ISSN 2175-6295 Rio de Janeiro- Brasil, 12 e 13 de agosto de 2010

UMA APLICAÇÃO DO *SHIFT DESIGN PROBLEM* AO PLANEJAMENTO DE HORÁRIOS DE TRABALHO DOS FUNCIONÁRIOS DE *CALL CENTERS*, INCLUINDO INTERVALOS DE PAUSAS E DIAS DE DESCANSO

Cynthia da Silva Barbosa, Sérgio Ricardo de Souza, Gray Farias 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

RESUMO

Este trabalho apresenta a aplicação da Metaheurística *Iterated Local Search* (ILS) 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. 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 os intervalos de pausas dos funcionários e os dias de descanso, respeitando as restrições das leis trabalhistas. O problema em tela pertence à classe de problemas NP – difíceis, justificando, assim, o uso de técnicas heurísticas para solucioná-lo. O método de busca local utilizado para a geração da solução inicial para o ILS é o Método da Descida, que apresenta baixo custo computacional na implementação realizada. Os resultados mostram que os métodos propostos são capazes de gerar soluções viáveis, tanto na qualidade da solução final, quanto na rapidez.
Palavras-Chaves: *Shift Design Problem. Call Center*. Metaheurística. *Iterated Local Search.*

ABSTRACT

This paper presents the application of Iterated Local Search metaheuristic for solving the Shift Design Problem applied to the creation of shifts of a Call Center company, including break intervals and days off. The aim of this study is to determine a set of feasible solutions containing shifts and number of employees per shift in order to minimize excess and shortage of workers per shift, and the differences in the average number of tasks performed by employees per week, including break intervals of employees and days off, respecting the constraints of labor laws. The problem in analysis belongs to the class of problems NP – hard, justifying the use of heuristic methods for solving it. The local search method used for generating the initial solutions for the ILS is the Descent Method, which has low computational cost implementing performed. The results show that the proposed methods are able to generate viable solutions in both the quality of the final solution, as in speed.
Keywords. Shift Design Problem. Call Center. Metaheuristic. Iterated Local Search.

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 descando) dos funcionários. O problema em questão é tratado através de uma reformulação do *Shift Design Problem* (SDP), apresentado em Musli et al. (2004)

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. Este problema considera a alocação dos funcionários após as mudanças geradas nos turnos de trabalho utilizando métodos de busca local. Para a geração das mudanças dos turnos, é definido um algoritmo para a geração de uma boa solução inicial e um conjunto de movimentos para a geração da vizinhança. Para a definição dos intervalos de pausas, é definida uma formulação matemática aplicando algumas restrições.

Um problema similar ao SDP, relatado na literatura no início da década de 50, é o *Shift Scheduling Problem*, proposto por Dantzig (1954) e solucionado através de programação inteira. Nesta formulação, encontra-se um conjunto ótimo de trocas (permutação) de horários. As possíveis mudanças são enumeradas com base no início e na duração do turno e no intervalo de tempo para descanso. Quando o número de turnos aumenta, essa formulação se torna impraticável.

Bechtold and Jacobs (1990) apresentaram uma nova formulação implícita de programação linear, com a inclusão de intervalo de descanso para a programação de turnos. Os autores encontraram resultados superiores aos obtidos em Dantzig (1954), porém esta abordagem é limitada às organizações que operam com menos de 24 horas de trabalho por dia.

Thompson (1995) apresenta um modelo de programação inteira para o desenvolvimento de horários de turnos de trabalho, permitindo uma grande flexibilidade em relação a turnos de trabalhos alternados, a comprimento dos turnos e aos intervalos de descanso. O modelo está relacionado aos trabalhos de Moondra (1976) e Bechtold e Jacobs (1990), em que o comprimento dos turnos e os intervalos de pausas são modelados implicitamente.

Em Aykin (1996) é proposto um modelo de programação inteira sem limitações e em Aykin (2000) são comparados diferentes tipos de abordagem para o *Shift Scheduling Problem.*

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, utilizando um algoritmo guloso para a definição de uma solução inicial e o método heurístico *Iterated Local Search* (ILS) para refinar a solução final, com o objetivo 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.

Na literatura, são encontradas algumas abordagens para resolver a fase de criação de horários de trabalho de funcionários de uma empresa. Em Glover e McMillan (1986), os problemas de geração dos turnos e a atribuição dos deslocamentos de horários para a designação dos empregados são resolvidos como um único problema. Segundo Musliu et al. (2004), a principal desvantagem desta abordagem é que ela não garante que uma boa solução para a atribuição das mudanças dos turnos seja encontrada.

O *Shift Design Problema* (SDP) é introduzido por Musliu et al. (2004) e considera a alocação real dos funcionários após as mudanças geradas nos turnos de trabalho. 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 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, são apresentadas a metodologia adotada e a aplicação do método da descida e *ILS*. Na seção 6 são apresentados e discutidos os resultados computacionais. A seção 7 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 $[a_i, a_{i+1}]$ está relacionado com o número w_i (quantidade de funcionários), indicando a quantidade ideal de funcionários para um determinado intervalo de tempo. Cada intervalo $[a_i, a_{i+1}]$ 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 instante de tempo a_1 - às 06:00 horas da manhã, e o finaliza no instante de tempo a_{n+1} , ou seja, às 12:00 horas. Assim, a jornada de trabalho de um funcionário possui 24 intervalos de tempo, tendo 15 minutos de duração cada um deles;
- 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;

- $v_{i.}$ min-comp e $v_{i.}$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.

Tabela 1 – Tipos de turnos

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

Conhecidas as variáveis aplicadas a este trabalho, o objetivo é gerar um conjunto k de turnos $s_1,...,s_k$, minimizando a quantidade de turnos distintos e a quantidade de funcionários em cada turno, tendo cada turno s_l parâmetros de início de turno $s_{l.}$ *inicio* e de duração de turno s_l *.duração*. O horizonte de planejamento é feito para uma semana.

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 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 multicritério. 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, e é dada por:

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

sendo:

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

Esta expressão 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 e a escassez F_2 (em minutos) dos funcionários em todos os intervalos de tempo durante o período de planejamento são definidos como:

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

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

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 como:

$$F_4 = \max(AvD - AS, 0)$$

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:

$$FO = \alpha F_1 + \alpha F_2 + \alpha F_3$$

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: fator de ponderação, dependente das instâncias utilizadas.

2.2. PLANEJAMENTO DOS DIAS DE FOLGAS DOS FUNCIONÁRIOS

O planejamento dos dias de descanso dos funcionários para o problema em tela será aos 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.

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*.

As restrições relacionadas com a legislação trabalhista brasileira em vigor, e aplicadas a este trabalho estão descritas abaixo, adaptadas a partir de Musli et al. (2008):

- Restrição 1: a pausa deve iniciar em um determinado intervalo de tempo após o início da jornada de trabalho. Para o problema em tela, foi definido que um funcionário poderá sair para a pausa lanche após uma hora do início de sua jornada de trabalho.
- Restrição 2: a pausa não deve ser inferior ao tempo mínimo permitido. O tempo mínimo de pausa aceita é de 8 minutos.
- Restrição 3: a pausa não pode iniciar no fim da jornada de trabalho. Esta restrição garante que um funcionário não poderá trabalhar mais de quatro horas consecutivas sem intervalo de descanso.
- Restrição 4: o comprimento da pausa não pode ultrapassar o limite máximo definido, ou seja, 15 minutos.

- Restrição 5: é necessário que haja um intervalo mínimo entre as pausas lanche e banheiro.
- Restrição 6: o número de funcionários em cada intervalo de tempo não pode ser inferior a quantidade de funcionários necessários para atender a demanda.

Para garantir a quantidade mínima necessária de funcionários em cada intervalo de tempo (15 minutos) atendendo a Restrição 6, 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 Restrição 6, segundo Tellier e White (2006), é calculada como:

$$P(V) = \sum_i (t_i - p_i)^2$$

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

Entende-se por vizinho de uma solução *s* uma solução *s'* alcançada aplicando-se uma transformação *m* em *s.* Representa-se essa operação por $s' \leftarrow s \oplus m$. 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.

A Fig. 1, encontrada em Lourenço et al. (2003), apresenta o pseudocódigo deste método.

```
Procedimento ILS()
1.      s0 ← GeraSolucaoInicial();
2.      s ← BuscaLocal(s0);
3.      iter ← 0;
4.      enquanto (iter < itermax)
5.          iter ←  iter + 1;
6.          s' ← Perturbacao(nível, s);
7.          s" ← BuscaLocal(s');
8.          s ← CriterioAceitacao(s");
9.      fim-enquanto;
10.     retorne s;
fim ILS;
```

Figura 1: Pseudocódigo do método *ILS*.

5. METODOLOGIA

5.1. REPRESENTAÇÃO DE UMA SOLUÇÃO

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

5.2. 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
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

5.3. *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,
- 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, encontrada apresenta o pseudocódigo do método *ILS* aplicado ao SDP.

```
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. 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. Para efeito de um melhor planejamento cada intervalo de tempo corresponde a quinze minutos.

Na Tabela 2, são apresentados os conjuntos de soluções de 3 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.

Os dados da Tabela 2 mostram que, após a inclusão das pausas, os resultados apresentados em cada dia da semana melhoraram consideravelmente a solução final, tanto na quantidade de funcionários no intervalo de tempo (ALOC), quanto no excesso (EXC) de funcionários em cada intervalo de tempo.

Na Tabela 3, são apresentados os resultados da função objetivo gerado nas 15 instâncias testes antes da execução do método ILS e após a execução do método *ILS* incluindo os dias de descanso e os intervalos de pausa.

Tabela 2: Conjunto das soluções de 3 instâncias antes e após o processamento do método ILS e após a inclusão dos intervalos de pausas.

	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
2º Instância	SEG	2102	119	2856	770	16	SEG	2102	119	2737	651	16
	TER	2802	119	2856	194	140	TER	2802	119	2737	102	167
	QUA	2738	119	2856	232	114	QUA	2738	119	2737	130	131
	QUI	2680	119	2856	288	112	QUI	2680	119	2737	174	117
	SEX	2638	119	2856	340	122	SEX	2638	119	2737	221	122
	SAB	2578	117	2808	389	159	SAB	2578	117	2691	272	159
	DOM	2530	123	2952	576	154	DOM	2530	123	2829	453	154
3º Instância	SEG	2102	90	2160	129	71	SEG	2102	90	2070	101	133
	TER	2136	90	2160	128	104	TER	2136	90	2070	96	162
	QUA	2076	90	2160	149	65	QUA	2076	90	2070	105	111
	QUI	2018	90	2160	201	59	QUI	2018	90	2070	185	83
	SEX	1966	90	2160	262	68	SEX	1966	90	2070	179	75
	SAB	1902	89	2136	322	88	SAB	1902	89	2047	233	88
	DOM	1850	93	2232	462	80	DOM	1850	93	2139	369	80

Tabela 3: Resultados das instâncias para função objetivo antes e após a execução do método *ILS* incluindo os dias de descanso e os intervalos de pausa.

	FO GULOSO	FO ILS COM DIAS DE FOLGA	FO ILS COM AS PAUSAS
1º Instância	17918.40	14874.40	12494.50
2º Instância	7969.20	5407.30	4537.60
3º Instância	5412.20	3418.10	3191.60
4º Instância	10988.20	10030.50	9172.80
5º Instância	13277.20	11798.60	11747.60
6º Instância	7866.20	6627.20	6237.80
7º Instância	16230.80	13647.00	11496.90
8º Instância	16227.00	12419.40	11077.50
9º Instância	6892.40	5267.80	4743.70
10º Instância	10143.20	9661.40	8678.90
11º Instância	10350.20	8547.40	7436.50
12º Instância	16679.60	14556.40	12712.00
13º Instância	19029.20	16149.90	15511.80
14º Instância	18292.80	13099.20	11514.30
15º Instância	17990.80	15320.10	13648.80

7. Conclusões

Este trabalho apresentou o método *ILS* aplicado ao *Shift Design Problem* à criação de turnos de trabalho de uma empresa de *Call Center* incluindo os dias de folga e os intervalos de descanso. O método heurístico *ILS* é uma técnica de busca local, retornando boas soluções. Para a geração da solução inicial para o método ILS, foi utilizado o método da descida, também conhecido como método guloso, que analisa todos os possíveis vizinhos de uma solução, escolhendo a cada passo, aquele que tem o menor valor para a função objetivo. O algoritmo *ILS* foi facilmente implementado e o tempo de processamento foi baixo, produzindo ótimos resultados. Isto mostra que é um método simples e apresenta bom desempenho, podendo ser aplicado para diferentes classes de problemas. Os resultados obtidos para a função objetivo após a inclusão dos intervalos de descanso melhoraram consideravelmente a solução final. 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

Aykin T. (1996). Optimal shift scheduling with multiple break windows, Managament Science 42 (4) 591–602.

Aykin T. (2000). A comparative evaluation of modeling approaches to the labor shift scheduling problem, European Journal of Operational Research 125 381–397.

Bechtold S.E. **e Jacobs L.W**. (1990). Implicit modeling of flexible break assignments in optimal shift scheduling, Management Science 36 (11) 1339–1351.

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

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

Danzig G.B. (1954). A comment on eddie's traffic delays at toll booths.Operations Research 2, pp. 339–341.

Glover F., McMillan C. (1986). The general employee scheduling problem: An integration of MS and AI, Computers and Operations Research 13 (5) 563–573.

Glover, F. W. e Kochenberger, G. A. (2003). Handbook of metaheuristics.Hardcover.

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

Moondra L. (1976). An LP model for work force scheduling for banks, Journal of Bank Research 7 299–301.

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

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

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

Thompson G. (1995). Improved implicit modeling of the labor shift scheduling problem, Management Science 41 (4) 595–607.