Relatório Técnico: Busca Tabu Aplicada ao Problema do Caixeiro Viajante

André Britto de Carvalho, Rodolfo Barriveira, Carlos Tavares, Kelly Rodrigues, Frederico Losco

Resumo. *Este relatório apresenta uma comparação entre diferentes algoritmos para o problema do Caixeiro Viajante. Foram implementados algoritmos da Subida de Encosta e da metaheurística Busca Tabu. Os algoritmos foram implementados por três diferentes grupos e a comparação buscou encontrar os melhores resultados entre todos os algoritmos implementados. Um conjunto de casos de teste é executado para a validaçao dos resultados finais.*

1. Introdução

Problemas de otimização combinatória consitem, basicamente, em se obter uma melhor configuração de um conjunto de atributos [1]. Estes problemas se aplicam em diversas áreas da computação e é necessário encontrar boas soluções para estes problemas. Porém, uma grande parte destes problemas é NP-Difícil, sendo necessário assim o uso de heurísticas para se encontrar uma solução aproximada, mas de boa qualidade.

O Problema do Caixeiro Viajente é um problema NP-Difícil utilizado em diversas áreas da computação como robótica, redes, entre outras. Neste problema é preciso visitar *n* cidades, uma por vez, retornando ao ponto de partida, sem passar mais de uma vez por uma mesma cidade.

Para solução deste problema, diversas heurísticas foram proposta, como por exemplo, Colônia de Formigas, Algoritmos Genéticos, Busca Tabu, entre outros. Este relatório visa apresentar os resultados da implementação de algoritmos que utilizam a metaheurística Busca Tabu para solucionar o problema do Caixeiro Viajante.

Foram implementados um conjunto de seis algoritmos distribuídos por três grupos. O grupo 1, composto por André Carvalho e Rodolfo Barriviera, o grupo 2, composto por Carlos Tavares e o grupo 3, composto por Kelly Rodrigues e Frederico Losco. Cada grupo implementou uma versão do algoritmo de Subida de Encosta (*Hill Climbing*), básico, sem o uso de nenhuma metra-heurística e um algoritmo utilizando a Busca Tabu.

O algoritmo pode ser diferenciado através de como a solução inicial é gerada e como e feita a estrutura da vizinhança. Em relação a solução inicial, alguns procedimentos podem ser feitos para melhorar esta solução. Assim, o algoritmo inícia a busca

Neste relatório os detalhes e caracteríticas de cada algoritmos desenvolvidos são descritos. Além disto é feita uma comparação entre os algoritmos utilizando-se quatro conjuntos diferentes de cidades.

Este relatório está organizado como descrito a seguir. A Seção 2 apresenta uma introdução à Busca Tabu. Na Seção 3 são apresentadas as características dos diferentes

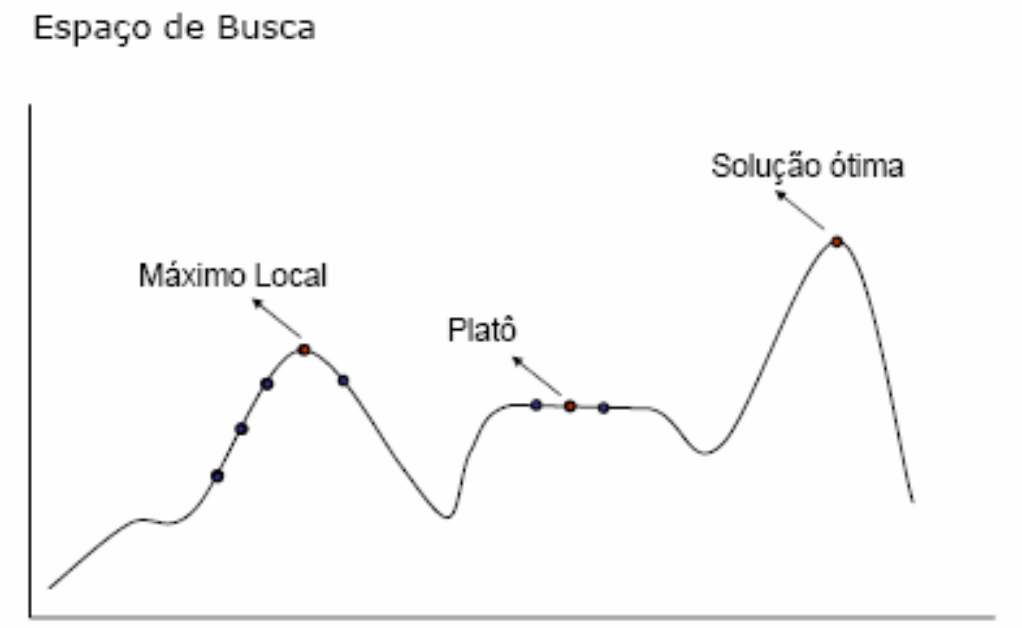
algoritmos implementados. Os resultados são discutidos na Seção 4 e por fim a Seção 5 apresenta as conclusões deste relatório.

2. Subida de Encosta

A Subida de Encosta é uma heurísitca que busca por uma combinação ótima de atributos, variando um valor de cada atributos por vez, com o intuito de encontrar um melhor o resultado para um problema. Enquanto há melhora na solução esse procedimento é repedito. Casa não haja mais uma combinação que melhore a função objetivo do problema a busca termina.

O nome Subida de Encosta é derivado do gráfico da função objetivo em relação às configurações possíveis do problema de entrada. Este gráfico possui picos, que podem ser ótimos locais ou ótimo globais. Há a possibilidade também da existência de de planícies, onde o valor da função objetivo não varia com a mudança de configuração dos atributos (Figura 1). No algoritmo, inicialmente é dada uma solução inicial e a busca tenta subir o gráfico da função objetivo tentando alcançar um pico. Porém caso este pico seja um ótimo local, ou uma planície, o algoritmo não consegue mais continuar a busca, e não atinge o ótimo do problema.

Figura 1. Gráfico da Subida de Encosta



O algoritmo as Subida de Encosta basicamente executa uma busca local sem nenhuma técnica que utiliza algum conhecimento sobre a busca. Uma solução inicial é gerada e é definida como a solução corrente. Após isto o algoritmo repete o seguinte procedimento: é gerada de um vizinhança em relação à solução corrente. Em seguida, é escolhida a melhor solução desta vizinhança. Caso esta solução seja melhor que a solução corrente, ela é atribuída como a solução corrente.

3. Busca Tabu

A Busca Tabu é uma metaheurística que tem por finalidade encontrar um melhor resultado através de uma busca heurística local. O método guarda informações sobre soluções já visitadas, buscando fugir de mínimos locais.

As principais características da Busca Tabu são: memória (lista) adaptativa, estratégias de buscas baseadas em memória e critério de aspiração. A memória adaptativa armazena os melhores elementos encontrados na busca, podendo ser memória de curta ou longa duração. A lista tabu também pode armazenar alguns atributos das soluções ao invés da solução completa. Um exemplo é o armazenamente das trocas efetuadas na geração de uma nova solução.Através desta memória, soluções não são revisitadas, assim, soluções com pior valor da função heurística podem ser visitadas, possibilitando a fuga dos mínimos locais, e o direcionamento para os minímos globais.

As estratégias de busca baseadas em memória visam explorar as boas soluções e as regiões promissoras. Em cada iteração do algoritmo é gerada uma vizinhança de tamanho *N*, que podem conter qualquer solução válida, menos as soluções que estão presentes na lista tabu. Desta vizinhança é escolhida a melhor solução. O critério de aspiração é definido para que seja armazenado na memória apenas soluções melhores e que não estão armazenadas. Assim, através deste critério pode-se escolher uma solução que está presente na lista tabu, ao invés de se escolher um solução de baixa qualidade.

Os tipos de memória possíveis na Busca Tabu são o de curta e de longa duração. Memória de curta duração é o tipo de memória que armazena as melhores soluções por um tempo pequeno. Assim que, novas soluções surgem e são melhores que as armazenadas anteriormente, a solução armazenada é retirada da memória e armazenada a nova solução. A Memória de longa duração, por ser mais longa, armazena um número de soluções maiores que de curta duração. Por isso, as soluções ficam um tempo maior alocadas na lista. Através deste tipo de memória pode-se induzir informações sobre as soluções obtidas durante a busca e assim direcioná-la para um ótimo global.

O algoritmo da Busca Tabu é apresentado a seguir:

Algoritmo 1. Algoritmo básico da Busca Tabu.

```
procedimento BuscaTabu(f(.), N(.), |V|, |T|, BTmax, s)
1  s* ← s;                    {Melhor solução obtida até então}
2  Iter ← 0;                  {Contador do número de iterações}
3  MelhorIter ← 0;            {Iteração mais recente que forneceu s*}
4  T ← ∅;                     {Lista Tabu}
5  enquanto (Critério de parada não satisfeito) faça
6      Iter ← Iter + 1;
7      Seja s' ← s ⊕ m o melhor elemento de V ⊂ N(s) tal que
         o movimento m não seja tabu (m ∉ T) ou f(s') < f(s*);
8      Atualize a lista tabu;
9      s ← s';
10     se (f(s) < f(s*)) então
11         s* ← s;
12         MelhorIter ← Iter;
13     fim-se;
14 fim-enquanto;
15 s ← s*;
16 Retorne s;
fim BuscaTabu;
```

O algoritmo descreve as característica da Busca Tabu apresentadas, porém não utiliza um critério de aspiração. No início de cada iteração é gerada uma vizinhança. Esta vizinhança possui um conjunto *n* soluções, excetuando-se as soluções já vistadas

até então. Neste algorimot, a lista tabu armazena as trocas efetudas pelo algoritmo. Assim, se ao gerar os elementos da vizinhança uma troca estiver na lista tabu a solução que seria gerada através destra troca é retirada da vizinhança e uma nova deve ser gerada. A partir da vizinhança é escolhida a solução com melhor valor da função heurística. Caso esta nova solução seja melhor que a solução corrente, ela passa a ser a solução corrente e é adicionada na lista tabu.

4. Características das Implementações

Nesta Seção serão descritas as características dos três algoritmos implementados pelos grupos 1, 2 e 3, apresentados na introdução. Para cada algoritmo serão descritos os seguintes tópicos: linguagem de programação e especificação da vizinhança, para a Subida de Encosta e Busca Tabu e o critério de aspiração, especificação da lista tabu e os parâmetros do algoritmo, somente para a Busca Tabu.

O algoritmo da Subida de Encosta desenvolvido pelo grupo 1 foi desenvolvido na linguagem Java. Foi implementado um conjunto de classes que possibilitam a leitura das bases de dados com as cidades e a execução da Busca Tabu para o problema do caixeiro viajante. A vizinhança é composta por somente um elemento. Este elemento é gerado de forma aleatória. Caso esta nova solução seja melhor que a solução corrente, ela passa a ser escolhida como a solução corrente. O algoritmo não utiliza nenhuma estrutura para evitar a escolha de soluções repetidas, pois esta é um característica importante da Busca Tabu, e não foi implementada para se obter uma diferenciação entre as técnicas estudadas. O algoritmo tem como parâmetro o numero máximo das iterações, porém pára de executar caso não haja nenhuma mudanção na soluçao corrente.

A algoritmo da Busca Tabu do grupo 1, também foi implementado na linguagem Java e utiliza o memo conjunto de classes do Subida de Encosta. A vizinhança gerada pelo algortimo é aleatória. Duas cidades são escolhidas de forma aleatória, é feita a troca entre elas e um novo estado é gerado e adicionado a nova vizinhança. O critério de aspiração utilizado foi se a melhor solução da vizinhança possui um melhor tour que a solução corrente. Se este critério for superado, a melhor solução da vizinhança deve ser escolhida, mesmo que esteja presente na lista tabu. A lista tabu foi implementada através da estrutura de dados Fila, assim, soluções que foram visitadas há muito tempo são retiradas da lista com o passar da execução. Os parâmetros utilizados pelo algoritmo são o número de iterações que o algoritmo irá executar, *numIter*, o tamanho da lista tabu, *tamList*, e o tamanho da vizinhança, *tamVisit*. O algoritmo executa todas as iterações, mesmo que não consiga efetuar alguma troca com muitas iterações. Os parâmetros ecolhidos para o teste foram:

numIter = 50.000 iterações.

tamList = tamVisit = número de instâncias do problema.

A implementação do algoritmo da Subida de Encosta do grupo 2 foi feita na linguagen C++. A vizinhança no algoritmo é gerada criando todas as permutações de pedaços da seqüência de cidades. Essa seqüência tem tamanho variável num intervalo de [2, 5] visto que para criar permutações de seqüências muito grandes o custo seria muito alto.

Por exemplo:

A seqüência,

1 ; 5 ; 8 ; 3 ; 10 ; 9 ; 2 ; 4 ; 7 ; 6

Com permutações de tamanho 2:

As permutações em seqüência seriam dos números (1 e 5), (8 e 3), (10 e 9), (2 e 4), (7 e 6).

Se fossem de tamanho 3:

Seriam (1, 5 e 8), (3, 10 e 9), (2, 4 e 7) e o 6 não seria permutado (obvio).

De tamanho 4:

Seriam (1, 5, 8 e 3), (10, 9, 2 e 4), (7 e 6).

E assim por diante. O tamanho *n* da seqüência a ser permutada é incrementado cada vez que se esgotaram todas as possibilidades. O numero de possibilidades se da por: ((NdeSequencias)*(TAMdaSeq)!). O algoritmo da Subida de Encosta foi implementado com numero máximo de iterações pois aparentemente entrava em loop infinito em alguns casos e não terminava.

O algoritmo da Busca Tabu também foi implementada na linguagem C++. A vizinhança foi gerada com o mesmo algoritmo utilizado na Subida de Encosta. O Critério de aspiração utilizado foi o de uma troca que está na lista TABU pode ser reutilizada caso seja melhor que o valor da função objetivo atual. A lista TABU foi implementada com uma fila e guarda as trocas efetuadas durante as iterações. Os Parâmetros são MaxTB e MaxIter que representam tamanho máximo da lista TABU e número máximo de iterações sem melhora na função objetivo. E os valores utilizados foram:

MaxTB = 45.
MaxIter = 50000.

O grupo 3 implentou o algoritmo da Subida de Encosta na linguagem C. O algoritmo possui como parâmetros a solução inicial, o número de cidades e a distância entre as cidades. A solução inicial é gerada através de um algoritmo guloso, implementado pelo grupo ou pode ser obtida através de um arquivo texto. A vizinhança possui apenas um elemento e é gerada de forma sequêncial da seguinte forma:

Número de cidades: 5 (A,B,C,D,E)
Vizinhança gerada:

BACDE	CBADE	DBCAE	EBCDA
ACBDE	ADCBE	AECDB	
ABDCE	ABEDC		
ABCED			

Ou seja, 'A' troca com todas as cidades depois dele, depois as cidades voltam à forma inicial (ABCDE), 'B' troca com todas as cidades depois dele e assim por diante. O algoritmo executa enquanto houver mudanças na solução corrente.

O algoritmo da Busca Tabu do grupo 3 também foi implementado na linguagem C. Os parâmetros utilizados são a solução Inicial, número de cidades, dstâncias entre as cidades, número máximo de iterações e o tamanho da lista tabu. A solução inicial pode ser gerada por um algoritmo guloso ou lida de um arquivo texto. O número máximo de iterações e o tamanho da lista tabu são definidos pelo usuário. Os valores utilizados para os teste do número máximo de iterações varia entre 2000 e 30000.A lista tabu é uma fila, tem tamanho fixo, então quando está cheia, a cada novo elemento adicionado à lista, o primeiro (mais antigo) é retirado da lista. São inseridas na lista tabu as trocas efetudas durante a iteração ao invés da solução. A vizinhança gerada pelo algoritmo é feita da mesma forma que o algoritmo da Subida de Encosta. O critério de aspiração utilizado é o seguinte: caso o melhor vizinho encontrado esteja na lista Tabu, mas ele

seja melhor que a melhor função objetiva encontrada até agora, então este melhor vizinho é usado. O algoritmo executa o número de iterações passado como parâmetro pelo usuário, e ao final das iterações o resultado corrente das iterações é apresentado.

5. Resultados

Os algoritmo descritos na seção anterior executaram um conjunto de 40 casos de teste, distribuídos em quatro conjunto de cidades diferentes. As cidades utilizadas nos casos de teste foram obtidas através da base de dados *tsplib*[2]. Foram escolhidos os seguintes *tours*: *dantzig42*, conjunto de 42 cidades com valor de tour ótimo 699, *fri26*, conjunto com 26 cidades com valor de tour ótimo 937, *gr48*, conjunto com 48 cidades com valor de tour ótimo 5046 e *hk48*, conjunto com 48 cidades com valor de tour ótimo 11461. Para cada conjunto de cidades foram realizados 10 casos de teste, sendo que cinco foram gerados aleatoriamente, quatro gerados por um algoritmo guloso e um foi obtido na sequência que foi disponibilizada no arquivo texto.

Foram executados três tipos de comparação. A primeira compara o resultado dos algoritmos de *Hill Climbing* implementados por cada grupo. Esta comparação visa encontrar os melhores resultados obtidos por um algoritmo de subida de encosta entre os grupos. A segunda comparação ocorre entre os algoritmos da Busca Tabu de cada grupo. Igualmente à primeira comparação, estes testes visam encontrar o melhor resultado obtido pela Busca Tabu entre os três grupos. Por fim, será feita uma comparação entre os melhores resultados obtidos entre a comparação dos algoritmos de *Hill Climbingi* e a Busca Tabu, visando obter qual técnica produz melhores resultados para o problema do Caixeiro Viajante.

5.1. Metodologia

Para cada comparação realizada foram executadas as seguintes medidas de avaliação: melhor resultado, resultado médio e tempo médio de execução. Em cada conjunto de cidades é obtido o melhor *tour* entre os dez casos de teste, a média entres os dez teste e o tempo médio da execução de cada caso de teste.

Em cada comparação foi analisado qual algoritmo obteve o melhor resultado final, qual obteve o melhor resultado médio e qual foi o mais rápido.

5.2. Discussão dos Resultados

A primeira comparação realizada análisa os algoritmos da Subida de Encosta implementados. A tabela 1 apresenta um resumo dos resultados obtidos para os três algoritmos.

Tabela 1. Síntese dos 40 casos de teste executados para dos algoritmos de Subida de encosta.

gr48	Grupo 1	Grupo 2	Grupo 3		dantzig42	Grupo 1	Grupo 2	Grupo 3
Média	7156,3	5371,5	7340,5		**Média**	914,7	754,1	924,4
Melhor	6628	5079	6458		**Melhor**	699	699	699
Tempo Medio (ms)	157		13		**Tempo Medio (ms)**	75		10
hk48	**Grupo 1**	**Grupo 2**	**Grupo 3**		**fri26**	**Grupo 1**	**Grupo 2**	**Grupo 3**
Média	15446,7	12347,8	16508,2		**Média**	1175,8	998,3	1115,4
Melhor	14260	11968	14646		**Melhor**	1044	937	957
Tempo Medio (ms)	176		13		**Tempo Medio (ms)**	10		10

A tabela 1 mostra que nos quatro conjuntos de cidades utilizados nos testes, o algoritmo do grupo 2 apresentou os melhores resultados. Este algoritmo conseguiu encontrar a melhor resultado final e também conseguiu uma melhor resultado médio entre os dez casos de teste executados para cada conjunto de cidade. Os melhores resultados encontrados por este algoritmo variaram entre 97% e 100% em relação ao ótimo. O ótimo foi atingido nas bases *dantzig42* e *fri26*. Em relação ao tempo médio, os resultado variou entre 93% e 96% do ótimo. Os outros dois algoritmos apresentaram resultados equivalentes entre si, porém bem inferiores aos apresentados pelo grupo 2. Em relação ao tempo médio de execução, o algoritmo do grupo 3 obteve os melhores resultados, tendo um tempo de execução entre 10 e 13 milisegundos. A partir destes resultados pode-se concluir que um fator importante para a melhora do resultado é a estrutura da vizinhança. Os algoritmos do grupo 1 e 3 implementaram uma vizinhança simples, avaliand um elemento por vez. Enquanto o algoritmo do grupo 2 implementou uma estrutra de vizinhança que buscava um melhor resultado dentre um sequência de vizinhos. Como foi mostrado acima, o algoritmo do grupo 2 obteve resultados melhores que os outros algoritmos implementados.

O segundo cojunto de testes visa comparar os algoritmos de Busca Tabu implementados. A tabela 2 apresenta os resultados obtidos:

Tabela 2. Síntese dos 40 casos de teste executados para os algoritmos da Busca Tabu.

gr48	Grupo 1	Grupo 2	Grupo 3
Média	6114,4	5115,7	5116,3
Melhor	5699	5063	5058
Tempo Medio (ms)	1339		2696

dantzig42	Grupo 1	Grupo 2	Grupo 3
Média	750	701,8	700,7
Melhor	733	699	699
Tempo Medio (ms)	2463		7398

hk48	Grupo 1	Grupo 2	Grupo 3
Média	13073	11573	11718
Melhor	12517	11474	11461
Tempo Medio (ms)	3060		20667

fri26	Grupo 1	Grupo 2	Grupo 3
Média	977,2	937	937
Melhor	937	937	937
Tempo Medio (ms)	1825		853

Nesta segunda comparação, os algoritmos do grupo 2 e 3 apresentaram resultados equivalentes e bem próximos ao *tour* ótimo. Nas bases de dados *dantzig42* e *fri26* estes algoritmos chegaram ao resultado ótimo em todas as 10 execuções. Nas outras duas bases o melhor resultado foi bastante próximo do ótimo, sendo igual ao ótimo para o algoritmo 3. Em relação ao valor médio das 10 execuções o algoritmo 2 obteve resultados muito bons para os quatro conjuntos de cidades, tem estes valores próximos de 99% do valor ótimo. Os resultados médios obtidos pelo algoritmo 3 foram equivalentes em relação ao grupo 2. O algoritmo implementado pelo grupo 1 apresentou resultados piores que os outros. Porém, os resultados apresentados para os quatro conjuntos de cidades foram bons, tendo o melhor valor dos casos de teste variando entre 85% a 90% do valor ótimo. Em relação ao tempo médio de execução, o algoritmo 1 apresentou o melhor resultado para 3 conjuntos de cidades, e o algoritmo 3 apresentou o melhor resultado para 1 conjunto.

Estes resultados mostram como a escolha da lista tabu ajuda a encontrar um resultado melhor. Como exposto na Seção 3, os grupos 2 e 3 guardam na lista tabu os movimentos executados durante a busca. Enquanto o algoritmo do grupo 1 guarda uma solução inteira na lista e utiliza uma comparação com o estado corrente como critério de

aspiração. As diferenças no critério de aspiração não alteraram de forma significativa o resultado final. Os algoritmos do grupo 2 e 3 construiram critérios de aspiração diferentes, porém obtiveram resultados equivalentes.

A terceira comparação utiliza os melhores resultados apresentados nas tabelas 1 e 2 visando obter qual técnica apresenta o melhor resultado para problema do caixeiro viajante. A tabela 3 apresenta os resultados desta comparação

Tabela 3. Comparação entre os resultados da Subida de Encosta e da Busca Tabu

gr48	Subida de Encosta	Busca Tabu
Média	5371,5	5115,7
Melhor	6628	5058

dantzig42	Subida de Encosta	Busca Tabu
Média	754,1	700,7
Melhor	699	699

hk48	Subida de Encosta	Busca Tabu
Média	11968	11573
Melhor	11968	11461

fri26	Subida de Encosta	Busca Tabu
Média	998,3	937
Melhor	937	937

Esta tabela mostra que a Busca Tabu encontrou melhores resultados do que a Subida de Encosta para todos os casos de teste analisados. Tanto o melhor valor encontrando quanto à média dos casos de teste os resultado da Busca Tabu foi melhor. Além disso, percebe-se que além de obter resultados melhores, a Busca Tabu consegue obter resultados bastante próximos ao ótimo global.

6. Conclusões

Este relatório apresentou um conjunto de algoritmos para o problema do Caixeiro Viajante. Estes algoritmos utilizaram a heurística da Subida de Encosta e da metaheurística da Busca Tabu. Os algoritmos foram implementados por três diferentes grupos.

A partir das implementações foram feitos um conjunto de 40 casos de teste, divididos entre 4 diferentes bases de teste. Para cada base estudada foram feitas comparações entre os métodos implementados e entre as heurísticas estudadas.

Em relação aos algoritmos implentados, o algoritmo do grupo 2 apresentou os melhore resultados para a heurística Subida de Encosta. Isto ocorreu devida a escolha da estrutra da vizinhança, que se mostrou um fator importante para resolução do problema. Para a Busca Tabu, os algorimtos 2 e 3 apresentaram os melhores resultados. A grande diferença em relação ao algoritmo do grupo 1 foi à escolha do que é gravado na lista tabu. Através do testes, foi percebido que a busca consegue encontrar melhores resultados quando é gravado informações sobre as trocas efetuadas durante a busca, ao invés de se gravar as soluções completas. Em relação ao critério de aspiração, os resultados mostraram que diferentes critérios podem obter resultados equivalentes.

Em relação à comparação entre as heurísticas, a Busca Tabu encontrou melhores resultados em todos casos de teste executados. Além de obter resultados melhores, a Busca Tabu obteve resultados médios bem próximos aos valores ótimos do problema.

Referências Bibliográficas

[1] BLUM C. e ROLI A. “Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison”, ACM Computing Surveys, Vol. 35, No. 3, September 2003, pp. 268–308.

[2] TSPLIB, disponível em: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/