-value**
01 (baixo)	NSGA-II (r) x NSGA-III (r)	0.002089
01 (médio)	NSGA-II (r) x NSGA-III (r)	$1.083x10^{-5}$
01 (alto)	NSGA-II (r) x NSGA-III (r)	$1.083x10^{-5}$
02 (baixo)	NSGA-II (r) x NSGA-III (r)	0.4813
02 (médio)	NSGA-II (r) x NSGA-III	0.5288

5.4.2.2 Média das Medidas

Tal como descrito na subseção 5.4.1.2, além do hipervolume e do teste de *Wilcoxon*, nesse experimento também é feito o cálculo dos valores médios de todas as funções objetivo, de todas as soluções não dominadas encontradas para ambos algoritmos. Aqui, as Tabelas 9 e 10 apresentam os valores encontrados, respectivamente, nos algoritmos sem aplicar a técnica de redução de dimensionalidade e ao aplicar a técnica de redução de dimensionalidade. Nesse experimento apenas foi realizada uma comparação entre os dois melhores algoritmos para cada configuração de *benchmark*. Sendo assim, uma comparação mais detalhada entre as médias das medidas, entre os quatro algoritmos, não foi realizada.

Tabela 9 – Experimento 02. Média das medidas sem redução de dimensionalidade

Medidas	**NSGA-II**					**NSGA-III**				
	Cenário 01			Cenário 02		Cenário 01			Cenário 02	
	Baixo	Médio	Alto	Baixo	Médio	Baixo	Médio	Alto	Baixo	Médio
depart delay	0.000305	3519	23126	0.000877	7298.7	0.000305	3691	23298	0.000877	7272
duration	70554.8	368016	709656.9	247836.6	1130188	69248	464813.9	849979.6	244018.7	1093999
wait steps	19837.6	164404	297362.7	100400	470833	18651	257930.8	433935	97220	442742
time loss	49319	293652	560984.9	185373.7	911293	48020.8	390379.6	701183	181553	874937.7
CO_2 abs	3549	11783.6	23495	8047	26884.8	3558.7	11666.6	23293.6	8050.8	26861
fuel abs	1415122	4697951	9367190	3208260	10718570.9	1418813.7	4651296.7	9286813.8	3209737.8	10709251

Tabela 10 – Experimento 02. Média das medidas com redução de dimensionalidade

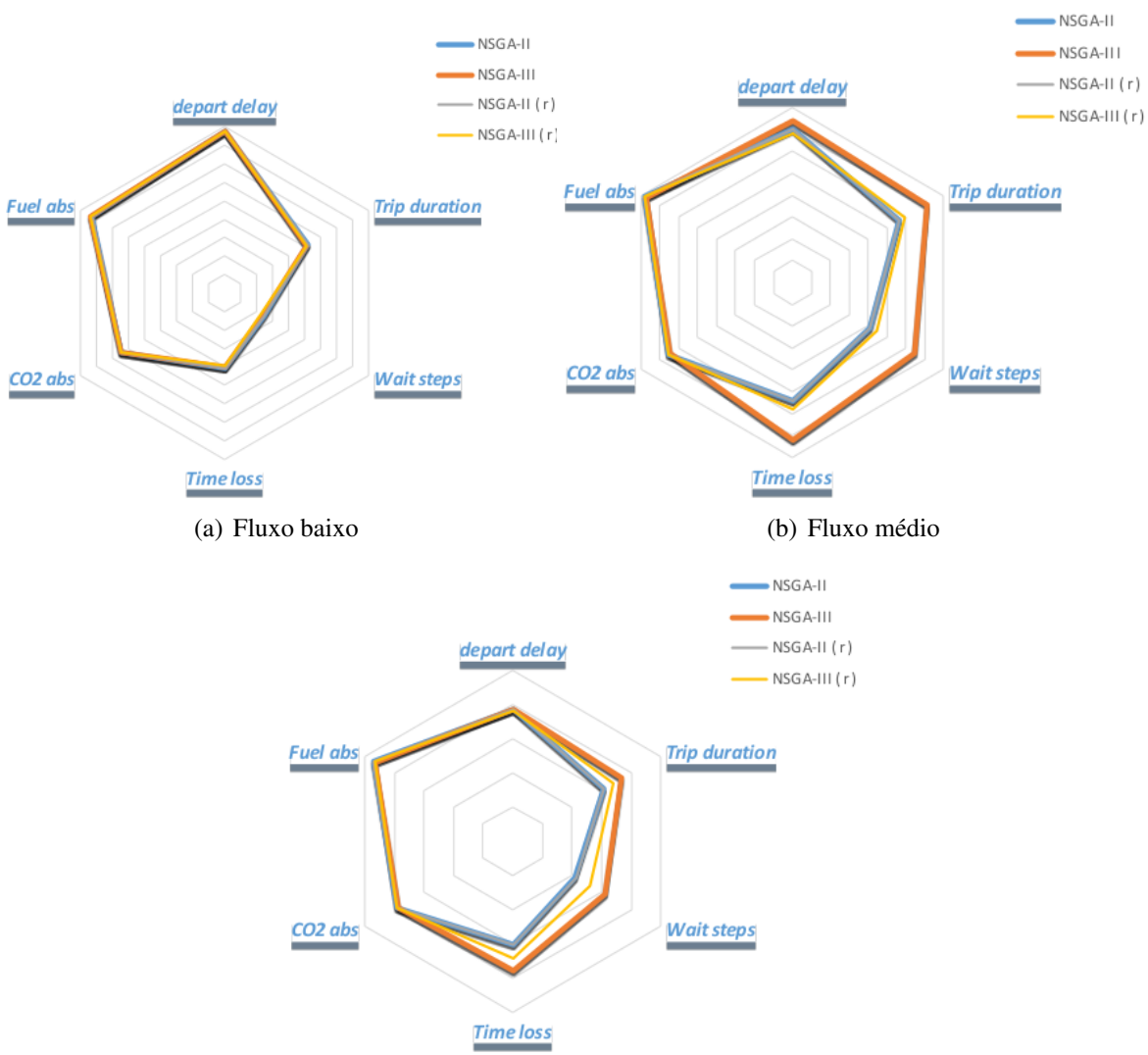
Medidas	**NSGA-II (r)**					**NSGA-III (r)**				
	Cenário 01			Cenário 02		Cenário 01			Cenário 02	
	Baixo	Médio	Alto	Baixo	Médio	Baixo	Médio	Alto	Baixo	Médio
depart delay	0.000305	3516	23119	0.000877	7272	0.000305	3411.6	22877	0.000877	7257.6
duration	70524.7	368003	711685	237079	1136189	68784	386059.9	783842	245340	1121139
wait steps	19796	164379	298772	88902.6	469352.8	18181	178345.7	364634	97915.8	464446.8
time loss	49289	293642	563011	174619.7	917262	47553.6	311685.7	635108.7	182876	902131.7
CO_2 abs	3548	11783	23497.8	8086	26933	3556	11771.7	23403	8053	26836
fuel abs	1414608.8	4697929	9368230	3223871	10737937	1417807.7	4693199	9330653	3210703	10699268.7

Algumas medidas tiveram resultados muito semelhantes para ambos algoritmos, como os encontrados no Cenário 01 e Cenário 02 com fluxo baixo. No Cenário 01, o NSGA-II obteve as menores médias e também obteve uma melhor perfomance. Já no Cenário 02, o NSGA-III obteve as menores médias e também obteve uma melhor perfomance.

Observando o Cenário 02 com fluxo médio, onde o NSGA-III também obteve uma melhor perfomance, quem obteve os menores valores das médias das medidas foi o NSGA-II. Isso evidencia que, ao menos nesse experimento, não há relação direta entre a performance de um algoritmo e este obter os menores valores para média das medidas.

Por fim, no Cenário 01 com fluxo alto (onde o NSGA-II obteve melhor perfomance) e no Cenário 02 com fluxo baixo (onde o NSGA-III obteve melhor perfomance), o NSGA-III foi quem obteve, na maioria das medidas, menores valores em média, perdendo apenas para as medidas CO_2 *abs* e $fuel$ *abs*. Para complementar as referidas tabelas, as Figuras 27 e 28 fazem um comparativo entre as médias das medidas alcançadas pelos quatro algoritmos, respectivamente, no Cenário 01 e Cenário 02.

Figura 27 – Experimento 02. Cenário 01. Média das medidas

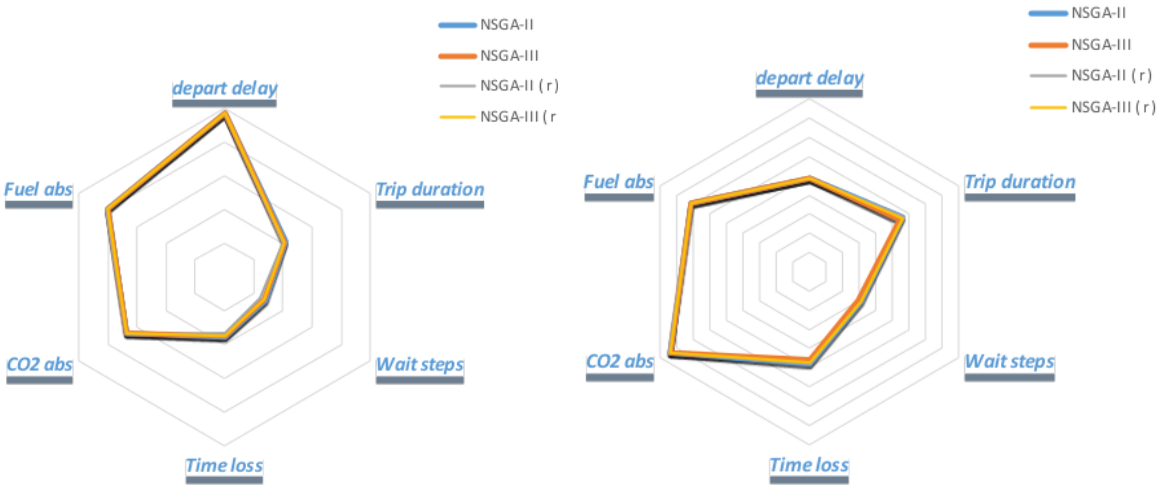


(a) Fluxo baixo

(b) Fluxo médio

(c) Fluxo alto

Figura 28 – Experimento 02. Cenário 02. Média das medidas



(a) Fluxo baixo

(b) Fluxo médio

6
Conclusão

A principal contribuição deste trabalho foi mostrar que é possível modelar e resolver o problema da sincronização de semáforos como um Problema de Otimização com Muito Objetivos. Para tal, neste estudo, o NSGA-II e o NSGA-III foram aplicados para resolver um problema de otimização com seis e quatro funções objetivo. Os objetivos foram obtidos através das viagens de cada veículo que participava de cada simulação, são eles: $Depart\ delay$, $Trip\ duration$, $Wait\ steps$ ($v < 0.1m/s$), $Time\ loss$ ($v < 60km/h$), $CO_2\ abs$ emitido e $fuel\ abs$ consumido. Quando foi aplicada a técnica de redução de dimensionalidade os objetivos $Time\ loss$ e $fuel\ abs$ foram omitidos.

Os métodos foram aplicados em uma rede com 1 cruzamento e com 4 cruzamentos em duas regiões hipotéticas. Para construir essas regiões, foi necessário definir o uso de um simulador de tráfego (o SUMO) para representar malhas viárias e obter as medidas de qualidade relacionadas ao tráfego após as simulações. A ideia foi utilizar o SUMO para construir um cenário relativamente próximo ao que poderia ser encontrado em uma região de uma cidade, ao menos no que tange a temporização dos semáforos, velocidade dos veículos e as características das malhas viárias.

Para tal, definiu-se configurações de *benchmark* que representassem três fluxos de veículos (baixo, médio e alto) em duas regiões hipotéticas. Porém, é importante frisar que o objetivo do *benchmark* não foi aproximar-se ao máximo de situações do mundo real, o objetivo foi testar a qualidade dos algoritmos. Então, foi desenvolvido um sistema que possibilitasse a comunicação entre os NSGAs e o SUMO. Esse sistema funcionou em conjunto com cinco módulos principais (MJM, MAB, MBD, MRD e MSI), onde diversas tarefas se comunicavam.

Em estudos na literatura, o NSGA-III supera NSGA-II em problemas com muitos objetivos, mas neste trabalho, os experimentos mostraram que NSGA-II superou NSGA-III para o problema em todas configurações de *benchmark* no Experimento 01, quando o vetor solução utilizou uma codificação inteira e os dados não foram normalizados. Porém, no Experimento

02, quando o vetor solução utilizou uma codificação binária e os dados foram normalizados, o NSGA-III obteve melhora na perfomance quando comparado ao NSGA-II em todas configuraçes de *benchmark*, superando o NSGA-II em duas configurações.

Ao analizar o Experimento 02, os resultados mostraram que a técnica de redução de dimensionalidade foi eficaz, pois houve uma melhora para os dois algoritmos em quase todas as configurações de *benchmark*. Apenas em uma única configuração, para o NSGA-III, não houve melhora. Também foi possível notar que no Cenário 02, onde há um número maior de variáveis, o NSGA-III superou o NSGA-II, algo que não ocorreu no Experimento 01.

Ainda no Experimento 02, observou-se que utilizar uma base de dados no Cenário 01 foi viável por conta da pequena quantidade de variáveis. A partir dessa constatação, também foi possível notar que para o Cenário 01, mais experimentos, com diferentes configurações de *benchmark*, poderiam ser realizados em um intervalo de tempo bastante reduzido.

6.1 Limitações do Trabalho

Apesar dos bons resultados alcançados por esse trabalho, ao mostrar que é possível modelar e resolver o problema da sincronização de semáforos como um MAOP, e ao mostrar a eficácia ao aplicar uma técnica de redução de dimensionalidade. Este estudo encontrou algumas limitações. A primeira limitação é referente a distância que o *benchmark* se encontra do que é encontrado no mundo real. Isso ocorre devido ao fato de um configuração de *benchmark* representar apenas uma situação muito específica do trânsito, em um dado momento, em uma dada região. Sendo que em uma cidade, o fluxo de veículos pode variar bastante a depender do horário e do dia da semana. Sendo assim, no que tange ao fluxo de veículos, para este trabalho se aproximar do mundo real, seria necessário que no mínimo, para cada cruzamento existisse mais do que três fluxos de veículos. Além do que, para cada fluxo, fossem definidas diferentes demandas de véiculos por cada injetor. Outro ponto importante é que, em uma cidade, os diversos cruzamentos existentes sofrem, e causam, influência a diversos outros cruzamentos e, neste trabalho, os cenários possuem apenas 1 e 4 cruzamentos.

Outra limitação é que, dado que esperava-se que o NSGA-III não fosse derrotado pelo NSGA-II, algumas hipóteses (listadas no capítulo anterior) foram levantadas, porém não foram completamente respondidas. Restando descobrir o que fez com que o NSGA-III melhorasse após a integração da normalização e da codificação binária no Experimento 02. Se foi apenas a normalização, apenas a codificação binária, ou ambas.

Outra limitação é que apesar da melhora, em quase todas as configurações de *benchmark*, do NSGA-II e NSGA-III ao aplicar a técnica de redução de dimensionalidade, não há como definir qual o melhor algoritmo dentre os dois experimentos. Isso ocorre porque a métrica utilizada foi a média do hipervolume, calculada com dados em escalas muito diferentes nos dois experimentos. Para que fosse possível fazer a comparação, seria necessário rodar ao menos

o Experimento 01 mais uma vez, utilizando como parâmetro para simulação, os valores das variáveis de todas as soluções não dominadas desse experimento. Porém, dessa vez os dados seriam normalizados, e então as médias dos hipervolumes do Experimento 01 seriam recalculadas e comparadas ao Experimento 02.

Por fim, uma limitação importante se refere a como evitar que os algoritmos de otimização gerem soluções muito ruins, especialmente no início das execuções. Notou-se que duas situações levam os algoritmos a gerarem soluções muito ruins: os valores gerados aleatoriamente pelo algoritmo e o fato de existirem certos valores que nunca deveriam ser atribuídos as variáveis. Por exemplo, se o melhor valor para o tempo de uma fase está em um intervalo entre 25 e 40 segundos, e para outro está entre 90 e 100 segundos, o algoritmo pode levar muito tempo para atribuir esses valores a essas duas fases, e isto piora a medida que o número de fases aumenta.

6.2 Trabalhos Futuros

Para os trabalhos futuros serão realizados novos experimentos buscando entender o que foi responsável pela melhora do NSGA-III após a integração da normalização e a codificação binária, se a normalização ou a codificação, ou ambas. Também serão realizados experimentos aplicando técnicas de redução de dimensionalidade *online* e técnicas de paralelização. Nos novos experimentos serão utilizados objetivos menos correlacionados, além de unir os objetivos correlacionados em um único objetivo, como certas emissões (e.g. CO e NO_x). Também serão testadas novas estratégias de como atacar o problema da sincronização de semáforos, além de buscar parcerias com as entidades responsáveis pela gerência do trânsito, com intuito de realizar estudos de caso reais.

Outra estratégia, a ser utilizada em um trabalho futuro, consiste em dividir um cenário em cenários menores, atacando localmente cada cruzamento, o que implicaria em reduzir o número de variáveis e consequente o tempo de execução dos experimentos, além de que nesses casos, têm-se a possibilidade real de integrar uma base de dados, pois existiriam poucas variáveis. A ideia por trás dessa divisão será analisar se os resultados encontrados ao realizar diversas otimizações locais (através da divisão) são similares aos resultados encontrados através de uma otimização global.

Será construido um módulo para gerar soluções viáveis para cada configuração de *benchmark*, onde estas servirão como parâmetro de corte inicial para cada solução gerada pelo algoritmo de otimização. Assim, a partir do procedimento de corte desse módulo, será possível tornar esta abordagem mais suscetível em criar soluções melhores (viáveis). Visto que, normalmente o algoritmo de otimização depende de "sorte" para encontrar uma solução razoável (mesmo longe do ideal), especialmente nas primeiras simulações. Este módulo teria efeito direto não apenas nos valores das medidas, mas também no tempo de simulação.

Por fim, outra estratégia é reduzir o número total de veículos, focando em situações de

tráfego com duração menor, entretanto, mantendo as características dos cruzamentos idênticas as utilizadas neste trabalho. Assim, será possível reduzir drasticamente o tempo de simulação, permitindo um maior número de execuções. Essa estratégia também permite que se analise mais configurações de *benchmark*.

Referências

AGUIRRE, H.; TANAKA, K. Many-objective optimization by space partitioning and adaptive ε-ranking on mnk-landscapes. In: *Proceedings of the 5th International Conference on Evolutionary Multi-Criterion Optimization*. Berlin, Heidelberg: Springer-Verlag, 2009. (EMO '09), p. 407–422. ISBN 978-3-642-01019-4. Disponível em: <http://dx.doi.org/10.1007/978-3-642-01020-0_33>. Citado na página 36.

AN, S.; LEE, B.; SHIN, D. A survey of intelligent transportation systems. In: . [S.l.]: Computational Intelligence, Communication Systems and Networks (CICSyN), 2011. p. 332–337. Citado na página 17.

AUGERI, M. G.; COZZO, P.; GRECO, S. Dominance-based rough set approach: An application case study for setting speed limits for vehicles in speed controlled zones. *Knowledge-Based Systems*, v. 89, p. 288–300, 2015. ISSN 0950-7051. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0950705115002622>. Citado na página 91.

BRITTO, A.; MOSTAGHIM, S.; POZO, A. Iterated multi-swarm: A multi-swarm algorithm based on archiving methods. In: . [S.l.]: GECCO 13, 2013. p. 6–10. Citado na página 38.

BRITTO, A.; POZO, A. Using archiving methods to control convergence and diversity for many-objective problems in particle swarm optimization. In: . [S.l.]: WCCI 2012 IEEE World Congress on Computational Intelligence, 2012. p. 10–15. Citado na página 37.

BROCKHOFF, D.; ZITZLER, E. Are all objectives necessary? on dimensionality reduction in evolutionary multiobjective optimization. In: *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature*. Berlin, Heidelberg: Springer-Verlag, 2006. (PPSN'06), p. 533–542. ISBN 3-540-38990-3, 978-3-540-38990-3. Disponível em: <http://dx.doi.org/10.1007/11844297_54>. Citado 4 vezes nas páginas 37, 42, 43 e 44.

BROCKHOFF, D.; ZITZLER, E. Dimensionality reduction in multiobjective optimization: The minimum objective subset problem. p. 423–429, 2006. Citado na página 44.

BROCKHOFF, D.; ZITZLER, E. Objective reduction in evolutionary multiobjective optimization: Theory and applications. *Evolutionary Computation*, MIT Press, Cambridge, MA, USA, v. 17, n. 2, p. 135–166, jun. 2009. ISSN 1063-6560. Disponível em: <http://dx.doi.org/10.1162/evco.2009.17.2.135>. Citado 3 vezes nas páginas 43, 45 e 57.

C. CORPORATION. TRANSMODELER. *Transmodeler*. [S.l.], 2009. <http://www.caliper.com/transmodeler/default.htm>. Citado na página 26.

CAČON-LOZANO, Y. et al. Web service platform for automatic generation of o/d matrix for mass transportation systems. In: . [S.l.]: IEEE, 2013. Citado na página 18.

CARVALHO, A. B. Novas estratégias para otimização por nuvem de partículas aplicadas a problemas com muitos objetivos. In: *Tese de doutorado apresentada como requisito à obtenção do grau de Doutor, Universidade Federal do Paraná*. [S.l.: s.n.], 2013. Citado 3 vezes nas páginas 32, 33 e 36.

COELLO, C. A. C.; LAMONT, G. B.; VELDHUIZEN, D. A. V. *Evolutionary Algorithms for Solving Multi-Objective Problems*. [S.l.: s.n.], 2006. Citado na página 32.

DAS, I.; DENNIS, J. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal of Optimization*, v. 8, n. 3, p. 631–657, 1998. Citado na página 39.

DEB, K.; JAIN, H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. In: . [S.l.]: IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, 2014. v. 18, n. 4. Citado 5 vezes nas páginas 36, 37, 38, 57 e 68.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, Evolutionary Computation, IEEE Transactions on, v. 6, n. 2, p. 182–197, August 2002. Citado na página 33.

DEB, K.; SAXENA, D. K. Searching for pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. *IEEE Congress on Evolutionary Computation (CEC)*, p. 3353–3360, 2006. Citado 3 vezes nas páginas 37, 42 e 44.

DEB, K.; SAXENA, D. K. Non-linear dimensionality reduction procedures for certain large-dimensional multi-objective optimization problems: Employing correntropy and a novel maximum variance unfolding. In: *Proceedings of the 4th International Conference on Evolutionary Multi-criterion Optimization*. Berlin, Heidelberg: Springer-Verlag, 2007. (EMO'07), p. 772–787. ISBN 978-3-540-70927-5. Disponível em: <http://dl.acm.org/citation.cfm?id=1762545.1762611>. Citado 2 vezes nas páginas 42 e 44.

DEB, K.; SUNDAR, J. Reference point based multi-objective optimization using evolutionary algorithms. In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2006. (GECCO '06), p. 635–642. ISBN 1-59593-186-4. Disponível em: <http://doi.acm.org/10.1145/1143997.1144112>. Citado na página 37.

DIAKAKI, C.; PAPAGEORGIU, M.; Y., A. K. A multivariable regulator approach to traffic-responsive network-wide signal control. *Control Engineering Practice*, v. 10, n. 2, p. 183 – 195, 2002. ISSN 0967-0661. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0967066101001216>. Citado na página 22.

D.L.R. *SUMO User Documentation*. [S.l.], 2017. <http://sumo.dlr.de/wiki/SUMO_User_Documentation>. Citado na página 27.

ELVIK, R. *The Handbook of Road Safety Measures*. [S.l.]: Emerald, 2009. ISBN 9781848552500. Citado na página 70.

EZELL, S. Intelligent transportation systems. In: . [S.l.]: IEEE, 2010. Citado 2 vezes nas páginas 94 e 95.

FREY, H. C.; UNAL, A.; COLYAR, J. D. On-road measurement of vehicle tailpipe emissions using a portable instrument. In: *Journal of the Air & Waste Management Association*. [S.l.: s.n.], 2003. v. 53, p. 992–1002. Citado na página 70.

GARCÍA-NIETO, J.; OLIVEIRA, A. C.; ALBA, E. Optimal cycle program of traffic lights with particle swarm optimization. In: . [S.l.]: IEEE, 2013. p. 823–839. Citado 3 vezes nas páginas 14, 28 e 29.

GIFFINGER, R. et al. Smart cities: Ranking of european medium-sized cities. In: . [S.l.: s.n.], 2007. Citado na página 13.

HAJBABAIE, A. Intelligent dynamic signal timing optimization program. In: . [S.l.: s.n.], 2012. <https://www.ideals.illinois.edu/handle/2142/32023>. Citado na página 23.

HAJBABAIE, A.; BENEKOHAL, R. F. A program for simultaneous network signal timing optimization and traffic assignment. In: . [S.l.]: IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, 2015. v. 16, p. 2573–2586. Citado 2 vezes nas páginas 23 e 47.

HANCKE, G. P.; SILVA, B. C.; JR, G. P. H. The role of advanced sensing in smart cities. In: . [S.l.: s.n.], 2013. Citado na página 18.

HERRERA-QUINTERO, L. F. et al. Smart cities approach for colombian context. learning from its experiences and linking with government organization. In: . [S.l.]: IEEE, 2015. Citado na página 17.

HUGHES, E. J. Msops-ii: A general-purpose many-objective optimiser. In: *2007 IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 2007. p. 3944–3951. ISSN 1089-778X. Citado na página 37.

ISHIBUCHI, H. et al. Behavior of emo algorithms on many-objective optimization problems with correlated objectives. In: . [S.l.]: Evolutionary Computation (CEC), 2011 IEEE Congress on, 2011. p. 1465 –1472. ISSN Pending. Citado na página 36.

ISHIBUCHI, H.; TSUKAMOTO, N.; NOJIMA, Y. Evolutionary many-objective optimization: A short review. In: . [S.l.]: IEEE Congress on Evolutionary Computation (CEC), 2008. Citado 3 vezes nas páginas 14, 31 e 35.

JAIMES, A. L.; COELLO, C. A. Some techniques to deal with many-objective problems. In: *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*. New York, NY, USA: ACM, 2009. (GECCO '09), p. 2693–2696. ISBN 978-1-60558-505-5. Disponível em: <http://doi.acm.org/10.1145/1570256.1570386>. Citado na página 45.

JAIMES, A. L.; COELLO, C. A.; BARRIENTOS, J. E. Online objective reduction to deal with many-objective problems. *EMO 2009*, p. 423–437, 2009. Citado na página 40.

JAIMES, A. L.; COELLO, C. A.; CHAKRABORTY, D. Objective reduction using a feature selection technique. In: *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2008. (GECCO '08), p. 673–680. ISBN 978-1-60558-130-9. Disponível em: <http://doi.acm.org/10.1145/1389095.1389228>. Citado 2 vezes nas páginas 42 e 45.

JUAN, J. D.; ANTONIO, J. N. jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, v. 42, n. 10, p. 760–771, 2011. ISSN 0965-9978. Citado na página 15.

KRAJZEWICZ, D.; BONERT, M.; WAGNER, P. The open source traffic simulation package sumo. In: . [S.l.]: RoboCup 2006 Infrastructure Simulation Competition, German Aerospace Centre, Institute of Transportation Research, 2006. Citado 2 vezes nas páginas 28 e 29.

KWASNICKA, H.; STANEK, M. Genetic approach to optimize traffic flow by timing plan manipulation. In: . [S.l.]: Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06), 2006. Citado 2 vezes nas páginas 14 e 46.

LEITE, J. G. M. Engenharia de tráfego: métodos de pesquisa, características de tráfego, interseções e sinais luminosos. In: SÃO PAULO: COMPANHIA DE ENGENHARIA DE TRÁFEGO - CET. [S.l.], 1980. Citado na página 20.

LEUNG, J. Y.-T. Handbook of scheduling: Algorithms, models, and performance analysis. In: . [S.l.: s.n.], 2004. Citado na página 29.

LOWRIE, P. R. The sydney coordinate adaptive traffic system - principles, methodology, algorithms. In: . [S.l.]: International Conference on Road Traffic Signalling, 1982. p. 67–70. ISBN 0852962592. Citado na página 26.

MACHADO, R. J. F. Adaptação de mapas geográficos e planeamento de rotas na simulação de tráfego de veículos em ambiente citadino. In: . [S.l.]: Tese submetira para Mestrado Integrado em Engenharia Electrotécnica de Computadores, Faculdade de Ciências e Tecnologia, Portugal, Coimbra., 2009. Citado na página 25.

MATSUEDA, L. C. O. Análise e otimização do problema de roteamento de veículos com muitos objetivos e janelas de tempo flexíveis. In: *Dissertação submetida ao Instituto de Ciências Exatas e Biológicas da Universidade Federal de Ouro Preto para a obtenção do título de Mestre em Ciências da Computação*. [S.l.: s.n.], 2015. Citado na página 45.

MCTRANS CENTER. *Corsim*. [S.l.], 2009. <http://mctrans.ce.ufl.edu>. Citado na página 26.

MONIREH, A.; NASSER, M.; ANA, L. C. B. Traffic light control in non-stationary environments based on multi agent q-learning. In: . [S.l.]: 14th International IEEE Conference on Intelligent Transportation, Systems, 2011. p. 5–7. Citado na página 47.

MYSQL. *MySQL Documentation*. [S.l.], 2017. <https://dev.mysql.com/doc>. Citado na página 15.

OLIVEIRA, D. Um estudo de coordenação dinâmica de agentes aplicado ao gerenciamento de tráfego veicular urbano. In: *Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciências da Computação, UFRS*. [S.l.: s.n.], 2005. Citado 2 vezes nas páginas 20 e 25.

OLIVEIRA, D.; BAZZAN, A. L. C. Traffic lights control with adaptive group formation based on swarm intelligence. In: *Lecture Notes in Computer Science*. [S.l.]: Ant Colony Optimization and Swarm Intelligence (ANTS), 2006. v. 4150. Citado na página 22.

OPENSTREETMAP FOUNDATION. *Open Street Map*. [S.l.], 2017. <https://www.openstreetmap.org>. Citado na página 27.

PTV GROUP. *VISSIM*. [S.l.], 2009. <http://www.ptvag.com>. Citado na página 26.

PURSHOUSE, R. C.; JALBa, C.; FLEMING, P. J. Preference-driven co-evolutionary algorithms show promise for many-objective optimisation. In: *Proceedings of the 6th International Conference on Evolutionary Multi-criterion Optimization*. Berlin, Heidelberg: Springer-Verlag, 2011. (EMO'11), p. 136–150. ISBN 978-3-642-19892-2. Disponível em: <http://dl.acm.org/citation.cfm?id=1987637.1987648>. Citado na página 37.

Q. PARAMICS. *Paramics*. [S.l.], 2009. <http://www.paramics-online.com/product_estimator.php>. Citado na página 26.

Q., Z.; LI, H. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, v. 11, n. 6, p. 712–731, December 2007. ISSN 1089-778X. Citado na página 36.

ROBERTSON, D. I.; BRETHERTON, R. D. Optimizing networks of traffic signals in real time - the scoot method. In: . [S.l.]: IEEE Transactions on Vehicular Technology, 1991. v. 40, n. 1, p. 11–15. Citado 2 vezes nas páginas 20 e 26.

SÁNCHEZ-MEDINA, J. J.; GALÁN-MORENO, M. J.; RUBIO-ROYO, E. Traffic signal optimization in "la almozara" district in saragossa under congestion conditions, using genetic algorithms, traffic microsimulation, and cluster computing. In: . [S.l.]: IEEE, 2010. v. 11, n. 1. Citado na página 47.

SATO, H.; AGUIRRE, H. E.; TANAKA, K. Controlling dominance area of solutions and its impact on the performance of moeas. In: *Proceedings of the 4th International Conference on Evolutionary Multi-criterion Optimization*. Berlin, Heidelberg: Springer-Verlag, 2007. (EMO'07), p. 5–20. ISBN 978-3-540-70927-5. Disponível em: <http://dl.acm.org/citation.cfm?id=1762545.1762552>. Citado na página 36.

SAXENA, D. K. et al. Objective reduction in many-obectice optimization: Linear and nonlinear algorithms. *IEEE Transactions on Evolutionary Computation*, v. 17, n. 1, p. 77–99, February 2013. Citado 3 vezes nas páginas 41, 42 e 44.

SEREDYNSKI, M.; MAZURCZYK, W.; KHADRAOUI, D. Multi-segment green light optimal speed advisory. In: . [S.l.]: Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International, 2013. p. 459–465. Citado 2 vezes nas páginas 13 e 17.

SHEN, Z.; WANG, K.; Y., W. F. Gpu based non-dominated sorting genetic algorithm-ii for multi-objective traffic light signaling optimization with agent based modeling. In: . [S.l.]: Proceedings of the 16th International IEEE Annual Conference on Intelligent Transportation Systems (ITSC 2013), 2013. Citado 2 vezes nas páginas 14 e 46.

SINGH, H. K.; ISAACS, A.; RAY, T. A pareto corner search evolutionary algorithm and dimensionality reduction in many-objective optimization problems. *Trans. Evol. Comp*, IEEE Press, Piscataway, NJ, USA, v. 15, n. 4, p. 539–556, August 2011. ISSN 1089-778X. Disponível em: <http://dx.doi.org/10.1109/TEVC.2010.2093579>. Citado 3 vezes nas páginas 37, 42 e 45.

TAKAHASHI, R. H. C. Otimização escalar e vetorial. In: . [S.l.: s.n.], 2007. v. 1. Citado na página 31.

TRAFFICWARE. *Sim traffic*. [S.l.], 2009. <http://www.trafficware.com>. Citado na página 26.

UNITED STATES DEPARTMENT OF TRANSPORTATION. *Connected Vehicle Technology*. [S.l.], 2015. <http://its.dot.gov>. Citado 3 vezes nas páginas 91, 92 e 93.

WÜNSCH; GREGOR. Coordination of traffic signals in networks and related graph theoretical problems on spanning trees. In: . [S.l.: s.n.], 2008. Citado na página 23.

ZITZLER, E.; KÜNZLI, S. Indicator-based selection in multiobjective search. In: ____. *Parallel Problem Solving from Nature - PPSN VIII: 8th International Conference, Birmingham, UK, September 18-22, 2004. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 832–842. ISBN 978-3-540-30217-9. Disponível em: <https://doi.org/10.1007/978-3-540-30217-9_84>. Citado na página 37.

ZITZLER, E. et al. Performance assessment of multiobjective optimizers: An analysis and review. *Trans. Evol. Comp*, IEEE Press, Piscataway, NJ, USA, v. 7, n. 2, p. 117–132, April 2003. ISSN 1089-778X. Disponível em: <http://dx.doi.org/10.1109/TEVC.2003.810758>. Citado na página 69.

Apêndices

APÊNDICE A – Soluções em ITS

As tecnologias em ITS englobam diferentes espaços de aplicação e estão abrindo o caminho em direção a melhores redes de transporte que beneficiam a todos. As soluções ITS melhoram não só a mobilidade das cidades, mas também diversas áreas associadas. Por conta disso, muitos são os problemas atacados no âmbito do ITS em diversas áreas e subáreas. Tudo que envolva qualquer tipo de transporte, e os agentes envolvidos, são problemas de ITS. Desde pedestres, ciclistas, estradas, veículos, câmeras, semáforos, sistemas de ônibus, de metrôs, de aeroportos, ferroviário, fluvial, etc.

Algumas das soluções demandam um alto custo financeiro e tecnológico, outras não, e apesar de mais simples, tem grande impacto no trânsito. O que diferencia esses dois tipos de soluções é que no primeiro grupo diversos dispositivos inteligentes podem fazer parte da solução. Já no segundo grupo, normalmente nenhum, ou apenas um, sistema de computador é utilizado. E, por vezes, apenas uma programação prévia é utilizada. Dentre as soluções que não demandam um alto custo financeiro e tecnológico, podemos citar: o rodízio de carros, análise (humana) de tráfego através de imagens enviadas por câmeras instaladas em pontos estratégicos, sistemas que otimizam as regras e restrições de trânsito (regras de: estacionamento e desembarque e localização da faixa de pedestre), etc.

A.0.1 Velocidades Econômicas e Seguras

Sistemas podem definir a velocidade máxima visando maior segurança ou gerar menos poluição e economia de combustível. Em (AUGERI; COZZO; GRECO, 2015), um sistema de apoio à decisão (DSS, do inglês *Decision Support System*) sugere um limite de velocidade seguro a partir de certas características das vias. Como consequência o aumento da segurança dos condutores, devido a redução do risco de acidentes e a redução das emissões sonoras e de poluentes. As eco faixas dinâmicas (do inglês *Dynamic eco-lanes*) são vias onde o motorista pode optar por utilizar uma das faixas em uma velocidade diferente das demais faixas da via. Visando gastar menos combustível e consequentemente poluir menos, esta faixa funciona de forma similar a uma faixa exclusiva de ônibus, pois se trata de uma faixa exclusiva para os motoristas que optem por trafegar a uma velocidade específica, dita como mais econômica e menos poluente (UNITED STATES DEPARTMENT OF TRANSPORTATION, 2015).

A.0.2 Auxílio aos Pedestres e Ciclistas

Os usuários (pedestres, ciclistas e motoristas) interagem através de dispositivos inteligentes ou páginas na internet por meio de redes móveis (públicas ou privadas). Tendo acesso a diversas informações, tais como: características do trânsito na rota desejada, melhor rota a seguir,

e horário, distância e sugestões a respeito dos transportes público disponíveis, dentre outras. O pedestre também pode alimentar os sistemas, por exemplo deixando o seu *bluetooth* ligado ao chegar no (s) ponto (s) de ônibus, auxiliando atualizações a respeito do sistema de transporte público.

Os sistemas também fornecem auxílio a pessoas com dificuldade para se locomover (um cadeirante ou idoso), dando a possibilidade dessa pessoa informar (enviar sinal) para o semáforo que deseja atravessar a rua. E este altera o seu estado para que a pessoa possa atravessar com segurança. Esse mesmo semáforo pode enviar um alerta de volta a pessoa, como também enviar ao veículo que esteja se aproximando (UNITED STATES DEPARTMENT OF TRANSPORTATION, 2015). Visando favorecer os ciclistas, podem ser efetuadas, em tempo real, alterações semafóricas ou de sentido de direção. Um exemplo são detectores de bicicletas em alguns cruzamentos. Esse detectores ficam localizados na via e possuem com símbolos para auxiliar o ciclista onde para e aguardar com sua bicicleta, facilitando a detecção. Em poucos instantes, será detectada a presença do (s) ciclista (s) e o semáforo irá alterar seu estado liberando a passagem.

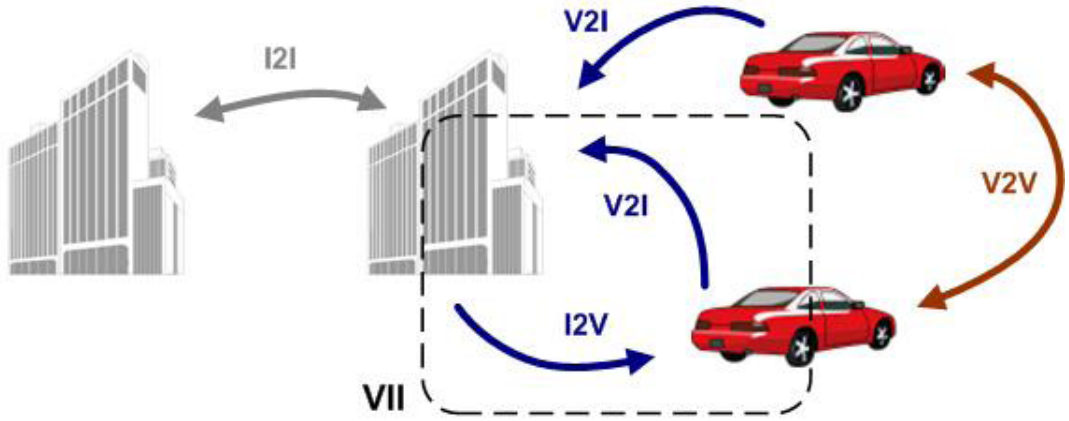
A.0.3 Redes de Veículos e Sistemas de Comunicação

Redes de Veículos e Sistemas de Comunicação são áreas de importância significativa em um mundo cada vez mais conectado e móvel. Técnicas eficazes de conectividade veicular podem melhorar significativamente a eficiência de viagens, reduzir os incidentes de trânsito, melhorar a segurança, reduzir o impacto do congestionamento e fornecer uma experiência mais confortável. Dentre os sistemas de comunicação (ilustrados na Figura 29) podemos citar a comunicação entre veículos (V2V, do inglês *Vehicle-to-Vehicle*), comunicação entre veículo e infraestrutura (V2I, do inglês *Vehicle-to-Infrastructure*), comunicação entre infraestruturas (I2I, do inglês *Infrastructure-to-Infrastructure*) e a comunicação entre veículos, pessoas e "outros" (V2X, do inglês, *Vehicle-to-X*), como por exemplo a comunicação de veículos com passageiros, com ciclistas e pedestres.

A.0.4 Auxílio os Motoristas

No quis diz respeito a comunicação V2I, as informações oriundas de um ou mais veículos podem ser enviadas para os Centros de Gerenciamento de Tráfego (TMCs, do inglês *Traffic Management Centers*), que por sua vez encaminham os dados para um Sistema de Gerenciamento de Transporte (TMS, do inglês *Transportation Management System*) (UNITED STATES DEPARTMENT OF TRANSPORTATION, 2015). Com isso podem tomar diferentes medidas (por exemplo: alterar a velocidade máxima, alterar o tempo ou estado dos semáforos ou enviar veículos para suporte de manutenção das pistas), bem como podem divulgar informações de diversas naturezas, seja exibindo-as em painéis de informações ou divulgando-as na internet.

Figura 29 – Caminhos da comunicação em ITS cooperativos



Fonte: Adaptado de Nijssen, B. J. W. (2008)

As infraestruturas podem se comunicar constantemente com os veículos para mantê-los informados, como por exemplo informar ao motorista: em qual velocidade fixar-se para conseguir uma "onda verde" (programação prévia de semáforos ao longa de uma via que proporciona a passagem dos veículos por vários cruzamentos sem a necessidade de parar) e onde há vaga (s) para estacionar. Veículos conectados entre si (V2V) provem mais segurança e agilidade através de avisos. Dentre os diversos avisos existentes, podemos citar cinco encontrados em (UNITED STATES DEPARTMENT OF TRANSPORTATION, 2015):

- Aviso de mudança de faixa (LCW, do inglês *lane change warning*) fornece um aviso se o condutor tem a intenção de mudar de faixa e a faixa estará em breve ocupada, por exemplo por um carro que venha em alta velocidade;
- Aviso de ponto cego (BSW, do inglês *blind spot warning*) permite que o motorista saiba que há um veículo em um ponto não visível. Avisando que não é seguro trocar de faixa;
- Aviso de curva à esquerda não segura (LTA, do inglês *left turn assist*) alerta ao condutor que se optar por fazer uma curva à esquerda, poderá não ser seguro, devido ao tráfego no local;
- Aviso de colisão à frente (FCW, do inglês *forward collision warning*) alerta ao condutor de uma potencial colisão traseira com um veículo parado ou que esteja em velocidade mais lenta;
- Aviso de algum problema na condição da pista. Por exemplo, se um ou mais carros já passaram por uma via que esteja danificada ou muito alagada. Essa informação é repassada (V2V) para os carros que ainda não chegaram ao trecho da pista em questão.

APÊNDICE B – Tecnologias Utilizadas

Para resolver os problemas apresentados na seção anterior, diversas tecnologias são utilizadas. Dentre elas podemos citar a comunicação sem fio, principalmente para prover comunicação entre veículos (V2V), entre infraestruturas (I2I) e entre ambos (V2I). As tecnologias de processamento visual, aplicada no monitoramento de tráfego, e na detecção classificação e rastreamento de veículos. E os simuladores, que permitem predizer como diversos pontos do tráfego em uma cidade se comportariam. Esses resultados gerados podem ser úteis de diversas formas, tais como: redução de congestionamento, redução da poluição, previsão e o impacto que as novas redes viárias terão no tráfegos, etc.

Para efetuar a coleta, transporte, iteração e tratamento dos dados, são utilizadas diversas técnicas e diversos aparelhos de diversas marcas e que podem possuir diferentes protocolos. Além do que para cada cenário a rede pode ser *outdoor* ou *indoor*, e a comunicação pode se dar por *WiFi*, *Bluetooth*, *ZigBee*, *Ant* +, *Dash* 7, etc. Por isso é muito importante a integração de sistemas e dados e a interoperabilidade. A coleta de dados pode ser feita com o uso de radares, câmeras, semáforos, sensores (nas pistas e nos veículos), dispositivos móveis, dentre outros. Os dados depois de coletados e tratados, geram estatísticas e relatórios que podem servir como parâmetros para tomada de decisão, comunicação com diversos dispositivos, veículos e pessoas.

DSRC (do inglês *Dedicated-Short Range Communications*) é um canal de comunicação sem fio, unidirecional ou bidirecional, que opera no espectro de 5,8 ou 5,9 GHz (do inglês *giga-hertz*) e são projetados especificamente para uso automotivo. DSRC é capaz de prover uma comunicação bidirecional entre um veículo e o equipamento instalado na via. Ele desempenha papel principal em muitos ITS, incluindo integração de V2I, comunicação V2V, tempo de semáforo adaptável, cobrança eletrônica de pedágios, taxas de congestionamento, provimento de informações, etc (EZELL, 2010).

As redes sem fio são acessíveis para comunicações rápidas entre veículos e a estrada. Semelhante à tecnologia utilizada para acesso à Internet sem fios, no entanto, têm um alcance de apenas algumas centenas de metros. Este intervalo pode ser estendido, passando a informação para o próximo nó ou veículo, por cada nó na estrada ou por cada veículo (EZELL, 2010). Terceira ou quarta geração (3G ou 4G) de redes de telefonia móvel padrão podem ser utilizadas para transmitir informações em aplicações de ITS. Existem vantagens, tais como a grande disponibilidade em cidades e ao longo das principais estradas. No entanto, a telefonia móvel pode não ser adequada para algumas aplicações ITS críticas de segurança, uma vez que pode ser muito lenta (EZELL, 2010).

Os chamados "*probe vehicles*" (geralmente táxis ou veículos de propriedade do governo equipados com DSRC ou outra tecnologia sem fio) são implementados por vários países. Este

equipamento abrange a velocidade e localização dos veículos a um centro de gerenciamento de operações de tráfego, onde os dados são montados para gerar uma imagem do fluxo de tráfego em toda a região e para avaliar locais congestionados. A pesquisa extensiva também tem sido conduzida com a utilização de celulares que os motoristas carregam, sendo um mecanismo para gerar as informações de trânsito em tempo real, utilizando a localização do telefone derivada por o sistema de posicionamento global (GPS, do inglês *Global Positioning System*), uma vez que se move junto com o veículo. Por exemplo, em Pequim, mais de 10.000 táxis e veículos comerciais foram equipados com chips GPS, que enviam informações sobre a velocidade de viagem a um satélite, que envia as informações para o *Beijing Information Transportation Center* e, por fim, os dados são convertidos para velocidades médias de viagem em todas as estradas na cidade (EZELL, 2010).

As tecnologias de processamento visual tornaram-se um bem valioso e um componente crítico para libertar todo o potencial de muitas aplicações ITS relacionadas ao tema. No que diz respeito a essas tecnologias, podemos citar algumas áreas: vídeo vigilância, monitoramento de tráfego, veículos autônomos ou semiautônomos, sistemas visuais para assistência de condução, detecção classificação e rastreamento de veículos, detecção e rastreamento de pedestres, previsão de intenção do motorista; reconhecimento de placa, etc. Como exemplo de um dispositivo que faz parte desse tipo de aplicação temos a câmera de tráfego inteligente (do inglês *smart traffic camera*), que pode analisar a condição do tráfico, a condição climática ou detectar acidentes.