Otimização por Nuvens de Partículas para o Problema do Caixeiro Viajante

Elizabeth Ferreira Gouvêa Goldbarg
Universidade Federal do Rio Grande do Norte – DIMAp
Natal – Brasil – 59070-900
beth@dimap.ufrn.br

Givanaldo Rocha de Souza
Universidade Federal do Rio Grande do Norte – DIMAp
Natal – Brasil – 59070-900
givanaldo@yahoo.com.br

Marco César Goldbarg
Universidade Federal do Rio Grande do Norte – DIMAp
Natal – Brasil – 59070-900
gold@dimap.ufrn.br

Abstract

This paper presents an algorithm developed under the approach of Particle Swarm for the Traveling Salesman Problem. The proposed technique is based upon Path-relinking operators and a local search method that utilizes inversions of node sequences. The results obtained with the proposed algorithm are compared with the results of another Particle Swarm algorithm presented previously for the same problem. A computational experiment with twenty-nine benchmark instances is reported. It is showed that the method proposed in this paper finds results significantly better than the other algorithm.

Keywords: Traveling Salesman, Particle Swarm, Path Relinking.

Resumo

Nesse artigo é apresentado um algoritmo que se baseia na técnica de Nuvens de Partículas para o Problema do Caixeiro Viajante. A abordagem proposta se baseia em operadores de Path-relinking e busca local através de um método de inversão. Uma comparação é feita com um algoritmo desenvolvido segundo a mesma abordagem. Os experimentos computacionais mostram que o algoritmo proposto apresenta resultados significativamente melhores que o outro algoritmo.

Palavras Chave: Problema do Caixeiro Viajante, Nuvens de Partículas, Path-relinking.

1 Introdução

O termo “Vida Artificial” (ALife, como é conhecido em Inglês) é usado para descrever pesquisas em sistemas feitos pelo homem que possuem algumas das propriedades essenciais da vida real, englobando duas linhas gerais de pesquisa:

- Como as técnicas computacionais podem ajudar quando se estudam os fenômenos biológicos;
- Como as técnicas biológicas podem ajudar em problemas computacionais.

Diversos são os métodos propostos como base na Biologia para a solução de problemas de Otimização Combinatória, como Algoritmos Genéticos (Holland, 1975), Algoritmos Meméticos (Moscato, 1989) e Algoritmos de Colônia de Formigas (Dorigo *et al*., 1996), entre outros. Os algoritmos de otimização por Nuvens de Partículas também se encontram nessa classe. A Otimização por Nuvem de Partículas é uma técnica baseada em população, desenvolvida por um Psicólogo, James Kennedy, e um Engenheiro Eletricista, Russell Eberhart, com base em modelos do comportamento de bandos de pássaros em revoadas (Kennedy e Eberhart, 1995). Reynolds (1987) e Heppner e Grenander (1990) apresentaram modelos de simulações de bando de pássaros. Reynolds estava curioso por causa da estética da coreografia do vôo dos pássaros, e Heppner e Grenander estavam interessados em descobrir a base das regras que permitiam um grande número de pássaros juntarem-se simultaneamente, sempre mudando a direção rapidamente, separando-se e novamente reagrupando-se. Estes cientistas tinham a percepção que processos locais, tais como os modelados por autômatos celulares, poderiam fundamentar a dinâmica de grupo imprevisível do comportamento social dos pássaros. Ambos os modelos se concentravam na manipulação das distâncias inter-individuais, ou seja, a sincronia de comportamento do bando foi pensada como uma função de esforços dos pássaros em manter uma distância ótima entre eles e seus vizinhos.

O conceito de nuvem de partículas originou-se a partir da simulação de um sistema social simplificado. A idéia original foi simular graficamente a "coreografia" de bandos de pássaros ou cardumes de peixes. Contudo, descobriu-se que esse modelo poderia ser usado na otimização de problemas.

Nesse trabalho, a técnica de Nuvens de Partículas é utilizada para solucionar o Problema do Caixeiro Viajante, PCV. Anteriormente, algoritmos com base nessa abordagem para a solução do PCV foram apresentados por Maurice Clerc (2000) e por Wang *et al*. (2003).

O PCV é um problema clássico da Otimização Combinatória que tem sido utilizado por muitos pesquisadores para testar suas idéias algorítmicas. Ele consiste em determinar, num grafo ponderado, um ciclo Hamiltoniano de custo mínimo. Uma revisão do problema pode ser encontrada no trabalho de Gutin e Punnen (2002).

Na Seção 2 é apresentada a abordagem Nuvem de Partículas. A Seção 3 faz uma revisão dos algoritmos propostos para o PCV com base nessa abordagem. Na Seção 4 é apresentado o algoritmo proposto nesse trabalho. Na Seção 5, são relatados os resultados de um experimento computacional que compara o algoritmo proposto com a abordagem de Wang *et al*. (2003). Conclusões e comentários finais são apresentados na Seção 6.

2 Nuvens de Partículas

Suponha o seguinte cenário: um grupo de pássaros voa procurando alimento em uma região qualquer onde há somente uma porção de alimento e nenhum pássaro sabe onde ela está. Entretanto, os pássaros sabem o quão distante estão do alimento em cada iteração. Qual será a melhor estratégia a ser seguida a fim de encontrar a porção de alimento? Talvez, uma boa escolha seja seguir o pássaro que está mais próximo da comida. A abordagem de Nuvens de Partículas toma como base este cenário, utilizando-o para resolver problemas de otimização. Considera-se que cada solução potencial é um pássaro no espaço de soluções, o qual é chamado de partícula. Todas as partículas possuem valores de "adequação" os quais são avaliados pela função-objetivo a ser otimizada. As partículas também têm velocidades as quais direcionam seu vôo. As partículas voam sobre o espaço de soluções possuindo uma tendência de seguir as melhores dentre elas.

No algoritmo clássico, cada partícula tem as seguintes características:

- Possui uma posição e uma velocidade;
- Tem conhecimento da sua posição, e também do valor da função-objetivo para esta posição;
- Tem conhecimento dos seus vizinhos, assim como da melhor posição e do melhor valor da função-objetivo já atingida por cada um deles;

- Guarda sua melhor posição já atingida.

O algoritmo Nuvens de Partículas é iniciado com um grupo de partículas aleatórias (que são as soluções potenciais) e então procura por soluções ótimas atualizando as partículas a cada iteração. Uma partícula é atualizada com base na melhor solução encontrada por ela ao longo da busca, *p_melhor*, e na melhor solução encontrada até o momento na busca por alguma partícula, *melhor_global*. Em cada iteração a partícula pode escolher:

- Seguir seu próprio caminho;
- Seguir em direção à sua melhor posição já encontrada (*p_melhor*);
- Seguir em direção à melhor posição do melhor vizinho (*melhor_global*);

Uma partícula atualiza sua velocidade e posição de acordo com as seguintes equações:

$$v[] = w.v[] + c_1.rand().(p\_melhor[] - p[]) + c_2.rand().(melhor\_global[] - p[]) \quad (1)$$

$$p[] = p[] + v[] \quad (2)$$

onde, v[] é a velocidade da partícula, p[] se refere à partícula corrente (solução), rand() corresponde a um número aleatório entre (0,1), w é chamado de fator de inércia e c_1, c_2 são fatores de aprendizagem, usualmente $c_1 = c_2 = 2$. Os três coeficientes (que representam papéis social/cognitivo) w, c_1 e c_2 respectivamente significam:

- O quanto a partícula confia em si mesma;
- O quanto ela confia na sua experiência;
- O quanto ela confia nos seus vizinhos.

O pseudo-código para um algoritmo de Nuvens de Partículas é mostrado na Figura 1.

```
Para cada partícula
   Inicie partícula;
fim_Para

Faça

   Para cada partícula p[]
      Calcule o valor de adequação de p[]
      Se a adequação de p[] é melhor que a adequação de p_melhor[] então
            p_melhor [] ← p[]
   Fim_Para

   Defina a partícula de melhor adequação como global_melhor[]
   Para cada partícula p[]
      Calcule a velocidade
      Atualize a posição da partícula
   Fim_Para

Enquanto (iterações máximas) ou (critérios de erro mínimo) não são atendidos.
```

Figura 1. Algoritmo clássico segundo a abordagem de Nuvens de Partículas

Há muitas maneiras de se definir uma vizinhança (Kennedy, 1999), mas duas classes podem ser distinguidas:

- Vizinhança "física" (ou geográfica), que leva em conta distâncias definidas entre as partículas. Em geral, as distâncias são computadas a cada passo e tomadas, como vizinhas, as k partículas mais próximas.
- Vizinhança "social", que leva em conta os relacionamentos. Na prática, para cada partícula, sua vizinhança é definida com uma lista de partículas. Portanto, não é necessário calcular distâncias, sendo uma grande vantagem para alguns casos, particularmente para espaços discretos. Uma outra característica é que, no caso da convergência do processo de busca, uma vizinhança social tende a se tornar uma vizinhança física.

Há dois pontos importantes a serem examinados quando se aplica uma heurística de Nuvens de Partículas em problemas de otimização: a representação da solução e a função de adequação. No caso das abordagens onde se utiliza o conceito de vizinhança física, é necessário definir-se uma distância.

3 Nuvens de Partículas para o Problema do Caixeiro Viajante

Nessa seção são apresentados os algoritmos propostos em trabalhos anteriores para o problema do Caixeiro Viajante (Clerc, 2000; Wang *et al*., 2003).

Considere $G = (N, E)$, onde N é o conjunto de vértices e E o conjunto de arestas valoradas. Os nós do grafo são numerados de 1 até n (número de nós). Cada elemento de E consiste em uma tripla $(i, j, w_{i,j})$ com $i, j \in \{1, ..., n\}$ e $w_{i,j} \in \Re$.

São consideradas seqüências com n+1 nós, todos diferentes, exceto o último nó que é igual ao primeiro. Esta seqüência é vista como uma partícula no espaço de soluções. Portanto, o espaço de soluções é definido como o conjunto finito de todas as partículas.

Considere uma partícula $p = (x_1, x_2, ..., x_n, x_{n+1})$, $x_1 = x_{n+1}$. Esta partícula somente é aceita se existirem todos os arcos (x_i, x_{i+1}). A função adequação de uma partícula p é definida como o custo do ciclo formado pelas arestas definidas em p.

Para o cálculo da velocidade, define-se o operador v o qual, quando aplicado à partícula, retorna outra posição. Define-se então uma permutação de n elementos, onde há uma lista de transposições. O comprimento dessa lista é $||v||$. A velocidade é então definida por uma lista de pares de vértices, que indicam uma seqüência de operações 2-troca a ser realizada em uma partícula p, para realizar a movimentação da partícula p. Como exemplo, considere $p = (1, 2, 3, 4, 5, 1)$ e $v = \mathbf{(}(1, 2), (2, 3)\mathbf{)}$. A partícula $p' = p + v$ é definida aplicando a primeira transposição de v em x, a segunda ao resultado, etc. Aplicando v em p, tem-se: (2, 1, 3, 4, 5, 2)

(3, 1, 2, 4, 5, 3) = p'.

A diferença $p_2 - p_1$ é definida como a velocidade v, encontrada por um algoritmo, de forma que aplicando v em p_1 resulta em p_2.

Considere v_1 e v_2 duas velocidades. A fim de computar a soma de duas velocidades, $v_1 \oplus v_2$, considera-se uma lista de transposições que contém primeiro as transposições de v_1, seguidas pelas de v_2.

Na multiplicação de uma constante c, um coeficiente real, por uma velocidade v, há diferentes casos, dependendo do valor de c.

- Caso em que $c = 0 \Rightarrow c.v = \varnothing$.
- Caso em que $c \in (0,1] \Rightarrow$ trunca-se v. Considere $||c.v||$ como o maior inteiro menor ou igual do que $c.||v||$. Então, define-se $c.v$ como a lista formada pelos $\lfloor ||c.v|| \rfloor$ primeiros pares de v.

- Caso em que $c > 1 \Rightarrow$ significa ter $c = k + c'$, $k \in N^*$, $c' \in [0,1)$. Então, define-se
  $c.v = \underbrace{v \oplus v \oplus \ldots v}_{k \text{ vezes}} \oplus c'.v$

A distância é definida por $d(p_1, p_2) = ||p_2 - p_1||$. Para essa distância, tem-se as seguintes propriedades (sendo p_3 uma terceira partícula):

$||p_2 - p_1|| = ||p_1 - p_2||$
$||p_2 - p_1|| = 0 \Leftrightarrow p_1 = p_2$
$||p_2 - p_1|| \leq ||p_2 - p_3|| + ||p_3 - p_1||$

Com as definições acima, pode-se reescrever as equações (1) e (2) como a seguir:

$$v[] = w.v[] \oplus c_1.\text{rand}().(p\_melhor[] - p[]) \oplus c_2.\text{rand}().(global\_melhor[] - p[]) \quad (3)$$

$$p[] = p[] + v[] \quad (4)$$

Neste algoritmo, considera-se $c_1 = c_2$.

O principal objetivo de Clerc (2000) foi mostrar como a abordagem poderia ser utilizada em problemas discretos. Por isso, ele usou os parâmetros mais usuais, não se preocupando em fazer testes para ajuste.

Os parâmetros foram os seguintes:

- $w \in (0,1]$; tipicamente 0.5.
- $c_1 = c_2 \in [0,2]$; tipicamente 2.
- *Número de partículas* = $n - 1$; sendo n o número de nós.
- *Vizinhança* = 4; incluindo a própria partícula (vizinhança social).

O algoritmo é aplicado a instância br17.atsp (Reinelt, 1995), onde é encontrado o valor ótimo.

O algoritmo apresentado em Clerc(2000) é implementado no trabalho de Wang *et al.* (2003) com as seguintes diferenças:

- $w = 0$.
- $c_1 = c_2 \in [0,1]$.
- *Número de partículas* = 100.
- *Vizinhança* = 100; incluindo a própria partícula.

O algoritmo é aplicado a uma instância do TSPLIB (Reinelt, 1995) com 14 nós, onde é encontrado o valor ótimo.

Em ambos os algoritmos, as soluções iniciais (partículas) são geradas aleatoriamente.

4 O Algoritmo Proposto

Nessa seção é descrito o algoritmo heurístico segundo a abordagem de Nuvens de Partículas proposto nesse trabalho para o PCV.

As partículas e a função de adequação têm a mesma representação descrita na Seção 3. São definidas duas velocidades: uma para o caminho próprio da partícula, v_1, e outra para a movimentação da partícula em relação à outra partícula, v_2.

A primeira velocidade é dada por um par (a, b) representando dois índices da partícula. Quando v_1 é aplicada a uma partícula p, a seqüência de nós entre as posições a e b em p é invertida. O tamanho da seqüência varia entre 2 (simulando uma 2-troca entre posições vizinhas) até n-1. As inversões são aplicadas até se encontrar uma solução melhor. No caso, de não existir uma solução melhor, a partícula não é modificada.

A segunda velocidade leva uma partícula *p* de sua posição até a posição de uma outra (*p_melhor* ou *melhor_global*). A estratégia é utilizar um operador de Path-relinking (Glover, 1996). A estratégia Path Relinking utilizada é a *Para Frente e Para Trás* (Forward and Backward): aplica-se simultaneamente o Path Relinking tanto da pior solução para a melhor quanto da melhor para a pior.

Partindo de uma solução inicial p_s, o método calcula todas as trocas possíveis em relação à solução alvo p_t, associando a cada uma delas um custo referente ao ganho ou perda que esta troca irá trazer para a solução no momento atual. Se não houver nenhuma troca a ser feita, o método para.

Após determinar quais trocas devem ser realizadas, seleciona-se a que tem o menor custo associado, sendo que este custo será negativo sempre que melhorar a solução atual. Para realizar a troca, executa-se uma seqüência de operações de 2-troca entre nós vizinhos na solução, até que um vértice seja levado a posição alvo. Este procedimento é realizado nos dois sentidos, com o intuito de explorar um maior número de soluções. Se alguma solução intermediária gerada for melhor que a melhor solução encontrada até o momento, esta será atualizada. A Figura 2 mostra um exemplo deste procedimento.

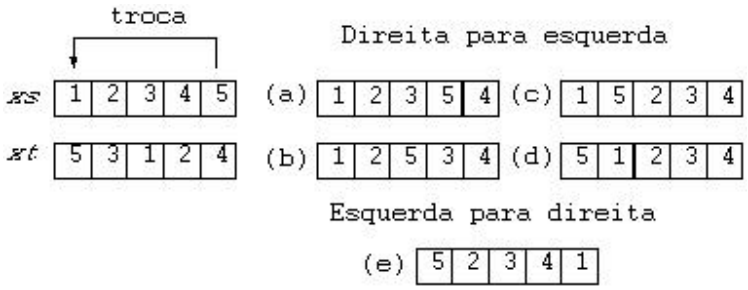

Figura 2: Procedimento de Troca

Como visto no procedimento da Figura 2, ao realizar a troca em ambos os sentidos, tem-se duas possíveis situações finais, a apresentada em (d) e a apresentada em (e). A escolha da situação final que substituirá a solução atual é gulosa, ou seja, seleciona-se a de menor custo.

São definidas probabilidades para a escolha dos caminhos que a partícula deve percorrer. Essas probabilidades são ajustadas ao longo do processamento. Inicialmente, é dada maior probabilidade para movimentações onde a partícula siga seu próprio caminho (v_1). Com o correr do processamento, essa probabilidade é diminuída e é dada maior probabilidade para a partícula seguir em direção a outra (v_2) (*p_melhor* ou *melhor_global*).

Os passos gerais do algoritmo são descritos na Figura 3. Inicialmente, as partículas são geradas pelo método da Inserção do Vizinho Mais Próximo (Goldbarg e Luna, 2005). O critério de parada foi atingir um número máximo de iterações, ou um número máximo de iterações sem melhoria da melhor solução, ou um tempo máximo definido ou atingir o resultado ótimo, caso seja conhecido.

A seguinte configuração foi adotada:

- *Tamanho da População (Número de Partículas) = 20*
- *Número Máximo de Iterações = 2000*
- *Tempo Máximo Definido =*
  - *Instâncias menores que 1000 nós = 60 segundos (1 minuto)*
  - *Instâncias a partir de 1000 nós = 300 segundos (5 minutos)*
- *Iterações sem Melhoria = 20*

Os passos gerais do algoritmo são descritos na Figura 3.

Ler grafo *G*
Definir as probabilidades iniciais de escolha da velocidade
p1 = 60% (escolher próprio caminho – inversão de sequências)
p2 = 30% (escolher ir em direção à *p_melhor*)
p3 = 10% (escolher ir em direção à *melhor_global*)
Gerar solução inicial
Repetir
Calcular a adequação, guardar *p_melhor* e *melhorglobal*.
Para *i* = 1 até número_de_partículas faça
Atualizar posição da partícula p_i
Atualizar as probabilidades:
- p1 = p1 * 0.95
- p2 = p2 * 1.01
- p3 = 100% - (p1+p2)

até que critério de parada seja atendido
Escreve resultado final (*fitness* e *tempo de execução*)

Figura 3. Algoritmo de Nuvens de Partículas para o PCV

5 Testes Computacionais

Com o objetivo de comparação dos resultados obtidos, foi implementada a versão do algoritmo de Wang *et al*. (2003). Os algoritmos foram implementados em linguagem C++, plataforma LINUX, em um Pentium IV (Athlon) com processador de 3.0GHz e memória Ram de 512Mb. Os algoritmos foram testados em vinte e oito instâncias simétricas e uma instância assimétrica do TSPLIB (Reinelt, 1995). Foram feitas dez rodadas independentes de cada algoritmo para cada uma das vinte e nove instâncias do experimento. As Tabelas 1 e 2 mostram os resultados do algoritmo proposto e do algoritmo de Wang *et al.*(2003) para as vinte e oito instâncias simétricas. As colunas das duas tabelas mostram o nome da instância, o valor da solução ótima conhecida, o valor da melhor solução obtida, o valor da média das melhores soluções obtidas e a diferença percentual da média em relação ao ótimo. A Tabela 1 mostra, ainda, o máximo tempo de execução em segundos para alguma das dez execuções do algoritmo proposto. Na Tabela 2, essa coluna é omitida uma vez que o algoritmo de Wang *et al*. (2003) sempre terminou pelo critério de parada tempo máximo de processamento.

Como pode ser verificado através da observação das duas tabelas o algoritmo proposto obtém resultados significativamente melhores tanto em relação à solução mínima, como à média e aos tempos de processamento. A coluna dos afastamentos percentuais em relação à solução ótima mostra que para instâncias com menos de 1000 nós, o máximo afastamento está em torno de 6% e para instâncias com mais de 1000 nós, ele chega a valores no entorno de 12%. Por outro lado, o algoritmo de Wang *et al*. (2003) tem esses valores em torno de 683% e 3202%.

Tabela 1. Resultados do algoritmo proposto para instâncias simétricas do TSPLIB

Instância	Ótimo	Mínimo	Média	% GAP	$T_{máx}$ (s)
eil51	426	428	433,1	1,05	0,51
berlin52	7542	**7542**	7729,7	2,47	0,5
st70	675	681	693	3,03	1
eil76	538	552	564,3	5,20	1,45
pr76	108159	108853	109677,3	1,38	1,27
rat99	1211	1266	1293	6,60	2,18
kroA100	21282	21369	21894,9	3,10	2,44
kroB100	22141	22745	22947,7	3,48	2,3
kroC100	20749	20962	21439,4	3,39	2,3
kroD100	21294	21836	22053,7	3,14	2,5
kroE100	22068	22514	22912	3,82	2,07
rd100	7910	8083	8285,4	4,60	2,67
eil101	629	659	667,7	6,28	2,37
lin105	14379	**14379**	14629,6	1,75	2,98
pr107	44303	44621	45076,7	1,75	2,55
pr124	59030	59087	59860,4	1,56	3,79
bier127	118282	119480	123257,4	4,43	4,56
ch130	6110	6304	6412,3	4,78	4,55
pr136	96772	99773	101698,2	4,74	4,44
pr144	58537	58623	58947,5	0,48	4,75
ch150	6528	6886	6953,1	6,40	6,23
kroB150	26130	26590	27152,2	4,42	5,08
pr152	73682	74412	74912	1,38	6,56
u159	42080	43845	44629,1	6,10	6,79
pr1002	259045	284580	287783,9	11,19	324,44
u1060	224094	247108	249229,7	11,16	326,87
vm1084	239297	264108	266509,8	11,33	338,45
pcb1173	56892	63340	63960,4	12,36	344,24

A Figura 4 ilustra graficamente a diferença entre as diferenças percentuais obtidas pelos dois algoritmos.

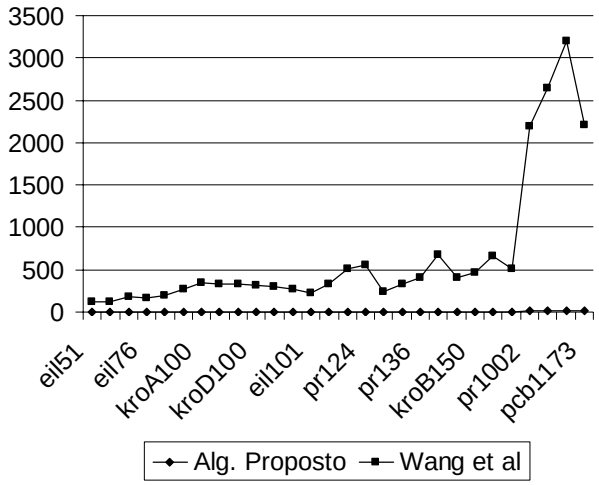

Figura 4. Comparação das diferenças percentuais

Tabela 2. Resultados do algoritmo de Wang *et al.* (2003) para instâncias simétricas do TSPLIB

Instância	Ótimo	Mínimo	Média	% GAP
eil51	426	777	913,4	114
berlin52	7542	14214	15942,8	115
st70	675	1814	1950,5	183
eil76	538	1162	1385,9	162
pr76	108159	276856	321271,8	192
rat99	1211	3897	4468,1	276
kroA100	21282	86598	97475,2	351
kroB100	22141	80392	93382,6	327
kroC100	20749	82039	91195,8	332
kroD100	21294	77946	87888,4	308
kroE100	22068	79852	89718,9	303
rd100	7910	28727	30532,1	274
eil101	629	1852	2067,9	230
lin105	14379	55621	64084	336
pr107	44303	221729	275346,6	515
pr124	59030	288190	383670,4	557
bier127	118282	359424	405681,7	241
ch130	6110	23348	26731,1	333
pr136	96772	419592	483272,2	399
pr144	58537	403258	466902,4	683
ch150	6528	30750	33403,1	407
kroB150	26130	130003	148892,1	464
pr152	73682	472320	568315,8	657
u159	42080	232635	262582,9	516
pr1002	259045	5808654	5957914	2194
u1060	224094	6084374	6155638	2646
vm1084	239297	7719324	7878503	3202
pcb1173	56892	1273117	1315801	2210

A Tabela 3 mostra os resultados obtidos pelos dois algoritmos para a instância assimétrica de teste. Para essa instância, também pode ser verificado o desempenho superior do algoritmo proposto no que se refere à média das soluções nas dez execuções dos algoritmos e o tempo de processamento,

Tabela 3. Resultados do algoritmo proposto para instâncias simétricas do TSPLIB

Algoritmo	Instância	Ótimo	Mínimo	Média	% GAP	$T_{máx}$ (s)
Proposto	br17 (ATSP)	39	**39**	**39**	0	< 0,01
Wang *et al.*(2003)	br17 (ATSP)	39	39	47,4	16,66	60,04

6 Conclusões

Esse trabalho apresentou um novo algoritmo baseado na técnica de Nuvens de Partículas para o Problema do Caixeiro Viajante. Até onde é de conhecimento dos autores, apenas dois trabalhos com o mesmo objetivo haviam sido publicados, onde eram apresentados resultados para uma única instância. Nesse trabalho, é apresentado um experimento computacional com vinte e nove instâncias de um banco de

instâncias do PCV, onde é mostrado que o algoritmo proposto encontra soluções bastante superiores às obtidas nas abordagens anteriores.

Como trabalhos futuros, pretende-se estender a aplicação da técnica utilizando outros operadores de Path-relinking e busca local, aplicando-os às instâncias assimétricas e a problemas correlatos ao Problema do Caixeiro Viajante.

7 Referências Bibliográficas

CLERC, M. *Discrete Particle Swarm optimization: Illustrated by the Traveling Salesman Problem*. 2000 Artigo disponível em http://clerc.maurice.free.fr/pso/pso_tsp/Discrete_PSO_TSP.htm. Acessado em 16/11/2004.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. The ant system: Optimization by a colony of cooperating agents, *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, v. 26, 29-41, 1996.

GLOVER, F. Tabu search and adaptive memory programming – Advances, applications and challenges, In: R. S. Barr, R. V. Helgason, J. L. Kennington (Eds.), *Interfaces in Computer Sciences and Operations Research*, 1-75, Kluwer, 1996.

GOLDBARG, M. C. ; LUNA, H. P. L. *Programação Linear e Otimização Combinatória, modelos e algoritmos*, Ed. Campus, 2ª Edição, 2005.

GUTIN, G.; PUNNEN, A. P. (Ed.) *Traveling salesman problem and its variations*. Kluwer Academic Publishers, 2002.

HEPPNER, F.; GRENANDER, U. (1990). A stochastic nonlinear model for coordinated bird flocks. In: S. Krasner (Ed.), *The Ubiquity of Caos*, AAAS Publications, Washington, DC, 1990.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, 1975.

KENNEDY, J.; EBERHART, R. Particle Swarm Optimization, *Proceedings of the IEEE International Conference on Neural Networks*, v. 4, Perth, Australia, 1942-1948, 1995.

KENNEDY, J. Small worlds and mega-minds: Effects of neighborhood topology on Particle Swarm performance. *Congress on Evolutionary Computation*, Washington, DC, IEEE, 1999.

MOSCATO, P. On evolution, search, optimization, *Genetic Algorithms and martial arts: Towards Memetic Algorithms*, Caltech Concurrent Computation Program, C3P Report 826, 1989.

REINELT, G. *TSPLIB*, 1995. Disponível em: <http://www.iwr.uni-heidelberg.de/iwr/comopt/software/TSPLIB95/>

REYNOLDS, C. W. Flocks, Herds and Schools: a Distributed Behavioral Model, *Computer Graphics*, v.21, n. 4, 24-34, 1987.

WANG, K.-P.; HUANG, L.; ZHOU, C.-G.; PANG, W. Particle swarm optimization for Traveling Salesman Problem. *Proceedings of the Second International Conference on Machine Learning and Cybernetics*, Xi'an, China, 1583-1585, 2003.