

Faculdade de Engenharia da Universidade do Porto



O problema de escalonamento de autocarros elétricos

Gonçalo Ribeiro

VERSÃO DE TRABALHO

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Informática e Computação

Orientador: Prof.^a Teresa Galvão Dias

Julho de 2019

Resumo

A presente tese aborda o problema de escalonamento de veículos (VSP) no caso dos autocarros elétricos. Este é um problema recente que surgiu com a adoção cada vez mais prevalente de veículos elétricos no setor dos transportes públicos. A adoção de autocarros elétricos é levada a cabo devido a motivos ecológicos, mas também, em alguns casos, para redução de custos, tanto em combustíveis como também em subsídios governamentais.

Um problema VSP consiste em atribuir veículos de uma frota a um conjunto de viagens previamente planeadas, com redução de custos em mente. Pretende-se otimizar esta atribuição, minimizando o número de veículos e condutores necessários para cobrir todas as viagens agendadas. Este tipo de problemas pode ser formalizado como um problema de Programação Inteira, podendo ser resolvido por métodos exatos de Investigação Operacional, como por exemplo *Branch-and-Bound* e *column generation*. No entanto, a complexidade NP-hard deste tipo de problemas, aliado ao elevado número de restrições adicionais impostas no caso específico dos autocarros elétricos (como a capacidade e velocidade de recarga das baterias), faz com que algoritmos exatos sejam demasiado dispendiosos computacionalmente, requerendo muitas vezes o relaxamento de algumas restrições para serem utilizados. Nestes casos, adequam-se mais os algoritmos meta-heurísticos porque conseguem equilibrar a qualidade das soluções com o tempo computacional despendido.

Foram desenvolvidos dois algoritmos meta-heurísticos multi-objetivo para gerar soluções de escalonamento de autocarros elétricos: um baseado em algoritmos genéticos, outro em *Simulated Annealing*. Ambos foram testados para os mesmos conjuntos de viagens. As soluções produzidas foram comparadas qualitativamente. Também se avaliou a performance computacional exibida.

Concluiu-se que ambas as abordagens são aptas para produzir múltiplas soluções viáveis em tempo útil. Ambos os algoritmos apresentam uma performance e qualidade de soluções semelhante para escalonamentos sobre o mesmo conjunto de dados. Estas implementações, embora simplifiquem vários aspectos envolvidos na escalonamento de autocarros elétricos, podem funcionar como um estimador do tamanho da frota e dos condutores necessários para se migrar de um serviço de autocarros normais para um de autocarros elétricos. Podem também servir como base de desenvolvimento para soluções aplicáveis a casos reais.

Palavras-chave: Problema de Escalonamento de Veículos; Autocarros Elétricos, Otimização Multiobjetivo; Algoritmo Genético; *Simulated Annealing*.

Abstract

This thesis addresses the vehicle scheduling problem (VSP) in the case of electric buses. This problem gained visibility and interest due to the progressively more widespread adoption of electric vehicles on the public transportation sector. This adoption has been taking place not only because of ecological and environmental reasons, but also, in some cases, for cost reduction, in both fuel spendings and governmental subsidies.

A VSP problem consists on the assignment of a fleet of available vehicles to previously scheduled trips, with cost reduction in mind. The purpose is to optimize this assignment, thus minimizing the number of vehicles and drivers needed to cover all the scheduled trips. This type of problems can be formulated as an Integer Programming problem, and therefore are able to be solved by Operational Research methods, such as Branch-and-Bound and column generation algorithms. However, the NP-hard complexity of these problems, together with the substantial amount of hard constraints imposed by an electrical bus scheduling scenario (such as the battery capacity and recharge rate), makes these exact methods become too computationally expensive, requiring some constraint relaxations to produce solutions in feasible time. In these cases, meta-heuristic algorithms may be more appropriate since they are able to balance exploration and exploitation of the search space to produce viable solutions in a reasonable amount of time.

Two multi objective meta-heuristics were implemented to solve this problem: one based in Genetic Algorithms, another based in Simulated Annealing. Both were tested with the same sets of scheduled trips as to provide a fair comparison. The resulting solutions were evaluated, as well as computational performance.

The implemented algorithms were able to produce multiple viable solutions in feasible time. Both approaches display similar performance and solution quality for scheduling problems over the same set of data. These implementations, although simplified, present a viable estimator for the number of vehicles and number of drivers needed to perform a transition from a public transportation service that uses regular buses to one that uses electric buses. This work may as well serve as the basis of development for more elaborate solutions, able to be applied to real life scenarios.

Keywords: Vehicle Scheduling Problem; Electric Buses; Multiobjective Optimization; Genetic Algorithms; Simulated Annealing.

Índice

Capítulo 1.....	1
Introdução	1
Contexto.....	1
Motivação	2
Objetivos	3
Estrutura do Documento	3
Capítulo 2.....	4
Revisão Bibliográfica	4
Problema de Escalonamento de Veículos	4
Problema de Escalonamento de Veículos Elétricos	5
Algoritmos Metaheurísticos	8
Capítulo 3.....	10
Formalização do Problema	10
Formulação Básica.....	10
Restrições de Veículos Elétricos.....	11
Objetivos de Otimização.....	12
Capítulo 4.....	13
Implementação.....	13
Proposta de Solução.....	13
Geração dos Blocos.....	13
Algoritmo Genético	16
Simulated Annealing.....	19
Ferramentas e Tecnologias.....	20
Capítulo 5.....	21
Testes 21	
Metodologia	21
Resultados	22
Análise	33
Capítulo 6.....	34
Conclusões	34
Resumo do Documento.....	34
Trabalho Futuro e Melhorias.....	35

Lista de figuras

Figura 1 - As 5 etapas do processo de planeamento de uma rede de transportes urbanos (Posthoorn 2016). 1

Figura 2 - Fluxo de passos do algoritmo k-Greedy (Paul & Yamada 2014)..... 7

Figura 3 - Exemplo de uma tabela de viagens - linha 601 dos transportes urbanos STCP, no Porto.. 10

Figura 4 - Exemplo de uma solução representada por um subconjunto de blocos (Chen & Zuo 2014).. 15

Figura 5 - Seleção da próxima viagem de um bloco (Chen & Zuo 2014). 16

Lista de tabelas

Tabela 1 - Soluções para 10 repetições do algoritmo genético, com 16 indivíduos e 100 gerações, com diferentes probabilidades de cruzamento e mutação por repetição.....	22
Tabela 2 - Soluções para 10 repetições do algoritmo genético, com 16 indivíduos, probabilidade de cruzamento de 80%, probabilidade de mutação de 5% e número variável de gerações por repetição.	23
Tabela 3 - Soluções para 10 repetições do algoritmo genético, probabilidade de cruzamento de 80%, probabilidade de mutação de 5%, 50 gerações por repetição e número variável de indivíduos por repetição.	24
Tabela 4 - Soluções para 10 repetições do algoritmo <i>simulated annealing</i> , com 25 iterações por repetição e parâmetro de temperatura variável.....	25
Tabela 5 - Soluções para 10 repetições do algoritmo <i>simulated annealing</i> , com parâmetro de temperatura igual a 100 e número de iterações variável.....	26
Tabela 6 - Soluções para 10 repetições do algoritmo genético, probabilidade de cruzamento de 80%, probabilidade de mutação de 5%, 100 gerações e 16 indivíduos por repetição e número variável de capacidade de bateria dos autocarros.	27
Tabela 7 - Soluções para 10 repetições de <i>simulated annealing</i> , com parâmetro de temperatura igual a 100, 25 iterações por repetição e número variável de capacidade de bateria dos autocarros.....	28
Tabela 8 - Soluções para 10 repetições de algoritmo genético vs. <i>simulated annealing</i> , para um conjunto de viagens correspondente a 16h de trabalho numa única linha de autocarros.....	29
Tabela 9 - Soluções para 10 repetições de algoritmo genético vs. <i>simulated annealing</i> , para um conjunto de viagens correspondente a 16h de trabalho numa rede de autocarros com 4 linhas.	30
Tabela 10 - Soluções para 10 repetições de algoritmo genético com diferentes métodos de produção de blocos.....	31
Tabela 11 - Soluções para 10 repetições de <i>simulated annealing</i> com diferentes métodos de produção de blocos.....	32

Abreviaturas e Símbolos

CDP	Candidate Departure Point
DR	Demand Response
EV	Electric Vehicle
GA	Genetic Algorithm
SSB	Stationary Storage Batteries
VSP	Vehicle Scheduling Problem
SDVSP	Single Depot Vehicle Scheduling Problem
MDVSP	Multi Depot Vehicle Scheduling Problem

Capítulo 1

Introdução

Contexto

Se uma cidade quiser implementar uma rede de transportes urbanos, deve elaborar um concurso público que declara as suas necessidades e expectativas a serem correspondidas pelas propostas das empresas participantes. Isto passará por fazer um levantamento à população e tentar compreender a necessidade de transporte para cada área da cidade, de forma a construir uma ideia geral da quantidade e morfologia das rotas necessárias. Também é necessário, por parte do município, estimar o fluxo de passageiros em função da hora e dia da semana. Com esta informação, uma empresa participante pode começar a desenhar uma rede completa de transportes. O processo de planejar uma proposta é feito em várias etapas cronológicas.

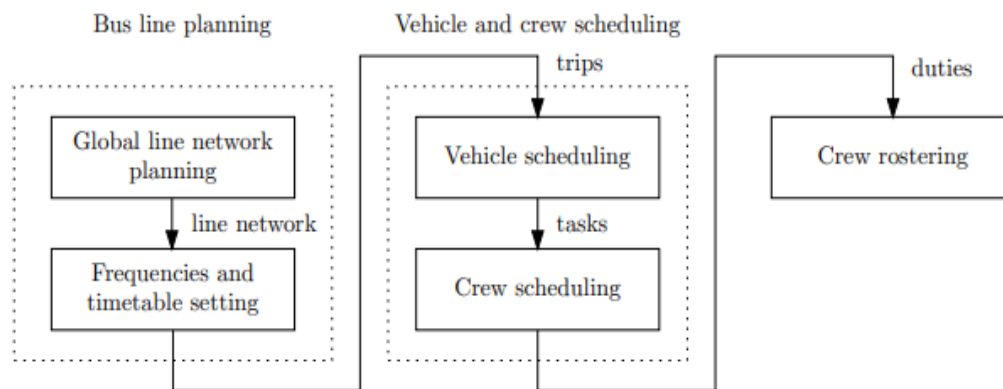


Figura 1 - As 5 etapas do processo de planeamento de uma rede de transportes urbanos (Posthoorn 2016).

Na fase "Global line network planning", é necessário planejar uma rede de rotas. Esta rede tem que tomar em consideração a afluência de passageiros para cada zona da cidade. É necessário definir os melhores locais para se estabelecerem paragens. Uma paragem, ou ponto, é um ponto de serviço onde os veículos param temporariamente para poderem recolher ou entregar passageiros. O local de cada paragem é escolhido de forma a equilibrar em termos de custo os seguintes objetivos:

- Maximizar o número de clientes que podem ser servidos por cada paragem, reduzindo o número total de paragens necessárias e, consequentemente, o custo final do serviço.
- Maximizar o número total de clientes que podem ser servidos pela cidade.

Trata-se então de encontrar o melhor compromisso entre melhorar conveniência para os passageiros (mais paragens) e reduzir o orçamento final (menos paragens).

Na fase "Frequencies and timetable setting", é determinada a frequência e a hora de cada rota. Uma rota é um conjunto sequencial de viagens entre duas paragens, produzindo uma linha de transporte.

Inicialmente, definem-se as frequências para cada rota através de regras gerais, tais como:

- Frequências altas durante as horas de ponta da manhã e do final de tarde.
- Frequências baixas antes do amanhecer e depois da hora de ponta no final de tarde.
- Frequências normais entre as horas de ponta durante o dia.

Outra abordagem é simular a afluência de passageiros em função do tempo e estimar as frequências necessárias para satisfazer a demanda.

Depois de se escolherem as frequências, definem-se os horários. É feito um cronograma no qual os horários de partida e chegada para cada viagem são declarados. Durante a definição deste cronograma, as transferências para outras linhas (de autocarro, metro, comboio ou outros) deverão ser levadas em consideração, de forma a otimizar as transferências. O horário é definido em torno de um tipo de dia. Na maioria das vezes são decididos três tipos de dia: dias úteis, fins-de-semana e dias especiais (feriados ou outras ocorrências extraordinárias).

Depois de serem definidos os horários, vem a fase de "Vehicle scheduling", onde os veículos devem ser atribuídos a cada viagem. Esta escalonamento de veículos é feita de tal forma a minimizar o número total de veículos necessários para cobrir todas as viagens. A escalonamento de veículos é o foco desta dissertação.

De seguida, a fase de "Crew scheduling". Depois de conhecida a atribuição de veículos às diferentes viagens, é necessário atribuir uma tripulação a cada veículo. A tripulação é um conjunto de motoristas que dividem entre si por turnos o encargo de conduzir um determinado veículo. Dentro da escalonamento da tripulação, a legislação trabalhista e os acordos devem ser levados em consideração. O objetivo é novamente minimizar os custos de modo que a cada veículo seja atribuída uma tripulação compatível, ou seja, que consiga garantir um motorista para cada viagem sem violar legislação.

Finalmente, vem a fase de "Crew rostering". É necessário atribuir um elemento de cada tripulação a cada viagem do veículo. Esta escalonamento pode ser alterada com alguma frequência, de forma a melhor adaptar-se à disponibilidade ou conveniência dos condutores. Leis do trabalho têm que ser tomadas em conta, como a necessidade de dias de folga e limites de duração de turnos.

Motivação

Dentro do setor dos transportes urbanos, nota-se cada vez mais uma consciencialização ambiental. O número de iniciativas para redução de emissões atmosféricas tem vindo a aumentar cada vez mais. Incluída nessas iniciativas inclui-se a substituição de autocarros a diesel e gás natural por autocarros elétricos, iniciativa tal que já pode ser observada na Eindhoven, na Holanda (Posthoorn 2016), e em Shenzhen, na China (Wu et al. 2013). Em vários países, já existem incentivos governamentais para empresas que adotem veículos elétricos.

No entanto, a transição de veículos a combustíveis fósseis para veículos eléctricos trás consigo novos problemas que têm que ser resolvidos de antemão. Um desses problemas é a escalonamento de veículos mencionada previamente. As restrições inerentes aos veículos eléctricos têm que ser tomadas em consideração na produção de uma escalonamento viável. Atualmente, a capacidade da bateria de um autocarro eléctrico não é suficiente para um dia inteiro de operação, o que não acontece com um depósito de autocarros híbridos ou a combustíveis fósseis. Isto implica a inclusão de recargas intermitentes ao longo do dia, o que geralmente implica um maior número de veículos em serviço para garantir a cobertura de todas as viagens. Devido à atualidade destes problemas, a investigação feita ainda é escassa. A motivação desta dissertação é exatamente contribuir para a resolução destes problemas.

Objetivos

Os objetivos desta dissertação são os seguintes:

- Análise e revisão dos algoritmos de estado da arte para o problema geral de escalonamento de veículos.
- Análise e revisão dos algoritmos de estado da arte para o problema de escalonamento de veículos eléctricos, mais especificamente, no caso de autocarros eléctricos.
- Desenhar e implementar uma solução de escalonamento de veículos que tenha em consideração as restrições adicionais dos autocarro eléctricos.
- Testar a solução produzida com instâncias reais ou geradas aleatoriamente e comparar com outras soluções existentes atualmente.

Estrutura do Documento

Os próximos capítulos deste documento vão abordar o seguinte:

- O Capítulo 2 contém toda a pesquisa científica relativamente aos problemas de escalonamento de veículos, distinguindo entre problemas de depósito único (single depot) e múltiplos depósitos (multiple depots). São analisadas as soluções do estado da arte para escalonamento de autocarros eléctricos e não eléctricos.
- O Capítulo 3 contém a formalização do problema de escalonamento de veículos eléctricos, apresentando os termos e conceitos básicos que serão utilizados ao longo do documento.
- O Capítulo 4 explica a implementação das soluções desenvolvida, baseada na revisão bibliográfica do Capítulo 2. Também se abordam as tecnologias e ferramentas utilizadas.
- O Capítulo 5 apresenta uma metodologia para testar as soluções desenvolvidas, assim como os resultados obtidos e análise dos mesmos.
- O Capítulo 6 finda este documento, resumindo as conclusões obtidas em cada capítulo, apresentando possíveis melhorias e trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

Problema de Escalonamento de Veículos

O problema de escalonamento de autocarros pertence à classe geral de problemas de escalonamento de veículos (VSP). Os problemas VSP visam atribuir um veículo a cada viagem calendarizada, procurando fazer uso do menor número de veículos possíveis. Os veículos podem partir de um depósito - Single Depot VSP (SDVSP) - ou mais depósitos - Multi Depot VSP (MDVSP) - e têm que visitar colectivamente um certo número de locais antes de regressarem ao depósito do qual partiram. Ora, isto trata-se de um problema inteiro - a solução para este problema trata-se de uma associação unívoca de cada viagem a um autocarro, sendo o número total de autocarros um número inteiro.

Este tipo de problemas em instâncias reais é bastante complexo. O grande número de viagens e transferências a serem considerados aumentam rapidamente o número de variáveis e restrições em qualquer modelo desenvolvido. São problemas que, em geral, pertencem à classe de complexidade NP-hard. Isto significa que a carga computacional para os resolver aumenta exponencialmente à medida que o tamanho do problema aumenta (Haghani et al. 2003). Estes problemas podem ser resolvidos de forma exata, usando técnicas de otimização ou métodos heurísticos de forma a reduzir a sua complexidade. As considerações envolvidas nos problemas de escalonamento de viaturas incluem uma função objetivo, restrições operacionais e restrições específicas. As seguintes condições devem ser satisfeitas:

1. Cada viagem é executada por exatamente um veículo.
2. Cada bloco de viagens começa e termina no mesmo depósito.
3. Cada depósito tem um determinado número máximo de veículos (capacidade).
4. Todas as restrições operacionais, incluindo qualquer restrição no tempo total que um veículo passa longe do depósito, são satisfeitas.

Este tipo de problemas já foi estudado extensivamente ao longo dos anos, existindo vários algoritmos capazes de obter soluções aceitáveis em tempo viável:

- Branch-and-Cut (Hadjar et al. 2006)
- Column Generation (Ceselli et al. 2009)
- Large Neighborhood Search (Ceselli et al. 2009; Pepin et al. 2008)
- Tabu Search (Ceselli et al. 2009; Pepin et al. 2008)
- Genetic Algorithm (GA) (Chen & Zuo 2014).

No entanto, estas abordagens teriam que ser adaptadas ou estendidas a problemas de escalonamento de veículos elétricos, visto que estes impõem novas restrições que têm de ser tomadas em consideração pelos algoritmos.

Depósito Único

Um problema de escalonamento de veículos de depósito único contém a estrutura básica do problema de escalonamento. Suponha que existem viagens a serem servidas por veículos a partir de um único depósito. Cada viagem tem um ponto de partida, um ponto final, uma hora inicial e uma hora final. Duas viagens podem ser servidas pelo mesmo veículo se a hora de início de uma for maior que o tempo de locomoção entre o ponto final da primeira e o ponto inicial da segunda mais a hora final desta última. O objetivo do problema é encontrar o número mínimo de veículos para atender a todas as viagens.

Uma grafo poderia ser construído para representar este problema: cada viagem seria representada por um nó, e uma aresta existe se o tempo final da viagem mais o tempo de viagem entre o ponto final da viagem e o ponto inicial da viagem é menor que o tempo de partida da viagem. Cada veículo deverá realizar um conjunto de viagens a começar e terminar no mesmo depósito. Este problema poderia ser resolvido como um problema *minimal flow cost* (Haghani et al. 2003).

Às vezes, se o tempo de espera nos pontos de término das viagens for considerado na função objetivo, o problema pode ser resolvido usando um procedimento de duas fases (Bodin & Golden 1981). Basicamente, o problema de escalonamento de um único depósito pode ser resolvido com relativa facilidade. No entanto, quando outras restrições realistas, como as restrições de tempo de rota, são introduzidas, este torna-se num problema NP-Hard. Grandes problemas deste tipo apenas podem ser resolvidos por procedimentos heurísticos.

Múltiplos Depósitos

O problema de escalonamento de veículo de depósito múltiplo (MDVSP) é uma extensão do problema de escalonamento de veículo de depósito único. A principal diferença entre os dois é que no caso de vários depósitos, os veículos são alojados em diferentes depósitos. O objetivo é determinar o número mínimo de veículos para atender a todas as viagens e identificar os melhores depósitos iniciais para cada veículo de forma a minimizar o custo total.

Este tipo de problemas, assim como generalizações e casos especiais, foram analisados extensivamente nos últimos 30 anos por vários autores (Adler & Mirchandani 2017; Bodin 1983; Bunte & Kliwer 2009; Ceder 2002; Daduna & Pinto Paixão 1995; Desaulniers & Hickman 2007; Desrosiers et al. 1995). (Bertossi et al. 1987) prova que este tipo de problemas é NP-Hard.

Problema de Escalonamento de Veículos Elétricos

O problema de escalonamento de veículos para veículos elétricos (eVSP) difere de um VSP em múltiplos aspectos. O aspecto mais notável é a autonomia reduzida que veículos elétricos têm em comparação com veículos a combustíveis fósseis, devido à capacidade das baterias atualmente existentes. Como consequência, um autocarro elétrico precisa de ser reabastecido várias vezes ao dia, enquanto que um autocarro a diesel apenas é reabastecido uma vez por dia. O reabastecimento de autocarros elétricos pode ser feito de duas maneiras:

- Através de uma recarga das suas baterias em estações de recarga designadas.
- Através da substituição das baterias usadas por baterias carregadas.

Ambas as alternativas implicam uma quantidade significativa de tempo morto (em que o autocarro não está a operar), que deve ser tomado em conta na produção de soluções de escalonamentos de veículos.

Como já foi mencionado neste capítulo, o objetivo de um VSP é encontrar uma escalonamento de veículos de modo a que cada viagens seja atribuído um veículo, que a sequência das viagens de cada veículo seja válida e que os custos totais sejam minimizados. Os veículos elétricos têm um impacto na viabilidade da sequência das viagens atribuídas a cada veículo. Um sequência de viagens é considerada viável quando não viola as seguintes restrições:

- A sequência começa e termina num depósito.
- O tempo de conexão entre duas viagens ou uma viagem e um depósito é suficiente para um veículo cobrir a distância.

No entanto, mais restrições têm que ser consideradas para eVSP:

- O consumo energético não excede os limites da bateria, havendo sempre energia suficiente para as viagens.
- Um autocarro deve visitar uma estação de recarga caso não tenha energia suficiente para próxima viagem.
- O número máximo de autocarros a carregar simultaneamente numa estação de recarga não deve exceder os limites da estação.

Um eVSP pode ser visto com a combinação de um problema VSP e um problema de escalonamento das recargas dos veículos. O problema do escalonamento das recargas tem sido estudado nos últimos anos, havendo já algumas soluções publicadas na literatura relevante:

- Genetic Algorithm (Chen & Zuo 2014).
- Ant Colony Algorithm (Wang & Shen 2007).
- Dynamic Scheduling (Lan et al. 2012).
- Exponential Smoothing Model (Lan et al. 2012; Aabrandt et al. 2012).
- Locally Optimal Scheduling (He et al. 2012).

No entanto, estas soluções não são completas, na medida em que não abordam também a escalonamento de veículos requerida por eVSP. Ao contrário de outras tarefas de escalonamento, a complexidade deste problema reside no cálculo da quantidade de carga dos autocarros elétricos e da energia restante nas suas baterias, porque a energia restante de uma bateria é uma função da energia de todos os eventos anteriores de carga e descarga (Paul & Yamada 2014). Assim, este documento vai-se focar na análise de três soluções completas e atuais para eVSP, testadas com sucesso com dados de instâncias reais:

- k-Greedy Algorithm (Paul & Yamada 2014).
- Modified Column Generation (Posthoorn 2016).
- Branch-and-Price Algorithm (Adler & Mirchandani 2017).

k-Greedy

O algoritmo *k-Greedy* é um tipo de algoritmo ganancioso, proposto em (Paul & Yamada 2014), que simula um algoritmo *k-nearest-neighbours* para encontrar os melhores candidatos a depósito de partida (CDP) em problemas eVSP com múltiplos depósitos.

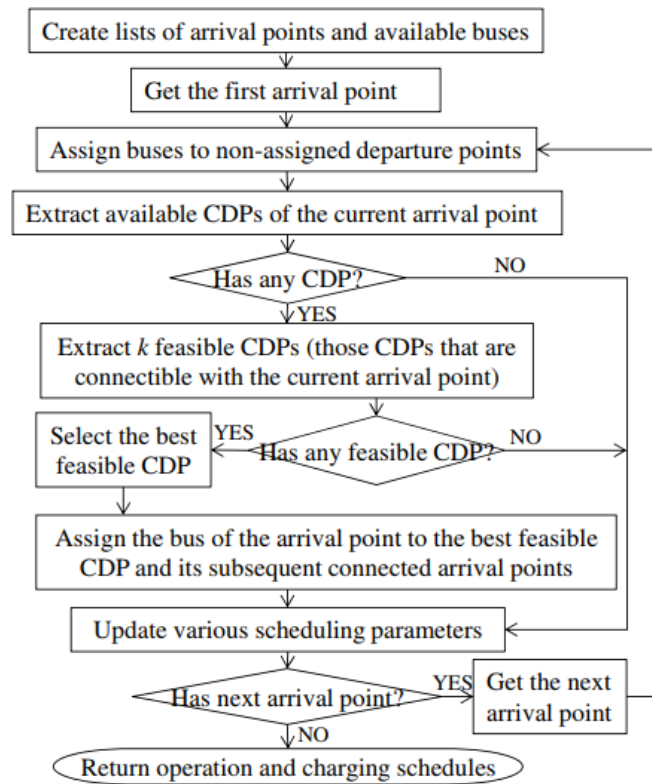


Figura 2 - Fluxo de passos do algoritmo k-Greedy (Paul & Yamada 2014).

Os passos iterativos deste algoritmo podem ser observados na Figura 2. Primeiramente, CDPs são extraídos da lista de CDPs viáveis para cada ponto de chegada. Para este propósito, a viabilidade de uma deslocação entre cada ponto de chegada e cada um dos seus CDPs é verificada um a um. Quando o melhor CDP viável é encontrado, é seguidamente adicionado à lista de possíveis CDPs que podem integrar a solução final. Quando CDPs viáveis são encontrados, deixa-se de verificar a validade das translocações descrita anteriormente. Em seguida, os CDPs viáveis são avaliados e o melhor CDP viável é retornado. Depois, o ponto de chegada e o melhor CDP possível são conectados e vários parâmetros de escalonamento são atualizados. Como critério de avaliação, a distância ou a energia de consumo requerida da viagem de autocarro originada no CDP pode ser usada. A definição exata do algoritmo, incluindo fórmulas, encontra-se em (Paul & Yamada 2014).

Os autores sugerem que este algoritmo pode ser utilizado em frotas mistas com autocarros a diesel e autocarros elétricos. Foram feitas simulações utilizando dados reais recolhidos de quatro cidades no Japão. Dos resultados obtidos, os autores concluíram que este algoritmo retorna maiores distâncias totais a serem percorrida pelos veículos, por quanto mais veículos tiver a solução final. No entanto, notaram que em comparação com outra solução baseada em Algoritmos Genéticos, este algoritmo, em média, terá sempre uma melhor performance.

Column Generation

O algoritmo *Column Generation* adaptado é um tipo de algoritmo proposto em (Posthoorn 2016) que se baseia no algoritmo *Column Generation* original para VSP (Desaulniers et al. 1998), modificado para tomar em consideração as novas restrições de veículos elétricos. Este tipo de algoritmos adequa-se

melhor a problemas inteiros grandes, com mais variáveis que restrições, com é o caso em eVSP. A definição exata do algoritmo, incluindo fórmulas, encontra-se em (Posthoorn 2016).

O algoritmo foi testado com dados reais de sete linhas de autocarro do sistema de transportes urbanos da cidade de Eindhoven, na Holanda. Esta cidade já utiliza exclusivamente veículos elétricos e já tinha uma solução de escalonamento em vigor. Este algoritmo de *Column Generation* adaptado conseguiu, sobre as mesmas condições, obter uma solução final melhor (40 autocarros) do que a solução implementada pelo município (48 autocarros). O autor da tese realça que certos aspectos deste modelo foram simplificados, como as funções do custo da eletricidade ao longo do dia e a velocidade da recarga das baterias ao longo do tempo.

Branch-and-Price

O algoritmo *Branch-and-Price* é um tipo de algoritmo baseados em *Branch-and-Bound*, proposto em (Adler & Mirchandani 2017), para problemas eVSP com múltiplos depósitos. Algoritmos *Branch-and-Bound* são uma classe de algoritmos que encontram soluções para problemas inteiros resolvendo repetidamente relaxamentos lineares do problema original. Se a solução para o relaxamento linear não é inteira, então o problema é seguidamente ramificada em múltiplos subproblemas que também são relaxamentos lineares. Algoritmos *Branch-and-Price* são um caso especial de algoritmos *Branch-and-Bound*, onde a geração de colunas é usada para resolver o subproblema de cada ramificação. Nos algoritmos de *Branch-and-Price*, as colunas das soluções aos relaxamentos lineares são usadas como colunas iniciais nos problemas ramificados. A definição exata do algoritmo, incluindo fórmulas, encontra-se em (Adler & Mirchandani 2017).

Este algoritmo foi testado com ambos dados aleatórios e dados extraídos da rede Valley Metro da cidade de Phoenix, Arizona. Apresentou uma boa performance, no entanto os autores sugerem a implementação de heurísticas para melhorar mais o desempenho do algoritmo.

Algoritmos Metaheurísticos

Num algoritmo genético, cada indivíduo é representado por um cromossoma (ou *string*) que contém um conjunto de genes que o caracteriza. Cada cromossoma representa uma potencial solução (válida ou não) para o problema de escalonamento (Watanabe et al. 2005). Gera-se então uma população inicial de cromossomas. Os cromossomas são então cruzados entre si e alguns dos seus genes sofrem mutações de forma promover exploração do espaço de soluções. São selecionados os cromossomas de melhor qualidade para formar a próxima geração, voltando-se a repetir os mesmos procedimentos até que uma condição de paragem seja atingida. Pretende-se que a qualidade média dos cromossomas aumente de geração para geração.

Algoritmos genéticos há muito que são utilizado para problemas de escalonamento (Malmberg 1996; Brandão & Mercer 1997). A codificação em cromossomas mais utilizada para este tipo de problemas é uma codificação de permutação (Yamada 1995). Numa codificação por permutação, todos os cromossomas contêm exatamente os mesmos genes mas em ordens diferentes. Este tipo de codificação é utilizada em soluções para o problema de escalonamento de veículos, tais como (Park 2001), que propõe um algoritmo genético com restrições relaxadas combinado com um reparador de cromossomas. Há várias outras soluções, propondo codificações e operadores diferentes, cada uma adaptada ao seu cenários específicos (Zhou et al. 2017; Malmberg 1996). O artigo (Chen & Zuo 2014) propõe uma maneira de contornar a necessidade de conceber operadores específicos para uma dada escalonamento de veículos, dividindo o algoritmo genético em duas fases: uma primeira fase para gerar sequência de viagens válidas,

e uma segunda fase onde se usa um algoritmo genético genérico para atribuir as tais sequências a veículos, de forma a que todas as viagens agendadas sejam cobertas. Esta técnica permite, efetivamente, migrar complexidade de operadores especializados para a primeira fase de geração de sequências de viagens.

Um algoritmo *simulated annealing* procura simular uma estratégia de otimização observada no arrefecimento de metais. Uma variável que representa a temperatura vai sendo decrementada de iteração em iteração. Começa-se por gerar uma solução inicial. Para cada iteração, gera-se uma nova solução vizinha a partir da última solução em vigor. Quanto maior a temperatura numa dada iteração, maior a probabilidade de a solução em vigor ser substituída pela nova solução vizinha, mesmo que esta última seja pior que a primeira. Isto promove exploração global no início do algoritmo, prevenindo convergência para máximos locais. À medida que a temperatura baixa a cada iteração, a transição para a solução vizinha torna-se mais rigorosa, promovendo exploração local. A probabilidade de transitar para uma solução pior acompanha a redução da temperatura.

Não sendo tão prevalentes como os genéticos, existem algumas soluções de escalonamento de veículos baseadas em simulated annealing, tais como (Brandão & Mercer 1997), (Xia & Yi 2014), (Wang 2014) e (Phan 2017).

Capítulo 3

Formalização do Problema

Formulação Básica

Esta secção visa introduzir formalmente o problema de escalonamento de veículos elétricos, com as devidas definições e termos técnicos que serão usados ao longo da tese.

CORDOARIA	BOAVISTA	ROTUNDA AEP	MARSHOPPING	FREIXEIRO	AEROPORTO
	05:30	05:37	05:47	05:57	06:01
06:00	06:07	06:14	06:24	06:34	06:38
06:25	06:32	06:39	06:49	06:59	07:03
06:48	06:56	07:04	07:15	07:27	07:31
07:13	07:21	07:29	07:40	07:52	07:56
07:38	07:46	07:56	08:08	08:20	08:25
08:03	08:14	08:24	08:36	08:48	08:52
08:38	08:49	08:58	09:10	09:22	09:26
09:10	09:21	09:30	09:42	09:54	09:58
09:45	09:56	10:05	10:17	10:29	10:33
10:20	10:31	10:40	10:52	11:04	11:08
10:55	11:06	11:15	11:27	11:39	11:43
11:30	11:41	11:50	12:02	12:14	12:18
12:05	12:16	12:25	12:37	12:49	12:53
12:40	12:51	13:00	13:12	13:24	13:28
13:13	13:24	13:33	13:45	13:58	14:03
13:43	13:54	14:03	14:15	14:28	14:33
14:13	14:24	14:33	14:45	14:58	15:03
14:43	14:54	15:03	15:16	15:29	15:34
15:13	15:24	15:33	15:46	15:59	16:04
15:43	15:53	16:02	16:15	16:28	16:33
16:13	16:25	16:34	16:47	17:00	17:05
16:43	16:55	17:05	17:18	17:31	17:36
17:13	17:25	17:35	17:48	18:01	18:06
17:43	17:55	18:07	18:20	18:33	18:38
18:13	18:25	18:37	18:50	19:02	19:06
18:43	18:55	19:07	19:20	19:31	19:35
19:13	19:23	19:33	19:45	19:56	20:00
19:45	19:55	20:03	20:15	20:26	20:30
20:25	20:33	20:41	20:53	21:04	21:08
21:05	21:12	21:19	21:30	21:41	21:45
21:55	22:02	22:09	22:20	22:31	22:35
22:45	22:52	22:59	23:10	23:21	23:25
23:35	23:42	23:49	00:00	00:11	00:15
00:25	00:32	00:39	00:50	01:01	01:05
00:55	01:02				

Figura 3 - Exemplo de uma tabela de viagens - linha 601 dos transportes urbanos STCP, no Porto.

- Seja $t_i = (s_a, s_b, h_i, \Delta h)$ o tuplo correspondente uma viagem da paragem s_a para a paragem s_b à hora h_i com duração Δh . A tabela T é o conjunto de todas as viagens agendadas. A duração de cada viagem é estimada no processo de roteamento e agendamento das viagens, que precede a escalonamento.
- Seja v_i um veículo. V é a frota ou o conjuntos dos veículos disponíveis.

O problema de escalonamento de veículos (VSP) reside em atribuir um veículo $v \in V$ a cada viagem $t \in T$, produzindo uma sequência de viagens para cada veículo utilizado. Um exemplo de uma sequência de viagens, baseada na Figura 3, seria:

$$S_0 = \{(Boavista, RotundaAEP, 5:30, 7'), (RotundaAEP, Marshopping, 5:47, 10'), \dots\}$$

Uma escalonamento seria então um conjunto de sequências $A = \{S_0, S_1, \dots, S_n\}$ tal que o conjunto das viagens agendadas estivesse contido na reunião de todas as sequências, ou seja, $T \subseteq S_0 \cup S_1 \cup \dots \cup S_n$ para todo o $S \in A$. No entanto, estas sequências de viagens estão sujeitas às seguintes restrições (*hard constraints*):

- Não podem haver duas viagens que se intersetem no tempo na mesma sequência. Por exemplo, a sequência $S = \{(A, B, h, 10'), (B, C, h + 5', 10')\}$ é inválida porque a segunda viagem começa antes da primeira acabar, $h + 5' < h + 10'$.
- Uma sequência tem que começar e acabar num depósito. Seja D o conjunto de todos os depósitos, $S = \{(A, B, h_0, 10'), \dots, (Y, Z, h_n, 10')\}$ só é uma sequência válida caso A e Z pertençam a D , ou seja, $\{A, Z\} \in D$. Na maioria das situações, o depósito de chegada deverá ser o mesmo que o de saída, ou seja, $A = Z$. Caso só exista um depósito, $|D| = 1$, trata-se de um problema de escalonamento de depósito único. Caso , $|D| > 1$, trata-se de um problema de escalonamento de múltiplos depósitos. As viagens de e para os depósitos não costumam ser agendadas previamente, pelo que têm que ser incluídas no processo de escalonamento.
- Numa sequência, cada viagem tem que começar onde a anterior acabou, com exceção da primeira dado que não tem uma viagem que a antecede. Por exemplo, a sequência $S = \{(A, B, h, 5'), (C, D, h + 10', 5')\}$ é inválida visto que a segunda viagem não começa na paragem B, onde a primeira viagem acaba.

Atendendo à última restrição, pode ser permitido a adição de novas viagens conetoras para garantir a continuidade da sequência de paragens. No exemplo anterior, a sequência $S = \{(A, B, h, 5'), (C, D, h + 15', 5')\}$ pode ser corrigida com a adição de uma nova viagem conetora de B para C , ficando agora $S = \{(A, B, h, 5'), (B, C, h + 5', 5'), (C, D, h + 15', 5')\}$.

Este tipo de correções só é possível se for disponibilizada uma função *interval*: $L \times L \rightarrow R$, sendo L o conjunto de todas as paragens, que retorne o intervalo de tempo necessário para se conduzir entre quaisquer duas paragens. No exemplo anterior, a distância entre B e C era menor do que a diferença entre a hora de partida da segunda viagem, $h + 15'$, e a hora de chegada da primeira viagem, $h + 5'$, ou seja, $interval(B, C) = 5' \leq (h + 15') - (h + 5')$. Só se podem adicionar viagens conectoras caso o intervalo de tempo entre a chegada e a partida de duas viagens consecutivas seja maior que o tempo necessário para se viajar do local de chegada ao local de partida da próxima viagem. Estas viagens conectoras são criadas durante o processo e muitas vezes fazem parte da função objetivo, pretendendo-se que o seu número (ou duração) seja o menor possível.

Restrições de Veículos Elétricos

Adicionalmente, no caso dos veículos serem elétricos, tem que se tomar em consideração a nova restrição imposta pela autonomia das suas baterias:

- Considere-se que todos os veículos $v \in V$ têm o mesmo tempo de bateria máximo, quantificado como um intervalo de tempo constante *maxBattery*, e tempo de bateria atual b .
- Considere-se que um veículo pode recarregar a sua bateria num depósito $d \in D$. De modo a simplificar, considere-se que todos os depósitos oferecem a mesma velocidade de recarga e

que não existe limite máximo de número de veículos a recarregar num determinado depósito simultaneamente.

A capacidade da bateria de um veículo após um recarga é dada pela fórmula $b_f = \min(b_i + rechargeMultiplier \times \Delta h, maxBattery)$, sendo b_f a bateria após a recarga, b_i a bateria antes da recarga, $rechargeMultiplier$ uma constante e Δh é o intervalo de tempo que o veículo esteve parado a recarregar num depósito. Por exemplo, seja a bateria atual de um veículo $b_i = 60'$, e as constantes $rechargeMultiplier = 3$ e $maxBattery = 90'$. Considerando a seguinte sequência de viagens para o mesmo veículo $S = \{(d_0, A, h, 5'), (A, B, h + 20', 5')\}$, tal que $d_0 \in D$ é um depósito. O intervalo de tempo a recarregar é dado por $\Delta h = (h + 20') - (h + 5') = 15'$, sendo a capacidade da bateria após a recarga $b_f = \min(60' + 3 \times 15', 90') = \min(105', 90') = 90'$.

Num caso de escalonamento de veículos elétricos, é necessário garantir que, para cada veículo, a bateria restante, b , após cada viagem nunca desça abaixo de 0 ao longo da sequência de viagens atribuída ao veículo. Para verificar esta restrição, calcula-se sequencialmente a bateria do autocarro após cada viagem, utilizando as seguintes fórmulas:

- Seja h_n a hora de partida de uma viagem n .
- Seja h'_n a hora de chegada de uma viagem n .
- Seja l_n o local de partida de uma viagem n .
- Seja l'_n o local de chegada de uma viagem n .
- Se o local de partida de uma viagem n for um depósito, ou seja, $l_n \in D$, então quantidade de bateria recarregada é $recharge_n = rechargeMultiplier \times (h_{n+1} - h'_n)$, caso contrário $recharge_n = 0$.
- A bateria inicial de cada autocarro é dada por $b_0 = maxBattery$ enquanto que a bateria após uma viagem n é dada por $b_n = \min(b_{n-1} + recharge_{n-1}, maxBattery) - interval(l_n, l'_n)$.
- Para uma sequência S de n viagens, a sequência diz-se inválida se houver pelo menos uma viagem que resulte numa bateria restante igual ou inferior a 0, ou seja, $\exists i \in [0, n]: b_i \leq 0$.

Objetivos de Otimização

Podem ser definidas várias funções a otimizar neste problema. No âmbito desta dissertação, apenas serão tomadas em consideração as seguintes:

- Minimização do número de veículos necessários para cobrir todas as viagens agendadas.
- Minimização do número de condutores necessários para cobrir todas as viagens agendadas.
- Minimização do tempo não útil (tempo despendido em descanso ou em viagens conetoras não agendadas).

Capítulo 4

Implementação

Proposta de Solução

Como já foi mencionado no capítulo anterior, o problema de escalonamento dos autocarros elétricos pode ser formalizado como um problema de Programação Inteira, e como tal é possível resolvê-lo usando métodos iterativos de otimização linear, tais como *column-generation* (Guedes & Borenstein 2015; Faiz et al. 2019) e *Branch-and-Bound* (Carpaneto et al. 1989). No entanto, a pesquisa bibliográfica revela que devido à complexidade destes métodos exatos, não é prático considerar todas as novas restrições impostas pelos veículos elétricos visto que aumentam exponencialmente o tempo computacional requerido. Para combater este obstáculo, relaxam-se algumas das restrições para se chegar a uma solução sub ótima, e posteriormente utiliza-se algum método de otimização local para corrigir a solução de forma a não violar restrições.

Para esta dissertação, foi decidido tomar uma abordagem metaheurística por vários motivos:

1. O atual estado da arte do escalonamento de autocarros elétricos apenas contempla soluções baseadas em métodos exatos de Investigação Operacional. Implementar uma solução baseada em metaheurísticas apresentaria assim uma contribuição ao estado da arte.
2. As metaheurísticas conseguem equilibrar exploração global e exploração local do espaço de soluções. Isto permite a flexibilidade necessária para se obterem soluções viáveis em tempo útil.

Foram implementadas duas abordagens metaheurísticas ao problema de escalonamento de autocarros elétricos: uma baseada em algoritmos genéticos e outra baseada em *simulated annealing*. A escolha destes dois tipos de metaheurística baseou-se no facto de já terem sido usadas para resolver problemas de escalonamento de veículos não elétricos (Chen & Zuo 2014; Xia & Yi 2014).

Geração dos Blocos

Inicialmente, procurou-se representar soluções como um mapeamento unívoco de veículos para viagens agendadas. Por exemplo, para o conjunto de viagens $T = \{t_0, t_1, t_2, \dots, t_8\}$, a representação de uma solução pode ser $S = \{V_0, V_1, V_2\}$ tal que V_0, V_1, V_2 representam os conjuntos de viagens atribuídas a cada veículo, como por exemplo, $V_0 = \{t_2, t_5, t_6\}$, $V_1 = \{t_0, t_1, t_4\}$ e $V_2 = \{t_3, t_7, t_8\}$.

Esta maneira trivial de representar uma solução permitia codificações e decodificações eficientes das soluções em indivíduos prontos a serem utilizados no algoritmo genético ou no algoritmo de *simulated annealing*. No entanto, surge um problema. Este tipo de representações triviais resulta num enorme

espaço de soluções, sendo que a maioria destas soluções violam uma ou mais restrições de escalonamento. Isto traz várias desvantagens para algoritmos metaheurísticos:

- Impossibilidade de gerar indivíduos válidos a partir de soluções aleatórias, o que seria uma mais valia na inicialização dos algoritmos genético e simulated annealing.
- Difícil verificar eficientemente se uma solução é válida ou não.
- No caso do algoritmo genético, difícil manter uma população de indivíduos que representem soluções válida. Também é difícil criar operadores que garantam a produção de novos indivíduos válidos.
- No caso de simulated annealing, é difícil gerar novos vizinhos válidos a partir de uma solução prévia.

Tendo em conta estes entraves, optou-se por uma representação de soluções baseada em blocos de viagens, representação tal adaptada do artigo (Chen & Zuo 2014). Em vez de se mapear os veículos diretamente a viagens, mapeiam-se os veículos a sequências de viagens (como formalizado no capítulo anterior), chamados de blocos. Cada bloco já compreende em si uma sequência válida de viagens e cada bloco é construído de forma a ser percorrido por um único veículo e um único condutor. Um indivíduo passa então a ser representado por um mapeamento unívoco de um subconjunto de blocos para o conjunto de todas as viagens agendadas, tal que um bloco só pode ser atribuído a uma viagem se esta última estiver incluída na sequência de viagens do bloco. Por exemplo, para o conjunto de viagens $T = \{t_0, t_1, t_2, \dots, t_8\}$ formam-se os seguintes blocos $B_0 = \{t_0, t_1, t_2, t_3\}$, $B_1 = \{t_3, t_4, t_5, t_6\}$, $B_2 = \{t_6, t_7, t_8\}$, $B_3 = \{t_0, t_1, t_2, t_7, t_8\}$. Uma solução pode ser representada por qualquer conjunto de blocos, desde que o conjunto das viagens agendadas esteja contida na reunião destes blocos. A solução $S = \{B_1, B_3\}$ é válida visto que $T \in B_1 \cup B_2$.

Esta representação implica gerar antecipadamente o conjunto de blocos disponíveis, mas a sua utilização traz várias vantagens:

- Calcular a qualidade da solução passa a ser trivial. O número de condutores de uma solução corresponde ao número de blocos da solução e o número de veículos corresponde ao máximo número de blocos num dado instante.
- Qualquer solução usando esta representação é válida. Isto facilita a implementação e performance dos algoritmos genético e *simulated annealing*. Podem-se inicializar aleatoriamente os indivíduos iniciais, o que promove a exploração global do espaço de soluções.
- No caso do algoritmo genético, não é necessário inventar ou adaptar novos operadores. Os operadores predefinidos de cruzamento a um ou dois pontos são rápidos e funcionam como esperado.
- No caso do *simulated annealing*, gerar uma nova solução vizinha é trivial. Basta selecionar uma viagem aleatória da solução atual e escolher um outro bloco que a cubra.

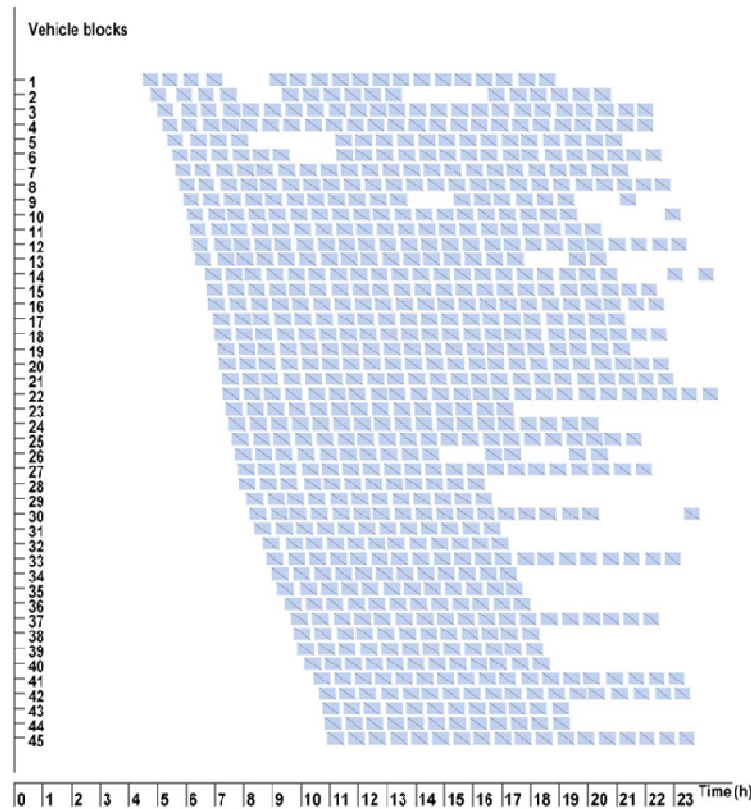


Figura 4 - Exemplo de uma solução representada por um subconjunto de blocos (Chen & Zuo 2014).

A geração dos blocos pode ser descrita sucintamente da seguinte forma:

1. Em primeiro lugar, seleciona-se um conjunto de viagens iniciais. Geralmente, só se selecionam as primeiras viagens de cada linha para este conjunto de viagens iniciais. Dependendo do número de blocos que se quer produzir, podem-se adicionar mais ou menos viagens iniciais.
2. Seguidamente, para cada uma destas viagens iniciais, geram-se todos os blocos nelas começados. A geração dos blocos pode ser vista como uma pesquisa heurística em profundidade, simulando um possível trajeto para um autocarro elétrico.
 - a. Mantém-se uma pilha de viagens, sendo que o fundo da pilha é uma viagem do depósito mais próximo ao local inicial, seguido da viagem inicial selecionada.
 - b. Depois expande-se a pilha, explorando em profundidade todas as sequência válidas de viagens, decrementando a bateria do veículo e incrementando a duração total do bloco por cada viagem adicionada ao topo da pilha.
 - c. No caso de o autocarro ficar sem bateria ou se a sequência de viagens na pilha ultrapassar o limite de duração máxima, adiciona-se uma viagem de retorno ao depósito mais próximo ao topo da pilha. A sequência de viagens da pilha pode agora ser adicionada à lista de blocos (os blocos têm que começar e acabar em depósitos).
 - d. Se a duração deste último bloco ainda não tiver ultrapassado o limite máximo e se ainda existirem mais viagens para onde se possa expandir a sequência, continua-se a exploração. Incrementar a bateria do autocarro de acordo com o tempo que ele esteve parado no depósito.

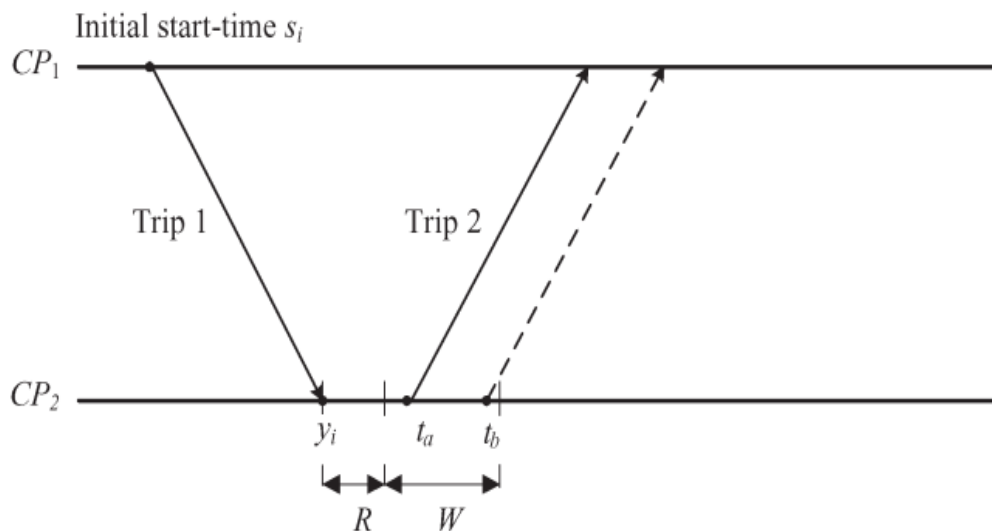


Figura 5 - Seleção da próxima viagem de um bloco (Chen & Zuo 2014).

Para evitar uma explosão combinatória no número de blocos possíveis e reduzir o espaço de soluções (que, futuramente, resultará numa convergência mais rápida para boas soluções), implementam-se simples heurísticas para impedir que o algoritmo explore sequências de viagens que venham a produzir blocos de baixa qualidade.

Na escolha da próxima viagem para a sequência, apenas se consideram viagens cujo tempo de partida se compreenda entre $[y_i + R, y_i + R + W]$, exemplificado na imagem anterior, em que y_i é o tempo de chegada da última viagem na pilha, R é o tempo de descanso mínimo, e W é o tempo de espera máximo.

Manter-se a soma dos tempos de descanso entre todas as viagens da pilha. Não expandir para certa viagem se isso fizer ultrapassar o limite máximo de descanso por bloco.

Algoritmo Genético

Após a geração dos blocos, prossegue-se para a produção de soluções. O algoritmo genético implementado segue os seguintes passos:

1. Gerar uma população inicial, $P(0)$, de indivíduos:
 - a. 40% dos indivíduos são gerados por uma cobertura do início para o fim.
 - b. 40% dos indivíduos são gerados por uma cobertura do fim para o início.
 - c. 20% dos indivíduos são gerados aleatoriamente.
2. Codificar a população $P(n)$ em cromossomas $C(n)$. Baralhar igualmente a ordem dos genes de todos os cromossomas em $C(n)$. Isto serve para atenuar o facto que diferentes viagens têm diferentes número de blocos que as cobrem. Visto que os genes representam viagens ordenadas cronologicamente, e atendendo à maneira que os blocos são gerados, os genes à esquerda do cromossoma podem tomar menos valores que os genes à direita. Este desequilíbrio ia reduzir a eficácia do posterior cruzamento de cromossomas, daí se baralhar a ordem dos genes.
3. Para todos os cromossomas em $C(n)$, com uma probabilidade p_c , seleccionar outro cromossoma aleatoriamente de $C(n)$ e realizar um cruzamento num ponto aleatório. Os dois cromossomas resultantes são adicionados ao novo conjunto $G(n)$.
4. Para todos os cromossomas em $C(n)$, fazer uma mutação a cada um dos seus genes com uma probabilidade p_m . A mutação consiste em substituir o atual bloco de um gene por outro

que cubra a mesma viagem representada pelo gene. Os cromossomos resultantes são adicionados ao novo conjunto $G(n)$.

5. Juntam-se todos os cromossomos em $C(n+1) = C(n) + G(n)$.
6. Descodificam-se todos os cromossomos em $C(n+1)$ para o conjunto $P(n)$.
7. Calcular a aptidão (*fitness*) de todas as soluções em $P(n)$. Neste caso, a aptidão é um triploto (*NrCondutores, NrVeículos, RácioUtilidade*). *RácioUtilidade* é a fração de tempo de uma solução ocupada com atividades suplementares (como descansar, esperar ou realizar viagens conetoras não agendadas). É desejável minimizar esta fração.
8. Calcular o *ranking* de todas as soluções em $P(n)$. O *ranking* é o número de outras soluções na mesma população que a dominam. Diz-se que uma solução domina outra quando todas as suas componentes são menores (maiores, no caso de maximização) ou iguais às componentes da outra e pelo menos uma das componentes é estritamente maior. Por exemplo, uma avaliação (1,1,1) domina (1,1,2), mas não domina (1,2,0) ou (1,1,1). As melhores soluções não são dominadas, tendo um *ranking* de 0.
9. As soluções não dominadas desta geração são adicionadas a um conjunto de arquivo A . Volta-se a calcular os *rankings* das soluções dentro do arquivo. Removem-se aquelas que são dominadas, ou seja, têm uma classificação maior do que 0.
10. Normalizam-se os *rankings* da população $P(n)$ e ordenam-se as soluções. Faz-se uma seleção por roleta da geração seguinte para a população $P(n+1)$.
11. Se as condições de paragem forem atingidas, acabar. Caso contrário, regressar a 2.

Exemplo

Viagens agendadas:

$$T = \{t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$$

Produção dos Blocos:

$$\begin{aligned} B_0 &= \{t_0, t_1, t_4, t_5\} \\ B_1 &= \{t_0, t_2, t_3, t_4, t_7\} \\ B_2 &= \{t_1, t_2, t_5, t_6\} \\ B_3 &= \{t_3, t_5, t_6, t_7\} \end{aligned}$$

$$\begin{aligned} B(t_0) &= \{B_0, B_1\} \\ B(t_1) &= \{B_0, B_2\} \\ B(t_2) &= \{B_1, B_2\} \\ B(t_3) &= \{B_1, B_3\} \\ B(t_4) &= \{B_0, B_1\} \\ B(t_5) &= \{B_0, B_2\} \\ B(t_6) &= \{B_2, B_3\} \\ B(t_7) &= \{B_1, B_3\} \end{aligned}$$

Produção de indivíduo com cobertura do início ao fim:

$$I_0 = \{-, -, -, -, -, -, -, -\}$$

De $B(t_0)$ escolheu-se aleatoriamente B_1 .

$$I_0 = \{B_1, -, B_1, B_1, B_1, -, -, B_1\}$$

De $B(t_1)$ escolheu-se aleatoriamente B_0 .

$$I_0 = \{B_0, B_0, B_1, B_1, B_0, B_0, -, B_1\}$$

De $B(t_6)$ escolheu-se aleatoriamente B_3 .

$$I_0 = \{B_0, B_0, B_1, B_3, B_0, B_3, B_3, B_3\}$$

Produção de indivíduo com cobertura do fim ao início:

$$I_1 = \{-, -, -, -, -, -, -, -\}$$

De $B(t_7)$ escolheu-se aleatoriamente B_1 .

$$I_1 = \{B_1, -, B_1, B_1, B_1, -, -, B_1\}$$

De $B(t_6)$ escolheu-se aleatoriamente B_3 .

$$I_1 = \{B_1, -, B_3, B_1, B_1, B_3, B_3, B_3\}$$

De $B(t_1)$ escolheu-se aleatoriamente B_2 .

$$I_1 = \{B_1, B_2, B_2, B_1, B_1, B_2, B_2, B_3\}$$

Produção de indivíduo com cobertura aleatória:

$$I_2 = \{-, -, -, -, -, -, -, -\}$$

De $\{t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$ escolheu-se aleatoriamente t_3 .

De $B(t_3)$ escolheu-se aleatoriamente B_3 .

$$I_2 = \{-, -, -, B_3, -, B_3, B_3, B_3\}$$

De $\{t_0, t_1, t_2, t_4\}$ escolheu-se aleatoriamente t_2 .

De $B(t_2)$ escolheu-se aleatoriamente B_2 .

$$I_2 = \{-, B_2, B_2, B_3, -, B_2, B_2, B_3\}$$

De $\{t_0, t_4\}$ escolheu-se aleatoriamente t_4 .

De $B(t_4)$ escolheu-se aleatoriamente B_0 .

$$I_2 = \{B_0, B_0, B_2, B_3, B_0, B_0, B_2, B_3\}$$

Baralhar igualmente os genes de todos os cromossomas:

$$I_0 = \{B_0, B_0, B_1, B_3, B_0, B_3, B_3, B_3\}$$

$$I_1 = \{B_1, B_2, B_2, B_1, B_1, B_2, B_2, B_3\}$$

$$I_2 = \{B_0, B_0, B_2, B_3, B_0, B_0, B_2, B_3\}$$

Substituir elementos por tuplos de forma a manter o mapeamento de blocos a viagens após se baralhar os genes.

$$I_0 = \{(t_0, B_0), (t_1, B_0), (t_2, B_1), (t_3, B_3), (t_4, B_0), (t_5, B_3), (t_6, B_3), (t_7, B_3)\}$$

$$I_1 = \{(t_0, B_1), (t_1, B_2), (t_2, B_2), (t_3, B_1), (t_4, B_1), (t_5, B_2), (t_6, B_2), (t_7, B_3)\}$$

$$I_2 = \{(t_0, B_0), (t_1, B_0), (t_2, B_2), (t_3, B_3), (t_4, B_0), (t_5, B_0), (t_6, B_2), (t_7, B_3)\}$$

Produzir uma nova ordem dos genes para se baralhar igualmente todos os cromossomas. É importante que todos os cromossomas sejam baralhados na mesma ordem para garantir que posteriores cruzamentos mantenham um gene por viagem.

$$shuffle(T) = \{t_7, t_1, t_4, t_6, t_2, t_0, t_5, t_3\}$$

$$I_0 = \{(t_7, B_3), (t_1, B_0), (t_4, B_0), (t_6, B_3), (t_2, B_1), (t_0, B_0), (t_5, B_3), (t_3, B_3)\}$$

$$I_1 = \{(t_7, B_3), (t_1, B_2), (t_4, B_1), (t_6, B_2), (t_2, B_2), (t_0, B_1), (t_5, B_2), (t_3, B_1)\}$$

$$I_2 = \{(t_7, B_3), (t_1, B_0), (t_4, B_0), (t_6, B_2), (t_2, B_2), (t_0, B_0), (t_5, B_0), (t_3, B_3)\}$$

Cruzamento de dois cromossomas:

$$I_1 = \{(t_7, B_3), (t_1, B_2), (t_4, B_1), (t_6, B_2), (t_2, B_2), (t_0, B_1), (t_5, B_2), (t_3, B_1)\}$$

$$I_2 = \{(t_7, B_3), (t_1, B_0), (t_4, B_0), (t_6, B_2), (t_2, B_2), (t_0, B_0), (t_5, B_0), (t_3, B_3)\}$$

Gerar um número inteiro aleatório entre 0 e $|T| - 1$.

Neste caso, obteve-se o número 5, correspondendo a um cruzamento após o 5º gene.

$$I'_1 = \{(t_7, B_3), (t_1, B_2), (t_4, B_1), (t_6, B_2), (t_2, B_2), (t_0, B_0), (t_5, B_0), (t_3, B_3)\}$$

$$I'_2 = \{(t_7, B_3), (t_1, B_0), (t_4, B_0), (t_6, B_2), (t_2, B_2), (t_0, B_1), (t_5, B_2), (t_3, B_1)\}$$

Mutação de um cromossoma:

$$I_0 = \{(t_7, B_3), (t_1, B_0), (t_4, B_0), (t_6, B_3), (t_2, B_1), (t_0, B_0), (t_5, B_3), (t_3, B_3)\}$$

Produzir $|T|$ números aleatórios.

$$prob = \{0.45, 0.03, 0.84, 0.64, 0.34, 0.32, 0.49, 0.44\}$$

Para $p_m = 0.1$, realiza-se mutação em todos os genes i tal que $prob_i < p_m$.

Neste caso, apenas se realiza a mutação dos 2º e 6º genes.

$$I_1(1) = (t_1, B_0)$$

$$I_1(5) = (t_0, B_0)$$

De $B(t_0)$ escolheu-se aleatoriamente B_1 .

De $B(t_0)$ escolheu-se aleatoriamente B_1 .

O cromossoma final fica então:

$$I'_0 = \{(t_7, B_3), (t_1, B_1), (t_4, B_0), (t_6, B_3), (t_2, B_1), (t_0, B_1), (t_5, B_3), (t_3, B_3)\}$$

Simulated Annealing

Após a geração dos blocos, prossegue-se para a produção de soluções. O algoritmo *simulated annealing* implementado segue os seguintes passos:

1. Gerar uma solução inicial, selecionando um subconjunto aleatório de blocos que cobre todas as viagens agendadas. A solução inicial torna-se no estado atual. Adiciona-se a solução inicial ao conjunto de arquivo A .
2. Produzir um vizinho aleatório do estado atual. Isto pode ser feito selecionando um bloco aleatório e incorporando-o na solução atual. Há medida que se vão adicionando novos blocos, alguns blocos deixam de pertencer à solução porque se tornam redundantes - todas as viagens que cobrem já estão a ser cobertas por outros blocos. Adicionar solução ao conjunto de arquivo.
3. Calcular a aptidão (*fitness*) da solução atual e da solução vizinha. Caso a solução atual domine a vizinha, substituir o estado atual pela a solução vizinha com probabilidade $e^{-\frac{1}{T}}$, em que T representa a temperatura atual.. Caso contrário, tornar a solução vizinha como o estado atual. Decrementa-se a temperatura. Se a condição de término ainda não foi atingida e $T \neq 0$, regressar a 2.
4. Ordenar as soluções do conjunto de arquivo por ordem crescente de ranque, remover aquelas com ranque diferente de 0. As que sobram, são as soluções não dominadas de uma iteração deste algoritmo.

Ferramentas e Tecnologias

Todo o código foi desenvolvido em Python 2.7 nativo. Não foram usadas *libraries* ou *frameworks* externas. Para controlo de versões foi usado Git. O código do projeto foi hospedado num repositório Github (<https://github.com/gibeiro/electric-bus-scheduling>). O código foi desenvolvido e testado num computador Thinkpad x220, com um processador *dual core* Intel i5 2.3Ghz de segunda geração e 8GB de memória RAM.

Capítulo 5

Testes

Metodologia

Pretende-se determinar os parâmetros do algoritmo genético que mais se adequam a este tipo de problemas. Para o efeito, foram obtidas diferentes soluções sobre o mesmo conjunto de viagens agendadas, mas alterando os parâmetros de probabilidade de cruzamento, probabilidade de mutação e tamanho da população. Um procedimento semelhante foi realizado para o algoritmo *simulated annealing*, desta vez para o parâmetro de temperatura. No final comparou-se a qualidade das soluções para diferentes parâmetros.

Uma análise interessante seria ver de que forma a autonomia dos veículos elétricos afeta a qualidade das soluções, ou seja, número de veículos e condutores necessários. Para o efeito, vão-se repetir escalonamentos, tanto por algoritmo genético como por *simulated annealing*, sobre o mesmo conjunto de viagens, mas fazendo alterar a capacidade das baterias e a sua velocidade de recarga. Como teste de referência, também é possível fazer uma escalonamento em que se considere bateria infinita, simulando uma escalonamento de autocarros normais a combustíveis fósseis. Prevê-se que quanto menor a autonomia dos veículos, maior o número de veículos necessários para servir todas as viagens.

Outra análise pertinente consiste em comparar a performance e a qualidade das soluções produzidas entre o algoritmo genético e o algoritmo *simulated annealing*. Para tal, são obtidas soluções com ambos os algoritmos sobre vários conjuntos de viagens, de forma a minimizar a variância de resultados. Depois será possível comparar a *performance* do algoritmo genético vs. algoritmo *simulated annealing* sobre os mesmos conjuntos de dados.

Pretende-se também saber de que forma a quantidade de blocos gerado influencia a qualidade das soluções produzidas. Para o efeito, sobre o mesmo conjunto de viagens são gerados vários conjuntos de viagens iniciais. Quantas mais viagens iniciais em cada conjunto, maior o número de blocos produzidos para esse conjunto (o processo de criação dos blocos é explicado na secção 4.2). Depois de gerados os vários conjuntos com diferentes números de blocos, realizam-se as mesmas escalonamentos com os mesmo parâmetros sobre todos os conjuntos e comparam-se as soluções produzidas. Quanto maior for o número de blocos disponíveis, maior é o espaço de soluções e mais lentamente se converge para as soluções, mas em contrapartida possibilitam-se novas soluções que podem ser de maior qualidade do que aquelas nos espaço de soluções com um menor número de blocos.

As soluções são apresentadas no formato de tuplos (*NrVehicles*, *NrDrivers*), representando as duas funções-objetivo escolhidas para serem minimizadas: o número de veículos e o número de condutores necessitados para cada solução.

Resultados

Tabela 1 - Soluções para 10 repetições do algoritmo genético, com 16 indivíduos e 100 gerações, com diferentes probabilidades de cruzamento e mutação por repetição.

	$p_c = 0.6$ $p_m = 0.1$	$p_c = 0.8$ $p_m = 0.05$	$p_c = 0.95$ $p_m = 0.01$	<i>Blocks</i>
$i = 0$	(11, 38)	(13, 44)	(12, 40)	847
$i = 1$	(13, 38)	(13, 42) (14, 41) (15, 40)	(14, 41) (15, 40)	1220
$i = 2$	(13, 36)	(12, 38)	(11, 37)	990
$i = 3$	(12, 41)	(12, 39) (13, 38)	(11, 40) (12, 39) (13, 37)	700
$i = 4$	(12, 38) (13, 37)	(10, 35)	(9, 32)	313
$i = 5$	(12, 38)	(13, 42)	(11, 42) (12, 40) (13, 39)	686
$i = 6$	(11, 35)	(12, 42) (14, 41)	(11, 38)	656
$i = 7$	(14, 44) (15, 43)	(12, 44) (13, 41) (14, 40)	(13, 40)	976
$i = 8$	(13, 46)	(12, 44)	(12, 45) (13, 42)	1105
$i = 9$	(13, 53) (14, 52)	(13, 53) (14, 51)	(13, 54) (14, 52)	2989
$\sum_{i=0}^9 rank(x_i)$	21	19	14	

Tabela 2 - Soluções para 10 repetições do algoritmo genético, com 16 indivíduos, probabilidade de cruzamento de 80%, probabilidade de mutação de 5% e número variável de gerações por repetição.

	50 generations	100 generations	200 generations	Blocks
$i = 0$	(12, 42) (13, 40)	(10, 41) (11, 39)	(12, 43) (13, 42)	769
$i = 1$	(12, 40)	(12, 42) (13, 40)	(12, 41) (13, 38)	633
$i = 2$	(12, 40)	(12, 38)	(11, 37)	375
$i = 3$	(14, 44)	(12, 40)	(12, 44) (13, 42)	1274
$i = 4$	(14, 44)	(14, 45) (15, 43)	(13, 43)	1479
$i = 5$	(12, 34)	(12, 38) (13, 37)	(12, 36)	555
$i = 6$	(14, 53) (15, 52)	(13, 52)	(15, 56) (16, 53)	5986
$i = 7$	(14, 42)	(13, 40)	(12, 43) (14, 40)	923
$i = 8$	(16, 50)	(13, 49)	(14, 50) (15, 49)	3028
$i = 9$	(16, 40)	(14, 38) (15, 37)	(12, 36) (13, 35)	722
$\sum_{i=0}^9 rank(x_i)$	23	17	22	

Tabela 3 - Soluções para 10 repetições do algoritmo genético, probabilidade de cruzamento de 80%, probabilidade de mutação de 5%, 50 gerações por repetição e número variável de indivíduos por repetição.

	16 <i>individuals</i>	32 <i>individuals</i>	64 <i>individuals</i>	<i>Blocks</i>
$i = 0$	(14, 44)	(14, 43) (15, 42)	(13, 43)	1932
$i = 1$	(14, 44)	(14, 47) (15, 42)	(13, 41)	1208
$i = 2$	(12, 38)	(11, 37)	(12, 38)	793
$i = 3$	(15, 40)	(13, 38)	(14, 41) (15, 40) (16, 38)	950
$i = 4$	(12, 36)	(11, 35)	(10, 34) (11, 32)	437
$i = 5$	(11, 37)	(11, 33)	(12, 36)	486
$i = 6$	(12, 41) (13, 40)	(13, 41)	(11, 36)	602
$i = 7$	(14, 44) (15, 41)	(11, 40)	(12, 40)	1095
$i = 8$	(13, 37)	(12, 40)	(12, 38)	369
$i = 9$	(11, 43) (12, 41) (14, 38)	(11, 42) (13, 41)	(11, 43) (12, 42) (13, 40)	1111
$\sum_{i=0}^9 rank(x_i)$	16	12	12	

Tabela 4 - Soluções para 10 repetições do algoritmo *simulated annealing*, com 25 iterações por repetição e parâmetro de temperatura variável.

	$T = 50$	$T = 100$	$T = 200$	<i>Blocks</i>
$i = 0$	(12, 42) (13, 39)	(12, 39)	(11, 40) (14, 39)	424
$i = 1$	(12, 41)	(14, 40)	(13, 41)	1420
$i = 2$	(12, 35)	(13, 32)	(12, 34)	471
$i = 3$	(13, 39)	(14, 38)	(13, 41) (15, 40)	522
$i = 4$	(12, 44) (13, 43)	(12, 46) (13, 41)	(12, 40)	613
$i = 5$	(13, 49) (15, 48)	(13, 47)	(14, 51) (15, 50)	1924
$i = 6$	(13, 41) (14, 37)	(14, 38)	(14, 39) (15, 38) (16, 37)	632
$i = 7$	(14, 41)	(13, 41)	(14, 43) (15, 41)	1354
$i = 8$	(14, 46)	(14, 46)	(13, 46) (14, 45)	1630
$i = 9$	(11, 40) (12, 38)	(12, 41)	(12, 41) (13, 40)	529
$\sum_{i=0}^9 rank(x_i)$	14	9	25	

Tabela 5 - Soluções para 10 repetições do algoritmo *simulated annealing*, com parâmetro de temperatura igual a 100 e número de iterações variável.

	<i>25 iterations</i>	<i>50 iterations</i>	<i>100 iterations</i>	<i>Blocks</i>
$i = 0$	(12, 41)	(13, 40)	(12, 44) (13, 41) (15, 39)	1183
$i = 1$	(14, 40) (15, 38)	(13, 39) (14, 37)	(13, 42) (14, 37)	812
$i = 2$	(13, 44) (14, 43) (15, 41)	(12, 39)	(13, 39)	814
$i = 3$	(14, 44) (15, 43)	(13, 42) (16, 41)	(13, 43) (14, 42)	1540
$i = 4$	(13, 40)	(13, 41)	(13, 42) (14, 40)	903
$i = 5$	(13, 45) (14, 44)	(14, 47) (15, 46)	(12, 41)	2194
$i = 6$	(13, 40) (14, 38)	(13, 36)	(12, 35)	631
$i = 7$	(13, 46) (16, 45)	(13, 43)	(12, 44) (13, 42)	2004
$i = 8$	(12, 43) (13, 40)	(13, 41) (15, 40)	(12, 38)	1217
$i = 9$	(13, 40)	(11, 40)	(12, 41) (13, 39)	641
$\sum_{i=0}^9 rank(x_i)$	33	13	12	

Tabela 6 - Soluções para 10 repetições do algoritmo genético, probabilidade de cruzamento de 80%, probabilidade de mutação de 5%, 100 gerações e 16 indivíduos por repetição e capacidade de bateria dos autocarros variável.

	$b_{max} = 120'$	$b_{max} = 240'$	$b_{max} = 480'$	$b_{max} = 1440'$
$i = 0$	(13, 41)	(10, 33) (12, 31)	(11, 34) (12, 33) (13, 32)	(10, 31) (11, 29)
$i = 1$	(13, 37)	(11, 30) (12, 29)	(11, 26)	(10, 26)
$i = 2$	(12, 37)	(12, 28)	(11, 25)	(10, 22)
$i = 3$	(13, 42) (14, 41)	(12, 35)	(11, 31) (12, 30)	(10, 31)
$i = 4$	(14, 43)	(11, 38)	(10, 34)	(12, 37)
$i = 5$	(12, 37)	(10, 28) (11, 27)	(11, 27)	(10, 27)
$i = 6$	(13, 44) (14, 43) (13, 44) (14, 43)	(12, 38) (13, 37) (14, 36)	(11, 34)	(12, 35)
$i = 7$	(13, 42) (16, 41)	(11, 34)	(11, 31)	(11, 34) (12, 32)
$i = 8$	(13, 44) (14, 41)	(13, 33)	(12, 32)	(12, 33) (14, 32)
$i = 9$	(14, 44)	(13, 37)	(10, 32)	(11, 36)
$\sum_{i=0}^9 rank(x_i)$	77	21	15	10

Tabela 7 - Soluções para 10 repetições de *simulated annealing*, com parâmetro de temperatura igual a 100, 25 iterações por repetição e número variável de capacidade de bateria dos autocarros.

	$b_{max} = 120'$	$b_{max} = 240'$	$b_{max} = 480'$	$b_{max} = 1440'$
$i = 0$	(12, 38)	(10, 31)	(11, 33) (12, 31) (13, 29)	(10, 32) (11, 31)
$i = 1$	(14, 44)	(12, 39) (13, 36) (14, 34)	(12, 37) (13, 36)	(12, 34)
$i = 2$	(13, 37)	(12, 32) (16, 31)	(12, 28)	(11, 33) (12, 29)
$i = 3$	(13, 45) (14, 42)	(11, 37) (12, 36) (13, 35)	(11, 38) (12, 34)	(11, 33)
$i = 4$	(14, 46) (15, 43) (16, 42)	(12, 32) (13, 30)	(11, 30) (13, 29)	(12, 28)
$i = 5$	(12, 35)	(11, 26)	(10, 22)	(9, 23) (10, 22) (11, 21)
$i = 6$	(13, 45) (14, 44)	(13, 37) (14, 36)	(12, 32)	(12, 33)
$i = 7$	(14, 47) (16, 44)	(12, 34)	(11, 37) (12, 35)	(11, 40) (13, 34)
$i = 8$	(13, 38) (14, 37)	(11, 27) (13, 26)	(12, 25)	(12, 25)
$i = 9$	(13, 48)	(13, 43)	(12, 42)	(14, 47) (17, 46)
$\sum_{i=0}^9 rank(x_i)$	84	29	13	13

Tabela 8 - Soluções para 10 repetições de algoritmo genético vs. *simulated annealing*, para um conjunto de viagens correspondente a 16h de trabalho numa única linha de autocarros.

	<i>Genetic Algorithm</i>	<i>Simulated Annealing</i>	<i>Blocks</i>
$i = 0$	(2, 9)	(2, 8)	16
$i = 1$	(2, 9)	(2, 8)	16
$i = 2$	(2, 8)	(2, 8)	16
$i = 3$	(1, 7)	(1, 7)	14
$i = 4$	(2, 9)	(2, 8)	16
$i = 5$	(2, 10)	(2, 10)	18
$i = 6$	(2, 8)	(2, 8)	16
$i = 7$	(2, 9)	(2, 8)	16
$i = 8$	(3, 9)	(3, 9)	14
$i = 9$	(2, 9)	(2, 9)	24
$\sum_{i=0}^9 rank(x_i)$	4	0	

Tabela 9 - Soluções para 10 repetições de algoritmo genético vs. *simulated annealing*, para um conjunto de viagens correspondente a 16h de trabalho numa rede de autocarros com 4 linhas.

	<i>Genetic Algorithm</i>	<i>Simulated Annealing</i>	<i>Blocks</i>
$i = 0$	(13, 49) (14, 45)	(13, 44)	1205
$i = 1$	(12, 39)	(13, 40)	1387
$i = 2$	(12, 38)	(11, 41) (12, 40) (13, 38)	466
$i = 3$	(14, 46) (15, 45)	(13, 42)	2206
$i = 4$	(11, 38)	(12, 37)	413
$i = 5$	(12, 37)	(14, 43)	1624
$i = 6$	(12, 40)	(11, 39)	768
$i = 7$	(12, 38)	(13, 45) (14, 44) (15, 41)	696
$i = 8$	(13, 40)	(13, 39)	808
$i = 9$	(15, 38)	(14, 43) (15, 40)	1256
$\sum_{i=0}^9 rank(x_i)$	6	8	

Tabela 10 - Soluções para 10 repetições de algoritmo genético com diferentes métodos de produção de blocos.

	$Blocks_{regular}$	$Solutions$	$Elapsed (s)$	$Blocks_{pedantic}$	$Solutions$	$Elapsed (s)$
$i = 0$	592	(18, 52) (21, 50)	2.8851630	3164	(13, 43) (14, 42)	4.6964600
$i = 1$	150	(13, 38)	2.2345540	517	(11, 37) (12, 36)	3.0144269
$i = 2$	153	(15, 38)	2.0658230	632	(13, 40) (14, 39)	3.1343500
$i = 3$	198	(14, 41) (15, 38)	2.2536840	1024	(12, 38)	3.4733150
$i = 4$	148	(12, 41) (13, 40)	2.1349360	483	(13, 38)	2.9966721
$i = 5$	256	(11, 36)	2.2435250	1018	(14, 33)	3.2956571
$i = 6$	179	(16, 43)	2.3452150	742	(11, 40)	3.4818332
$i = 7$	142	(12, 37)	2.2106540	580	(12, 38)	3.2081089
$i = 8$	220	(13, 38)	2.3133480	717	(11, 36)	3.2699511
$i = 9$	188	(12, 46)	2.1425070	1195	(13, 45) (14, 44)	3.591818
$\sum_{i=0}^9 rank(x_i)$		11			1	

Tabela 11 - Soluções para 10 repetições de *simulated annealing* com diferentes métodos de produção de blocos.

	$Blocks_{regular}$	$Solutions$	$Elapsed (s)$	$Blocks_{pedantic}$	$Solutions$	$Elapsed (s)$
$i = 0$	247	(13, 40)	2.6046118	1082	(12, 42) (13, 40)	2.9032061
$i = 1$	121	(13, 41) (14, 40) (15, 38)	2.2035419	362	(12, 41) (14, 39)	2.7138051
$i = 2$	147	(12, 39) (13, 38)	2.9680211	770	(12, 42) (13, 41) (16, 40)	3.2732059
$i = 3$	290	(18, 52) (19, 49)	2.9975910	1639	(15, 45) (18, 43)	3.4834899
$i = 4$	232	(17, 48) (18, 47) (19, 46)	2.7424919	1774	(13, 47) (14, 42)	4.4207229
$i = 5$	794	(14, 47) (15, 46)	3.2409329	6053	(15, 49) (16, 47)	4.0458669
$i = 6$	233	(16, 40)	3.8730838	847	(13, 40) (15, 38)	3.5553758
$i = 7$	240	(14, 40)	3.2352690	1026	(13, 40)	3.3782150
$i = 8$	231	(13, 41) (14, 40)	3.1135191	524	(11, 35)	3.0032510
$i = 9$	391	(13, 42) (14, 41)	3.3024759	1477	(13, 40) (14, 39)	3.5656468
$\sum_{i=0}^9 rank(x_i)$		18			11	

Análise

- Atendendo à Tabela 1, a combinação de parâmetros $p_c = 0.95$ e $p_m = 0.01$ obteve marginalmente melhores resultados que as restantes combinações sobre os mesmos conjuntos de dados para cada iteração. A performance pode ser quantificada calculando os ranques de cada solução por linha, e depois somando os todos os ranques por coluna. O ranque, como já foi explicado na secção 4.3, representa, para uma dada solução, o número de outras soluções que a dominam. Quanto menor for o somatório dos ranques, melhor a qualidade das soluções em geral.
- Atendendo à Tabela 2, o uso de 100 gerações no algoritmo genético apresenta soluções em média melhores que outros valores testados.
- Os resultados da Tabela 3 apresentam um empate entre o uso de 32 ou 64 indivíduos por geração no algoritmo genético.
- Uma análise à Tabela 4 permite concluir que o parâmetro $T = 100$ apresenta os melhores resultados em simulated annealing.
- Atendendo à Tabela 5, verifica-se que 50 e 100 repetições por iteração de simulated annealing apresentam resultados idênticos em termos de qualidade. No entanto, 25 repetições é consideravelmente pior.
- A Tabela 6 e a Tabela 7 confirmam a hipótese já considerada na secção de metodologia 5.1. Quanto maior a autonomia de cada autocarro, melhor a qualidade das soluções de escalonamento. Estes resultados evidenciam também a utilidade destes algoritmos como um simulador de custos para qualquer empresa de transportes que queira adoptar autocarros eléctricos, porque permite ver em antemão o tamanho da frota e o número de condutores necessários.
- O resultado da Tabela 8 permite concluir que o algoritmo de simulated annealing é marginalmente melhor que o algoritmo genético em casos de escalonamento de linhas singulares. No entanto a diferença é pouca. Mais iterações seriam necessárias para se obter dados mais conclusivos.
- A Tabela 9 demonstra que ambos os algoritmos genético e simulated annealing têm performances equiparáveis para problemas de escalonamento de média e grande escala.
- A Tabela 10 e a Tabela 11 permitem concluir que o algoritmo genético é mais sensível ao tamanho do espaço de soluções que simulated annealing. Em geral, uma produção de blocos exaustiva (que considera mais tempos iniciais para começar blocos) resulta numa melhor qualidade de soluções, à custa de tempo computacional. No entanto, o algoritmo da produção de blocos é de complexidade espacial e temporal exponenciais. Em casos de maior dimensões, talvez não seja possível considerar tantos tempos iniciais na criação de blocos.

Capítulo 6

Conclusões

Resumo do Documento

No Capítulo 1 foi estabelecida a motivação por detrás desta dissertação. Será realizada no âmbito de resolver novos problemas que surgem no setor de transportes urbanos com a adopção cada vez mais prevalente de autocarros eléctricos, devido a iniciativas governamentais. Foram expostas as diferentes fases de criação de uma rede de autocarros urbana, entre as quais a fase da escalonamento dos veículos. Foram também definidos os objetivos desta dissertação, sendo o principal objetivo produzir uma nova solução de escalonamento de autocarros eléctricos que possa ser utilizada em instâncias reais.

No Capítulo 2 reúne-se a pesquisa relativa a problemas de escalonamento de veículos. Após uma breve introdução sobre o que são este tipo de problemas, e feita a distinção entre as suas variantes de depósito único e múltiplos depósitos, são expostos vários artigos que compõem a base bibliográfica desta dissertação. Uma breve análise bibliográfica permitiu concluir que são um tipo de problemas bastante estudados, com vários algoritmos publicadas. No entanto, estes algoritmos não tomam em consideração as restrições adicionais dos veículos eléctricos, não sendo compatíveis com a escalonamento de autocarros eléctricos sem primeiro serem adaptados. São também analisados os algoritmos do estado da arte do problema de escalonamento de autocarros eléctricos, incluindo k-Greedy Algorithm(Paul & Yamada 2014), Modified Column Generation(Posthoorn 2016) e Branch-and-Price Algorithm(Adler & Mirchandani 2017). Este tipo de problemas são uma combinação de um problema VSP com o problema de escalonar a recarga das baterias dos veículos. A análise bibliográfica permitiu concluir que é um problema bastante recente, existindo poucas soluções completas publicadas. No entanto, já existe bastante literatura que aborda exclusivamente o escalonamento das recargas.

No Capítulo 3 é formalizado teoricamente o problema central da dissertação. Em vez de uma formalização como um Problema Inteiro, optou-se por uma formalização de mais alto nível, visto que se adequa mais ao tipo de abordagens heurísticas desenvolvidas, a serem expostas no capítulo seguinte. É também apresentada uma formalização alternativa em blocos de viagens, baseada no trabalho desenvolvido em (Chen & Zuo 2014), que acabou por ser utilizada para a implementação das soluções. São também formalizados os diferentes objetivos de otimização possíveis: minimização da frota, minimização de empregados e minimização do tempo de trabalho não útil (por exemplo, tempo despendido em viagens de conexão não agendadas).

No Capítulo 4 são explicadas a escolha e implementação das soluções metaheurísticas desenvolvidas para esta dissertação. A decisão em fazer um algoritmo genético e outro baseado em simulated annealing é fundamentada no facto de ainda não existirem soluções para escalonamento de autocarros eléctricos

baseadas em metaheurísticas. Decidiu-se que ambas as soluções iam ser divididas em duas fases: uma primeira fase para gerar blocos de viagens válidos e uma segunda fase onde se calculam heurísticamente subconjuntos destes blocos para formar soluções de escalonamento. A geração dos blocos assemelha-se a uma heurística de pesquisa em profundidade - vão-se gerando sequências de viagens válidas tendo em conta várias simplificações impostas para impedir uma explosão combinatória do número de blocos possíveis. A segunda parte consiste em utilizar uma metaheurística para explorar o espaço de soluções (neste caso, conjuntos de blocos). Ambas as implementações são multiobjetivo, gerando conjuntos de soluções não dominadas em vez de combinar todos os objetivos num só, atribuindo pesos. O algoritmo genético usa uma codificação em que o valor de cada gene representa o bloco que está a cobrir uma dada viagem, havendo um gene para cada viagem. O operador de mutação sobre um gene escolhe um novo bloco de cobre a viagem. O operador de cruzamento é um cruzamento em ponto único. A seleção é feita pelo método de roleta. Como as diferentes funções-objetivo não são combinadas num só único valor, usa-se o ranque de cada solução (número de outras soluções que a dominam) para se ordenar as soluções no processo de seleção. O algoritmo simulated annealing diverge muito pouco da implementação padrão. Apenas foi necessário alterar a fórmula de calcular a probabilidade de transição em função da fitness para ser agora em função do ranque da solução em relação à solução vizinha gerada a cada iteração.

No Capítulo 5 propõe-se uma metodologia para testar as soluções desenvolvidas. São apresentados e interpretados os resultados obtidos. Testou-se a melhor combinação de de parâmetros para cada algoritmo implementado. Demonstrou-se empiricamente que a qualidade das soluções de escalonamento decresce o quanto menor for a autonomia dos autocarros. Ambos os algoritmos apresentaram uma qualidade de soluções semelhante para, aproximadamente, o mesmo tempo computacional disponível. Verificou-se que a qualidade das soluções de ambos os algoritmos aumenta com a quantidade de blocos gerados inicialmente. Verificou-se também que a qualidade das soluções do algoritmo genético dependem mais do tamanho do espaço das soluções que no algoritmo simulated annealing.

Por fim, espera-se que esta dissertação contribua de alguma forma para o atual estado da arte dos algoritmos de escalonamento de autocarros elétricos. Pretende-se que as soluções implementadas tenham valor suficiente para poderem ser posteriormente desenvolvida e servirem de base para eventuais aplicações em casos reais.

Trabalho Futuro e Melhorias

- Devido ao hardware disponível, não foi possível fazer testes de grande dimensão ou stress tests. Seria pertinente testar estes algoritmos em casos extremos.
- Não foi possível obter dados reais de escalonamento de veículo elétricos. Inicialmente decidiu-se utilizar casos de escalonamento de autocarros normais e, virtualmente, adicionar restrições de autocarros elétricos. No entanto, optou-se por um gerador aleatório de redes de autocarros realistas, parametrizado com número de paragens, número de linha, número de depósitos e horário de serviço da rede. Isto permitiu testes com várias instâncias diferentes de redes, o que de certa forma permite combater a variância implícita em soluções metaheurísticas para se obterem resultados mais conclusivos.
- Vários aspectos da escalonamento de veículos elétricos foram simplificados ou até mesmo omitidos. Não foram consideradas baterias substituíveis. Foi considerado que todos os veículos tinham exatamente a mesma capacidade de bateria máxima, o que na realidade pode não ser verdade - uma frota de autocarros pode ser constituída por modelos de veículo diferentes. Não foi simulado o preço da energia elétrica ao longo do dia. Considerou-se que a velocidade de recarga dos depósitos era constante e igual para todos os depósitos, o que, numa solução real, dificilmente seria verdade.

- Alguns aspetos de vehicle scheduling também foram bastante simplificados. As rotas foram reduzidas a simples intervalos de tempo de viagem. Num caso real, estes intervalos de viagem variam ao longo do dia e da semana, dependendo das condições de tráfego. Altitude também não foi considerada.

Referências

- Aabrandt, A. et al., 2012. Prediction and optimization methods for electric vehicle charging schedules in the EDISON project. In *2012 IEEE PES Innovative Smart Grid Technologies (ISGT)*. Available at: <http://dx.doi.org/10.1109/isgt.2012.6175718>.
- Adler, J.D. & Mirchandani, P.B., 2017. The Vehicle Scheduling Problem for Fleets with Alternative-Fuel Vehicles. *Transportation Science*, 51(2), pp.441–456.
- Bertossi, A.A., Carraresi, P. & Gallo, G., 1987. On some matching problems arising in vehicle scheduling models. *Networks. An International Journal*, 17(3), pp.271–281.
- Bodin, L., 1983. Solving large vehicle routing and scheduling problems in small core. In *Proceedings of the 1983 annual conference on Computers : Extending the human resource - ACM 83*. Available at: <http://dx.doi.org/10.1145/800173.809693>.
- Bodin, L. & Golden, B., 1981. Classification in vehicle routing and scheduling. *Networks. An International Journal*, 11(2), pp.97–108.
- Brandão, J. & Mercer, A., 1997. A tabu search algorithm for the multi-trip vehicle routing and scheduling problem. *European Journal of Operational Research*, 100(1), pp.180–191. Available at: [http://dx.doi.org/10.1016/s0377-2217\(97\)00010-6](http://dx.doi.org/10.1016/s0377-2217(97)00010-6).
- Bunte, S. & Kliewer, N., 2009. An overview on vehicle scheduling models. *Public Transport*, 1(4), pp.299–317.
- Carpaneto, G. et al., 1989. A branch and bound algorithm for the multiple depot vehicle scheduling problem. *Networks*, 19(5), pp.531–548. Available at: <http://dx.doi.org/10.1002/net.3230190505>.
- Ceder, A., 2002. Designing Public Transport Networks and Routes. In *Advanced Modeling for Transit Operations and Service Planning*. pp. 59–91.
- Ceselli, A., Righini, G. & Salani, M., 2009. A Column Generation Algorithm for a Rich Vehicle-Routing Problem. *Transportation Science*, 43(1), pp.56–69.
- Chen, C. & Zuo, X., 2014. A multi-objective genetic algorithm based bus vehicle scheduling approach. In *The 26th Chinese Control and Decision Conference (2014 CCDC)*. Available at: <http://dx.doi.org/10.1109/ccdc.2014.6852625>.
- Daduna, J.R. & Pinto Paixão, J.M., 1995. Vehicle Scheduling for Public Mass Transit — An Overview. In *Lecture Notes in Economics and Mathematical Systems*. pp. 76–90.
- Desaulniers, G. & Hickman, M.D., 2007. Chapter 2 Public Transit. In *Handbooks in Operations Research and Management Science*. pp. 69–127.
- Desaulniers, G., Lavigne, J. & Soumis, F., 1998. Multi-depot vehicle scheduling problems with time windows and waiting costs. *European journal of operational research*, 111(3), pp.479–494.

- Desrosiers, J. et al., 1995. Chapter 2 Time constrained routing and scheduling. In *Handbooks in Operations Research and Management Science*. pp. 35–139.
- Faiz, T.I., Vogiatzis, C. & Noor-E-Alam, M., 2019. A column generation algorithm for vehicle scheduling and routing problems. *Computers & Industrial Engineering*, 130, pp.222–236. Available at: <http://dx.doi.org/10.1016/j.cie.2019.02.032>.
- Guedes, P.C. & Borenstein, D., 2015. Column generation based heuristic framework for the multiple-depot vehicle type scheduling problem. *Computers & Industrial Engineering*, 90, pp.361–370. Available at: <http://dx.doi.org/10.1016/j.cie.2015.10.004>.
- Hadjar, A., Marcotte, O. & Soumis, F., 2006. A Branch-and-Cut Algorithm for the Multiple Depot Vehicle Scheduling Problem. *Operations research*, 54(1), pp.130–149.
- Haghani, A., Banihashemi, M. & Chiang, K.-H., 2003. A comparative analysis of bus transit vehicle scheduling models. *Transportation Research Part B: Methodological*, 37(4), pp.301–322.
- He, Y., Venkatesh, B. & Guan, L., 2012. Optimal Scheduling for Charging and Discharging of Electric Vehicles. *IEEE transactions on smart grid*, 3(3), pp.1095–1105.
- Lan, T. et al., 2012. Optimal control of an electric vehicle's charging schedule under electricity markets. *Neural computing & applications*, 23(7-8), pp.1865–1872.
- Malmberg, C.J., 1996. A genetic algorithm for service level based vehicle scheduling. *European Journal of Operational Research*, 93(1), pp.121–134. Available at: [http://dx.doi.org/10.1016/0377-2217\(95\)00185-9](http://dx.doi.org/10.1016/0377-2217(95)00185-9).
- Park, Y.-B., 2001. A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines. *International Journal of Production Economics*, 73(2), pp.175–188. Available at: [http://dx.doi.org/10.1016/S0925-5273\(00\)00174-2](http://dx.doi.org/10.1016/S0925-5273(00)00174-2).
- Paul, T. & Yamada, H., 2014. Operation and charging scheduling of electric buses in a city bus route network. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. pp. 2780–2786.
- Pepin, A.-S. et al., 2008. A comparison of five heuristics for the multiple depot vehicle scheduling problem. *Journal of Scheduling*, 12(1), pp.17–30.
- Phan, H.T.M., 2017. A simulated annealing algorithm for vehicle scheduling problem. *Science & Technology Development Journal - Economics - Law and Management*, 1(Q4), pp.70–77. Available at: <http://dx.doi.org/10.32508/stdjelm.v1i4q4.477>.
- Posthoorn, C., 2016. *Vehicle Scheduling of Electric City Buses: A Column Generation Approach*. Applied Mathematics. TUDelft.
- Wang, H. & Shen, J., 2007. Heuristic approaches for solving transit vehicle scheduling problem with route and fueling time constraints. *Applied mathematics and computation*, 190(2), pp.1237–1249.
- Wang, Q., 2014. Research on Application of a Type of Improved Simulated Annealing Algorithm for the Transit Vehicle Scheduling. *Journal of Information and Computational Science*, 11(3), pp.867–874. Available at: <http://dx.doi.org/10.12733/jics20102548>.
- Watanabe, M., Ida, K. & Gen, M., 2005. A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem. *Computers & Industrial Engineering*, 48(4), pp.743–752. Available at: <http://dx.doi.org/10.1016/j.cie.2004.12.008>.
- Wu, X. et al., 2013. The influence factor analysis of energy consumption on all electric range of electric city bus in China. In *2013 World Electric Vehicle Symposium and Exhibition (EVS27)*. Available at: <http://dx.doi.org/10.1109/evs.2013.6914850>.
- Xia, Y. & Yi, C., 2014. Optimization Vehicle Scheduling Based on Simulated Annealing Genetic

Algorithm. *Advanced Materials Research*, 998-999, pp.1052–1057. Available at: <http://dx.doi.org/10.4028/www.scientific.net/amr.998-999.1052>.

Yamada, T., 1995. A genetic algorithm with multi-step crossover for job-shop scheduling problems. *1st International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA)*. Available at: <http://dx.doi.org/10.1049/cp:19951040>.

Zhou, Y. et al., 2017. A multi-objective evolutionary algorithm for multi-period dynamic emergency resource scheduling problems. *Transportation Research Part E: Logistics and Transportation Review*, 99, pp.77–95. Available at: <http://dx.doi.org/10.1016/j.tre.2016.12.011>.