Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXIX, págs. 9091-9105 (artículo completo)
Eduardo Dvorkin, Marcela Goldschmit, Mario Storti (Eds.)
Buenos Aires, Argentina, 15-18 Noviembre 2010

UMA APLICAÇÃO DO *SHIFT DESIGN PROBLEM* À CRIAÇÃO, VIA METAHEURÍSTICAS, DOS HORÁRIOS DE TRABALHO DOS FUNCIONÁRIOS DE *CALL CENTERS* COM INTERVALOS DE PAUSAS E DIAS DE DESCANSO

Cynthia da S. Barbosa, Sérgio R. de Souza, Gray F. Moita

Centro Federal de Educação Tecnológica de Minas Gerais
Av. Amazonas, 7675, 30.510-000, Belo Horizonte, MG, Brasil
cysb@terra.com.br, sergio@dppg.cefetmg.br, gray@dppg.cefetmg.br

Palavras-Chaves: *Shift Design Problem, Call Center*, Metaheurística, *Iterated Local Search,* Método AjustaCoordenadas.

Resumo. Este trabalho apresenta a aplicação da Metaheurística *Iterated Local Search* (ILS) e do Método AjustaCoordenadas à solução do *Shift Design Problem* (SDP) aplicado à criação de turnos de trabalho de uma empresa de *Call Center* com intervalos de pausas e dias de descanso dos funcionários. O objetivo deste trabalho é determinar um conjunto de soluções factíveis que contenham turnos e o número de funcionários por turno que minimizem o excesso e a escassez de funcionários por turno, e as diferenças do número médio de tarefas executadas por funcionários, por semana, incluindo e os intervalos de pausas dos funcionários e os dias de descanso, respeitando as restrições das leis trabalhistas. A alocação definitiva dos funcionários aos respectivos turnos de trabalho somente é considerada após as mudanças geradas nos turnos de trabalho através de métodos de busca local. O problema em tela pertence à classe dos problemas NP - difíceis, possuindo grande aplicação de cunho econômico, como o planejamento de escalas de funcionários de hospitais e enfermeiros, o planejamento das escalas de trabalho de empresas de transportes urbanos, ferroviários e aviação, no planejamento de professores, dentre outros. O uso de técnicas heurísticas se justifica pela elevada dimensão do problema em relação à quantidade de variáveis e restrições. A criação dos turnos de trabalho é feita em conjunto com as folgas dos funcionários e os intervalos de pausa durante a jornada de trabalho. Neste trabalho, o método de busca local utilizado para a geração da solução inicial para a metaheurística ILS é o Método da Descida, que apresenta baixo custo computacional na implementação realizada. O Método AjustaCoordenadas foi desenvolvido e aplicado a este trabalho, com o objetivo de perturbar todos as posições do vetor de uma solução, através de ajustes feitos em cada elemento da solução corrente, na obtenção da solução ótima. Para testar a eficiência dos algoritmos propostos, foram feitos testes computacionais utilizando instâncias reais de um *Call Center*, pois não foram encontradas instâncias disponíveis na literatura para testes. Os resultados encontrados são comparados com resultados obtidos pela aplicação da metaheurística *Iterated Local Search* (ILS) e o Método AjustaCoordenadas. Os resultados mostram que os métodos propostos são capazes de gerar soluções viáveis, tanto na qualidade da solução final, quanto na rapidez.

Copyright © 2010 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

1 INTRODUÇÃO

A criação de horários de trabalho de funcionários é uma das etapas do processo de planejamento de uma empresa de *Call Center,* segundo Bhulai et al. (2007). É nesta etapa que são gerados os turnos de trabalho para a designação dos funcionários incluindo os intervalos de pausas por funcionário.

O presente artigo estuda o problema de planejamento de turnos de trabalho dos funcionários de uma empresa de *Call Center*, considerando os intervalos de descanso (pausas de trabalho e dias de descanso) dos funcionários. O problema em questão é tratado através de uma reformulação do *Shift Design Problem* (SDP), por apresentar flexibilidade no planejamento dos funcionários de uma empresa.

Na literatura, o *Shift Design Problem* (SDP) é introduzido por Musliu et al. (2004) e considera a alocação real dos funcionários após as mudanças geradas nos turnos de trabalho. O SDP consiste em determinar um conjunto de soluções factíveis que contenham turnos e o número de funcionários por turno que minimizem o número de turnos distintos, o excesso e a escassez de funcionários, e as diferenças do número médio de tarefas executadas por funcionários, por semana. Segundo os autores, o problema de minimização do número de turnos é um problema NP - difícil e a definição da estrutura de vizinhança adequada é uma das características mais importantes das técnicas de busca local. Por outro lado, resolvendo-se o problema global das várias fases distintas, a solução do SDP torna-se mais fácil de ser obtida.

Em Musliu et al. (2008), os autores consideram os intervalos de descanso (pausas) para a geração dos turnos de trabalho dos funcionários. O grande desafio para esta classe de problema é satisfazer os requisitos legais e garantir que certo número de funcionários esteja presente em todo o turno. A quantidade de pausas em um turno deve ser programada de tal forma que o número de violações e restrições em relação às pausas e o excesso ou a escassez dos funcionários sejam minimizadas.

Em Tellier e White (2006) e Canon (2007), os autores apresentam o problema de planejamento de escalas de trabalho em *Call Centers* através do planejamento de pausas, com dias de descanso e férias dos funcionários. O problema é resolvido utilizando-se técnicas de busca local para encontrar soluções com qualidade aceitável, baseados em heurística para refinar a solução final.

Este trabalho trata o SDP para a criação dos turnos de trabalho dos funcionários de uma empresa de *Call Center* após as trocas de horários nos turnos de trabalho, a fim de obter soluções que contenham a quantidade de funcionários por turno e que minimize a quantidade de turnos distintos e a quantidade de funcionários por turno, de acordo com as leis trabalhistas brasileiras. Para a solução deste problema, foi utilizado o Método da Descida, uma técnica de busca local para a geração da solução inicial por apresentar baixo custo computacional na implementação realizada. A metaheurística utilizada é o *Iterated Local Search* (ILS), uma metaheurística de busca local de fácil implementação. Foi desenvolvido o Método AjustaCoordenadas e aplicado a este trabalho, com o objetivo de perturbar todas as posições do vetor de uma solução, através de ajustes feitos em cada elemento da solução corrente, na obtenção da solução ótima.

Este trabalho está organizado como segue. Na seção 2, são descritas as características do problema estudado e a formulação matemática do problema. Na seção 3, é detalhado o método da descida, enquanto na seção 4, é descrito o método *ILS*. Na seção 5 é descrito o método AjustaCoordenadas. Na seção 6, são apresentadas a metodologia adotada e a aplicação dos métodos da descida, *ILS* e AjustaCoordenadas. Na seção 7 são apresentados e

Copyright © 2010 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

discutidos os resultados computacionais. A seção 8 conclui o trabalho.

2 DESCRIÇÃO DO PROBLEMA ESTUDADO

O problema estudado neste trabalho é o *Shift Design Problem* (SDP) aplicado à criação dos horários de trabalho dos funcionários de um *Call Center* considerando os intervalos de descanso (pausas) e os dias de folga dos funcionários. A solução para este problema consiste em encontrar a quantidade de funcionários por turno de trabalho, com os intervalos de pausas por funcionários, de modo a minimizar a quantidade de funcionários por turno, de acordo com as leis trabalhistas brasileiras, satisfazendo os requisitos legais e garantindo que o devido número de funcionários esteja presente em todo o turno.

Neste trabalho, os requisitos das tarefas de trabalho por um determinado período de tempo, juntamente com as restrições sobre os possíveis horários de início e da duração dos turnos e a quantidade média de ligações atendidas por funcionário por semana são conhecidos.

2.1 Formulação matemática

A formulação matemática do SDP adotada neste artigo foi adaptada do trabalho de Musliu et al. (2004) e é descrita a seguir. Considere, então, que:

- n representa o número de intervalos de tempo consecutivos ($[a_1,a_2]$, $[a_2,a_3]$, ..., $[a_n, a_{n+1}]$) , todos com o mesmo comprimento, e representados em minutos. Cada intervalo $[a_i, ai_{+1}]$ está relacionado com o número w_i (quantidade de funcionários) indicando a quantidade ideal de funcionários para um determinado intervalo de tempo. Cada intervalo $[a_i, ai_{+1}]$ possui uma duração de 15 minutos para um melhor planejamento dos intervalos de pausa. O instante de tempo a_1 representa o início do turno de trabalho e o instante de tempo a_{n+1} representa o fim do turno de trabalho. Por exemplo, um funcionário com a jornada de trabalho de 6 horas, inicia o turno no intervalo de tempo a_1 - às 06:00 horas da manhã, e o finaliza no intervalo de tempo a_{n+1} , ou seja, às 12:00 horas. Assim a jornada de trabalho de um funcionário representa 24 intervalos de tempo de 15 minutos cada;
- y representa o tipo do turno $v_1,..., v_y$, conforme apresentados na Tab. 1. Cada tipo de turno v_j possui os seguintes parâmetros:

  - v_j início-min e v_j início-max representam a faixa de tempo para o início-mínimo e o início-máximo em que o turno poderá iniciar;

  - v_j min-comp e v_j max-comp representam o comprimento mínimo e o comprimento máximo do turno de trabalho.

Neste trabalho, considera-se que a duração máxima do turno não poderá ultrapassar 6 horas de trabalho por dia. Além disso, considera-se que o planejamento é feito para uma semana.

Tipo Turno	Início-Min	Início-Max	Min-Comp	Max-Comp
Manhã	06:00	09:00	06:00	08:00
Tarde	12:00	15:00	06:00	08:00
Noite	18:00	21:00	06:00	08:00

Tabela 1 – Tipos de turnos

O objetivo é minimizar os quatro componentes abaixo:

- F_1: soma dos excessos de funcionários em cada intervalo de tempo durante o período de planejamento.
- F_2: soma da escassez de trabalhadores em cada intervalo de tempo durante o período

Copyright © 2010 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

de planejamento.

- F_3: número de turnos k.
- F_4: média da carga de trabalho por semana, caso esteja acima do limite.

Em Musliu et al. (2004) este problema é posto como um problema de otimização multicritérios. Os critérios possuem importâncias diferentes, dependendo da situação. A função objetivo é a soma ponderada dos quatro componentes citados, nos quais os pesos dependem dos dados da instância.

A carga de trabalho l_d para um determinado intervalo de tempo d, para definir o excesso e a escassez de trabalhadores, representa a quantidade de turnos de trabalho que um funcionário poderá trabalhar por semana, é dada pela Eq. (1):

$$l_d = \sum_{p=1}^{k} X_{p,d} \tag{1}$$

sendo:

$$X_{p,d} = \begin{cases} s_p w_i & \text{se o intervalo de tempo } d \text{ pertence ao turno } s_p \text{ no dia } i; \\ 0 & \text{outros} \end{cases}$$

A Eq. (1) garante que o número de funcionários trabalhando em um intervalo de tempo d não pode ser inferior à quantidade necessária em um turno de trabalho.

O excesso F_1 representado pela Eq. (2) e a escassez F_2 representado pela Eq. (3) (ambas em minutos) dos funcionários em todos os intervalos de tempo durante o período de planejamento são definidos como:

$$F_1 = \sum_{d=1}^{n} (\max(l_d - wd, 0) duracaoturno) \tag{2}$$

$$F_2 = \sum_{d=1}^{n} (\max(wd - l_d, 0) duracaoturno) \tag{3}$$

sendo wd a quantidade de funcionários em cada intervalo de tempo.

A penalidade associada à média da carga de trabalho, referente ao número de turnos que um funcionário poderá trabalhar por semana, é definida pela Eq. (4):

$$F_4 = \max(AvD - AS, 0) \tag{4}$$

para:

- AvD: média do número de turnos de trabalho por semana por funcionário.
- AS: limite superior para a média do número de turnos de trabalho por semana por funcionário.

A penalidade F_4 não é utilizada no presente trabalho, pois o limite máximo da média do número de turnos de trabalho é 6 horas por dia e a média por semana é sempre um; logo, a penalidade F_4 será sempre zero.

Assim, a função objetivo a ser minimizada e aplicada a este trabalho é definida como:

$$FO = \alpha F_1 + \alpha F_2 + \alpha F_3 \tag{5}$$

sendo:

- F_1: representa o excesso de funcionários em um determinado intervalo de tempo
- F_2: representa a escassez de funcionários em um determinado intervalo de tempo
- F_3: representa o número de turnos k
- α: representa o fator de ponderação, dependente das instâncias utilizadas.

Copyright © 2010 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

2.2 Planejamento dos dias de folgas dos funcionários

Os dias de descanso dos funcionários para o problema em tela serão sábados e domingos, em função do volume da demanda ser menor para estes dias. Um funcionário tem uma jornada de trabalho de 36 horas semanais, trabalhando 6 horas por dia. Assim, em certa semana, um funcionário trabalhará de segunda a sábado e folgará no domingo e, na próxima semana, o funcionário trabalhará de segunda a domingo, folgando no sábado. Para o sábado, serão alocados 60% dos funcionários e, para o domingo, 40%, devido ao fato do sábado ter uma demanda maior em relação ao domingo.

2.3 Modelo de pausas

O modelo aplicado à criação dos turnos de trabalho dos funcionários de um *Call Center* contemplando os intervalos de pausas, adotado neste artigo, foi adaptada dos trabalhos de Musliu et al. (2008) e Tellier e White (2006) e é descrita a seguir. A quantidade de pausas em um turno deve ser programada de tal forma que o número de restrições em relação às pausas e o excesso ou a escassez dos funcionários sejam minimizadas. Considere, então, que:

- n representa o número de intervalos de tempo consecutivos ($[a_1,a_2]$, $[a_2,a_3]$, ..., $[a_n, a_{n+1}]$) , todos com o mesmo comprimento, e representados em minutos. Cada intervalo $[a_i, ai_{+1}]$ possui uma duração de 15 minutos. O instante de tempo a_1 representa o início do turno de trabalho e o instante de tempo a_{n+1} representa o fim do turno de trabalho;
- $t_1, t_2,...t_n$ representam os funcionários que trabalham em cada turno, possuindo parâmetros de início $t_i.inicio$ e de duração $t_i.duração$ do turno.
- Tipos pausas: os tipos de pausas aplicadas a este trabalho são a pausa lanche com uma duração de 15 minutos e a pausa banheiro de 5 minutos. Os intervalos de pausas p são caracterizados pelos parâmetros $p_i.início$ e $p_i.duração$.

Para garantir a quantidade mínima necessária de funcionários em cada intervalo de tempo (15 minutos, é verificado se existe excesso ou escassez de funcionários no intervalo de tempo. Se $t_i - p_i$ for negativo, ou seja, se a quantidade de pessoas trabalhando em certo intervalo de tempo for menor que a quantidade de pessoas que estarão em pausa, há escassez de funcionários e será necessário procurar no vetor o próximo intervalo de tempo com excesso de funcionários para o planejamento das pausas. Se t_i - p_i for positivo, significa que há excesso de funcionários no intervalo de tempo, podendo assim planejar os funcionários para a pausa.

A penalidade aplicada a este trabalho para garantir a quantidade mínima necessária de funcionários em cada intervalo de tempo, segundo Tellier e White (2006), é calculada como:

$$P(V) = \sum_i (t_i - p_i)^2 \tag{6}$$

sendo V o vetor de intervalos de tempo.

3 MÉTODO DA DESCIDA

É um método de busca local que analisa todos os possíveis vizinhos de uma solução s em sua vizinhança $N(s)$, escolhendo, a cada passo, aquele que tem menor valor para a função de avaliação. Neste sentido, trata-se de um método guloso. É importante observar, que para efetivar a mudança, o vizinho candidato deve melhorar estritamente o valor da melhor solução obtida até o momento. O critério de parada se dá quando um mínimo local é encontrado. O

Copyright © 2010 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

mínimo local é a solução *s* em que nenhum de seus vizinhos $s' \in N(s)$ tem o valor de função de avaliação menor.

Entende-se por vizinho de uma solução *s* alcançada aplicando-se uma transformação em *s*. Representa-se essa operação por $s' \leftarrow s$. Uma solução s' faz parte da vizinhança da solução *s* se, e somente se, s' é resultado de uma mudança em *s*, causada por um determinado movimento *m*, de tal maneira que continue a fazer parte do conjunto de soluções possíveis.

4 *ITERATED LOCAL SEARCH*

O método *Iterated Local Search* (ILS), apresentado em Lourenço et al. (2003), é baseado na idéia de que um procedimento de busca local pode ser melhorado, gerando-se novas soluções de partida, as quais são obtidas por meio de perturbações na solução ótima local. A perturbação precisa ser suficientemente forte para permitir que a busca local explore diferentes soluções, mas também fraca o suficiente para evitar um reinício aleatório.

Para aplicar um algoritmo *ILS*, quatro componentes têm que ser especificados, segundo Lourenço et al. (2003):

- *Procedimento GeraSolucaoInicial()*, que gera uma solução inicial *so* para o problema;
- *Procedimento BuscaLocal,* que retorna uma solução melhorada *s''*;
- *ProcedimentoPerturbacao*, que modifica a solução corrente *s* guiando a uma solução intermediária *s'* e;
- *Procedimento CriterioAceitacao*, que decide de qual solução a próxima perturbação será aplicada.

5 MÉTODO AJUSTACOORDENADAS

O método AjustaCoordenadas desenvolvido e aplicado a este trabalho, tem o objetivo de perturbar todos os elementos de um vetor, através de ajustes feitos em cada elemento (coordenada) do vetor corrente, a fim de obter uma solução ótima. Os ajustes são feitos somando ou subtraindo uma unidade no primeiro elemento do vetor da solução corrente. Deste modo, calcula-se o valor da função objetivo para a solução na qual foi realizado o ajuste. Se o valor da função objetivo após o ajuste apresentar melhores resultados, então o valor é armazenado e o procedimento é repetido até que não se obtenha melhores resultados na função objetivo. O ajuste é realizado em todos os elementos do vetor. A Fig. 1 apresenta os ajustes feitos para o método AjustaCoordenadas.

Procedimento *Método Ajusta Coordenadas()*
1. decrementa uma unidade na primeira coordenada do vetor e calcula-se a função objetivo até que não se obtenha melhores resultados.
2. incrementa uma unidade na primeira coordenada do vetor e calcula-se a função objetivo até que não se obtenha melhores resultados.
3. repita os passos 1 e 2 para a próxima coordenada.

fim *Método Ajusta Coordenadas*;

Figura 1: Pseudocódigo do método AjustaCoordenadas

6 METODOLOGIA

6.1 Representação de uma solução

Uma solução inicial *s* para o SDP é gerada através de um método guloso, definindo a

Copyright © 2010 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

quantidade necessária de funcionários para cada intervalo de tempo, para o atendimento da demanda a cada dia da semana. O algoritmo guloso escolhe uma solução adequada uma por vez, fazendo uma escolha ótima local.

Para explorar o espaço de soluções do problema, são aplicados dois tipos diferentes de movimentos, para definir as estruturas de vizinhança:

- Movimento da quantidade de funcionários: nesse movimento, a vizinhança da solução é obtida alterando-se a quantidade de funcionários em um determinado intervalo de tempo, acrescida ou decrescida de uma unidade, retornando-se como vizinho aquela solução que apresentar o melhor valor da função objetivo.
- Movimento de início do turno: nesse movimento, a vizinhança da solução é obtida alterando o início do turno, acrescida ou decrescida de um intervalo de tempo (30 minutos), retornando-se como vizinho aquela solução que apresentar o melhor valor da função objetivo.

6.2 Método Da Descida Aplicado Ao SDP

Seja *s* uma solução do problema e seja uma solução *s'* pertencente a uma vizinhança de *s*, definida pela quantidade necessária de funcionários por turno. Assim, *s'* é gerada a partir do movimento *m* realizado em *s*. Um movimento *m* em *s* é definido como acrescentar ou diminuir a quantidade de funcionários em um determinado intervalo de tempo. Por exemplo, para o turmo que se inicia às 06:30hs são necessários 22 funcionários para o atendimento da demanda. Assim, move-se um funcionário para o turno anterior (06:00 hs) e outro funcionário para o turno posterior (07:00 hs). Esta forma de seleção de funcionários aplica-se também aos movimentos do início dos turnos de trabalho. Estes movimentos implicam em uma chance maior da solução corrente se tornar viável. O critério de parada consiste no número máximo de iterações sem melhora.

O método da descida aplicado a este trabalho realiza sempre a melhor troca de posições no vetor de entrada que contém a quantidade de funcionários. Todas as possíveis trocas são avaliadas, mas somente a melhor é realizada para todos os dias da semana. A função ainda retorna o menor valor entre os melhores índices (**iMelhor** e **jMelhor**), ou seja, os índices que participaram da melhor troca.

A função recebe como entrada os seguintes parâmetros:

- matriz s: matriz que armazena sempre a melhor solução encontrada,
- int nSlots: quantidade de intervalos de tempo a cada 30 minutos,
- int slotsPessoa: quantidade de intervalos de tempo que uma pessoa ocupa de forma contínua, ou seja, o comprimento do turno de trabalho.

A Fig. 2 apresenta o pseudocódigo do método da descida aplicado ao SDP.

Algoritmo *MetodoDescida*()

```
1.    int iMelhor
2.    int jMelhor
3.    para i ← 1 até (nSlots - slotsPessoa) faça
4.          para j ← (i+1) até (nSlots – slotsPessoa + 1) faça
5.                para cada dia da semana considera faça
6.                      troca de posições i e j
7.                fim para
8.                avalia se a função objetivo melhorou
9.                se melhorou, armazena as posições em iMelhor e jMelhor
```

Copyright © 2010 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

```
10.                    desfaz a troca
11.              fim para
12.       fim para
13.       para cada dia da semana considera faça
14.                faz a troca entre as posições iMelhor e jMelhor
15.       fim para
16.       retorne iMelhor
fim MetodoDescida;
```

Figura 2: Pseudocódigo do método da descida aplicado ao SDP

6.3 *ILS* aplicado ao SDP

Para resolver o problema proposto, o método *ILS* foi adaptado da seguinte forma: como método de busca local utilizou-se o método da descida descrito na seção anterior, recebendo o valor do menor índice referente a melhor troca. Este índice é armazenado na variável **iMelhor**. Logo após, são realizados dois tipos de pertubação:

- Perturbação 1: neste movimento, um intervalo de tempo é selecionado e o horário de início do turno é acrescido ou decrescido de uma unidade, retornando-se como vizinho aquele que apresentar o melhor valor da função objetivo atual.
- Pertubação 2: neste movimento, a quantidade de funcionários por intervalo de tempo é acrescida ou decrescida de uma unidade, retornando-se como vizinho aquele que apresentar o melhor valor da função objetivo atual.

Portanto, essas perturbações consistem em aumentar ou diminuir a quantidade de funcionários e o horário de início do turno, em um intervalo de tempo. O critério de aceitação define que uma solução gerada pelo método de busca local é aceita, isto é, $s \leftarrow s'$, se s' apresentar valor da função objetivo menor que a da melhor solução s encontrada até o momento, isto é, se $f(s') < f(s)$. Caso a função objetivo apresente a melhor solução, são armazenados os melhores resultados e o processo é reiniciado para o nível de perturbação igual a 1, como uma nova posição de referência definido pela execução do *metodoDescida* com a nova solução. O segundo nível de pertubação consiste em realizar dois movimentos, ou seja, são trocados dois funcionários, e assim por diante. A cada iteração sem melhora, o nível de perturbação é modificado de acordo com o seguinte esquema:

- Nível 1 de perturbação: consiste em realizar um único movimento.
- Nível 2 de perturbação: são realizados dois movimentos, e assim sucessivamente, até o nível máximo definido.

Estas perturbações são realizadas pela função *perturbacaoLocal* descrita a seguir.

A função *perturbacaoLocal* gera perturbações de uma posição específica referente a quantidade de funcionários em um intervalo de tempo. As perturbações têm como referência, a posição calculada a partir das variáveis **iMelhor** e **distPertub**. São feitas as alterações para mais e para menos começando com o acréscimo (ou decréscimo) de 1 e indo até a quantidade especificada em **pertubMax**. A função recebe como entrada os seguintes parâmetros:

- matriz s - matriz que armazena sempre a melhor solução encontrada,
- matriz s' – matriz que realiza cálculos,
- int distPertub – distância a ser pertubada de acordo com o resultado armazenado em **iMelhor**,
- int pertubMax – limite superior para a quantidade de perturbações,
- int *iMelhor – posição de referência para calcular a posição a ser perturbada,
- int nSlots - quantidade de intervalos de tempo a cada 30 minutos,

Copyright © 2010 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

- int slotsPessoa - quantidade de intervalos de tempo que uma pessoa ocupa de forma contínua, ou seja, o comprimento do turno de trabalho.

A Fig. 3, apresenta a função *perturbacaoLocal.*

```
Algoritmo perturbacaoLocal ()

1.          int perturb ← 1
2.          indicePerturbacao ← iMelhor + distPertub
3.          enquanto (não atinge pertubação máxima) faça
4.              s' ← s
5.              para cada semana de s' faça
6.                      pertuba na posição indicePertubacao com pertub
7.
8.              iMelhor ← metodoDescida(s')
9.              se fo (s') < fo (s) então faça
10.                     s ← s'
11.                     pertub ← 1
12.             senão faça
13.                     s' ← s
14.                     para casa semana de s' faça
15.                             perturba na posição índicePertubacao com –pertub
16.                     fim para
17.                     iMelhor ← metodoDescida (s')
18.                     se fo (s') < fo (s) então faça
19.                             s ← s'
20.                             pertub ← 1
21.                     fim se
22.             fim se
23.         fim enquanto
fim perturbacaoLocal;
```

Figura 3: Função *perturbacaoLocal*

Uma perturbação no método ILS consiste em trocar um funcionário de um intervalo de tempo para outro, ou seja, na perturbação de nível 1, são trocados, um funcionário por intervalo de tempo, na perturbação de nível 2, são trocados dois funcionários, e assim por diante. Estas pertubações são realizadas para todos os dias da semana. O mesmo processo é realizado com as trocas realizadas no início do turno de trabalho, ou seja, na perturbação de nível 1, são trocados os horários em que se inicia um turno, na perturbação de nível 2, são trocados dois intervalos de tempo, nos quais se inicia um turno, e assim por diante. O *metodoILS* chama a função *pertubacaoLocal* para realizar estes movimentos.

Sempre que uma solução *s*' é aceita, a perturbação volta ao nível 1. O critério de parada do ILS é o número máximo de perturbações feitas sem melhora durante a execução do método, representado pela variável **distPertubMax**.

O método *ILS* recebe como entrada os seguintes parâmetros:

- matriz s – matriz que armazena sempre a melhor solução encontrada
- int nSlots – quantidade de intervalos de tempo a cada 30 minutos
- int slotsPessoa - quantidade de intervalos de tempo que uma pessoa ocupa de forma contínua, ou seja, o comprimento do turno de trabalho.

A Fig. 4 apresenta o pseudocódigo do método *ILS* aplicado ao SDP.

Copyright © 2010 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

```
Procedimento metodoILS()

1.      int distPertubMax ← valor definido de acordo com o número máximo de perturbações
2.      int perturbMax ← valor definido de acordo com as perturbações a serem realizadas
3.      int iMelhor
4.      int distPerturb
5.      matriz s'
6.      iMelhor ← metodoDescida(s)
7.      aloca s' na memória
8.      para distPerturb ← 1 até distPertubMax  faça
9.          perturbacaolocal(s, s', -distPerturb, perturbMax, iMelhor, nSlots, slotsPessoa)
10.         perturbacaolocal(s, s', distPerturb, perturbMax, iMelhor, nSlots, slotsPessoa)
fim metodoILS;
```

Figura 4: Pseudocódigo do método *ILS* aplicado ao SDP

6.4 Método AjustaCoordenadas aplicado ao SDP

Para resolver o problema, o método AjustaCoordenadas foi desenvolvido para perturbar todas as posições do vetor. O método AC realiza os seguintes ajustes:

- Ajuste 1: é realizado um ajuste reduzindo uma unidade na coordenada, ou seja, o número de funcionários presentes no turno é decrescido de uma unidade, retornando-se como vizinho aquele que apresentar o melhor valor da função objetivo. O procedimento é repetido para os *n* elementos do vetor. O critério de parada se dá quando não existam melhores resultados para a função objetivo.
- Ajuste 2: é realizado um ajuste aumentando uma unidade na coordenada, ou seja, o número de funcionários presentes no turno é acrescido de uma unidade, retornando-se como vizinho aquele que apresentar o melhor valor da função objetivo. O procedimento é repetido para os *n* elementos do vetor. O critério de parada se dá quando não existam melhores resultados para a função objetivo.

A Fig. 5 apresenta o pseudocódigo do método AjustaCoordenadas aplicado ao SDP.

```
Procedimento AjustaCoordenadas()

1.      vetor s              /* vetor-solução */
2.      vetor s'             /* vetor auxiliar para realizar modificações */
3.      int  n               /* tamanho do vetor */
4.      para i ← 1 até n faça
5.              s' ← s
6.              s' [i] ← s' [i] – 1
7.              enquanto (fo (s') < fo (s)) faça
8.                      s' ← s
9.                      s' [i] ← s' [i] – 1
10.             fim enquanto
11.             s' ← s
12.             s' [i] ← s' [i] + 1
13.             enquanto (fo (s') < fo (s)) faça
14.                     s' ← s
15.                     s' [i] ← s' [i] + 1
```

Copyright © 2010 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

```
16.          fim enquanto
17.      fim para
fim método AjustaCoordenadas;
```

Figura 5: Pseudocódigo do método AjustaCoordenadas aplicado ao SDP

7 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Os algoritmos foram implementados na linguagem C, e compilados em DEV C++. Os testes foram realizados em um computador Intel Core i5 430M, com 4GB de memória RAM DDR 3, sob o sistema operacional Windows 7. Para avaliá-lo, utilizaram-se 15 instâncias contendo dados de teste com base em dados reais de um *Call Center*, com a quantidade total de atendentes necessários a cada intervalo de tempo em cada dia da semana. As instâncias usadas nos testes são representadas por um arquivo texto contendo todas as informações pertinentes ao problema. Os dados contidos nas instâncias são de um *Call Center* real, pois não foram encontradas instâncias teste na literatura para o problema tratado. A situação real mostrada por estas instâncias foi comparada com os resultados obtidos pelos algoritmos ILS e AC.

7.1 Resultados computacionais para o Método ILS

Na Tabela 2, são apresentados os conjuntos de soluções de 1 das 15 instâncias utilizadas, antes e depois de processar o método *ILS,*com a inclusão dos intervalos de pausas. As colunas representam:

- DEM: representa a demanda total de funcionários por intervalo de tempo.
- FUNC: representa a quantidade de funcionários que iniciaram o turno de trabalho no intervalo de tempo.
- ALOC: representa a quantidade de funcionários no intervalo de tempo.
- ESC: representa a escassez de funcionários no intervalo de tempo.
- EXC: representa o excesso de funcionários no intervalo de tempo.

	ANTES DAS PAUSAS - ILS						DEPOIS DAS PAUSAS - ILS					
	DIA	DEM	FUNC	ALOC	EXC	ESC	DIA	DEM	FUNC	ALOC	EXC	ESC
1º Instância	SEG	7480	384	9216	1837	101	SEG	7480	384	8832	1453	101
	TER	8148	384	9216	1355	287	TER	8148	384	8832	1012	328
	QUA	7554	384	9216	1739	77	QUA	7554	384	8832	1355	77
	QUI	7720	384	9216	1608	112	QUI	7720	384	8832	1235	123
	SEX	7840	384	9216	1505	129	SEX	7840	384	8832	1140	148
	SAB	5746	259	6216	612	142	SAB	5746	259	5957	405	194
	DOM	3752	171	4104	516	164	DOM	3752	171	3933	385	204

Tabela 2: Conjunto das soluções de 1 instância antes e após o processamento do método ILS e após a inclusão dos intervalos de pausas.

Para a inclusão dos intervalos de pausas, o intervalo de tempo passou a ser considerado como sendo de 15 minutos, em lugar da duração padrão de 30 minutos normalmente adotada. Assim, o turno de trabalho do funcionário tem 24 *slots*, com a duração de 15 minutos cada, totalizando 6 horas de trabalho.

A demanda total dos funcionários - (DEM) e a quantidade de funcionários que iniciaram o turno de trabalho em certo intervalo de tempo (FUNC) é fixa, antes e depois de executar o

Copyright © 2010 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

método ILS com a inclusão das pausas, para todas as instâncias processadas. Em relação à quantidade de funcionários em certo intervalo de tempo, representado por ALOC, e o excesso de funcionários no intervalo de tempo - EXC pode-se observar que sofreram uma considerável redução, após a inclusão dos intervalos de pausas, devido a um melhor aproveitamento e dimensionamento dos funcionários, evitando-se assim, a ociosidade. Referente à escassez de funcionários em certos intervalos de tempo (ESC), pode-se observar que houve um aumento, em todas as instâncias processadas após a inclusão das pausas, devido a saída dos funcionários para os intervalos de descanso.

Desta forma, através dos dados apresentados na Tabela 2, observa-se que após a inclusão dos intervalos de pausas, os resultados encontrados em cada dia da semana melhoraram consideravelmente a solução final, tanto na quantidade de funcionários (ALOC), quanto no excesso (EXC) de funcionários a cada intervalo de tempo.

7.2 Resultados computacionais para o Método AjustaCoordenadas

Na Tabela 3, são apresentados os conjuntos de soluções de 1 das 15 instâncias utilizadas, antes e depois de processar o método AjustaCoordenadas,com a inclusão dos intervalos de pausas. As colunas representam:

- DEM: representa a demanda total de funcionários por intervalo de tempo.
- FUNC: representa a quantidade de funcionários que iniciaram o turno de trabalho no intervalo de tempo.
- ALOC: representa a quantidade de funcionários no intervalo de tempo.
- ESC: representa a escassez de funcionários no intervalo de tempo.
- EXC: representa o excesso de funcionários no intervalo de tempo.

	ANTES DAS PAUSAS - AC						DEPOIS DAS PAUSAS - AC					
	DIA	DEM	FUNC	ALOC	EXC	ESC	DIA	DEM	FUNC	ALOC	EXC	ESC
1º Instância	SEG	7480	346	8304	1128	304	SEG	7480	346	7958	832	354
	TER	8148	346	8304	846	690	TER	8148	346	7958	601	791
	QUA	7554	346	8304	1023	273	QUA	7554	346	7958	755	351
	QUI	7720	346	8304	961	377	QUI	7720	346	7958	705	467
	SEX	7840	346	8304	907	443	SEX	7840	346	7958	661	543
	SAB	5746	209	5016	440	1170	SAB	5746	209	4807	316	1255
	DOM	3752	137	3288	328	792	DOM	3752	137	3151	254	855

Tabela 3: Conjunto das soluções de 1 instância antes e após o processamento do método AjustaCoordenadas e após a inclusão dos intervalos de pausas.

A demanda total dos funcionários - (DEM) e a quantidade de funcionários que iniciaram o turno de trabalho em um certo intervalo de tempo (FUNC) é fixa, antes e depois de executar o método AC, com a inclusão das pausas, para todas as instâncias processadas. Em relação à quantidade de funcionários - ALOC, e o excesso de funcionários - EXC, pode-se observar que sofreram uma considerável redução, após a inclusão dos intervalos de pausas, devido a um melhor aproveitamento e dimensionamento dos funcionários. Referente à escassez de funcionários – ESC, pode-se observar que, houve um aumento, em todas as instâncias processadas após a inclusão das pausas, devido a saída dos funcionários para os intervalos de descanso.

Assim, os dados apresentados na Tabela 3, após a inclusão dos intervalos de pausas, melhoraram consideravelmente a solução final, tanto na quantidade de funcionários (ALOC),

Copyright © 2010 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

quanto no excesso (EXC) de funcionários.

7.3 Comparação dos Métodos ILS e AC - através dos resultados semanais

Na Tabela 4, são apresentados os resultados da função objetivo gerados pelos métodos ILS e AC, antes e após a inclusão dos dias de descanso e dos intervalos de pausa. As colunas representam:

- FO GULOSO: representa o resultado obtido para a função objetivo na execução do método guloso.
- FO ILS C/ FOLGA: representa o resultado obtido para a função objetivo após o processamento do método ILS com a inclusão dos dias de folga dos funcionários.
- FO APÓS PAUSAS ILS: representa o resultado obtido para a função objetivo após o processamento do método ILS com a inclusão dos intervalos de pausas.
- FO AC C/ FOLGA: representa o resultado obtido para a função objetivo após o processamento do método AC com a inclusão dos dias de folga dos funcionários.
- FO APÓS PAUSAS AC: representa o resultado obtido para a função objetivo após o processamento do método AC com a inclusão dos intervalos de pausas.

		ILS		AC	
INST.	FO GULOSO	FO COM FOLGA	FO APOS PAUSAS	FO COM FOLGA	FO APOS PAUSAS
1º	17918.40	14874.40	12494.50	14909.10	13948.80
2º	7969.20	5407.30	4537.60	7885.20	7555.80
3º	5412.20	3418.10	3191.60	5720.10	5493.30
4º	10988.20	10030.50	9172.80	8544.00	9829.20
5º	13277.20	11798.60	11747.60	15585.90	15398.70
6º	7866.20	6627.20	6237.80	7451.10	7448.40
7º	16230.80	13647.00	11496.90	15440.10	15376.20
8º	16227.00	12419.40	11077.50	15213.30	15506.70
9º	6892.40	5267.80	4743.70	7655.10	7251.90
10º	10143.20	9661.40	8678.90	5416.20	5107.50
11º	10350.20	8547.40	7436.50	11140.80	10828.50
12º	16679.60	14556.40	12712.00	17066.70	16537.50
13º	19029.20	16149.90	15511.80	18037.80	17956.80
14º	18292.80	13099.20	11514.30	15674.10	15250.20
15º	17990.80	15320.10	13648.80	17919.90	16967.70

Tabela 4: Resultados semanais obtidos para a função objetivo antes e após a execução dos métodos ILS e AC incluindo os dias de descanso e os intervalos de pausa

Conforme se pode observar através da Tabela 4, os resultados obtidos para a função objetivo geradas pelo método ILS, tanto na inclusão dos finais de semana, quanto na inclusão dos intervalos de pausas, é menor e melhor, quando comparados aos resultados obtidos pelo método AC. Isto acontece porque o método AC consegue fazer com que o sábado e o domingo tenham a quantidade precisa de funcionários a cada intervalo de tempo. Já o ILS sempre aloca funcionários à mais do que o necessário, fazendo com que a escassez de funcionários seja reduzida e aumente o excesso de funcionários. Assim, na escala gerada pelo método ILS, é preciso ter funcionários extras trabalhando aos fins de semana, mesmo não

Copyright © 2010 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

sendo necessário, uma vez que a demanda para o sábado e domingo é menor.

Em todas as instâncias processadas, o método ILS não respeitou o número de funcionários estabelecidos para os fins de semana, pelo fato de ser um método que pára por não encontrar uma "troca melhor", ou seja, o ILS pára o processamento antes de obter o número de funcionários desejado. Já o método AC é um método persistente e executa enquanto houver melhores resultados para a função objetivo. Assim, o método AC consegue encontrar a quantidade ideal de funcionários, porém o índice dos excessos e de escassez de funcionários é elevado, fazendo com que o valor da função objetivo seja maior.

7.4 Escala final gerada pelos Métodos ILS e AC

A escala semanal gerada pelos métodos ILS e AC, comparadas com a escala real utilizada no *Call Center*, é apresentada na Figura 6.

Conforme pode-se observar, a escala de trabalho dos funcionários geradas pelos métodos AC e ILS mostrou um melhor desempenho e uma melhor distribuição dos funcionários, em relação a escala de trabalho real do *Call Center* - (ATUAL). O método AC fez um dimensionamento preciso e enxuto, quando comparado ao método ILS e a escala de trabalho do *Call Center*, perdendo apenas entre 05:00 e 06:00 horas. O método ILS, em muitos instantes, acompanha a escala de trabalho do *Call Center* e dimensiona melhor os funcionários no período da noite, entre 18:00 e 19:00 horas. Quanto à escala de trabalho do *Call Center*, existe uma má distribuição dos funcionários no período noturno, no horário entre 20:00 e 21:00 horas, sem existir demanda de ligações para esta mão-de-obra.

Portanto, a escala mensal gerada pelo método ILS pode ser aplicada a escala de trabalho dos funcionários do *Call Center*, apresentando ganhos com a redução do número de funcionários, com a redução da quantidade de turnos distintos e com a qualidade da solução final obtida. Após estes resultados, basta alocar os funcionários para o trabalho, levando-se em consideração as preferências individuais de cada um.

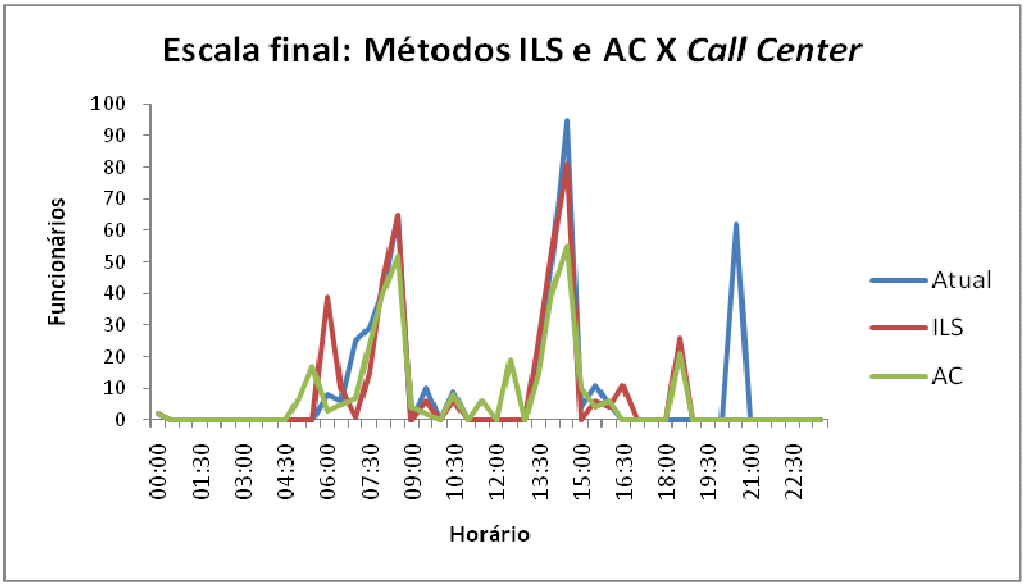

Figura 6: Escala final: Métodos ILS e AC X *Call Center*

8 CONCLUSÕES

Este trabalho apresentou o método *ILS* e o método AjustaCoordenadas (AC) para a

Copyright © 2010 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

resolução do *Shift Design Problem* aplicados à criação de turnos de trabalho de uma empresa de *Call Center,* considerando a minimização do número de turnos distintos, o excesso e a escassez de funcionários, e as diferenças do número médio de tarefas executadas por funcionários, por semana, incluindo os dias de folga e os intervalos de descanso. Nos testes computacionais realizados, os algoritmos mostraram-se capazes de produzir um conjunto de soluções que mostram o compromisso entre os objetivos propostos. Além disso, verificou-se que os métodos são facilmente implementados e os tempos de processamentos foram baixos, produzindo ótimos resultados. Os resultados obtidos para a função objetivo após a inclusão dos intervalos de descanso melhoraram consideravelmente a solução final. Em relação, aos resultados obtidos através do método AC, pode-se observar que obtém os piores resultados e, em alguns casos, piora a solução inicial gerada pelo método guloso, pelo fato de alocar a quantidade precisa de funcionários para os finais de semana. Para os resultados obtidos semanalmente, em todos os testes computacionais, a metaheurística ILS obtém bons resultados na solução do SDP aplicado a criação dos turnos de trabalho dos funcionários de um *Call Center*, aqui avaliado, podendo ser aplicado a um *Call Center* real.

Não foram feitas comparações com resultados da literatura, pois não foram encontradas instâncias disponíveis para testes. As instâncias utilizadas são reais de um *Call Center* em funcionamento. Após estes resultados, basta alocar os funcionários para o trabalho, levando-se em consideração as preferências individuais de cada um. Esta é a próxima etapa deste trabalho.

REFERÊNCIAS

Bhulai, S., G. Koole, A. Pot. Simple methods for shift scheduling in multi-skill call centers. *Manufacturing & Service perations Management*, forthcoming, 2007.

Canon, C. Personnel scheduling in the call center industry. 4OR: A Quarterly Journal of Operations Research, 5(1):89–92, 2007.

Lourenço, H. R., Martin, O., Stützle, T. Iterated Local Search. *In* F.Glover and G. Kochenberger (eds), *Handbook of Metaheuristics*, p. 321 - 353, Kluwer Academic Publishers, Norwell, MA, 2003.

Musliu, N., Schaerf, A., and Slany, W. Local search for shift design. European Journal of Operational Research, 153(1):51–64, 2004.

Musliu, N., Beer, A., Schafhauser W., Gartner J. and Slany, W. Scheduling Breaks in Shift Plans for Call Centers, 2008.

Tellier, P. and White, G. Generating personnel schedules in an industrial setting using a tabu search algorithm. E. K. Burke, H. Rudov (Eds.): PATAT 2006, pages 293–302, 2006.

Copyright © 2010 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar