unesp UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
CAMPUS DE GUARATINGUETÁ

FÁBIO GAVIÃO AVELINO DE MÉLLO

MÉTODO HEURÍSTICO PARA CRIAÇÃO DE LINHAS DE TRABALHO EM PROBLEMAS DE ESCALONAMENTO DE PESSOAL

Guaratinguetá

2014

FÁBIO GAVIÃO AVELINO DE MÉLLO

MÉTODO HEURÍSTICO PARA CRIAÇÃO DE LINHAS DE TRABALHO EM PROBLEMAS DE ESCALONAMENTO DE PESSOAL

Tese apresentada à Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, para a obtenção do título de Doutor em Engenharia Mecânica na área de Engenharia de Produção.

Orientador: Prof. Dr. Edson Luiz França Senne

Guaratinguetá

2014

Mello, Fabio Gavião Avelino de
M527m Método heurístico para criação de linhas de trabalho em problemas de escalonamento de pessoal / Fabio Gavião Avelino de Mello - Guaratinguetá : [s.n.], 2014
112 f .: il.
Bibliografia: f. 109-112

Tese (doutorado) – Universidade Estadual Paulista, Faculdade de Engenharia de Guaratinguetá, 2014.
Orientador: Prof. Dr Edson Luiz França Senne

1. Algoritmos genéticos 2. Programação heurística 3. Heurística I. Título

CDU 519.712(043)

unesp

UNIVERSIDADE ESTADUAL PAULISTA
CAMPUS DE GUARATINGUETÁ

FÁBIO GAVIÃO AVELINO DE MELLO

ESTA TESE FOI JULGADA ADEQUADA PARA A OBTENÇÃO DO TÍTULO DE
"DOUTOR EM ENGENHARIA MECÂNICA"

PROGRAMA: ENGENHARIA MECÂNICA
ÁREA: GESTÃO E OTIMIZAÇÃO

APROVADA EM SUA FORMA FINAL PELO PROGRAMA DE PÓS-GRADUAÇÃO

Prof. Dr. Edson Cocchieri Botelho
Coordenador

BANCA EXAMINADORA:

Prof. Dr. EDSON LUIZ FRANÇA SENNE
Orientador / UNESP/FEG

Prof. Dr. GALENO JOSÉ DE SENA
UNESP/FEG

Prof. Dr. JOSÉ ROBERTO DALE LUCHE
UNESP/FEG

Prof. Dr. ANIBAL TAVARES DE AZEVEDO
UNICAMP

Kelly Cristina Poldi
Profª. Drª. KELLY CRISTINA POLDI
UNIFESP

Fevereiro de 2014

DADOS CURRICULARES

FÁBIO GAVIÃO AVELINO DE MÉLLO

NASCIMENTO	18.07.1955 – Poços de Caldas, MG
FILIAÇÃO	Geraldo Avelino de Méllo Olinda Gavião Avelino de Méllo
1973/1978	Graduação em Engenharia Elétrica, UNIVERSIDADE FEDERAL DE ITAJUBÁ, Unifei, Itajubá, MG
1984/1985	Especialização em Processamento de Dados Análise de Sistemas, SPEI, Curitiba, PR
1991	Especialização em Sensoriamento Remoto, EUROPEAN SPACE AGENCY, ESA, Roma, Frascati, Italy.
1992/1996	Mestrado em Computação Aplicada, INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS, INPE, São José dos Campos, SP

AGRADECIMENTOS

Ao meu orientador, *Prof. Dr. Edson Luiz França Senne*, que sempre me incentivou e acreditou em mim desde o começo.

Aos meus filhos, Tiago, Danilo e Gabriel, pelo apoio constante e pela compreensão demonstrada desde a época do mestrado, quando muitas vezes tive que subtrair deles um tempo precioso de convívio para me dedicar aos estudos e a pesquisa.

Em especial, ao Gabriel, por ter efetivamente ajudado com boas sugestões na elaboração das apresentações, além de se disponibilizar como ouvinte, na época da apresentação no Chile e durante a qualificação e defesa.

Deus abençoe a todos.

Aos meus pais *in-memoriam*,

Geraldo Avelino de Méllo

Olinda Gavião Avelino de Méllo

MELLO, F. G. A. de. **Método heurístico para a criação de linhas de trabalho em problemas de escalonamento de pessoal**. 2014. 112f. Tese (Doutorado em Engenharia Mecânica) – Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2014.

RESUMO

A presente pesquisa trata do desenvolvimento de um método de solução do problema de construção de linhas de trabalho para a área de escalonamento de pessoal. Foram investigados diversos artigos da área de programação de pessoal com o objetivo de escolher precisamente o tema da pesquisa. Este tema escolhido foi o da construção de linhas de trabalho para empresas de ônibus interurbanos no Brasil. De posse do tema escolhido, foram analisados os métodos usados para formular e resolver o problema. Como resultado foi decidido o uso de uma formulação de cobertura de conjuntos não unicusto para representar o problema em estudo e o uso de um método heurístico para resolver o mesmo. Esta heurística divide a solução do problema em duas fases. A primeira é a fase construtiva, em que o espaço de solução é montado e linhas de trabalho são investigadas e aquelas viáveis são agrupadas formando um conjunto de linhas viáveis e qualificadas. A segunda é a fase de otimização ou de busca local em que um algoritmo evolutivo, baseado em algoritmo genético, irá procurar a melhor solução dentro desse subconjunto de linhas viáveis e qualificadas obtidas na primeira fase. Estes dois procedimentos se repetem até que um critério de parada seja atingido. Testes computacionais foram realizados no sentido de demonstrar a eficácia e eficiência do método proposto. Em seguida, o problema da programação de dias de expediente e de folga, neste trabalho denominado problema de padrões de folga, é formulado e resolvido. Algumas propostas para integrar a solução do problema de criação de linhas de trabalho à do problema de padrões de folga são apresentadas e discutidas.

PALAVRAS-CHAVE: Heurística. Problemas Multiobjetivos. Algoritmo Genético. Heurística evolutiva. Escalonamento de pessoal. *Rostering*.

MELLO, F. G. A. **Heuristic method for creating lines of work for personnel scheduling problems**. 2014. 112f. Doctorate Thesis – Faculdade de Engenharia do Campus de Guaratinguetá, UNESP – Universidade Estadual Paulista, Guaratinguetá, 2014.

ABSTRACT

This thesis deals with the development of a method for solving the problem of construction of lines of work for the application area of personnel scheduling. Several articles were analyzed in order to matching precisely the subject of the research. An in-depth review of the processes used for formulating and solving such a kind of problem in the literature was conducted. As a result, it was decided to formulating the problem as a non unicost set covering problem and to use a heuristic method to solve it. The proposed heuristic is a two-fold algorithm. The first is the construction phase, in which the solution space is scanned and working lines are investigated and those feasible are grouped together forming a set of feasible and qualified lines. The second phase is the optimization or local search in which an evolutionary algorithm based on genetic algorithm will search for the best solution within this set of feasible and qualified lines obtained in the first phase. These two phases are repeated until a stop criterion is reached. Computational tests were performed to demonstrate the effectiveness and efficiency of the proposed method. Then, the tour scheduling problem is addressed in the context of finding shifts of work-days and days-off scheduling. Its resolved by deterministic techniques. Some methods are then discussed on how to integrating both of the solutions of the lines of work and the tour scheduling problems.

KEYWORDS: Heuristic. Multi-objective problems. Genetic Algorithm. Evolutionary heuristic. Personnel scheduling. Rostering.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ACR	*Airline Crew Rostering*, designação do problema de programação de pessoal para a área de aviação.
BIP	*Binary Integer Programming*, designa problemas de programação matemática cujas variáveis independentes são do tipo inteira e binária (0 ou 1).
GRASP	Procedimento de Busca Adaptável Aleatória e Gulosa (*Greedy Random Adaptive Search Procedure*).
MIP	*Mixed Integer Programming*, designação de problemas de programação matemática empregando variáveis independentes inteiras e reais juntas.
NP	Designa uma classe de complexidade cujos problemas não possuem um algoritmo eficiente de solução.
RAM	*Random Access Memory*, designa um tipo de memória em computador cujo acesso aos dados armazenados pode ser realizado de maneira aleatória.

SUMÁRIO

1 INTRODUÇÃO

1.1 DESCRIÇÃO DO PROBLEMA

A área de transporte público possui problemas de gestão e de otimização que oferecem desafios para sua formulação e solução. A existência de grande número de eventos e periódicos especializados mostra o interesse científico para tentar resolvê-los eficaz e eficientemente.

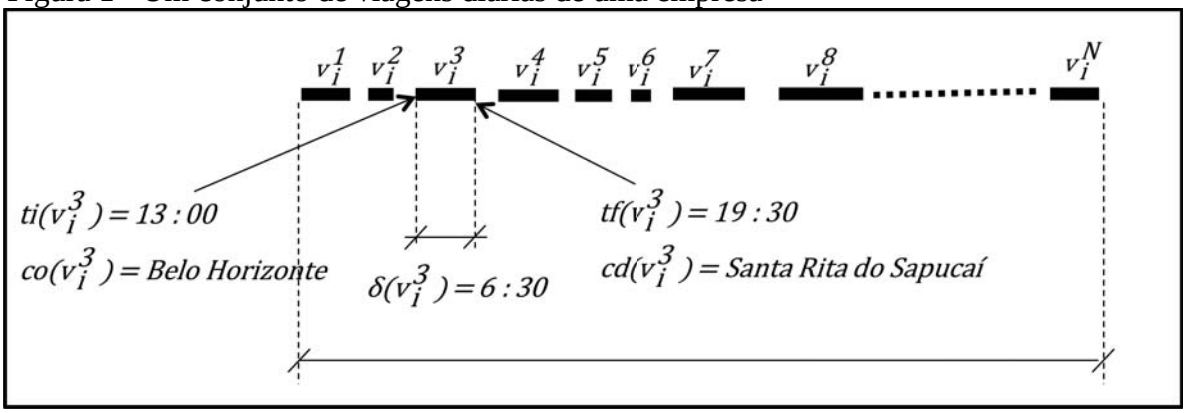
Neste trabalho é estudado o problema de criação de linhas de trabalho para empresas de transporte de passageiros de ônibus interurbanos. Este problema está inserido no contexto dos problemas relativos ao escalonamento de pessoal. A taxonomia adotada neste estudo segue a proposta por Ernst *et al*. (2004a). O termo *rostering* é empregado na literatura para designar o problema genérico de escalonamento de pessoal e será aqui adotado na sua forma em inglês.

O problema estudado pode ser entendido como o da criação de escalas de serviço para motoristas de ônibus para um período levando-se em consideração um conjunto de restrições operacionais e atendendo a alguns objetivos.

Os dados para o desenvolvimento da pesquisa foram baseados em uma empresa brasileira de transporte público interestadual de médio porte, cuja região de atuação compreende os estados de Minas Gerais e de São Paulo. Mediante visitas locais foram obtidas as planilhas contendo os dados de interesse. Em seguida, foram executados procedimentos para a eliminação de redundâncias do tipo duplicação de viagens e algumas inconsistências existentes em alguns horários de viagens.

A seguir, procura-se formalizar o problema em estudo. Uma viagem v_i^k é definida pelos índices i e k. i indica o dia no qual a viagem ocorre e k as viagens daquele dia. Uma viagem consiste na tarefa de conduzir pessoas de um local a outro. Para este estudo, os locais de partida e de chegada se referem, quando não dito o contrário, a cidades distintas, podendo estar situadas em estados diferentes inclusive. Cada motorista só pode ser atribuído a uma única viagem em cada dia de sua escala. Cada viagem é definida precisamente por cinco grandezas: ti (horário de início da viagem), tf (horário de finalização da viagem), co (cidade de origem), cd (cidade de destino) e dv (data da viagem). A Figura 1 ilustra um conjunto de viagens de um dia i genérico de uma empresa. O número de viagens diárias neste estudo é constante e igual a N. As cinco grandezas que caracterizam a viagem estão destacadas na ilustração da Figura 1.

Figura 1 - Um conjunto de viagens diárias de uma empresa



Conforme se pode observar na Figura 1, v_i^1 se refere a primeira viagem de um dia genérico i. v_i^2 a segunda viagem deste mesmo dia e assim sucessivamente até a última viagem do dia. Obviamente neste caso o número de viagens diárias é N. A duração de uma viagem v_i^k é designada por $\delta(v_i^k)$ e pode ser definida pela equação (1) a seguir.

$$\delta(v_i^k)=tf(v_i^k)\text{-}ti(v_i^k) \quad (1)$$

$\delta(v_i^k)$ é medida em horas e minutos. A seguir algumas definições.

Sejam,

$D = \{1, \dots, P\}$ o conjunto dos dias do *rostering*, onde P é o número máximo de dias de programação da escala;

$V_{i \in D} = \{ v_i^k | k \in \{1, \dots, N\} \}$ o subconjunto de viagens contendo as viagens do *i-ésimo* dia.

Cada subconjunto V_i possui N viagens que a empresa precisa cumprir diariamente. N é constante para o período de programação. Cada viagem pode ser definida através da função (2) mostrada em seguida.

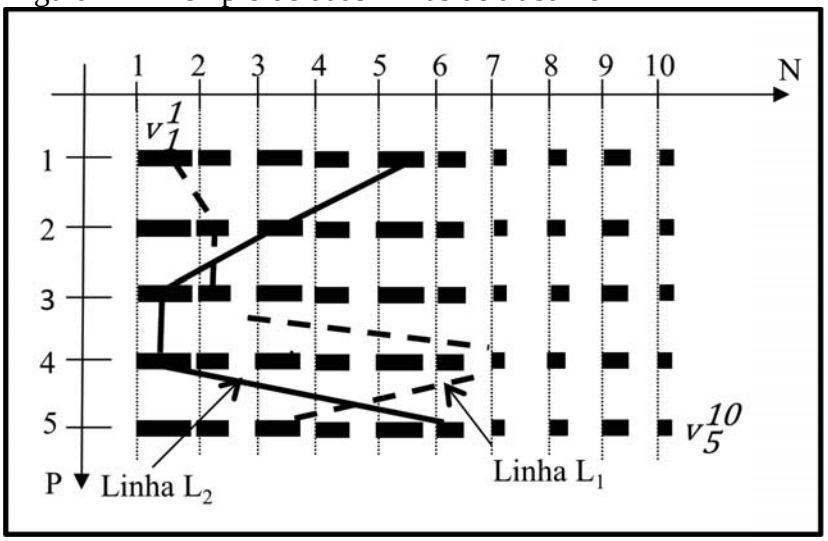
$$v_i^k = f(ti, tf, co, cd, \delta(v)), \forall i \in \{1, \dots, P\}, \forall k \in \{1, \dots, N\} \quad (2)$$

O conjunto V de todas as viagens da empresa no período de programação escolhido é definido por:

$$V = \cup_{i=1}^{P} V_i \quad (3)$$

Uma linha de trabalho é um conjunto possível de viagens diárias que um motorista pode fazer obedecendo a um conjunto de restrições operacionais e trabalhistas. Neste estudo designa-se por R este conjunto. R possui 7 restrições as quais são descritas adiante. A Figura 2 ilustra duas possíveis linhas de trabalho, L_1 e L_2, para uma empresa com $N = 10$ viagens por dia, para um período de planejamento $P = 5$ dias.

Figura 2 – Exemplo de duas linhas de trabalho



Na Figura 2 o eixo vertical representa os dias de programação. O eixo horizontal se refere às viagens diárias de cada dia. Os retângulos em preto representam cada viagem de cada dia da empresa. O comprimento destes retângulos representa a duração de cada viagem. A primeira viagem do conjunto *V* de viagens da empresa é a viagem v_1^1 enquanto a última viagem é a v_5^{10} ambas mostradas nesta figura. Uma linha de trabalho é um conjunto de várias viagens de dias consecutivos. Nesta figura, duas linhas de trabalho são exemplificadas, a $L_1 = \{v_1^1, v_2^2, v_3^2, v_4^7, v_5^3\}$ e a $L_2 = \{v_1^5, v_2^3, v_3^1, v_4^1, v_5^6\}$. A conexão entre duas viagens consecutivas de uma linha de trabalho constitui um par de viagens ordenado cronologicamente, $\left(v_i^j, v_{i+1}^k,\right), \forall i \in \{1, \cdots, P-1\}, \forall j, k \in \{1, \cdots, N\}$. Assim, uma linha de trabalho pode também ser representada como um conjunto, T, desses pares ordenados. Por exemplo, a linha de trabalho L_1 na Figura 2 pode ser representada pelo seguinte conjunto de pares $\{(v_1^1, v_2^2), (v_2^2, v_3^2), (v_3^2, v_4^7), (v_4^7, v_5^3)\}$. A equação (4) mostra este conjunto T de uma forma geral.

$$T = \left\{\left(v_1^{j_1}, v_2^{j_2}\right), \left(v_2^{j_2}, v_3^{j_3}\right), \ldots, \left(v_{P-1}^{j_{(n-1)}}, v_P^{j_n}\right)\right\} j_1, j_2, \ldots, j_n \in \{1, \cdots, N\} \quad (4)$$

Os locais de chegada de uma viagem e de partida da próxima viagem na mesma linha de trabalho devem ser iguais.

Seja X o conjunto de todas as linhas de trabalho possíveis no espaço de solução do problema. A cardinalidade deste conjunto é dada por:

$$card(X) = N^P \quad (5)$$

O conjunto das linhas de trabalho viáveis é $F \subseteq X$ e o conjunto das linhas de trabalho que serão selecionadas para satisfazer o problema é $S \subset F$.

Neste trabalho foram consideradas duas maneiras para mensurar a qualidade de uma linha de trabalho. A primeira é através da diferença entre a soma da duração de cada viagem da linha em relação a um valor de duração pré-definido. A segunda através da dispersão em relação à duração de cada viagem dentro da mesma linha.

A diferença de duração da linha $L_j \in X \mid j \in \{1, 2, \ldots, card(X)\}$ em relação a uma duração pré-definida é denominada θ_j e é calculada pela equação (6).

$$\theta_j = \left|VI - \Delta_j\right|, \quad (6)$$

onde,

VI é o valor pré-definido de duração ideal de uma linha;

$\Delta_j = \sum_{i=1}^{P} \delta\left(v_i^k\right) para\ k \in \{1, \ldots, N\}\ e\ v_i^k \in L_j$ (7)

designa “a soma das durações de cada viagem da linha *j*”;

Quanto menor θ_j melhor qualidade tem a linha de trabalho j.

A dispersão interna da linha $L_j \in X$ é o desvio padrão, σ_j, dos valores de duração de cada viagem v_i^k de L_j, dado por $\delta(v_i^k)$.

A forma de se obter estes parâmetros, diferença de duração de uma linha e dispersão da mesma, será discutida em detalhes no Capítulo 3.

Observe que, por construção, a primeira viagem de cada linha de trabalho deve se iniciar no primeiro dia do *rostering*. Além disso, em cada linha de trabalho existe apenas uma

viagem de cada um dos subconjuntos $V_{i \in D}$ e não pode haver um dia sem nenhuma viagem, qualquer que seja a linha de trabalho.

Cada linha de trabalho deve ser atribuída posteriormente a um motorista e constituirá sua escala de trabalho. Deve-se observar pela definição de linha de trabalho que não se permite que um motorista faça mais do que uma viagem no mesmo dia. No entanto, viagens de curtíssima duração, como as viagens especiais denominadas “circulares”, ou seja, diversas idas e voltas entre duas cidades vizinhas, serão representadas como uma única viagem de duração igual a soma da duração de todas idas e voltas realizadas no dia e poderão ser alocadas a um motorista. Elas serão mais bem explicadas na seção 3.1.3 que trata de viagens especiais.

Outra observação importante refere-se aos dias de folga de um motorista. Por construção, uma linha de trabalho não pode ter dias sem expediente. Neste trabalho, considera-se que cada motorista é alocado previamente a um padrão de folgas e este padrão de folgas será associado a uma das linhas de trabalho $L_j \in S$. A alocação de motoristas à padrões de folga da empresa pode ser representado como um problema de particionamento e é resolvido como tal à parte. As folgas atribuídas a cada motorista pela escala serão cobertas por um número de motoristas extras denominados “motoristas cobre-turnos”. Os detalhes sobre este problema serão cobertos nos capítulos posteriores.

Quando uma viagem $v_i^k \in V_i$ estiver incluída numa linha L_j diz-se que L_j “cobre” v_i^k. Cobrir no contexto de escalonamento de motoristas de ônibus quer dizer que a viagem v_i^k será atendida por um motorista de alguma das linhas L_j. O conjunto de todas as linhas de trabalho L_j viáveis que satisfazem R constitui uma matriz de cobertura designada por M_c.

As restrições impostas ao problema de escalonamento definidas pelo conjunto R são as seguintes:

r_1: deverá haver um período mínimo de descanso de 11 horas entre o horário de chegada de uma viagem num dia e o horário de partida da viagem consecutiva no próximo dia;

r_2: a soma das durações das viagens em qualquer linha de trabalho, ou seja, a carga de trabalho que um motorista receberá em sua escala de serviço, não poderá ser inferior a um mínimo e não pode ultrapassar a um valor máximo;

r_3: a dispersão em relação à duração das viagens dentro das linhas na solução final deve estar num nível satisfatório pré-fixado;

r_4: todas as viagens da empresa deverão estar cobertas por pelo menos uma das linhas de trabalho de S;

r_5: uma viagem diária não deverá ser incluída em mais de uma linha de trabalho de S;

r_6: o número de dias de trabalho consecutivos entre duas folgas consecutivas não deverá exceder a um limite máximo;

r_7: não poderá haver mais folgas do que um valor máximo dentro do período de programação.

A restrição r_1 garante um período de descanso remunerado antes do motorista iniciar uma nova viagem. É uma exigência trabalhista. Nos dados utilizados neste estudo, a soma de 11 horas a $tf(v_i^k)$ resulta num horário tal que sempre há pelo menos uma próxima viagem disponível para o mesmo motorista no dia $i + 1$.

A restrição r_2 visa não permitir que os motoristas trabalhem além de um máximo ou aquém de um mínimo de horas estipulado pela empresa. É uma restrição operacional/trabalhista. Tal medida é considerada difícil de ser conseguida na prática usando métodos tradicionais de programação manual das escalas.

A r_3 é uma restrição operacional/trabalhista. Força a não haver numa mesma linha de trabalho viagens que tenham muita variação de duração entre si. É uma restrição difícil de ser implementada na prática por métodos manuais.

A restrição r_4 se refere à condição de cobertura das viagens pelas escalas. Ela garante que nenhuma viagem diária da empresa ficará descoberta no final, isto é, sem ter um motorista a ela atribuído.

A restrição r_5 implica em não permitir que dois motoristas se apresentem para a mesma viagem num dia qualquer do período de programação.

A restrição r_6 limita o número máximo de dias de trabalho a que pode ser submetido um motorista entre duas folgas consecutivas e é uma restrição tanto operacional quanto trabalhista.

A restrição r_7 se destina a evitar que um motorista tenha em sua escala, um número de folgas maior do que um valor máximo permitido. É restrição operacional e trabalhista.

Alguns trabalhos na literatura consideram restrições parecidas com estas, mas em geral a escolha das restrições é direcionada a um tipo específico de problema, como mostra Ernst et al. (2004b) e, um exemplo especifico pode ser visto em Lourenço, Paixão e Portugal (2001).

O objetivo do problema é encontrar o conjunto *S* de linhas de trabalho viáveis, isto é, que cubram todas as viagens do conjunto *V* de viagens da empresa no máximo uma vez cada

sem deixar nenhuma a descoberto e que, ao mesmo tempo promova a redução nos custos, minimizando o número de motoristas, e melhore o nível de satisfação dos empregados, minimizando a dispersão σ_j e a diferença de duração θ_j de cada linha de trabalho do conjunto de linhas da solução final *S*.

O problema pode ser caracterizado como um problema de Otimização Combinatória, (WOLSEY, 1998), da seguinte forma:

$$\min\left\{\sum_{j\in S} c_j : S \subset F\right\} \tag{8}$$

Onde,

$F \subseteq X$ é o conjunto de linhas de trabalho viáveis, ou seja, linhas de trabalho que satisfazem ao conjunto de restrições *R*;

c_j é o custo de cada linha de trabalho $L_j \in X$, definido como $f(\theta_j, \sigma_j)$.

A obtenção destes custos será descrita no Capítulo 3.

Para formulação deste problema vários modelos foram considerados, entre eles o de emparelhamento, de máximo *matching, de* particionamento de conjuntos, de atribuição, fluxo em rede, etc.

No entanto, pela natureza multiobjetiva do problema tratado e para proporcionar maior flexibilidade na decisão das melhores linhas de trabalho a serem atribuídas a motoristas, a representação do problema como problema de cobertura de conjuntos não unicusto foi a escolhida para sua representação.

1.2 RELEVÂNCIA

Trata-se de uma pesquisa que resolverá um problema considerado de difícil solução devido à sua natureza combinatória associada ao grande porte das instâncias e que ao mesmo tempo possui restrições operacionais e trabalhistas que dificultam sua formulação, além de ser um problema com mais de um objetivo. Por exemplo, Cappanera e Gallo (2004) descreveram as dificuldades de geração de uma estrutura de dados para a formulação do problema de *rostering* em companhias aéreas de médio porte. Estas ocorrem tanto pelo tamanho da instância do problema quanto pela dificuldade de se implementar as restrições operacionais/trabalhistas. No caso específico de companhias aéreas ainda surge um aumento

extra de dificuldade devido a algumas restrições do problema não poderem ser consideradas implicitamente na construção de uma estrutura robusta como a de um grafo. É dada a seguir uma ideia do tamanho das instâncias no nosso caso.

Numa empresa com N = 150 viagens por dia e um período de planejamento P = 30 dias, o número possível de linhas de trabalho é dado por 150^{30}. Porém, algumas restrições podem ajudar a diminuir este número. É o caso da restrição que obriga as linhas de trabalho a ter uma e apenas uma viagem v_i^k de cada dia i. Isto implica que não pode haver mais de uma linha de trabalho começando na mesma viagem v_i^k do primeiro dia. Uma maneira de se calcular um limite superior para o número total de linhas de trabalho que podem existir num problema com *N* viagens diárias e *P* dias de programação da escala é mostrado na equação (9). Nesta equação o efeito redutor é levado em consideração na parcela que representa a subtração. Esta é uma maneira aproximada de se calcular um limite superior de linhas de trabalho nesta pesquisa. Em outras palavras, o número de linhas total é igual a combinação de uma linha com as demais menos as combinações entre as linhas que começam na mesma viagem do primeiro dia.

$$card(S) = \binom{card(X)}{N} - N \cdot \binom{card(X)/N}{N} \quad (9)$$

Para ilustrar, considerando um *rostering* de 5 dias com 4 viagens por dia, ou seja, $P = 5\ e\ N = 4$. Pela equação (5) tem-se $card(X) = 4^5 = 1024$ linhas de trabalho. Já as combinações possíveis destas linhas entre si pela equação (9) é igual a $card(S) = \binom{1024}{4} - 4 \cdot \binom{1024/4}{4} = 44.845.858.816$ combinações.

Para uma empresa de porte médio, como a que foi estudada, em que $N = 136$ viagens diárias e o comprimento mínimo da escala é de $P = 30$ dias, o número de combinações possíveis de escalas é muito grande.

Estas dificuldades, por outro lado, geram grande interesse em tentar resolver o problema. Este interesse pode ser confirmado no levantamento de Ernst et al. (2004b) sobre o assunto. Ele pode ser considerado um dos mais completos da área de escalonamento de pessoal quando se leva em consideração: a) o período de tempo pesquisado na literatura, b) as áreas de aplicação abrangidas, c) as formulações propostas e d) as técnicas empregadas na solução do problema. Foram cobertos 50 anos de estudo, entre 1954 a 2004. Foram selecionados e estudados 1.114 artigos, posteriormente classificados de acordo com sua área

de aplicação em 17 categorias, conforme ilustra a Tabela 1[1], dentro de algumas das quais se encontra embutido o problema estudado nesta pesquisa.

Tabela 1 – Categorias de problemas existentes entre 1954 e 2004

Categorias	Número de artigos selecionados
Crew Scheduling	219
Tour Scheduling	185
Flexible Demand	107
Workforce Planning	99
Crew Rostering	76
Shift Scheduling	64
Cyclic Roster	62
Days Off Scheduling	61
Shift Demand	55
Task Based Demand	47
Demand Modeling	40
Task Assignment	32
Shift Assignment	24
Disruption management	16
Other Classification	12
Stint Based Roster	9
Roster Assignment	6
Total	1114

Este levantamento justifica a motivação para o estudo de uma técnica eficiente de solução para problemas de escalonamento em geral.

As características de nosso problema, que engloba tanto a construção de linhas de trabalho quanto o da atribuição destas linhas a motoristas, se enquadra nas categorias de *Crew Scheduling* e *Crew Rostering* bem como de *Rostering Assignment* e *Days Off Scheduling*.

Procurou-se definir uma metodologia adequada para o desenvolvimento da pesquisa. Empregou-se o método quantitativo de modelagem, e foi utilizada teoria de conjuntos e uma heurística baseada em algoritmo genético para resolução do problema proposto.

Este problema é conhecido pelo seu grande porte e dificuldade de resolução, o que dificulta lidar com sua formulação e solução imediata a partir do modelo. A pergunta

[1] Os termos foram deixados em inglês para que o leitor possa realizar uma busca na literatura de forma mais precisa. Além do que, não são todos que possuem correspondentes em português.

principal que a pesquisa pretendeu responder foi “como lidar com problemas de grande porte, formulá-los e resolvê-los eficaz e eficientemente?”.

Segundo a taxonomia de Morabito e Pureza (2010) e também de Bertrand e Fransoo (2002) é uma pesquisa de natureza aplicada, axiomática quantitativa.

1.3 SOLUÇÃO PROPOSTA

Neste trabalho, propõe-se solucionar o problema usando uma heurística com duas fases, a construtiva e a de busca local. O objetivo da fase construtiva é gerar o conjunto F de linhas de trabalho viáveis. Para tal, representa-se o problema por meio de um grafo $G = (V, E)$, em que o conjunto de nós V corresponde às viagens do *rostering* segundo a equação (3) e o conjunto de arcos E corresponde às transições possíveis de cada viagem a partir do primeiro dia, definido pela equação (4). Os elementos do conjunto F são gerados a partir de percorrimentos pelo grafo G, tanto no sentido direto (do primeiro para o último dia), como no sentido reverso (do último para o primeiro dia). Durante o percorrimento do grafo são obtidos os parâmetros de diferença de duração de uma linha, θ_j, e sua dispersão interna, σ_j, para o cálculo dos custos da função-objetivo do problema. Ao final, os elementos obtidos do conjunto F são ordenados em ordem crescente de custo. A utilização de grafos tem sido empregada com sucesso na formulação e solução de problemas de mesma natureza, (CAPRARA et al., 1997), (CAPPANERA; GALLO, 2004), (SHODI; NORRIS, 2004).

A fase de busca local (ou fase de otimização) utiliza uma heurística baseada em algoritmo genético, (BEASLEY; CHU, 1996), para a solução de um problema de cobertura de conjuntos, (*SCP - Set Covering Problem*), que é construído a partir do conjunto F obtido na fase construtiva.

É impossível visitar o espaço de busca completo do problema estudado em um período de tempo razoável, devido ao grande número de combinações. Por isto foi usada uma estratégia iterativa de selecionar regiões desse espaço (subespaços de busca) e aplicar às mesmas um procedimento de busca local. Os subespaços de busca são gerados de maneira randômica e procurando atingir de maneira uniforme a toda extensão do espaço de busca original do problema. Ao subespaço selecionado é aplicado um procedimento de otimização na tentativa de encontrar um ótimo local. Nas iterações seguintes outros subespaços com novos ótimos locais serão selecionados. Este procedimento se repete até que um critério de

parada seja atingido. Os detalhes do algoritmo de busca e seleção e de otimização estão descritos no Capítulo 3.

Esta abordagem de solução foi adotada como alternativa a uma técnica de Programação Matemática, tendo em vista as dificuldades que a formulação do problema traz em relação ao tamanho do mesmo e a dificuldade de acomodar todas as suas restrições. É uma alternativa também a um método exato (por exemplo, do tipo *branch-and-bound*), uma vez que a natureza combinatória do problema exigiria deste tipo de método, em situações reais, recursos computacionais (de tempo de execução e memória) muito grandes. Com a abordagem adotada é possível obter boas soluções para problemas reais em um período de tempo razoável.

1.4 CONTRIBUIÇÕES DO TRABALHO

Trata-se de uma nova investigação sobre processos algorítmicos não-determinísticos para tratamento de problema real de escalonamento de pessoal. Por sua natureza é reconhecidamente um problema combinatório de grandes proporções, cujas restrições e múltiplos objetivos o tornam de difícil formulação e solução por meio de técnicas exatas.

Este estudo contribui em duas partes com a área de pesquisa. A primeira parte oferecendo uma abordagem diferente através de algoritmo e teoria de grafo para representar o problema de criação de linhas de trabalho. A segunda contribuição vem na forma de integração da primeira parte com heurística evolutiva baseada em algoritmo genético.

Além disso, é mostrado como resolver dois subproblemas que constantemente aparecem na prática, o da criação de linhas de trabalho e o de atribuição de padrões de folga a motoristas. Mostra também como aperfeiçoar algoritmos genéticos genéricos para uso em problemas específicos como o de escalonamento.

1.5 ORGANIZAÇÃO DO TRABALHO

Além do Capítulo 1, este trabalho está organizado da forma descrita a seguir.

O Capítulo 2 contempla a revisão da literatura, na qual são apresentados trabalhos similares a este para solução de problemas em outras áreas como as de aviação, metrô, ferroviária, alfândega, de enfermagem, de *call centre*, além de estudos recentes propondo soluções para o problema de cobertura de conjuntos.

O Capítulo 3 descreve a solução proposta em detalhes mostrando como funciona e se interligam cada uma das fases nas quais o problema foi resolvido.

O Capítulo 4 apresenta os resultados obtidos com a heurística proposta a partir dos dados reais da empresa estudada. Também contém os resultados dos testes realizados para resolver o problema de padrões de folga.

O Capítulo 5 encerra este documento abordando uma discussão sobre as técnicas empregadas neste trabalho e as sugestões para futuras pesquisas. Nele também é discutida a questão da formulação e resolução do problema de alocação de motoristas aos turnos ou padrões de folga da empresa.

2 REVISÃO BIBLIOGRÁFICA

2.1 FASES DA PESQUISA BIBLIOGRÁFICA

A pesquisa bibliográfica foi dividida em duas fases. As fases horizontal e vertical (FLEURY, 2010). Os procedimentos desenvolvidos em cada uma serão discriminados em seguida.

Era esperado como resultado da fase horizontal, a elaboração precisa do tema. Porém não estavam descartadas pequenas mudanças e até mesmo, se fosse justificável, uma mudança no mesmo. No nosso caso, foram necessários apenas ajustes para especificar, com maior rigor, o problema estudado. Assim sendo, foi escolhido como problema principal o subproblema da construção de linhas de trabalho para empresas de ônibus entre os diversos subproblemas que constituem um problema de escalonamento de pessoal (ERNST et al., 2004a).

Em seguida, iniciou-se a pesquisa vertical. O objetivo foi o de conhecer e avaliar as soluções existentes para o problema escolhido e a partir desta análise propor uma nova formulação e técnica de solução para o mesmo.

Como resultado foi proposto novo algoritmo baseado na busca no espaço de solução do problema o qual, em conjunto com uma heurística evolutiva, permite lidar com a questão da magnitude do problema. Para a comprovação da eficiência do método proposto, foram realizados testes com dados reais de uma empresa de transporte interurbano.

Procurou-se selecionar para este documento artigos que mostrassem a abrangência da pesquisa realizada bem como a importância do conteúdo para o estudo proposto. A ordem de apresentação dos mesmos, no entanto, não segue nenhum critério.

2.2 PESQUISA HORIZONTAL

Recebe este nome a fase da pesquisa em que se tenta abranger o máximo possível de informações acerca do objeto de estudo (daí o nome horizontal), (FLEURY, 2010). O aprofundamento nos detalhes da formulação do problema e dos métodos usados para a sua solução serão tema da próxima fase da pesquisa. Um resumo dos principais artigos analisados é apresentado a seguir.

2.2.1 Levantamento sobre escalonamento de pessoal

Ernst et al. (2004b) procederam um levantamento abrangente sobre o tema de escalonamento ou programação de pessoal em geral. Em torno de mil artigos entre 1954 e 2004 foram revisados. Estes artigos foram classificados segundo o tipo de problema, sua área de aplicação e o método de solução proposto. A Tabela 1 no Capítulo 1 mostra a classificação dos tipos de problemas. As áreas de aplicação encontradas para os mesmos são mostradas a seguir:

Áreas de aplicação:

1. Companhias aéreas
2. Ônibus
3. Ferrovias
4. Metrôs
5. *Call Centres*
6. Enfermeiras
7. Área de saúde sem levar em consideração enfermeiras
8. Serviços emergenciais e de segurança
9. Bibliotecas, Correios, Universidades, Centrais Energéticas
10. Gerenciamento de entrepostos de mercadorias
11. Serviços financeiros
12. Serviços de hospedagem e turismo
13. Vendas
14. Manufatura
15. Engenharia de manutenção

Os principais métodos de solução encontrados para resolvê-los foram os seguintes:

Métodos de solução

1. Enumeração
2. Inteligência Artificial
3. Sistemas Especialistas
4. Programação por Lógica de Restrições (*Constraint logic programming*)
5. Heurística construtiva (*Constructive heuristic*)
6. Buscas locais simples: *Hill-climbing* e método do gradiente descendente
7. Recozimento Simulado (*Simulated Annealing*)
8. Busca Tabu (*Tabu Search*)
9. Procedimento de Busca Adaptável Aleatória e Gulosa (GRASP – *Greedy Random Adaptive Search Procedure*), Colônia de formigas, Busca no Espaço do Problema (*Problem Space Search*).
10. Procedimento evolutivo baseado em Algorítmo Genético (*Genetic Algorithm*)
11. Simulação
12. Teoria de fila
13. Programação Dinâmica
14. Programação por Metas (*Goal programming*)
15. Programação Matemática
16. Programação Linear
17. Programação Inteira
18. Relaxação lagrangeana
19. Geração de Colunas
20. *Branch and Bound*
21. *Branch and cut*
22. *Branch and price*
23. Cobertura de conjuntos
24. Particionamento de conjuntos
25. Fluxo em Redes
26. *Matching*
27. Métodos específicos

Foi observado durante a consulta a literatura que uma das linhas de pesquisa mais promissora dentro dos métodos de solução é a que tem usado soluções híbridas, isto é, o uso de métodos exatos (como os de programação linear, etc) juntamente com meta-heurísticas como Busca Tabu, Recozimento Simulado e Algoritmo Genético.

Foi decidido usar nesta pesquisa meta-heurística evolutiva junto com procedimentos algorítmicos de percorrimento do espaço de solução do problema. Trata-se de uma combinação que oferece novas perspectivas de estudo tanto para a modelagem quanto para a solução do problema.

2.2.2 Formulação e solução de problemas de escalonamento

Lezaun et al. (2010) descreveram um processo misto de escalonamento de pessoal. A área de aplicação do problema foi a área ferroviária mais especificamente de empresa de transporte ferroviário espanhola. O problema resolvido foi o de criar a escala de programação de pessoal para um ano de duração tendo em vista os horários de trabalho do pessoal das diversas estações presentes na ferrovia. O problema foi resolvido em duas fases. Na primeira fase da resolução foram usadas planilhas e a linguagem Visual Basic. Esta primeira fase consistiu em projetar um gráfico dividido em períodos de quatro semanas os quais foram atribuídos a cada estação, levando em consideração a repetição dos padrões para um ano bem como os períodos de férias dos funcionários. Na segunda fase da solução foram atribuídos turnos substitutivos para cobrir aqueles turnos deixados em aberto devido ao período de férias dos funcionários. Esta atribuição é feita objetivando minimizar a distância percorrida pelo pessoal de outras estações que teriam que ser trazidos para cobrir o turno não coberto. A programação é feita com base em janelas de quatro semanas. Os autores reportaram como resultado positivo desta pesquisa uma grande melhoria nas escalas produzidas por este método em comparação ao antigo sistema utilizado pela ferrovia.

Elizondo, Parada e Pradenas (2010) estudaram a questão de escalonamento de pessoal em problemas relacionados a metrô. Segundo os autores, o gerenciamento operacional de metrô está associado a problemas de otimização combinatória como programação de horários de trens e de tripulação bem como programação de pessoal os quais pertencem à classe de problemas NP-Hard. Por esta razão, em geral estes problemas são equacionados e resolvidos utilizando heurísticas em situações reais. A pesquisa foi realizada para resolver o problema da geração de conjuntos de viagens ótimas para serem atribuídas a motoristas em um dia de

trabalho. Como restrições do problema estudado, foram consideradas condições estabelecidas em acordos trabalhistas bem como condições estabelecidas pela empresa (condições operacionais). Como objetivos a serem atingidos na solução do problema foram considerados minimizar o número de motoristas e minimizar o intervalo entre viagens consecutivas. A proposta para solução do problema foi a de usar um enfoque construtivo híbrido. Segundo os autores, este método de solução tem a vantagem de se poder visualizar a construção da solução de maneira similar ao processo manual que, segundo os autores, ainda é bastante empregado. Além disso, o método proposto, segundo os autores, tem a vantagem de controlar a explosão combinatória encontrada em problemas desta natureza. Como resultado de seu estudo, as soluções obtidas ao usar a heurística proposta foram comparadas com resultados obtidos do sistema atualmente em funcionamento para o metrô da cidade de Santiago/Chile. Pelo exposto durante a descrição da pesquisa, ambos os resultados foram similares. Já na comparação da heurística proposta com a meta-heurística de busca tabu e gulosa, em dez problemas resolvidos, o método proposto apresentou soluções com um número mínimo de carga (viagem) em seis casos. Mas quanto a minimização do intervalo entre viagens a meta-heurística de busca tabu apresentou melhor resultado em todos os casos tanto em relação a heurística proposta quanto a heurística gulosa.

O trabalho de Oliveira (2001) é relacionado a otimização numérica para variáveis reais, o que o diferencia de nosso estudo, que está inserido na área de otimização combinatória discreta com restrições. Mas mesmo assim, apresenta uma concisa definição de operadores evolutivos e critérios de parada que contribuíram para a elaboração de alguns procedimentos usados em nossa heurística.

O trabalho estabelece o que vem a ser espaço de busca e vetor de solução de problemas de otimização em geral. Mostra a diferença entre a classe de problemas de otimização e de otimização combinatória, definindo matematicamente problemas de minimização e de maximização. Em seguida, mostra a diferença entre mínimos e máximos locais e globais e define funções objetivo uni e multimodal ao mesmo tempo em que mostra a influência de haver ou não restrições nas variáveis do problema. Um ponto de destaque neste trabalho é a clara distinção que faz entre os métodos de solução dos problemas de otimização através de algoritmos determinísticos e de algoritmos estocásticos. São estocásticos os algoritmos de Busca Tabu, Evolutivo e de Recozimento Simulado.

No nosso caso, durante a construção do conjunto de linhas de trabalho F (ver Capítulo 3) o algoritmo usado é um algoritmo estocástico pelo fato de, em qualquer iteração, não ser possível prever a próxima configuração a ser atingida. Entendendo-se por configuração o subconjunto de vetores do espaço de solução que é viável e pode otimizar os custos finais.

Uma ilustração interessante que exemplifica a diferença entre os dois tipos de algoritmos é mostrada na Figura 3. Embora esta figura ilustre um problema que usa variáveis contínuas, serve também para exemplificar o caso de problemas com variáveis discretas, como é o nosso caso.

Observe que no algoritmo da esquerda (determinístico) a partir de um ponto inicial x_0, o próximo ponto encontrado é único. No algoritmo da direita (estocástico) não há mais um único ponto, mas sim uma região de pontos que podem ser atingidos.

Figura 3 - Diferença entre algoritmos determinísticos e estocásticos.

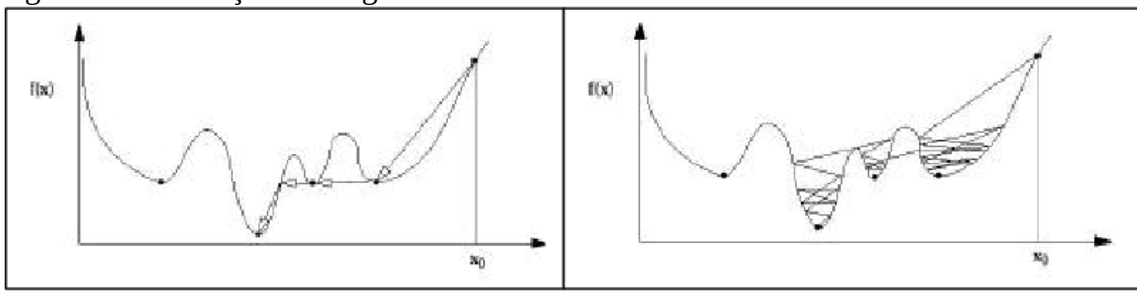


Figura 1.4 - Algoritmos de busca global: (a) determinístico; (b) estocástico

Fonte: (OLIVEIRA, 2001)

A revisão sobre os operadores dos algoritmos evolutivos fornecida no estudo auxiliou a compreender a mecânica dos operadores dos algoritmos genéticos. São descritos os operadores de cruzamento e mutação, bem como é fornecida uma discussão sobre a qualidade de cada operador avaliado. Os operadores de cruzamento analisados foram o N-Pontos, Uniforme, Aritmético usando média aritmética e usando média geométrica, Blend (BLX), Unimodal Normal Distribution Crossover (UNDX), Simulated Binary Crossover (SBX), Simplex (SPX) e Heurístico ou Direcional. Destes, o mais interessante para o nosso caso é o Heurístico ou Direcional que explora propriedades do problema em estudo para gerar filhos melhores, (OLIVEIRA, 2001).

Quanto aos operadores de mutação, são descritos o Binário, o Aleatório ou Uniforme e o Não Uniforme. Em nosso caso, usamos o operador de mutação sugerido por Beasley e Chu (1996) que se baseia na ideia de uma taxa variável de mutação de bits. A forma de calcular

esta taxa de mutação se baseia no tamanho das instâncias e na forma aleatória de geração do conjunto F. Será explicada em detalhes no Capítulo 3.

Chen, Liu e Chou (2013) propõe uma solução evolutiva alternativa ao método de solução convencional de duas fases para resolver o problema integrado de programação de horários de tripulação em companhias aéreas. O enfoque proposto modela e formula o problema como um problema de otimização combinatória com múltiplas restrições e objetivos. Uma estrutura evolutiva integrada é proposta para resolver simultaneamente os subproblemas de ida-e-volta da tripulação (crew pairing) e de construção de linhas de trabalho para a tripulação (crew rostering). Para melhorar a eficiência do explorador de conjunto de Pareto, os autores aplicam uma nova variante nos algoritmos genéticos de sorteio não dominante do tipo II (NSGA-II), que segundo os autores é um dos algoritmo evolutivos para otimização multi objetivos mais populares. A variante proposta lida com o manuseio das restrições totalmente dependentes do problema e uma política de fronteira para garantir indivíduos diversificados. A proposta é validada e verificada em um caso de teste real de uma empresa aérea. Os resultados da proposta evolutiva integrada são comparados com os resultados convencionais obtidos por métodos sequenciais. Também são feitas comparações entre o método variante e o convencional NSGA-II. Estes testes confirmaram que o método com a solução variante obtém resultados superiores sob dois aspectos: Primeiro, sob o aspecto da convergência em relação ao método original (sem a variante). Segundo, que o conjunto de horários a serem atribuídos à tripulação obtidos com o método variante permitiu aos tomadores de decisão da empresa escolher conjuntos de horários de melhor qualidade se comparados com os mesmos conjuntos obtidos da programação real da empresa tendo os mesmos objetivos e restrições.

Panton e Ryan (1999) estudaram o problema da alocação de mão de obra em tarefas com múltiplos *breaks*. Este subproblema pode ocorrer em diversos tipos de problemas de escalonamento de pessoal. Emprega como estrutura de dados listas de adjacências, matrizes e grafos. Formula o problema como um problema de cobertura de conjuntos não unicusto. Propõe a técnica de geração de colunas para a solução do problema e emprega diversos procedimentos para resolvê-lo, como o algoritmo de caminho mínimo, algoritmos de programação linear e de programação inteira mista, algoritmo de solução do problema dual e heurística gulosa (*greedy heuristic*).

Esclapès (2000) formulou o subproblema da alocação de motoristas de ônibus em empresas de transporte coletivo como um problema de cobertura de conjuntos generalizado. Usando a notação de Wolsey (1998) para a definição deste problema, pode-se entender sua

formulação para o problema de cobertura de conjuntos da seguinte maneira: ao invés de encontrar os locais onde deverão ser instalados os centros que darão cobertura a todas as regiões a um custo mínimo, os centros serão substituídos por escalas de serviço (linhas de trabalho) possíveis e as regiões serão as viagens diárias da empresa dentro do horizonte de planejamento. Cada viagem poderá ser coberta por uma ou mais escalas. Cada escala deve ter uma quantidade de motoristas maior ou igual a zero. O que se deseja encontrar para o problema é o número mínimo de escalas a serem atribuídas a motoristas de forma a atender a demanda mínima de viagens em todos os períodos do horizonte de planejamento.

A autora enviou um questionário a vinte empresas de uma região espanhola obtendo respostas de apenas 50% das mesmas. O resultado mostra que na prática as empresas não estão satisfeitas com soluções que conduzam apenas a um "ótimo resultado financeiro". Além do financeiro, deseja-se também, distribuir a carga de trabalho entre os motoristas de forma a não gerar insatisfação entre os mesmos.

Outro objetivo que também diverge do financeiro é o de fazer com que o total de carga de trabalho (horas de viagens) de cada motorista no fim do período de programação fique o mais próximo possível de um valor fixado *a priori*.

Esclapès (2000) resolveu o problema por partes usando programação linear e utilizando um processo heurístico construtivista através de um algoritmo guloso (*greedy*) cujo resultado é a determinação da melhor cobertura de viagens possível para posteriormente atribuí-la aos motoristas.

Lourenço, Paixão e Portugal (2001) trataram do problema de combinação de tarefas individuais (viagens) visando a montagem de linhas de trabalho as quais seriam posteriormente atribuídas a uma tripulação. A tripulação era constituída de motoristas e cobradores de ônibus. Procuram determinar quantas viagens seriam necessárias para cobrir toda a demanda da empresa por viagens e quais características cada viagem deveria ter como local de partida, horário de início, horário de término, duração e o itinerário até o local de finalização da viagem. O problema foi por eles denominado *Bus Driver Scheduling Problem* ou simplesmente BDSP. Na literatura, é comum se referir a problemas de determinação das melhores viagens para cobrir a demanda como problemas de "*crew scheduling*".

O problema é tratado como um problema de cobertura de conjuntos modificado com múltiplos objetivos. A função objetivo é composta de várias funções individuais.

É ressaltado no estudo que o recente sucesso do uso de meta-heurísticas em problemas de cobertura de conjuntos com objetivos múltiplos sugere que elas sejam mais investigadas

para a continuação dos trabalhos nesta área. Meta-heurísticas que foram empregadas na pesquisa: Algorítmo Genético, GRASP e Busca Tabu.

Aickelin (1999) investigou como o conhecimento específico de uma dada área de aplicação influencia no sentido de melhorar uma meta-heurística baseada em algoritmo genético para problemas de otimização com múltiplas escolhas. A área de aplicação à qual a meta-heurística foi aplicada foi a área de saúde, mais precisamente, a área de enfermagem em hospitais.

Embora tenha traços semelhantes com o problema tratado por Lourenço, Paixão e Portugal (2001), certas características são peculiares deste problema, como o horizonte de planejamento que não é mais tão extenso quanto o de motoristas de ônibus que é de um ou mais meses. Para o caso de enfermeiras, os horizontes de planejamento normalmente usados são da ordem de alguns dias até uma ou duas semanas. Os turnos ou escalas de serviços, assim como no caso de motoristas, são as combinações possíveis de dias de trabalho intercalados com dias de folga. Num mesmo horizonte de 7 dias, o funcionário de enfermagem pode trabalhar 5 turnos diários e folgar 2 dias ou, trabalhar 4 noites (ou turnos noturnos) e folgar 3 dias. O número de combinações possíveis é grande e os funcionários não podem optar por esta ou aquela combinação. Ao invés disso, é o sistema que encontrará a combinação de turnos correta para eles. No entanto, o sistema levará em consideração as preferências individuais de cada um. Eles podem indicar se desejam trabalhar só de dia, só de noite, ou se tanto faz trabalhar de dia ou de noite e em quais dias da semana. Por exemplo, para um funcionário pode ser que trabalhar de segunda-feira a quinta-feira não tenha problema, porém pode ser que ele não deseje trabalhar na sexta-feira. Já outro não tem problema trabalhar na sexta-feira, porém não gostaria de trabalhar na segunda-feira. Além destas preferências, os turnos, quer sejam diurnos ou noturnos, são classificados em tipos de acordo com o grau de habilidade requeridos em cada um. Os funcionários são escolhidos de acordo com suas habilidades para atender a estes tipos de turnos criados.

O problema de escala de enfermagem se resume em programar um número de funcionários para uma ala do hospital sem deixar nenhum turno descoberto atendendo todas as restrições citadas e, se possível, o desejo individual de cada um, dentro do horizonte de planejamento.

O problema é tratado como um problema de cobertura de conjuntos, porém designado de problema de múltipla escolha, pelo fato do funcionário poder escolher entre vários padrões de turnos disponíveis apenas um. Pode ser tratado também como um problema de atribuição, por atribuir funcionários a tarefas dentro de uma ala de hospital.

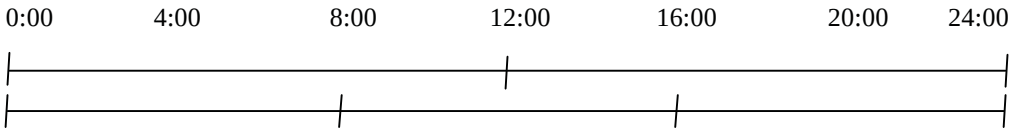
Mason e Nielsen (1999) verificaram a possibilidade de se formular e resolver diferentes problemas de programação de pessoal, utilizando módulos genéricos que podem ser combinados entre si para direcionar posteriormente o tratamento para um problema específico. O resultado do estudo foi testado com sucesso nas áreas de *call center*, serviço de turno em alfândega e escalonamento de enfermagem para três hospitais.

Para *call centres* as linhas de trabalho são segmentadas em turnos. Na construção das linhas de trabalho as seguintes variáveis de controle são consideradas: horário de início e de término, habilidades exigidas para cada turno e em qual escritório o trabalho será executado. O número de empregados que devem trabalhar em cada turno também são variáveis do problema. Quanto ao comprimento de cada turno, há um curto de 4h e outro longo de 6h45. Os funcionários poderão ser alocados em, no máximo, dois desses turnos por dia, desde que sejam localizados no mesmo prédio. Existem limitações no tempo de folga entre turnos consecutivos. Tipicamente se formam de 2300 a 3000 diferentes turnos. A programação é feita no horizonte de uma semana.

A segunda área de aplicação onde o modelo foi empregado no trabalho de Mason e Nilson (1999), foi a de escalonamento de pessoal de meio período na alfândega do aeroporto internacional de Auckland, Nova Zelândia. As variáveis do problema são os turnos que possuem comprimento de três a oito horas e os horários de início e de término dos mesmos que devem ser flexíveis o suficiente para fazer coincidir o máximo possível com os horários dos voos do aeroporto. Não há diferentes habilidades exigidas dos funcionários. Mas as preferências de horários (a cada meia hora) são considerados na formulação do problema. Há um procedimento de recompensa para os empregados que trabalharem em turnos menos desejados. O problema é formulado como um problema de otimização multiobjetivo onde um dos objetivos é o de minimizar a diferença de horários de início de turnos entre dias consecutivos para um mesmo funcionário. Outro objetivo é o de maximizar o atendimento as preferências dos empregados em relação à especificação de meia hora usando uma escala de um a cinco.

Na terceira área de aplicação, a área de enfermagem, as linhas de trabalho não são cíclicas. Os turnos são fixos e em número de cinco por dia a saber, o turno de dia com 12 horas, o da noite também com 12 horas, o da manhã com 8 horas, o de pós-almoço com 8 horas e o do entardecer com 8 horas. Estes turnos são quebrados em segmentos de 4 horas cada conforme mostra a Figura 4.

Figura 4 – Esquema da divisão de turnos do problema de enfermagem



Competências e habilidades específicas dos trabalhadores são modeladas neste problema. São levados em consideração o número mínimo e máximo de habilidades e competências exigidas em cada um dos segmentos de um turno.

O problema é formulado como um problema de partição generalizada de conjuntos e resolvido usando técnicas matemáticas de programação linear com métodos de solução do tipo *branch-and-bound*.

Kohl e Karich (2004) analisam o problema de programação de pessoal para companhias de aviação. Dentre os problemas revisados, este parece ser o mais complexo por apresentar um grande número de diferentes tipos de restrições e mais objetivos do que os problemas de programação de outras áreas. O problema é formulado como um problema de programação inteira binária a partir de uma representação do problema como um problema de particionamento de conjuntos.

O problema de escalonamento de pessoal é um dos mais estudados na área de aviação havendo muitos artigos a respeito na literatura conforme se pode concluir ao analisar a bibliografia anotada de Ernst et al. (2004).

2.2.3 **Resultados da pesquisa horizontal**

Definição do tema

O estudo realizado durante esta fase mostrou que os problemas de escalonamento de pessoal surgem em diversas áreas de atividade e empregam várias formulações, sendo resolvidos com diversos métodos e técnicas, empregados tanto de forma independente como a partir de combinações dos mesmos entre si. Em qualquer destas áreas, o problema de escalonamento é composto sempre de um conjunto de subproblemas menores, (ERNST et al., 2004a). Um destes subproblemas é o problema da construção de linhas de trabalho para a programação de pessoal. Este foi escolhido como tema do presente trabalho. Também foi definida que a área alvo será a área de transporte público intermunicipal. Observou-se, na literatura, poucos estudos neste setor no Brasil. A título de exemplo pode-se citar o trabalho de Silva e Cunha (2011) na área de transporte urbano.

Tratamento e formulação do problema

O problema de construção de linhas de trabalho para o escalonamento de pessoal tem sido tratado como um problema de cobertura de conjuntos não unicusto, um problema de particionamento de conjuntos e um problema de atribuição. Por isto uma revisão dos mesmos foi realizada e é mostrada a seguir.

O Problema de Cobertura de Conjuntos não unicusto

Utilizando a notação empregada por Nemhauser e Wolsey (1988) o problema de cobertura de conjuntos não unicusto pode ser definido conforme a Definição 1.

Definição 1 – Problema de cobertura de conjuntos não unicusto

$Sejam,$

1. $M = \{1, \dots, m\}$ um conjunto finito qualquer,
2. $N = \{1, \dots, n\}$ um conjunto finito qualquer,
3. $\{M_j\}$ uma coleção de subconjuntos de M para $j \in N$,
4. $F \subseteq N\ cobre\ M\ se\ e\ somente\ se\ \bigcup_{j \in F} M_j = M$
5. $C = (c_1, \dots c_n)$ um vetor n-dimensional de valores reais,
6. $c(F) = \sum_{j \in F} c_j$ o custo do conjunto F,

Então, o problema de cobertura de conjuntos não unicusto é o problema de otimização combinatória definido por:

7. $min(z = c(F))$

O Problema de Particionamento de Conjuntos

Pode ser definido conforme mostra a Definição 2.

Definição 2 – Problema de particionamento de conjuntos não unicusto

$Dadas\ as\ 6\ condições\ iniciais\ do\ problema\ de\ cobertura\ de\ conjuntos\ não\ unicusto\ mostradas\ na\ Definição\ 1\ e\ acrescentando - se\ mais\ a\ condição$

$M_j \cap M_k = \emptyset, \forall (j,k) \in F, j \neq k$

Então, o problema de particionamento de conjuntos não unicusto é o problema de otimização combinatória definido por:

$min(z = c(F))$

O Problema de Atribuição

O problema de atribuição pode ser definido pela Definição 3.

Definição 3 – Problema de atribuição

Sejam,

$P = \{1, \dots, n\}$ um conjunto finito de pessoas designadas pelo índice *j*;

$T = \{1, \dots, m\}$ um conjunto finito de tarefas designadas pelo índice *i*;

$m \leq n$ a condição do problema que diz que há um número suficiente de pessoas para realizar todas as tarefas;

$x_{ij} = \{0,1\}$ a variável independente que designa se uma tarefa *i* for atribuída a uma pessoa *j*, isto é, $x_{ij} = 1$ ou $x_{ij} = 0$ caso contrário;

c_{ij} uma variável real que designa o custo envolvido em atribuir a tarefa *i* a pessoa *j*;

$\sum_{j=1}^{n} x_{ij} = 1, i = 1, \dots, m$ a restrição que impõe a condição de que uma tarefa só poderá ser feita por uma única pessoa;

$\sum_{i=1}^{m} x_{ij} = 1, j = 1, \dots, n$ a restrição que impõe a condição de que uma pessoa só pode fazer uma única tarefa de cada vez;

Então o problema de atribuição é o problema de otimização combinatória definido por $min\left(z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}\right)$.

Os três problemas anteriores, podem ser formulados como problemas de programação inteira binária, (BIP - *Binary Integer Programming*) conforme mostrado pela Formulação 1.

Formulação 1 – Formulação dos problemas estudados como BIP.

Dadas as Definições 1 e 2 dos problemas de cobertura e particionamento, seja A uma matriz m x n cujas linhas e colunas estão associadas a M e N respectivamente e $a_{ij} = 1$ se a linha *i* é coberta pela coluna *j* e $a_{ij} = 0$ caso contrário, isto é, se a linha *i* não é coberta pela coluna *j*;

$x_{ij} = \{0,1\}$ a variável independente;

Então: $min\left(z = \sum_{j=1}^{n} c_j x_j : \sum_{j=1}^{n} a_{ij} x_j \geq 1, i = 1, \cdots, m\right)$,

é um problema de cobertura de conjuntos não unicusto e $min\left\{z = \sum_{j=1}^{n} c_j x_j : \sum_{j=1}^{n} a_{ij} x_j = 1, i = 1, \cdots, m\right\}$ é um problema de particionamento de conjuntos não unicusto.

E,

$x \in B^{mn}$ onde B^{mn} é um conjunto de vetores mn dimensionais e,

$\sum_{j=1}^{n} x_{ij} = 1 \quad para\ i = 1, \cdots, m;$

$\sum_{i=1}^{m} x_{ij} \leq 1 \quad para\ j = 1, \cdots, n;$

O problema de construção das linhas de trabalho no escalonamento de pessoal para esta pesquisa será tratado como um problema de cobertura de conjuntos não unicusto.

O método de solução

Observa-se na literatura o uso de métodos de solução exatos (determinísticos) e heurísticos (não determinísticos). Neste trabalho foi dada ênfase à pesquisa dos métodos heurísticos de solução devido ao tamanho das instâncias do problema. Entre os mesmos destacam-se os evolutivos baseados em algoritmo genético, os baseados em busca tabu (GLOVER, 1989) e (GLOVER, 1990) e os que utilizam a idéia do recozimento simulado, (ERNST et al., 2004b). Destes três, o mais empregado tem sido o baseado em algoritmo genético. Por esta razão, procuramos basear nosso trabalho também em algoritmo genético.

Um algoritmo genético é um procedimento evolutivo baseado na ciência da evolução natural dos seres vivos (HOLAND, 1975)**.** Parte-se de uma população inicial, que deve ser gerada de acordo com as características do problema real que se deseja resolver. A partir desta população inicial, como no processo evolutivo, os indivíduos mais aptos à sobrevivência

serão escolhidos para cruzarem entre si gerando filhos, os quais, se espera, formarão uma nova população com maior índice de adaptação ao ambiente do que a anterior. Normalmente a adaptabilidade é medida em termos do objetivo que se deseja atingir para o problema. Por isto a importância de se fazer uma boa representação do mesmo. Se o objetivo é reduzir custos, o grau de adaptação do indivíduo será medido pelo valor do custo de cada indivíduo na população. Quanto maior o seu custo, menos adaptado ele estará ao ambiente. E, quanto menor o seu custo, mais adaptado ele estará a este mesmo ambiente.

Um algoritmo genético clássico possui a estrutura de funcionamento mostrada no Algoritmo 1 e cujo princípio foi empregado em nosso trabalho.

Algoritmo 1 – Algoritmo genético clássico

1. Gerar uma população inicial
2. Repetir
    3. Selecionar pais da população inicial
    4. Cruzar os pais selecionados gerando novos filhos (nova geração)
    5. Aplicar mutação nos filhos gerados
    6. Avaliar a adaptação dos filhos gerados segundo um critério
    7. Substituir na população atual alguns ou todos os filhos pelos gerados
8. Até algum critério de parada baseado na adaptação da nova geração ou no número de iterações ser atingido

O Algoritmo 1 inicia com a geração de uma população inicial no passo (1). Um estudo prévio deve ser feito para definir a estrutura do indivíduo no sentido de o mesmo representar de forma adequada uma solução para o problema. Idem para o cálculo de adaptação dos indivíduos ao ambiente. Depois de gerada a população inicial, entra-se num laço de repetição que executa os passos de (3) a (7) várias vezes. No passo (3) são selecionados os pais que cruzarão entre si para gerar um novo filho. No passo (4) os pais selecionados efetuam a operação de cruzamento. No passo (5) é realizada a operação que transforma os genes dos indivíduos novos de acordo com uma lei de formação. No passo (6) é avaliado o grau de adaptação do filho gerado ao ambiente. Caso ele tenha boa adaptação substituirá um outro

filho de pior adaptação na população atual, passo (7). O processo se repete até que um critério de parada seja atingido, passo (8).

Uma vez tendo obtido o ferramental de como modelar e formular o problema em estudo, o próximo passo é analisar em detalhes como o problema tem sido resolvido e a partir deste estudo elaborar uma técnica de solução para resolvê-lo.

2.3 PESQUISA VERTICAL

Recebe este nome (FLEURY, 2010) a fase de pesquisa em que se procura aprofundar (daí o nome vertical) o entendimento sobre um assunto específico. Em particular, no nosso caso, foram compreendidos os detalhes das técnicas de resolução do problema em estudo. O objetivo é o de propor melhorias ou técnicas alternativas mais eficientes para resolvê-lo.

Salienta-se que o problema de cobertura de conjuntos usado na nossa representação do problema pertence à categoria de problemas de complexidade NP-Hard (KARP, 1972).

Os subproblemas de criação de linhas de trabalho, conforme já explicado anteriormente, são tratados dentro do contexto de problemas maiores que são os problemas de escalonamento de pessoal em geral. Uma descrição desses problemas, ressaltando em cada um o subproblema da criação das linhas de trabalho sob diferentes formatos como turnos, serviços etc, é fornecida em seguida, juntamente com uma descrição detalhada dos procedimentos que empregam teoria de cobertura de conjuntos para achar as melhores linhas.

2.3.1 Resolução do problema de escalonamento de pessoal

Mesquita et al. (2011) desenvolveram em sua pesquisa técnicas para a formulação e solução de problemas da área de escalonamento de pessoal que envolvem três subproblemas ao mesmo tempo: o subproblema de programação de veículos, o da programação de tripulação e o subproblema do escalonamento de pessoal. O estudo apresentado pelas autoras faz uma nova formulação ao considerar estes três problemas de forma integrada como um único problema de programação matemática inteira. O método de solução usado pelas autoras para resolver o problema emprega programação por metas e um algoritmo sequencial que procura obter soluções viáveis tendo em vista a natureza multi objetivo não linear e binária do problema. O método proposto dá mais prioridade para atingir as metas de integração dos três problemas do que atender as metas de atribuição de motoristas às escalas. Uma heurística foi

desenvolvida para deixar a cargo da empresa a escolha entre várias opções de programação de veículos e tripulação enquanto ao mesmo tempo procura respeitar o máximo possível os interesses gerenciais e as preferências dos motoristas. Uma aplicação da proposta a uma empresa de ônibus portuguesa mostra a influência da otimização veículo-tripulação-programação para o escalonamento de pessoal.

O problema da programação de horário para tripulação na indústria aérea é um problema intensamente investigado da literatura de pesquisa operacional pois pode reduzir dramaticamente os custos envolvidos para as companhias aéreas. Dada a programação dos horários de voos de uma companhia aérea, o problema consiste em atribuir a cada horário os membros necessários da tripulação para cobrir todas os voos programados além de desenvolver uma linha de trabalho (rostering) para cada empregado minimizando o custo total envolvido com pessoal. A pesquisa propõe como método de resolução do problema um algoritmo de busca espalhada (scatter search algoritm). Como objetivo do problema este algiritmo deve procurar por um conjunto de linhas de trabalho para cada membro da tripulação tal que minimize o custo total envolvido além de assegurar o máximo de qualidade social para a programação como um todo. Várias estratégias de melhoramento juntamente com o uso de meta-heurísticas foram empregadas de forma complementar. Testes com dados reais de problemas foram apresentados com o intuito de investigar todas as características da proposta. Além do mais, os autores também apresentaram comparações entre o método por eles proposto com método exato utilizando algoritmo branch-and-price e método de busca em vizinhança usando algoritmo de gradiente em profundidade.

Cappanera e Gallo (2004) formularam o problema de escalonamento de tripulação para empresas de aviação designado de ACR – *Airline Crew Rostering* - como um problema de fluxo de mercadorias onde cada membro da tripulação é a mercadoria que flui pela rede. O escalonamento equivale a achar um caminho no grafo satisfazendo as restrições trabalhistas. Em outras palavras, cada caminho será uma escala ou linha de trabalho (*roster*).

Para a montagem do problema, foi empregado um grafo acíclico direcionado e multicamada, G, definido como,

$G = (N, E)$ onde,

N = conjunto dos nós. Cada nó representa o horário de início e de final de uma atividade para um mesmo dia t;

E = conjunto de arcos. Será definido adiante em função dos conjuntos A, C e R, também caracterizados em seguida;

Sejam,

m = número de camadas do grafo. Cada camada equivale a um dia da linha de trabalho. Portanto, *m* também é igual ao número total de dias da linha de trabalho;

t = um dia particular da linha de trabalho;

n = número de funcionários a serem atribuídos em todo escalonamento. Também equivale ao número de linhas de trabalho a serem determinadas;

τ_i = horário associado ao início da atividade $i \in N$;

x_e^h é a variável de decisão do problema e representa o fluxo da mercadoria h (funcionário) através do arco *e*.

A é o conjunto dos arcos que representam uma atividade real da empresa. Esta atividade pode ser tanto de trabalho como de descanso;

R = subconjunto de arcos de A referente apenas as atividades de descanso;

C = conjunto de arcos que representam as transições compatíveis de uma atividade para a próxima, da camada *t* para a $t + 1$;

Portanto,

$E = A \cup C, R \subset A$;

r_{ek} é a quantidade de recursos *k* consumidos pelo arco *e*;

c_e é o número de empregados requeridos no arco *e*;

p é o número máximo de dias de trabalho entre duas atividades de descanso consecutivas;

$F_s(i,h)$ é o conjunto de arcos que saem do nó i e que podem ser usados pela mercadoria (empregado) h;

$B_s(i,h)$ é o conjunto de arcos que entram no nó i e que podem ser usados pela mercadoria (funcionário) h;

S_t é o conjunto de arcos que correspondem a atividades de descanso cobrindo $t, t+1, \dots, t+p$;

P^h é o conjunto de atividades pré-atribuídas ao funcionário no período que antecede ao do planejamento;

F^h é o conjunto de atividades que não podem ser atribuídas ao funcionário h;

A primeira fase da solução do problema é a montagem do grafo *G*. Trata-se de tarefa complexa e trabalhosa devido às restrições operacionais e trabalhistas a serem incluídas na conexão dos diversos nós do grafo. Algumas são mais simples como, por exemplo, a questão trabalhista relativa ao número máximo de dias de trabalho, *p*, entre duas folgas consecutivas.

Ela obriga a estender *G p* camadas antes do período corrente de programação para cada funcionário *h*. Para implementar tal regra é necessário que conheçamos as atividades anteriores feitas pelo funcionário *h*. Se for novo funcionário, seu passado não é conhecido, pois ele não trabalhava na empresa. Será obrigatório conectar o nó o^h (que é o nó de origem de uma linha) com todas as atividades possíveis de serem realizadas pelo funcionário *h* no primeiro dia do período atual de programação.

Certas restrições não podem ser consideradas durante a construção do grafo por serem resultados de cálculos que variam em função de como a linha vai sendo construída. Elas só poderão ser conhecidas no final, quando a linha completa for constituida. É o caso das horas de serviço e de voo que podem ser acumuladas a cada semana ou a cada mês. Mesmo problema ocorre para o número de períodos de descanso no mês, que não deve ultrapassar nove descansos, para o problema modelado.

Outro exemplo é o tempo de voo. Existem vários tipos de serviço durante o que se convencionou chamar de voo os quais não têm a ver com o ato de pilotar o avião propriamente dito. Por exemplo, o tempo que o piloto gasta desde que o avião pousou num aeroporto até o momento em que ele continuará sua escala no mesmo ou em outra aeronave, no mesmo ou em outro aeroporto. Há situações em que nesse tempo total pode-se incluir o tempo de espera por um voo desta ou de outra companhia que irá levá-lo até o novo aeroporto de onde sairá o seu próximo voo, mais os tempos de transporte da tripulação nos sentidos “hotel-aeroporto” e “aeroporto-hotel”, se existirem, mais o tempo de pernoite caso haja algum, etc. Nesta mesma categoria são incluídos outros tempos relativos a reuniões trabalhistas e períodos de treinamento.

Para finalizar tem-se que levar em consideração várias programações pré-estabelecidas para os funcionários, como saídas de férias e períodos de treinamento longo.

A partir da construção do grafo, o problema é resolvido como um problema de programação linear binária, formulado como mostrado na Formulação 2 a seguir.

Formulação 2 – O problema ACR

$$\boldsymbol{min} \sum_{e \in E} \boldsymbol{\gamma_e} \sum_h \boldsymbol{x_e^h} \quad (10)$$

Sujeito a

$$\sum_{e \in F_s(o^h,h)} \boldsymbol{x_e^h} = \boldsymbol{1}, \forall \boldsymbol{h} \quad (11)$$

$$\sum_{e \in B_s(d^h,h)} \boldsymbol{x_e^h} = \boldsymbol{1}, \forall \boldsymbol{h} \quad (12)$$

$$\sum_{e \in B_s(i,h)} \boldsymbol{x_e^h} - \sum_{e \in F_s(i,h)} \boldsymbol{x_e^h} = \boldsymbol{0}, \forall \boldsymbol{h} \forall \boldsymbol{i} \in \boldsymbol{N} \quad (13)$$

$$\sum_h \boldsymbol{x_e^h} \leq \boldsymbol{c_e}\,, \forall \boldsymbol{e} \in \boldsymbol{A} \quad (14)$$

$$\sum_{e \in S_t} \boldsymbol{x_e^h} \geq \boldsymbol{1}, \forall \boldsymbol{h} \forall \boldsymbol{t} \in \{\boldsymbol{t^h}, \boldsymbol{t^h + 1}, \dots, \boldsymbol{m}\} \quad (15)$$

$$\sum_{e \in A} \boldsymbol{r_{ek}} \boldsymbol{x_e^h} \leq \boldsymbol{r_k^h}, \forall \boldsymbol{h} \forall \boldsymbol{k} \quad (16)$$

$$\boldsymbol{x_e^h} = \boldsymbol{1},\ \forall \boldsymbol{h} \forall \boldsymbol{e} \in \boldsymbol{P^h} \quad (17)$$

$$\boldsymbol{x_e^h} = \boldsymbol{0},\ \forall \boldsymbol{h} \forall \boldsymbol{e} \in \boldsymbol{F^h} \quad (18)$$

$$\boldsymbol{x_e^h} \in \{\boldsymbol{0}, \boldsymbol{1}\}\ \forall \boldsymbol{h} \forall \boldsymbol{e} \in \boldsymbol{E} \quad (19)$$

$$[(\boldsymbol{\gamma_e} = \boldsymbol{-1}) \leftrightarrow (\boldsymbol{c_e} < +\infty) \vee (\boldsymbol{\gamma_e} = \boldsymbol{0})] \quad (20)$$

$$[(\boldsymbol{\gamma_e} = \boldsymbol{-l_c}) \leftrightarrow (\boldsymbol{c_e} < +\infty) \vee (\boldsymbol{\gamma_e} = \boldsymbol{0})] \quad (21)$$

A equação (10) representa a função objetivo a qual depende do parâmetro γ_e definido pelas funções (20 e (21) a seguir. As restrições (11), (12) e (13) são as restrições de balanço de fluxo. Para cada mercadoria (funcionário) h um fluxo unitário da mesma sai de o^h e é consumida no nó destino d^h. A restrição (14) limita o número de funcionários no arco $e \in A$ independente da mercadoria. A restrição (15) garante que o número de períodos de descanso atribuídos ao funcionário h na janela de tempo $[t, t + p]$ é maior ou igual a 1. A restrição (16) não permite que os recursos que o funcionário usará em cada trecho da linha de trabalho exceda o limite r_k^h. Exemplos destes limites são o número de voos semanais ou mensais, as horas de serviço (sem considerar horas de voo), o número de períodos de descanso etc. A restrição (17) fixa em 1 o fluxo de mercadoria (funcionário) em cada arco correspondente as atividades pré-atribuídas. A restrição (18) fixa em zero o fluxo de mercadoria em cada arco correspondente as atividades que o funcionário não pode fazer, isto é, que pertencem ao conjunto F^h. A restrição (19) nomeia a variável independente do problema x_e^h como o fluxo no arco $e \in E$ relativo à mercadoria *h*. As duas últimas equações se referem às duas funções objetivo do problema. A função (20) refere-se ao caso de máxima cardinalidade cujo objetivo

é o de cobrir o máximo número de atividades. Já a função (21), referida como o caso de máximo comprimento, pretende-se maximizar o comprimento das atividades cobertas de forma a minimizar a duração das atividades não cobertas que terão que ser atribuídas a empregados terceirizados para atendimento dos requisitos funcionais.

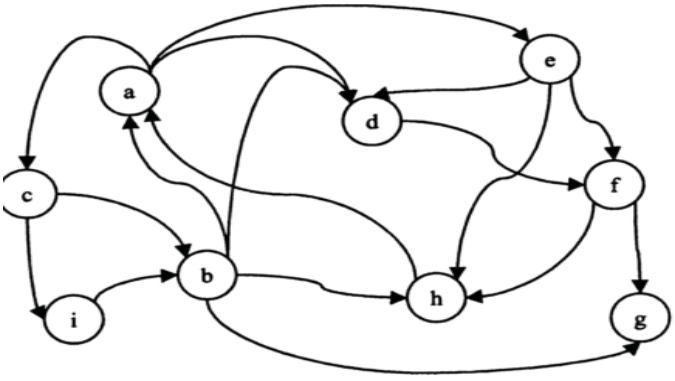
Os autores procedem a uma análise de algumas questões relativas à estrutura poliédrica do modelo de formulação do ACR e algumas restrições adicionais são incluídas na formulação já apresentada. As investigações levam em consideração algumas propriedades do modelo formulado além de definições e provas matemáticas de alguns fatos importantes para a reformulação do problema original.

Quanto aos resultados, é salientado que para instâncias médias um bom desempenho foi conseguido quando se aplica a fase de pré-processamento, isto é, a fase de construção de G.

Entre os casos de teste realizados, salienta-se o de uma instância com 40.352 colunas e 14.780 linhas. O problema foi resolvido através do otimizador CPLEX e demorou aproximadamente 4 horas para encontrar a solução ótima. O mesmo otimizador falhou em achar ótimo em 15 horas quando a fase de pré-processamento foi retirada do processo de solução.

Shodi e Norris (2004) propuseram uma solução para o problema de criação de escalas de serviço do metrô de Londres. O problema foi modelado e resolvido em três fases. Na primeira fase o problema foi representado como um grafo direcionado cíclico no qual os nós são os padrões de dias de folga semanais ou multi semanais e os arcos as transições possíveis entre os padrões, obedecendo as restrições operacionais do problema. A Figura 5 mostra um exemplo do grafo gerado nesta fase onde os nós representam padrões de folga de cada centro de pessoal do metrô.

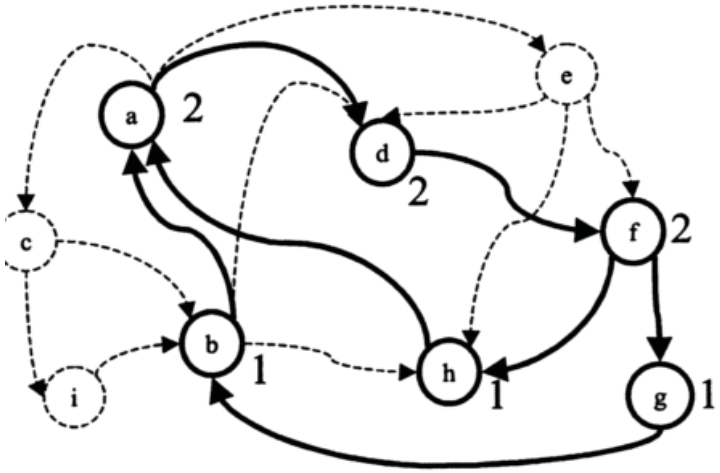
Figura 5 - Grafo direcionado cíclico da primeira fase



Fonte: (SHODI;NORRIS, 2004)

A partir deste grafo, formula-se um novo problema de programação linear inteira mista, cujo objetivo é o de encontrar um subgrafo ótimo atendendo as restrições que não foram modeladas na 1ª. fase. Um subgrafo resultante é mostrado em negrito na Figura 6.

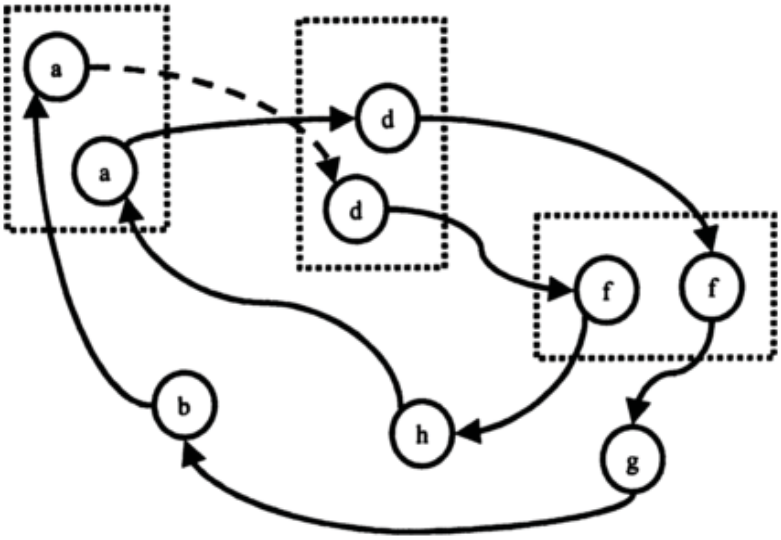
Figura 6 - Exemplo de um subgrafo ótimo



Fonte: (SHODI;NORRIS, 2004)

Depois de encontrado um subgrafo ótimo, um algoritmo de busca em profundidade percorre o mesmo a procura de um caminho válido. O primeiro caminho encontrado é a primeira escala produzida pela heurística proposta. O resultado é mostrado na Figura 7.

Figura 7 - Primeira escala válida do escalonador (*roster*)



Fonte: (SHODI;NORRIS, 2004)

Na Figura 7, a escala encontrada é a escala ‘d-f-h-a-d-f-g-b-a’ na qual se pode ver que, na última semana, o condutor que cumpriu o turno ‘a’ volta ao seu ponto de partida que é o turno ‘d’. Isto mostra o caráter cíclico deste problema. Como resultado interessante é descrito um procedimento para medir a otimalidade da escala produzida. Segundo os dados experimentais mostrados, em estações do Metrô com 30 a 50 condutores, o modelo demorou entre segundos a minutos para achar uma solução com 98% de otimalidade.

Neste modelo, o número de variáveis inteiras é igual à soma do número de arcos mais o número de nós e o número de restrições é igual a duas vezes o número de nós. Para padrões de dias de folga de uma a duas semanas há 500 nós e 100.000 arcos. O programa foi codificado em C++ para Workstation Sun Sparc Ultra.

Este enfoque de resolução foi um dos mais interessantes em relação ao uso de teoria de grafos. Porém, comparando este problema com o nosso vemos que no nosso problema o grafo teve que ser acíclico, pois as viagens ocorrem sequencialmente até o final do período de construção das linhas de trabalho.

Caprara et al. (1997) apresentaram uma ideia baseada na teoria de grafos para representar o problema de gerenciamento de tripulação para empresas de transporte ferroviário. O problema é tratado em duas fases, a de programação de curto-prazo com viagens diárias agrupadas num curto intervalo de tempo, por exemplo, dois dias de duração, e a fase da criação de escalas propriamente ditas, agrupando os programas de curto prazo em linhas ou escalas de maior duração, tipicamente de 30 a 60 dias.

O problema da primeira fase é formulado como um problema de cobertura de conjuntos não unicusto. Os autores relembram que na literatura, até aquela data, os algoritmos existentes resolviam instâncias de algumas centenas de linhas a alguns milhares de colunas e quando maiores instâncias de problemas eram envolvidas, dever-se-ia recorrer a “soluções heurísticas”. Neste contexto, os algoritmos gulosos são comuns; porém, embora rápidos, não oferecem boas respostas do ponto de vista de otimização. Os trabalhos mais recentes baseados em Relaxação Lagrangeana e, em particular Beasley, Bull e Martin (1993), Beasley (1990), Beasley e Chu (1996), conduzem a um bom equilíbrio entre a velocidade e qualidade final das soluções.

O interesse é verificar como os autores procederam para a representação e posterior equacionamento e solução do problema na segunda fase, já que é um tipo de problema que se assemelha ao nosso.

No levantamento do problema analisado, observou-se que são levados em consideração um conjunto detalhado de “janelas de intervalo” entre tarefas, bem como regras operacionais

para mudar de uma para outra tarefa, as quais são pouco comuns em empresas de transporte rodoviário como é o caso do nosso problema. Porém, formam um conjunto abrangente de restrições que podem ser empregadas em qualquer problema de escalonamento ferroviário, pois devem cobrir a maioria das restrições presentes em problemas desta natureza.

A título de exemplo veremos algumas restrições a seguir. Começaremos pelo tipo de intervalo entre tarefas. Existem dois tipos. Um primeiro denominado de “folga simples” e um segundo denominado “folga dupla”. Ambos são restrições obrigatórias. O intervalo simples implica em se dar ao funcionário 48 horas consecutivas de descanso. O intervalo duplo deve obrigatoriamente incluir mais dois dias completos de descanso, os quais podem ocorrer juntos ou entre a folga simples, isto é, um dia antes da folga simples e outro dia depois da folga simples.

Há restrições quanto ao local onde a tripulação tirará suas folgas. Elas podem ser externas ou internas ao local de trabalho.

Outras restrições dizem respeito a limites inferiores tais como quantidade mínima de tarefas que um condutor deve atender na escala e valores mínimos de duração das escalas. Quanto menor for a duração das escalas na solução final, menor será o custo para a empresa, dado que estas escalas são pagas pelas horas trabalhadas.

Outra restrição operacional é que o total de períodos de descanso de todas as escalas que fazem parte da solução final deve ser igual ao número de semanas presentes na escala e o total de períodos de folga dupla deve ser no mínimo 40% do valor total de folgas da escala.

Foram determinados vários tipos de tarefas no problema, os quais devem ser considerados na modelagem. Há “tarefas longas”, isto é, as que não incluem descanso externo e cujas horas trabalhadas são maiores do que 8 horas e 5 minutos. As tarefas noturnas que são aquelas que requerem algum trabalho entre meia-noite e cinco horas da manhã. Tarefas noturnas pesadas, por sua vez, são as que requerem um período contínuo de trabalho maior do que 1 hora e 30 minutos entre meia-noite e 5 da manhã, sem descanso externo.

Estas restrições operacionais são modeladas inicialmente utilizando teoria de grafo. Os vértices são associados às viagens a cumprir. Emprega-se um multigrafo direcionado cíclico, $G = (V,A)$, $V = \{1,2,...,n\}$ onde n é o número de viagens total no período pré-estabelecido de programação da companhia. Os arcos $(i,j) \in A$ correspondem às transições entre duas viagens consecutivas i e j. Três tipos de transições são possíveis entre um vértice e o posterior: transição direta sem descanso, transição com descanso simples e transição com descanso duplo.

Portanto A será formado por estes três subconjuntos de arcos, que podem ser definidos como $\{A_k: k = \{1,2,3\}\}$. A variável de decisão x_{ij}^k é igual a 1 se o arco $(i,j) \in A$ estiver na solução ótima e 0 caso contrário. Os custos de um arco$(i,j) \in A$ são funções do seu tipo e podem ser designados por c_{ij}^k. Um valor r representa o número mínimo de arcos de descanso simples ou duplo na solução, e uma variável z representa o número mínimo de arcos de descanso duplo na solução final. Seja $\alpha = 8640$ minutos, o número de minutos em uma semana.

Cada percorrimento de *G* corresponde a uma linha de trabalho ou escala e o custo do percorrimento é medido pelo tempo requerido para fazer as respectivas viagens do percorrimento em um determinado prazo de programação.

O problema a se resolver consiste em determinar um conjunto disjunto de caminhos de custo mínimo que satisfaçam as restrições e cubram todos os nós do grafo uma única vez. Algumas condições denominadas pelos autores de relaxações são introduzidas ao longo do estudo. Por exemplo, como já mencionado, a quantidade total de arcos de descanso duplos tem que ser no mínimo igual a 0,40 vezes o número total de arcos de descanso simples mais duplos. No final, o valor ótimo encontrado deverá minimizar o somatório das durações de todas as tarefas envolvidas sem deixar nenhuma tarefa a descoberto. Como o tempo de trabalho é pago, estamos ao mesmo tempo minimizando o custo total da operação.

Uma vez definidas as variáveis de decisão do problema, os custos, restrições e relaxações, o problema representado pelo grafo G é resolvido como um problema de programação linear inteira designado problema RP, cuja formulação $v(RP)$ é fornecida como Formulação 3.

Formulação 3 – O problema ferroviário com relaxação

$$\boldsymbol{v(RP) = min \sum_{i=1}^{n} \sum_{j=1}^{n} \left(c_{ij}^1 x_{ij}^1 + c_{ij}^2 x_{ij}^2 + c_{ij}^1 x_{ij}^3\right)} \quad (22)$$

$$\boldsymbol{\sum_{i=1}^{n} \left(x_{ij}^1 + x_{ij}^2 + x_{ij}^3\right) = 1, j = 1, \dots, n} \quad (23)$$

$$\boldsymbol{\sum_{j=1}^{n} \left(x_{ij}^1 + x_{ij}^2 + x_{ij}^3\right) = 1, i = 1, \dots, n} \quad (24)$$

$$\boldsymbol{r \geq (1/\alpha) \sum_{i=1}^{n} \sum_{j=1}^{n} \left(c_{ij}^1 x_{ij}^1 + c_{ij}^2 x_{ij}^2 + c_{ij}^1 x_{ij}^3\right)} \quad (25)$$

$$\boldsymbol{\sum_{i=1}^{n} \sum_{j=1}^{n} \left(x_{ij}^2 + x_{ij}^3\right) \geq r} \quad (26)$$

$$\boldsymbol{z \geq 0.4r} \quad (27)$$

$$\boldsymbol{\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij}^3 \geq z} \quad (28)$$

$$\boldsymbol{x_{ij}^1, x_{ij}^2, x_{ij}^3 \in \{0,1\}, \quad i,j = 1, \dots, n} \quad (29)$$

$$\boldsymbol{r, z \geq 0, \quad r, z \; inteiros} \quad (30)$$

Problemas deste tipo são classificados como NP-completos e difíceis de resolver. Uma série de métodos de programação, entre eles o método de Decomposição de Dantzig-Wolfe, bem com o método de Relaxação Lagrangeana, são técnicas que podem ser empregadas nestes casos. Os autores empregaram um algoritmo baseado na técnica de Relaxação Lagrangeana com multiplicadores não negativos λ_1 e λ_2 nas restrições (22) e (24) transformando o sistema de equações anterior em outro problema de minimização denominado problema de minimização de Lagrange, $v\big(LRP(\lambda_1, \lambda_2)\big)$.

O valor ótimo do problema transformado, $v(LRP(\lambda_1, \lambda_2))$ é um valor válido de limite inferior para o problema principal $v(RP)$ e possui ordem de complexidade $O(n^3)$.

Neste ponto da heurística proposta, um procedimento heurístico de construção das linhas de trabalho que usa a informação obtida na solução do problema relaxado definido anteriormente é executado. São escolhidos trechos da linha que serão inseridos na nova linha de trabalho que está sendo criada. Uma vez completada a primeira linha de trabalho viável, todos os trechos constantes da mesma são retirados da formulação do problema. Uma nova iteração tem início a qual produzirá uma nova linha e assim por diante até todas as linhas serem construídas. O Algoritmo 2 exibe o comportamento da heurística na construção de uma linha de trabalho logo após o primeiro trecho de viagens ter sido escolhido.

Algoritmo 2 – Iterações executadas para a construção de uma linha de trabalho

i) **Repetir**
   a. **Selecionar** a próxima melhor viagem para fazer parte da escala atual, levando em consideração o aumento por ela proporcionado no limite inferior pela escolha do arco (i,j) e as características da viagem j;
   b. **Atualizar** o limite inferior de Lagrange (em $O(n^2)$);
   c. **Verificar** se o escalonamento pode ser concluído atualizando a melhor linha encontrada;

ii) **Até** que nenhuma linha de trabalho melhor possa ser construída ou a linha atual ultrapasse 10 semanas de comprimento.

Finalmente quando todas as linhas de trabalho são obtidas, um procedimento de refinamento ainda pode ser aplicado para remover da solução final as últimas linhas obtidas, pois estas sempre serão piores do que as primeiras. Em seguida, deve-se reaplicar o procedimento heurístico.

Alfares (2004) utilizou de modelos de cobertura de conjuntos na formulação de problemas de "*tour scheduling*" ou programação de turnos para o atendimento de tarefas

especializadas como as de enfermagem, de manutenção industrial como trabalhos em centrais nucleares, chãos de fábrica em geral, tripulação de submarinos etc. O modelo de programação inteira usado, quando a força de trabalho é constituída por trabalhadores da mesma especialidade e com o mesmo nível de conhecimento entre si e os *tours* têm o mesmo custo (problema de cobertura unicusto), ainda é o mesmo encontrado em muitos artigos clássicos como o de Dantzig (1954) e pode ser observado na Formulação 4.

Formulação 4 – Formulação tradicional do problema de *tour scheduling*

$min\ W = \sum_{j=1}^{J} x_j$
Sujeito a
$\sum_{j=1}^{J} a_{ij} x_j \geq r_i,\ i = 1, \dots, I$
$x_j \geq 0$, j = 1,...,J
Onde,
W é a força de trabalho total alocada;
x_j quantidade de funcionários alocados no *tour j*;
$a_{ij} = 1$ se período i for um período de trabalho para o tour j, a_{ij}=0 caso contrário;
r_i número mínimo de funcionários requeridos no período *i*;
I número de períodos de tempo a serem considerados na programação para uma semana;
J número de *tours* a serem considerados na programação;

A partir desta formulação, várias propostas de solução são analisadas em detalhes no trabalho. Dentre elas, são de particular interesse as meta-heurísticas usadas. Os três tipos mais citados são Recozimento Simulado, Algoritmo Genético e Busca Tabu. Das três a mais promissora na época era a Recozimento Simulado.

Bautista e Pereira (2006) descreveram um problema de coleta de lixos da cidade de Barcelona formulando o mesmo como um problema de cobertura de conjuntos. O interesse para a nossa pesquisa neste artigo foi o de estudar melhor a representação por cobertura de conjuntos de forma a complementar o conhecimento já adquirido sobre a mesma em casos práticos.

Primeiramente apresentam a notação de grafos a ser empregada. Dois grafos são utilizados. O *GL* e o *GP*. *GL* será usada para representar os possíveis locais de instalação das caçambas de coleta de lixo. *GP* representará os pontos onde as demandas de coleta de lixo ocorrem. Trata-se de dois modelos discretos de otimização resolvidos em duas fases. 1) Achar um número mínimo de pontos em *GL* tal que, para qualquer ponto *f* em *GP*, existe um ponto em *GL* cuja distância para *f* seja menor ou igual a *L*, um valor pré-definido. Então se diz que

cada ponto de *GP* é coberto por um número mínimo de pontos de *GL*. 2) Achar um número mínimo de pontos em *GL* que cubra o máximo número de pontos em *GP*. Trata-se de um problema multiobjetivo. De um lado se procura minimizar o número de pontos de *GL*, do outro se procura maximizar o número de pontos cobertos em *GP*. Como o modelo inicial é um modelo matemático contínuo, primeiramente ele deve ser discretizado para ser resolvido como um problema de cobertura de conjuntos. A discretização usada é baseada em Tamir[2] (1985 apud BAUTISTA 2006, p. 619).

As caçambas podem ser alocadas nas arestas ou fora do *GP*. Em ambos os casos, os pontos distam entre si e dos vértices do extremo dos arcos um inteiro positivo. Trata-se de um modelo de cobertura de conjuntos com o custo de colocação das caçambas sempre o mesmo (custo uniforme).

Os autores utilizaram as representações do problema como problema de cobertura de conjuntos e de MAX-SAT. O problema formulado como de cobertura de conjuntos foi resolvido através de Algoritmo Genético e o problema de MAX-SAT foi resolvido através de heurística GRASP.

Panthon e Ryan (1999) estudaram o problema de alocação de mão de obra em tarefas com múltiplos *breaks*. Exemplo de restrições desse problema são o número mínimo e máximo de trabalhadores com um determinado perfil necessários para o atendimento das tarefas. As variáveis são os turnos os quais são formados por tempos de trabalho intercalados por tempos de descanso. Os turnos são divididos nos de tempo cheio (8 horas) e de tempo parcial (4 horas). Outra restrição é a razão entre o número de trabalhadores de turno cheio para o número de trabalhadores de turno parcial. Caso o problema não tenha *breaks* pode ser modelado como um problema de cobertura de conjuntos unicusto. Os múltiplos *breaks* bem como a existência de diferentes tipos de turnos aumentam a complexidade. Os autores propõem a resolução através de dois métodos. O de adição de colunas e o de geração de colunas.

No método de adição de colunas, primeiramente o problema é resolvido como um problema de programação inteira mista (MIP), (WOLSEY, 1998). A partir das colunas ótimas obtidas (sem *breaks*) são adicionadas novas contendo possíveis combinações de *breaks*. O custo de cada coluna original não básica no novo problema é penalizado de forma que elas não entrem mais nas soluções posteriores. As colunas que têm breaks muito próximos uns dos outros possuem custos menores. O problema é resolvido através do método simplex e caso

[2] TAMIR, A. **A finite algorithm for the continuous p-center location problem on a graph.** Mathematical programming, 1985, 31, p.298-306.

apareçam na solução final colunas sem *breaks*, o processo é repetido. Se não aparecerem mais colunas sem *breaks* o procedimento é interrompido com esta solução. Como o problema partiu de uma solução inicial sem *breaks* é possível que a solução não seja ótima. Mesmo assim o algoritmo é encerrado.

No método de geração de colunas, ao invés de calcular o custo reduzido das colunas não básicas do problema, como é o caso do método simplex revisado, emprega-se um modelo de rede representando o problema através de um grafo. Em seguida, usa-se o método do caminho mínimo para determinar quais colunas poderão entrar na base. Tipicamente os nós se referem às linhas da matriz de cobertura e os arcos são os custos relacionados às variáveis duais associadas a cada linha. Cada caminho válido representa uma coluna a ser inserida na base. Com esta técnica, ao invés de manter as colunas em memória, o que se torna um obstáculo quando a instância do problema cresce, o custo reduzido de todas as colunas não básicas é calculado através do grafo, não necessitando manter todas em memória. O percurso no grafo é feito empregando-se o algoritmo bem conhecido de caminho mínimo de *Dijkstra*. O modelo de programação linear é resolvido diversas vezes até que não haja mais caminhos com custos reduzidos negativos. A solução neste momento é não inteira por causa da relaxação por programação linear aplicada ao modelo inicial que era um problema inteiro. Alguns procedimentos para tornar inteira a solução do problema de programação linear são sugeridas. Entre elas a técnica de *branch-and-bound*, heurísticas de arredondamento etc.

Testes realizados com conjuntos de dados da literatura mostram que, para pequenas instâncias do problema, o método de geração de colunas é mais lento do que o de adição de colunas (possivelmente devido à sobrecarga gerada pelo mesmo). Mas quando cresce o número de colunas consideravelmente, o método de adição de colunas se torna ineficiente. No nosso caso, o número de colunas a ser gerado é muito alto, e por isto deveríamos utilizar o método de geração de colunas adicionando os aperfeiçoamentos sugeridos pelos autores no algoritmo de *branch-and-bound*.

2.3.2 Resolução do problema de cobertura de conjuntos

Naji-Azimi, Toth e Galli (2010) propuseram o uso de uma meta-heurística baseada nas leis do eletromagnetismo para resolver o problema de cobertura de conjuntos unicusto. Ela funciona da seguinte forma: cada solução do problema representa uma partícula carregada de carga positiva ou negativa. Um conjunto de partículas compõe uma população como as

populações em algoritmos genéticos. A carga de cada partícula é calculada através da função objetivo do problema. No espaço eletromagnético elas atraem e repelem umas as outras dependendo do valor das cargas de cada partícula. Uma partícula exerce força sobre todas as demais presentes no espaço eletromagnético. Cada partícula recebe uma força resultante a qual a impulsiona numa determinada direção do espaço. As que têm maior carga são atraídas e as de menor carga repelidas. A medida que o algoritmo converge as melhores soluções vão ficando separadas das piores. Após um tempo ou um número máximo de iterações o procedimento é interrompido e a melhor solução é obtida.

Uma comparação com os resultados obtidos por outros pesquisadores é feita a partir dos resultados anteriores obtidos por Bautista e Pereira (2007) e Lan, Depuy e Whitehouse (2007).

Foi empregada neste estudo a Formulação 1 como representação do problema de cobertura de conjuntos não unicusto.

É interessante observar que os autores classificaram este problema de cobertura de conjuntos não unicusto como NP-completo.

O uso desta meta-heurística foi interessante para verificar comparativamente a sua eficácia e eficiência em relação a outros métodos heurísticos de solução do problema de cobertura de conjuntos.

Seu procedimento de busca local é idêntico ao observado em outras heurísticas do tipo GRASP, (LAN;DEPUY;WHITEHOUSE, 2007), (CAPRARA;FISCHETTI;TOTH, 1999).

O processador utilizado nos testes foi um Duo Core da Intel a uma frequência de 1.7 GHz e contendo uma memória apenas de leitura (RAM) de 1.0 GB. Naji-Azimi, Toth, e Galli (2010) salientam que se trata de classe de processador semelhante ao usado em 2007 pelos outros pesquisadores em suas pesquisas e por isto os resultados são perfeitamente compatíveis para comparação. A linguagem de programação empregada foi a linguagem C.

Além de comparar os resultados com os de outras meta-heurísticas, também foi verificado o quanto os procedimentos exclusivos desta meta-heurística, isto é, o cálculo de forças e do movimento das partículas no espaço de solução do problema influencia nos resultados finais. Para isto, estas duas partes foram retiradas do algoritmo deixando apenas em funcionamento o procedimento guloso. O resultado foi uma “piora” nas soluções obtidas.

Por outro lado, para a verificação da eficácia destes dois cálculos, o de força e o de movimento das partículas, suas duas estruturas foram retiradas e a parte de pré-processamento da mesma foi inserida em outra heurística. O objetivo foi observar se o bom nível de respostas se deveu apenas ao funcionamento da fase construtiva e de busca local ou, pelo contrário, se os procedimentos retirados é que fazem a diferença.

Foi escolhida a heurística evolutiva por sua estrutura de funcionamento semelhante a da meta-heurística baseada na teoria do eletromagnetismo. Para este novo ensaio, foi escolhida a heurística baseada em algoritmo genético proposta por Beasely e Chu (1996). Os principais procedimentos desta não sofreram alteração. A eles foram adicionados apenas os procedimentos de pré-processamento construtivo e de busca local. Chamou-se a nova heurística criada "*Genetic* Algorithm modificado" ou simplesmente "GA modificado".

O resultado mostra que a eletromagnética é superior tanto em rapidez quanto em qualidade da solução em relação à GA modificado, mostrando, desta forma, a importância deste novo enfoque. Para demonstrar que tal raciocínio é verdadeiro foram comparados os resultados obtidos pela meta-heurística eletromagnética original com os resultados obtidos pelo emprego da GA modificado nos conjuntos de teste 4,5,6,A,B,C,D,NRE, NRF, NRG, NRH da OR-Library, (BEASLEY, 1990).

O GA modificado obtém a melhor solução em apenas um caso de teste e é pior do que a meta-heurística eletromagnética original em um total de quinze casos. Neste único caso, se os parâmetros da eletromagnética são ajustados ela obtém a melhor solução.

Em relação ao tempo de processamento, considerando apenas os casos em que os dois métodos chegaram ao mesmo valor da função objetivo, o método que obtém a resposta em menor tempo é o da meta-heurística eletromagnética. Considerando apenas os 39 experimentos em que os dois métodos chegam ao mesmo resultado, a eletromagnética gasta 1.744,32s, com uma média de 44,73s por experimento enquanto que a GA modificado leva no total 3.053,81s o que dá a média de 78,30s por experimento.

Para que seja possível aplicar a meta-heurística eletromagnética em problemas **não unicusto**, que é o nosso caso, os autores necessitaram alterar a parte construtiva do pré-processamento e a parte referente ao procedimento de eliminação de colunas redundantes ambas do algoritmo principal. Na parte do procedimento construtivo, o critério para saber se uma coluna *j* deve entrar na lista de candidatas deve ser mudado para levar em consideração um valor de *fitness* (adaptação) definido como $c_j/k_j^2 < (1+\alpha)p$ onde p é o menor valor de *fitness* da solução atual, c_j é o custo da coluna *j* e k_j^2 é o quadrado do número de colunas descobertas que a coluna *j* cobre. No processo da eliminação de colunas redundantes eles sugerem verificar se o custo da coluna a ser eliminada é maior do que o custo global de um subconjunto de colunas que cobrem as mesmas linhas.

A nova heurística assim modificada consegue encontrar o melhor resultado conhecido em 53 casos. Para os 12 restantes encontra soluções piores. Em termos de tempo médio de

solução para cada caso de teste a eletromagnética consome 28,55s enquanto a meta-heurística Meta-RaPS resolve em 44,10s.

Tornando variável a semente do procedimento de geração de números aleatórios e executando a mais apenas 5 casos de teste a melhor solução conhecida é alcançada em todos os casos e o tempo de computação é comparável com os da meta-heurística Meta-RaPS.

Portanto, conclui-se que ela pode ser empregada tanto em problemas de cobertura de conjuntos unicusto quanto em problemas não unicusto, pois obtém os mesmos valores de ótimos conhecidos até o momento, melhorando inclusive alguns, e o tempo de processamento está dentro do padrão de tempo atualmente encontrado na literatura para problemas do mesmo porte.

No nosso caso, as instâncias do problema são muito maiores e não há análise de aplicação desta meta-heurística para casos deste porte na literatura atualmente.

Em outra pesquisa analisada, Bautista e Pereira (2007) estudaram uma técnica que utiliza uma meta-heurística do tipo GRASP para resolver o problema de cobertura de conjuntos unicusto. Nesta proposta, um procedimento de melhoria local usando a mesma técnica empregada para resolver o problema de satisfabilidade de cláusulas disjuntivas foi incorporado à heurística GRASP.

O algoritmo proposto possui as seguintes duas fases:

1) Fase 1 – Construir nova solução usando procedimentos do tipo GRASP quando o problema for do tipo máxima satisfabilidade (MAX SAT) ou de atribuição aleatória ou randômica quando o problema for do tipo de satisfabilidade (SAT).
2) Fase 2 – A partir da solução encontrada na primeira fase, na segunda fase propõe-se trabalhar com busca local para melhorar a solução atual. Tal procedimento será conduzido iterativamente até que um máximo número de tentativas seja atingido ou que não haja mais melhora na solução atual.

O algoritmo foi programado em C. O compilador utilizado foi o gcc, da organização GNU, versão 3.2 com otimização –O3. O processador empregado foi um Pentium 4, em 1,8 Ghz, com 512 Mb de RAM. O sistema operacional empregado foi o Linux. As instâncias de problema usadas nos testes foram retiradas da biblioteca OR-Library, (BEASLEY, 1990).

A curva da variação da função objetivo em função do número de iterações tem o aspecto de uma função exponencial negativa convergindo para um valor mínimo.

Os resultados do experimento são comparados com os resultados das heurísticas propostas por Grossman e Wool (1997), trabalho este que também analisamos, como se descreverá a seguir, chegando-se à conclusão de que a proposta de Bautista e Pereira (2007) melhora os resultados obtidos em 47 instâncias das 70 analisadas. No entanto, o procedimento que lida com o problema de máxima satisfabilidade requer um tempo muito grande de processamento, o que inviabiliza o uso da proposta para o nosso caso.

Outra observação importante é sobre a densidade das matrizes de cobertura das várias instâncias envolvidas. Quanto menos densa a matriz de cobertura melhor é o desempenho da heurística proposta. Para matrizes muito densas, o desempenho da heurística proposta piora em relação ao dos algoritmos das heurísticas comparadas, os quais melhoram muito nestas circunstâncias encontrando inclusive soluções muito próximas dos ótimos quando conhecidos.

Outro artigo que mereceu nossa revisão na pesquisa vertical foi o de Grossman e Wool (1997) o qual trata de uma revisão dos principais métodos de solução empregados para o problema de cobertura de conjuntos “unicusto” até o ano de 1997. “Nove” algoritmos heurísticos são apresentados no estudo. Seis usam métodos determinísticos, isto é, a modelagem é feita com variáveis discretas e conhecidas. Três usam métodos estocásticos, isto é, os modelos usam variáveis aleatórias. Eles são apresentados a seguir:

Algoritmos usando métodos determinísticos:

1) Algoritmo do tipo “*greedy*” (guloso) simples (Gr);
2) Heurística baseada na generalização de Gavril (NoLP);
3) Heurística gulosa alternativa (Alt-Gr);
4) Algoritmo de Hochbaum (Thresh);
5) Algoritmo em que as variáveis são checadas na ordem decrescente de seus valores fracionários (SortLP);
6) Algoritmo T-Gr que usa uma regra gulosa para obter as variáveis da solução do problema e quando há empate retira a primeira variável com o maior valor.

Algoritmos usando métodos estocásticos:

7) Método de Rede Neural (NN);
8) Algoritmo aleatório com arredondamento (RR);
9) Algoritmo guloso com método aleatório (R-Gr);

É interessante salientar três métodos de solução que, embora não estudados no artigo, foram reportados na época em que o mesmo foi escrito. Um algoritmo exato baseado em

busca em árvore (*tree-search*) que segundo Grossman e Wool (1997) só deu resultado satisfatório para pequenas instâncias da matriz de cobertura. Tipicamente usando matrizes de até 50 linhas por 500 colunas. Uma heurística evolutiva baseada em algoritmo genético e outra heurística baseada no recozimento simulado de cristais (*simulated annealing*).

Para o estudo Grossman e Wool (1997) realizaram testes em matrizes cujo número de colunas variou de 500 a 5.000 colunas, sendo geradas de forma aleatória. E também em dois problemas de natureza combinatória analisados, a dimensão da matriz de cobertura nas respectivas instâncias, chegou a 28.160 x 11264.

Também é interessante saber que de muitas propostas de redes neurais encontradas na época, nenhuma foi utilizada para resolver o SCP.

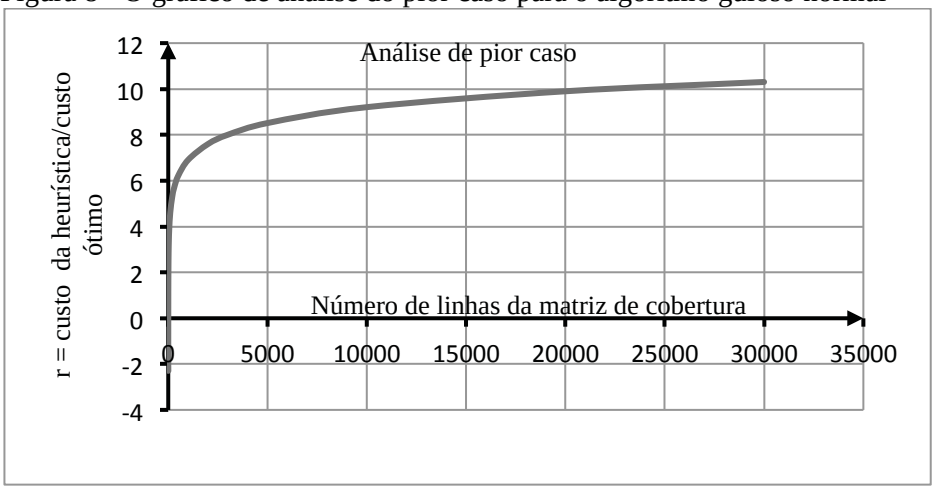
Outro aspecto que vale a pena mencionar é que houve um caso em que a heurística analisada chegou a ser empregada em instância real na área de programação de tripulação de companhias aéreas.

Em alguns algoritmos empregados, os autores desenvolveram uma análise do pior caso em função dos parâmetros de cada problema em particular. Esta análise será apresentada em seguida, pois foi útil para o nosso estudo. Será apresentada a curva correspondente a análise de pior caso realizada para o Algoritmo Gr.

Esta análise compreende a determinação da razão entre o custo da solução obtida pela heurística G_r, z_h, e o custo da solução ótima conhecida, z_O. É interessante como medida do desempenho de uma heurística qualquer e pode ser expressa por $r = z_h/z_O$, onde z_h é o ótimo da heurística e z_o o ótimo conhecido.

Segundo Grossman e Wool (1997), esta razão pode ser aproximada à função do logaritmo neperiano do número de linhas da matriz de cobertura mais um. Ou seja, $r = \ln(m) + 1$ onde m é o número de linhas da matriz de cobertura. O gráfico da Figura 8 a seguir ilustra esta curva para diferentes valores de *m*.

Figura 8 - O gráfico de análise do pior caso para o algoritmo guloso normal



Fonte: (GROSSMAN; WOOL, 1997)

A Figura 8 mostra uma tendência interessante. A razão r tende a ficar constante a partir de um certo número de linhas, no caso maior do que 30.000, refletindo o fato de que a partir deste valor o número de linhas da matriz de cobertura pode crescer que não influenciará tanto no desempenho da heurística.

Já nos algoritmos baseados em aproximação, esta razão é proporcional diretamente ao número máximo de uns das linhas da matriz de cobertura.

Nos algoritmos do tipo do SortLP com métodos randômicos a razão anterior vale $r = k\left(1 - (c/m)^{1/k}\right)$ onde k é o número máximo de 1s por linha e c é uma constante de valor pequeno pré-definida.

Voltando à questão dos tipos de dados empregados nos testes, a massa de dados foi a descrita por Beasley (1990), já apresentada.

Nos problemas de natureza aleatória os resultados dos algoritmos são comparados entre os próprios algoritmos estudados. Para os problemas de natureza combinatorial, o resultado é comparado com o melhor da literatura obtido até então.

O melhor método para os problemas aleatórios (60 problemas) foi o de redes neurais, pois conseguiu encontrar o melhor resultado num maior número de casos de teste. Em 60 casos conseguiu encontrar a melhor solução em 53.

Nos problemas combinatoriais os melhores algoritmos foram os gulosos para o problema do hipercubo e o de redes neurais no de coloração, o qual perdeu apenas em uma instância que foi a CLR-12.

Finalizando esta seção sobre pesquisa vertical, o último estudo aqui referido é o estudo de Beasley e Chu (1996) propriamente dito, o qual se destaca pelo grande número de citações

na literatura. Foram 172 entre 1996 e 2010. Sua heurística inclusive foi a escolhida como base da implementação de nossa proposta de solução na fase de otimização. Sugerimos adaptações ao mesmo para ser usado em nosso estudo, além do trabalho de adaptação necessário à integração com a heurística responsável pela construção das linhas de trabalho, a qual é baseada na teoria de grafos.

O estudo de Beasley e Chu (1996) se baseia no algoritmo genético clássico proposto originalmente por Holand (1975). Nele foram incluidas novas estratégias para a resolução do problema de cobertura de conjuntos não unicusto.

Não o descreveremos nesta seção, pois seria duplicar a descrição do mesmo que já é feita durante o desenvolvimento de nossa proposta no Capítulo 3.

Das alterações por nós introduzidas no original de Beasley e Chu (1996) se destacam: 1) novo operador de viabilização, 2) novo procedimento para formar a população inicial, 3) novo critério de cruzamento além dos procedimentos de interface entre este procedimento e o procedimento de construção das linhas de trabalho.

2.3.3 Conclusão da pesquisa vertical

A pesquisa vertical mostrou diversos enfoques e alternativas de resolução do problema de criação de linhas de trabalho dentro do contexto de diversos tipos de problemas de escalonamento de pessoal. Mostrou também que o problema de cobertura de conjuntos tem sido bastante empregado para representação de diversos problemas reais, e que existem várias técnicas para resolvê-lo, em particular aquelas usando algoritmos heurísticos.

Em particular, a técnica de solução utilizando uma heurística evolutiva baseada em algoritmo genético mostrou-se adequada para ser aplicada a nosso problema. Aliando esta técnica ao uso de grafo, o qual permite modelar restrições difíceis e adicionando técnicas de percorrimento específicas na formação das linhas de trabalho, conseguiu-se obter um conjunto de linhas qualificadas e viáveis que pode ser submetido iterativamente ao procedimento evolutivo e assim resolver de maneira satisfatória o problema proposto.

A descrição destes procedimentos encontra-se detalhada no Capítulo 3.

3 PROPOSTA DE SOLUÇÃO

O problema em estudo, caracterizado no Capítulo 1 como um problema de Otimização Combinatória, foi modelado por meio da equação (8). Neste capítulo apresenta-se uma proposta de método de solução para este problema. No algoritmo proposto há duas operações relevantes, a de percorrimento e a de otimização. O Algoritmo 3 apresenta o método proposto para solucionar o problema em estudo.

Algoritmo 3 – Método de solução proposto

1. Criar o espaço de busca para o problema, respeitando o conjunto de restrições R (obtenção do conjunto X de linhas de trabalho);

2. Repetir

3. Percorrer o espaço de busca gerando linhas de trabalho **viáveis** (obtenção do conjunto $F: F \subseteq X$);

4. Buscar em F o subconjunto de linhas de trabalho $S: S \subset F$ que melhora o valor da função objetivo do problema;

5. Otimizar S fazendo obedecer a restrição $r_{5;}$

6. Até que o critério de parada seja atingido.

7. Resolver o problema dos padrões de folga;

8. Associar cada elemento do S final (S^*) aos padrões de folga encontrados no passo 7;

A ideia do Algoritmo 3 é dividir o espaço de busca do problema, que é muito grande, em vários subespaços de busca menores, representados pelos conjuntos F.

Assim, ao invés de resolver apenas um, serão resolvidos diversos problemas menores a cada iteração, até que um número máximo de iterações seja atingido ou que não haja variação significativa da função objetivo entre soluções consecutivas.

O Algoritmo 3 possui três tarefas as quais podem ser identificadas através dos passos (1), [(2) - (7)] e (8). O passo (1) é o responsável pela construção do espaço X. Nos passos de (2) a (6) ocorrem os percorrimentos e otimizações necessárias para encontrar uma boa solução final, S^*, e no passo (7) é resolvido o problema de determinação do número de expedientes e folgas, aqui designado por problema de padrões de folga. Finalmente, no passo (8) os padrões de folga resultantes da solução deste problema são associados às linhas de trabalho do

conjunto S^*. As fases construtiva e de otimização são os principais pilares da heurística proposta. Elas estão embutidas dentro dos passos do Algoritmo 3 e se referem exclusivamente ao problema da construção de linhas de trabalho. A fase construtiva está delimitada pelos passos de (1) até (3) e a de otimização está presente nos passos (4) e (5). Os passos (7) e (8) são relativos à resolução do problema dos padrões de folga e sua associação posterior às linhas de trabalho e foram resolvidas de maneira independente do problema de construção de linhas de trabalho nesta pesquisa.

Neste trabalho, o espaço de busca é representado por um grafo único cujas estruturas em forma de listas entrelaçadas oferecem flexibilidade para visitar rapidamente qualquer região deste espaço. Durante a montagem do grafo, é obedecida a restrição r_1, referente ao período de descanso que deve existir entre duas viagens consecutivas da escala de serviço. Criado o espaço de busca, a próxima tarefa é gerar um conjunto *F* de linhas de trabalho. Esta tarefa é desempenhada por um procedimento de percorrimento no grafo, o qual também observa a questão de viabilidade das linhas geradas para a fase de otimização. Embutido no próprio passo (3) do Algoritmo 3 um procedimento viabiliza o conjunto F caso este ainda não seja viável.

Uma vez gerado um conjunto viável F de linhas de trabalho, uma heurística evolutiva no passo (4) executa um procedimento de otimização para tentar encontrar um subconjunto próprio de F com bom valor de custo total das linhas de trabalho. Cada solução destas é memorizada juntamente com o custo correspondente. Estes procedimentos de percorrimento e otimização se repetem por um número de iterações. Vários conjuntos *F* e *S* são criados. Após um número pré-estabelecido de iterações ou se o custo da função objetivo não apresentar melhoras significativas entre duas soluções S consecutivas, o processo iterativo é interrompido. O melhor conjunto S obtido é a solução do problema proposto.

É importante ressaltar que a heurística proposta não garante nem que o melhor conjunto *F* tenha sido obtido nem que a melhor solução *S* tenha sido encontrada do ponto de vista do espaço global de busca do problema, pois não se pode fazer uma busca completa no espaço de solução devido ao tamanho do mesmo. O que se pode afirmar com certeza é que se trata de uma boa solução para o problema.

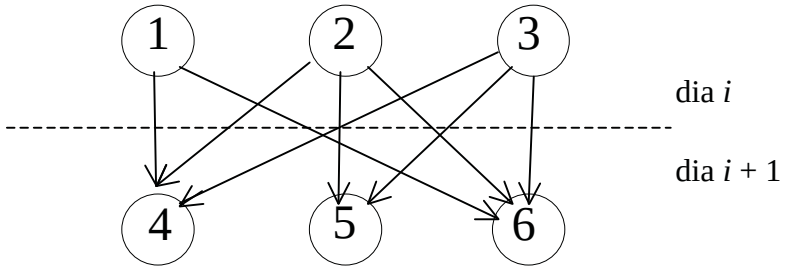
A seguir são fornecidos os detalhes de cada etapa do algoritmo proposto.

3.1 FASE CONSTRUTIVA

3.1.1 Criação do espaço de busca do problema

Os dados brutos da empresa são lidos de planilhas e transferidos para um grafo $G = (V, A)$ que modela as transições entre viagens que um motorista pode fazer na sua escala entre dois dias consecutivos quaisquer, *i* e *i* + 1, dentro do conjunto de dias de programação *D*. O conjunto de vértices, *V*, representa as viagens v_i^k de cada dia, onde *v* é uma viagem diária, o índice *i* representa a data de ocorrência da viagem e o índice *k* identifica uma das viagens do conjunto de viagens diárias da empresa. O conjunto de arcos, *A*, representa as transições entre estas viagens e são criadas respeitando a restrição r_1 de *R*. Um exemplo de *G* pode ser visto na Figura 9.

Figura 9 – Exemplo do grafo G



$G = (V, A)$

onde,

$V = \{1, 2, 3, 4, 5, 6\}$ representa as viagens dos dias *i* e *i* + 1, $\left\{v_i^1, v_i^2, v_i^3, v_{i+1}^1, v_{i+1}^2, v_{i+1}^3\right\}$

$A = \{(1,4), (1,6), (2,4), (2,5), (2,6), (3,4), (3,5), (3,6)\}$ representa as transições possíveis do dia *i* para o dia *i* + 1 obedecendo a restrição r_1 referente ao intervalo mínimo entre viagens consecutivas.

O grafo G é representado por uma matriz de adjacências, M, com um número variável de colunas. O número de linhas da matriz M é igual ao número de vértices de G do dia i e o número máximo de colunas diferente de zero de M é igual ao número de vértices do dia $i + 1$. O elemento $M[k][l]$ é igual ao vértice de destino do *l*-ésimo arco de *A* que possui *k* como vértice de origem. Caso não exista tal arco, $M[k][l] = 0$. Para o grafo da Figura 9, M tem 3 linhas e 3 colunas. Então o elemento $M[1][1]$ é igual ao vértice de destino do primeiro arco de *A* que possui 1 como origem, ou seja, o arco (1,4). Portanto, $M[1][1] = 4$. O elemento $M[1][2]$

é igual ao vértice de destino do segundo arco de *A* que possui 1 como origem, ou seja, o arco (1,6). Portanto, *M*[1][2] = 6. Assim, a matriz M para o grafo da Figura 8 é dada por:

$$M = \begin{bmatrix} 4 & 6 & 0 \\ 4 & 5 & 6 \\ 4 & 5 & 6 \end{bmatrix}$$

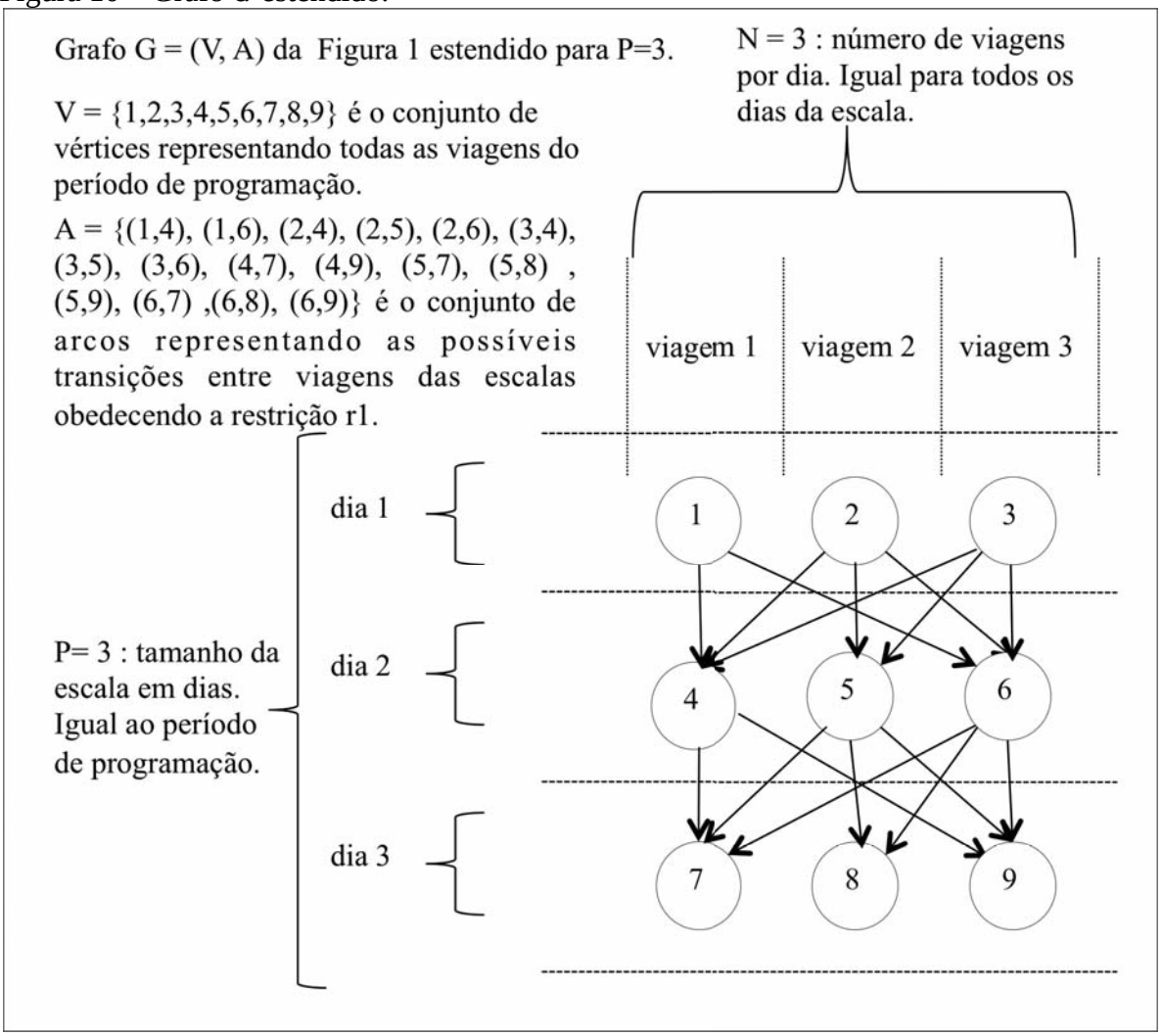
Na prática, ela é uma matriz irregular em relação às colunas, pois se optou por não armazenar os valores quando forem zero. O algoritmo que gera o grafo G possui complexidade polinomial, $O\big(f(n)\big) = K_1.N$, onde K_1 é uma constante (que independe de *N*) e N é o número de viagens diárias da empresa. Nos testes realizados com $N = 137$ viagens diárias da empresa estudada, a construção deste grafo é feita em décimos de segundo.

É possível estender G para todos os dias da escala, isto é, para $i = 2, \dots, P$ transformando-o em um grafo multicamadas a partir do qual, através de um algoritmo de percorrimento em profundidade, será possível obter o conjunto *F* das linhas de trabalho viáveis do problema. Na próxima seção este procedimento é descrito em detalhes.

3.1.2 Construção das linhas de trabalho

O grafo *G* pode ser estendido, considerando uma escala de *P* dias, com *N* viagens por dia. Neste caso, os vértices de *G* são enumerados de 1 até $P \cdot N$. A Figura 10 ilustra um grafo estendido para uma escala de $P = 3$ dias, com $N = 3$ viagens por dia. A construção deste grafo é única para todo o Algoritmo 3, ou seja, uma vez construído os percorrimentos para a geração dos conjuntos *F* são efetuados sempre neste mesmo grafo.

Figura 10 – Grafo $\boldsymbol{G}$ estendido.



Conforme mostra a Figura 10, cada dia é uma camada do grafo. O número de viagens por dia se repete para todos os *P* dias da programação e é igual a *N*. Ao estender o grafo, a restrição r_1 é estendida automaticamente para os demais dias da escala. Embora a estrutura de listas entrelaçadas usada na construção deste grafo estendido seja mais complexa do que outras representações comuns de grafo, a complexidade do algoritmo que a usa e estende G é polinomial e depende diretamente do número de vértices total, ou seja, $O\big(f(n)\big) = K_2 \cdot P \cdot N$, onde K_2 é uma constante.

Um algoritmo de percorrimento em profundidade atravessa G a partir de cada vértice do primeiro dia no passo (3) do Algoritmo 3. Durante o percorrimento, a escolha do próximo vértice é aleatória. No final de cada percorrimento, é obtida uma linha de trabalho candidata ao conjunto F, denominada $L_j : j \in \{1,2,3,\dots, card(F)\}$. O valor de $card(F)$ depende do

número de percorrimentos que se deseja e é parametrizado na solução proposta. O procedimento de percorrimento é implementado através do Algoritmo 4.

Algoritmo 4 – Percorrimento do grafo G a partir do vértice do primeiro dia mais à esquerda (passo (3) do Algoritmo 3)

1. **Para** cada vértice do primeiro dia
   2. Percorrer G a partir do vértice apontado pela próxima transição do vértice atual;
   3. Salvar os vértices visitados numa linha de trabalho candidata;
   4. Verificar se os critérios de classificação são atendidos para a linha de trabalho candidata;
   Se a linha de trabalho não for classificada em algum critério **então**
      5. Descartar a linha de trabalho candidata;
   Senão
      6. Incluir linha de trabalho candidata no conjunto *F*;

O percorrimento inicia no passo (1) do Algoritmo 4, utilizando um procedimento recursivo, passo (2), o qual é descrito no Algoritmo 5. No passo (2) é iniciado novamente o percorrimento no vértice apontado pela próxima transição possível a partir do vértice atual. À medida que os vértices são trocados neste procedimento recursivo, as respectivas viagens que eles representam são armazenadas, passo (3). Ao atingir no passo (2) um vértice do último dia, termina o percorrimento e uma linha de trabalho candidata é obtida. Iniciam-se os procedimentos de classificação desta linha de trabalho. Estes procedimentos visam selecionar linhas de qualidade para o conjunto *F*. Três critérios de classificação são usados no presente trabalho e se referem às propriedades de tamanho, duração e dispersão de uma linha. A verificação se a nova linha atende aos três critérios é feita no passo (4). Basta um dos critérios não ser atendido e a linha candidata não será incluída no conjunto *F*, passo (5). A linha de trabalho só é incluída em F quando os três critérios forem atendidos, passo (6).

O primeiro critério de classificação refere-se ao tamanho da linha em número de viagens diárias, v_i^k, contidas na mesma. Este tamanho deve ser sempre igual a *P*. Um procedimento verifica se este critério foi atendido ou não. Caso exista alguma linha que não atenda a este critério, ela será descartada.

O segundo critério de classificação refere-se à duração da linha de trabalho. Entende-se por duração de uma linha de trabalho o somatório de duração de cada uma das *P* viagens da mesma, segundo a equação (7). Este somatório, Δ_j, não deverá estar aquém de um limite mínimo (LI) e nem além de um limite máximo (LS) pré-estabelecidos. Linhas com Δ_j dentro destes limites serão inseridas no conjunto *F* e com valor fora dos limites serão descartadas.

Sejam,

LI = limite inferior para a duração de uma escala;

LS = limite superior para a duração de uma escala;

Lema 1. Se $LI < \Delta_j < LS$ então L_j é candidata a ser incluída em F, caso contrário será descartada.

O terceiro critério de classificação refere-se à dispersão de uma linha de trabalho. Entende-se por dispersão de uma linha de trabalho L_j a dispersão das durações das viagens dentro da mesma. Ela é identificado pela variável σ_j e calculada da seguinte forma:

$$\sigma_j = \sqrt{\left(\sum_{i=1}^{P}\left(\delta\left(v_i^k\right) - \mu_j\right)^2 \Big/ P\right)}, \quad k \in \{1, \dots, N\}, v_i^k \in L_j; \tag{31}$$

onde:

$\mu_j = \left(\sum_{i=1}^{P} \delta\left(v_i^k\right)\right) \Big/ P$ é a duração média das viagens em uma linha L_j; (32)

σ_j é o desvio padrão das durações das viagens de L_j;

$\sigma_j' = \sigma_j \cdot 0{,}30$ é o desvio padrão reduzido pela aplicação de um fator, obtido por ensaios, de 30% ao desvio padrão original com o objetivo de garantir uma baixa dispersão para as linhas a serem escolhidas para o conjunto *F*.

Para atingir este objetivo, quantitativamente é feita a seguinte operação:

Lema 2 – Se 70% das viagens do percurso L_j tiverem duração dentro do intervalo $\left[\left(\mu_j - \sigma_j'\right), \left(\mu_j + \sigma_j'\right)\right]$ então L_j é candidata a ser inserida em *F*, caso contrário será descartada.

Ao término da aplicação dos critérios, o percorrimento continua a partir do mesmo vértice do primeiro dia da linha anterior variando da próxima vez primeiramente as viagens do final da escala e mais à esquerda do grafo. Lembrando que neste trabalho os novos vértices têm um peso distribuido no momento da construção do grafo por uma função randômica o que confere ao algoritmo de percorrimento em profundidade uma propriedade aleatória na escolha dos próximos vértices que farão parte da nova linha de trabalho que está sendo formada. Conforme mais linhas vão sendo adicionadas a F chega uma hora que o número máximo de percorrimentos por vértice do primeiro dia é atingido. Neste momento, o vértice de origem da nova escala passa a ser o segundo vértice do primeiro dia e, assim sucessivamente, até no final, as linhas criadas serem aquelas que partem do último vértice do primeiro dia.

O critério de seleção de linhas por tamanho, duração e dispersão, faz com que a linha de trabalho gerada obedeça as restrições r_2 e r_3.

No passo (2) do Algoritmo 4 é empregado o Algoritmo 5 descrito a seguir.

Algoritmo 5 – Procedimento recursivo do algoritmo de percorrimento de $\boldsymbol{G}$

1. Encontrar novo vértice a partir do vértice do primeiro dia
    2. Incluir novo vértice na linha de trabalho candidata;
    3. Fazer transição da viagem atual para a próxima;
    4. Se não existir mais transições **então**
        5. Construir linha de trabalho candidata;
    6. Senão
        7. Enquanto existir transição;
            8. Encontrar novo vértice a partir da próxima transição;
            9. Fazer transição da viagem atual para a próxima.

O algoritmo parte de um vértice conhecido que é um vértice do primeiro dia no passo (1) e o inclui na linha de trabalho candidata no passo (2). Em seguida ele procura novo vértice a partir da lista de vértices possíveis, passo (3). O conjunto de vértices possíveis têm pesos associados aos mesmo conforme já discutido. O objetivo é desviar o sentido de percorrimento do algoritmo de busca em profundidade para visitar diferentes sub-regiões do espaço de solução do problema. Um dos vértices de peso mínimo do conjunto de vértices possíveis é sorteado aleatoriamente e neste momento o algoritmo chama a si próprio no passo (8).

Quando chega num último vértice da escala, passo (4) a linha de trabalho candidata é construída no passo (5).

A complexidade deste algoritmo é exponencial em função de N e de P. Seja K_3 o número de operações necessárias para gerar uma linha de trabalho. A complexidade é $O(f(n)) = K_3 \cdot N^P$. O desempenho do algoritmo para gerar o conjunto F é proporcional a N^P o que torna o problema difícil de solucionar para os valores de N e P empregados no mundo real, (N da ordem de 10^2 e P da ordem de 10^2).

Embora o conjunto F obtido obedeça aos três critérios de qualidade, não se pode afirmar, a princípio, que seja um conjunto de linhas de trabalho viável. Para ser viável é preciso que todas as viagens diárias da empresa estejam cobertas por pelo menos uma linha de trabalho. Pode ocorrer de alguma viagem diária não ser coberta por nenhuma linha presente em F e, neste caso, a próxima fase falhará em determinar uma solução para o problema. Por esta razão, o Algoritmo 4 possui um procedimento embutido para garantir que o conjunto F obtido seja viável. Isto não quer dizer que ela não seja o destino de transições a partir de outras viagens do dia $i-1$ e nem que não existam transições partindo dela para outras viagens no dia $i+1$. O problema dela não ter sido incluída em nenhuma linha de trabalho ocorre devido ao caráter randômico da distribuição de pesos fazendo com que no percorrimento em profundidade um vértice fique isolado. Um novo procedimento, denominado procedimento de viabilização, parte do vértice não coberto e percorre o grafo em direção ao primeiro e, em seguida, em direção ao último dia da escala, seleciona aleatoriamente um vértice a cada dia até completar o novo percorrimento. Como o percorrimento parte exatamente do vértice não coberto, garante-se que o mesmo ficará coberto pela nova linha gerada.

Apesar da escolha de um novo vértice ser randômica, o procedimento dá prioridade para os vértices que ocorrem na vizinhança do vértice não coberto nos dias anteriores e posteriores ao dia não coberto. O objetivo com este procedimento é o de garantir um baixo nível de dispersão para a nova linha que está sendo gerada. As questões de duração e tamanho também são tratadas durante este percorrimento.

As novas linhas obtidas são finalmente inseridas em F que passa a ser um conjunto de linhas de boa qualidade e viáveis.

Neste ponto, deve ser calculado o custo de cada linha de trabalho. Ele é diretamente proporcional à diferença de duração da linha com uma duração pré-estabelecida e ao desvio padrão de cada linha e ponderado por um fator constante. Assim, o custo da linha de trabalho $j(c_j)$ é dado por:

$$c_j = (\theta_j \cdot \sigma_j)/VI \quad (33)$$

onde,

θ_j é a diferença de duração de uma linha de trabalho j em relação à duração ideal pré-estabelecida, VI. O valor de θ_j é definido pela equação (6);

σ_j é o desvio padrão de uma linha de trabalho j definido pela equação (31) do presente capítulo;

VI é o fator de ponderação constante. Ele é igual a duração ideal pré-estabelecida de uma linha, pois observou-se que este valor gera coeficientes c_j entre 0 e 100, fáceis de se analisar.

Cada linha de trabalho de F tem agora um custo associado. Deseja-se encontrar um conjunto $S \subseteq F$ cujas linhas de trabalho cubram cada viagem diária apenas uma vez com o mínimo custo possível. Para resolver esta questão, o problema é tratado como um problema de cobertura de conjuntos não unicusto e formulado como um problema de programação inteira binária cuja resolução é feita através de uma heurística evolutiva baseada em Algoritmo Genético.

Como o problema será modelado como um problema de cobertura de conjuntos cujas restrições são do tipo maior ou igual, a restrição r_5 pode não ser respeitada no conjunto S, pois ela estabelece que uma viagem diária deve aparecer apenas em uma linha de trabalho. Em outras palavras, dois motoristas não podem se apresentar no mesmo dia para a mesma viagem. Este problema será resolvido de uma forma heurística quando se determinar o número mínimo de motoristas por dia para serem atribuídos aos padrões de folga da empresa. A abordagem desta questão bem como a solução do problema de alocação de padrões de folga a motoristas serão revisitadas no Capítulo 5.

3.1.3 Tratamento de viagem especial

Neste trabalho foi detectada a presença de viagens com algumas características especiais, o que exigiu um tratamento adicional para incluí-las no modelo. Elas foram designadas como viagens do tipo 1. Caso novas viagens com características especiais venham a surgir no futuro, serão designadas na sequência como do tipo 2, 3 e assim sucessivamente. Uma viagem do tipo 1 pode ser descrita como uma viagem de muito curta duração entre

cidades próximas. O valor máximo desta curta duração, na empresa estudada, variava entre meia-hora e 1 hora.

As viagens do tipo 1 são comuns nas empresas de transporte[3]. Trata-se de viagens que ocorrem em sequência cronológica, ao longo de um dia, tendo várias idas e voltas entre as cidades de origem e destino. São coloquialmente denominadas “viagens circulares”. Na presente proposta, para simplificar o modelo, estas viagens foram agrupadas numa viagem única cuja duração é igual a soma da duração das viagens de ida e volta ao longo do dia e as cidades de origem e destino passam a ser uma única cidade, a cidade de origem.

A viagem agrupada pode ser incluída numa linha de trabalho respeitando as restrições operacionais. Estas viagens têm características bem diferentes das demais, tais como, o tipo de ônibus empregado, a existência de vários pontos de embarque/desembarque dentro de cidades, o perfil do motorista de circular, etc. Procurou-se uma forma de minimizar o aparecimento de linhas de trabalho na solução final contendo estas viagens junto com as demais viagens (consideradas normais do ponto de vista de duração) atribuindo a estas linhas um custo mais alto.

Uma questão oposta surge quando uma linha de trabalho de F tiver apenas viagens circulares. Serão linhas praticamente sem dispersão. Por isto o seu custo é reduzido nestes casos com a intenção de alocá-las na solução final.

Caso a duração real destas viagens circulares ultrapasse o valor ideal pré-definido, a viagem pode ser dividida em duas ou mais, conforme o caso, e novas linhas de trabalho serão criadas a partir destas divisões. Estas operações são executadas minuciosamente pela heurística durante a geração do conjunto F.

Uma nova ideia surgiu nas últimas visitas feitas à empresa piloto que foi a de simplesmente tirar estas linhas do modelo matemático e deixá-las como linhas independentes a serem atribuídas a motoristas específicos. Esta foi a solução adotada no presente estudo.

3.2 FASE DE OTIMIZAÇÃO/BUSCA LOCAL

O problema proposto no passo (4) do Algoritmo 3, pode ser formulado como um problema de cobertura de conjuntos não unicusto como mostra a Formulação 5 seguir.

[3] Segundo informação colhida verbalmente pelo autor na empresa piloto

Formulação 5 – Problema de determinação de S, passo (4) do Algoritmo 3.

$$Min\ z = \sum_{j=1}^{n} c_j x_j$$
$$s.a.$$
$$\sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad (i = 1, \dots, m)$$
$$x_j \in \{0,1\}$$

onde,

i é o índice das linhas da matriz *A*;

j é o índice das colunas da matriz *A*;

$A =$ matriz de cobertura do problema em que cada coluna representa uma linha de trabalho de F e as linhas de *A* representam cada uma das viagens diárias da empresa; $a_{ij} = 1$ se a linha i de *A* for coberta pela coluna j e $a_{ij} = 0$ caso contrário;

$n = card(F)$ é o número de colunas de *A* e é igual ao número de linhas de trabalho de *F*;

$m = N \cdot P$ é igual ao número de linhas de *A* e é igual ao número de vértices do grafo G estendido para o período de programação;

x_j é o vetor solução do problema, $x_j = 1$ se a coluna *j* está na solução final, S, $x_j = 0$ caso contrário.

c_j é o custo da linha de trabalho *j* do conjunto *F* calculado segundo a equação (33).

O novo problema de cobertura de conjuntos é resolvido neste trabalho através de uma heurística evolutiva baseada em Algoritmo Genético e usa muitas ideias propostas por Beasley e Chu (1996) na solução de problemas genéricos de cobertura de conjuntos.

As colunas da matriz A são dispostas em ordem crescente do custo de cada linha de trabalho de F visando facilitar a convergência do Algoritmo Genético.

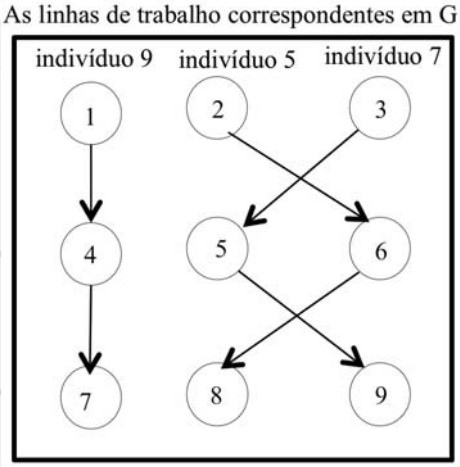
O primeiro passo no projeto de um Algoritmo Genético para um problema específico é determinar a representação de um indivíduo. Esta questão é explicada na próxima seção.

3.2.1 Parâmetros e formação da população inicial

No Algoritmo Genético proposto, um indivíduo ou cromossomo da população inicial está associado à solução $S \subset F$. Seus genes[4] são representados por variáveis inteiras binárias que estão associadas às linhas de trabalho de *F*. Se o *j-ésimo* elemento de ι é igual a 1, a linha de trabalho *j* de *F* pertence a solução *S*. Caso contrário, se o *j-ésimo* elemento de ι for 0 a linha de trabalho *j* de *F* não pertence a *S*. A Figura 11 mostra um indivíduo para o problema anterior, $N = 3$ e $P = 3$, para o qual o algoritmo de percorrimento gerou um conjunto F de 9 linhas de trabalho, as colunas da Figura 11, cada linha com 3 viagens diárias de comprimento (os 1s da linha). Como se pode observar o tamanho do cromossomo (individuo) é igual ao número de linhas de trabalho do conjunto F gerado. Neste exemplo, $card(\iota) = card(F) = 9$.

Figura 11 – Representação binária de um indivíduo $\boldsymbol{\iota}$

		Índice das colunas de *F*								
		1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
Índice das linhas de *F*	*1*	1	1	1	0	0	0	0	0	1
	2	0	0	0	1	1	0	0	1	0
	3	0	0	0	0	0	1	1	0	0
	4	0	0	1	1	0	0	0	0	1
	5	1	1	0	0	0	1	1	1	0
	6	0	0	0	0	1	0	0	0	0
	7	1	0	0	0	0	1	0	1	1
	8	0	0	1	1	1	0	0	0	0
	9	0	1	0	0	0	0	1	0	0
		Indivíduo ι								
		0	0	0	0	1	0	1	0	1



Exemplificando, um possível indivíduo ι de 9 genes ou bits pode ser visto na parte inferior da Figura 11. Cada bit deste indivíduo representa uma coluna de A. Os genes 1, 2, 3, 4, 6 e 8 de ι valem zero indicando que estas colunas de A não fazem parte de S. Já os bits 5, 7 e 9 valem um indicando que as colunas 5,7 e 9 de A fazem parte de S.

[4] Genes e bits neste trabalho são usados como sinônimos devido à representação binária adotada para o cromossomo.

O próximo passo, após a representação do indivíduo, é descobrir uma maneira de medir o grau de adaptação do mesmo na população. População é um agrupamento de indivíduos, que receberá a designação de Π. Neste trabalho, o grau de adaptação de um indivíduo ao ambiente está relacionado à função objetivo do problema. O indivíduo será melhor classificado dentro de sua população quanto melhor for o valor de sua função objetivo, (BEASLEY;CHU, 1996). No nosso problema isto quer dizer que quanto menor o valor da função objetivo melhor o grau de adaptação de um indivíduo. A função de adaptação de um indivíduo ι_s em Π, designada por f_s pode ser equacionada coforme mostra a equação (34).

$$f_s = \sum_{j=1}^{n} c_j \, \iota_{sj} \tag{34}$$

Onde,

$s = 1,2, \ldots card(\Pi)$ são os índices dos indivíduos dentro da população Π;

j é o índice do gene j do indivíduo ι na população Π;

c_j é o custo da linha de trabalho de A;

O próximo passo do projeto do Algoritmo Genético é a determinação do número de indivíduos que a população inicial deve ter. Tal número tem influência no nível de cobertura que se deseja atingir com a aplicação do Algoritmo Genético. Seja $\rho = card(\Pi)$ este número. Nesta proposta, ρ é calculado pela equação (35).

$$\rho = \varphi \cdot n \cdot \mu, \tag{35}$$

onde,

φ é a densidade da matriz de cobertura *A* definida pela equação (36);

μ é um fator redutor ajustado manualmente através de testes;

n é o número de colunas da matriz *A*, $n = card(F)$;

O fator redutor μ foi ajustado manualmente durante os ensaios. Uma faixa foi definida em função de resultados de observação de testes realizados com vários tamanhos e tipos de instâncias do problema. Deve-se usar uma população inicial pequena para pequenas instâncias sem aumentar exageradamente quando se tratar de grandes instâncias. O intuito é o de reduzir o tempo de convergência do algoritmo, mas sem perder a qualidade na solução final.

Uma solução para esta questão é trabalhar com a hipótese de cobrir apenas uma parte do conjunto de viagens diárias e não todas. Assim pode-se trabalhar com um espaço de busca do problema menor do que o real, o que implicará num número menor de indivíduos para se obter a cobertura de todas as viagens. No seu trabalho, Beasley e Chu (1996) reduzem o número de viagens diárias para um número tal que possa ser coberto por apenas 5 colunas de *A*. Com este valor, uma população inicial de apenas $\rho = 100$ indivíduos é suficiente para garantir a cobertura para os problemas estudados. É possível fixar este valor, uma vez que a estrutura e tamanho da matriz de cobertura dos problemas de teste o permitem, pois são conhecidas *a priori* e não variam muito entre si.

No problema em estudo, as instâncias são bem maiores, a densidade da matriz de cobertura é bem menor do que as densidades das matrizes usadas nos conjuntos de dados de teste usados por Beasley e Chu (1996) e a disposição dos uns dentro da matriz é diferente. Uma faixa entre 4 e 200 pode ser "inicialmente" observada através de testes.

No nosso caso, a matriz F não é conhecida *à priori* e dependerá exclusivamente da geração das linhas de trabalho do algoritmo de percorrimento. A cada iteração do Algoritmo 3, passos (2) a (6) um novo conjunto F que é diferente do anterior é gerado. Portanto, ao invés do tamanho da população inicial ser um valor constante, ele é calculado a cada iteração. A densidade φ também varia a cada nova matriz A gerada e sua influência no tamanho da população inicial é considerada fundamental para controlar a convergência do Algoritmo Genético.

A densidade da matriz de cobertura é definida pela razão $(\text{Número de ums em A})/(\text{Total de elementos de A})$, ou seja, conforme a equação (36).

$$\varphi = (\text{Numero de ums em A})/(\text{N} \cdot \text{P} \cdot \text{n}) \tag{36}$$

A quantidade de ums em A é variável e depende do algoritmo de percorrimento. No entanto, uma análise da estrutura desta matriz permite obter a fórmula mais simplificada mostrada na equação (37).

Lema 3 – A densidade da matriz de cobertura para o problema em estudo pode ser obtida por $\varphi = 1/N$ (37)

Prova: $F \subseteq X$. Mas, o número exato de linhas de F só poderá ser determinado após o algoritmo de percorrimento encerrar sua operação, pois existem linhas que serão descartadas pelo mesmo durante o procedimento de classificação. Logo, o número real de linhas de

trabalho em F pode ser aproximando para $card(F) = card(X) \cdot K$, onde K é uma constante. Portanto, levando em conta a equação (5), $card(A) = card(F) = N^P \cdot K$. Mas é sabido que o número de 1s em cada coluna de A é igual a P (lembrar que cada linha de trabalho tem uma e apenas uma viagem por dia e que são ao todo P dias). Portanto o número total de 1s da matriz A é $N^P \cdot K \cdot P$. Por outro lado, o número total de elementos de A é igual ao número de colunas vezes o número de linhas de A, ou seja, $(N^P \cdot K) \cdot (P \cdot N)$. Portanto, a densidade da matriz A é $\varphi = (N^P \cdot K \cdot P)/(N^P \cdot K \cdot N \cdot P) = 1/N$.

Pela equação (37) pode-se concluir que, conforme cresce o número de viagens por dia da empresa, a matriz de cobertura se torna mais esparsa. A maior densidade ocorre quando o total de viagens de um dia for 1. Neste caso $\varphi = 1/1 = 1$ ou 100% de densidade. Se o número cresce para o total de duas viagens por dia, $\varphi = 1/2 = 0{,}5\ ou\ 50\ \%$. Considerando que a empresa piloto tem 137 viagens por dia, a densidade é $\varphi = 1/137 = 0{,}00729\ ou\ 0{,}73\%$. Menor do que as densidades observadas nos conjuntos de teste da biblioteca *OR-Library*, (BEASLEY, 1990).

Estas considerações foram necessárias para mostrar que o formato das matrizes nas quais o Algoritmo Genético está sendo empregado são diferentes entre si e diferentes das matrizes dos casos de teste da *OR-Library*, requerendo uma série de testes e ajustes para equacionar e determinar o tamanho da população inicial e obter bons resultados na resolução do problema em estudo.

Com os parâmetros obtidos, pode-se finalmente gerar a população inicial. O Algoritmo 6 mostra o procedimento de geração da população inicial.

Algoritmo 6 – Geração da população inicial

Sejam

ι o novo indivíduo a ser inserido na população inicial;

ρ o número de indivíduos da população inicial;

w_i vetor com o número de colunas que cobre cada linha i de A ;

1. Inicializar $\Pi = \emptyset$; $\iota = 0$; $w_i = 0$;

2. Para cada linha i de A

3. Sortear aleatoriamente uma coluna j de A que cobre i;

4. Atualizar a coluna j no respectivo gene do indivíduo ι;

5. Incrementar w_i para todo $i: A[i][j] \neq 0$;

6. Para cada coluna j de ι em ordem decrescente de c_j

7. Se $w_i \geq 2$ para todo $i: A[i][j] \neq 0$ eliminar a coluna j de ι;

8. **Fazer** $w_i = w_i - 1$ para todo $i: A[i][j] \neq 0$;

9. Inserir ι resultante em Π;

10. **Repetir** (2) e (6) até que seja atingido o número de elementos ρ desejado para a população inicial;

No passo (2) do Algoritmo 6, para cada linha de A é escolhida uma coluna que a cobre e à posição desta coluna em ι é atribuído o valor 1. Desta forma, ι sempre representará uma solução viável. No passo (6) são eliminadas colunas redundantes de ι a partir das de maior para as de menor custo. Concluídas estas duas operações, o indivíduo gerado é adicionado à população inicial no passo (9). Este processo é repetido, passo (10), até que se tenha obtida uma população com os ρ indivíduos desejados.

Neste trabalho tentou-se otimizar a geração da população inicial deixando permanecer na mesma, sempre que possível, colunas de menor custo. Tal procedimento pode agilizar o processo de convergência. Beasley e Chu (1996) permitem que a população inicial tenha colunas de alto custo o que pode favorecer a diversidade, porém ao custo de poder elevar o tempo de processamento.

Gerada a população inicial, o processo evolutivo é iniciado com os indivíduos sendo submetidos a ações de cruzamento e mutação as quais serão descritas nas próximas seções.

3.2.2 Operação de cruzamento (*crossover*)

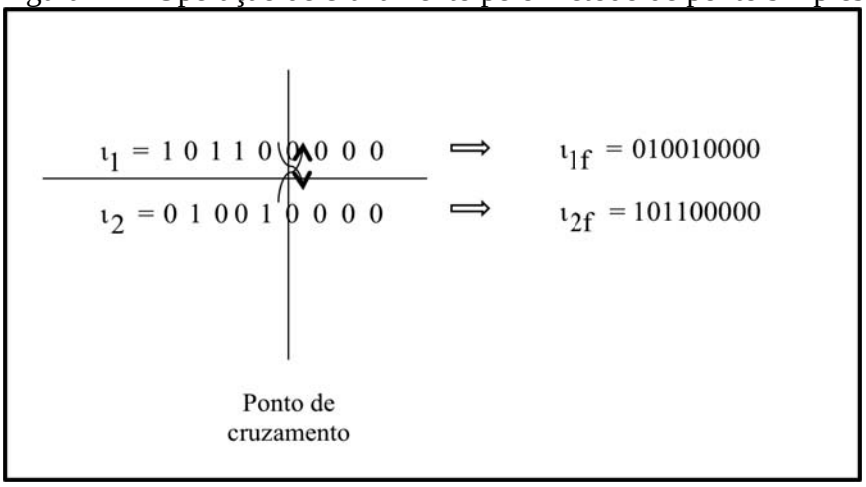
Esta operação consiste em cruzar indivíduos (pais) para gerar novos indivíduos (filhos) a partir da população atual. Na literatura podem-se observar muitas propostas diferentes para se proceder a este cruzamento. Entre elas, a de um ponto simples, ponto duplo, cruzamento uniforme etc.

Neste trabalho usa-se a ideia de Beasley e Chu (1996) de mescla dos genes dos pais para a geração de filhos. Esta mescla consiste em manter no filho os genes comuns entre os pais e os genes diferentes são copiados para o filho na proporção do grau de adaptação de cada pai.

No começo do processamento, o método de cruzamento pode ser uma boa ferramenta para acelerar a convergência. No final, quando o algoritmo estiver próximo da convergência, deve ser tratado com cautela, pois os indivíduos nesta fase requerem mudanças genéticas de pequena extensão, visto que eles já apresentam um alto grau de adaptabilidade e a troca de muitos genes pode levar a solução para outras regiões do espaço de busca nas quais a nova solução poderá ser inviável ou possuir um grau de adaptação muito ruim.

Uma das vantagens do tipo de cruzamento proposto é evitar filhos iguais aos pais. Isto é o que ocorre com a aplicação de métodos tradicionais de cruzamento como os de um ou dois pontos. Como a matriz de cobertura é organizada das colunas de menor para as de maior custo, à medida que o algoritmo converge, as soluções apresentam os bits mais à direita no indivíduo zerados. Com os métodos de um ou dois pontos, por exemplo, se o ponto de cruzamento é escolhido exatamente em algum lugar dentro desta região de zeros, o filho será idêntico aos pais. A Figura 12 ilustra este fato.

Figura 12 – Operação de cruzamento pelo método de ponto simples



Suponha inicialmente que dois indivíduos da população inicial, ι_1 e ι_2, foram selecionados como pais do novo filho a ser gerado. Suponha que o ponto de cruzamento destes dois indivíduos ocorre entre o gene 5 e o gene 6 como ilustrado na Figura 12. O resultado do cruzamento são dois filhos, ι_{1f} e ι_{2f}. Estes filhos ι_{1f} e ι_{2f} se tornam idênticos aos pais como mostrado nesta figura, $\iota_{1f} = \iota_2$ e $\iota_{2f} = \iota_1$. Isto ocorre porque os lados direitos de ι_1e ι_2 se tornam zero depois de várias iterações do processo evolutivo.

Nesta proposta a operação de cruzamento é feita da seguinte forma: primeiramente, os genes iguais nos pais são copiados para o filho. Depois os genes diferentes são copiados na proporção da adaptação de cada pai. Por exemplo, suponha dois pais, ι_1 e ι_2 com fator de adaptação $f_1 = 4$ e $f_2 = 6$, respectivamente. O indivíduo ι_1 possui a melhor adaptação (menor valor da função objetivo) dos dois e igual a 4. Os genes diferentes serão copiados na proporção de 60% ($f_2/(f_1 + f_2)$) dos bits de ι_1 primeiro e os restantes 40% ($f_1/(f_1 + f_2)$) de bits serão copiados do ι_2. Esta proposta, segundo Beasley e Chu (1996), tem a vantagem de gerar uma maior quantidade de indivíduos diferentes, quando os pais têm estrutura similar. O Algoritmo 7 mostra o funcionamento do processo de cruzamento descrito.

Algoritmo 7 – A operação de cruzamento

$Sejam$

ι_1= um indivíduo escolhido para o cruzamento;

ι_2= outro indivíduo escolhido para o cruzamento;

f_1= adaptabilidade do indivíduo 1;

f_2= adaptabilidade do indivíduo 2;

ι_3= indivíduo a ser gerado pelo cruzamento de $\iota_1$com ι_2;

1. Copiar para ι_3 os genes iguais entre ι_1 e ι_2;

2. Para cada gene diferente entre ι_1 e ι_2:

3. Se $f_1 < f_2$

4. Copiar $(f_2/(f_1 + f_2)) * card(\iota_1)$ genes de $\iota_1$para ι_3;

5. Copiar os restantes $(f_1/(f_1 + f_2)) * card(\iota_2)$ genes de ι_2 para ι_3;

6. Senão

7. Copiar $(f_1/(f_1 + f_2)) * card(\iota_2)$ genes de ι_2 para ι_3;

8. Copiar os restantes $f_2/(f_1 + f_2) * card(\iota_1)$ genes de $\iota_1 para\ \iota_3$;

Observando-se o Algoritmo 7, primeiramente são transferidos para o novo filho os genes iguais, passo (1). No passo (2) são copiados primeiramente os genes do pai mais adaptado. Se for ι_1 passo (3), se for ι_2 passo (6).

Para o funcionamento do Algoritmo 7, dois indivíduos, ι_1 e ι_2, têm que ser escolhidos da população Π para o cruzamento. Uma forma de proceder esta escolha é através do "torneio seletivo binário". Trata-se de um procedimento no qual são escolhidos aleatoriamente 4 indivíduos e divididos em grupos de 2 indivíduos cada. De cada grupo escolhe-se o melhor indivíduo, segundo o grau de adaptação, para ser um dos pais da nova geração.

Nesta proposta, os 4 indivíduos ao invés de separados em 2 grupos, são ordenados por valor crescente da adaptabilidade e os dois primeiros da fila são os escolhidos. Tal procedimento seleciona sempre os dois melhores pais ao invés de permitir a escolha de pais menos adaptados, que é o caso do torneio binário sugerido por Beasley e Chu (1996).

À medida que os novos filhos são gerados pela operação de cruzamento eles são encaminhados à operação de mutação. O procedimento de mutação empregado neste trabalho é descrito em detalhes na próxima seção.

3.2.3 Operação de mutação

A operação de mutação implica em trocar bits (de um para zero e de zero para um) de um indivíduo. Os bits podem ser escolhidos mediante um critério ou a escolha pode ser randômica. Quantos bits serão trocados, onde e quando também deverão ser escolhidos, e os valores são fatores que influenciarão o resultado da heurística.

A mutação auxilia a fugir de ótimos locais, os quais podem ocorrer no final da convergência. Beasley e Chu (1996) sugerem utilizar uma taxa de mutação de bits, η, variável em função do tempo de convergência do algoritmo. A seguinte função exponencial é sugerida neste caso:

$$\eta = f(t) = \left\lceil \frac{m_f}{1+e^{\left(-4m_g(t-m_c)/m_f\right)}} \right\rceil \tag{38}$$

Onde,

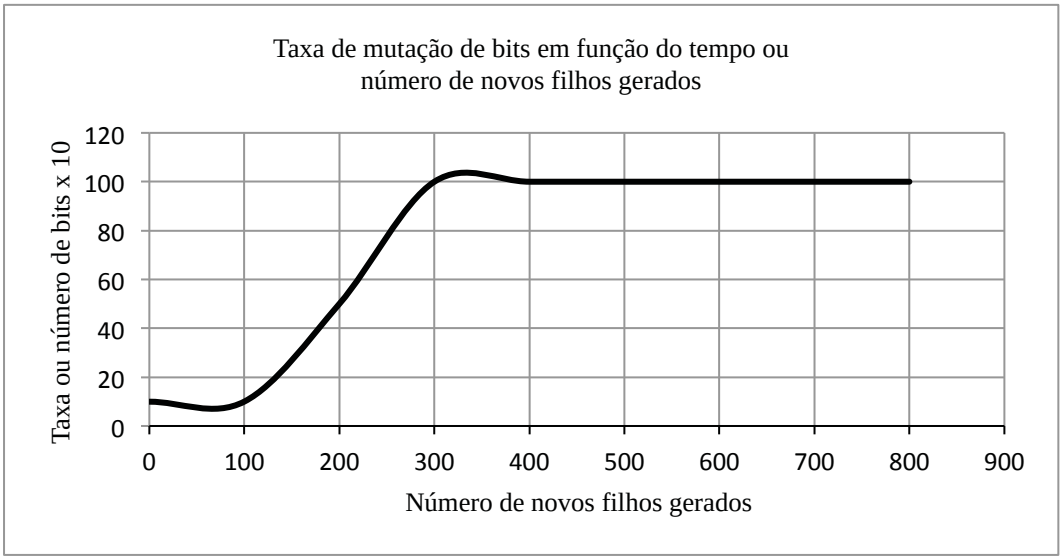
m_f = valor final da taxa de mutação de bits, também denominada taxa de mutação estável. Sugere-se usar um valor entre 5 e 100 para evitar que o Algoritmo Genético fique preso em um ótimo local;

m_c = o número de filhos t gerados quando a taxa de mutação atingir metade da taxa de mutação estável;

m_g = o gradiente da função de mutação quando $t = m_c$;

A Figura 13 ilustra esta função para um caso de teste com valores de $m_f, m_c e\ m_g$ obtidos da literatura.

Figura 13 – Variação da taxa de mutação para $\boldsymbol{m_f = 10, m_c = 200, m_g = 0,5}$.



Na Figura 13, a taxa de mutação que aparece no eixo vertical já aparece multiplicada por 10 para tornar o gráfico mais fácil de observar. No início, até 90 filhos gerados, a taxa de mutação é da ordem de 1 bit apenas. Quando o número de filhos gerados está entre 90 e 130 a taxa de mutação de bits começa a aumentar e inicia o processo de convergência. A partir do 300º filho a taxa se estabiliza em torno de 10 bits e é quando o algoritmo converge chegando no ótimo quando conhecido. Este valor de 10 é o valor atribuído a $m_f$no início e, a partir deste momento, não varia mais até o final.

No presente trabalho, os valores de $m_c, m_g\ e\ m_f$ são calculados a cada novo conjunto F produzido na fase construtiva. Foi necessário efetuar um ajuste manual devido às características de algumas instâncias nas quais a aplicação direta dos parâmetros ocasionava a necessidade de longos tempos de convergência. Foi observado que esta demora ocorreu quando as instâncias eram pequenas e havia a presença de indivíduos duplicados do novo filho gerado na população atual. Quando isto ocorre o filho é descartado. Este problema foi resolvido reduzindo o número máximo de gerações a serem criadas para estas instâncias.

Uma vez determinados quantos bits devem ser mudados e quando, o próximo passo é determinar em que lugar do cromossomo estes bits serão alterados. Para o problema estudado, a melhor região é a dos bits mais à esquerda, por se referirem às colunas de menor custo. Não há sentido em mudar bits nas colunas de alto custo durante a convergência, porque isto criará

um filho menos adaptado, o que acarretará um tempo maior de convergência. A escolha do lugar onde os bits serão trocados é aleatória, mas a heurística garante que será usada a região mais à esquerda do indivíduo.

A aplicação das operações de cruzamento e de mutação em sequencia podem tornar o novo indivíduo gerado, ι_s, inviável. Sempre que isto ocorrer o novo filho ι_s deverá ser reconduzido para a região de viabilidade. Tal procedimento é realizado por uma operação denominada operação de viabilização, a qual é inserida no algoritmo após concluir a operação de mutação. Esta nova operação está explicada na próxima seção.

3.2.4 Operação de viabilização

As operações de cruzamento e de mutação podem ter levado o filho ι para uma região de inviabilidade. É necessário fazer com que este filho ι retorne a condição de viabilidade do problema. A operação de viabilização é a operação responsável pela reinserção de um filho, ι, no espaço de viabilidade do problema.

Neste trabalho, uma heurística do tipo “gulosa” é empregada para esta finalidade. Ela é baseada no operador de viabilização do trabalho de Beasley e Chu (1996). A escolha da nova linha de trabalho $j \in F$ dentre um conjunto de linhas possíveis que cobrem a mesma viagem diária é feita analisando o resultado da divisão do custo da linha, c_j, pelo número de viagens diárias não cobertas que a linha j cobre. A linha que apresentar o menor valor para esta razão será a escolhida para entrar na solução. Como se observa não é usado apenas o custo da linha para decidir se a linha deverá entrar ou não na solução. A razão de se usar a razão entre o custo e o número de linhas descobertas que a linha candidata cobre tem a finalidade de acelerar o processo de convergência e não permitir que o novo filho que irá viabilizar a solução, ι, tenha uma adaptação muito baixa. O Algoritmo 8 mostra como funciona a operação de cruzamento como um todo.

Algoritmo 8 – A operação de viabilização

Sejam,

U = um conjunto contendo as viagens diárias não cobertas em ι_{sj};

j^* = a linha de trabalho j de F que cobre uma viagem de U com o menor valor de c_j /(número de viagens não cobertas que serão cobertas por j);

1. Para cada elemento i de U em ordem crescente de i;

2. Adicione j^* referente a esta linha a ι_{sj};

3. Faça $w_i = w_i + 1$ para todo $i: A[i][j^*] \neq 0$;

4. Faça $U = U - \{i\}$;

5.Para todo $j \in \iota_{sj}: j \neq 0$ em ordem decrescente de j;

6. Se $w_i \geq 2$ para todo $i: A[i][j] \neq 0$ então

7. Faça $w_i = w_i - 1$;

8. Faça $\iota_{sj} = \iota_{sj} - \{j\}$;

No Algoritmo 8 passo (1), para cada elemento de U, é encontrada a melhor linha de trabalho de F para cobri-la usando a razão explicada anteriormente. Esta nova linha de trabalho é adicionada a ι_{sj}, e as variáveis w_i e U são atualizadas. No entanto, esta operação de viabilização pode tornar algumas linhas de trabalho de ι_{sj} redundantes, isto é, sem esta linha a solução continua viável. É necessário eliminá-las antes de prosseguir. Isto é feito no passo (5). Observe que são eliminadas as linhas de trabalho redundantes de maior custo em primeiro lugar.

Uma vez viabilizado, o novo filho será inserido na população atual. Um método especial de mudança na população foi desenvolvido e é apresentado na próxima seção, designado por método de substituição da população atual.

3.2.5 Método de substituição da população atual

O método de substituição de uma população pela nova geração de filhos criada a partir das operações de cruzamento/mutação/viabilização se baseia na proposta de Beasley e Chu (1996). O procedimento usa uma forma de substituição denominada substituição estável (*steady-state replacement*), na qual o novo indivíduo deve substituir um outro da população

atual, cujo grau de adaptação seja imediatamente pior do que o seu, ao invés de substituir a população inteira, como em outros métodos.

Este método tem a vantagem de manter os bons indivíduos na população atual de forma a poderem ser novamente empregados para seleção e reprodução, melhorando o desempenho global do Algoritmo Genético, embora a comprovação deste fato ainda requeira mais testes práticos.

Antes de inserir o novo filho, se faz se necessário verificar se o mesmo não existe na população atual. Quando existe, ocorre “filho duplicado”. Neste caso, o novo filho gerado será descartado e o procedimento para a geração de novo filho se iniciará.

Se o novo filho não for duplicado ele substituirá um cuja adaptação seja imediatamente superior ou igual a sua.

Encerrado o processo de substituição da população pelo novo filho, deve-se analisar se o critério de parada do algoritmo foi ou não atingido. A próxima seção descreve este procedimento.

3.2.6 Finalização da fase de otimização – Critério de parada

Para Beasley e Chu (1996), o procedimento de parada ocorre quando o número de filhos gerados atinge um valor máximo pré-definido. Neste momento, a heurística procura o indivíduo ι_s^* na população Π atual com o melhor valor de função de adaptação f_s^*. Este indivíduo é o escolhido como solução do problema. Uma variável controla o número de filhos gerados e é incrementada a cada iteração, mesmo que o novo filho seja descartado por ocorrência de duplicidade. É interessante observar que este valor máximo na proposta de Beasley e Chu (1996) não varia em função da variação das instâncias de teste empregadas. Ele é de 100.000 para todas as instâncias de problemas empregadas nos testes.

No nosso problema, ao invés disso, o critério de parada é função do número máximo de filhos e este número varia em função do conteúdo e formato de cada conjunto F. A variável τ é designada para representar este número máximo de filhos. O Algoritmo 3 divide, conforme já foi explicado, o espaço de busca em espaços menores, obtendo a cada novo espaço gerado um novo e diferente conjunto F. Para cada conjunto F um novo valor de τ deverá ser encontrado através de testes.

Entre os parâmetros que influenciam o valor de τ, destacam-se a densidade da matriz de cobertura, sua estrutura, a quantidade de linhas de trabalho por vértice do primeiro dia e o

número de linhas e colunas da mesma. Além destes, observou-se nos testes realizados, que a forma de calcular a taxa de mutação, η, a qual foi definida pela equação (38) para um mesmo conjunto F, deve sofrer variações para acompanhar a variação de τ. Quando esta taxa varia muito lentamente e se existem muitos filhos duplicados na população atual e o problema é de pequeno porte (por exemplo, número de linhas da ordem de 10^2 linhas e número de colunas da ordem de 10^3 colunas), o algoritmo demora a convergir. Fazendo a taxa variar mais rápido no início, a convergência é acelerada.

No entanto, ainda é cedo para estabelecer uma relação entre estes parâmetros e τ devendo ser motivo de mais estudos no futuro. A ideia adotada para resolver a questão neste trabalho foi a de observar a variação do tempo de convergência e, ao mesmo tempo, comparar o resultado com o resultado obtido pela aplicação de um método exato de solução. O método exato utilizado para comparação de resultados foram os métodos do otimizador Xpress ©, versão 7.4.

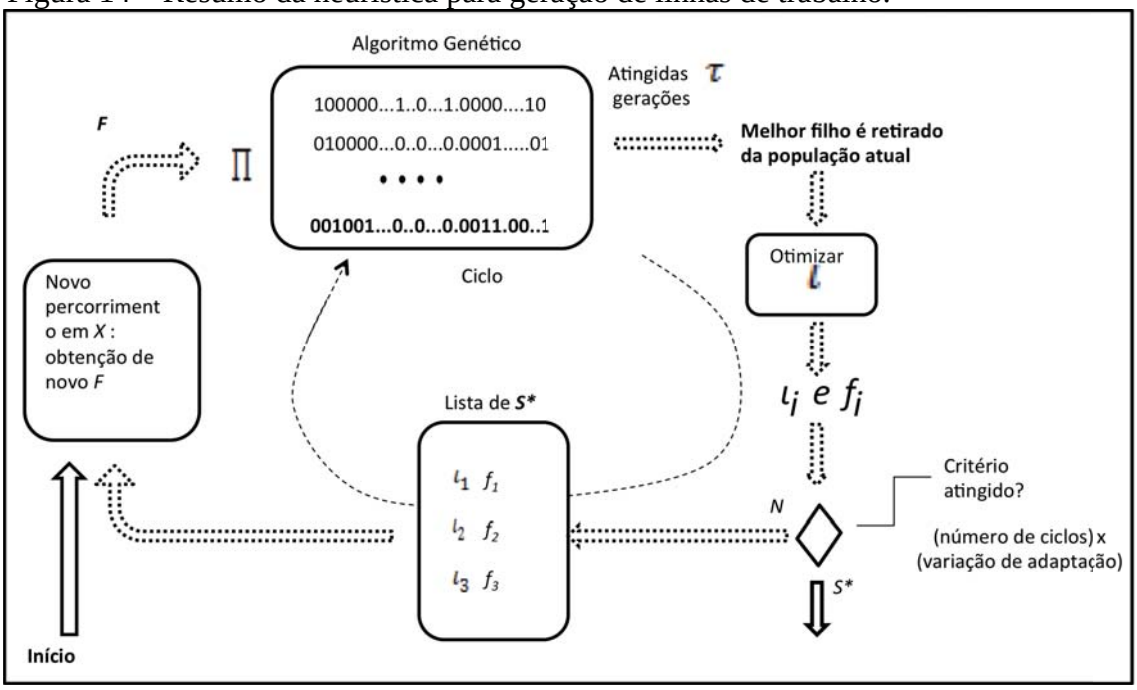
Nestas comparações foi observado que para instâncias pequenas do problema, por exemplo, até P=20 dias e N=6 viagens por dia, a convergência é rápida, da ordem de minutos; Porém, ainda é lenta se comparada ao tempo de resolução do método exato, da ordem de segundos. No entanto, o ótimo encontrado no método exato é sempre atingido depois de algumas tentativas com o método heurístico.

3.2.7 Finalização do algoritmo proposto

A Figura 14 ilustra esquematicamente o funcionamento da heurística até a geração do conjunto S^* das melhores linhas de trabalho. A parte esquerda da Figura 14 mostra a criação de X e a geração de F. Nela se pode observar o conjunto π contendo os indivíduos da população. A figura mostra que após ter passado um tempo evolutivo medido por τ gerações (filhos novos gerados), extrai-se da população atual π o melhor filho ι, isto é, aquele que apresenta o melhor valor da função de adaptação, f. Este filho é submetido a um processo de otimização local para eliminar redundâncias nas viagens diárias da empresa quando houver. Este procedimento reduz o número de linhas do conjunto S referente ao filho ι procurando sempre que possível atingir $card(S) = N$. Cada filho ι destes é armazenado numa lista de candidatas denominada Lista de S^* conforme mostra a figura. Essas iterações ou ciclos continuam até que um número pré-definido de iterações (atualmente 10) seja atingido ou que, durante certo número de iterações, o valor de f não se altere significativamente. O percentual

de alteração da função objetivo para encerrar a heurística ainda deve ser fixado através de mais testes.

Figura 14 – Resumo da heurística para geração de linhas de trabalho.



Como resultado tem-se o conjunto final de linhas de trabalho S^* com um número de linhas de trabalho igual ao número de viagens diárias da empresa, *N*, respeitando a restrição $r_{5,}$ mas em alguns casos sem respeitar a restrição r_4 o que exigirá no

3.3 PADRÕES DE FOLGA

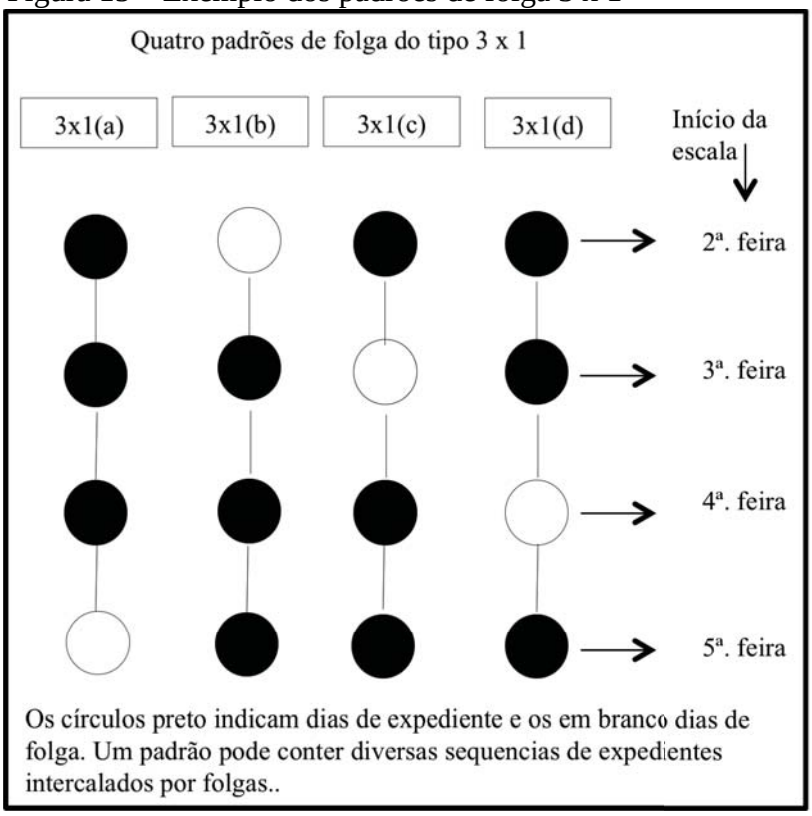
O problema dos padrões de folga consiste em achar um número mínimo de motoristas para cada um dos diferentes tipos de padrões com os quais a empresa trabalha de modo a que a soma desses motoristas por dia se iguale a demanda de viagens diárias da empresa que é igual a *N*.

Padrões de folga ou turnos neste trabalho são escalas contendo apenas dias de expediente e dias de folga que o motorista tem que cumprir na prática quando estiver alocado em sua linha de trabalho. Em um mesmo padrão de folga a razão entre o número de dias de expediente e o número de dias de folga é sempre constante. Os padrões de folga são, em geral, designados por dois números, por exemplo, 4x1. O primeiro número indica o número de dias de expediente e o segundo a quantidade de dias de folga.

Desta forma, as restrições r_6 e r_7, as quais se referem ao número máximo de dias consecutivos de trabalho entre folgas e ao número máximo de folgas no período de programação, são questões que poderão ser resolvidas quando da definição dos turnos com os quais a empresa irá trabalhar. Este problema não é de difícil solução.

Já o problema dos padrões de folga definido no início desta seção é um problema combinatório. Ele pode ser visualizado na Figura 15.

Figura 15 – Exemplo dos padrões de folga 3 x 1



Existem folgas nos diferentes padrões conforme mostram os círculos em branco da Figura 15. Se elas coincidirem num mesmo dia é necessário fazer combinações com outros expedientes no mesmo dia para ter certeza de que neste dia comparecerão *N* motoristas para cumprir as viagens diárias da empresa. Mas o problema não é de grande porte, uma vez que as empresas usam um pequeno conjunto destes padrões e o período de programação, em geral, é de no máximo 360 dias. A matriz para a formulação não será grande.

Propõe-se modelar este problema como um problema de particionamento de conjuntos unicusto, no qual as variáveis são as quantidades de motoristas de cada turno, as restrições a demanda de viagens diárias, *N*, de cada dia e que é constante para toda a programação. Os coeficientes da função objetivo são unitários, pois não há exigência de especialização de

acordo com a escala de serviço, como ocorre em outras áreas de aplicação, por exemplo, a área de enfermagem. Nem diferenciação dos motoristas quanto a habilidade, experiência ou salário para serem levados em consideração. Para a formulação do problema sugere-se usar a Formulação 6:

Formulação 6 – Formulação do problema dos turnos como um problema de particionamento de conjuntos unicusto

$$\min z = \sum_{j=1}^{N} x_j$$

$$s.a.$$

$$\sum_{j=1}^{U} a_{ij} x_j = N \; para \; i = 1, \dots, P$$

$$x_j \geq 0$$

$$x_j \in \mathrm{Z}^*$$

Onde,

x_j é a variável do problema e representa a quantidade de padrões de folga de cada tipo presentes na solução;

U é o número de tipos de padrões de folga diferentes no conjunto de padrões com que a empresa trabalha. Por exemplo, U = 18 quer dizer que a empresa trabalha com 18 tipos diferentes de padrões, os quais poderão ser quatro turnos do tipo 4x1, três 5x2, seis 4x2 e cinco 3x1. Estes tipos são diferenciados entre si em função do dia de início dos mesmos. Por exemplo, o padrão 4x1(a) inicia numa segunda-feira, o 4x1(b) na terça-feira, o 4x1(c) na quarta-feira e o último 4x1(d) na quinta-feira dando um total de 4 diferentes tipos de padrões 4x1, identificados pelas letras a,b,c e d. Na elaboração de um padrão pode-se misturar tipos diferentes. Por exemplo, para os primeiros 10 dias da escala usa-se o tipo 4x1. Os próximos 10 dias usam o padrão 5x2 e os últimos 10 dias de uma escala de 30 dias usa o tipo 4x2.

j é o índice dos turnos, ou seja, $j = 1, 2, 3, \dots, U$.

a_{ij} é matriz binária que representa os dias de expediente e folga do turno. $a_{ij} = 0$ quando for dia de folga no dia i para o turno j e $a_{ij} = 1$ quando o dia i for de expediente no turno j.

Sugere-se resolver este modelo através de métodos exatos utilizando técnicas de programação linear inteira. O Anexo A apresenta uma maneira de como este problema foi

formulado para solução no otimizador comercial Xpress © usando a linguagem de modelagem Mosel ©. O problema modelado se refere ao problema do caso de teste P10N20.

A quantidade de cada padrão de folga na solução pode variar. Inclusive, podem haver padrões que não participarão da solução final ficando com zero. O que é importante entender é que cada padrão destes representa uma quantidade de dias de expediente e de dias de folga disponíveis em cada dia da programação da escala. A soma dos dias de expediente levando em conta todos os padrões da solução, dia a dia, tem que ser igual ao número de viagens diárias da empresa. Tal fato é garantido, pois está implicitamente incluído na montagem das restrições do problema, lembrando que é um problema de particionamento de conjuntos.

A Tabela 2 ilustra um exemplo de solução de um problema real discutido na seção 4.4 no qual há 13 tipos de padrão de folga no total e destes apenas 6 foram escolhidos na solução obtida pelo otimizador. Na ultima linha pode-se ver a solução. Deve haver 4 padrões do tipo 5x1 e zero padrões de quaquer outro tipo para manter um número de expedientes igual a 20 em todos os dias da escala.

Tabela 2 – Exemplo de resultado do problema de padrões de folga

Dias	5x1(a)	5x1(b)	5x1(c)	5x1(d)	5x1(e)	5x1(f)	Expedientes por dia
1	1	0	1	1	1	1	20
2	1	1	0	1	1	1	20
3	1	1	1	0	1	1	20
4	1	1	1	1	0	1	20
5	1	1	1	1	1	0	20
6	0	1	1	1	1	1	20
7	1	0	1	1	1	1	20
8	1	1	0	1	1	1	20
9	1	1	1	0	1	1	20
10	1	1	1	1	0	1	20
Z	$x_1=4$	$x_2=4$	$x_3=4$	$x_4=4$	$x_5=4$	$x_6=4$	24

O valor ótimo mostrado na última linha desta tabela é de 24 expedientes. Este é o número mínimo de motoristas que a empresa deverá ter como recurso para operar as linhas de trabalho sendo que pela sequencia de expedientes e folgas dos padrões escolhidos na solução apenas 20 desses 24 estarão em expediente e os outros 4 estarão num dia de folga de suas escalas.

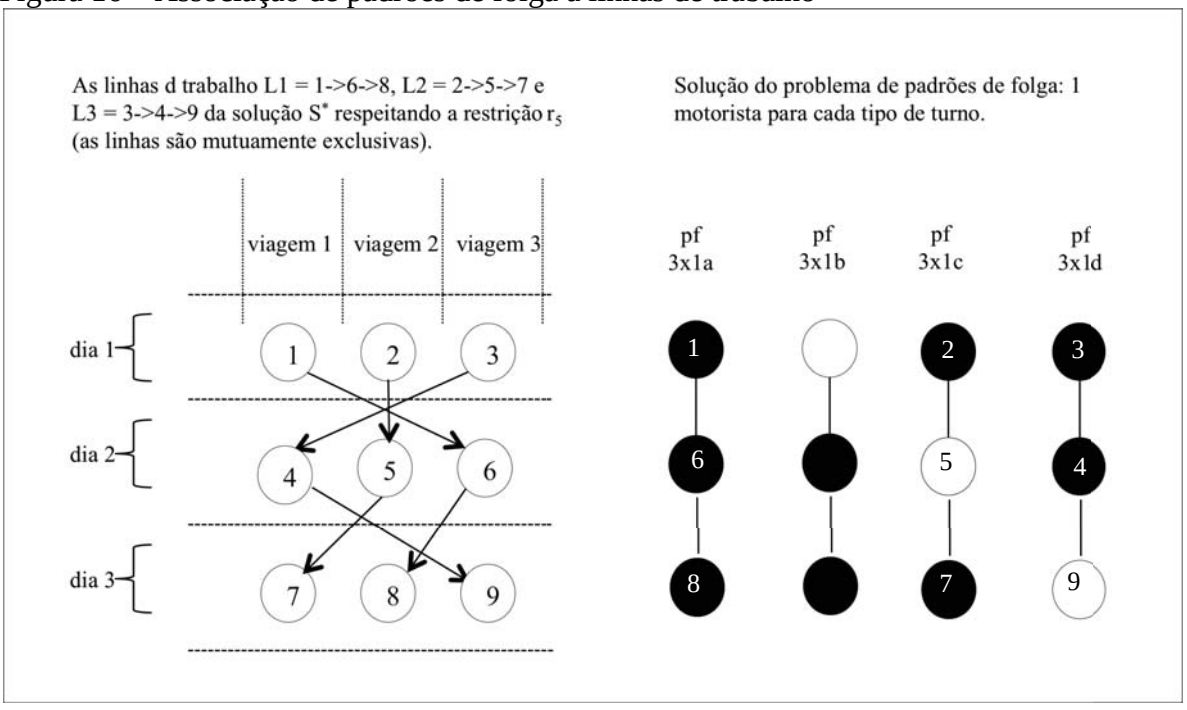
Observe que a parte da matriz de particionamento da Tabela 2 apresenta os padrões referentes aos tipos 5x1, pois para os demais tipos o valor da variável x_j é zero. Na parte

mostrada ocorre apenas um zero em cada linha e x_1 = 4 padrões do tipo 5x1(a), x_2 = 4 padrões do tipo 5x1(b) e assim sucessivamente até o padrão 5x1(f). A soma de expedientes dará sempre 20, pois 4 motoristas sempre estarão no seu dia de folga.

Do que foi dito até aqui, deve haver um *pool* de *N* motoristas diariamente a disposição na empresa. Estes padrões serão distribuídos entre as linhas de trabalho encontradas na solução do problema de construção das linhas. Devido à operação de otimização no passo(5) do Algoritmo 3, o número de linhas de trabalho da solução também é igual a N e as restrições r_4 e r_5 foram obedecidas. Portanto pode-se associar cada linha a cada um dos padrões de folga encontrados na solução do problema dos padrões conforme mostra a Figura 16. Esta situação é obtida naturalmente no passo (6) da heurística, mas depende do tipo do conjunto *F* obtido a cada percorrimento. Nela o número de motoristas extras, isto é aqueles motoristas que devem ser contratados para cumprir viagens diárias não cobertas é zero.

Uma proposta para associação das linhas de *S* com os padrões de folga é fazer uma associação de 1 x 1, ou seja, uma linha de trabalho para um padrão de folga. Evidentemente, isto só será possível se as linhas formarem um conjunto mutuamente exclusivo de viagens e o problema dos padrões tiver uma solução ótima. A Figura 16 ilustra este caso para P = 3 dias de programação e *N = 3* viagens diárias. Do lado esquerdo é mostrado o conjunto de linhas mutuamente exclusivas de S^* e do lado direito uma possível solução do problema de padrões com quatro tipos de turnos e um motorista em cada turno.

Figura 16 – Associação de padrões de folga a linhas de trabalho



Observe, nesta figura, que pela solução do problema de padrões, sempre haverá três motoristas disponíveis a cada dia. Supõe-se que seja possível a qualquer motorista cobrir qualquer viagem em qualquer dia, independente da cidade em que o motorista estiver quando for designado para uma nova viagem. Suponha que um motorista ‘a’ fará o turno ‘a’, um motorista ‘b’ o turno ‘b’, um motorista ‘c’ o turno ‘c’ e um motorista ‘d’ o turno ‘d’. Na associação ‘escalas x turnos’ suponha que a escala *L1* (1-6-8) seja designada ao turno ‘a’, a *L2*(2-5-7) ao ‘c’ e a *L3*(3-4-9) ao ‘d’. No segundo dia, devido à folga do turno ‘c’ a escala *L2* ficará desguarnecida, pois não há motorista para a viagem 5, v_2^2. Mas há um motorista de serviço neste dia, que está cumprindo o turno ‘b’ e que estava de folga no dia anterior. Para que ele assuma a viagem 5 é necessário que ele esteja na cidade de origem da viagem 5, $co(v_2^2)$. Caso contrário a empresa terá gastos de translado. Esta é outra questão que pode ser tratada nas futuras pesquisas.

Quando uma das condições (linhas mutuamente exclusivas e solução ótima do problema de padrões de folga) ou eventualmente as duas não ocorrerem simultaneamente, teremos um novo problema a ser resolvido, pois a restrição r_5 não será atendida. Para tratar este problema são apresentadas algumas soluções.

Para a discussão que se segue é importante ressaltar que o conjunto *F* obtido, bem como, o conjunto *S* obtido a partir dele, possuem linhas de trabalho que cobrem todas as viagens da empresa várias vezes.

Vejamos uma proposta para tornar as linhas de *S* mutuamente exclusivas.

A ideia é usar um otimizador no lugar da heurística evolutiva para tentar produzir o conjunto *S** a partir de *F*. Para tal, o problema de cobertura original deve ser convertido para um problema de particionamento de conjuntos, tendo como matriz de particionamento o conjunto de linhas *F*. Porém a Tabela 10 (última linha) mostra que o otimizador não consegue obter uma solução ótima a partir desta formulação no problema P10N20.

Uma alternativa que pode melhorar o conjunto das linhas de *F* para o otimizador leva em consideração que o algoritmo de busca em profundidade possibilita a ocorrência de vértices com múltiplas visitas e que é possível mudar tal comportamento através da mudança da curva de distribuição de probabilidade responsável por atribuir pesos aos vértices de *G* durante o percorrimento. Na versão atual a distribuição de pesos é uniforme. Uma curva de distribuição melhor projetada e ajustada dinamicamente durante o percorrimento pode ajudar a evitar esta concentração e assim gerar um conjunto S^* sem vértices redundantes.

Outra proposta leva em consideração, (observando os testes realizados até o momento), que o conjunto *S** é pequeno (na empresa em estudo, é da ordem de 10^2). A relação observada em testes, entre os tamanhos de *S* e S^* foi de 2:1 até 3:1. O que indica uma quantidade de linhas de *S* da ordem de 10^3. Portanto, pode-se supor que um algoritmo simples e eficiente de 'exclusão/inclusão' de linhas em *S* foi projetado com a finalidade de determinar o maior número de linhas de *S** que resulte no menor número de vértices não cobertos na solução final. O Algoritmo 9 foi usado para esta finalidade.

Algoritmo 9 – Procedimento exaustivo de eliminação de viagens diárias redundantes do conjunto *S** original.

1. Para as viagens diárias do primeiro dia

2. Incluir a linha de menor custo, a partir de cada viagem diária, em S^-;

3. Atualizar os vértices respectivos de G com peso máximo;

4. Se ocorrer viagem diária coincidente durante o processo de inserção em S^- **então**

5. Descartar a nova linha e procurar a próxima de custo imediatamente superior;

6. Repetir o procedimento 1-5 alternando as linhas do primeiro dia com as demais do segundo dia em diante **até**

Serem testadas todas as combinações possíveis para minimizar o número de viagens diárias não cobertas e aumentar o número de elementos do conjunto S^*;

7. Atribuir as viagens diárias não cobertas obtidas a motoristas cobre-turnos.

Este algoritmo começaria por copiar a linha de menor custo de *S*, a partir do primeiro vértice do primeiro dia, para um novo conjunto S^-. Por exemplo, se na solução *S* existirem 50 linhas que partem do primeiro vértice, será escolhida apenas aquela de menor custo dentre as cinquenta existentes. Repete-se o procedimento para o segundo vértice e assim por diante até o último vértice visando obter exatamente *N* linhas uma de cada viagem diária do primeiro dia. Se neste processo uma nova escala possuir um vértice em comum com um vértice de uma linha já inserida em S^-, ela é descartada e a próxima linha de custo mais baixo, partindo do próximo vértice é testada. Para cada linha inserida em S^-, os respectivos vértices de *G* deverão ter seus pesos aumentados para o máximo. O objetivo é excluir estes vértices de futuros percorrimentos.

No final, um conjunto de linhas S^- é obtido. Sua cardinalidade pode ser menor do que *N*, porém pode ocorrer que alguns vértices tenham ficado descobertos. O pequeno conjunto de vértices não cobertos resultantes deverá ser designado para os motoristas cobre turnos, os quais não fazem parte do corpo permanente de funcionários. Nos testes realizados até

momento, o número de viagens a descoberto com esta alternativa é bem baixo sendo da ordem de 5% de N. Tal valor é inferior ao observado na empresa em estudo na qual o número destes extras está na faixa de 10 a 15% de N.

Conforme se pode observar este novo problema é de difícil solução e requer mais estudos até se construir evidências suficientes para formular uma solução definitiva.

Desta forma encerra-se a heurística proposta. No próximo capítulo serão mostrados os resultados dos testes computacionais realizados até o momento.

4 TESTES COMPUTACIONAIS

Vários testes foram realizados tanto com dados da literatura quanto com dados obtidos da empresa. Alguns testes foram realizados de forma independente. Primeiramente foi verificada a efetividade do algoritmo de percorrimento bem como sua precisão e robustez. Depois de ajustes necessários, foi verificada a eficiência da nova heurística evolutiva a qual foi baseada no algoritmo genético proposto por Beasley e Chu (1996). Devidas várias alterações realizadas no mesmo para se adequar às características do problema real em estudo, foi necessário, antes de mais nada, testar o novo algoritmo sobre a mesma base de dados usada por Beasley e Chu (1996). Foi observada principalmente a questão da convergência do Algoritmo Genético modificado.

Uma vez testados e ajustados os dois procedimentos individualmente, o próximo passo foi testar os procedimentos integrados e com dados reais da empresa.

Todos os testes realizados até o momento foram realizados em um processador Pentium Core 2 Duo, 1.4 Ghz, 32 bits, 2 Gb de memória RAM. A linguagem de programação empregada é a linguagem C++. Quanto ao otimizador empregado nos testes trata-se do otimizador Xpress© versão 7.2 da FICO©.

A seguir são descritos os resultados obtidos.

4.1 TESTES NO PROCEDIMENTO CONSTRUTIVO

O procedimento da fase construtiva é responsável por montar vários conjuntos, *F*, de linhas de trabalho (LT), todas viáveis e qualificadas, para depois serem submetidos à fase de otimização.

O objetivo dos testes neste momento foi o de verificar se o conjunto F gerado atende aos seguintes requisitos:

i) Diversificação;
ii) Índice de 100% de cobertura das viagens diárias;
iii) Não redundância;
iv) Qualidade da LT segundo os critérios de duração e dispersão.

Foram obtidas informações sobre estes quatro requisitos. A Tabela 3 mostra algumas características dos testes realizados. O objetivo é o de verificar a eficiência e eficácia do procedimento usado.

Tabela 3 – Testes do procedimento construtivo

N	P	Número de linhas de trabalho em X	Número de linhas de trabalho em *F*	Tempo total [mm:ss]
14	14	1,073,741,824	8,225	13:16
14	30	12,417,064,117	17,616	20:18
30	14	19,717,665,514	38,520	28:15
30	30	48,990,219,122	72,147	38:42

A partir destes testes, pode-se concluir que o algoritmo da fase construtiva é eficaz no sentido de gerar um conjunto F de linhas de boa qualidade (filtradas) e não redundantes. As linhas de trabalho de F foram checadas e obedecem aos critérios de duração, tamanho e dispersão bem como não deixam nenhuma viagem diária a descoberto. Em termos de eficiência, a velocidade da heurística não pode ser comparada com a de outros algoritmos por ser muito específica para um tipo de problema.

Os valores de tempo total encontrados nos testes poderão sofrer mudanças, no futuro, em virtude de aperfeiçoamentos que poderão ser feitos no algoritmo de construção e de percorrimento do grafo G.

4.2 TESTES NO PROCEDIMENTO DE OTIMIZAÇÃO

A finalidade do ensaio foi verificar a eficácia e eficiência da heurística de otimização proposta usando para tal vários problemas de cobertura de conjuntos não unicusto, problemas estes cujas melhores soluções são conhecidas. Com isto pode-se ajustar os parâmetros da heurística evolutiva além de melhorar seus procedimentos.

Para atingir estes objetivos foram empregados os conjuntos de dados da OR-Library, (BEASLEY, 1990). A Tabela 4 mostra algumas características dos mesmos. O número do conjunto de dados está localizado na primeira coluna. As próximas duas colunas mostram o número de linhas e de colunas da matriz de cobertura A. A quarta coluna a densidade da

matriz A. E, finalmente, a última coluna mostra o número de instâncias de teste disponíveis para os testes.

Tabela 4 - Conjuntos de testes usados nos ensaios

Conjunto	Linhas (m)	Colunas (n)	Densidade (%)	Número de instâncias
4	200	1000	2	10
5	200	2000	2	10
6	200	1000	5	5
A	300	3000	2	5
B	300	3000	5	5
C	400	4000	2	5
D	400	4000	5	5
E	500	5000	10	5
F	500	5000	20	5
G	1000	10000	2	5
H	1000	10000	5	5

Cada conjunto de dados mostrados na Tabela 4 possui 5 ou 10 instâncias de problemas diferentes. Os conjuntos F, G e H não foram usados por não estarem disponíveis na época dos ensaios. A instância E estava sem o valor da melhor solução e por isto não se pode comparar os valores obtidos com algum valor de referência.

Os dados referentes aos testes realizados se encontram na Tabela 5, com exceção do tempo de resposta de cada teste. Na primeira coluna vê-se o nome do conjunto de dados de teste. As próximas quatro colunas mostram resultados intermediários da função objetivo durante a convergência do algoritmo. Foram escolhidos alguns número de gerações entre 1 e 100.000 gerações para observar o comportamento da adaptação do melhor indivíduo durante a evolução. As gerações intermediárias escolhidas foram: 80, 2.000, 50.000 e 100.000 gerações. Na última coluna é mostrado o melhor resultado conhecido para o problema.

Tabela 5 – Comparação da heurística proposta com a de Beasley e Chu (1996)

	Número de gerações da heurística proposta				
Conjunto	80	2000	50000	100000	Melhor solução conhecida
4.1	523(50)	432(20000)	432(50000)	429	429
4.2	651	515	512	512	512
4.3	682	538	516	516	516
4.4	681	511	494	494	494
4.5	647	526	512	512	512
4.6	727	570	560	560	560
4.7	517	438	432	**432**	**430 ***
4.8	684	499	492	492	492
4.9	747	680	644	**644**	**641 ***
4.10	738	541	514	514	514
5.1	285	261	253	253	253
5.2	370	316	302	302	302
5.3	261	231	228	226	226
5.4	280	269	242	242	242
5.5	275	212	211	211	211
5.6	288	214	213	213	213
5.7	384	305	294	293	293
5.8	306	290	289	288	288
5.9	357	283	279	279	279
5.10	347	272	265	265	265
6.1	222	145	138	138	138
6.2	213	148	146	146	146
6.3	223	148	145	145	145
6.4	200	132	131	131	131
6.5	219	163	161	161	161
A.1	374	256	254	253	253
A.2	432	262	252	252	252
A.3	349	243	233	232	232
A.4	347	240	234	234	234
A.5	314	240	236	236	236
B.1	98	70	69	69	69
B.2	113	76	76	76	76
C.1	357	234	227	227	227
C.2	327	233	219	219	219
C.3	379	254	249	**246**	**243***
C.4	352	227	219	219	219
C.5	347	217	215	215	215
D.3	108	75	73	72	72

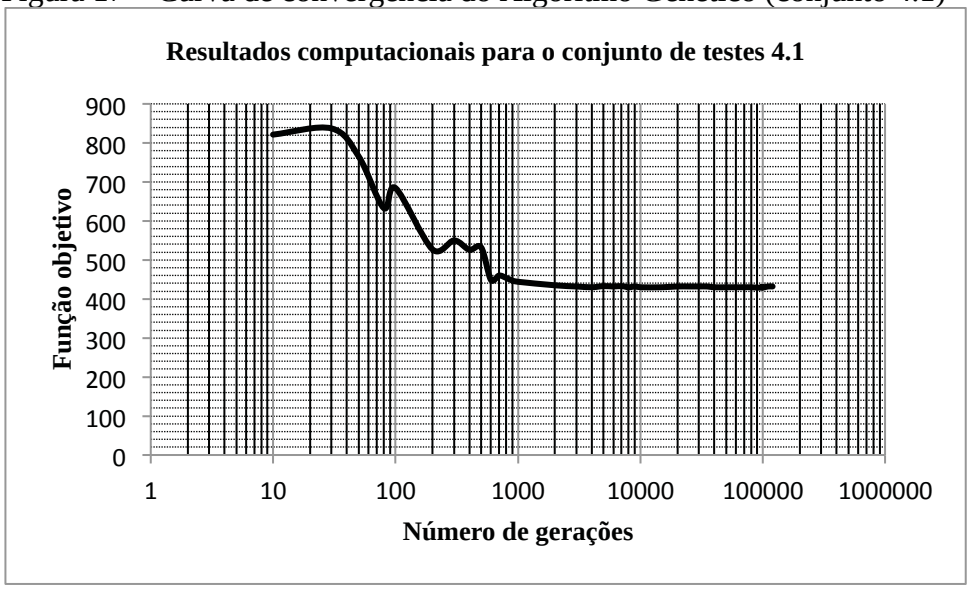
Observa-se que apenas em três testes dos 38 casos, (7,8%), o algoritmo proposto não atingiu o melhor conhecido. Cada caso foi repetido diversas vezes gerando 282 testes realizados no total.

Nos testes realizados até o momento, apenas se procurou observar questões relativas à convergência do algoritmo. Sua eficiência em termos de tempo de reposta não foi computada nesta fase.

Um aspecto interessante de ser observado nos testes é o formato da curva de convergência do algoritmo com objetivo de verificar se há instabilidade e onde, como é o formato durante a estabilização. Se há estabilização ou não de variação dos resultados. Observou-se que o formato da curva de convergência do algoritmo é semelhante para todos os conjuntos de dados testados. A Figura 17 a seguir mostra um deles para o conjunto de dados C.1.

Na Figura 17 pode-se observar uma multimodalidade discreta na região anterior a de estabilização do resultado. Aparentemente tal instabilidade não causa uma influência prejudicial na convergência. Vários testes foram realizados e o mesmo tipo de comportamento se verificou em todos eles. O importante é que o resultado, depois de um certo número de gerações se estabiliza.

Figura 17 – Curva de convergência do Algoritmo Genético (conjunto 4.1)



Pelos resultados obtidos pode-se concluir que o algoritmo evolutivo proposto converge conforme o desejado. Isto é importante uma vez que o algoritmo original de Beasley e Chu (1996) foi bastante alterado. O tempo de resposta é superior ao do Beasley e Chu (1996), porém em nossa proposta há mais processamento devido a questões específicas do nosso

problema. De um modo geral, pode-se dizer que ele não perde em eficácia deixando um pouco a desejar em eficiência, o que pode ter relação com ajustes de parâmetros.

4.3 TESTES NA HEURÍSTICA INTEGRADA EMPREGANDO DADOS REAIS DE UMA EMPRESA

Esta seção mostra o andamento das análises de desempenho realizados até então na heurística proposta. Os testes analisados são os testes realizados considerando o Algoritmo 3 como um todo. Como estamos lidando com um problema real de escalonamento de motoristas, para facilidade de identificação de cada caso, os conjuntos de dados de teste serão designados por PxNy, onde o ‘x’ é o número de dias previsto para o escalonamento e ‘y’ quantas viagens por dia tem a empresa. Exemplo P4N5 quer dizer que a empresa tem uma escala de 4 dias com 5 viagens por dia.

Nos testes realizados fizemos duas comparações. Primeiro submetemos os diferentes conjuntos F gerados a um otimizador comercial. Em seguida, o submetemos ao nosso Algoritmo Genético modificado. O objetivo foi constatar se a parte evolutiva continuava a obter boas respostas quando a massa de dados era a real, pois deve ser lembrado que os dados reais são diferentes em tamanho, densidade e estrutura em relação aos dados da OR-Library.

Em todas as tabelas que se seguem, uma linha representa um conjunto de testes realizados. As colunas são os diferentes tipos de parâmetros da heurística usados naqueles testes. Em geral o número de testes realizados por linha varia de 7 a 10. Dependendo da forma e tamanho da instância e do que se deseja observar, o número de colunas pode variar. A primeira coluna indica o tipo de algoritmo usado (heurística ou otimizador). A segunda coluna identifica o teste realizado a partir do número um. Os dados mostrados em cada linha sempre se referem aos melhores valores observados entre vários testes realizados.

Iniciaremos a análise dos resultados pelos testes com instâncias do problema muito pequenas. Os resultados são mostrados na Tabela 6. A coluna FO mostra o valor final da função objetivo. A próxima coluna à direita desta mostra τ, o número máximo de gerações usado para interromper a heurística. A linha realçada em cinza mostra o resultado do otimizador comercial empregado

.

Tabela 6 – Otimizador x Heurística evolutiva.

Problema: P3N3 F[9x6]										
Tipo Algoritmo	Identifi-cação dos ensaios	Tempo de execução total [s]	Colunas da solução ótima	FO	Número máximo de gerações	Qtde LT por vértice	Param. Baseados nos do set	Coef. mf	Coef. mc	Coef. mg
Heurística	1	4,00	4	**10**	100000	50	set.4	10	200	1,3
Otimizador	1	0,00		**X**			Não achou o ótimo!			
Heurística	1	5,00	4	**11,9**	100000	50	set.4	10	200	1,3
Heurística	2	5,00	4	**11,9**	100000	50	set.4	10	200	1,3
Heurística	3	5,00	4	**11,9**	100000	50	set.4	10	200	1,3
Heurística	4	5,00	4	**11,9**	100000	50	set.4	10	200	1,3
Heurística	5	5,00	4	**11,9**	100000	50	set.4	10	200	1,3
Heurística	6	4,00	4	**11,9**	100000	50	set.4	10	200	1,3
Heurística	7	5,00	4	**11,9**	100000	50	set.4	10	200	1,3
Otimizador	2	0,00		**X**			Não achou o ótimo!			
Heurística	8	0,00	4	**11,9**	10000	50	set.4	10	200	1,3
Heurística	9	0,00	4	**11,9**	10000	50	set.4	10	200	1,3
Heurística	10	0,10	4	**11,9**	1000	50	set.4	10	200	1,3
Heurística	11	0,00	4	**11,9**	500	50	set.4	10	200	1,3
Otimizador	3	0,00		**X**			Não achou o ótimo!			

Conforme se pode observar na Tabela 6, para instâncias muito pequenas do problema o otimizador não consegue obter resposta ótima. A heurística proposta obtém um bom valor em um tempo menor do que 5s.

Para instâncias um pouco maiores, como os mostrados na Tabela 7 e 8, a heurística consegue atingir o ótimo, porém num tempo muito maior do que o do otimizador comercial. Tal resultado pode ser observado na Tabela 7 onde primeiramente o conjunto de dados foi submetido ao otimizador para obter o valor ótimo. Em seguida, variou-se os parâmetros mais significativos da heurística como os coeficientes da função de mutação e número máximo de gerações permitido, sendo observado se a heurística conseguia atingir o mesmo valor do ótimo.

O resultado mostra que a heurística atinge o ótimo, porém em um intervalo de tempo maior do que o do otimizador, conforme se pode observar na coluna de ‘Tempo total de execução’.

Tabela 7 – Otimizador x Heurística evolutiva.

Problema: P5N6 F[30x37]										
Tipo Algoritmo	Id	Tempo de execução total [s]	Número de colunas da solução ótima	FO	Núm. máx. ger.	Qtde LT / vért.	Param. do conjunto	Coef. mf	Coef. mc	Coef. mg
Heurística	1	15,00	8	35,71	100000	50	set.4	10	200	1,3
Heurística	**2**	**14,00**	**9**	**29,83**	**100000**	**50**	**set.4**	**10**	**200**	**1,3**
Heurística	3	00,00	8	43,14	100000	50	set.4	10	200	1,3
Heurística	4	00,00	7	38,4	100000	50	set.4	10	200	1,3
Heurística	5	0,00	8	36,08	50	50	caso real	5	5	1,3
Heurística	6	7,00	9	35,36	50000	50	caso real	10	150	0,09
Heurística	7	14,00	9	29,83	100000	50	caso real	10	150	0,09
Heurística	8	14,00	9	29,83	100000	50	caso real	10	50	0,09
Heurística	9	7,00	9	29,83	50000	50	caso real	10	50	0,09
Heurística	10	7,00	9	29,83	50000	50	caso real	10	50	0,01
Otimizador	1	0,30	9	29,83	Achou ótimo					
Heurística	1	1,00	9	**29,83**	10000	50	caso real	10	50	0,01
Heurística	2	1,00	9	*	10000	50	set.4	10	200	1,3
Heurística	3	1,00s	9	*	10000	50	caso real	10	50	0,01
Heurística	4	1,00	9	*	10000	50	caso real	10	150	0,15
Otimizador	2	0,00	6	34,33	Achou o ótimo					

Na última linha está o caso em que o otimizador acha o ótimo com um número mínimo de colunas, pois as restrições foram mudadas de maior ou igual para apenas igual a 1. O ótimo porém não é encontrado quando as instâncias crescem. Veja o caso do P10N20 mais adiante (Tabela 10).

A vantagem aparente do otimizador sobre a heurística evolutiva reflete o fato do conjunto *F* submetido ao otimizador ser formado por linhas de trabalho de boa qualidade. Não são linhas de trabalho brutas como aquelas que foram descartadas no procedimento de filtragem. Não deve ser esquecido também o fato de que *F* como subconjunto de *X* possui menos linhas do que o conjunto X e caso este fosse submetido diretamente ao otimizador não haveria memória suficiente para carregar o modelo do otimizador em memória.

A Tabela 8 mostra o caso de uma instância onde os coeficientes da função objetivo foram transformados em inteiros. Na primeira coluna, a letra H significa heurística e a letra O é de otimizador. A conclusão destes testes é a mesma dos testes anteriores, ou seja, a heurística atinge o ótimo, porém conforme aumenta o tamanho da instância do problema aumenta também o tempo de processamento da heurística conforme se pode ver na coluna de tempo de execução total.

Tabela 8 – Otimizador x Heurística evolutiva.

Problema: P10N6, F[60x423]									
Tipo	Tempo de execução total [s]	Número de colunas da solução ótima	FO	Núm. máx. ger.	Qtde LT por vértice	Coef. do conjunto	Coef mf	Coef mc	Coef mg
H	5,00	8	2500	5000	50	set.04	10	200	1,3
H	10,00	9	2500	10000	50	set.04	10	200	1,3
H	10,00	7	2100	10000	50	set.04	10	200	1,3
H	22,00	8	2400	20000	50	set.04	10	200	1,3
H	0,00	8	2300	30000	50	set.04	10	200	1,3
H	0,00	7	2100	40000	50	set.04	10	200	1,3
H	56,00	7	2100	50000	50	set.04	10	200	1,3
H	120,02	8	2100	100000	50	set.04	10	200	1,3
H	120,06	8	2100	100000	50	set.E	10	350	1.1
H	**120,04**	**7**	**1800**	**100000**	**50**	**set.D**	**10**	**200**	**2**
H	120,06	7	2100	100000	50	set.D	10	200	2
H	**120,02**	**7**	**1800**	**100000**	**50**	**set.D**	**10**	**100**	**2**
H	119,00	7	2100	100000	50	set.04	10	200	1,3
H	115,00	7	2100	100000	50	set.04	10	200	1,3
H	**120,00**	**6**	**1800**	**100000**	**50**	**set.04**	**10**	**200**	**1,3**
H	**120,02**	**7**	**1800**	**100000**	**50**	**set.04**	**10**	**200**	**1,3**
H	**120,19**	**7**	**1800**	**100000**	**50**	**set.04**	**10**	**200**	**1,3**
H	**120,00**	**6**	**1800**	**100000**	**50**	**set.04**	**10**	**200**	**1,3**
H	**116,00**	**7**	**1800**	**100000**	**50**	**set.04**	**10**	**200**	**1,3**
H	**120,00**	**7**	**1800**	**100000**	**50**	**set.04**	**10**	**200**	**1,3**
H	120,00	8	2000	100000	50	set.04	10	200	1,3
H	119,00	8	2100	100000	50	set.04	10	200	1,3
H	**113,00**	**6**	**1800**	**100000**	**50**	**set.04**	**10**	**200**	**1,3**
H	113,00	7	2100	100000	50	set.04	10	200	1,3
H	0,00	6	2000	100000	50	set.04	10	200	1,3
O	**0.10**	**7**	**1800**	**Ótimo atingido**					

Conforme se observa na tabela 8 o tempo de busca da heurística aumenta consideravelmente para a ordem de 2 minutos. A variação da taxa de bits ao longo do tempo no procedimento de mutação sugere que não há influencia da mesma na determinação do ótimo pela heurística. Outras tentativas de mudanças de parâmetros como o tamanho da população, os coeficientes da função objetivo, foram tentadas para reduzir este tempo sem sucesso. Porém devem ser feitos mais testes para que haja evidências de que o tempo realmente não possa ser reduzido. Com valores de número máximo de gerações inferiores a cem mil filhos, a heurística não atinge o ótimo.

A Tabela 9 mostra a aplicação da heurística para um caso maior, o P10N10.

Tabela 9 - Otimizador x Heurística evolutiva.

Problema P10N10										
Tipo Algoritmo	Ensaios	Tempo de execução total	Número de colunas da solução ótima	FO	Núm. máx. ger.	Qtde LT/ vért.	Param.	Coef. mf	Coef. mc	Coef. mg
Heurística	1	133,00	19	**8200**	100000	50	set.E	10	350	1,1
Otimizador	1	0,40	19	**8200**	Ótimo atingido					
Heurística	2	146,00	18	**5400**	100000	50	set.E	10	350	1,1
Otimizador	2	0,00	16	**5000**	Ótimo atingido					
Heurística	3	144,00	20	**6800**	100000	50	set.E	10	350	1,1
Otimizador	3	0,10	20	**6800**	Ótimo atingido					
Heurística	4	133,00	15	**6600**	100000	50	set.E	10	350	1,1
Otimizador	4	0,10	15	**6600**	Ótimo atingido					
Heurística	5	146,00	20	**7800**	100000	50	set.E	10	350	1,1
Otimizador	5	0,.30	20	**7800**	Ótimo atingido					
Heurística	6	144,00	15	**5000**	100000	50	set.4	10	200	1,3
Otimizador	6	0,10	14	**4800**	Ótimo atingido					
Heurística	7	142,00	16	**6200**	100000	50	set.4	10	200	1,3
Otimizador	7	0,40	16	**6100**	Ótimo atingido					

Em quatro dos sete conjuntos de testes realizados a heurística atinge o mesmo valor do ótimo, porém a um custo maior de tempo. Nos demais conjuntos de testes realizados a heurística não atingiu o ótimo com os parâmetros usados. Mais testes deverão ser feitos alterando os parâmetros do algoritmo genético para este tipo de conjunto F.

O ultimo teste realizado nesta bateria de testes foi com uma matriz de cobertura de 200 linhas por 10989 colunas. O resultado é mostrado na Tabela 10 onde cada teste foi realiado para 10 iterações da heurística.

Tabela 10 - Otimizador x Heurística evolutiva. Caso de teste: P10N20 F[200 x 10989]

Método	Tempo de execução [s]	Número de colunas da solução ótima	Melhor FO	Núm. máx. ger.	Qtde LT/ vért	LTs p/ cobrir viagem não coberta	Param. do set	mf	Mc	mg
Heurística	1756,00	67	**461,313**	50000	50	<= 100	Ajus-tados	10	5000	0,0015
Otimizador	0,40	67	**450,157**	Ótimo atingido – Formulação de cobertura						
Otimizador	-	-	-	Não atingido ótimo – Formulação de particionamento						

Conforme se observa neste teste o tempo de resposta da heurística é excessivamente grande em relação ao tempo de execução do método determinístico. A razão disto pode ser que o problema formulado e submetido ao otimizador se refere a colunas boas do ponto de vista de cobertura, viabilidade e função objetivo. Supõe-se que este fato leve o otimizador a encontrar rapidamente o valor ótimo. A realidade é que ao montar o problema para o otimizador se eliminou colunas de baixa qualidade durante a fase construtiva.

Um teste ideal de ser feito seria o de criar um modelo matemático exato para tratar o problema real sem a fase construtiva e submetê-lo ao otimizador comercial. Isto, porém não é possível devido ao tamanho da instância, dificuldades de modelar as restrições e a função multi objetivos do problema.

No entanto, uma comparação em relação à solução manual na época em que os dados da empresa foram cedidos para o estudo mostra que a solução da empresa não resolvia o problema devido a ter que ser empregado um livro de compensação de horas trabalhadas e de folga entre os motoristas, tamanha era a discrepância de horas trabalhadas entre os mesmos no final de um mês. Além disso, vale dizer que as escalas mensais acertadas previamente para o próximo mês não tinham nenhuma otimização em relação à carga a ser dada aos motoristas nem tampouco em relação a dispersão de duração de viagens em cada escala. Tais ganhos certamente poderiam ser mensurados e ficam para um trabalho futuro se as escalas reais puderem ser obtidas.

4.4 SOLUÇÃO DO PROBLEMA DOS PADRÕES DE FOLGA

Nesta seção são apresentados os resultados obtidos com a resolução deste problema usando a Formulação 6 e um otimizador.

O otimizador usado foi o Xpress© versão 7.2 da FICO©. Primeiramente, a Tabela 11 mostra a matriz de padrões de folga da empresa que foi empregada nos testes.

Tabela 11 - Matriz de padrões de folga e de demanda de motoristas

Dias	pf 5x1a	pf 5x1b	pf 5x1c	pf 5x1d	pf 5x1e	pf 5x1f	pf 4x2a	pf 4x2b	pf 4x2c	pf 3x1a	pf 3x1b	pf 3x1c	pf 3x1d	Motoristas
1	1	0	1	1	1	1	1	0	1	0	1	1	1	20
2	1	1	0	1	1	1	1	0	1	1	0	1	1	20
3	1	1	1	0	1	1	1	1	0	1	1	0	1	20
4	1	1	1	1	0	1	1	1	0	1	1	1	0	20
5	1	1	1	1	1	0	0	1	1	0	1	1	1	20
6	0	1	1	1	1	1	0	1	1	1	0	1	1	20
7	1	0	1	1	1	1	1	0	1	1	1	0	1	20
8	1	1	0	1	1	1	1	0	1	1	1	1	0	20
9	1	1	1	0	1	1	1	1	0	0	1	1	1	20
10	1	1	1	1	0	1	1	1	0	1	0	1	1	20

Conforme a Tabela 11 ilustra, foram utilizados 13 padrões de folga. Cada padrão tem um certo número de dias de expediente intercalados por dias de folga. O resultado encontrado pelo otimizador é mostrado na Tabela 12.

Tabela 12 – Solução do problema de padrões de folga

Z	24
x(1)	4
x(2)	4
x(3)	4
x(4)	4
x(5)	4
x(6)	4
x(7)	0
x(8)	0
x(9)	0
x(10)	0
x(11)	0
x(12)	0
x(13)	0

Esta Tabela 12 mostra que deverão existir 4 motoristas alocados para o 5x1a, 4 para o 5x1b até o 5x1f. No total a empresa deverá ter 24 motoristas por dia para atender a demanda de 20 viagens diárias apenas uma vez que sempre existirão 4 motoristas que estarão de folga

dentro do seu padrão de folga. Observa-se que nesta solução não foi preciso utilizar os outros padrões de folga da empresa.

5 CONCLUSÃO E TRABALHOS FUTUROS

Procurou-se resolver nesta pesquisa o problema do escalonamento de motoristas de ônibus. Os dados de uma empresa real foram empregados para testar o modelo criado. As restrições e a função objetivo do problema foram obtidos de uma empresa. Usou-se teoria de grafo para percorrimento do espaço de solução do problema e geração das linhas de trabalho viáveis e qualificadas numa primeira fase da solução, denominada fase construtiva. Um novo problema foi obtido nesta fase e formulado como um problema de cobertura de conjuntos não unicusto sendo resolvido por intermédio de heurística evolutiva baseada em algoritmo genético.

Conforme se pode observar neste trabalho, várias alterações foram introduzidas no algoritmo proposto por Beasley e Chu (1996) tais como o tratamento do número de elementos da população inicial, da taxa de mutação, o procedimento de escolha de novos pais, etc. Tais mudanças foram necessárias para adequar um Algoritmo Genético especializado em resolver problemas gerais de cobertura de conjuntos não-unicusto, que é o caso de Beasley e Chu (1996), para o nosso caso que é o da resolução de problemas específicos de criação de linhas de trabalho em empresas de transporte público, nos quais o tamanho das instâncias e formato das matrizes de cobertura são diferentes do formato clássico do problema de cobertura não unicusto.

Uma sugestão para trabalhos futuros é incorporar à formulação do problema atual uma nova restrição para diminuir as variações de trocas de cidades ao longo dos dias da escala buscando criar ciclos repetitivos de forma que o mesmo motorista possa a cada semana, 15 dias, 21 dias etc repetir-se na mesma viagem da escala.

A questão da influência de tamanho e estrutura das instâncias de F na determinação da melhor faixa para ρ (o número de indivíduos da população inicial) também é um ponto que deverá ser mais pesquisado no futuro.

Finalmente é necessário justificar o motivo pelo qual ainda não foi possível comparar os resultados obtidos até o presente momento com escalas reais de trabalho e padrões de folga. O motivo é que não temos acessos às escalas reais por serem confidenciais. Portanto o máximo que poderemos fazer, num futuro próximo, é levar alguns resultados da heurística para a empresa avaliar em termos de qualidade.

ANEXO A – Programa em Mosel para resolver o caso do padrão de folgas P10N20

Ilustra-se a seguir como o subproblema dos padrões de folga do caso de teste P10N20 foi modelado para solução usando a linguagem de modelagem do otimizador comercial Xpress ©, denominada Mosel © da empresa FICO ©.

```
! Problema dos turnos
! SPP unicusto 10 x 13. Restrições = 20 para o caso P10N20
model TurnosP10N20

uses "mmxprs"; !gain access to the Xpress-Optimizer solver

!sample declarations section
declarations
! inicializações
        ! Índices
        linha = 1..10
        coluna = 1..13

        ! Vetores e matrizes
        CUSTO  : array(coluna) of integer
        B      : array(linha) of integer
        MATCOB : array(linha,coluna) of integer
        x_sol  : array(coluna) of real

        ! Variáveis de decisão
        x : array(coluna) of mpvar

end-declarations
forall (i in coluna) x(i) is_integer
forall (j in coluna) x_sol(j) := 0

! le a formulação do arquivo
initializations from 'Formulacao.dat'
        CUSTO B MATCOB
end-initializations

! monta as equações matemáticas correspondentes e manda otimizar
! f.o.
z:= sum(j in coluna) CUSTO(j) * x(j)

!restrições
forall(i in linha) sum(j in coluna) MATCOB(i,j) * x(j) = B(i)

! escreve a formulação do problema num arquivo extensão ".lp"
! e depois na tela do XPress
writeln("Formulação do problema:")
exportprob(EP_MIN, "TurnosP10N20",z)
writeln

writeln("Executando o solver")
writeln("...")
writeln
! otimizar
minimize(z)
writeln("Fim de execução do otimizador")
```

```
! PRIMEIRO escreve o resultado na tela do XPress
writeln("RESULTADO")
if(getprobstat=XPRS_OPT) then
        writeln('Otimo atingido!')
        writeln("z = ", getobjval);
        forall(j in coluna)
                writeln("x(", j, ") = ", getsol(x(j)))
        writeln
else
        writeln('otimo não atingido!')
end-if

! SEGUNDO no arquivo TurnosP10N20.sol
forall(j in coluna) do
        x_sol(j) := getsol(x(j))
        z_sol := getobjval
end-do

initializations to 'TurnosP10N20.sol'
        x_sol
        z_sol
end-initializations

end-model
```

REFERÊNCIAS

AICKELIN, U. **Genetic Algorithms for Multiple-Choice Optimisation Problems**. Ph.D. Thesis. Nottingham, Universidade de Auckland, 1999.

ALFARES, H.K. Survey, Categorization, and Comparison of Recent Tour Scheduling Literature. **Annals of Operations Research** - Special Issue on Staff Scheduling and Rostering, v. 127, p. 145-175, 2004.

BAUTISTA, P.; PEREIRA, J. A GRASP algorithm to solve the unicost set covering problem. **Computers & Operations Research**, p. 3162-3173, 2007.

BAUTISTA, P.; PEREIRA, J. Modeling the problem of locating collection areas for urban waste management: An application to the metropolitan area of Barcelona. **The International Journal of Management Science**, v. 34, p. 617-629, 2006.

BEASLEY, J.E. OR-library: Distributing test problems by eletronic mail. **Journal of the Operations Research Society**, p. 1069-1072, 1990.

BEASLEY, J.E.; CHU, P.C. A genetic algorithm for the set covering problem. **European Journal of Operational Research**, v. 94, p. 392-404, 1996.

BEASLEY, D.; BULL, D.R.; MARTIN, R.R. An overview of genetic algorithms: 1. Fundamentals. **University Computing**, v. 15, n. 2, p. 58-69, 1993.

BEASLEY, D.; BULL, D.R.; MARTIN, R.R. An overview of genetic algorithms: 2. Research Topics. **University Computing**, v. 15, n. 4, p. 170-181, 1993.

BERTRAND, J.W.M.; FRANSOO, J. Operations management research metodologies using quantitative modeling. **International Journal of Operations & Production Management**, v. 22, n. 2, p. 241-264, 2001.

CAPPANERA, P.; GALLO, G. A Multicommodity Flow Approach to the Crew Rostering Problem. **Operations Research**, v. 52, n. 4, p. 583-596, 2004.

CAPRARA, A.; FISCHETTI, M.; TOTH, P. A heuristic method for the set covering problem. **Operations Research**, v.47, n. 5, p. 730-743, 1999.

CAPRARA, A.; FISCHETTI, M.; TOTH, P.; VIGO, D.; GUIDA, P.L. Algorithms for railway crew management. **Mathematical Programming**, v. 79, p. 125-141, 1997.

CHEN, C.; LIU, T.; CHOU, J. Integrated Short-Haul Airline Crew Scheduling Using Multiobjective Optimization Genetic Algorithms. **IEEE Transactions on Systems Man Cybernetics Systems**, v. 43, n. 5, p.1077-1090, 2013.

DANTZIG, G. A comment on edie traffic delays at tolls booths. **Journal of the Operations Research Society of America**, v. 2, n. 2, p. 107-138, 1954.

ELISONDO, R.; PARADA, V.; PRADENAS, L. An evolutionary and constructive approach to a crew scheduling problem in underground passenger transport. **Journal of Heuristics**, v. 16, n. 4, p. 575-591, 2010.

ERNST, A.T.; JIANG, H.; KRISHNAMOORTHY, M.; SIER, D. Staff scheduling and rostering: A review of applications, methods and models. **European Journal of operational Research**, v. 153, n. 1, p. 3-27, 2004a.

ERNST, A.T.; JIANG, H.; KRISHNAMOORTHY, M.; OWENS, B.; SIER, D. An Annotated Bibliography of Personnel Scheduling and Rostering. **Annals of Operations Research**, v. 127, p. 21-144, 2004b.

ESCLAPÈS, C. **Asignación de conductores a jornadas de trabajo en empresas de trasnporte colectivo**. Tesis doctoral, Universitat Politécnica de Catalunya, Departamento d'Estatistica i Investigació Operativa, 2000.

FLEURY, A. **Planejamento do projeto de pesquisa e definição do modelo teórico**, Em: P. A. Miguel, Metodologia de pesquisa em engenharia de produção e gestão de operações. p. 31-44. Rio de Janeiro, Elsevier, 2010.

GLOVER, F. Tabu Search - Part I. **ORSA Journal on Computing**, v. 1, n. 3, p. 190-206, 1989.

GLOVER, F. Tabu Search - Part II. **ORSA Journal on Computing**, v. 2, n. 1, p. 4-32, 1990.

GROSSMAN, T.; WOOL, A. Computational experience with approximation algorithms for the set covering problem. **European Journal of Operational Research**, v. 101, n. 1, p. 81-92, 1997.

HOLAND, J.H. **Adaption in Natural and Artificial Systems**, Cambridge, MA: MIT Press, 1975.

KARP, R. Reducibility among combinatorial problems. **Symposium on Mathematical Programming**, Madison: University of Wisconsin, 1972.

KOHL, N.; KARISH, S. E. Airline Crew Rostering: problem Types, Modeling, and Optimization. **Operations Research**, v. 127, p. 223-257, 2004.

LAN, G.; DEPUY, G.W.; WHITEHOUSE, G.E. An effective and simpleheuristic for the set covering problem. **European Journal of Operational Research**, v. 176, n. 3, p. 1387-1403, 2007.

LESAUN, M.; PEREZ, G.; DE LA MAZA, E.S. Staff rostering for the station personnel of a railway company. **Journal of the Operational Research Society**, v. 61, n. 7, p. 1104-1111, 2010.

LOURENÇO, H. R.; PAIXÃO, J. P.; PORTUGAL, R. Multiobjective Metaheuristics for the Bus-Driver Scheduling Problem**. Transportation Science**, v. 35, n. 3, p. 331, 2001.

MAENHOUT, B.; VANHOUCKE, M. A hybrid scatter search heuristic for personalized crew rostering in the airline industry. **European Journal of Operational Research**, v. 206 , n. 1, p. 155-167, 2010.

MASON, A.; NIELSEN, D. **PETRA: A programmable optimisation engine and toolbox for personnel rostering applications**. Ph.D. Thesis, University of Auckland, New Zealand, 1999.

MESQUITA, M.; MOZ, M.; PAIAS, A. A new model for the integrated vehicle-crew-rostering problem and a computational study on rosters. **Journal of Scheduling**, v. 14, n. 4, p. 319-334, 2011.

MORABITO, R.; PUREZA, V. **Modelagem e Simulação**. Em: P. A. Miguel, Metodologia de Pesquisa em Engenharia de Produção e Gestão de Operações, p. 165-192. Rio de Janeiro, Elsevier, 2010.

NAJI-AZIMI, Z.; TOTH, P.; GALLI, L. An eletromagnetism metaheuristic for the unicost set covering problem. **European Journal of Operational Research**, v. 205, n. 2, p. 290-300, 2010.

NEMHAUSER, G.L.; WOLSEY, L. A.**Integer and Combinatorial Optimization**. New York: John Wiley & Sons, 1988.

OLIVEIRA, A.C. **Algoritmos Evolutivos Para Problemas de Otimização Numérica Com Variáveis Reais**. Monografia do Exame de Qualificação do Curso de Computação Aplicada, Instituto Nacional de Pesquisas Espaciais - INPE, Computação Aplicada, São José dos Campos, 2001.

PANTON, D.M.; RYAN, D.M. **Column Generation Models for Optimal Workforce Allocation with Multiple Breaks**. New Zeland, 1999.

SHODI, M.; NORRIS, S. A flexible, fast, and optimal modeling approach applied to crew rostering at London Underground. **Annals od Operations Research**, v. 127, p. 259-281, 2004.

WOLSEY, L.A. **Integer Programming**. New York: John Wiley & Sons, Inc, 1998.