Iterated Local Search para o Problema de P-Próximo Centro

Mariana Alves Londe
Pontifícia Universidade Católica do Rio de Janeiro
R. Marquês de São Vicente, 225 - Gávea, Rio de Janeiro - RJ, 22452-900 - Departamento de Engenharia Industrial (DEI)
mlonde@aluno.puc-rio.br

Rodrigo Sarlo
Pontifícia Universidade Católica do Rio de Janeiro
R. Marquês de São Vicente, 225 - Gávea, Rio de Janeiro - RJ, 22452-900 - Departamento de Engenharia Elétrica (DEE)
rodrigosarloaf@aluno.puc-rio.br

Sofia Pontes de Miranda
Pontifícia Universidade Católica do Rio de Janeiro
R. Marquês de São Vicente, 225 - Gávea, Rio de Janeiro - RJ, 22452-900 - Departamento de Engenharia Industrial (DEI)
sofia8miranda@hotmail.com

Luciana S. Pessoa
Pontifícia Universidade Católica do Rio de Janeiro
R. Marquês de São Vicente, 225 - Gávea, Rio de Janeiro - RJ, 22452-900 - Departamento de Engenharia Industrial (DEI)
lucianapessoa@puc-rio.br

RESUMO

Este artigo apresenta o algoritmo de metaheurística *Iterated Local Search* (ILS) para a solução do problema de *p-próximo centro*, sendo este considerado como uma variação do problema de *p-centro*. O problema consiste em escolher p de n centros e em seguida atribuí-os para cada usuário de forma a minimizar a distância do mesmo até o seu centro correspondente mais a distância deste centro até o seu centro de reserva. Este tipo de problema surge da necessidade, na logística humanitária, de se preparar para casos de desastre no qual o centro inicial pode estar indisponível e necessita-se de uma solução que considere uma segunda opção. A metaheuristica utilizada se baseia em buscas locais intercaladas com um processo de perturbação, o qual é utilizado para chegar a novas vizinhanças de busca. O ILS proposto consegue chegar ao ótimo de 75% das instâncias utilizadas, e, nos melhores casos, o diferença percentual entre o resultado obtido pela metaheuristica e o obtido pelo resolvedor CPLEX é de 1,17% em média.

PALAVRAS CHAVE. Problema de P-Próximo Centro, Problema de P-Centro, Iterated Local Search

Tópicos: MH (Metaheurísticas), L&T (Logística e Transportes)

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

ABSTRACT

This article presents the metaheuristic Iterated Local Search (ILS) for the P-Next Center Problem, which is a variation of the P-Center Problem. The P-Next Center Problem consists on the allocation of p centers to n points, and then allocate backup centers to each p, in order to minimize the distance between the n users and their backups. This problem is related to the necessity, in humanitarian logistics, to prepare for disasters in which the allocated center may be unavailable and thus a second option is needed. The developed metaheuristic is based on local searches interspersed with a perturbation process, used to find new neighborhoods for searching.The proposed ILS found the optimal solution in 75% of the used instances, and, in the best cases, the proportional difference between the result found by the metaheuristic and the one obtained by CPLEX is 1.17% in average.

KEYWORDS. P-Next Center problem, P-Center problem, Iterated Local Search

Paper topics: MH (Metaheuristics), L&T (Logistics and Transports)

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

1. Introdução

Na ocorrência de um desastre, seja este causado pela natureza ou por seres humanos, qualquer indivíduo afetado é encaminhado para centros de referência, que podem ser hospitais, campos de refugiados ou abrigos. Entretanto, serviços de emergência podem falhar em seu momento de uso. Exemplos de falhas incluem falta de remédios ou produtos hospitalares em um hospital, a falta de alimentos em um campo de refugiados ou destruição de um abrigo. Por consequência, estes centros de referência não são capazes de auxiliar a área afetada, sendo necessário que os usuários recorram a outro centro. Este tipo de problema, chamado de *p-próximo centro*, originou-se a partir de problemas de *p-centro* e motivou o desenvolvimento deste artigo, para a exploração de um novo método de resolução.

1.1. P-Centro

O *problema de p-centro* é um problema de localização em grafos no qual deseja-se selecionar p de n vértices e a alocação dos demais n-p ao p mais próximo.

Este problema está intimamente relacionado a situações em que é necessário atender a demandas que envolvem custos de deslocamento. Deseja-se alocar um centro de atendimento de demanda de forma a minimizar o custo de suprimento de um serviço para o fornecedor, no caso de serviços de entrega de produto; ou minimizar o custo para o usuário final, que precisa se locomover até o local adequado para recebimento do serviço, como caso de um hospital. Um exemplo ilustrativo do problema pode ser visto na Figura 1, na próxima subseção. Essa figura representa a qual centro o vértice ilustrado por um triângulo (dentre os p possíveis como círculos) foi atribuído. Na próxima seção teremos uma comparação com a solução para o *p-próximo centro*.

Dentre os métodos de resolução mais citados para o problema, temos uma heurística bastante utilizada que envolve a solução iterativa de p problemas de *1-center*, apresentada em [Drezner, 1984], e também citado como alternativa em [Mladenovic et al., 2003]. Em [Dyer e Frieze, 1985] é apresentado um método aproximativo para construção de uma solução do problema. O método apresentado possui, garantidamente, uma solução de no máximo 2 vezes o ótimo. O Iterated Local Search foi utilizado para resolução de uma versão com restições de capacidade para cada centro em [Quevedo-Orozco e Ríos-Mercado, 2015].

1.2. P-Próximo Centro

O problema de *p-próximo centro*, proposto pela primeira vez em [Albareda-Sambola et al., 2015], surge num contexto de logística humanitária: em situações de desastre, é possível que o centro ao qual usuários de uma dada região foram alocados esteja lotado ou danificado. Neste caso, o usuário precisa se locomover do seu centro de referência a um centro auxiliar. Neste problema, deseja-se minimizar o custo da ida ao primeiro centro (de referência) *somado* ao o custo da ida deste centro ao mais próximo (backup) como observado na Figura 2, onde exibimos as soluções do *p-centro* e do *p-próximo centro* da perspectiva de um usuário.

Analisando a mesma instância, o vértice triangular é atribuído a um centro diferente em cada versão do problema. A solução do *p-próximo centro* leva em conta que, no caso de falha no primeiro centro, o usuário precisa se locomover para o centro mais próximo. Assim, a distância percorrida (total) por este usuário é minimizada quando mudamos seu centro de referência em relação ao que seria feito sem levar em conta essa possibilidade.

Na elaboração da função objetivo ficam implícitas duas hipóteses: a probabilidade do centro de referência **e** backup falharem ao mesmo tempo é nula; e o usuário só tem a informação de que seu centro de referência não poderá atende-lo assim que chega lá - caso soubesse de antemão, a função objetivo levaria em conta a distância entre o usuário e o centro de referência *somado* a distância deste usuário e o centro auxiliar.

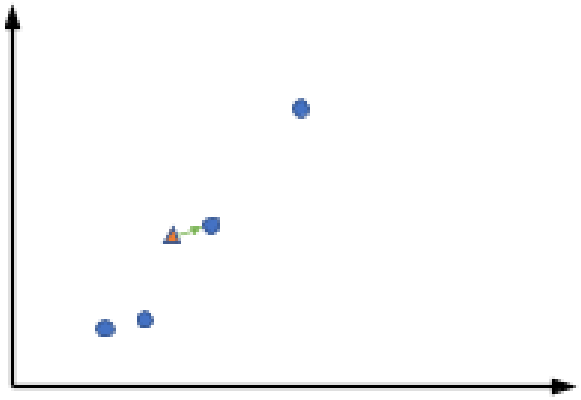


Figura 1: Figura 1: Ilustração do p-centro

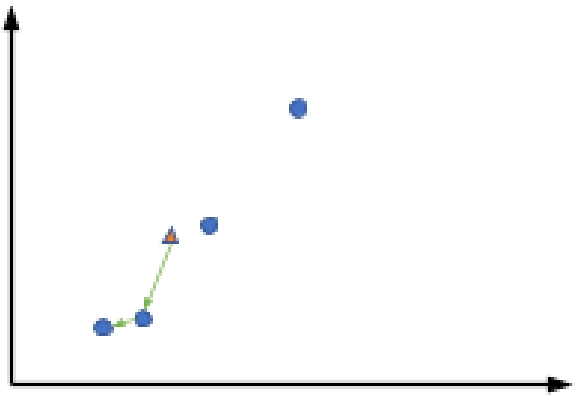


Figura 2: Figura 2: Ilustração do p-próximo centro

Conforme exposto em [Lopez et al., 2018] e ilustrado nas Figuras 1 e 2, até para instâncias simples os problemas de *p-centro* e *p-próximo centro* tem custos e alocações de centros bastante diferentes. O problema de *p-próximo centro* pode ser considerado NP-difícil, como provaram Albareda-Sambola et al. [2015].

A seguir apresentamos formalmente o problema conforme definido por Albareda-Sambola et al. [2015]:

Seja $A = 1, \ldots, n$ o conjunto de vértices do grafo que representam usuários e também locais canditatos a receberem centros. Para cada par $(i, j) \in A$, denomine $d_{i,j}$ a distância (custo) de ir do vértice i para vértice j. Seja $p \geq 2$ o número de centros a serem alocandos dentre os n possíveis.

O problema de *p-próximo centro* é definido como:

$$\min_{\substack{Q \subset A \\ |Q|=P}} \max_{i \in A} \left\{ \min_{j \in Q} \{d_{i,j}\} + \min_{j' \in \operatorname{argmin}_{j \in Q}\{d_{i,j}\}} \min_{\substack{k \in Q \\ k \neq j'}} \{d_{j',k}\} \right\}$$

Equação 1: Formulação de Albareda-Sambola et al. [2015]

Como mostrado na Equação 1, este problema procura minimizar a maior dentre as distâncias entre usuário e centro auxiliar, passando pelo centro de referência. Essa distância é obtida a partir da alocação, para cada usuário, do centro mais próximo como seu centro de referência e, para cada centro, do centro mais próximo como centro auxiliar.

2. Método de Resolução

2.1. Iterated Local Search

O algoritmo *Iterated Local Search* (ILS) pode ser descrito da seguinte maneira: considerando um mínimo local s*, aplica-se uma perturbação ao sistema que conduz a um estado intermediário s'. Uma busca local é então aplicada a s' de forma que se atinge uma nova solução s*'. Caso esta nova solução, s*', seja aprovado por um teste de aceitação, ela se torna o próximo ponto de partida. Caso não seja aceita, retorna-se a s* [Lourenço et al., 2010].

A força desta perturbação possui grande influência e é essencial ao processo. Caso seja pequena demais, acaba-se, normalmente, retornando a s*, de tal forma que poucas novas soluções

são exploradas. Entretanto, se a perturbação for grande demais, isto faz com que o procedimento se torne um algoritmo de reinicialização aleatória. O procedimento básico de um ILS é demonstrado na figura 3 abaixo e a sua estrutura é definida pelo Algoritmo 1, conforme apresentado em [Lourenço et al., 2010]. Na figura abaixo, à solução inicial, s*, é aplicada uma perturbação que leva ao valor s'. Após a execução de uma busca local, encontra-se um novo mínimo s*' podendo este ser melhor ou não que s*.

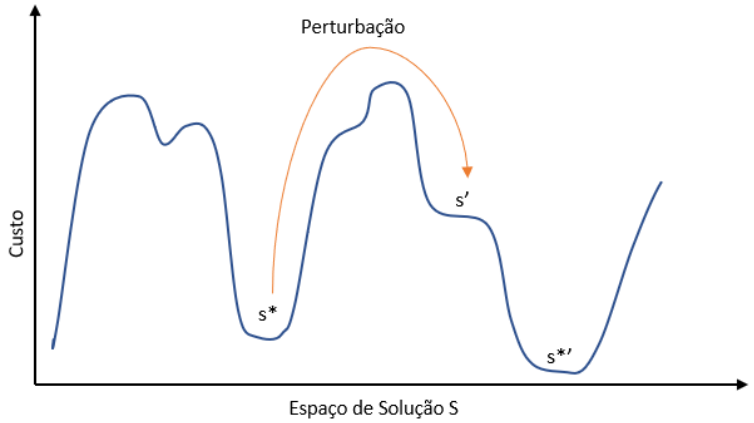


Figura 3: Representação do comportamento do ILS. (Adaptado de Lourenço et al. [2010])

O Algoritmo do ILS se inicia com a definição de uma solução inicial, que pode ser aleatória ou gulosa, a qual é aplicada uma busca local. Então é feito o procedimento de perturbação, que irá pegar o ótimo local encontrado na busca e leva-lo para outro ponto no espaço de soluções. A esse ponto, será aplicada novamente uma busca local, e será feita a comparação entre o resultado da mesma e a melhor solução encontrada. Esse processo de perturbação, busca e comparação é feito iterativamente, até que a condição de parada seja atingida.

Algoritmo 1: Iterated Local Search

```
1 s_0 = SolucaoInicial();
2 s* = BuscaLocal(s_0);
3 repita
4     s' = Perturbacao(s*);
5     s*' = BuscaLocal(s');
6     s* = Criterio(s*, s*');
7 até condição de parada;
```

ILS já foi explorado em diversos problemas tais como *flowshop scheduling* [Dong et al., 2015; Xu et al., 2016; Zohali et al., 2018], roteamento de veículos [Silva et al., 2015], problema da mochila [Avci e Topaloglu, 2017], problemas de p-centro [Quevedo-Orozco e Ríos-Mercado, 2015] e formação de células [Brusco, 2015], entre outros.

Nas próximas subseções, discutiremos a customização de cada uma das funções do Algoritmo 1 para aplicação ao problema do p-próximo centro.

2.2. Construtivo

O algoritmo construtivo utilizado é baseado em um sorteio aleatório dos **p** centros entre os diferentes **n** vértices possíveis. Primeiramente define-se um vetor de pares $\{n, rand\}$, sendo a segunda posição um número aleatório entre zero e um. Esse vetor é ordenado em ordem crescente dos números randômicos, e são selecionados os primeiros *p* itens como os centros de referência.

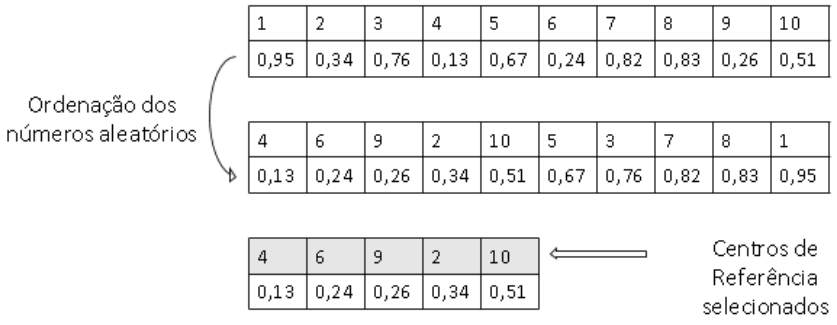


Figura 4: Ilustração do processo de seleção dos p's para um exemplo com n igual a 10 e p igual a 5.

Esta geração de soluções iniciais se assemelha à geração de soluções aleatórias juntamente com sua decodificação no Biased Random-Key Genetic Algorithm [Gonçalves e Resende, 2011].

Com a definição dos centros de referência ilustrada na Figura 4, é feito o procedimento "CalculaCusto", presente no Algoritmo 2. Nesse, inicia-se com a alocação, para cada vértice *n*, o centro de referência como sendo o mais próximo. Então alocam-se os *backups* de cada centro de referência de maneira análoga, e calcula-se a distância de cada *n* ao seu centro de referência *mais* a distância deste para o centro de backup. O valor máximo dentre essas distâncias (para todos os vértices) é o custo da solução.

Algoritmo 2: Calcula Custo

para p_i ***em*** $VetorP$ **faça**
  $\texttt{RefCenter}(n_j) = p_i \mid \texttt{Distancia}(n_j, p_i) \leq \texttt{Distancia}(n_j, p_i^*)$;
  $\texttt{BkpCenter}(p_i) = p_i^* \mid \texttt{Distancia}(p_i, p_i^*) \leq \texttt{Distancia}(p_i, p_i^{**})$;
  $\texttt{BkpCenter}(n_j$ ***não em*** $VetorP) = \texttt{BkpCenter}(\texttt{RefCenter}(n_j))$;
  $\texttt{distToBkpCenter}(n_i) = \texttt{Distancia}(n_j, \texttt{RefCenter}(n_j))$ $+\texttt{Distancia}(\texttt{RefCenter}(n_j), \texttt{BkpCenter}(n_j))$;
  $custo_s = \max(\texttt{distToBkpCenter}(n_j))$;
fim
retorna $s0$;

2.3. Busca Local

A busca local se inicia com a definição dos *n-p* vértices que não fazem parte do conjunto de centros de referência. São feitas trocas 1 para 1 entre os *p*'s da solução e os *p'* fora, e se compara o custo da solução nova com o anterior à Busca Local. Usamos o critério de *First Improvement*, o que significa que se escolhe a primeira solução que leva à melhoria no custo como resposta a ser retornada. Um exemplo do procedimento da busca local pode ser encontrada a seguir na figura 5 e o seu algoritmo pode ser visto no Algoritmo 3.

Temos, primeiro, a definição dos pontos que não se encontram dentro do vetor de centros de referência, que são agrupados em *VetorForaP*. Então, os p centros de referência são trocados, um por um, pelos pontos que não são centros. Isso ocorre dentro dos *loops* do algoritmo. Para o novo vetor de centros, é então calculado o custo da solução, e ela é comparada com a original. Caso ocorra uma melhora no resultado, então o procedimento termina e retorna a nova solução.

Algoritmo 3: Busca Local

```
1  VetorForaP = { n_j | n_j ∉ VetorP };
2  para p_i em VetorP faça
3      para p_j^* em VetorForaP faça
4          s^* = CalculaCusto (VetorP , com p_i trocado por p_j^* ) ;
5          se custo_s^* < custo_s então
6              Sol = s^*;
7              Quebra;
8          fim
9      fim
10 fim
11 retorna Sol;
```

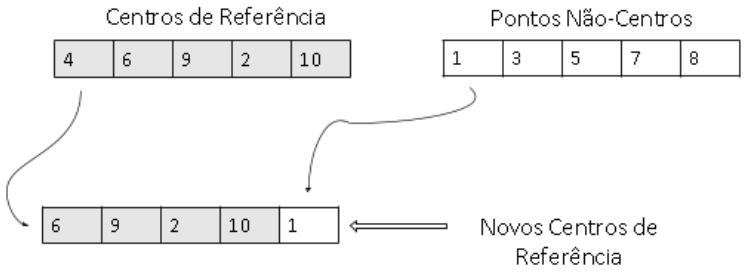


Figura 5: Ilustração do procedimento de Busca Local para um caso com n igual a 10 e p igual a 5.

2.4. Perturbação

O algoritmo de perturbação utilizado foi inspirado no procedimento análogo de Quevedo-Orozco e Ríos-Mercado [2015], em que são retirados os centros com maiores razões entre a máxima distância para um vértice alocado ao respectivo centro e a soma das distâncias para todos os vértices que este centro atende. Para o ILS, os centros de referência foram ordenados com base no valor da maior distância para cada centro, sendo sorteado um número, cujo máximo é o parâmetro *força* no pseudo-código a seguir, que determina quantos dos pontos irão permanecer como centro.

Como devemos ter, obrigatoriamente, p centros, devemos então acrescentar novos centros ao vetor de p's até que seu tamanho seja o necessário. As opções para acrescentar são os pontos que, anteriormente, não eram centros de referência, e são acrescentados com base em sua distância para o centro de Backup.

A figura 6 ilustra o procedimento de Perturbação para um caso com n igual a 10 e p igual a 5. É mostrado como é feita a ordenação, tanto dos centros de referência, quanto dos pontos que não

são centros. Após isso, temos a seleção dos centros que irão continuar na solução - no caso, 3. Isso é indicado pelo valor de rand**força*, que é relativo à força da perturbação e, no caso exemplificado, deve ser entre 3 e 4. Para ter o número correto de centros, então, os pontos não-centros com menores distâncias para o Backup são transformados em centros.

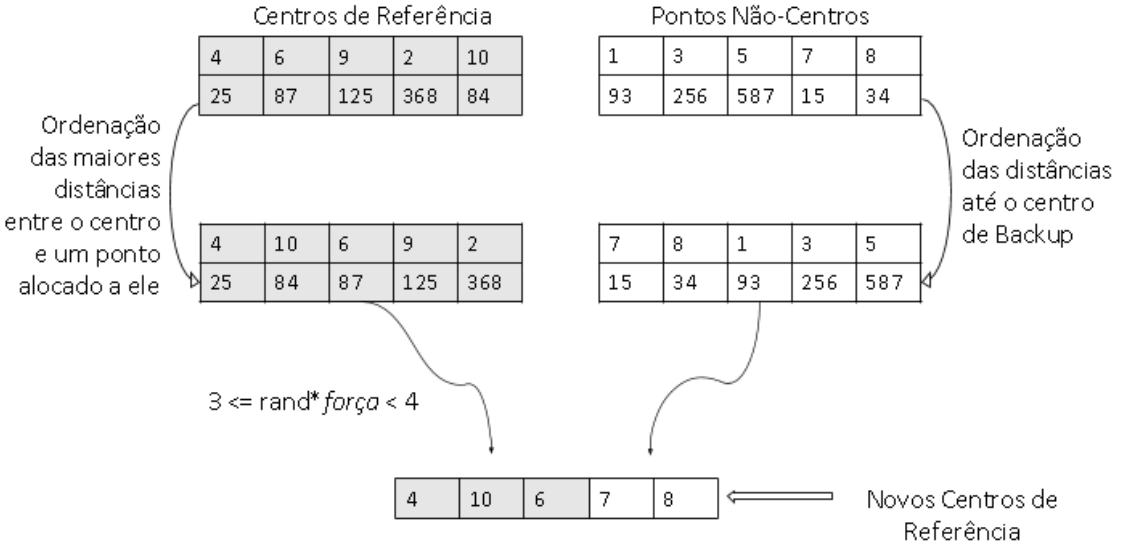


Figura 6: Ilustração do procedimento de Perturbação para um caso com n igual a 10 e p igual a 5.

Algoritmo 4: Perturbação

1 $\texttt{maxDistRefC}(p_i) = \texttt{max}(\texttt{Distancia}(n_j \mid \texttt{RefCenter}(n_j) = p_i, p_i))$;
2 $VetorDentroP = \{ p_i \in VetorP, \texttt{maxDistRefC}(p_i) \}$;
3 $\texttt{Ordena}(VetorDentroP.segundo)$;
4 $VetorForaP = \{ n_i \mid n \notin VetorP, \texttt{distToBkpCenter}(n_i) \}$;
5 $\texttt{Ordena}(VetorForaP.segundo)$;
6 **enquanto** $VetorP2.tamanho < rand * força$ **faça**
7 $\quad VetorP2.adiciona(VetorDentroP[1].primeiro)$
8 $\quad VetorDentroP[1].apaga$
9 **fim**
10 **enquanto** $VetorP2.tamanho < p$ **faça**
11 $\quad VetorP2.adiciona(VetrorForaP[1].primeiro)$
12 $\quad VetorForaP[1].apaga$
13 **fim**
14 $s' = \texttt{CalculaCusto}(VetorP2)$
15 **retorna** *s'*

O Algoritmo 4 se inicia com a definição da distância máxima entre um centro de referência e os pontos alocados a ele. Essa distância é, então, acrescentada ao *VetorDentroP*, juntamente com

o índice do centro correspondente. Esse vetor é ordenado com base nas distâncias, da menor para a maior. Em seguida, são identificados os pontos que não são centros de referência, ou seja, que não se encontram dentro do *VetorP*. Esses pontos são relacionados com a distância até seu centro de *back-up*, e o vetor correspondente é ordenado a partir desse valor.

O primeiro *loop* indica que o procedimento manterá como centro de referência os primeiros *rand*força* pontos, ordenados anteriormente na linha 3. Já o segundo *loop* considera que o valor de *rand*força* poderá ser menor do que o número p de centros de referência necessários, de modo que ele acrescenta ao vetor de centros os pontos que anteriormente não faziam parte da solução, ordenados anteriormente na linha 5, até que tenhamos p centros de referência. Com isso, é feito o procedimento de calcular o custo da solução, que é retornado pela perturbação.

2.5. Critério de Aceitação do ILS

Com relação ao critério de aceitação, foi utilizado um critério similar ao de *Simulated Anneling*. Assim, aceita-se a solução $s^{*'}$ caso seu custo seja menor do que a s^* anterior; caso contrário, aceita-se $s^{*'}$ com uma probabilidade dependente de um parâmetro de temperatura T, que diminui a medida que são completadas iterações.

3. Resultados Computacionais

Essa seção apresenta os resultados da aplicação do ILS para o conjunto de instâncias pequenas utilizadas em [Albareda-Sambola et al., 2015] e [Lopez et al., 2018]. Elas podem ser encontradas na *OR-Library*. Para comparação, foram obtidos os valores exatos da solução com o *solver* CPLEX 12.6, provenientes da formulação (F2) de Albareda-Sambola et al. [2015]. A metaheurística do ILS foi implementada em C++ e tanto o modelo de metaheurística quanto o modelo exato foram solucionados no mesmo computador. As especificações computaciosais do mesmo são: processador Intel Core i7-7700HQ de 2.8 GHz, e memória RAM de 16 GB.

3.1. Experimentos Preliminares

Foram feitos experimentos para determinar os valores corretos da força da perturbação, da temperatura do sistema e da taxa de diminuição dessa última. Os testes foram feitos nessa ordem, com 5 instâncias escolhidas aleatoriamente entre as utilizadas. Para cada instância foi escolhida uma semente aleatória diferente para o teste.

Para a análise do valor da Força da Perturbação, foi utilizado como critério de parada 5 segundos, e utilizou-se o valor inicial de T como 20000. Nesse caso, a temperatura em uma iteração seria a metade da temperatura na anterior.

Os valores para a força da perturbação testados sempre são relacionados com um valor aleatório sorteado a cada perturbação. Isso ocorre de maneira a diminuir a chance de serem repetidas as combinações resultantes, o que pode ocorrer por causa das ordenações dos vetores. Assim, deve-se definir o valor, denominado no pseudo-código como *força*, que diminui o valor da diferença percentual média encontrada. A seguir, encontram-se os resultados do experimento.

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

Tabela 1: Resultados dos Experimentos

			Força							
Problema	N	P	p	p/2	p/3	p/4	p/n	2p/n	p/2n	3p/n
pmed1	40	20	0.000	0.000	0.000	0.000	0.022	0.022	0.022	0.022
pmed3	10	5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
pmed2	40	10	0.205	0.205	0.232	0.205	0.420	0.420	0.420	0.420
pmed3	50	20	0.080	0.184	0.046	0.172	0.379	0.379	0.379	0.379
pmed3	30	5	0.000	0.000	0.000	0.000	0.083	0.083	0.083	0.083
Médias			0.057	0.078	0.056	0.076	0.181	0.181	0.181	0.181

É visível na tabela que os valores de força mais adequados não dependem da razão entre p e n. Além disso, observa-se que a menor diferença percentual média é relativa a uma força de *p/3*. Desta forma, este valor é o utilizado em todas as análises subsequentes. O valor inicial de T e sua taxa de diminuição foram testados em conjunto, de modo a avaliar qualquer possível interferência entre os dois. Seus valores foram definidos em 20000 e em $T/2$, respectivamente.

3.2. Comparação entre Modelo Matemático e ILS

Com os melhores valores dos parâmetros definidos, foi feita a comparação entre os resultados do CPLEX e do ILS, como anteriormente mencionado. Para cada instância, foram utilizadas 5 sementes aleatórias.

A Tabela 2 apresenta os resultados obtidos. Suas primeiras colunas mostram qual é a instância utilizada, seguidos pelos resultados do CPLEX, com seu valor ótimo e tempo de processamento, em segundos, até a solução ótima. Nas colunas seguintes estão os resultados para o Método Construtivo e Busca Local utilizados, ambos sendo executados por 5 segundos. Por fim. apresentamos os resultados do ILS com critérios de parada de 10 segundos e 40 segundos. Para cada uma das heurísticas, temos o tempo gasto até a solução encontrada (em segundos) e as *Diferenças Percentuais* encontradas em relação ao ótimo. Para cada um dos métodos apresentados, exceto para o CPLEX, temos os tempos e as diferencas percentuais para o melhor caso (Mínimo) e para a média das execuções.

Na tabela podemos observar a evolução dos resultados a cada nova etapa do ILS. Utilizando somente o construtivo, existe uma diferença percentual de 6,59% para a média dos melhores resultados quando comparado ao valor exato obtido pelo CPLEX, coluna de Mínimo. Na média de todas as execuções, a diferença percentual é ligeiramente pior, na casa de 8%. É interessante notar que apenas com a execução por repetidas vezes do método construtivo por 5 segundos, chegamos ao ótimo em 25% das vezes para essas instâncias.

Ao adicionar uma busca local, a diferença percentual diminui para 1,61% na média dos melhores casos, ficando em 2,6% para todas as execuções. Para os melhores casos, a busca local atinge o ótimo em 29 das 40 instâncias e em 25 das 40 instâncias em média. Ao aplicar o ILS com critério de parada de 40 segundos, a diferença dos melhores casos em relação ao ótimo chega a 1,17%, e a 2,21% em média para todas as execuções. Para os resutados em (Mínimo), o número de instâncias que alcançam o valor ótimo aumenta, chegando ao ótimo em 30 de 40 possíveis. Em média, porém, há uma ligeira piora em relação ao que vemos na busca local.

Para as médias de tempos utilizados, temos que em apenas uma instância a Busca Local demorou mais tempo do que o reportado pelo CPLEX para encontrar o ótimo. Nessa instância,

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

a Busca Local foi encerrada ainda fora do mínimo. Isto acontece 10 vezes para o ILS com 40 segundos de critério de parada.

Comparando com resultados da literatura, Lopez et al. [2018] conseguem atingir as soluções em 38 de 40 instâncias utilizando uma metaheurística que combina um GRASP e um VNS. Os autores obtiveram esses resultados com um tempo de CPU médio de 0,71 segundos, utilizando um Intel Core 2 Duo E8400 (3 GHz) com 4 GB RAM e implementado os algoritmos em Java 8.

Abaixo, podemos avaliar os melhores resultados por meio de um boxplot, apresentado na Figura 7. Nele vemos a evolução das diferenças percentuais em relação ao ótimo recuperado pelo CPLEX.Assim como demonstrado na tabela, vemos que por si só a busca local proposta já apresenta bons resultados. Ao adicionarmos a reinicialização viesada proposta do ILS passamos a ter, inicialmente, resultados mais dispersos. Isso está relacionado com a nossa estratégia de *first improvement*. Como no ILS passamos por uma perturbação e depois por uma busca local, essa mudança constante de vizinhança pode impedir que se atinja um mínimo rapidamente. Assim, pode ser que a busca local, por se manter numa mesma vizinhança, chegue a um ótimo antes do ILS para um tempo limite de execução pequeno. Com o aumento do tempo limite de execução, o ILS passa a apresentar resultados ainda mais concentrados em torno do ótimo retornado pelo Cplex.

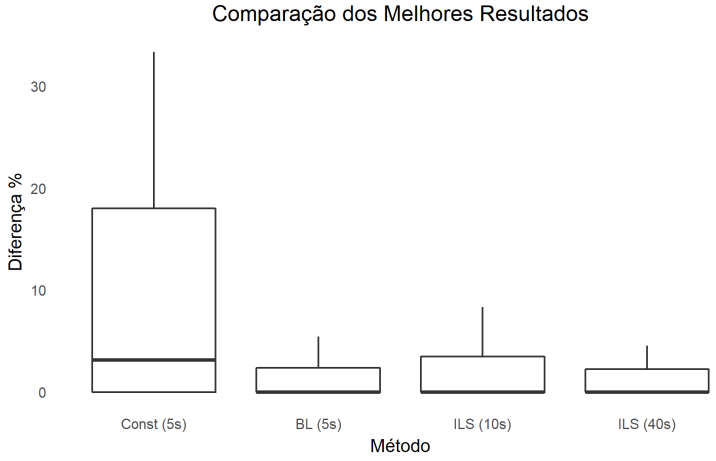


Figura 7: Representação Gráfica das Diferenças Percentuais apresentadas na tabela anterior

Tabela 2: Resultados

			Cplex 12.6	Const (5 s)				BL (5 s)				ILS (10 s)				ILS (40 s)			
				Mínimo		Médio		Mínimo		Médio		Mínimo		Médio		Mínimo		Médio	
Problema	N	P	Valor	Tempo(s)	Dif %	Tempo(s)*	Dif %	Tempo(s)*	Dif %	Tempo(s)*	Dif %	Tempo(s)*	Dif %	Tempo(s)*	Dif %	Tempo(s)*	Dif %	Tempo(s)*	Dif %
pmed1	10	5	84	0.03	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	6.67	0.01	0.00	0.00	6.67
pmed1	20	5	120	0.41	0.00	0.06	0.00	0.34	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00
pmed1	20	10	95	0.11	0.00	0.00	0.00	0.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00
pmed1	30	5	126	3.73	0.00	0.64	0.00	2.58	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.03	0.00	0.01	0.00
pmed1	30	10	95	0.58	2.11	0.50	3.79	2.22	0.00	0.02	0.00	0.05	0.00	0.10	0.00	0.20	0.00	0.10	0.00
pmed1	40	5	144	12.19	3.47	0.09	4.03	1.02	0.00	0.02	0.97	2.23	3.50	0.00	3.47	0.03	3.50	0.00	3.47
pmed1	40	10	111	3.61	4.50	1.00	7.57	2.10	3.60	0.03	3.60	0.21	1.80	0.30	1.80	2.24	0.00	11.36	1.44
pmed1	40	20	89	0.92	0.00	0.37	1.80	1.33	0.00	0.01	0.00	0.04	0.00	0.62	0.00	0.92	0.00	0.62	0.00
pmed1	50	10	110	20.31	17.27	1.59	21.45	2.04	4.55	0.95	6.36	2.85	0.90	0.27	4.54	2.85	0.90	0.27	2.36
pmed1	50	20	89	2.69	17.98	0.23	20.67	2.29	2.25	1.58	3.15	2.41	0.00	5.18	3.37	4.95	0.00	5.18	1.80
pmed2	10	5	121	0.05	0.00	0.02	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00
pmed2	20	5	147	0.59	0.00	0.00	0.00	1.11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00
pmed2	20	10	99	0.16	0.00	0.11	0.40	0.66	0.00	0.00	0.00	0.05	0.00	0.14	0.00	1.07	0.00	0.14	0.00
pmed2	30	5	169	4.05	0.00	0.88	0.71	2.28	0.00	0.01	0.00	0.02	0.00	0.06	0.00	0.26	0.00	0.06	0.00
pmed2	30	10	110	0.66	7.27	1.19	13.27	3.35	0.00	0.12	0.00	0.72	0.00	0.20	0.00	0.94	0.00	0.20	0.00
pmed2	40	5	164	16.27	0.00	0.02	0.24	1.17	0.00	0.03	0.00	0.09	7.30	0.00	7.32	0.01	7.30	0.00	7.32
pmed2	40	10	112	6.33	25.00	0.40	27.32	1.83	14.29	0.07	16.79	1.34	0.00	5.88	14.29	4.91	0.00	5.88	12.32
pmed2	40	20	96	1.00	3.12	1.18	3.96	2.68	0.00	0.14	0.00	0.59	2.10	0.01	2.08	0.89	0.00	30.61	1.67
pmed2	50	10	140	48.88	7.14	0.49	9.43	1.59	0.00	0.50	0.14	1.69	0.70	0.21	1.86	2.20	0.00	12.96	0.43
pmed2	50	20	99	3.22	20.20	0.17	21.62	1.86	2.02	0.03	2.83	1.52	3.00	0.72	3.23	6.47	2.00	11.48	2.83
pmed3	10	5	77	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
pmed3	20	5	145	0.55	0.00	0.08	0.00	1.11	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00
pmed3	20	10	77	0.09	0.00	0.06	2.86	1.04	0.00	0.01	0.00	0.01	0.00	0.07	0.00	0.08	0.00	0.07	0.00
pmed3	30	5	157	2.02	0.00	0.89	1.53	1.95	0.00	0.02	0.00	0.17	0.00	0.00	0.00	0.02	0.00	0.00	0.00
pmed3	30	10	122	1.38	0.00	0.19	0.33	0.89	0.00	0.02	0.00	0.07	0.00	0.02	0.00	0.14	0.00	0.02	0.00
pmed3	40	5	157	11.30	0.00	1.03	3.44	2.26	0.00	0.10	0.00	0.27	0.00	0.05	0.00	0.30	0.00	0.05	0.00
pmed3	40	10	105	6.91	20.00	0.67	23.81	2.67	14.29	0.38	14.29	1.47	13.30	0.06	14.10	2.92	10.50	15.41	12.57
pmed3	40	20	77	1.14	15.58	0.06	15.58	1.33	6.49	0.05	8.83	1.00	10.40	1.54	12.47	3.01	10.40	1.54	11.17
pmed3	50	10	125	52.52	12.00	1.32	16.80	2.34	7.20	0.01	7.20	1.04	1.60	8.66	6.08	4.16	1.60	8.66	1.60
pmed3	50	20	87	3.16	22.99	1.37	26.90	3.02	1.15	0.31	3.45	1.86	1.10	5.48	6.67	5.10	1.10	5.48	1.38
pmed4	10	5	126	0.05	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
pmed4	20	5	139	0.49	0.00	0.81	0.00	1.86	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01	0.00
pmed4	20	10	125	0.33	0.00	0.06	0.48	1.65	0.00	0.01	0.00	0.06	0.00	0.28	0.00	0.68	0.00	0.28	0.00
pmed4	30	5	173	4.14	0.00	0.21	0.00	1.29	0.00	0.01	0.00	0.05	0.00	0.00	0.00	0.03	0.00	0.00	0.00
pmed4	30	10	122	1.47	0.00	0.28	2.46	0.70	0.00	0.01	0.00	0.05	0.00	0.05	0.00	0.20	0.00	0.05	0.00
pmed4	40	5	175	14.30	1.14	0.22	2.86	2.09	0.00	0.03	0.00	0.12	0.00	0.01	0.00	0.08	0.00	0.01	0.00
pmed4	40	10	122	5.98	18.03	0.18	21.97	1.67	0.82	0.01	2.46	0.50	0.00	8.23	2.62	1.90	0.00	8.23	2.62
pmed4	40	20	85	1.23	18.82	2.28	22.35	2.77	0.00	1.19	2.35	2.90	7.10	5.10	14.12	4.91	7.10	5.10	12.94
pmed4	50	10	126	28.19	21.43	0.19	26.51	3.45	0.00	0.27	1.27	1.21	0.00	1.36	1.90	3.09	0.00	1.36	0.00
pmed4	50	20	91	2.67	24.18	0.31	28.35	1.90	7.69	0.46	8.57	1.29	2.20	1.68	7.47	2.50	2.20	1.68	5.71
Médias				6.59	6.56	0.48	8.31	1.62	1.61	0.16	2.06	0.65	1.38	1.16	2.85	1.43	1.17	3.17	2.21
Nº de Ótimos					21.00		11.00		29.00		25.00		27.00		22.00		30.00		23.00

Nota:
* valores referentes ao tempo até a solução, não o tempo total de execução

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

4. Conclusões e Pesquisas Futuras

Este artigo aborda o problema p-próximo centro, descrito inicialmente na área de logística humanitária. Para este problema foi proposta a aplicação da metaheurística Iterated Local Search (ILS) como forma de solução. As instâncias utilizadas são as mesmas que Albareda-Sambola et al. [2015] e Lopez et al. [2018] buscando poder comparar os resultados obtidos entre as metaheurística já estudadas. Usando um critério de parada de 40 segundos, a metaheurística aqui proposta foi capaz de chegar ao ótimo em 30 das 40 instâncias testadas. Em média, a diferença percentual entre o resultado obtido pela metaheurística, nos melhores casos, e o resultado exato obtido pelo CPLEX 12.6 foi de 1,17%.

Como sugestões pesquisas futuras, ainda ainda há diversas possibilidades a serem exploradas, sendo algumas delas: (i) a aplicação de novas heurísticas, buscando entender a sua eficácia em resolver o *problema de p-próximo centro*; (ii) analisar em um ambiente estocástico, onde cada centro possui a sua própria probabilidade de falha; (iii) estender um problema de *p-2-próximo centro*, onde para cada centro, existiria dois centros diferentes de backup.

Agradecimentos

Os autores gostariam de agradecer Yolanda Hinojosa pelo envio dos resultados obtidos pelos modelos em [Albareda-Sambola et al., 2015]. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001, da FAPERJ(E-26/211.086/2019 e E-26/211.389/2019), e do CNPq.

Referências

Albareda-Sambola, M., Hinojosa, Y., Marín, A., e Puerto, J. (2015). When centers can fail: A close second opportunity. *Computers Operations Research*, 62.

Avci, M. e Topaloglu, S. (2017). A multi-start iterated local search algorithm for the generalized quadratic multiple knapsack problem. *Computers Operations Research*, 83.

Brusco, M. (2015). An iterated local search heuristic for cell formation. *Computers Industrial Engineering*, 90:292–304.

Dong, X., Nowak, M., Chen, P., e Lin, Y. (2015). Self-adaptive perturbation and multi-neighborhood search for iterated local search on the permutation flow shop problem. *Computers Industrial Engineering*, 87.

Drezner, Z. (1984). The p-centre problem—heuristic and optimal algorithms. *Journal of the Operational Research Society*, 35(8):741–748.

Dyer, M. e Frieze, A. (1985). A simple heuristic for the p-center problem. *Operations Research Letters*, 3:285–288.

Gonçalves, J. e Resende, M. (2011). Biased random-key genetic algorithms for combinatorial optimization. *J. Heuristics*, 17:487–525.

Lopez, A., Sánchez-Oro Calvo, J., e Hernández-Díaz, A. (2018). Grasp and vns for solving the p-next center problem. *Computers Operations Research*, 104.

Lourenço, H., Martin, O., e Stützle, T. (2010). *Iterated Local Search: Framework and Applications*, volume 146, p. 363–397. ISBN 978-1-4419-1663-1.

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

Mladenovic, N., Labbé, M., e Hansen, P. (2003). Solving the p-center problem with tabu search and variable neighborhood search. *Networks*, 42:48–64.

Quevedo-Orozco, D. e Ríos-Mercado, R. (2015). Improving the quality of heuristic solutions for the capacitated vertex p-center problem through iterated greedy local search with variable neighborhood descent. *Computers Operations Research*, 62:133–144.

Silva, M., Subramanian, A., e Ochi, L. (2015). An iterated local search heuristic for the split delivery vehicle routing problem. *Computers Operations Research*, 53:234–249.

Xu, J., Wu, C.-C., Yin, Y., e Win-Chin, L. (2016). An iterated local search for the multi-objective permutation flowshop scheduling problem with sequence-dependent setup times. *Applied Soft Computing*, 52.

Zohali, H., Naderi, B., Mohammadi, M., e Roshanaei, V. (2018). Reformulation, linearization, and a hybrid iterated local search algorithm for economic lot-sizing and sequencing in hybrid flow shop problems. *Computers Operations Research*, 104.