PSO PARA OTIMIZAÇÃO COMBINATÓRIA: UMA ANÁLISE DA EQUAÇÃO DE ATUALIZAÇÃO DA VELOCIDADE DAS PARTÍCULAS

Stefan Campana Fuchs
Universidade Tecnológica Federal do Paraná
Departamento de Informática
Av. Sete de Setembro, 3165, Centro, Curitiba-PR, CEP 80230-901
stefan1234@gmail.com

Myriam Regattieri Delgado, Ricardo Lüders
Universidade Tecnológica Federal do Paraná
Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial
Av. Sete de Setembro, 3165, Centro, Curitiba-PR, CEP 80230-901
myriamdelg@utfpr.edu.br, luders@utfpr.edu.br

RESUMO

A Otimização por Enxame de Partículas (Particle Swarm Optimization ou PSO) é inspirada no comportamento emergente de bandos de aves. Esta técnica foi originalmente desenvolvida para otimização contínua. Entretanto, sua utilização em problemas de otimização em espaços de busca discretos tem proporcionado bons resultados em trabalhos recentes. Neste artigo, o esquema de codificação de partículas do PSO utiliza permutações para resolver um problema de alocação de tarefas (Task Assignment Problem ou TAP). Neste caso, a notação utilizada para descrever a movimentação das partículas é adaptada para tratar aspectos de otimização combinatória. Porém, na literatura em geral esta notação não é clara em relação à ordem e tipos de movimentos utilizados. Este artigo mostra que esta definição é importante e afeta de maneira significativa os resultados. Para tanto, são realizados experimentos com diversas implementações do PSO, levando em conta o tipo e ordem dos movimentos de atualização das partículas, especialmente para movimentações baseadas em path-relinking.

PALAVARAS CHAVE. Otimização combinatória, Otimização por enxame de partículas, Análise experimental, Alocação de tarefas.

ABSTRACT

Particle Swarm Optimization (PSO) is inspired on emergent behavior of flocks of birds. This technique was originally designed for continuous optimization. Nevertheless, it has been applied to discrete optimization problems with good results in recent studies as well. In this paper, the PSO encoding scheme utilizes permutations for solving a Task Assignment Problem (TAP). In this case, the notation used to update a particle is adjusted to deal with combinatorial optimization issues. However, the notation found in the literature is not quite clear about sequences and types of particle updates. We claim that this is an important issue to be considered and changes a lot the obtained results. This is accomplished by carrying experiments with several PSO implementations taking in account different types and sequences of particle updates. Particularly, those based on path-relinking technique.

KEYWORDS. Combinatorial optimization, Particle swarm optimization, Experimental analysis, Task allocation.

1 Introdução

Os problemas de otimização combinatória representam uma importante área de pesquisa devido à grande aplicação prática. Em geral, consideram-se problemas de otimização em relação à alocação, sequenciamento, ou distribuição de itens ou objetos num espaço contável de possibilidades. Devido à natureza combinatória, esses problemas geram espaços de busca finitos, porém de elevada cardinalidade. Dentre as diversas abordagens utilizadas na solução desses problemas, destacam-se as técnicas metaheurísticas (Talbi, 2009) e, em especial, os algoritmos de Inteligência Coletiva (Bonabeau et al., 1999).

O termo inteligência coletiva (*Swarm Intelligence*), foi cunhado no final dos anos 1980 para referenciar sistemas de robótica celular nos quais uma coleção de agentes interage localmente em um ambiente. Esta interaçao é baseada em regras simples de comportamento (Beni, 1988) *apud* DeCastro (2006). Atualmente, o termo está sendo utilizado para descrever qualquer tentativa de projetar algoritmos, inspirada pelo comportamento coletivo de organismos sociais, desde colônias de insetos até sociedades humanas. Dentre os representantes mais conhecidos, destaque para a Colônia de Formigas (Dorigo et al., 1996) e Enxame de Partículas (Kennedy e Eberhart, 1995). A técnica de Otimização do PSO (*Particle Swarm Optimization*) é baseada em população, e foi proposta em (Kennedy e Eberhart, 1995), com base no comportamento de pássaros em revoadas. O PSO foi inicialmente concebido para otimização em espaços de busca contínuos. Existem variações do PSO que utilizam diferentes estratégias para atualização das partículas. Basicamente, as partículas são atualizadas por uma equação de velocidade composta por três componentes: inercial, cognitiva e social.

Mais recentemente, tem-se observado na literatura a aplicação do PSO em problemas combinatórios. A primeira versão do PSO para problemas discretos foi apresentada em Kennedy e Eberhart (1997). Neste trabalho, Kennedy e Eberhart propuseram um conjunto de valores binários para codificar cada partícula. A velocidade das partículas foi, então, definida como uma matriz de probabilidades de mudança de 1 para 0 e vice-versa. Em Hu (2003) a velocidade foi definida em uma matriz com valores no intervalo [0,1] como sendo uma chance (probabilidade) de mudança na sequência numérica das posições. Entretanto, apenas a componente social é considerada no momento de movimentação da partícula. Já em Wang et al. (2003) e Clerc (2004) as principais alterações sugeridas são: o vetor posição é substituído por um conjunto de valores discretos, e o vetor velocidade é alterado para uma lista de permutações que mudam a posição da partícula. A principal diferença das abordagens apresentadas em Wang et al. (2003) e Clerc (2004) diz respeito à representação da velocidade, uma vez que permutações em Clerc (2004) estão associadas a mudanças diretas de valores (e não de posições como em Wang et al. (2003)). Alguns exemplos do uso de PSO com o Problema do Caixeiro Viajante (TSP) são descritos em Pang (2004); Machado e Lopes (2005); Rosendo (2010). Em Pang (2004), os autores propuseram um mapeamento do espaço contínuo para o discreto baseado no método do maior valor de prioridade. Codificação inteira e velocidade em função da distância de Hamming são consideradas em Machado e Lopes (2005). Finalmente, o PSO é usado em Rosendo (2010) para resolver o TSP com um procedimento de busca local substituindo a componente inercial da velocidade.

Apesar das atualizações da velocidade e da posição da partícula serem bem definidas em aplicações com espaço de busca contínuo, a forma com que as partículas são atualizadas no PSO combinatório ainda carece de formalização e muitas vezes fica restrita

ao tipo de problema tratado. Em Rosendo (2010) a técnica do *path-relinking* é usada para mover uma partícula de uma posição a outra. A componente inercial da velocidade é substituída por uma busca local e a ordem de aplicação das componentes é definida como sendo social seguida pela cognitiva, com base nos experimentos realizados. Embora a ordem de aplicação das componentes tenha sido destacada no trabalho de Rosendo (2010), esta ordem é desconsiderada em grande parte dos trabalhos de PSO para otimização combinatória. Além disso, não é do conhecimento dos autores, nenhum trabalho recente publicado que considere a sequência das trocas realizadas dentro da lista de permutações que representa cada componente da velocidade.

A proposta deste artigo é apresentar uma formalização inicial para a equação de atualização da velocidade, juntamente com uma análise experimental dos diferentes parâmetros que influenciam o PSO no contexto da otimização combinatória. Em especial, a sequência de movimentos dentro do *path-relinking*. Os resultados são obtidos para um problema de otimização conhecido como TAP (*Task Assignment Problem*). Basicamente, esse é o problema de alocar n recursos distintos a n tarefas distintas, sendo que nenhuma tarefa fica sem recurso. Ou seja, o resultado da alocação é uma permutação de n elementos distintos. Esse problema foi escolhido pois gera um espaço de busca que cresce rapidamente com o número de tarefas. Além disso, vários casos de teste podem ser construídos, pois a geração de soluções factíveis é trivial, assim como a avaliação do *fitness* de cada solução. Ainda assim, captura-se o comportamento do algoritmo (PSO) que se deseja analisar.

A Seção 2 deste artigo apresenta detalhes do algoritmo clássico do PSO assim como discute algumas abordagens para problemas discretos. A Seção 3 descreve o algoritmo PSO utilizado neste trabalho, destacando a técnica de *path-relinking* e o ajuste dos parâmetros do algoritmo. A Seção 4 apresenta os resultados para diferentes dimensões do problema e parâmetros utilizados, cujas conclusões são apresentadas na Seção 5.

2 Algoritmo PSO: da versão contínua para a versão discreta

PSO é uma técnica desenvolvida por James Kennedy e Russell Eberhart (Kennedy e Eberhart, 1995). É inspirado no comportamento de aves em bandos onde as soluções para um determinado problema de otimização, chamadas partículas, "voam" (como aves) através de um espaço de busca multidimensional. Tal como os algoritmos genéticos, o PSO pode ser classificado como um paradigma bio-inspirado. O algoritmo de enxame de partículas ajusta trajetórias de uma população de "partículas" utilizando informações sobre o melhor desempenho de cada partícula e o melhor desempenho de seus vizinhos (vide Kennedy e Eberhart (1997) e Kennedy e Eberhart (1995) para mais detalhes).

Considere uma partícula $\mathbf{p}^t = (p_1^t, p_2^t, ..., p_z^t)$ posicionada no espaço de busca $\mathbf{R}^z$ no tempo discreto t. A partícula $\mathbf{p}^t$ é movimentada por um vetor de velocidade $\mathbf{v}^{t+1} = (v_1^{t+1}, ..., v_z^{t+1})$ de acordo com (1).

$$\mathbf{p}^{t+1} = \mathbf{p}^t + \mathbf{v}^{t+1}. \tag{1}$$

Usualmente, a atualização do vetor velocidade é realizada para cada elemento j com base em três componentes conforme definido em (2).

$$v_j^{t+1} = w \cdot v_j^t + (c_1 r_1) \cdot v_{j(cogn)}^t + (c_2 r_2) \cdot v_{j(social)}^t, \tag{2}$$

onde v_j^t corresponde à j-ésima componente inercial (velocidade atual), $v_{j(cogn)}^t = (best_j - p_j^t)$ corresponde à j-ésima componente cognitiva com $best_j$ definido como o j-ésimo elemento da melhor posição alcançada até o momento pela partícula, $v_{j(social)}^t = (best_{j(global)} - p_j^t)$ corresponde à j-ésima componente social com $best_{j(global)}$ definido como o j-ésimo elemento da melhor posição alcançada até o momento pelo enxame (ou por um subconjunto de partículas definido por uma estrutura de vizinhança específica). Os coeficientes w, c_1 e c_2 são associados às componentes inercial, cognitiva e social da velocidade, respectivamente. Eles definem o quanto a partícula confia em seu movimento anterior, na sua própria história e no conjunto de partículas, respectivamente. Os termos r_1 e r_2 são números aleatórios com distribuição uniforme.

O algoritmo PSO inicia com um enxame aleatório, ou seja, posição e velocidade de cada partícula são geradas aleatoriamente, e são atualizadas a cada iteração ($best_j$ é atualizado para cada partícula e $best_{j(global)}$ para todo o enxame ou um subconjunto dependendo da estrutura de vizinhança adotada). Quando um máximo de T iterações é alcançado, $best_{j(global)}$ representa a melhor solução encontrada pelo enxame.

Embora o PSO tenha sido inicialmente desenvolvido para problemas de otimização contínua, resultados interessantes têm sido obtidos para otimização combinatória (ver Poli (2008) para uma revisão em algumas aplicações de PSO). Na maioria dos casos, as equações originais são mantidas. A primeira versão do PSO para problemas discretos foi apresentada em Kennedy e Eberhart (1997). Neste trabalho, se $v_j^{t+1} = 0.20$, então existe 20% de chance de p_j^{t+1} se tornar um e 80% de chance de se tornar zero. Se a melhor posição anterior tem um zero no j-ésimo elemento (i.e. $best_j = 0$), então $(best_j - p_j^t)$ pode ser calculada como -1 ou 0 e usada para ponderar a mudança na probabilidade de v_j^{t+1} na próxima iteração. A Equação (2) permanence inalterada, exceto que agora cada valor considerado no j-ésimo elemento da partícula pertence ao conjunto $\{0, 1\}$ e v_j^{t+1} deve ser restrita ao intervalo [0, 1] de modo a representar uma medidade de probabilidade. Desta forma, a dimensão de um movimento da partícula pode ser vista como o número de bits alterados de uma iteração para outra, em um espaço de busca binário. Uma partícula não se move, se nenhum bit é alternado ao passo que executa um movimento completo se todos os bits são alternados.

Hu e colaboradores (Hu, 2003) desenvolveram um PSO discreto, que codifica as partículas como sequências numéricas de posições, e definem o movimento de partículas como uma permutação na sequência dessas posições. No contexto do problema tratado (mais especificamente, o problema das n-rainhas), a velocidade foi definida como uma possibilidade de mudança na sequência numérica das posições. A Equação (2) foi novamente preservada neste algoritmo e a velocidade também foi normalizada para o intervalo [0, 1]. Portanto, as partículas sofrem alterações de posição em cada iteração da busca heurística de acordo com a probabilidade determinada pela velocidade. Se a decisão é por realizar uma permutação, busca-se na melhor partícula do enxame qual a posição do referido valor para que a troca possa ser realizada levando a partícula na "direção" da melhor partícula até o momento.

Em Rosendo (2010) e Goldbarg et al. (2006), a técnica *path-relinking* é usada para mover uma partícula de uma posição a outra. O *path-relinking* é uma técnica de intensificação cuja ideia foi originalmente proposta por Glover no contexto dos métodos de programação com melhores regras de decisão para problemas de *job shop* (Glover et al.,

1999). De um modo geral, esta estratégia gera um caminho entre duas soluções. Assumindo uma solução origem $\mathbf{p}^t$ e uma solução destino $\mathbf{p}^{t+1}$, um caminho representa uma sequência de mudanças (passos) $\mathbf{p}^{t0}, \mathbf{p}^{t1} \cdots, \mathbf{p}^{tm}$, com $\mathbf{p}^t = \mathbf{p}^{t0}$ e $\mathbf{p}^{tm} = \mathbf{p}^{t+1}$, tal que $\mathbf{p}^{tk+1}$ é obtido de $\mathbf{p}^{tk}$ por meio de movimentos que reduzem a distância entre a origem e o destino. Esta ideia é usada em alguns trabalhos de otimização em espaços discretos para redefinir a velocidade em termos da posição de duas partículas $\mathbf{p}_a$ e $\mathbf{p}_b$ como $v = \mathbf{p}_a - \mathbf{p}_b$ ou $\mathbf{p}_a = \mathbf{p}_b + v$, ou seja, aplicando v a $\mathbf{p}_b$ para obter $\mathbf{p}_a$. Estes movimentos serão detalhados na próxima seção com base em uma notação específica para PSO em problemas de otimização combinatória.

3 PSO Combinatório

Esta seção apresenta o algoritmo utilizado neste artigo. Primeiramente são apresentados os detalhes da codificação adotada e características da movimentação realizada a partir de uma lista de permutações. Na sequência, o pseudo-código do algoritmo é apresentado e discutido.

3.1 Detalhes da codificação e movimentação

Nesta seção, a notação utilizada para adaptar o PSO a um problema combinatório é detalhada a partir do problema TAP de alocação de tarefas. Uma partícula é representada por uma lista de valores que indicam a solução do problema a ser otimizado. Uma solução do problema TAP é representada em uma partícula da seguinte forma: cada posição indica um recurso, e o valor de cada posição determina a tarefa para a qual o recurso foi alocado. Por exemplo, dada uma partícula $\mathbf{p}^t = (11, 22, 33, 44, 55)$. a primeira posição indica que o primeiro recurso foi alocado para a tarefa "11", a segunda posição indica que o segundo recurso foi alocado para a tarefa "22", e assim por diante.

A velocidade é representada por uma lista de permutações necessárias para mover a partícula em uma nova direção. O movimento da partícula baseia-se em (3).

$$\mathbf{p}^{t+1} = v^{t+1}\left(\mathbf{p}^t\right) \tag{3}$$

onde a velocidade v^{t+1} é um operador representado por uma lista de pares de índices indicando quais elementos de $\mathbf{p}^t$ devem ser trocados. Por exemplo, assumindo que a solução do problema é uma permutação de cinco elementos distintos dados pelo conjunto $\{11, 22, 33, 44, 55\}$, a movimentação da partícula $\mathbf{p}^t = (11, 22, 33, 44, 55)$ sujeita à velocidade $v^{t+1} = \{(1, 2), (2, 3)\}$, resulta na partícula $\mathbf{p}^{t+1} = (22, 33, 11, 44, 55)$.

Considerando-se a técnica de *path-relinking* discutida na Seção 2, define-se agora como uma partícula $\mathbf{p}^t = (22, 33, 11, 55, 44)$ pode ser movimentada para uma nova posição $\mathbf{p}^{t+1} = (11, 22, 33, 44, 55)$ usando uma lista de três permutações. Assuma $v^{t+1} = \{v_1^{t+1}, v_2^{t+1}, v_3^{t+1}\}$ e um caminho $\mathbf{p}^{t0}, \mathbf{p}^{t1}, \mathbf{p}^{t2}, \mathbf{p}^{t3}$ tal que $\mathbf{p}^{t(k+1)} = v_{k+1}^{t+1}\left(\mathbf{p}^{tk}\right)$, $k = 0, 1, 2$. Como $p_1^{t+1} = p_3^t = 11$, $v_1^{t+1} = (1, 3)$ e $\mathbf{p}^{t1} = (\mathbf{11}, 33, \mathbf{22}, 55, 44)$. De forma análoga, $v_2^{t+1} = (2, 3)$, $\mathbf{p}^{t2} = (11, \mathbf{22}, \mathbf{33}, 55, 44)$, $v_3^{t+1} = (4, 5)$ e $\mathbf{p}^{t3} = (11, 22, 33, \mathbf{44}, \mathbf{55})$. Então, $v^{t+1} = \{(1, 3), (2, 3), (4, 5)\}$.

Este procedimento pode ser generalizado, considerando as componentes inercial, cognitiva e social da velocidade na seguinte ordem, sendo $\lfloor \cdot \rfloor$ a função piso e $|\cdot|$ a cardinalidade:

1. $u = \lfloor w \cdot |v^t| \rfloor$ permutações para a componente inercial v^t;

2. $q = \lfloor (r_1 c_1) \cdot |v^t_{(cogn)}| \rfloor$ permutações para a componente cognitiva $v^t_{(cogn)}$, necessárias para mover $\mathbf{p}^t$ na direção da melhor posição alcançada pela partícula até o momento;
3. $s = \lfloor (r_2 c_2) \cdot |v^t_{(social)}| \rfloor$ permutações para a componente social $v^t_{(social)}$ necessárias para mover $\mathbf{p}^t$ na direção da melhor posição alcançada pelo enxame até o momento.

A velocidade (4) é então definida como

$$v^{t+1} = \{v_1^{t+1}, \cdots, v_u^{t+1}, v_{(u+1)}^{t+1}, \cdots, v_{(u+q)}^{t+1}, v_{(u+q+1)}^{t+1}, \cdots, v_{(u+q+s)}^{t+1}\} \quad (4)$$

onde os totais de permutações u, q e s são afetados pelos coeficientes de w, c_1 e c_2. Estes coeficientes definem quantas permutações de cada componente da velocidade serão aplicadas. Se u, q e s são maiores do que a cardinalidade da componente correspondente, então a lista inteira de permutações é aplicada tantas vezes quanto necessário.

Alguns aspectos são importantes de serem observados em (4), conforme mostra a Figura 1.

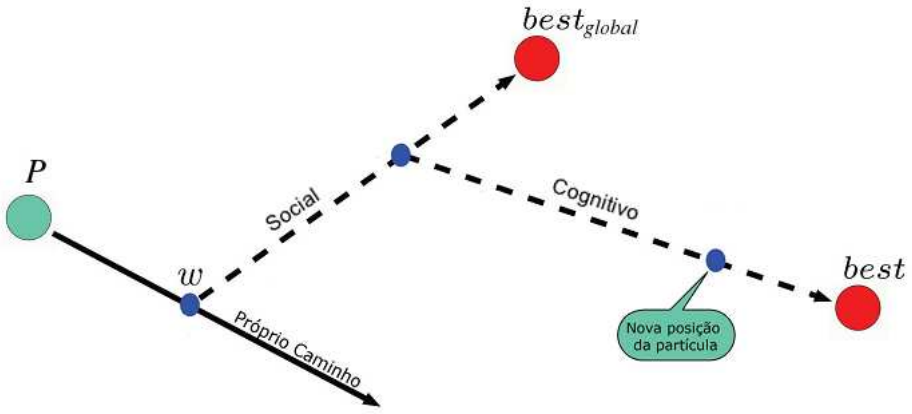

Figura 1. *Path-relinking*: movimento devido às três componentes da velocidade [Adaptado de Rosendo (2010)].

Tanto (4) quanto a Figura 1 consideram que a ordem de aplicação das componentes foi pré-definida como: inercial, social e cognitiva. No espaço contínuo, a resultante não depende da ordem de soma de cada componente, mas no espaço discreto, em especial o da otimização combinatória, esta ordem é relevante. Um outro aspecto importante a ser observado, diz respeito ao tempo de atualização da posição da partícula. No PSO contínuo, as três componentes da velocidade são considerados conjuntamente, resultando num novo vetor velocidade, o qual é utilizado para atualizar a posição da partícula. No caso de otimização combinatória, uma vez que uma componente é calculada há duas opções para a atualização da posição da partícula: (i) calculam-se todas as componentes considerando a posição original da partícula para em seguida movimentá-la numa ordem pré-estabelecida das componentes calculadas; (ii) cada componente é aplicada com base no resultado da aplicação da componente anterior numa ordem também pré-estabelecida. O primeiro caso é denominado movimentação sem atualização (status da atualização = Falso) e o segundo caso é chamado de movimentação com atualização (ou seja, há atualização da posição da partícula antes de cada cálculo). A Figura 1 ilustra o segundo caso (movimentação com status de atualização=Verdadeiro), onde o cálculo da componente social foi feito após a

movimentação da partícula pela componente inercial, e o cálculo da cognitiva foi feito após a movimentação pela componente social. Por fim, outro aspecto fundamental, está relacionado à sequência de operações que levarão a partícula de uma posição a outra dentro do *path-relinking* realizado por cada componente da velocidade. No exemplo mostrado anteriormente nesta seção, temos que a lista completa de $m = 3$ permutações, permite à partícula, atingir o alvo. No entanto, quando o movimento não se realiza por completo (como é o caso dos movimentos descritos na Equação 4), a sequência das trocas realizadas pode gerar resultados diferentes. Assim, é essencial que a sequência de trocas dentro de cada *path-relinking* seja também investigada. Estes aspectos serão analisados na Seção 4.

3.2 Algoritmo

O pseudo-código mostrado no Algoritmo 1 ilustra os passos principais da abordagem proposta neste trabalho.

Algorithm 1 Algoritmo do PSO discreto utilizado

1: Definir: *ordem* das componentes, *atualização* e tipo de path relinking (*sequência*)
2: Inicializa o enxame com S partículas aleatórias
3: Avalia o fitness de cada partícula
4: Atualiza o $\mathbf{best}_{global}$ com o melhor fitness
5: t=1
6: **while** $t \leq T$ **do**
7: **for** Cada partícula $\mathbf{p}^t = (p_1^t, p_2^t, ..., p_N^t)$ **do**
8: Calcula *primeira* componente da velocidade
9: **if** *atualização* = Verdadeiro **then**
10: Aplica numa determinada *sequência*, *primeira* componente da velocidade
11: **end if**
12: Calcula *segunda* componente da velocidade
13: **if** *atualização* = Verdadeiro **then**
14: Aplica numa determinada *sequência*, *segunda* componente da velocidade
15: **end if**
16: Calcula *terceira* componente da velocidade
17: **if** *atualização* = Verdadeiro **then**
18: Aplica numa determinada *sequência*, *terceira* componente da velocidade
19: **end if**
20: **if** *atualização* = Falso **then**
21: Aplica numa determinada *sequência*, *primeira*, *segunda* e *terceira* componente da velocidade
22: **end if**
23: Atualiza o fitness da partícula $\mathbf{p}^{t+1}$
24: Atualiza a melhor posição da partícula (Melhor entre $\mathbf{best}$ e $\mathbf{p}^{t+1}$)
25: Atualiza a melhor posição global (Melhor entre $\mathbf{best}_{global}$ e $\mathbf{best}$))
26: **end for**
27: $t = t + 1$
28: **end while**

O algorimto inicia com a definição (pelo usuário) das características específicas para o contexto da otimização combinatória. Primeiramente, define-se a ordem de

aplicação de cada componente da velocidade (inercial, cognitiva e social). Posteriormente, define-se *status* de atualização da posição da partícula (Verdadeiro ou Falso, ou seja, com ou sem atualização entre os cálculos de cada componente). Por fim, define-se o tipo de *path-relinking* a ser utilizado, ou seja, qual a sequência de operações a ser realizada na movimentação realizada por cada componente da velocidade. Na seção 3.3, cada tipo será detalhado. A seguir, o algoritmo efetua os passos de um PSO tradicional, ou seja inicialização (aleatória) das posições e velocidades das partículas que compõem o enxame, cálculo do *fitness* e definição da melhor solução do enxame. Após estes cálculos, o algoritmo entra no laço principal. No laço de repetição, os cálculos de cada componente da velocidade podem ou não ser intercalados com movimentações da partícula, dependendo do status da atualização. Caso o status seja Verdadeiro, cada cálculo é precedido de uma movimentação realizada pela componente calculada anteriormente. Caso o status seja Falso, a atualização da posição da partícula só é executada depois dos cálculos de todas as componentes (neste caso, os cálculos consideram a posição inicial da partícula). A seguir, o *fitness* é recalculado, assim como a atualização das memórias (de cada partícula e do enxame). Este processo representa uma iteração do algoritmo que é repetido até que a condição de parada seja atingida.

3.3 Tipos de *path-relinking*

Todos os tipos de *path-relinking* utilizados neste trabalho estão baseados na comparação de uma partícula qualquer $\mathbf{p}^t$ com uma partícula de referência $\mathbf{p}^t_{target}$ que pode ser a posição $\mathbf{p}^{t-1}$ usada na última movimentação no caso da componente inercial, $\mathbf{best}$ no caso da componente cognitiva ou $\mathbf{best}_{global}$ no caso da componente social. Quando uma diferença é encontrada, numa determinada posição k da partícula $\mathbf{p}^t$, busca-se qual permutação deve ser feita em $\mathbf{p}^t$ para corrigir essa posição. Ou seja, busca-se qual posição c em $\mathbf{p}^t$ tem um elemento a ser permutado com o elemento da posição k para que na posição k, $\mathbf{p}^t$ e $\mathbf{p}^t_{target}$ fiquem iguais. Vale lembrar que no caso do TAP abordado neste artigo há uma relação de um para um entre trabalhadores e tarefas. Portanto, não há valores repetidos nem faltando, ou seja, todos os números da sequência aparecem exatamente uma vez no vetor de cada partícula.

Tipos de *path-relinking* considerados:

1. **Normal**: neste caso, algoritmo faz uma busca sequencial (da esquerda para a direita) nos elementos da partícula $\mathbf{p}^t_{target}$. Assim, cada troca (c, k) é adicionada ao vetor de velocidade e os elementos das posições k e c são permutados. O processo se repete até que o total de permutações previsto seja executado. Este método apresenta uma polarização, pois tende a organizar sempre a parte esquerda da partícula.
2. **Aleatório**: este caso é similar ao anterior, mas o algoritmo faz uma busca aleatória nos elementos da partícula $\mathbf{p}^t_{target}$, ao invés de sequencial. Este método permite que toda a extensão da partícula seja varrida sem que nenhuma região seja privilegiada.
3. **Encadeado**: neste caso, algoritmo detecta a primeira posição diferente k da partícula $\mathbf{p}^t_{target}$, da esquerda para direita. Após a troca (c, k), o algoritmo considera $k = c$ e repete o procedimento anterior, dando uma sequência de encadeamento. Caso ocorra uma situação em que o novo valor presente em c já esteja na posição correta, o algoritmo reinicia a busca sequencial por uma posição k que seja diferente. O processo ocorre até que o total de permutações previsto seja executado. Este método permite que toda a extensão da partícula seja varrida, mas a sequência de troca é definida pelo próprio arranjo presente na partícula.

4 Resultados e Análise

Duas instâncias do TAP foram consideradas para testes. Cada instância é caracterizada por uma matriz de custos, sendo que cada linha i da matriz representa o custo de alocar o trabalhador i à tarefa da coluna j. Como o número de trabalhadores e tarefas é igual, foram consideradas matrizes quadradas de ordens $N = 10$ e $N = 100$.

A matriz para $N = 10$ é dada por (5). Neste caso a solução ótima, obtida pela enumeração de todas as soluções possíveis, é conhecida e tem custo total igual a 5.

$$\begin{pmatrix} 3 & 1 & 2 & 5 & 0 & 2 & 3 & 1 & 2 & 2 \\ 0 & 2 & 7 & 3 & 2 & 2 & 1 & 8 & 9 & 2 \\ 1 & 1 & 2 & 3 & 4 & 7 & 8 & 3 & 2 & 1 \\ 1 & 1 & 1 & 4 & 5 & 3 & 2 & 2 & 1 & 9 \\ 2 & 3 & 4 & 5 & 2 & 1 & 2 & 4 & 5 & 8 \\ 1 & 2 & 1 & 2 & 3 & 7 & 2 & 1 & 1 & 3 \\ 6 & 5 & 5 & 3 & 2 & 1 & 1 & 1 & 7 & 8 \\ 0 & 4 & 5 & 0 & 2 & 1 & 6 & 8 & 3 & 2 \\ 5 & 6 & 7 & 2 & 1 & 1 & 1 & 0 & 0 & 1 \\ 3 & 2 & 1 & 0 & 0 & 3 & 5 & 1 & 2 & 0 \end{pmatrix} \tag{5}$$

Na segunda instância com $N = 100$, a matriz de custos foi gerada de forma aleatória com valores inteiros $\in \{0, 1, \ldots, 99\}$ (não mostrada por questões de espaço). Para esta instância, o valor ótimo é considerado o melhor resultado obtido pelo PSO dentre todos os testes.

A Tabela 1 apresenta os parâmetros do PSO testado. Vale lembrar que o objetivo do trabalho não é propor uma abordagem para a solução das instâncias do TAP considerado, mas sim investigar a influência de alguns parâmetros específicos da otimização combinatória.

Tabela 1. Parâmetros e características do PSO testado.

Parâmetro	Valor
Número de partículas (S)	{100,200,500,1000}
Número de iterações (T)	100
Número de rodadas (R)	30
Coeficiente inercial (2)	0,0
Coeficiente cognitivo (c_1)	0,7
Coeficiente social (c_2)	0,8
Ordem de movimentação	{C-S,S-C}
Atualiza posição da partícula	{Verdadeiro, Falso}
Tipo de *path-relinking*	{aleatório,encadeado,normal}

De forma a reduzir o número de combinações possíveis, a componente inercial não foi investigada. Desta maneira, o coeficiente associado (w) foi considerado nulo, resultando em apenas duas componentes na equação da velocidade (cognitiva e social). Além disso, tanto c_1 quanto c_2 são definidos no intervalo $[0, 1]$ para garantir que o número de permutações realizadas em cada partícula não exceda o número de elementos na lista de permutações. Durante a movimentação da partícula, a componente cognitiva pode ser aplicada antes da social, representada pela ordem de movimentação C-S (e vice-versa, S-C).

Além disso, investiga-se o momento de atualização da posição da partícula. Finalmente, o *path-relinking* pode ser aplicado de forma aleatória, encadeada ou normal, conforme descrito na Seção 3.3.

A Tabela 2 traz os resultados de quatro diferentes versões do PSO com *path-relinking* aleatório, combinando duas ordens de movimentação (C-S e S-C) com duas formas de atualização da posição da partícula (atualização com *status* Verdadeiro e Falso). Os valores numéricos mostrados na Tabela 2 são valores médios obtidos para $R = 30$ rodadas de cada caso. A métrica $DMOt$ representa a média da distância relativa para o valor ótimo Ot, ou seja, $DMOt = (1/R)\sum_r (Best_r - Ot)/Ot)$, $r = 1, ..., R$. Os valores de C/(C+S) e S/(C+S) representam o percentual de movimentações de cada componente (C ou S) em relação ao total de movimentações (C+S).

Tabela 2. Resultado para o *path-relinking* aleatório ($N = 10$ e $N = 100$).

			atualização = Falso			*atualização* = Verdadeiro		
		Ordem	DMOt	C/(C+S)	S/(C+S)	DMOt	C/(C+S)	S/(C+S)
$N = 10$	$S = 100$	C-S	0,2333	11,01%	88,99%	0,2933	12,1%	87,9%
		S-C	0,1867	10,40%	89,60%	**0,1667**	32,8%	67,2%
	$S = 200$	C-S	0,1067	12,2%	87,8%	0,2267	9,65%	90,3%
		S-C	0,1667	15,4%	84,6%	**0,0933**	32,2%	67,8%
$N = 100$	$S = 500$	C-S	0,7948	10,2%	89,8%	0,8174	7,3%	92,7%
		S-C	0,7568	11,2%	88,8%	**0,5371**	36,0%	64,0%
	$S = 1000$	C-S	0,4052	10,4%	89,6%	0,4972	7,5%	92,5%
		S-C	0,4948	10,4%	88,6%	**0,2742**	35,7%	64,3%

Como pode ser observado na Tabela 2, em todas as abordagens testadas, o percentual de movimentos efetivamente realizados pela componente social é bem superior ao da componente cognitiva (em especial para as abordagens cuja ordem é Cognitivo-Social). Isto pode ser explicado pelo fato de que a componente social da velocidade tem maior probabilidade de melhorar o *fitness* da partícula $\mathbf{p}^t$ e com isso melhorar sua memória local ($\mathbf{best}^t$). Assim, na próxima iteração, quando a componente cognitiva é avaliada no início da movimentação, há maior probabilidade de ser nula (visto que $\mathbf{p}^{t+1}$=$\mathbf{best}^t$) e por isso o movimento cognitivo estimado no cálculo inicial não se efetiva.

A Tabela 2 também permite concluir que a abordagem cuja a ordem de movimentos é S-C e com atualização da posição da partícula resulta em desempenho ligeiramente superior, para todos os casos testados. Entrentanto, após a aplicação do teste estatístico Kruskal-Wallis, a diferença estatística só foi comprovada para o caso N=100. Portanto, os desempenhos das abordagens podem ser considerados semelhantes para o caso N=10 (exceto as abordagens com atualização (C-S x S-C) que apresentam diferença estatística). Para a abordagem S-C com atualização, verifica-se que o menor valor na média vem acompanhado de um ligeiro aumento na proporção dos movimentos efetivados para a componente cognitiva em relação às outras abordagens. Com isso, o enxame tende a manter maior a diversidade podendo alcançar assim resultados melhores. A questão então é avaliar por que há este ligeiro aumento. No caso da ordem S-C, com atualização, como a partícula efetivamente se movimenta depois da ação da componente social (dada a atualização da sua posição), há uma menor probabilidade da componente cognitiva ser nula, pois a partícula é diferente de sua memória local. Os experimentos foram repetidos para os outros tipos

de *path-relinking* (normal e encadeado) e os resultados são similares aos apresentados na Tabela 2.

A Tabelas 3 traz os resultados de DMOt para as melhores abordagens de cada caso testado, considerando os três tipos de *path-relinking*.

Tabela 3. Melhores resultados para cada tipo de *path-relinking* ($N = 10$, $N = 100$).

		Aleatório			Encadeado			Normal		
		DMOt	Ordem	Atz	DMOt	Ordem	Atz	DMOt	Ordem	Atz
$N = 10$	$S = 100$	**0,17**	S-C	V	0,35	S-C	V	0,54	S-C	V
	$S = 200$	**0,09**	S-C	V	0,21	S-C	V	0,41	S-C	V
$N = 100$	$S = 500$	**0,54**	S-C	V	1,04	S-C	V	2,12	S-C	V
	$S = 1000$	**0,27**	S-C	V	0,78	S-C	V	1,93	S-C	V

Conforme pode ser observado na Tabelas 3, a abordagem S-C com atualização se destaca novamente e é possível concluir que o *path-relinking* aleatório fornece os menores valores de DMOt para todos os casos analisados.

5 Conclusões

Este trabalho apresentou uma análise experimental do desempenho de uma abordagem baseada na otimização por enxame de partículas (PSO) aplicada a um problema combinatório. Embora pouco explorada na literatura, esta análise mostrou a influência que alguns parâmetros e sequências de atualizações das partículas têm no desempenho geral deste algoritmo adaptado para otimização combinatória. Os resultados mostraram que tanto a sequência quanto a forma com que as componentes da velocidade são aplicadas na atualização da posição da partícula precisam ser adequadamente caracterizadas. Algumas delas com desempenho melhor do que outras, como a sequência aleatória na aplicação de cada *path-relinking*. Isso deve ainda ser devidamente formalizado na equação de atualização da velocidade.

Referências

Beni, G. (1988). The concept of cellular robotic systems. In *3rd IEEE International Symposium on Intelligent Control*, pages 57–62.

Bonabeau, E., Theraulaz, G., and Dorigo, M. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press.

Clerc, M. (2004). Discrete particle swarm optimization: illustrated by the traveling salesman problem. In Onwubolu, G. C. and Babu, B. V., editors, *New optimization techniques in engineering*. Springer-Verlag.

DeCastro, L. N. (2006). *Fundamentals of Natural Computing: Basic Concepts, Algorithms and Applications*. Chapman & Hall.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systemas, Man, and Cybernetics - part B*, 26(1):29–41.

Glover, F., Kelly, J., and Laguna, M. (1999). New advances for wedding optimization and simulation. In *Winter Simulation Conference Proceedings*, volume 1, pages 255 –260 vol.1.

Goldbarg, E., de Souza, G., and Goldbarg, M. (2006). Particle swarm for the traveling salesman problem. In Gottlieb, J. and Raidl, G., editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3906 of *Lecture Notes in Computer Science*, pages 99–110. Springer Berlin / Heidelberg.

Hu, X. (2003). PSO tutorial. http://www.swarmintelligence.org/tutorials.php (Last access: 20th june 2011).

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948.

Kennedy, J. and Eberhart, R. (1997). A discrete binary version of the particle swarm algorithm. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 5, pages 4104–4108.

Machado, T. and Lopes, H. (2005). A hybrid particle swarm optimization model for the traveling salesman problem. In Ribeiro, B., Albrecht, R. F., Dobnikar, A., Pearson, D. W., and Steele, N. C., editors, *Adaptive and Natural Computing Algorithms*, pages 255–258. Springer Vienna.

Pang, W. (2004). Modified particle swarm optimization based on space transformation for solving traveling salesman problem. In *International Conference on Machine Learning and Cybernetics, 2004*, volume 4, pages 2342–2346, Shanghai, China.

Poli, R. (2008). Analysis of the publications on the applications of particle swarm optimisation. *J. Artif. Evol. App.*, 2008:1–4.

Rosendo, M. (2010). Um algoritmo de otimização por nuvem de partículas para resolução de problemas combinatórios. Master's thesis, Universidade Federal do Paraná.

Talbi, E.-G. (2009). *Metaheuristics: from design to implementation*. John Wiley and Sons.

Wang, K.-P. W., Huang, L., Zhou, C.-G., and Pang, W. (2003). Particle swarm optimization for traveling salesman problem. In *International Conference on Machine Learning and Cybernetics*, pages 1583 – 1585.