AUTOMAÇÃO E OTIMIZAÇÃO DE ESCALONAMENTO NA INDÚSTRIA DE ABATE DE AVES

Ademir Aparecido Constantino
Departamento de Informática – Universidade Estadual de Maringá (UEM)
Avenida Colombo, 5790 – 87020-900 – Bloco C56 – PR – Brazil
aaconstantino@gmail.com

Wesley Romão
Departamento de Informática – Universidade Estadual de Maringá (UEM)
Avenida Colombo, 5790 – 87020-900 – Bloco C56 – PR – Brazil
wesley.uem@gmail.com

Dario Landa-Silva
School of Computer Science – University of Nottingham
Nottingham – NG8 1BB – UK
dario.landasilva@nottingham.ac.uk

RESUMO

Este artigo aborda um problema de escalonamento de pessoal em uma indústria de abate de aves. O objetivo é automatizar a geração de escala diárias e minimizar a quantidade de horas extras paga para o pessoal e o tempo de espera dos caminhões no pátio da fábrica. Este artigo propõe um algoritmo memético que utiliza busca local e simulação discreta determinística. Os experimentos foram executados com dados reais sobre um período de tres meses de operação e os resultados mostram que o método é efetivo para produzir escalas robustas.

PALAVRAS CHAVE. Escalonamento, Agroindústria, Algoritmos meméticos

Área principal. PO na Indústria.

ABSTRACT

This paper addresses a personal scheduling in a farming industry that kills poultries. The schedule should ensure constant supply of poultries to the factory to avoid idle processing time. The aim is to automate the daily scheduling generation and to minimizing the extra time paid to people and the waiting time of lorries at the factory parking. This paper proposes a memetic algorithm which uses local search and deterministic discrete simulation. The experiments were carried out with real data from a operation period of three months and the results show that the method is effective to produce robust schedules

KEYWORDS. Scheduling, Farming Industry, Memetic algorithm

Main area. OR in Industry.

1. Introdução

Escalonamento *(scheduling)* de atividades significa designar um conjunto de eventos ao longo do tempo, atendendo a um conjunto de restrições e, em geral, otimizando alguma função objetivo. Os problemas de escalonamento desempenham um papel importante dentro da computação, especialmente dentro da pesquisa operacional, pelo fato de fomentar a investigação de modelos computacionais e de novas técnicas algorítmicas.

O problema de escalonamento de atividades pode ser encontrado em diversos contextos, seja ele envolvendo máquinas (simples ou paralela) (SHEEN; LIAO, 2007; BLAZEWICZ, 1996), processos (SONMEZ; GURSOY, 2007; BANSAL *et al.*, 2009) ou pessoas (YUNES *et al.*, 2000). O escalonamento de pessoal pode ser dividido em dois tipos: escalonamento de pessoal em local fixo e escalonamento de pessoal no setor de transportes (BODIN, 1983). O escalonamento no setor de transportes é um dos mais complexos do ponto de vista computacional (BODIN, 1983). O desafio computacional, portanto, é encontrar uma metodologia que obtenha uma solução satisfatória para o problema em tempo computacional viável.

Como reportado por Bodin (1983), a maioria dos problemas de escalonamento não admitem solução em tempo polinomial e são poucos os casos modelados por programação matemática utilizando algoritmos exatos (YUNES *et al.*, 2000). O escalonamento na agroindústria tratado aqui se diferencia do escalonamento de transporte pelo fato da hora início de cada viagem ser desconhecido. O problema investigado surge no contexto de uma indústria que processa mais de um milhão de aves por semana onde pequenas melhorias no escalonamento pode resultar em redução significativa do desperdício implicando em redução de custos. O escalonamento automatizado deve ser capaz de definir a escala de atividades das equipes que realizam a apanha nas granjas, assegurando o constante fornecimento de suprimento para a indústria, respeitando um conjunto de restrições operacionais e otimizando uma função objetivo. Além disso, informações sobre o tamanho da fila de caminhões ou tempo de espera na fila são informações que devem ser otimizadas. Por outro lado, essas informações são complexas para serem integradas em um modelo de programação matemática.

O problema específico apresentado neste artigo não tem sido reportado na literatura, exceto uma variação do mesmo apresentado por Hart *et al.* (1999), também baseado em um caso real do norte da Inglaterra. Neste caso, a empresa não desejava um escalonamento otimizado, mas apenas um sistema que automatizasse o escalonamento minimizando o número de violações de restrições. Portanto, o caso foi tratado como um problema de satisfação de restrições, o qual foi abordado usando algoritmos genéticos. Além desse aspecto, outras características diferenciam do problema investigado, tais como, restrições sobre o tempo de espera no pátio da fábrica, restrições sobre a formação das escalas das equipes. Mais informações sugerimos a consulta do trabalho de Hart *et al.* (1999).

Este trabalho propõe um algoritmo baseado em algoritmos meméticos, integrando-o à programação matemática e técnica de simulação discreta determinística para cálculo das estatísticas das filas. O algoritmo possui duas fases hierárquicas, sendo que a primeira fase procura minimizar o custo com as equipes de apanha (equipes que realizam a coleta das aves nas granjas) enquanto que a segunda fase tenta melhorar o resultado focando na minimização do tamanho da fila de espera na fábrica.

O artigo esta estruturado em 5 seções. Na seção 2 o problema investigado é apresentado em detalhes. Na seção 3 é apresentado um modelo de programação matemática que visa a otimização dos custos de trasporte. O algoritmo proposto é apresentado na seção 4. Os experimentos computacionais são apresentados na seção 5. Finalmente na seção 6 são apresentadas as conclusões e propostas de novas investigações.

2. Descrição do Problema

O problema investigado surge de um caso real observado no setor da agroindústria, cujo insumo processado é oriundo de carga de aves vivas que não podem ser estocadas por muito tempo. O

processamento das aves ocorrem em três plantas (fábricas) situadas em diferentes localizações. Os caminhões com carga que chegam na fábrica devem aguardar num galpão ventilado, principalmente nos períodos de alta temperatura do verão. O desafio diário da empresa é realizar o escalonamento das equipes de apanha para manter as fábricas em constante funcionamento, atendendo as restrições operacionais e trabalhistas, além de minimizar os custos.

As aves devem ser coletadas em diferentes granjas em locais espalhados geograficamente num raio de aproximadamente 150 Km de cada fábrica. A cada dia novos locais de coleta das aves são definidos, implicando na necessidade de realizar escalonamentos diários. O pessoal que realiza a apanha das aves é dividido em várias equipes que podem ser designadas para diferentes locais durante o dia, logo o tempo de transporte entre os locais de apanha deve ser minimizado. As equipes têm sua base em um local diferente das fábricas. Algumas informações como a sequência de locais de apanha, a hora de início e término das atividades, a hora do carregamento e descarregamento dos caminhões e o destino de cada carregamento (carga) devem ser descobertas com a definição do escalonamento.

O escalonamento deve ser produzido de forma a minimizar os custos operacionais que incluem custos com horas extras para as equipes de apanha, custos com horas ociosas das equipes, custos com transporte e custos relacionados ao tempo de espera do caminhão no pátio da empresa. Considerando o alto custo da ociosidade da indústria, este custo é considerado como uma restrição do problema. Assim, a função objetivo se resume na minimização de custos com as equipes de apanha (horas pagas) e ao tempo de espera na fila (pátio da fábrica).

Além disso, algumas restrições (regras) devem ser atendidas na elaboração das escalas de trabalho. As regras a serem seguidas são as seguintes:

- O tempo normal de trabalho são 7 horas e 20 minutos (excluindo o tempo de descanso). O tempo que extrapolar é considerado hora extra decorrendo um acréscimo de 50% sobre o valor pago no tempo normal.
- Se o total de horas trabalhadas for superior a 4 horas e não exceder a 6 horas, deverá ser adicionado um tempo de descanso de, no mínimo,15 minutos.
- Se o total de horas trabalhas for superior a 6 horas deverá ser adicionado um descanso de no mínimo 1 hora.
- O descanso deve ser de no máximo 2 horas, pois acima desse tempo será considerado tempo pago. A hora extra também é contabilizada sobre o tempo mínimo de descanso caso não seja considerado na escala.

Além das restrições operacionais, existem alguns dados gerais que devem ser utilizados pelo algoritmo, tais como: localização de todas as granjas, localização da base das equipes de apanha, localização das fábricas, distância entre todos os locais, velocidade média dos caminhões, velocidade de abate de cada fábrica, hora de início de operação de cada fábrica, hora de término de operação de cada fábrica e capacidade de cada caminhão.

O escalonamento é realizado diariamente e os dados de entrada do algoritmo, específicos para cada dia, devem ser: as granjas, onde serão coletadas as aves, e o número de aves de cada granja.

3. Modelagem Matemática

Como resultado da divisão do número de aves de uma granja pela capacidade de transporte de cada veículo, temos o que denominamos de *carga*. Cada carga está associada as seguintes informações: local de origem (granja), local de destino (fábrica), hora de início do carregamento, duração do carregamento, equipe de apanha responsável e número de aves.

Uma vez conhecido o conjunto de granjas e suas informações, como localização e número de aves, então o número de cargas, as suas origens e a duração para carregamento são obtidos automaticamente. Portanto, resta descobrir o destino (fábrica), a equipe responsável pelo carregamento e a hora de início do carregamento. Uma etapa fundamental para minimização dos custos é descobrir a associação das cargas as fábricas. Para isso foi criado um modelo de programação linear inteira binária descrito a seguir:

$$Min \sum_{g=1}^{ng} \sum_{f=1}^{nf} d_{gf} x_{gf} \tag{1}$$

$$Sujeito\ a: \quad \sum_{f=1}^{nf} x_{gf} = 1;\ g = 1, ..., ng; \tag{2}$$

$$\sum_{g=1}^{ng} s_g x_{gf} \leq cp_f;\ f = 1, ..., nf \tag{3}$$

$$x_{gf} \in \{0, 1\},\ g = 1, ..., ng;\ f = 1, ..., nf$$

Onde ng é o *número de cargas*, nf é o *número de fábricas*, d_{gf} é a *distância da granja* que originou a carga g $(g = 1, ..., ng)$ até a fábrica f $(f = 1, ..., nf)$, s_g é o *número de aves da carga* g e cp_f a *capacidade de processamento* (número de aves) da fábrica f. Por fim, x_{gf} é a variável de decisão, sendo que $x_{gf} = 1$ se a carga g for alocada a fábrica f, sendo nf o número de fábricas. A restrição 2 garante que apenas uma carga seja alocada a uma única fábrica, enquanto a restrição 3 garante que a capacidade da fábrica seja respeitada. A função objetivo 1 procura minimizar a soma total das distâncias relacionadas com o transporte das aves.

Este modelo de programação matemática resolve apenas uma parte do problema global. Em função da dificuldade de se obter um modelo que resolva integralmente o problema, foi proposto de um algoritmo heurístico que utiliza o modelo de designação linear em duas fases: fase de construção da solução e fase de melhoramento (busca local). A formulação do problema de designação (PD), denotada por $PD([c_{ij}])$, é a seguinte:

$$z = Min \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} c_{ij} x_{ij} \tag{4}$$

$$Sujeito\ a: \quad \sum_{i=1}^{n_1} x_{ij} = 1,\ \forall j = 1, ..., n_2 \tag{5}$$

$$\sum_{j=1}^{n_2} x_{ij} = 1,\ \forall i = 1, ...n_1 \tag{6}$$

$$x_{ij} \in \{0, 1\},\ \forall i = 1, ..., n_1,\ \forall j = 1, ..., n_2$$

A variável de decisão x_{ij} binária $\{0, 1\}$, sendo que $x_{ij} = 1$ se o item i for associado com o item j, caso contrário $x_{ij} = 0$, e c_{ij} é o custo de associar o item i ao item j. Os itens podem ter diferentes significados de acordo com a fase de resolução (construção e melhoramento), apresentadas nas seções seguintes. O valor de n_1 deve ser igual a n_2, porém na fase de construção da solução. $n1$ representa o número de equipes (reais e fictícias) e $n2$ o número de cargas ng. Considerando que o número de cargas é sempre superior ao número de equipes, são criadas equipes fictícias para tornar a matriz $[c_{ij}]$ quadrada. Já na fase de melhoramento, n_1 e n_2 serão sempre o *número máximo de equipes* ne_{max} permitidas para o escalonamento. Embora o algoritmo utilize o número máximo de equipes disponíveis, deve-se observar que o algoritmo proposto procura automaticamente minimizar o número de equipes tendo em vista que o objetivo é minimizar o custo global com pessoal.

4. Algoritmo Proposto

O método proposto é um algoritmo evolutivo que utiliza alguns procedimentos construtivos e de busca local (melhoramento). Nesta seção serão apresentadas a forma de representação da solução, os procedimentos construtivos, o procedimento de busca local e o procedimento geral baseado em algoritmos genéticos.

Uma escala de trabalho para uma equipe é uma sequência de atividades a ser executada ao longo do dia. Porém, o tempo de carregamento de um caminhão é de aproximadamente uma hora. Portanto, a escala de trabalho pode ser discretizada em uma sequência de atividades com duração de uma hora. Cada atividade está relacionada com uma carga definida anteriormente. Para efeito de simplificação do problema, a duração do carregamento de todas as cargas será considerado de uma hora. Assim, considere t o índice de tempo ao longo do dia de trabalho, sendo $t = 1, ..., 24$, onde $t = 1$ significa o primeiro horário do dia em que alguma equipe inicia o trabalho de carregamento.

4.1. Procedimento Construtivo

Dado um número máximo de equipes ne_{max} pré-definido, a fase de construção da solução inicial parte do princípio de que a sequência de atividades (cargas) de cada equipe ainda está vazia e então as cargas são designadas sucessivamente até que todas as cargas sejam para o número máximo de equipes. A ideia do algoritmo é atribuir as cargas para as equipes considerando que todos os carregamentos iniciam em $t = 1$. Assim, em $t = 1$ todas as equipes recebem a designação de uma carga selecionada aleatoriamente. Para os demais intervalos de tempo $t(t > 1)$ a designação é resolvida utilizando o problema de designação o qual está associado a uma matriz de custo c_{ij}, de dimensão ng, relacionada ao custo de associar uma carga j a uma equipe i. Como o número de cargas é sempre maior que o número de equipes (ne_{max}), então a matriz é dividida em dois blocos e assume valores conforme definido na Figura 1.

	Cargas
Equipes	*Bloco I* $c_{ij} = fa(i, j)$
Equipes Fictícias	*Bloco II* $c_{ij} = \infty$

Figura 1. Estrutura da matriz de custo para a fase de construção da solução.

Onde os custos do bloco I da matriz são calculados pela seguinte função:

$$fa(i, j) = hp + ph, \tag{7}$$

sendo hp as horas pagas para a equipe i ao ser designada ao trabalho de carregamento da carga j, e ph uma penalidade caso alguma restrição não seja atendida. Para o cálculo das horas pagas são levadas em considerações as regras apresentadas na seção 2. Como penalidade foi considerado o limite de 9 horas de trabalho contínuo. O algoritmo do procedimento de construção é apresentado na Figura 2.

4.2. Busca Local baseada em Custo

A busca local tem o objetivo de realizar pesquisa em torno de uma solução corrente (algumas vezes chamada de busca na vizinhança) na tentativa de encontrar soluções aprimoradas. Neste trabalho propomos dois procedimentos de busca que são utilizados em um algoritmo baseado em VND (*Variable Neighborhood Descent*) (HANSEN; MLADENOVIC, 2001).

Procedimento *Construção*
Início
$t := 1$;
Para e=1 **até** ne_{max} **faça**:
Selecionar aleatoriamente uma granja e atribuir uma carga para a equipe e;
Associar a hora de início do carregamento igual a t;
Enquanto houver cargas não atribuídas **faça:**
$t := t+1$;
Construir a matriz de custo para o horário t, $[c_{ij}]$;
Resolver o problema $PD([c_{ij}])$;
Associar as cargas para as equipes conforme a solução de $PD([c_{ij}])$;
Associar a hora de início do carregamento igual a t;
Fim.

Figura 2. Procedimento de construção

Um procedimento de busca local, denominado *k-swap*, consiste em selecionar k intervalos de tempo consecutivos iniciando no tempo t; para cada equipe são agrupadas as atividades associadas nesse intervalo gerando os *blocos de atividades* denotado por $ba(k,t)$. O *k-swap* realiza a investigação de troca de blocos de atividades desses intervalos entre as escalas das equipes.

A investigação de busca do *k-swap* se dá pela resolução de sucessivos problemas de designação. Cada investigação requer uma matriz de custo $[c_{ij}]$, de dimensão ne_{max}, que representa o custo de associar um bloco de atividades i com a escala da equipe j, ou seja, $c_{i,j} = fa(i,j)$. Nota-se que, diferente da estrutura da matriz do procedimento de construção, agora a matriz de custo trata da relação de custos em associar escalas ou parte das escalas, enquanto que a matriz anterior relacionava o custo de associar cargas as equipes.

Considere ti o índice da hora iniciada mais cedo pelas escalas das equipes e tf o índice da hora finalizada mais tardiamente pelas escalas das equipes. A descrição do algoritmo do *k-swap* é dada pela Figura 3.

Procedimento *k-Swap*
Início
$z^* = \infty$;
Para t= (ti + 1) **até** (tf - k -1) **faça**:
Construir a matriz de custo $[c_{ij}]$ para o bloco $ba(k,t)$;
Resolver o problema $PD([c_{ij}])$;
Se $z<z^*$ **então**
$z^* := z$;
$t^* := t$;
Associar os blocos para as equipes conforme a solução de $PD([c_{ij}])$ de t^*;
Fim.

Figura 3. Procedimento de melhoria *K-Swap*.

A Figura 4 ilustra um exemplo de 1-*swap* para $t = 4$ sobre um grafo representando as atividades a serem realizadas por 4 equipes num período de 7 horas para o carregamento de 18 cargas. Os vértices rotulados representam as cargas, enquanto que os vértices vazios representam horários sem carregamento (descanso ou viagem). As arestas contínuas representam a sequência de atividades previamente definidas, enquanto as arestas tracejadas são possibilidades de trocas que serão investigadas pelo *k-swap*.

A Figura 5 ilustra um exemplo de 2-*swap* para o intervalo de tempo $h = 4$ e 5.

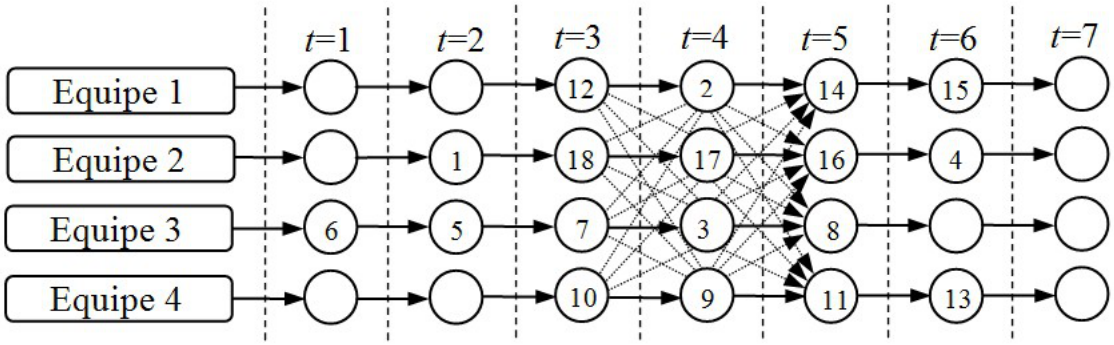

Figura 4. Ilustração de 1-*swap* considerando somente um intervalo de tempo no horário $t = 4$**.**

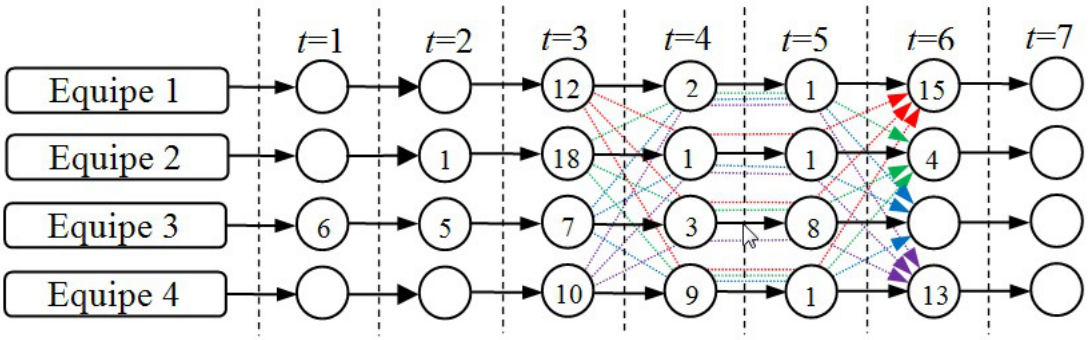

Figura 5. Ilustração de 2-*swap* considerando dois intervalos de tempo consecutivos, $t = 4$ **e** $t = 5$**.**

O segundo procedimento de busca é denominado de *PCR* (Procedimento de Corte e Recombinação) em que são realizados ”cortes”nas escalas entre dois horários de trabalho, separando as sequências de atividades em duas partes. As partes são recombinadas utilizando o PD. A Figura 6 ilustra o procedimento *PCR* para um corte.

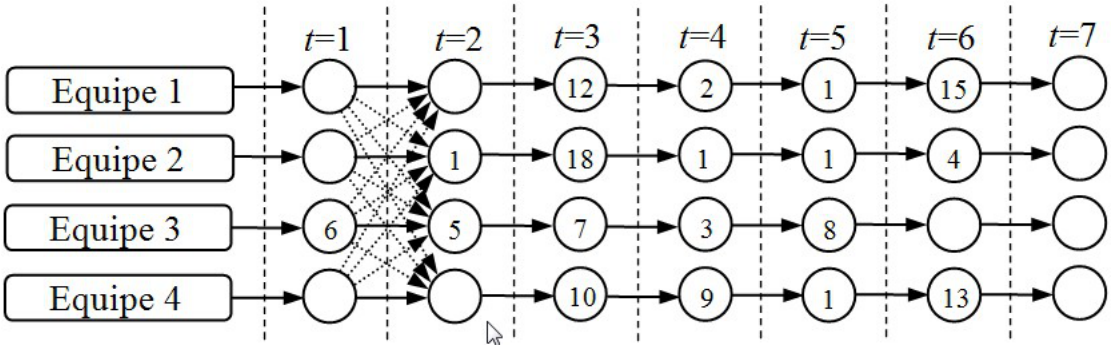

Figura 6. Ilustração do *PCR* sendo aplicado no corte entre o horário $t = 1$ **e** $t = 2$**.**

A matriz de custo $[c_{ij}]$, neste caso, representa o custo de associar a sequência de atividades i antes do corte com a sequência j depois do corte. O algoritmo que descreve PCR é apresentado na Figura 7.

O algoritmo geral de busca local, denominado *bl-VND*, é um algoritmo baseado na metaheurística VND, que explora o espaço de soluções por sistemáticas mudanças da estrutura de vizinhança (Hansen e Mladenovic, 2001). Seja R o conjunto de vizinhanças, $N_1, N_2, ..., N_R$. Se a

Procedimento PCR
Início
$z^* = \infty$;
Para $t = ti$ **até** $(tf - 1)$ **faça**:
Construir a matriz de custo $[c_{ij}]$para o corte entre t e t+1;
Resolver o problema $PD([c_{ij}])$;
Se z<z* **então**
$z^*:=z$;
$t^*:=t$;
Alocar as sequências de atividades conforme a solução de $PD([c_{ij}])$ de t^*;
Fim.

Figura 7. Procedimento de melhoria PCR

solução corrente não é melhorada usando uma particular vizinhança, então uma próxima vizinhança é explorada e assim por diante. Para o algoritmo em questão foram utilizadas $R = 6$ vizinhanças, sendo $N_1 =$ *1-swap* , $N_2 =$ *2-swap*, ..., $N_5 =$ *5-swap* e $N_6 = PCR$.

Em cada iteração do *bl-VND* todas as vizinhanças são exploradas e o algoritmo para quando nenhuma melhoria ocorre em uma das iterações. A avaliação de cada solução é derivada da função 7 que pode ser descrita como:

$$f1(s) = \sum_{i=1}^{ne_{max}} fa(i,j) \mid j \in SC(i), \tag{8}$$

onde s é a solução com as escalas das equipes, $SC(i)$ é o conjunto de atividades da escala da equipe i e $fa(i,j)$ é o custo da atividades j estar na escala da equipe i. A estrutura geral do algoritmo é apresentada na Figura 8. Quando não ocorrer melhoras na solução corrente depois passar por todas as R vizinhanças, então o algoritmo para (*critério de parada*).

Procedimento *bl-VND*;
Inicialização: Selecione um conjunto de estrutura de vizinhança N_r (r = 1, ..., 6); selecione uma solução inicial s;
Início
Repetir até que o critério de parada seja satisfeito:
(1) $r \leftarrow 1$;
(2) **Repetir** até que $r = R$:
(a) Busca local: Encontrar uma solução s' de s ($s' \in N_r(s)$);
(b) **Se** $f1(s') < f1(s)$ **então** $s \leftarrow s'$ e $r \leftarrow 1$; **senão** $r \leftarrow r+1$;
Fim.

Figura 8. Procedimento geral de busca local utilizando todos os procedimentos de melhoria anteriores.

4.3. Busca Local baseada em Fila

Os procedimentos construtivos e de busca local apresentados têm como foco a minimização dos custos de horas pagas. Portanto, são avaliadas apenas as horas trabalhadas pelas equipes sem considerar o impacto no funcionamento de cada fábrica no que diz respeito ao tempo de espera das cargas no pátio das fábricas e tampouco a hora ociosa das fábricas. Também, com o intuito de considerar esse custo, foi introduzida uma segunda função para avalizar uma solução definida como:

$$f2(s) = tq + po, \tag{9}$$

onde tq é o tempo médio na fila de espera das fábricas e po é uma penalidade caso ocorra alguma ociosidade das fábricas, neste caso foi penalizada a ocorrência de ociosidade da fábrica.

Visando a minimização do tempo de espera no pátio de cada fábrica e a ociosidade das mesmas, foi projetado um segundo algoritmo de busca local, denominado *bl-Queue*, que realiza mudanças na hora de início das escalas de cada equipe. A hora de início da escala para cada equipe é adiantada ou atrasada de tal maneira que o custo $f2$ seja minimizado. Isso pode ser feito avaliando o impacto de cada escala iniciando no tempo $t, t = 1, ..., t_{max}$, sendo t_{max} o *tempo máximo* permitido para que a última carga da escala chegue na fábrica em tempo para descarregamento e abate. Esse processo é repetido sequencialmente para cada equipe. Esse procedimento é apresentado pela Figura 9.

Procedimento *bl-Queue*
Início
  Para *e*=1 **até** *ne* **faça**:
    Encontrar um tempo *t* para início da escala da equipe *e* de tal maneira que $f2$ seja mínimo;
    Mudar a hora de início da escala da equipe *e* para *t*;
Fim.

Figura 9. Procedimento de busca local baseado na informação da fila de espera na fábrica.

Para calcular o valor de tq foi utilizada uma técnica de simulação discreta conhecida como método das 3-fases (CHWIF; MEDINA, 2007). O método considera dois tipos de eventos: condicional e programável. O evento programável é aquele que ocorre no tempo que estava sendo agendado para acontecer, mas o evento condicional é o tipo de evento que só ocorre quando as condições que ele requer forem satisfeitas. Todo evento condicional passa por uma fila de espera na qual permanece até que as condições sejam satisfeitas a fim de que seja executado (por exemplo, se a fábrica já iniciou o funcionamento ou se já é possível fazer o descarregamento de algum caminhão). Uma agenda é utilizada para colocar em ordem de ocorrência os eventos programáveis como também para atualizar o relógio da simulação. O método se resume nas três fases a seguir:

1. Checar o tempo de todos os eventos da agenda e selecionar o que ocorre primeiro. Atualizar o relógio da simulação para o tempo do evento selecionado.
2. O evento selecionado é executado e as entidades (cargas) são movidas para as respectivas filas de espera para a próxima atividade.
3. Pesquisar as entidades que estão na fila de espera e iniciar os eventos que satisfazem as condições do momento. Mover as entidades das filas para a atividade e calcular o tempo que a entidade permaneceu na fila.

Note que as próprias cargas, definidas anteriormente, podem ser interpretadas como as entidades que praticaram os eventos da simulação. Uma vez que se sabe a hora que as equipes iniciam suas escalas de trabalho, é possível saber que horas as cargas chegarão à fábrica de destino. Essa informação é utilizada para calcular o tempo que cada entidade (carga) permaneceu na fila de espera. Ao final da simulação se obtém as estatísticas das filas, como por exemplo o tempo de permanência e o tamanho médio das filas.

4.4. O Algoritmo Memético

Nesta investigação foi proposto um algoritmo memético (MOSCATO, 1999) que combina as ideias de algoritmos evolucionários e busca com busca local, ou seja, combinam as ideias de

diversificação e de intensificação da busca. O problema investigado é altamente restrito e o uso de algoritmos evolutivos tem sido indicado para essas situações (PODGORELEC; KOKOL, 1997).

O algoritmo é dividido em duas fases executadas consecutivamente, sendo que cada fase compreende um algoritmo genético com um tipo de busca local diferente. Na fase 1 o algoritmo procura minimizar as horas pagas, enquanto que fase 2 tenta minimizar o tempo de espera na fábrica. Portanto, a primeira fase utiliza uma *função de avaliação* $f1$ (equação 8) baseada em horas pagas, enquanto que a segunda fase utiliza $f1 + f2$ (equações 8 e 9 respectivamente). Antes de iniciar a execução das fases 1 e 2, um modelo de programação linear inteira é resolvido para identificar a designação das cargas as fábricas minimizando custos de transporte e respeitando a capacidade de processamento de cada fábrica.

Cada *indivíduo* s (solução do problema) é codificado por um cromossomo que representa o conjunto de cargas, implementada em uma lista. O objetivo do algoritmo é encontrar a hora de início de carregamento de cada carga e atribuí-la a uma equipe.

A *população inicial* é gerada aleatoriamente com auxílio do procedimento construção apresentado na seção 4.1. A população é atualizada com base na estratégia *steady state* (algumas vezes denominada de estacionária ou substituição incremental) para cada indivíduo gerado. O indivíduo substituirá aleatoriamente um membro da população com uma avaliação abaixo da média. Cada indivíduo gerado significa uma *iteração* do algoritmo. O *número máximo de iterações* é definido pelo parâmetro MaxIt.

A *seleção* dos indivíduos para reprodução é realizada pela técnica do torneio, selecionando dois indivíduos aleatoriamente e escolhendo o mais apto com base na função de avaliação de cada fase.

O operador de *cruzamento* seleciona dois indivíduos e realiza a recombinação dos dois indivíduos utilizando a técnica do cruzamento uniforme das cargas e as designações das cargas as equipes são realizadas com o procedimento de construção. Os indivíduos selecionados (pais) sempre geram um único indivíduo (filho).

A *mutação* é aplicada segundo uma taxa a ser definida como parâmetro do algoritmo. A mutação realiza a mudança da hora de início de duas cargas selecionadas aleatoriamente.

Na primeira fase do algoritmo os indivíduos são avaliados com base nos custos das horas pagas, enquanto que na segunda fase avalia-se o tamanho das filas nas fábricas utilizando uma técnica de simulação discreta. Assim, após cada reprodução e mutação é aplicado o algoritmo de busca local *bl-VND* na fase 1 e *bl-Queue* na fase 2.

O *critério de parada* é um parâmetro a ser definido em termos de *número de iterações*. Quando o critério de parada é alcançado o algoritmo devolve o melhor indivíduo da população. O número de iterações é utilizado como critério de parada para cada fase, porém a fase 2 utiliza apenas um percentual Ω do número de iterações da fase 1.

Cada fase do algoritmo segue os padrões clássicos dos algoritmos genéticos, mas a população é inicializada na fase 1 enquanto que a fase 2 continua operando sobre a população resultante da fase 1.

5. Resultados Computacionais

Foram realizados extensivos testes computacionais e diferentes parâmetros foram testados de forma empírica. Para os resultados computacionais apresentados nesta seção, os parâmetros finais utilizados são descritos na Tabela 1.

Os experimentos foram realizados num computador com CPU Intel I3, 2.26 GHz, com Windows 7 64 bits, usando a linguagem *Object Pascal* com compilador Freepascal. Foram utilizadas escalas reais de uma empresa obtidas ao longo de um período de 3 meses de operação, totalizando 62 dias (escalas). Envolveram um total de 112 catadores divididos em equipes de 14 pessoas (incluindo o motorista). A Tabela 2 resume os resultados computacionais obtidos com o algoritmo e compara os valores obtidos com as escalas produzidas manualmente pela empresa (coluna Manual).

Tabela 1. Parâmetros do algoritmo proposto

Parâmetro	Valor
Tamanho da população	80
Taxa de mutação	3%
Número de iterações - MaxIt	1000
Ω	50%
ph	100
po	10.000

Tabela 2. Resultados computacionais com o algoritmo proposto.

	Manual	Algoritmo	Redução	% de redução
Total de horas pagas (horas)	4.350,48	3.936,17	414,32	9,52%
Total de distância percorrida (Km)	428.787,00	405.195,80	23.591,20	5,50%
Total de horas da fábrica parada (horas)	45,93	0,00	45,93	100,00%
Número médio de caminhões no pátio	4,73	2,06	2,66	56,35%
Tempo médio de espera no pátio (minutos)	124,54	4,59	119,94	99,31%
Sobrecarga de caminhões no pátio[2]	93	0	93	100,00%
Número de escalas inviáveis	44	0	44	100,00%

[2]Quantidade de vezes que o limite de caminhões no pátio da fábrica foi ultrapassado.

Os resultados mostram uma redução significativa dos custos operacionais. Os custos com horas pagas são relativas às equipes, isto significa que o custo reduzido deve ser multiplicado pelo número de pessoas envolvidas, neste caso 14 pessoas. Do ponto de vista qualitativo foram observadas algumas eliminações de escalas indesejadas, como por exemplo escalas com extensos períodos de trabalho sem um intervalo de descanso.

O tempo total de experimentação foi de aproximadamente 100 minutos para executar todas as instâncias do problema. Algumas instâncias consumiram aproximadamente 10 minutos para obtenção de uma solução.

6. Discussão e Conclusões

Neste trabalho foi investigado um algoritmo heurístico baseado em algoritmos meméticos aplicado a um problema de escalonamento de atividades na agroindústria.

Os experimentos foram realizados com dados reais de dezenas de casos reais. Os resultados mostram que, em média, o método possibilitou a redução aproximada de 10% dos custos com horas pagas, 5,5% dos custos de transporte, 99% do tempo de espera no pátio do abatedouro e eliminou os casos de ociosidade da fábrica. Além disso, foram observadas outras melhorias nas escalas das equipes do ponto de vista qualitativo.

A utilização da abordagem baseada em algoritmos meméticos mostrou-se bastante adequada para o problema investigado, tendo em vista a flexibilidade que a técnica proporciona tanto em termos de operadores genéticos como a possibilidade de integração com métodos. Notou-se que os algoritmos de busca local tiveram um papel fundamental na melhoria dos resultados.

Como investigações futuras pretende-se utilizar a abordagem geracional, juntamente com elitismo e utilizar uma taxa de mutação variável, conforme proposto por Beasley e Chu (1996). Pesquisar, também, por novos procedimentos de busca local, novos operadores genéticos e por uma abordagem multi-objetivo.

7. Agradecimentos

Os autores agradecem a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro ao projeto.

Referências

BANSAL, S.; GOWTHAM, K.; HOTA, C. Novel adaptive scheduling algorithm for computational grid. In: *Proceedings of the 3rd IEEE international conference on Internet multimedia services architecture and applications*. Piscataway, NJ, USA: IEEE Press, 2009. (IMSAA'09), p. 308–312.

BEASLEY, J. E.; CHU, P. C. A genetic algorithm for the set covering problem. *European Journal of Operational Research*, v. 94, n. 2, p. 392 – 404, 1996.

BLAZEWICZ, J. *Scheduling in Computer and Manufacturing Systems*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1996.

BODIN, L. Solving large vehicle routing and scheduling problems in small core. In: *Proceedings of the 1983 annual conference on Computers : Extending the human resource*. New York, NY, USA: ACM, 1983. (ACM '83), p. 27–37.

CHWIF, L.; MEDINA, A. C. *Modelagem e Simulo de Eventos Discretos: Teoria e Aplicaes*. So Paulo, SP, Brasil: Editora dos Autores, 2007.

HANSEN, P.; MLADENOVIC, N. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, v. 130, n. 3, p. 449–467, May 2001.

HART, E.; ROSS, P.; NELSON, J. Scheduling chicken catching - an investigationinto the success of a genetic algorithm on areal-world scheduling problem. *Annals of Operations Research*, Springer Netherlands, v. 92, p. 363–380, 1999.

MOSCATO, P. *Memetic algorithms: a short introduction*. Maidenhead, UK, England: McGraw-Hill Ltd., UK, 1999. 219–234 p.

PODGORELEC, V.; KOKOL, P. Genetic algorithm based system for patient scheduling in highly constrained situations. *J. Med. Syst.*, v. 21, December 1997.

SHEEN, G.-J.; LIAO, L.-W. Scheduling machine-dependent jobs to minimize lateness on machines with identical speed under availability constraints. *Comput. Oper. Res.*, v. 34, August 2007.

SONMEZ, O. O.; GURSOY, A. A novel economic-based scheduling heuristic for computational grids. *Int. J. High Perform. Comput. Appl.*, Sage Publications, Inc., Thousand Oaks, CA, USA, v. 21, p. 21–29, February 2007.

YUNES, T. H.; MOURA, A. V.; SOUZA, C. C. de. Solving very large crew scheduling problems to optimality. In: *Proceedings of the 2000 ACM symposium on Applied computing - Volume 1*. New York, NY, USA: ACM, 2000. (SAC '00), p. 446–451.