EDILSON COSTA DE CASTRO

PRÉ-PROCESSAMENTO DO PROBLEMA DE COBERTURA DE CONJUNTOS APLICADO AO ESCALONAMENTO DE CONDUTORES

MARINGÁ

2006

EDILSON COSTA DE CASTRO

PRÉ-PROCESSAMENTO DO PROBLEMA DE COBERTURA DE CONJUNTOS APLICADO AO ESCALONAMENTO DE CONDUTORES

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual de Maringá, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Ademir Aparecido Constantino

MARINGÁ

2006

Dados Internacionais de Catalogação-na-Publicação (CIP)
(Biblioteca Central - UEM, Maringá – PR., Brasil)

C355p

Castro, Edilson Costa de
Pré-processamento do problema de cobertura de conjuntos aplicado ao escalonamento de condutores / Edilson Costa de Castro. -- Maringá : [s.n.], 2006.
56 f. : il., figs., tabs.

Orientador : Prof. Dr. Ademir Aparecido Constantino.
Dissertação (mestrado) - Universidade Estadual de Maringá. Programa de Pós-Graduação em Ciência da Computação, 2006.

1. Otimização. 2. Algoritmos heurísticos. 3. Escalonamento de condutores. I. Universidade Estadual de Maringá. Programa de Pós-Graduação em Computação.

CDD 21.ed.003

EDILSON COSTA DE CASTRO

PRÉ-PROCESSAMENTO DO PROBLEMA DE COBERTURA DE CONJUNTOS APLICADO AO ESCALONAMENTO DE CONDUTORES

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual de Maringá, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Aprovado em 10/11/2006.

BANCA EXAMINADORA

Prof. Dr. Ademir Aparecido Constantino
Universidade Estadual de Maringá – DIN/UEM

Prof. Dr. Sérgio Roberto Pereira da Silva
Universidade Estadual de Maringá – DIN/UEM

Prof. Dr. Celso Carnieri
Universidade Federal do Paraná – DMAT/UFPR

Dedico este trabalho
a todas as pessoas que, através dos tempos,
em qualquer lugar do mundo,
por curiosidade ou necessidade,
durante um segundo ou por toda a vida,
por meio da arte, ciência ou filosofia,
buscaram responder às perguntas fundamentais:

"De onde viemos?",
"Onde estamos?",
"Para onde vamos?"

...
..."E POR QUÊ?"

Agradecimentos

Ao meu orientador, Ademir Aparecido Constantino, pela confiança, dedicação e paciência.

À minha psicóloga Dra. Alessandra Herranz, por me ajudar a retomar o caminho da vida e, entre outras coisas, tornar possível a conclusão deste trabalho.

À minha amiga Letícia R. Bueno, por todos os valiosos auxílios prestados durante esse Mestrado.

À todos os meus amigos da Benner Saúde, pelo companheirismo e bom humor, fazendo mais divertida a difícil tarefa de levar trabalho e estudos em paralelo.

À vocês, muito obrigado.

"Faze o que tu queres,
há de ser tudo da Lei

Faze isso e ninguém ouse diga "não"
Pois não existe deus senão o Homem

Todo homem tem direito de viver a não ser pela sua própria lei
Da maneira que ele quer viver
De trabalhar como quiser e quando quiser
De brincar como ele quiser

Todo homem tem direito de descansar como quiser
De morrer quando quiser

O homem tem direito de amar como ele quiser
De beber o que ele quiser
De viver aonde quiser
De mover-se pela face do planeta livremente sem passaportes
Porque o planeta é dele,
o planeta é nosso

O homem tem direito de pensar o que ele quiser
De escrever o que ele quiser
De desenhar, de pintar, de cantar, de compor
O que ele quiser

Todo homem tem direito de vestir-se da maneira que ele quiser

O homem tem direito de amar como ele quiser
Tomai a vossa sede amor como quiseres e com quem quiseres
Há de ser tudo da Lei

E o homem tem direito de matar todos aqueles que contrariarem esses direitos
O Amor é a Lei, mas Amor sob Vontade

Os escravos servirão
Viva a Sociedade Alternativa.

VIVA! VIVA!"

Raul Seixas - A Lei

Resumo

O problema de escalonamento de condutores (PEC) consiste em distribuir de maneira eficiente o quadro de viagens de uma empresa de transporte coletivo entre os condutores disponíveis. Este problema é comumente modelado como um problema de cobertura de conjunto - PCC (*set covering problem - SCP*). Neste caso, um bom resultado para o PEC depende de uma boa formulação e resolução do PCC; porém, a maior parte da bibliografia relacionada trata apenas da resolução das instâncias do PCC, sem preocupação com a geração das mesmas. Este trabalho investiga e propõe metodologias heurísticas baseadas em *Simulated Annealing* para a geração de instâncias do PCC, cujas características possibilitem a algoritmos de resolução obterem melhores resultados. Nos testes efetuados, conseguiu-se uma redução de até 8% no custo das soluções apresentadas, em comparação com a resolução de instâncias geradas por um método mais simples baseado no algoritmo de Dijkstra.

Palavras-chave: Escalonamento de condutores, cobertura de conjunto, algoritmos heurísticos, *Simulated Annealing*.

Abstract

The crew scheduling problem consists on efficiently distribute trips of an urban transport company to available drivers. It's used to be modelled as a set covering problem (SCP). Most of works in this area propose only SCP resolution methods, but do not include the SCP generation in their matters. This thesis investigates and proposes heuristic methodologies based on Simulated Annealing to generate SCP instances, whose characteristics allow resolution algorithms to obtain better results. In executed tests, costs were reduced up to 8%, in comparison with a Dijkstra's algorithm based method of SCP generation.

Keywords: crew scheduling, set covering, heuristic algorithms, Simulated Annealing.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

1.1 Ambientação

Nesta seção, serão apresentadas algumas informações a fim de apresentar e delimitar o escopo a que pertence este trabalho.

1.1.1 Transporte urbano rodoviário

O transporte urbano rodoviário de passageiros é a modalidade de transporte rodoviário utilizado para a movimentação da população nos centros urbanos da cidades, periferia e regiões metropolitanas; caracteriza-se por apresentar linhas regulares de curta distância, com horários e itinerários bem definidos e de grande freqüência. Nesta modalidade de transporte, um veículo pode chegar a realizar até 40 viagens por dia. Geralmente, é um serviço prestado por uma empresa privada, que dispõe de um determinado número de ônibus e condutores (geralmente acompanhados de cobradores) para cumprir o seu quadro de viagens.

1.1.2 Escalonamento de veículos e condutores

O gerenciamento de uma empresa de transporte rodoviário de passageiros envolve uma variedade de problemas de decisão, que podem ser divididos em três níveis hierárquicos: o estratégico, o tático e o operacional. A nível estratégico podem ser citadas decisões como a determinação da região de atendimento, a implantação de novas linhas, a compra de outras empresas etc., enquanto as decisões a nível tático estão relacionadas com a aquisição de novos veículos e com a política de contratação de pessoal.

A nível operacional surgem problemas como a determinação dos roteiros a serem executados pelos veículos e o planejamento de escala dos veículos e dos motoristas. Este último problema é denominado pela literatura inglesa como *'vehicle and crew scheduling'*.

A resolução do problema de planejamento de escala de veículos e condutores consiste em determinar um plano de distribuição de viagens sobre um conjunto de veículos e motoristas envolvidos na operação, ao longo de um período de planejamento. Como resultado deste processo, para cada veículo é alocado um conjunto de viagens a ser realizado, e cada viagem é alocada a um condutor. Além disso, cada condutor receberá uma escala de trabalho ao longo do período de planejamento de forma que a carga de trabalho seja balanceada. A distribuição do trabalho leva em consideração as horas noturnas, as horas

extras, os dias de folgas, as horas efetivas de trabalho, etc. Todo este processo de alocação deve satisfazer as restrições da legislação trabalhista, regras internas da empresa e os acordos firmados entre a empresa e o sindicato da classe dos motoristas.

O planejamento de escala de veículos e condutores responde a uma série de questões que são comuns em uma empresa de transporte rodoviário de passageiros, por exemplo:

- quantos veículos são necessários;
- quais as viagens que devem ser realizadas por veículo;
- quando e onde cada veículo deve abastecer;
- quantos condutores são necessários;
- quando começa e termina cada turno de trabalho dos condutores;
- quais os dias de trabalho e os dias de folga de cada condutor;
- quais as viagens que cada condutor deve cumprir para cada dia do período de planejamento.

1.1.3 Escalonamento computadorizado

Em boa parte das empresas de transporte coletivo da Europa e dos Estados Unidos, já há algum tempo são utilizados sistemas computacionais para a geração de escalas de veículos e condutores. Muitos destes sistemas foram desenvolvidos em trabalhos de pesquisa acadêmica, por exemplo: HASTUS [16], HOT [9] e TRACS II [26].

Efetuar o escalonamento de veículos e condutores sem a ajuda de computadores não é uma tarefa impossível. Porém, a preferência pela utilização de sistemas computacionais para essa tarefa se deve às grandes vantagens sobre o escalonamento manual, a começar pelo tempo: um sistema computacional leva no máximo uma ou duas horas para realizar o mesmo trabalho que, manualmente, pode levar dias ou semanas. Considere-se ainda que esta redução no tempo necessário para o processo de escalonamento permite à empresa a realização de testes, observando os efeitos de cada opção sempre que uma decisão deva ser tomada: uma nova linha, novos ônibus, mudanças em regras da empresa ou nas leis, etc.

Além disso, um sistema computacional oferece a possibilidade de se adotar técnicas de otimização com o objetivo de reduzir custos operacionais. Estas reduções apresentam-se tanto em termos mensuráveis (períodos ociosos pagos aos condutores, deslocamentos improdutivos de veículos, horas-extras, quantidade de veículos/condutores utilizados, etc) como não mensuráveis a curto prazo (multas por descumprimento de leis trabalhistas, melhorias no atendimento em conseqüência da melhor alocação dos recursos, etc.). As vantagens do escalonamento de veículos e condutores efetuado por um sistema computacional não se restringem apenas à empresa de transporte:

- Os condutores são beneficiados por meio do recebimento de escalas de trabalho de melhor qualidade, pois o sistema pode gerar escalas de forma que todas as regras

de trabalho sejam obedecidas. O sistema também fornece maior flexibilidade na distribuição da carga de trabalho e das folgas;

- A população é favorecida pelo atendimento pontual, minimizando os eventuais atrasos causados pelas apertadas escalas geradas manualmente. Com um planejamento otimizado há possibilidade da empresa oferecer mais serviços nos horários críticos de maior demanda. Também há possibilidade da empresa repassar a redução dos custos de operação para as tarifas de transporte.

No Brasil, o escalonamento computadorizado de veículos e condutores é uma idéia recente. Os primeiros trabalhos acadêmicos nesta área datam da metade da década de 90 ([20]); e apenas nos últimos anos surgiram no país os primeiros sistemas de gestão empresarial para a área de transportes que agregam a funcionalidade de geração de escalas. Antes disso, e ainda hoje em boa parte das empresas nacionais, o escalonamento de veículos e condutores é realizado manualmente, sendo que sistemas computacionais são utilizados apenas para armazenamento das escalas e geração de relatórios; é provável que existam sistemas de escalonamento com pouca ou nenhuma base científica na sua concepção. Em ambos os casos, a qualidade da solução obtida é relegada a segundo plano, o que acarreta conseqüências como, inclusive, o não-cumprimento de leis trabalhistas [8].

1.1.4 Contextualização acadêmica

O problema de escalonamento de veículos e condutores em transporte rodoviário urbano de passageiros é um problema clássico da área de Pesquisa Operacional, assim como outros problemas semelhantes (ferroviário, aéreo; regional, internacional). Trata-se de um problema de otimização complexo e importante que por si só justificaria a dedicação de uma pesquisa sobre o assunto, tendo estimulado, inclusive, a realização periódica de eventos internacionais sobre o assunto - por exemplo, a *International Conference on Computer-Aided Scheduling of Public Transport*.

O problema de escalonamento de condutores - PEC -, na verdade, consiste em dois subproblemas: o escalonamento diário e o escalonamento semanal/mensal. No escalonamento diário (ou *crew scheduling*), a carga diária de trabalho é distribuída em escalas diárias de condutores, determinando-se assim quantos condutores serão necessários a cada dia e quais viagens fazem parte de cada escala. Porém, salvo algumas exceções, resta ainda definir os condutores que cumprirão cada uma destas escalas, tarefa esta efetuada no escalonamento semanal/mensal (*crew rostering*). Esta etapa do escalonamento é responsável, então, por planejar o cumprimento das escalas diárias ao longo de uma semana ou de um mês, determinando, por exemplo, as folgas semanais de cada condutor.

O presente trabalho enfoca apenas o escalonamento diário dos condutores; assim, a partir deste ponto do trabalho, sempre que for utilizada a expressão "escalonamento de condutores", estar-se-á fazendo referência somente ao escalonamento diário de condutores.

O problema de escalonamento de condutores é de natureza combinatorial. Mesmo considerando-se uma instância do problema correspondente a uma empresa de transporte de porte médio, o número de combinações entre as viagens, veículos e condutores pode ser extremamente alto do ponto de vista computacional ([20]). Várias propostas de modelagem e resolução deste problema já foram publicadas e implementadas com resultados

satisfatórios, como será visto na revisão bibliográfica mais adiante. Porém, por tratar-se de um problema de grande complexidade, esta área de pesquisa está aberta a novas propostas de resolução e a melhorias para as existentes. Ainda, mesmo que se consiga um resultado ótimo do ponto de vista do pesquisador (algo difícil de ocorrer para instâncias grandes do problema), talvez os parâmetros que foram otimizados não sejam aqueles desejados por uma empresa de transporte, de seu próprio ponto de vista [18]. Considere-se ainda as particularidades existentes de empresa para empresa ou, o que é mais importante no caso deste trabalho, as existentes de um país para outro: o problema de escalonamento de veículos e condutores em empresas brasileiras apresenta algumas particularidades com relação aos países europeus e EUA [29].

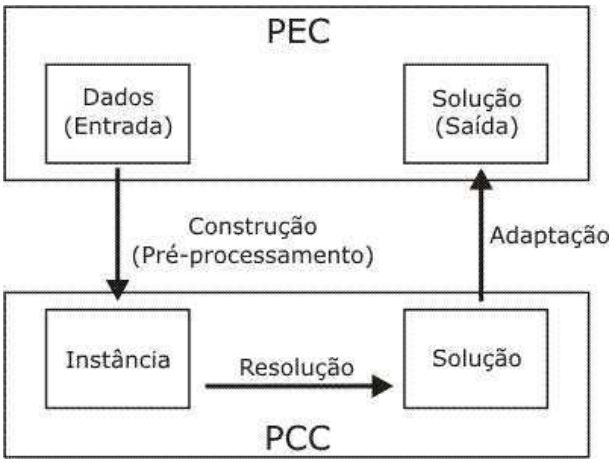

Figura 1.1: Processo de resolução do problema de escalonamento de condutores modelado como um problema de cobertura de conjuntos

É objeto deste trabalho a resolução do PEC modelado como um problema de cobertura de conjunto (PCC), tratado em detalhes no capitulo 3.

Nesta abordagem, largamente utilizada na literatura, a resolução do PEC é dividida em duas fases (figura 1.1): *construção* ou *pré-processamento* da instância do PCC a partir dos dados do PEC; e *resolução* da instância do PCC gerada na fase anterior (a "fase"de *adaptação* do resultado obtido com o PCC para ser utilizado como solução do PEC não tem relevância prática, pois mesmo quando é necessária alguma adaptação da solução obtida, esta é feita por meio de processos simples que não requerem maior preocupação). Um bom resultado para o PEC depende de uma boa formulação e resolução do PCC; porém, a maior parte da bibliografia relacionada trata apenas da resolução das instâncias do PCC, sem preocupação com a construção das mesmas. Assim, este trabalho propõe metodologias heurísticas a serem utilizadas em na fase de *pré-processamento* que antecede à resolução das instâncias do PCC; fase esta que visa gerar as instâncias do PCC de forma que os algoritmos de resolução aplicados sobre as mesmas possam obter melhores resultados para o PCC, e, conseqüentemente, para o PEC.

Ainda como fator de motivação e importância, contribuições para a área de escalonamento de veículos e condutores no transporte rodoviário urbano de passageiros podem trazer avanços para a resolução de outros problemas de escalonamento similares, ou mesmo outros problemas de otimização: são muitos os problemas importantes, abordados pela Pesquisa Operacional, que podem ser modelados como problema de cobertura/partição de conjuntos.

1.2 Objetivos

Dentro do contexto apresentado, o presente trabalho tem como objetivo investigar o uso de metodologias heurísticas para a geração de instâncias do problema de cobertura de conjunto, de forma a melhorar os resultados alcançados pelos algoritmos de resolução do PEC. Desta maneira, a geração das instâncias do PCC constituirá uma nova etapa do processo de escalonamento de veículos, aqui denominada de pré-processamento.

1.3 Estrutura do trabalho

Além deste capítulo de introdução, este trabalho é composto de mais quatro capítulos: no capítulo 2, é apresentada uma revisão bibliográfica sobre o problema de escalonamento de condutores e a fase de pré-processamento; na última seção do mesmo capítulo é feito um breve comentário sobre a meta-heurística *Simulated Annealing*, base para a metodologia proposta neste trabalho e exposta no capítulo 3. No capítulo 4 são apresentados alguns resultados obtidos durante os testes efetuados, que fundamentam as conclusões expostas no capítulo 5.

2 Fundamentação Teórica

Neste capítulo, serão apresentados alguns conceitos básicos que são utilizados no decorrer do trabalho; uma revisão bibliográfica sobre o problema de escalonamento de condutores e a utilização do modelo de cobertura de conjunto na sua modelagem e resolução; e uma visão geral sobre a meta-heurística *Simulated Annealing*, base da metodologia proposta neste trabalho.

2.1 Conceitos e formalizações

Abaixo são apresentados alguns conceitos utilizados neste trabalho, e algumas formalizações e notações que servirão de base para modelos, fórmulas e cálculos que serão expostos mais adiante. As expressões utilizadas para representar os conceitos relativos ao escalonamento de veículos e condutores podem não ser as mesmas que as existentes em empresas de transporte coletivo ou outras fontes bibliográficas, pois há muita diversidade neste aspecto.

Uma **viagem** (v_i) é o deslocamento efetuado por um ônibus (evidentemente guiado por um condutor), tendo o objetivo de realizar o transporte de passageiros. Uma viagem sempre possui um local e um horário (*"timestamp"*) iniciais (LI e HI, respectivamente), e um local e horário finais (LF e HF, respectivamente); a notação $LI(v_i)$ significa: "local de início da viagem v_i"($HI(v_i)$, $LF(v_i)$ e $HF(v_i)$ possuem significado análogo). Obviamente:

1. $HF(v_i) > HI(v_i)$;
2. um veículo (ou condutor) não pode estar realizando mais do que uma viagem em um dado instante t.

Uma **escala de veículo** é uma sequência de viagens a ser executada por um veículo (o conceito de **escala de condutor** é análogo). Um bloco de viagens executado por um determinado condutor, guiando o mesmo veículo, é denominado **partição**. Cada viagem deve ser executada uma única vez, por um único veículo e um único condutor.

É comum existir uma diferença na demanda de acordo com o dia da semana. Geralmente, os dias úteis são os dias onde a demanda a ser atendida é maior, com relação aos sábados e domingos/feriados. Por causa desta diferença, as empresas costumam ter um quadro de viagens para cada **dia típico** (geralmente os dias típicos adotados são: dia útil, sábado e domingo/feriado). É importante ter em mente que o problema global de escalonamento deve ser resolvido para cada um dos dias típicos.

Rede viária é o conjunto de todos os locais e vias que são percorridos pelos veículos, seja cumprindo o roteiro de uma viagem, seja efetuando um deslocamento operacional.

Espalhados pela rede viária, existem **pontos de troca**, locais onde um condutor passa o veículo que conduzia (no final de uma jornada de trabalho ou no início de um período de descanso) para outro condutor. Este é um fato muito comum nas empresas de transporte porque, enquanto um veículo pode rodar durante um dia inteiro ou mais, um condutor tem sua capacidade de trabalho contínuo limitada por leis trabalhistas (e pelas limitações físicas do corpo humano).

2.2 Revisão bibliográfica

2.2.1 Visão geral

Em sua revisão dos métodos de escalonamento de condutores existentes na literatura, Layfield [15] classifica-os em três categorias: métodos heurísticos, métodos baseados em Programação Matemática e métodos baseados em meta-heurísticas.

No primeiro grupo estão as metodologias mais antigas, datando dos anos 60, 70 e 80, que se baseavam nas heurísticas adotadas no escalonamento manual de condutores. Exemplos: TRACS [23], INTERPLAN [22], HOT [9], RUCUS [2] e COMPACS [28]. Segundo o autor, embora muitos destes sistemas ainda estejam em uso em empresas do mundo todo, em termos de pesquisa acadêmica essas metodologias já foram abandonadas.

Devido à melhoria dos recursos computacionais disponíveis, tornou-se possível a utilização da Programação Matemática como metodologia para o escalonamento de condutores, especialmente a partir dos anos 80. Com esta nova abordagem, modelos matemáticos começaram a ser usados para este problema; sendo mais comumente usado o modelo de cobertura de conjunto, comentado mais adiante. Alguns trabalhos que utilizam Programação Matemática e cobertura de conjunto: [21], sistemas IMPACS [25] e HASTUS [16].

Mais recentemente, com o surgimento de meta-heurísticas como Algoritmos Genéticos, *Simulated Annealing*, Busca Tabu, *Ant System* e outras, metodologias de resolução utilizando tais meta-heurísticas começaram a ser pesquisadas, apresentando resultados comparáveis à Programação Matemática. Também neste caso o modelo de cobertura de conjunto é bastante utilizado, como nos seguintes exemplos:

- Algoritmos Genéticos e variantes - [20], [17], [19], [14], [5];
- *Ant System* - [11];
- Busca Tabu - [18]; [24];

2.2.2 Modelo de cobertura de conjunto e o problema de escalonamento de condutores

O escalonamento de condutores tem estreita relação com o problema de escalonamento de veículos, e grande parte dos trabalhos nesta área efetua o (ou consideram a resolução do) escalonamento de veículos antes do escalonamento de condutores. Neste caso, o problema de escalonamento de condutores pode ser modelado como um problema de cobertura de conjunto (PCC) - *set covering problem (SCP)* -, cuja formulação matemática é a seguinte:

$$Min \quad \sum_{j \in J} c_j x_j \qquad (2.1)$$

$$s.a.: \quad \sum_{j \in J} a_{ij} x_j \geq 1 \quad i \in I \qquad (2.2)$$

$$x_j \in 0,1 \qquad j \in J \qquad (2.3)$$

onde:

- I é o conjunto de linhas a serem cobertas;
- J é o conjunto de colunas para cobertura das linhas;
- $a_{ij} = \begin{cases} 1, & \text{se a coluna } j \text{ cobre a linha } i \\ 0, & \text{caso contrário} \end{cases}$
- $x_j = \begin{cases} 1, & \text{se a coluna } j \text{ está na solução} \\ 0, & \text{caso contrário} \end{cases}$
- c_j é o custo da coluna j

No caso do escalonamento de condutores, as linhas do problema correspondem às viagens a serem executadas, e cada coluna representa uma - possibilidade de - escala de trabalho de um condutor, que cobre (executa) uma pequena quantidade de linhas (viagens) do problema.

Estas escalas são geradas a partir da enumeração completa de todas as combinações entre as linhas, que respeitem as restrições adotadas (sobreposição de horários, leis trabalhistas, regras da empresa e outros). Assim, a mesma linha pode ser coberta por várias colunas no modelo, sendo que a melhor solução consiste no subconjunto de colunas que cobre cada uma das linhas ao menos uma vez, com o menor custo possível. Nos métodos de resolução existentes na literatura, costuma-se agrupar as viagens em blocos ou partições, o que diminui consideravelmente o tamanho do problema (considerando-se, como dito acima, que as colunas são determinadas pela enumeração completa das combinações de linhas). Na verdade, estas partições são resultado da divisão das escalas de veículo geradas anteriormente. Assim, uma partição é um bloco de viagens a serem executadas em seqüência pelo mesmo condutor, guiando o mesmo veículo.

Já foi dito anteriormente que a resolução do PEC modelado como um problema de cobertura compreende duas fases principais: a construção da instância do PCC correspondente, e a resolução da mesma. Por sua vez, de acordo com o que foi explicado no parágrafo acima, a fase de construção é dividida também em duas fases: a divisão das escalas de

veículo em partições e a combinação destas partições em possibilidades de escalas de condutor, o que resulta no processo ilustrado na figura 2.1. Por sua vez, a instância do PCC montada a partir das partições e possibilidades de escalas é ilustrada na figura 2.2.

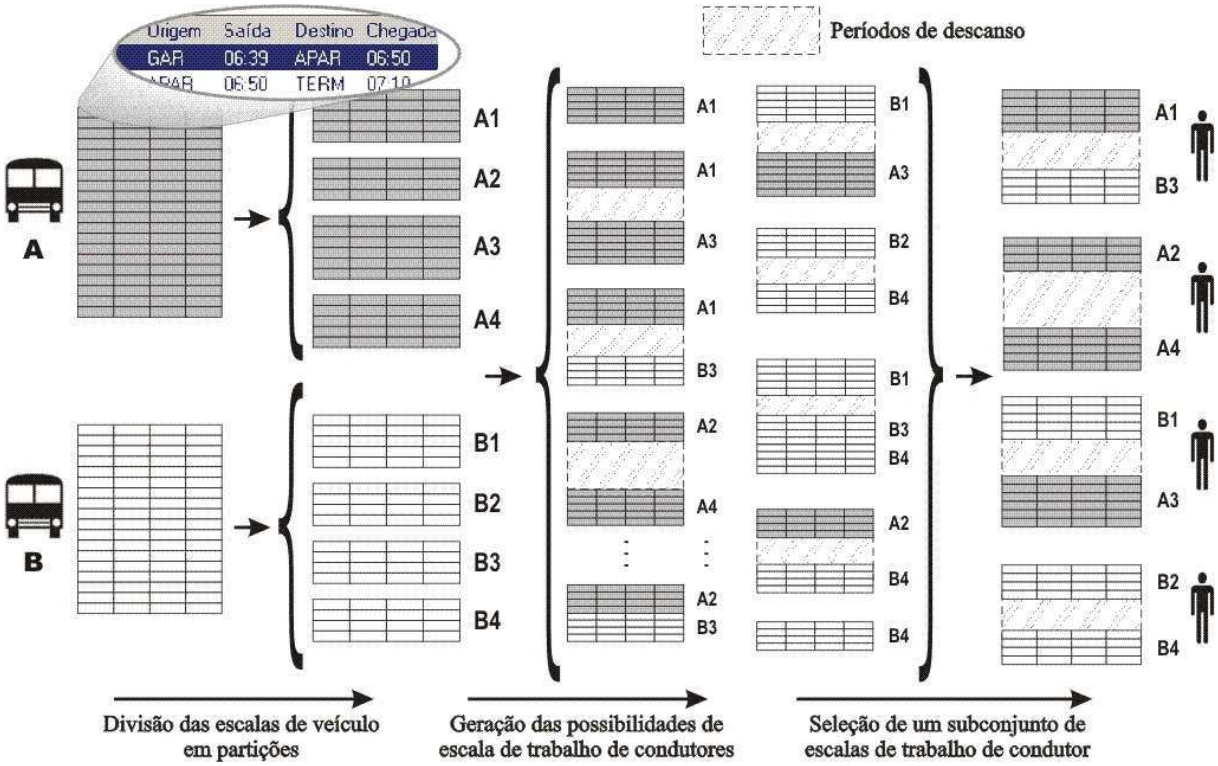

Figura 2.1: Visão geral do processo de escalonamento de condutores

As escalas de veículo não podem ser divididas entre duas viagens quaisquer. Considerando que o local de destino de uma viagem é o mesmo local de partida da viagem seguinte, uma escala só pode ser dividida entre duas viagens onde este "local intermediário"seja um **ponto de troca** de condutores, definido pela empresa de transporte. A posição na escala correspondente a um ponto de troca é chamada na literatura como uma **oportunidade de troca** (*relief opportunity*).

Possíveis redundâncias (mais de uma coluna cobrindo uma linha na solução) devem ser eliminadas após a resolução. Estas redundâncias poderiam ser evitadas, substituindo-se no modelo a desigualdade (2.2) por uma igualdade; desta forma, teríamos um problema de partição de conjuntos - *set partitioning*. Porém, muitos trabalhos optam pela cobertura de conjuntos a fim de facilitar o processo de resolução.

Ao conjunto de escalas de condutor que formam uma solução dá-se o nome de **plano de escalas** (*scheduling*). O custo c_j de uma coluna corresponde ao valor pago pela empresa pelo cumprimento da escala; este custo varia, por exemplo, de acordo com a quantidade de horas-extras contidas (outras variáveis podem entrar no cálculo de c_j, dependendo da metodologia adotada). É interessante notar que a formulação do problema de escalonamento de condutores como um problema de cobertura de conjunto oferece a vantagem de tornar a modelagem matemática do problema (e a resolução) independente das leis trabalhistas, regras da empresa e outros fatores que influem na formação das escalas, sendo estes fatores considerados apenas na etapa de geração destas colunas. A modelagem do problema de escalonamento de veículos desta forma permite ainda que sua

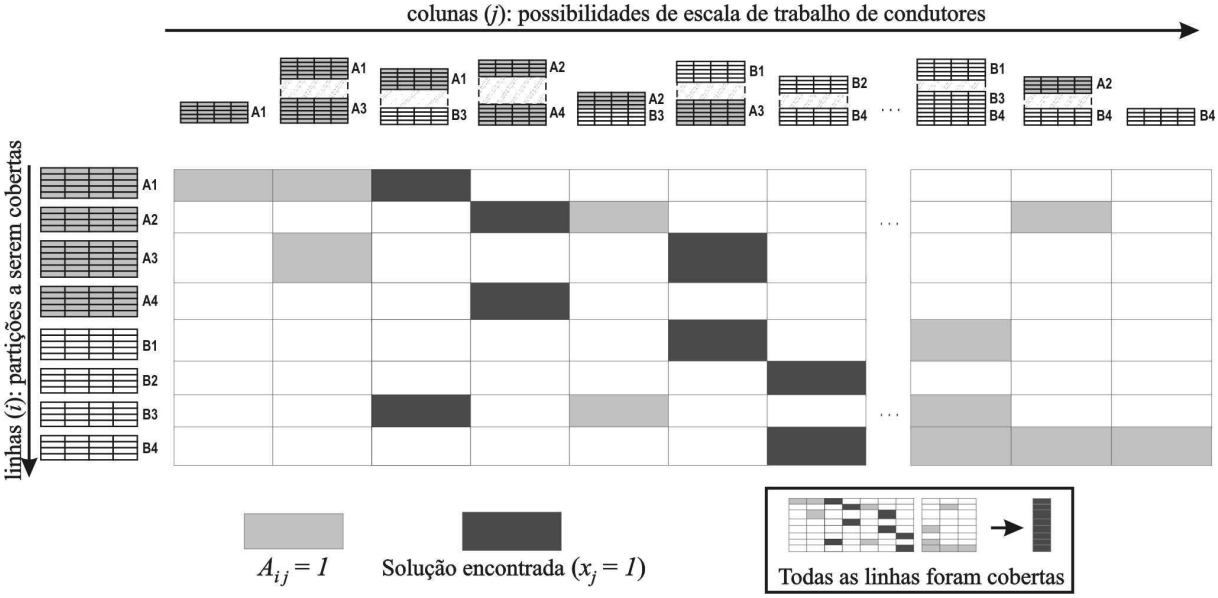

Figura 2.2: Ilustração do escalonamento de condutores modelado como um problema de cobertura de conjunto

resolução seja efetuada por meio dos vários algoritmos destinados ao PCC existentes na literatura (exemplos: [1], [3], [12], [27]).

2.2.3 Pré-processamento do problema de cobertura de conjunto

Grande parte dos trabalhos a respeito de escalonamento de condutores - e muito mais aqueles que tratam do PCC especificamente - iniciam seus estudos a partir do estágio onde o problema de cobertura já está montado, com suas linhas e colunas definidas. Apresentam métodos para selecionar o melhor subconjunto de colunas (aquele que cobre todas as linhas com o menor custo); porém, sem abordarem a geração destas linhas e colunas. Alguns trabalhos vão um pouco além, aplicando métodos que procuram selecionar apenas uma parte das colunas do PCC construído, a fim de reduzir o tamanho do problema e facilitar a resolução.

Em [5], colunas consideradas ineficientes são simplesmente retiradas do problema, a menos que alguma linha não esteja coberta suficientemente para permitir tal exclusão. Esta eficiência de uma escala de condutor é definida como a relação entre as horas pagas pelo seu cumprimento e as efetivamente trabalhadas.

Em [19], o método para a resolução de um PCC consiste em, a partir de um sub-PCC inicial (com apenas algumas das colunas do PCC original), utilizar-se programação linear na obtenção de um direcionamento para a formação de um novo sub-conjunto de colunas (e, conseqüentemente, um novo sub-PCC); e assim sucessivamente, até obter-se um sub-conjunto de colunas satisfatório segundo critérios estabelecidos. Este último sub-PCC é então resolvido utilizando-se programação linear inteira, obtendo-se assim uma solução para o PCC original (completo). Portanto, estes métodos trabalham sobre o PCC já construído; mas, embora procurem efetuar algum processamento sobre as colunas do problema, não tratam da formação das partições; ou seja, da geração das linhas da cobertura

de conjunto.

a) Divisão em partições utilizando o algoritmo de Dijkstra

Outra abordagem, sugerida em [20], consiste em efetuar uma estimativa do custo das escalas de condutor já na etapa de escalonamento dos veículos, incluindo-a no cálculo do custo de atribuição das viagens aos veículos. Na verdade, essa estimativa refere-se aos períodos ociosos pagos que surgem no processo de divisão das escalas de veículo em partições. A técnica utilizada tenta direcionar a atribuição de viagens às escalas de veículo de forma a reduzir a ocorrência de tais períodos, efetuando "previsões"da divisão das escalas durante a formação destas. A dificuldade com este método é que, com as partições geradas segundo este critério, não há garantias de que o PCC construído posteriormente apresente cobertura suficiente para suas linhas, podendo prejudicar o resultado final do processo de escalonamento. Aliás, esta dificuldade pode acontecer com qualquer critério de divisão que não leve em consideração as características do PCC que será construído.

A idéia proposta em [20] para a divisão de uma escala de veículo em partições baseia-se na formulação de um grafo $H(I, C)$, onde o conjunto I de vértices corresponde às oportunidades de troca existentes na escala, além dos pontos s e t, que representam, respectivamente, o início e o fim da mesma; a existência de uma aresta $(i_x, i_y) \in C$ indica a possibilidade de se formar uma partição com as viagens entre os intervalos i_x e i_y. O custo c_{xy} associado a cada arco é relativo ao intervalo de trabalho t_{xy} compreendido entre os intervalos i_x e i_y; c_{xy} é dado por:

$$c_{xy} = \begin{cases} P_{min}, & \text{se } t_{xy} < P_{min} \\ \infty, & \text{se } t_{xy} > P_{max} \\ t_{xy}, & \text{nos demais casos} \end{cases}$$

onde P_{min} e P_{max} são, respectivamente, os limites mínimo e máximo de comprimento das partições, estabelecidos pelo usuário. Observe que é permitido que uma partição tenha um comprimento menor que P_{min}; porém, sua utilização nas escalas é dificultada pelo artifício de se atribuir a estas pequenas partições um custo maior do que o seu comprimento (no caso, o próprio parâmetro P_{min}). A figura abaixo mostra um exemplo do grafo de formação de partições para uma escala de 5 viagens.

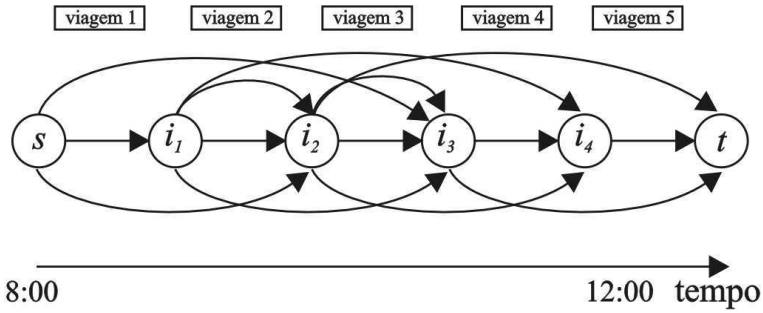

Figura 2.3: Exemplo de um grafo de formação de partições

Encontrar a melhor divisão de partições para a escala em questão seria equivalente a encontrar o caminho de custo mínimo no grafo H acima. Esta solução pode ser dada, por

exemplo, pelo algoritmo de Dijkstra. O objetivo principal de se modelar o problema da divisão em partições desta forma é procurar dividir as escalas de veículo nos pontos de maior intervalo entre as viagens, evitando assim que esses intervalos se tornem períodos ociosos dentro das escalas de condutor, mais adiante. Porém, nos experimentos realizados com esta maneira de montar o PCC, notou-se que algumas partições são cobertas por um número muito reduzido de colunas. Inclusive, se não for dada a oportunidade de se formar escalas de condutor com apenas uma partição, algumas delas poderão não ser cobertos por nenhuma coluna, inviabilizando o processo de escalonamento. Por exemplo, considere as escalas de veículo e as respectivas partições na figura 2.4.

Linha	Origem	Saída	Destino	Chegada	Partição
200	TERM	08:00	TERM	09:00	1
200	TERM	09:00	TERM	10:00	1
200	TERM	10:00	TERM	11:00	1
200	TERM	11:00	TERM	12:00	1
210	TERM	12:15	TERM	13:00	2
210	TERM	13:00	TERM	13:45	2
210	TERM	13:45	TERM	14:30	2
210	TERM	14:30	TERM	15:15	2

Linha	Origem	Saída	Destino	Chegada	Partição
300	TERM	08:00	TERM	09:00	3
300	TERM	09:00	TERM	10:00	3
300	TERM	10:00	TERM	11:00	3
300	TERM	11:00	TERM	12:00	3
310	TERM	12:15	TERM	13:00	4
310	TERM	13:00	TERM	13:45	4
310	TERM	13:45	TERM	14:30	4
310	TERM	14:30	TERM	15:15	4

Figura 2.4: Exemplo de uma divisão em partições utilizando o algoritmo de Dijkstra

Sendo o algoritmo de Dijkstra exato, caso ele encontre duas escalas idênticas como as mostradas na figura acima, ele dividirá as escalas em partições de maneira também idêntica. O problema é que, com as partições arranjadas desta forma, não é possível formar nenhuma escala de condutor com mais de uma partição, pois:

- as seguintes combinações de partições não são possíveis, dada a sobreposição de horários: (partição 1, partição 3) e (partição 2 e partição 4).
- quanto às combinações (partição 1 e partição 2), (partição 3 e partição 4), (partição 3 e partição 2) e (partição 1 e partição 4), o intervalo de 15 minutos (12:00 - 12:15) entre as partições é muito pequeno para formar um intervalo de almoço. O mínimo intervalo permitido dificilmente é menor que 40 minutos.
- ainda para estes mesmos quatro casos, se considerarmos o intervalo entre as partições como um período apenas para troca de veículos, obteríamos um período contínuo de trabalho muito extenso: 7 horas e 15 minutos; sendo que, geralmente, não se deve ultrapassar 6 horas de trabalho sem um intervalo para descanso.

Dessa forma, seriam necessários 4 condutores para cobrir estas 4 partições de trabalho.

b) Divisão em partições utilizando um algoritmo guloso randomizado

Visando melhorar a metodologia anterior, em [4] é proposta a alternativa de se utilizar um "algoritmo guloso randomizado", construído a partir do próprio algoritmo de Dijkstra, para que este apresente formas aleatórias (mas sempre com tendência ao menor caminho) de divisão de partições, relaxando um pouco a restrição sobre períodos ociosos dentro das partições em favor de um número maior de combinações (escalas). Dessa forma, uma

Linha	Origem	Saída	Destino	Chegada	Partição
200	TERM	08:00	TERM	09:00	1
200	TERM	09:00	TERM	10:00	1
200	TERM	10:00	TERM	11:00	1
200	TERM	11:00	TERM	12:00	2
210	TERM	12:15	TERM	13:00	2
210	TERM	13:00	TERM	13:45	2
210	TERM	13:45	TERM	14:30	2
210	TERM	14:30	TERM	15:15	2

Linha	Origem	Saída	Destino	Chegada	Partição
300	TERM	08:00	TERM	09:00	3
300	TERM	09:00	TERM	10:00	3
300	TERM	10:00	TERM	11:00	3
300	TERM	11:00	TERM	12:00	3
310	TERM	12:15	TERM	13:00	3
310	TERM	13:00	TERM	13:45	4
310	TERM	13:45	TERM	14:30	4
310	TERM	14:30	TERM	15:15	4

Figura 2.5: Uma possível solução obtida utilizando o algoritmo de Dijkstra modificado

possível saída apresentada pelo algoritmo de Dijkstra modificado para o exemplo anterior seria a mostrada na figura 2.5. Note que foi possível agora a formação de uma escala combinando duas partições, também mostrada na mesma figura.

Com esta nova possibilidade de escala de condutor, cobrindo as partições 1 e 4, pode-se agora cobrir as 4 partições do exemplo com apenas 3 condutores, um a menos que no exemplo anterior. Agora, com um algoritmo gerando várias possibilidades de se dividir cada escala, é necessário obter uma configuração apropriada de partições para todo o conjunto de escalas de veículo. O referido trabalho apresenta uma metodologia baseada na meta-heurística *Simulated Annealing*, que procura gerar as partições de maneira a maximizar o número de possibilidades de escalas de condutor, com as partições sendo cobertas de forma mais homogênea (ou seja, quanto ao número de colunas que cobre cada linha) e minimizando a quantidade de períodos ociosos fazendo parte das partições. Deve-se ressaltar a importância de se considerar a homogeneidade na avaliação do PCC construído, pois é com este artifício que se consegue, ao mesmo tempo, melhorar a cobertura de partições "raras", e evitar a geração de coberturas excessivas (o que aumentaria o tamanho do problema desnecessariamente). Nos testes apresentados em [4], conseguiu-se alguma redução, tanto em termos de custos operacionais, quanto ao número de condutores utilizados.

2.2.4 Encadeamento de intervalos - *mealbreak chain*

Os trabalhos [10] e [15] tratam a construção do problema de cobertura de conjunto segundo o conceito de *mealbreak chain*, algo que pode ser traduzido como "encadeamento de intervalos de descanso". A idéia de encadeamento de intervalos e seu efeito sobre o processo de escalonamento de condutores podem ser ilustrados através do seguinte exemplo: sejam as escalas de veículo da figura 2.6.

As posições marcadas com "X"representam o horário máximo possível que os respectivos condutores podem trabalhar sem um intervalo de descanso, considerando que o horário da próxima oportunidade de troca estaria fora do limite imposto pelas regras trabalhistas ou da empresa. Repare que, estando as partições (na figura, assim como no trabalho a que pertence, é utilizado o termo *stretch*) montadas daquela forma, seriam necessários três condutores para assumirem os veículos deixados pelos outros três que estão saindo para

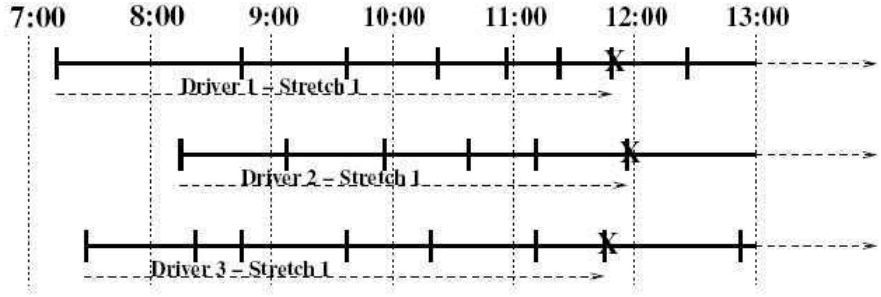

Figura 2.6: Escalas sem *mealbreak chains* (fonte: [15])

o almoço. Com um encadeamento de intervalos, pode-se conseguir que seja adicionado apenas um condutor ao plano de escalas, como no exemplo da figura 2.7.

A idéia aqui é montar as partições de tal forma que, quando um condutor sair para um intervalo de descanso, o veículo que ele estava conduzindo possa ser assumido por um condutor que, de preferência, esteja retornando de um intervalo. Assim, na figura, o condutor (*driver*) 1 foi substituído pelo único condutor adicionado ("*new driver 1*") aos já existentes. Quando ele termina seu intervalo de almoço, substitui o condutor 3 para que este possa iniciar seu intervalo. O condutor 3, por sua vez, ao voltar do intervalo, libera o condutor 2 e assim por diante. Desta maneira, o trabalho que, no exemplo anterior, seria efetuado por seis condutores, aqui é coberto por apenas quatro. Em [15], é proposta uma metodologia onde as partições são construídas de maneira a formar *mealbreak chains*; o algoritmo foi implementado como um módulo de pré-processamento para o sistema de escalonamento TRACS II. Os resultados obtidos foram equivalentes ao TRACS II original, mas com um tempo de processamento até 90% menor em certos casos.

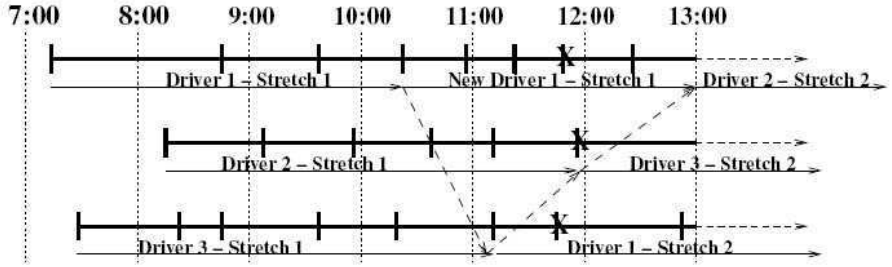

Figura 2.7: Exemplo de *mealbreak chain* (fonte: [15])

2.3 *Simulated Annealing*

Nesta seção, faz-se uma introdução à meta-heurística *Simulated Annealing*, base para a concepção do algoritmo aqui proposto.

2.3.1 Origem da meta-heurística

O termo **anelamento** (*annealing*) é dado ao processo de aquecimento de um sólido até o seu ponto de fusão, seguido de um resfriamento gradual e vagaroso, até que se alcance novamente o seu enrijecimento. Nesse processo, o resfriamento vagaroso é essencial para se manter um equilíbrio térmico no qual os átomos encontrarão tempo suficiente para se organizarem em uma estrutura uniforme com energia mínima. Se o sólido é resfriado bruscamente, seus átomos formarão uma estrutura irregular e fraca, com alta energia em conseqüência do esforço interno gasto.

Computacionalmente, o recozimento pode ser visto como um processo estocástico de determinação de uma organização dos átomos de um sólido, que apresente energia mínima. Em temperatura alta, os átomos se movem livremente e, com grande probabilidade, podem mover-se para posições que incrementarão a energia total do sistema. Quando se baixa a temperatura, os átomos gradualmente se movem em direção à uma estrutura regular e, somente com pequena probabilidade, incrementarão suas energias.

2.3.2 *Simulated Annealing* como um método de otimização

Simulated Annealing é considerado um tipo de algoritmo chamado de busca global. Ele se constitui em um método de obtenção de boas soluções para problemas de otimização de difícil resolução. Desde a sua introdução como um método de otimização combinatorial, esse método vem sendo vastamente utilizado em diversas áreas, tais como projeto de circuitos integrados auxiliado por computador, processamento de imagem, redes neuronais, etc.

A sua semelhança com o método original no qual foi inspirado é muito grande. Na sua apresentação, nos trabalhos independentes de Kirkpatrick *et al.* ([13]) e Cerny ([6]), é mostrado como um modelo de simulação de recozimento de sólidos pode ser utilizado em problemas de otimização, onde a função objetivo a ser minimizada corresponde à energia dos estados do sólido.

Antes de passar para a descrição do algoritmo propriamente dito, será definido o problema de otimização que se pretende resolver. Para tanto, seja S o espaço total de soluções de um problema combinatorial. Ou seja, S é o conjunto finito que contém todas as combinações possíveis que representam as soluções viáveis para o problema. Seja f uma função de valores reais definida sobre $S, f : S \rightarrow R$, que representa o custo da solução S. O problema se constitui em encontrar uma solução (ou estado) $i \in S$, tal que $f(i)$ seja mínimo.

Uma das formas mais simples de tentar resolver o problema utilizando busca local em S, conhecida como algoritmo descendente, é iniciar o processo de busca por uma solução normalmente tomada de forma aleatória. Uma outra solução j é então gerada, na vizinhança

desta, por meio de um mecanismo apropriado e dependente do problema. Caso uma redução do custo dessa nova solução seja verificada, ou seja, $f(j) < f(i)$, a mesma passa a ser considerada a solução corrente e o processo se repete. Caso contrário, a nova solução é rejeitada e uma outra gerada. Esse processo se repete até que nenhum melhoramento possa ser obtido na vizinhança da solução corrente, após um número determinado de insistências. O algoritmo retorna, então, o valor da última solução corrente, considerada uma solução de mínimo local.

O grande problema desse método, muito simples e rápido, é que o mínimo local encontrado pode estar longe de ser um mínimo global, o que se traduziria em uma solução inaceitável para o problema. Uma estratégia muito simples de aprimorar a solução obtida por meio desse tipo de algoritmo, seria escolher a menor solução, de um conjunto de soluções obtidas de execuções sucessivas, realizadas a partir de diferentes soluções iniciais.

Simulated annealing não utiliza essa estratégia. Esse método tenta evitar a convergência para um mínimo local, aceitando esporadicamente soluções que incrementem o valor de f. Baseando-se no resfriamento gradual que é uma das idéias centrais do processo natural em que se baseia o algoritmo, *simulated annealing* utiliza um parâmetro denominado *temperatura* - T para efetuar o controle da aceitação destas soluções. O aceite ou rejeição de uma nova solução que causará um incremento de δ em f, é determinado por um critério probabilístico, por meio de uma função g conhecida por **função de aceite**, sendo T um dos componentes da mesma. Normalmente, essa função é expressa por

$$g(\delta, T) = e^{-\delta/T} \tag{2.3.1}$$

Caso $\delta = f(j) - f(i)$ seja menor que zero, a solução j será aceita como a nova solução corrente. Caso contrário, a nova solução somente será aceita se

$$g(\delta, T) > random(0, 1) \tag{2.3.2}$$

Sendo *random* uma função geradora de números aleatórios entre 0 e 1.

Da mesma forma que no processo físico, a função $g(\delta, T)$ implica em:

- a probabilidade de aceite de uma nova solução é inversamente proporcional ao incremento de δ;
- quando T é alto, a maioria dos movimentos (de um estado para outro ou de uma solução para outra) é aceita; entretanto, a medida que T se aproxima de zero, a grande maioria das soluções são rejeitadas.

Procurando evitar uma convergência precoce para um mínimo local, o algoritmo inicia com um valor de T relativamente alto. Esse parâmetro é gradualmente diminuído e, para cada um dos seus valores, são realizadas várias tentativas de se alcançar uma melhor solução, nas vizinhanças da solução corrente.

Os passos principais da meta-heurística são apresentados em forma de algoritmo na figura 2.8.

```
Sejam os parâmetros de entrada:
-  MT = tempo limite para execução do algoritmo;
-  T = temperatura inicial; este parâmetro influencia na probabilidade
   de aceitação de soluções inferiores à incumbente;
-  TL = número de iterações para cada valor da temperatura (T);
-  CF = fator de resfriamento; coeficiente que determina o decréscimo da
   temperatura a cada TL iterações;

Algoritmo:
   construir uma solução inicial factível S, cujo custo é dado por f(S);
   S* := S'
   enquanto não ultrapassado o tempo MT faça
      para i := 1 até TL faça
            S' := solução vizinha a S; seu custo é dado por f(S');
            δ := f(S') - f(S);
            se δ <= 0
                  faça S := S';
                  se f(S') < f(S*) então S* := S';
            senão
                  com probabilidade e^(-δ/T), faça S := S';
            fim-se
      fim-para
      T := T . CF;
   fim-enquanto
   retorne S*
```

Figura 2.8: *Simulated Annealing*

3 Metodologia proposta

Como dito anteriormente, e representado na figura 2.1, a resolução do problema de escalonamento de condutores modelado como um problema de cobertura de conjunto (PCC) é dividida em três fases:

fase 1 divisão das escalas de veículo em partições - determinação das linhas do PCC;

fase 2 geração das possibilidades de escala de trabalho de condutor (alternativas de escala) - enumeração das colunas do PCC;

fase 3 resolução do problema de cobertura de conjunto, formado pelas partições (linhas) e alternativas de escalas de condutor (colunas).

O método de geração das possibilidades de escala de trabalho de condutor adotado neste trabalho será exposto mais adiante neste capítulo. Porém, é muito importante adiantar e ressaltar que esta segunda fase é uma conseqüência imediata da primeira, pois para cada conjunto de partições só pode ser gerado um único e respectivo conjunto de alternativas de escalas de condutor. Isto porque o conjunto de alternativas de escala é determinado tão somente pelas combinações entre as partições (1, 2, 3 ou 4 deles) que formem uma escala de condutor válida, conforme as regras adotadas pela metodologia de escalonamento - regras das quais a mais óbvia é não haver sobreposição de horários entre as partições.

Esta relação entre a primeira e segunda fases do escalonamento de condutores implica que *a resolução do subproblema de divisão das escalas de veículo em partições determina também o problema de cobertura de conjunto a ser resolvido na terceira etapa.*

Neste capítulo, será apresentada a metodologia de pré-processamento do PCC, desenvolvida para efetuar a primeira fase, buscando a geração um problema de cobertura de conjunto que possibilite a obtenção de melhores soluções na terceira fase do processo de escalonamento de condutores.

3.1 Descrição da metodologia proposta

A metodologia desenvolvida neste trabalho para a divisão das escalas de veículo em partições baseia-se na meta-heurística *Simulated Annealing*. Durante a descrição da metodologia, serão utilizadas as formalizações apresentadas na seção 2.1, bem como as seguintes:

- O conjunto de viagens:

$$V = \{v_1, v_2, v_3, ..., v_n\}$$

- Seja U_i a escala do veículo i, definida como:

$$U_i \subseteq V, \forall i = 1, 2, ..., u$$

tal que:

$$\bigcap_{i=1}^{u} U_i = \emptyset \quad \text{e}$$

$$\bigcup_{i=1}^{u} U_i = V$$

Ou seja, toda viagem deve estar atribuída a um e somente um veículo. Ainda, para cada escala de veículo, não podem haver viagens com horários sobrepostos.

- Considere P_{ij} a partição j da escala do veículo i, definida como:

$$P_{i,j} \subseteq U_i, \forall j = 1, 2, ..., p_i$$

$$\bigcap_{j=1}^{p_i} P_{i,j} = \emptyset$$

$$\bigcup_{j=1}^{p_i} P_{i,j} = U_i$$

Ou seja, cada viagem pertencente à escala U_i deve pertencer a uma e somente uma partição $P_{i,j}, j = 1, 2, ..., p_i$.

Além da regra exposta acima, em cada partição só podem existir viagens que são consecutivas dentro da escala de veículo.

Estas regras devem ser respeitadas para cada uma das escalas de veículo $U_i, i = 1, 2, ...u$.

Assim como em outras meta-heurísticas, utilizar-se do *Simulated Annealing* para criar um algoritmo consiste em implementar cada um dos passos da meta-heurística, adequando-a ao problema a ser resolvido. No caso do *Simulated Annealing*, existem três passos a serem implementados: a geração da solução inicial, a geração da solução vizinha e o cálculo da função f (custo da solução).

A entrada do algoritmo de divisão de partições é o conjunto U das escalas de veículo. Uma *solução* S, no caso do algoritmo de divisão em partições, é nada mais que um conjunto de todas as partições $P_{i,j}$ resultantes da divisão de todas as escalas de veículo U_i. Para gerar a solução inicial, utilizou-se o algoritmo de Dijkstra para efetuar a divisão das escalas de veículo em um conjunto de partições, como já exposto na seção 2.2.3a.

A implementação dos outros dois passos (geração da solução vizinha e o cálculo da função f) é apresentada nas sub-seções seguintes. Na verdade, foram implementadas duas variantes da metodologia:

- A primeira variante do algoritmo (que será denominada por APP_1 - Algoritmo de Pré-Processamento 1) busca por um PCC onde as linhas estejam cobertas pelo maior número de colunas, e da maneira mais homogênea possível. Assim, os problemas de cobertura que vão sendo gerados durante a execução do algoritmo são avaliados de acordo com esta cobertura e homogeneidade.
- A segunda variante (APP_2) avalia os problemas de cobertura gerados utilizando um método guloso de resolução, cujo custo da solução obtida é utilizada como uma estimativa da qualidade da solução.

3.1.1 Cálculo do custo de uma solução

Para ambas as variantes implementadas do algoritmo, é necessário construir o problema de cobertura de conjunto correspondente à solução S, antes de efetuar o cálculo de seu custo.

Por sua vez, para construir o problema de cobertura, é necessário enumerar todas as alternativas de escala de condutor possíveis de se formar com as partições da solução S.

Uma escala de condutor é formada pela combinação de 1, 2, 3 ou até 4 partições, e deve obedecer aos seguintes parâmetros (referentes à leis trabalhistas e/ou à certas regras internas da empresa), ilustrados na figura 3.1:

- TC (trabalho contínuo máximo): é o máximo período de tempo que um condutor pode trabalhar sem ter um período de descanso.
- JN (jornada normal de trabalho): é a duração máxima de uma escala de trabalho normal, sem horas extras.
- ID_{min} e ID_{max} (mínimo e máximo intervalo de descanso): delimitam o comprimento de um período de descanso.
- TV (mínimo intervalo para troca de veículo): é o tempo mínimo estimado para um condutor trocar de veículo, entre uma partição e outra.

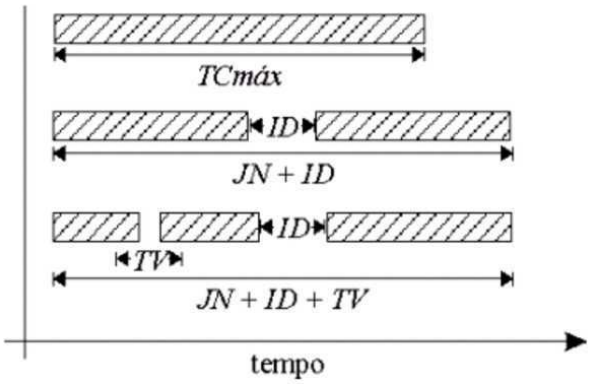

Figura 3.1: Ilustração dos parâmetros para formação de escalas de condutores

- HE (máximo período de horas extras): é a quantidade máxima de horas extras que uma escala pode conter. Uma escala não pode ter duração maior do que $JN + HE$.

Cabe ressaltar que o intervalo de descanso das escalas não são contabilizados como parte da jornada de trabalho, ao contrário do intervalo para troca de veículo.

Todas as combinações possíveis de 1, 2, 3 e 4 partições são testadas na geração de escalas de trabalho; porém, em razão das restrições acima e da possível sobreposição de horários, nem toda combinação de partições é capaz de gerar uma escala.

Após gerado o conjunto de alternativas de escalas de condutor (sempre lembrando que estas correspondem às colunas do problema de cobertura, como expresso na figura 2.2), o custo da solução S pode agora ser calculado.

***APP*$_1$** Na primeira variante do algoritmo, o custo de uma solução S baseia-se nas características do problema de cobertura de conjunto formado a partir de suas partições; sendo $cob(P)$ o número de colunas que cobrem a linha (partição) $P \in S$:

$$f(S) = col(S) * \delta(S)/\mu(S)^2$$

onde $col(S)$ é o número de colunas do problema, $\mu(S)$ e $\delta(S)$ são, respectivamente, a média e o desvio padrão de $cob(P)$.

Ao minimizar o valor de $f(S)$, o algoritmo APP_1 estará procurando por uma instância do problema de cobertura onde cada linha seja coberta pelo maior número de colunas possível ($\mu(S)^2$) , de forma homogênea ($\delta(S)$) e sem aumentar desnecessariamente o número de colunas ($col(S)$).

***APP*$_2$** Na segunda variante, o problema de cobertura construído é submetido a um algoritmo guloso (relativamente rápido) de resolução. O custo da solução obtida é utilizado como o custo $f(S)$.

O algoritmo guloso utilizado é proposto em [12] para a construção da solução inicial na metodologia de resolução do PCC (baseada também em *Simulated Annealing*) proposta naquele trabalho. A descrição do algoritmo é a seguinte:

Sendo :

I	=	o conjunto de linhas do problema de cobertura;
J	=	o conjunto de colunas do problema de cobertura;
J_R	=	o conjunto de todas as colunas, em ordem crescente de custo, que não pertencem à solução;
J_S	=	o conjunto de todas as colunas, em ordem crescente de custo, que pertencem à solução;
w_i	=	o número de colunas que cobrem a linha $i, i \in I$;
U	=	o conjunto de linhas não cobertas (ou seja, linhas onde $w_i = 0$)

Observação: as notações definidas aqui têm seu escopo limitado à descrição deste algoritmo, não devendo ser confundidas com outras notações utilizadas no restante deste trabalho.

1. Inicialização: $J_R = J, J_S = \emptyset, U = I, w_i = 0 \ \forall i \in I$;
2. Selecione aleatoriamente uma linha $i \in I$;
3. Selecione a primeira coluna $j \in J_R$, para a qual $a_{ij} = 1$. Mova a coluna j de J_R para J_S;
4. Faça $w_i = w_i + a_{ij} \ \forall i \in I$. Defina U como o conjunto de linhas para as quais $w_i = 0$. Se $U = \emptyset$, vá para o passo 5, senão retorne para o passo 2.
5. Finalização: examine cada coluna $k \in J_S$, em ordem decrescente de custo. Se $w_i - a_{ik} \geq 1 \ \forall i \in I$, então mova k de J_S para J_R e faça $w_i = w_i - a_{ik} \ \forall i \in I$.

3.1.2 Geração da solução vizinha

Em ambas as variantes do algoritmo, para a geração da solução vizinha S' é necessário, em primeiro lugar, selecionar-se uma partição P da solução atual S para ser modificada. A partição selecionada é então submetida a sete diferentes heurísticas de alteração, o que dá origem a sete novas soluções. Finalmente, é selecionada aleatoriamente uma destas soluções como sendo a solução vizinha S', com probabilidades de seleção inversamente proporcionais aos seus respectivos custos.

3.1.2.1 Seleção da partição a ser modificada

***APP*$_1$** Na primeira variante do algoritmo, a seleção da partição P a ser modificada leva em consideração o problema de cobertura de conjunto PCC_S, formado a partir da solução atual S.

A seleção da partição P a ser modificada é feita com base nos valores de $cob(P)$ para as partições $P \in S$: são selecionadas aleatoriamente duas partições, das quais aquela com maior $|cob(P) - \mu(S)|$ (abordagem *tournament*) será a partição escolhida para sofrer a modificação.

***APP*$_2$** Para a segunda variante do algoritmo, é selecionada uma partição aleatoriamente, sendo que todas as partições possuem a mesma probabilidade de serem selecionadas.

3.1.2.2 Modificação da partição selecionada

Após a seleção da partição P, esta será submetida à sete heurísticas de modificação. Sendo U_i a escala de veículo que contém a partição P, as heurísticas são as seguintes:

expansão do limite inferior caso exista uma partição $P' \in U_i$ anterior a P, a última viagem de P' será transferida para a partição P. O processo se repete até que o local de origem da viagem transferida seja um ponto de troca de condutores.

retração do limite inferior caso exista uma partição $P' \in U_i$ anterior a P, a primeira viagem de P será transferida para a partição P'. O processo se repete até que o local de destino da viagem transferida seja um ponto de troca de condutores.

expansão do limite superior caso exista uma partição $P' \in U_i$ posterior a P, a primeira viagem de P' será transferida para a partição P. O processo se repete até que o local de destino da viagem transferida seja um ponto de troca de condutores.

retração do limite superior caso exista uma partição $P' \in U_i$ posterior a P, a última viagem de P será transferida para a partição P'. O processo se repete até que o local de origem da viagem transferida seja um ponto de troca de condutores.

divisão seja k o número de oportunidades de troca existentes dentro da partição P (excluindo-se o início e o fim desta); caso $k > 1$, divide-se a partição P em duas novas partições, no ponto da n-ésima oportunidade de troca. O valor de n é dado por $\lfloor k/2 \rfloor$.

união une-se a partição P a uma de suas partições vizinhas (caso exista o vizinho anterior e posterior, seleciona-se aleatoriamente um deles); caso esta união resulte em uma partição maior do que o permitido, transfere-se para P apenas o número de viagens possível.

reconstrução das partições da escala refaz-se toda a divisão em partições da escala U_i, utilizando o algoritmo guloso randomizado descrito na seção 2.2.3b.

3.1.3 Relaxamento da restrição quanto a períodos ociosos dentro das escalas

Discutiu-se, nas seções 2.2.3a e 2.2.3b, que a utilização do algoritmo de Dijkstra para efetuar a divisão das escalas de veículos em partições faz com que estas últimas contenham, dentro de si, o mínimo possível de períodos ociosos, assim como as escalas de condutor que serão formadas a partir deles. A metodologia aqui proposta, ao procurar por novos conjuntos de partições, acaba efetuando um relaxamento nesta restrição; para controlar este relaxamento, desenvolveu-se um mecanismo para evitar que períodos ociosos sejam incorporados excessivamente às partições de uma solução: seja S_0 a solução inicial do algoritmo, $t(p)$ a duração de uma partição p qualquer; são descartadas automaticamente as soluções S' que apresentarem:

$$\sum_{p' \in S'} t(p') > \sum_{p \in S_0} t(p) + \%perda_{max} \tag{3.1.1}$$

sendo $\%perda_{max}$ um parâmetro de entrada definido para o algoritmo.

4 Resultados obtidos e análise

Neste capítulo, são apresentados alguns resultados obtidos com a utilização das duas variantes da metodologia de pré-processamento do PCC apresentadas no capítulo anterior. Para avaliação dos resultados, foram utilizadas três bases de dados distintas:

- as bases de dados utilizadas em [20], correspondentes aos dados reais de uma empresa de transporte coletivo de Florianópolis/SC. Foram utilizados os quadros de viagens de dias úteis (602 viagens) e de domingos e feriados (212 viagens). Essas bases serão designadas por M_{602} e M_{212}, respectivamente;
- uma base de dados conseguida junto a empresa Cidade Verde, que realiza o transporte coletivo entre as cidades de Maringá e Sarandi, estado do Paraná. Esta base de 657 viagens será referenciada por CV_{657}.

Os testes foram efetuados em um computador equipado com microprocessador AMD Athlon XP 2700 e 1 GB de memória RAM.

Como foi dito anteriormente, tanto APP_1 quanto APP_2 trabalham sobre uma solução inicial obtida utilizando-se o algoritmo de Dijkstra para efetuar a divisão das escalas de veículo (vide seção 2.2.3a). Assim, para efeito de comparação, os resultados obtidos com a aplicação de APP_1 e APP_2 serão confrontados com aqueles obtidos com a aplicação do algoritmo de Dijkstra.

Os problemas de cobertura gerados com a aplicação das três abordagens (algoritmo de Dijkstra, APP_1 e APP_2) sobre as três bases de dados citadas (M_{602}, M_{212} e CV_{657}) foram submetidos a dois algoritmos de resolução:

- o primeiro é um algoritmo genético, proposto em [4], e baseia-se em uma metodologia relativamente simples de seleção e reprodução dos planos de escalas que fazem parte da população; será designado nas tabelas de resultados por AG_1;
- o segundo, também um algoritmo genético, foi proposto em [7] e utiliza um algoritmo guloso randomizado para a geração de sua população inicial, além de propor novos operadores de cruzamento e mutação, que foram incorporados ao algoritmo para intensificar a busca; será designado por AG_2.

4.1 Experimento 1

Resultados obtidos sobre a base de dados M_{602}.

Para todos os problemas de coberturas gerados a partir desta base, foram utilizados os seguintes valores para os parâmetros de entrada de:

***AG*$_1$** Tamanho da população: 800 indivíduos; taxa de mutação: 3%;

***AG*$_2$** Tamanho da população: 400 indivíduos; ciclos após convergência: 2000.

Geração do PCC usando algoritmo de Dijkstra

a) Características do problema de cobertura gerado

partições	escalas	$\delta(S)$	$\mu(S)$
177	5451	18,01	41,96

b) Resolução do PCC usando AG_1

it.	cond.	custo	tempo
54085	90	40048	02:35
55706	90	(*)39787	02:41
53857	90	39912	02:49
51622	90	39823	02:29

c) Resolução do PCC usando AG_2

it.	cond.	custo	tempo
44800	91	(*)38408	02:05
43400	91	38435	01:10
47050	91	38439	01:52
44350	91	38414	00:49

Tabela 4.1: Resultados obtidos com algoritmo de Dijkstra sobre a base M_{602}

onde:

partições = número de partições (linhas) do problema gerado;

escalas = número de escalas (colunas) do problema gerado;

***δ*(*S*)** = valor de $\delta(S)$ para o problema gerado;

***μ*(*S*)** = valor de $\mu(S)$ para o problema gerado;

it. = número de iterações efetuadas pelo algoritmo de resolução;

cond. = número de condutores utilizados na solução gerada;

custo = custo da solução gerada;

tempo = tempo de processamento gasto pelo algoritmo de resolução;

Geração do PCC usando APP_1

a) Parametrizações utilizadas e características dos problemas de cobertura gerados

	Parametrização					Resultados				
PCC	T	CF	TL	MT	$\%perda_{max}$	partições	escalas	$\delta(S)$	$\mu(S)$	custo
1.1	1000	0,95	40	00:01	0,05	179	6276	17,23	44,72	33,87
1.2	1000	0,95	40	00:01	0,15	178	5886	16,94	43,18	34,27
1.3	1000	0,95	120	00:04	0,25	178	5376	15,99	42,18	33,09
1.4	1000	0,95	120	00:01	0,25	177	5494	17,11	42,72	35,05

b) Resolução do PCC usando AG_1

PCC	it.	cond.	custo	tempo
1.1	51596	89	39383	02:38
1.1	50954	89	39065	02:38
1.1	62585	90	39652	03:09
1.1	51574	89	39285	02:39
1.2	39731	90	39731	02:38
1.2	70044	90	39650	03:18
1.2	148124	89	(*)38478	06:42
1.2	57103	90	39609	02:44
1.3	49615	89	40016	02:22
1.3	55779	89	39787	02:38
1.3	75717	89	39033	03:01
1.3	50240	89	39760	02:24
1.4	49782	89	39820	02:19
1.4	97099	89	38979	04:24
1.4	76299	89	39191	03:32
1.4	64415	89	39562	03:02

c) Resolução do PCC usando AG_2

PCC	it.	cond.	custo	tempo
1.1	56950	90	38007	01:47
1.1	40500	90	38093	01:54
1.1	45450	91	38316	02:32
1.1	54150	90	38075	01:07
1.2	56350	91	38449	02:30
1.2	76750	91	38359	01:47
1.2	67700	91	38406	02:19
1.3	63200	90	38127	01:46
1.3	44250	90	38120	01:52
1.3	55350	91	38337	02:38
1.3	74800	90	38215	02:20
1.4	53800	90	38148	01:40
1.4	55450	89	(*)37826	00:41
1.4	54000	90	38059	01:35
1.4	54450	90	38063	01:20

Tabela 4.2: Resultados obtidos com APP_1 sobre a base M_{602}

onde:

PCC = rótulo para o problema gerado na tabela a), utilizado para relacionar este com as respectivas soluções encontradas pelos algoritmos de resolução nas tabelas b) e c);

T*, *CF*, *TL*, *MT = valores para os parâmetros de entrada do algoritmo de pré-processamento. Respectivamente: temperatura, fator de resfriamento, iterações por temperatura, tempo limite (parâmetros básicos da meta-heurística *Simulated Annealing*).

$\%perda_{max}$ = valor para o parâmetro de entrada $\%perda_{max}$ do algoritmo de pré-processamento;

partições = número de partições (linhas) do problema gerado;

escalas = número de escalas (colunas) do problema gerado;

$\boldsymbol{\delta(S)}$ = valor de $\delta(S)$ para o problema gerado;

$\boldsymbol{\mu(S)}$ = valor de $\mu(S)$ para o problema gerado;

it. = número de iterações efetuadas pelo algoritmo de resolução;

cond. = número de condutores utilizados na solução gerada;

custo = custo da solução gerada;

tempo = tempo de processamento gasto pelo algoritmo de resolução;

Resolução do PCC usando $\boldsymbol{AG_1}$ Comparando o custo mínimo obtido com os PCC's gerados utilizando APP_1, 38478 minutos de trabalho pago, com o mínimo obtido na tabela 4.1b (PCC gerado com algoritmo de Dijkstra - 39787 minutos), verifica-se uma redução de 3,3% no custo da solução. Verifique ainda que 12 das 16 soluções geradas aqui (ou seja, 75%) apresentam custo menor do que o mínimo 39787 obtido em 4.1b.

Ainda, na maioria dos casos, as soluções aqui apresentadas utilizam um condutor a menos do que as soluções obtidas com o PCC gerado a partir do algoritmo de Dijkstra.

Resolução do PCC usando $\boldsymbol{AG_2}$ De maneira semelhante, comparando o custo mínimo obtido com os PCC's gerados utilizando APP_1, 37826 minutos de trabalho pago, com o mínimo obtido em 4.1c (PCC gerado com algoritmo de Dijkstra - 38408 minutos), verifica-se uma redução de 1,5% no custo da solução. Verifique ainda que 14 das 16 soluções geradas aqui (ou seja, 87,5%) apresentam custo menor do que o mínimo 38408 obtido em 4.1c.

Ainda, na maioria dos casos, as soluções aqui apresentadas utilizam um condutor a menos do que as soluções obtidas com o PCC gerado a partir do algoritmo de Dijkstra; sendo que o PCC 1.4 chegou a ser resolvido com dois condutores a menos.

Geração do PCC usando APP_2

a) Parametrizações utilizadas e características dos problemas de cobertura gerados

	Parametrização					Resultados		
PCC	T	CF	TL	MT	$\%perda_{max}$	partições	escalas	custo
2.1	500000	0,95	60	00:02	0,01	177	5453	48452,36
2.2	500000	0,95	100	00:04	0,01	178	5383	47947,24
2.3	500000	0,95	60	00:02	0,1	177	5616	49108,28
2.4	500000	0,95	100	00:05	0,1	177	5611	48241,65

b) Resolução do PCC usando AG_1

PCC	it.	cond.	custo	tempo
2.1	56052	89	39814	02:40
2.1	58494	89	39125	02:42
2.1	82078	90	39297	03:56
2.1	56091	89	39069	02:45
2.2	61149	88	39307	02:54
2.2	70914	89	39156	03:28
2.2	54187	89	39953	02:45
2.2	53054	89	39977	02:41
2.3	51643	90	40153	02:40
2.3	49245	90	39845	02:26
2.3	38837	89	39735	01:46
2.3	63440	90	39491	03:12
2.4	58987	89	39028	02:41
2.4	48002	88	39659	02:15
2.4	68944	88	(*)38886	03:09
2.4	55507	88	39038	02:18

c) Resolução do PCC usando AG_2

PCC	it.	cond.	custo	tempo
2.1	61550	90	37994	02:21
2.1	45750	90	38014	02:04
2.1	54950	90	38015	01:58
2.1	57250	90	37988	03:29
2.2	66200	90	38221	03:04
2.2	66700	89	38291	01:14
2.2	49600	90	38294	03:09
2.2	55950	91	38481	03:27
2.3	73150	90	37944	04:03
2.3	68000	89	(*)37743	00:50
2.3	70100	89	37870	00:52
2.3	61350	90	37927	01:58
2.4	36750	90	38032	02:08
2.4	76400	89	37762	05:13
2.4	50150	90	38065	03:00
2.4	64700	89	37847	05:03

Tabela 4.3: Resultados obtidos com APP_2 sobre a base M_{602}

Resolução do PCC usando AG_1 Comparando o custo mínimo obtido com os PCC's gerados utilizando APP_2, 38886 minutos de trabalho pago, com o mínimo obtido na tabela 4.1b (PCC gerado com algoritmo de Dijkstra - 39787 minutos), verifica-se uma redução de 2,2% no custo da solução. Verifique ainda que 10 das 16 soluções geradas aqui (ou seja, 62,5%) apresentam custo menor do que o mínimo 39787 obtido na tabela 4.1b.

Ainda, na maioria dos casos, as soluções aqui apresentadas utilizam pelo menos um condutor a menos do que as soluções obtidas com o PCC gerado a partir do algoritmo de Dijkstra; sendo que o PCC's 2.2 e 2.4 chegaram a ser resolvidos com dois condutores a menos (90 contra 88).

Resolução do PCC usando AG_2 De maneira semelhante, comparando o custo mínimo obtido com os PCC's gerados utilizando APP_2, 37743 minutos de trabalho pago, com o mínimo obtido na tabela 4.1c (PCC gerado com algoritmo de Dijkstra - 38408 minutos), verifica-se uma redução de 1,7% no custo da solução. Verifique

ainda que 15 das 16 soluções geradas aqui (ou seja, 93%) apresentam custo menor do que o mínimo 38408 obtido em 4.1c.

Ainda, na maioria dos casos, as soluções aqui apresentadas utilizam, no mínimo, um condutor a menos do que as soluções obtidas com o PCC gerado a partir do algoritmo de Dijkstra; sendo que os PCC's 1.2, 1.3 e 1.4 chegaram a ser resolvidos com dois condutores a menos (89 contra 91).

-

Finalizando a análise do desempenho dos algoritmos de pré-processamento propostos, nos testes sobre a base de dados M_{602}, verifica-se que ambos os algoritmos (APP_1 e APP_2) conseguiram melhorar os resultados obtidos pelos algoritmos de resolução do PCC, AG_1 e AG_2 - em comparação com os problemas de cobertura gerados pelo algoritmo de Dijkstra. A redução do custo das soluções variou entre 1,5 e 3,3%, utilizando-se até dois condutores a menos.

4.2 Experimento 2

Resultados obtidos sobre a base de dados M_{212}

Para todos os problemas de coberturas gerados a partir desta base, foram utilizados os seguintes valores para os parâmetros de entrada de:

***AG*$_1$** Tamanho da população: 250 indivíduos; taxa de mutação: 3%;

***AG*$_2$** Tamanho da população: 150 indivíduos; ciclos após convergência: 800.

Geração do PCC usando algoritmo de Dijkstra

a) Características do problema de cobertura gerado

partições	escalas	$\delta(S)$	$\mu(S)$
66	528	4,63	14,18

b) Resolução do PCC usando AG_1

it.	cond.	custo	tempo
6194	33	(*)14589	00:14
5803	33	14922	00:13
5849	33	14731	00:13
6195	33	14658	00:14

c) Resolução do PCC usando AG_2

it.	cond.	custo	tempo
9150	33	14492	00:01
8150	33	14503	00:01
7300	33	14537	00:01
14600	33	(*)14416	00:02

Tabela 4.4: Resultados obtidos com algoritmo de Dijkstra sobre a base M_{212}

Geração do PCC usando APP_1

a) Parametrizações utilizadas e características dos problemas de cobertura gerados

	Parametrização					Resultados				
PCC	T	CF	TL	MT	$\%perda_{max}$	partições	escalas	$\delta(S)$	$\mu(S)$	custo
3.1	300	0,99	30	00:01	0,01	65	496	4,31	13,69	9,51
3.2	300	0,99	30	00:01	0,05	64	473	4,12	13,27	9,23
3.3	500	0,99	40	00:01	0,1	63	454	3,99	12,84	9,06
3.4	500	0,99	40	00:01	0,2	62	412	3,76	12,24	8,75

b) Resolução do PCC usando AG_1

PCC	it.	cond.	custo	tempo
3.1	5402	32	14934	00:12
3.1	4053	32	14692	00:09
3.1	4110	32	14377	00:09
3.1	5529	32	14510	00:13
3.2	4000	32	14614	00:12
3.2	5559	32	14708	00:12
3.2	5535	31	(*)14359	00:15
3.2	3944	31	14422	00:04
3.3	5077	31	14493	00:11
3.3	5620	31	14715	00:12
3.3	5496	32	14469	00:12
3.3	4773	31	14576	00:10
3.4	5252	33	15023	00:12
3.4	7116	33	15163	00:16
3.4	4264	33	15269	00:10
3.4	4014	33	15418	00:09

c) Resolução do PCC usando AG_2

PCC	it.	cond.	custo	tempo
3.1	13150	33	14547	00:01
3.1	17500	33	14453	00:02
3.1	11150	33	14570	00:01
3.1	10200	33	14608	00:01
3.2	20100	32	14426	00:02
3.2	15600	32	14488	00:02
3.2	18600	32	14392	00:02
3.2	15400	32	14454	00:02
3.3	11750	33	14543	00:01
3.3	11150	33	14556	00:01
3.3	13800	32	(*)14362	00:02
3.3	16850	33	14535	00:02
3.4	13400	34	15076	00:01
3.4	11200	34	15071	00:01
3.4	18950	34	15023	00:02
3.4	10750	34	15072	00:01

Tabela 4.5: Resultados obtidos com APP_1 sobre a base M_{212}

Resolução do PCC usando AG_1 Comparando o custo mínimo obtido com os PCC's gerados utilizando APP_1, 14359 minutos de trabalho pago, com o mínimo obtido na tabela 4.4b (PCC gerado com algoritmo de Dijkstra - 14589 minutos), verifica-se uma redução de 1,5% no custo da solução. Ainda, o pré-processamento efetuado por APP_1 possibilitou ao algoritmo de resolução AG_1 a utilização de até dois condutores a menos na solução.

Pode-se observar aqui que, pelos resultados obtidos com o PCC 3.4, relaxar demais o parâmetro $\%perda_{max}$ não é vantajoso, pois os problemas de cobertura gerados com $\%perda_{max} \leq 0.1$ (PCC's 3.1, 3.2 e 3.3) foram resolvidos com custo (e número de condutores) sempre inferior ao PCC 3.4 ($\%perda_{max} = 0.2$).

Resolução do PCC usando AG_2 Comparando o custo mínimo obtido com os PCC's gerados utilizando APP_1, 14362 minutos de trabalho pago, com o mínimo obtido na tabela 4.4c (PCC gerado com algoritmo de Dijkstra - 14416 minutos), verifica-se uma redução de 0,3% no custo da solução. Apesar da redução obtida ter sido muito

pequena, deve observar que o pré-processamento efetuado por APP_1 possibilitou ao algoritmo de resolução AG_2 obter soluções que utilizam um condutor a menos (32) do que as obtidas sobre o problema gerado com algoritmo de Dijkstra.

Assim como AG_1, o algoritmo AG_2 obteve resultados muito inferiores sobre o problema de cobertura 3.4 - gerado com um valor relativamente alto para o parâmetro $\%perda_{max}$ $(0,2\%)$ -, com relação aos outros três problemas de cobertura (3.1, 3.2 e 3.3); cabendo aqui então a mesma observação feita na seção anterior sobre o parâmetro $\%perda_{max}$.

Geração do PCC usando APP_2

a) Parametrizações utilizadas e características dos problemas de cobertura gerados

	Parametrização					Resultados		
PCC	T	CF	TL	MT	$\%perda_{max}$	partições	escalas	custo
4.1	200000	0,99	40	00:01	0,01	63	449	15352
4.2	200000	0,99	40	00:01	0,05	63	444	15389
4.3	200000	0,99	40	00:01	0,1	62	411	15420
4.4	300000	0,99	40	00:01	0,1	62	434	14755

b) Resolução do PCC usando AG_1

PCC	it.	cond.	custo	tempo
4.1	4135	31	14005	00:10
4.1	3408	31	13902	00:08
4.1	4081	31	(*)13740	00:09
4.1	4954	31	14021	00:12
4.2	3691	32	14568	00:07
4.2	4308	31	14025	00:08
4.2	6243	32	14338	00:07
4.2	3406	31	14319	00:06
4.3	6476	31	14042	00:13
4.3	5489	32	14325	00:11
4.3	4603	31	14029	00:09
4.3	3625	31	14257	00:06
4.4	5541	31	14070	00:12
4.4	5566	31	14101	00:12
4.4	4451	31	14380	00:10
4.4	5868	31	14073	00:11

c) Resolução do PCC usando AG_2

PCC	it.	cond.	custo	tempo
4.1	11650	31	(*)13729	00:01
4.1	12500	31	13764	00:01
4.1	7850	31	13778	00:01
4.1	8650	31	13765	00:01
4.2	14250	32	14151	00:01
4.2	6950	32	14158	00:01
4.2	9650	32	14121	00:01
4.2	10050	32	14118	00:01
4.3	11150	31	13988	00:01
4.3	6200	32	14109	00:01
4.3	8350	31	14018	00:01
4.3	8400	31	14059	00:01
4.4	13450	31	13874	00:01
4.4	14450	31	13842	00:01
4.4	18350	31	13826	00:02
4.4	6900	31	13997	00:01

Tabela 4.6: Resultados obtidos com APP_2 sobre a base M_{212}

Resolução do PCC usando AG_1 Comparando o custo mínimo obtido com os PCC's gerados utilizando APP_2, 13740 minutos de trabalho pago, com o mínimo obtido em 4.4b (PCC gerado com algoritmo de Dijkstra - 14359 minutos), verifica-se uma redução de 4,3% no custo da solução. Ainda, o pré-processamento efetuado por

APP_2 possibilitou ao algoritmo de resolução AG_1 a utilização de até dois condutores a menos na solução.

Deve ser observado ainda que *todas* as soluções aqui obtidas tem custo menor do que o mínimo 14731 apresentado pelo algoritmo de Dijkstra.

Resolução do PCC usando AG_2 Comparando o custo mínimo obtido com os PCC's gerados utilizando APP_2, 13729 minutos de trabalho pago, com o mínimo obtido em 4.4c (PCC gerado com algoritmo de Dijkstra - 14416 minutos), verifica-se uma redução de 4,7% no custo da solução. Ainda, o pré-processamento efetuado por APP_2 possibilitou ao algoritmo de resolução AG_2 obter soluções que utilizam até dois condutores a menos do que as obtidas sobre o problema gerado com algoritmo de Dijkstra.

Da mesma forma que AG_1, *todas* as soluções obtidas pelo algoritmo AG_2 sobre os problemas de cobertura gerados por APP_2 apresentam custos menores do que as soluções obtidas sobre o problema gerado pelo algoritmo de Dijkstra.

-

Finalizando a análise do desempenho dos algoritmos de pré-processamento propostos, nos testes sobre a base de dados M_{212}, verifica-se que ambos os algoritmos (APP_1 e APP_2) conseguiram melhorar significativamente os resultados obtidos pelos algoritmos de resolução do PCC, AG_1 e AG_2 - em comparação com os problemas de cobertura gerados pelo algoritmo de Dijkstra. A redução do custo das soluções chegou a 4,7%, utilizando-se até dois condutores a menos.

4.3 Experimento 3

Resultados obtidos sobre a base de dados CV_{657}.

Para todos os problemas de coberturas gerados a partir desta base, foram utilizados os seguintes valores para os parâmetros de entrada de:

***AG*$_1$** Tamanho da população: 600 indivíduos; taxa de mutação: 3%;

***AG*$_2$** Tamanho da população: 300 indivíduos; ciclos após convergência: 1200.

Geração do PCC usando algoritmo de Dijkstra

a) Características do problema de cobertura gerado

partições	escalas	$\delta(S)$	$\mu(S)$
125	1196	12,45	15,13

b) Resolução do PCC usando AG_1

it.	cond.	custo	tempo
18149	87	40090	00:19
20405	87	40220	00:19
19782	87	40056	00:36
20274	87	(*)39964	00:37

c) Resolução do PCC usando AG_2

it.	cond.	custo	tempo
17450	87	(*)38983	00:18
21450	87	38983	00:20
19800	87	38983	00:18
22550	87	38983	00:16

Tabela 4.7: Resultados obtidos com algoritmo de Dijkstra sobre a base CV_{657}

Geração do PCC usando APP_1

a) Parametrizações utilizadas e características dos problemas de cobertura gerados

	Parametrização					Resultados				
PCC	T	CF	TL	MT	$\%perda_{max}$	partições	escalas	$\delta(S)$	$\mu(S)$	custo
5.1	600	0,95	60	00:01	0,01	139	2607	12,7	31,81	27,01
5.2	600	0,95	120	00:01	0,01	144	3473	14,37	35,23	28,66
5.3	600	0,95	60	00:01	0,1	144	3210	13,01	35,21	25,97
5.4	600	0,95	120	00:01	0,1	143	3227	14,19	33,64	29,4

b) Resolução do PCC usando AG_1

PCC	it.	cond.	custo	tempo
5.1	42155	83	38431	00:52
5.1	35107	83	38032	00:39
5.1	33957	83	38716	00:31
5.1	34137	83	38350	01:26
5.2	34646	81	37982	00:39
5.2	32644	82	37776	00:51
5.2	32799	81	(*)37193	00:58
5.2	34426	82	37698	00:58
5.3	29958	81	37576	01:15
5.3	33257	81	37927	01:31
5.3	38911	81	37423	01:44
5.3	50550	81	37351	02:14
5.4	33729	81	37659	01:36
5.4	39209	81	37776	01:44
5.4	29074	81	37922	01:12
5.4	28587	81	38094	01:18

c) Resolução do PCC usando AG_2

PCC	it.	cond.	custo	tempo
5.1	47700	83	37117	00:37
5.1	30500	83	37112	00:30
5.1	46300	83	37063	01:12
5.1	41350	83	37067	00:37
5.2	58050	82	36606	02:25
5.2	57150	82	36619	00:42
5.2	57150	82	36677	00:56
5.2	74600	82	36628	00:59
5.3	51650	82	36712	01:15
5.3	58100	82	36715	00:43
5.3	55250	81	(*)36382	01:19
5.3	53850	81	36384	00:56
5.4	71750	81	36630	01:44
5.4	63100	81	36647	02:05
5.4	65750	81	36683	00:54
5.4	65150	81	36696	01:03

Tabela 4.8: Resultados obtidos com APP_1 sobre a base CV_{657}

Características das instâncias do PCC geradas Nota-se pelos resultados apresentados aqui que, ao contrário dos problemas de cobertura gerados para as duas bases de teste anteriores, o número de partições dos problemas gerados pelo algoritmo de pré-processamento é consideravelmente maior, em relação ao problema de cobertura gerado pelo algoritmo de Dijkstra. Em termos percentuais, o acréscimo no número de partições girou em torno de 15%.

Resolução do PCC usando AG_1 Comparando o custo mínimo obtido com os PCC's gerados utilizando APP_1, 37193 minutos de trabalho pago, com o mínimo obtido em 4.7b (PCC gerado com algoritmo de Dijkstra - 39964 minutos), verifica-se uma redução de 6,9% no custo da solução. Ainda, o pré-processamento efetuado por APP_1 possibilitou ao algoritmo de resolução AG_1 a utilização de até seis condutores a menos nas soluções.

Note ainda que *todas* as soluções aqui obtidas apresentam custo menor do que o mínimo (39964 minutos) obtido com a aplicação de AG_1 sobre o problema de

cobertura gerado com o algoritmo de Dijkstra.

Resolução do PCC usando AG_2 Comparando o custo mínimo obtido com os PCC's gerados utilizando APP_1, 36382 minutos de trabalho pago, com o mínimo obtido em 4.7c (PCC gerado com algoritmo de Dijkstra - 38983 minutos), verifica-se uma redução de 6,6% no custo da solução. Ainda, o pré-processamento efetuado por APP_1 possibilitou ao algoritmo de resolução AG_2 obter soluções que utilizam até seis condutores a menos do que as obtidas sobre o problema gerado com algoritmo de Dijkstra.

Ainda, assim como o algoritmo AG_1 na seção anterior, *todas* as soluções aqui obtidas apresentam custo menor do que o mínimo (38983 minutos) obtido com a aplicação de AG_2 sobre o problema de cobertura gerado com o algoritmo de Dijkstra.

Geração do PCC usando APP_2

a) Parametrizações utilizadas e características dos problemas de cobertura gerados

	Parametrização					Resultados		
PCC	T	CF	TL	MT	$\%perda_{max}$	partições	escalas	custo
6.1	2000000	0,95	40	00:01	0,01	128	2040	48953
6.2	2000000	0,95	40	00:01	0,05	138	3581	51349
6.3	2000000	0,95	80	00:01	0,05	130	2840	49749
6.4	2000000	0,95	60	00:01	0,1	132	2228	48842

b) Resolução do PCC usando AG_1

PCC	it.	cond.	custo	tempo
6.1	20123	83	37658	00:47
6.1	23365	83	37487	00:54
6.1	30440	83	37488	01:10
6.1	22592	83	37831	00:53
6.2	24809	82	38145	01:00
6.2	24102	82	38214	00:59
6.2	32031	82	37606	01:17
6.2	39510	82	37653	01:33
6.3	19417	81	36833	00:48
6.3	18886	81	37134	00:47
6.3	22093	81	37195	00:55
6.3	23031	81	(*)36677	00:57
6.4	21906	83	38026	00:53
6.4	24825	83	37779	01:05
6.4	22421	83	38054	00:51
6.4	22111	83	38609	00:50

c) Resolução do PCC usando AG_2

PCC	it.	cond.	custo	tempo
6.1	25700	83	36900	00:29
6.1	18550	83	36952	00:33
6.1	18600	83	36957	00:44
6.1	21700	83	36952	00:40
6.2	21400	82	36738	00:53
6.2	30600	82	36635	01:08
6.2	27600	82	36680	01:07
6.2	28250	82	36694	00:56
6.3	25950	82	36384	00:59
6.3	22250	81	(*)36074	00:25
6.3	33600	82	36402	01:26
6.3	24900	82	36402	00:29
6.4	24650	83	37235	00:54
6.4	25600	83	37235	00:40
6.4	26100	83	37237	00:47
6.4	26450	83	37234	01:00

Tabela 4.9: Resultados obtidos com APP_2 sobre a base CV_{657}

Resolução do PCC usando AG_1 Comparando o custo mínimo obtido com os PCC's gerados utilizando APP_2, 36677 minutos de trabalho pago, com o mínimo obtido

em 4.7b (PCC gerado com algoritmo de Dijkstra - 39964 minutos), verifica-se uma redução de 8,2% no custo da solução. Ainda, o pré-processamento efetuado por APP_2 possibilitou ao algoritmo de resolução AG_1 a utilização de até seis condutores a menos nas soluções.

Note ainda que *todas* as soluções aqui obtidas apresentam custo menor do que o mínimo (39964 minutos) obtido com a aplicação de AG_1 sobre o problema de cobertura gerado com o algoritmo de Dijkstra.

Resolução do PCC usando AG_2 Comparando o custo mínimo obtido com os PCC's gerados utilizando APP_2, 36074 minutos de trabalho pago, com o mínimo obtido em 4.7c (PCC gerado com algoritmo de Dijkstra - 38983 minutos), verifica-se uma redução de 7,4% no custo da solução. Ainda, o pré-processamento efetuado por APP_2 possibilitou ao algoritmo de resolução AG_2 obter soluções que utilizam até seis condutores a menos do que as obtidas sobre o problema gerado com algoritmo de Dijkstra.

Ainda, assim como o algoritmo AG_1 na seção anterior, *todas* as soluções aqui obtidas apresentam custo menor do que o mínimo (38983 minutos) obtido com a aplicação de AG_2 sobre o problema de cobertura gerado com o algoritmo de Dijkstra.

-

Finalizando a análise do desempenho dos algoritmos de pré-processamento propostos, nos testes sobre a base de dados CV_{657}, verifica-se que ambos os algoritmos (APP_1 e APP_2) conseguiram melhorar significativamente os resultados obtidos pelos algoritmos de resolução do PCC, AG_1 e AG_2 - em comparação com os problemas de cobertura gerados pelo algoritmo de Dijkstra. A redução do custo das soluções chegou a 8,2%, utilizando-se até seis condutores a menos.

4.4 Gráficos de execução

Para exemplificar o comportamento dos algoritmos de pré-processamento APP_1 e APP_2, abaixo são apresentados os gráficos de execução dos mesmos em cada um dos casos de teste.

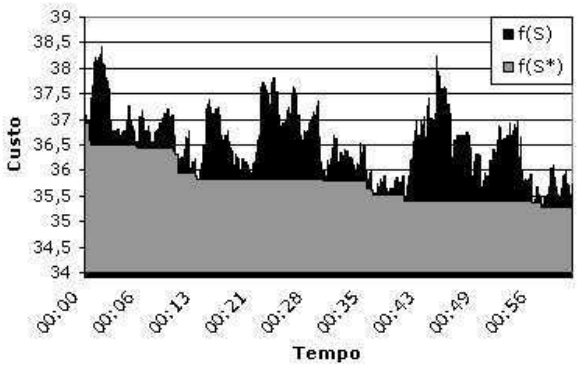

Figura 4.1: Gráfico de execução do algoritmo APP_1 sobre a base de teste M_{602}

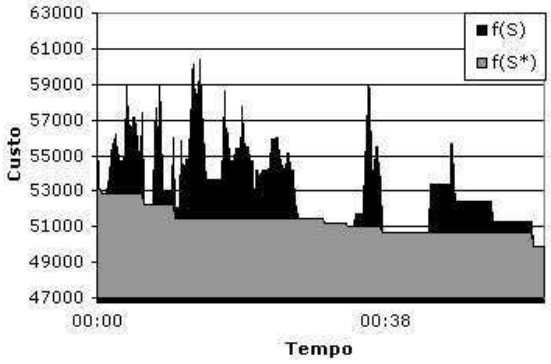

Figura 4.2: Gráfico de execução do algoritmo APP_2 sobre a base de teste M_{602}

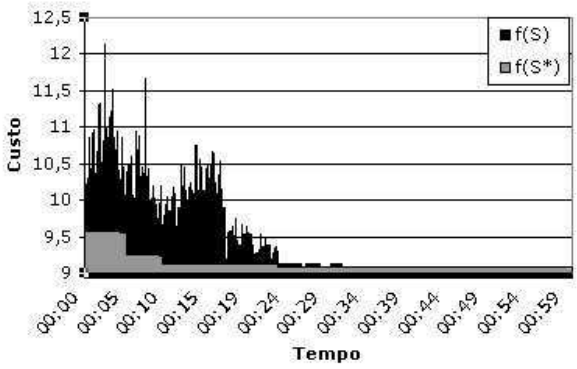

Figura 4.3: Gráfico de execução do algoritmo APP_1 sobre a base de teste M_{212}

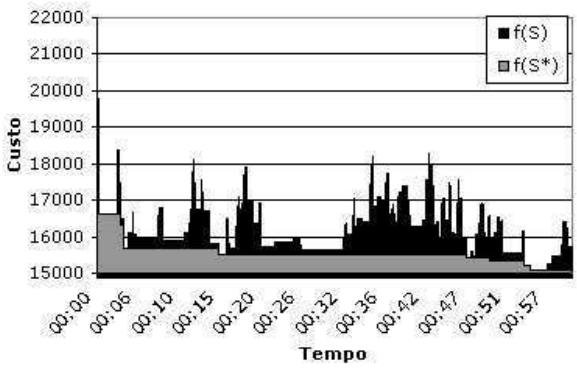

Figura 4.4: Gráfico de execução do algoritmo APP_2 sobre a base de teste M_{212}

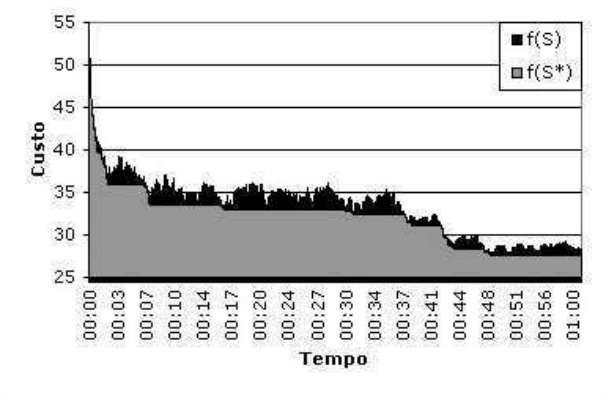

Figura 4.5: Gráfico de execução do algoritmo APP_1 sobre a base de teste CV_{657}

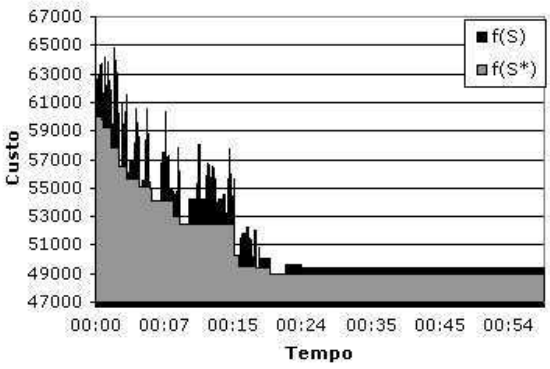

Figura 4.6: Gráfico de execução do algoritmo APP_2 sobre a base de teste CV_{657}

5 Conclusão

Considerando os dados e análises apresentados no capítulo anterior, conclui-se que:

- ambos os algoritmos de pré-processamento propostos, APP_1 e APP_2, apresentaram resultados significativos, possibilitando a dois diferentes algoritmos de resolução, AG_1 e AG_2, a obtenção de soluções com custos menores aos obtidos com a aplicação dos mesmos sobre os PCC's gerados apenas com o algoritmo de Dijkstra; isto em três bases de teste distintas. A redução no custo das soluções esteve entre 1 e 4% na maior parte dos testes, chegando a mais de 8% em alguns casos.

- os melhores resultados são obtidos após um ajuste nos parâmetros de entrada do algoritmo (temperatura, fator de resfriamento, etc.); pode-se observar que, até um certo ponto - que dependerá das características da instância do PEC - o incremento do parâmetro $\%perda_{max}$ tem um efeito positivo sobre os algoritmos de pré-processamento. Ultrapassado este ponto, os resultados apresentados na resolução do PCC tendem a ser inferiores.

- A utilização da meta-heurística *Simulated Annealing* mostrou-se satisfatória, permitindo a obtenção de resultados positivos com pouco tempo de processamento (um minuto, na maior parte dos testes). Por meio dos gráficos de execução, percebe-se que o comportamento do algoritmo está em conformidade do que se espera do *Simulated Annealing*: após o estabelecimento de parâmetros de entrada adequados, obtém-se uma convergência gradual para custos cada vez menores, enquanto o algoritmo explora o espaço de soluções por meio do mecanismo de aceitação probabilística de soluções vizinhas cujo custo é maior do que a solução atual.

Além disto, pode-se concluir que é possível a obtenção de melhores resultados para o problema de escalonamento de condutores, não apenas trabalhando-se sobre os algoritmos de resolução do problema de cobertura de colunas (como propõem a maior parte dos trabalhos sobre o assunto), mas preocupando-se também com métodos para a *construção* deste problema de cobertura.

Os resultados obtidos neste trabalho incentivam a pesquisa de novas metodologias para o pré-processamento do PCC na resolução do PEC. Propõe-se então, como trabalhos futuros:

- a experimentação de outras metodologias para o pré-processamento, baseadas ou não em meta-heurísticas;

- a investigação de novos critérios de direcionamento para o pré-processamento, baseados ou não em características das instâncias do PCC que são geradas durante o processo;
- deve-se considerar que, sendo o PEC um problema de natureza combinatorial, qualquer metodologia para o pré-processamento pode tornar-se ineficiente para instâncias do PEC a partir de um certo tamanho; logo, alternativas para tratar esta dificuldade da melhor forma devem também merecer atenção em trabalhos futuros.

Este trabalho então cumpre o seu objetivo: propor uma metodologia para efetuar este pré-processamento do PEC, que utiliza heurísticas para selecionar, dentre os problemas de cobertura que possam representar este PEC, aqueles que permitam aos algoritmos de resolução a obtenção de melhores resultados.

Referências

[1] Beasley, J. E., and Chu, P. C. A genetic algorithm for the set covering problem. *Technical Report, The Management School Imperial College. London, England* (1994), 1–16.

[2] Bennington, G., and Rebibio, K. Overview of RUCUS vehicle scheduling program (BLOCKS). Preprint: Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services.

[3] Caprara, A., Fischetti, M., and Toth, P. A heuristic algorithm for the set covering problem. *Technical Report, DEIS, University of Bologna, Italy* (1995), 1–24.

[4] Castro, E. C. Escalonamento de veículos e condutores em empresas de transporte urbano rodoviário. *Trabalho de Graduação. Universidade Estadual de Maringá - UEM* (2003).

[5] Castro, J. R. P. Geração de escalas de trabalho em transporte urbano de passageiros: uma técnica heurística para a formulação do problema de cobertura de conjuntos. Master's thesis, Universidade Federal de Santa Catarina - UFSC, 2002.

[6] Cerny, V. A thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Tech. rep., 1985.

[7] Constantino, A. A., Reis, P., Mendonça Neto, C. F. X. D., and Figueiredo, M. F. Aplicação de algoritmos genéticos ao problema de cobertura de conjunto. vol. 1, XXXV SBPO - Simpósio Brasileiro De Pesquisa Operacional, 2003, Natal. Anais, pp. 1–10.

[8] Constantino, A. C. *Otimização de Escala de Condutores de Trem: Sequenciamento de Tarefas e Alocação baseada em Preferência Declarada.* PhD thesis, Universidade Federal de Santa Catarina - UFSC, 1997.

[9] Daduna, J. R., and Mojsilovic, M. Computer-aided vehicle and duty scheduling using HOT programme system. *Computer-Aided Transit Scheduling. Berlin Springer-Verlag* (1988), 133–146.

[10] Fores, S. Column generation approaches to bus driver scheduling, 1996.

[11] Forsyth, P., and Wren, A. An ant system for bus driver scheduling. *7th International Workshop on Computer-Aided Scheduling of Public Transport* (1997).

[12] Jacobs, L. W., and Brusco, M. J. A local-search heuristic for large set-covering problems. *Naval Search Logistics 42* (1994), 1129–1140.

[13] Kirkpatrick, S., Gelatt Jr., C. D., and Vecchi, M. P. Optimization by simulated annealing. *Science 220* (1983), 671–683.

[14] Kwan, A. S. K. Driver scheduling using genetic algorithms with embedded combinatorial traits. Tech. rep., June 18 1997.

[15] Layfield, C. *A Constraint Programming Pre-Processor for Duty Scheduling.* PhD thesis, University of Leeds, England, 2002.

[16] Lessard, R., Rousseau, J.-M., and Dupuis, D. HASTUS I: A mathematical programming approach to the bus driver scheduling problem. *In Wren Computer Scheduling of Public Transport* (1981), 255–266.

[17] Li, J. *Fuzzy evolutionary approaches for bus and rail driver scheduling.* PhD thesis, The University of Leeds School of Computing, 2002.

[18] Lourenço, H. R., Portugal, R., Paixão, J., and M., P. Multiobjective metaheuristics for the bus-driver scheduling problem. *Transportation Science, 35, 3* (2001), 331–343.

[19] Mauri, G. R. Novas heurísticas para o problema de escalonamento de tripulações. Master's thesis, INPE, São José dos Campos, 2004.

[20] Mayerle, S. F. *Um sistema de apoio a decisão para o planejamento operacional de empresas de transporte rodoviário urbano de passageiros.* PhD thesis, Universidade Federal de Santa Catarina - UFSC, 1996.

[21] Mitra, G., and Darby-Dowman, K. CRU-SHED: A computer based bus crew scheduling system using integer programming. R. J.-M., Ed., Computer Scheduling of Public Transport 2, Elsivier.

[22] P., M., and Fritsche, H. INTERPLAN - an interactive program system for crew scheduling and rotering of public transport. Daduna and Wren, Eds., pp. 200–211.

[23] Parker, M. E. Smith, B. M. Two approaches to computer crew scheduling. A. Wren, Ed., pp. 193–221.

[24] Shen, Y., and Kwan, R. S. K. Tabu search for driver scheduling. Tech. rep., 121—135.

[25] Smith, B. M. IMPACS - a bus crew scheduling system using integer programming. *Math. Program. 42*, 1 (1988), 181–187.

[26] Wren, A., Fores, S., Kwan, A., Kwan, R., Parker, M., and Proll, L. A flexible system for scheduling drivers. *Journal of Scheduling 6* (2003), 437–455.

[27] Wren, A., and Rousseau, J. Bus driver scheduling-an overview, 1995.

[28] Wren, A., Smith, B. M., and Miller, A. J. Complementary approaches to crew scheduling. R. J.-M., Ed., Elsivier.

[29] Wren, A. E Gualda, N. D. F. Integrated scheduling of buses and drives. Tech. rep., 1997.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo