Recebido: 02/09/09
Aprovado: 15/03/12

Um algoritmo enxame de partículas para uma variante do problema de máxima cobertura

Prof. Bruno de Athayde Prata (UFC – CE/Brasil) - baprata@ufc.br
• Campus do Picí, Bloco 710, 60455-760, Fortaleza-CE, fone: (55) 85-3366-9600

RESUMO O presente trabalho tem por objetivo, reportar o desenvolvimento de uma heurística enxame de partículas binária, para uma variante do problema de máxima cobertura. Tal variante consiste em um problema de minimização, o qual se aproxima bastante do problema *p*-medianas. Foram geradas aleatoriamente 10 instâncias de médio porte, as quais foram solucionadas de forma exata. Um algoritmo PSO binário foi desenvolvido para a variante supracitada, o qual teve sua performance comparada com o método exato e com uma adaptação da heurística de Chvátal. Os resultados obtidos apontam para a efetividade da abordagem proposta, para os conjuntos de problemas analisados.

Palavras-chave Meta-heurísticas; Otimização Combinatória; Enxame de Partículas Binário.

ABSTRACT *This paper aims at reporting on the development of a Binary Particle Swarm Optimization Algorithm (BPSO) for a variant of the Maximum Covering Problem. This variant consists of a minimization problem, which is similar to the p-medians problem. A set of 10 medium size instances that were solved for optimality were randomly generated. A BPSO heuristic for the abovementioned variant was developed, which was compared with the exact method and with an adaptation of the classical Chvátal heuristic. The obtained results indicate that the proposed approach has considerable potential for achieving good solutions.*

Keywords *Metaheuristics; Combinatorial Optimization; Binary Particle Swarm Optimization.*

PRATA, B. A. Um algoritmo enxame de partículas para uma variante do problema de máxima cobertura. **GEPROS. Gestão da Produção, Operações e Sistemas**, Ano 7, nº 2, abr-jun/2012, p. 139-148.

1. INTRODUÇÃO

O Problema de Máxima Cobertura (*Maximum Covering Problem* – MCP) é um problema de otimização combinatória amplamente estudado, com diversas aplicações, tais como: localização de facilidades, telecomunicações e *scheduling* (RESENDE, 1998; ARAKAKI e LORENA, 2006; PARK e RYU, 2006). Tais autores salientam que o problema é de difícil resolução, sendo classificado como pertencente à classe NP-Hard. Deste modo, métodos aproximativos, tais como heurísticas, são desejáveis para a obtenção de boas soluções em tempo computacional admissível.

O MCP clássico é definido como um problema de maximização, no qual, às variáveis de decisão, são associados pesos. Contudo, existe um caso particular do MCP, em que os pesos são unitários e o problema pode ser descrito como um problema de minimização. Tal formulação será apresentada a seguir.

Dada uma matriz A, tal que $a_{ij} \in \{0,1\}$, o MCP consiste em cobrir a maior quantidade de linhas da matriz A, com uma quantidade de colunas da matriz A menor ou igual à d. As variáveis y_i representam as linhas da matriz A, de modo que $y_i = 1$ se a i-ésima linha não é coberta em uma solução, sendo $y_i = 0$ caso contrário. As variáveis x_j representam as colunas da matriz A, de modo que $x_j = 1$ se a j-ésima coluna faz parte da solução, sendo $x_j = 0$, caso contrário.

É pertinente salientar que o MCP é um problema de otimização combinatória cujas variáveis de decisão são de caráter binário.

A variante descrita do MCP pode ser formulada matematicamente como segue:

Minimizar $$z = \sum_{i=1}^{m} y_i \qquad (1)$$

Sujeito a:

$$\sum_{j=1}^{n} x_j \leq d \qquad (2)$$

$$\sum_{j-1}^{n} a_{ij} x_j + y_i \geq 1 \qquad \forall\ i = 1, \ldots, m. \qquad (3)$$

$$x_j \in \{0,1\} \qquad \forall\ j = 1, \ldots, n. \qquad (4)$$

$$y_i \in \{0,1\} \qquad \forall\ i = 1, \ldots, m. \qquad (5)$$

A função objetivo representada pela equação (1) consiste na minimização das linhas da matriz A descobertas. A restrição (2) impõe que um número máximo de d de colunas da matriz A seja selecionado na solução. Caso a i-ésima linha da matriz não possa ser coberta, o conjunto de restrições do tipo (3) implica que $y_i = 1$. Os conjuntos de restrições do tipo (4) e (5) ilustram o caráter binário das variáveis de decisão. Deve-se observar que as variáveis do tipo y podem ser consideradas como variáveis auxiliares, pois, na verdade, não se caracterizam efetivamente como decisões.

O MCP assemelha-se ao Problema de Cobertura de Conjuntos (*Set Covering Problem* – SCP). Enquanto no SCP busca-se cobrir as linhas da matriz A com custo mínimo, na variante do MCP busca-se minimizar o número de linhas de A descobertas, não sendo mensurado nenhum tipo de custo. A variante observada do MCP poderia ser vista como um problema de mínima não-cobertura.

Deve-se observar também, que a restrição do tipo (2) torna o MCP bastante difícil de ser solucionado. Sejam m linhas a serem cobertas por um conjunto de até d colunas, o número de soluções possíveis para o problema é dado por (CURTIN *et al.*, 2005):

$$\binom{m}{d} = \frac{m!}{d!\,(m-d)} \qquad (6)$$

Conforme Goldbarg e Luna (2005), a formulação do MCP, enquanto um problema de minimização, tem forte relação com o problema de *p*-medianas, o qual também, é bastante estudado na área de Pesquisa Operacional.

O artigo é estruturado em outras quatro seções, descritas a seguir. Na segunda seção, apresenta-se a meta-heurístca Enxame de Partículas. Na terceira seção, apresenta-se o algoritmo proposto para a resolução do MCP. Na quarta seção, são expostas a experimentação computacional e os resultados obtidos pela heurística. Por fim, na quinta seção, são apresentadas as considerações finais do estudo.

2. ENXAME DE PARTÍCULAS

Os algoritmos evolucionários preconizam a simulação de princípios relacionados à teoria de evolução das espécies e, para tanto, utilizam processos de recombinação de soluções, com vistas a efetuar uma busca inteligente no espaço de soluções viáveis. Como exemplos de algoritmos evolucionários, podem ser citados os Algoritmos Genéticos (AG) e colônias de insetos sociais, tais como formigas, abelhas e cupins.

Na década de 90, cientistas das áreas de computação e zoologia debruçavam-se sob o estudo do comportamento psicológico e social de determinadas espécies. Tais pesquisadores estavam fascinados com as interações que bandos de aves e cardumes de peixes realizavam em suas atividades cotidianas.

Tais grupos de animais, ao procurarem por comida ou ao fugirem de predadores, possuem um tipo de consciência coletiva. Cada elemento do grupo aprende com o comportamento dos seus demais componentes. Tendo em vista esse comportamento, diversos pesquisadores buscaram simular esses tipos de interação por intermédio de computadores.

Kennedy e Ebehart (1995a) propuseram uma nova meta-heurística, destinada à resolução de problemas irrestritos de otimização não-linear, denominada *Particle Swarm Optimization* (PSO).

Cada partícula é caracterizada por sua aptidão (valor da função objetivo), sua posição (conjunto de valores das variáveis independentes do problema) e por um vetor que regula a mudança de posições. Tal vetor é chamado 'velocidade' (DRÉO *et al*., 2006).

A filosofia de PSO é a seguinte: gera-se uma população inicial de partículas (indivíduos), cada qual numa posição inicial do espaço de soluções e dotada de uma velocidade inicial. Cada partícula percorrerá o espaço de busca *n*-dimensional e terá sua velocidade atualizada de acordo com a velocidade das demais partículas. Partículas que estiverem distantes das regiões promissoras do espaço de busca, terão sua velocidade incrementada, enquanto partículas que estiverem próximas das regiões promissoras do espaço de busca, terão sua velocidade moderada.

PSO possui uma estratégia de busca extremamente eficiente. Como vantagens do algoritmo supracitado, podem ser ressaltadas (KENNEDY e EBEHART 1995a, 1995b): (i) possui fácil implementação computacional, requerendo poucas linhas de código; (ii) utiliza pouca memória e requer pouca velocidade de processamento; e (iii) o processo de busca é racionalizado pelo contínuo aprendizado das partículas.

Após os promissores resultados advindos da aplicação de PSO em problemas de programação não-linear, foram iniciados diversos estudos que objetivavam avaliar o desempenho da meta-heurística supramencionada, em problemas de otimização discreta e combinatória.

Um levantamento de trabalhos com aplicações do PSO em otimização binária, é apresentado em Banks *et al*. (2007).

Kennedy e Ebehart (1997) desenvolveram uma versão binária para o algoritmo PSO e concluíram que a meta-heurística também é flexível e robusta para essa classe de problemas. Tasgetiren e Liang (2003) aplicaram um PSO binário para o *Lot Sizing Problem*, obtendo resultados de melhor qualidade do que um Algoritmo Genético. Yang *et al*. (2004) apresentam um PSO binário para o *Capacited Vehicle Routing Problem* (CVRP).

3. ALGORITMO PROPOSTO

Para resolução do MCP, por meio do PSO, foi utilizada a seguinte codificação: cada partícula consiste em um vetor binário de ordem n, em que cada componente representa uma coluna da matriz A. Em outras palavras, tal codificação representa o vetor de variáveis de decisão x.

Por se tratar de um algoritmo evolucionário, o PSO é composto por uma população de soluções (partículas), incorrendo em um parâmetro *maxpop*. A busca é efetuada em uma quantidade de iterações (gerações), controlada pelos parâmetros *gen* (iteração corrente) e *maxgen* (número máximo de gerações). A população inicial é gerada de forma aleatória, sem levar em consideração as características do MCP. Adotou-se $P[x_j=1] = 0{,}05$ e $P[x_j = 0] = 0{,}95$, para denotar as probabilidades de um coluna fazer parte (ou não) da solução.

A função de avaliação é a quantidade de linhas da matriz A descobertas em uma dada solução, a qual deve ser minimizada. Para que a restrição (2) não seja infringida, adotou-se um fator de penalidade igual a 20, por coluna excedente, para penalizar partículas que utilizassem mais de d colunas em uma solução.

Conforme Levine (1996), não existem métodos exatos para a determinação dos valores de uma penalização e a escolha do valor ideal é uma tarefa difícil. Optou-se por ajustar o valor empiricamente e, após experimentos computacionais, chegou-se a um valor de penalidade igual a $m/10$.

As velocidades são armazenadas em uma matriz denominada *velocity*, de ordem *maxpop* × n. Os valores são gerados por meio da seguinte expressão: $velocity[i,j] = v_{min}+(v_{max} - v_{min}) \times random[0,1]$. Os parâmetros v_{max} e v_{min} são os valores máximos e mínimos permitidos para a velocidade de cada partícula, sendo adotados $v_{max} = \ln(n)$ e $v_{min} = -\ln(n)$.

Sejam *pbest* e *gbest* as melhores soluções obtidas, respectivamente, pela i-ésima partícula e pelo enxame. Sejam x_{pbest} e x_{gbest} as posições de tais soluções, tem-se que a variação de velocidade é dada pela equação:

$$\Delta v[i,j]: = c_1 \times \text{random}[0,1] \times (x_{pbest} - x) + c_2 \times \text{random}[0,1] \times (x_{gbest} - x) \quad (7)$$

Em que c_1 e c_2 são duas constantes, denominadas, respectivamente, de constante cognitiva e constante social. A constante cognitiva está relacionada ao aprendizado da partícula, em relação ao seu próprio movimento no espaço de busca, enquanto a constante social está relacionada ao aprendizado da partícula, em relação ao comportamento do enxame como um todo.

A função sigmóide, apresentada na equação (8), é a mais empregada para este fim (KENNEDY e EBEHART, 1997; TASGETIREN e LIANG, 2003).

$$sigmoid\,(v[i, j]) = \frac{1}{1 + e^{-v[i, j]}} \quad (8)$$

Como em um problema de otimização binária as variáveis de decisão só assumem os valores 0 e 1, é necessária uma função que transforme uma velocidade real, definida no intervalo $[v_{min},v_{max}]$, para o intervalo [0,1].

Assim, calcula-se o valor da sigmóide para a velocidade da i-ésima partícula na j-ésima dimensão (*bit*). Se este valor for menor que um número aleatório uniformemente distribuído, o j-ésimo *bit* assume valor 1; caso contrário, este assume valor 0. Na Figura 1, é apresentado o pseudocódigo para atualização das posições das partículas.

Figura 1 – Pseudocódigo para atualização do movimento das partículas.

```
para i ← até maxpop faça
    para j ← 0 até n faça
        r ← [0,1]
        se r < sigmoid(v[i,j]) então x[i,j] ← 1 senão x[i,j] ← 0
    fim-do-para
fim-do-para
```

Fonte: Elaboração do autor.

Após os movimentos realizados, calcula-se a aptidão de cada partícula e penalizam-se as soluções inviáveis. Se o valor de *gbest* não sofrer melhorias durante *k* iterações do algoritmo, ocorrerá uma reinicialização das velocidades das partículas, exceto da posição da partícula de melhor aptidão (x_{gbest}). Na Figura 2, é ilustrado o pseudocódigo da heurística PSO proposta.

Figura 2 – Pseudocódigo da heurística PSO proposta.

```
Gerar população inicial
Gerar velocidades iniciais
Obter valores iniciais para pbest e gbest
Enquanto gen ≤ maxgen faça
    Obter pbest e gbest
    Calcular a velocidade da i-ésima partícula na j-ésima dimensão
    Atualizar a velocidade da i-ésima partícula na j-ésima dimensão
    Atualizar cada bit na string usando a função sigmóide (atualizar posição)
    Avaliar a aptidão de cada partícula
    Penalizar as soluções inviáveis
    Reiniciar as velocidades para cada k iterações em melhoria de gbest
   gen ← gen+1
fim-do-enquanto
```

Fonte: Elaboração do autor.

É pertinente salientar que algoritmos construtivos e buscas locais, conforme Resende (1998), são de vital importância para resolução do MCP com eficácia e eficiência. Todavia, o autor empregou um PSO sem a hibridização de tais técnicas, com vista a demonstrar que os mecanismos de aprendizagem social do PSO, por si só, podem acarretar na obtenção de boas soluções em problemas de otimização combinatória binária, como é o caso do MCP.

4. EXPERIMENTOS COMPUTACIONAIS E ANÁLISE DOS RESULTADOS OBTIDOS

Tendo em vista que, para a variante do MCP em foco, não existem instâncias disponíveis nas bibliotecas da área de Pesquisa Operacional, foi necessário criar conjuntos de dados para a etapa de experimentação computacional.

Foram geradas aleatoriamente 10 instâncias[1] de médio porte, as quais foram solucionadas de forma exata, por meio do *software* LINGO 8.0. A descrição das instâncias é reportada na Tabela 1. As características das instâncias são as seguintes: número de linhas da matriz A (m), número de colunas da matriz A (n), número máximo de colunas a serem utilizadas na solução (d) e densidade da matriz A (ρ). O valor de ρ é obtido pela divisão do número de elementos da matriz A iguais a 1 pelo número total de elementos de A.

Tabela 1 – Descrição das instâncias geradas.

Instância	*m*	*n*	*d*	ρ (%)	Ótimo global	*t*(s)
MCP01	200	200	16	4,02	48	5434
MCP02	200	200	10	3,82	91	48
MCP03	200	200	18	3,94	37	5906
MCP04	200	200	18	3,83	40	2896
MCP05	200	200	16	3,93	44	232
MCP06	200	200	6	9,22	65	396
MCP07	200	200	17	3,96	41	443
MCP08	200	200	32	3,83	0	1621
MCP09	200	200	16	3,95	52	2148
MCP10	200	200	23	4,12	15	19193

Fonte: Elaboração do autor.

O algoritmo PSO foi implementado em linguagem Pascal. Os experimentos foram realizados em um AMD Semprom 2400 + 1.67 GHz, 504MB RAM. Foram utilizados os seguintes parâmetros, os quais foram ajustados, após experimentos computacionais: *maxpop* = 15, *maxgen* = 2500 e c1 = c2 = 1. Para a rotina de reinicialização das velocidades, adotou-se k = 500 gerações.

Para avaliação da heurística, além de sua comparação com os valores ótimos obtidos pelo *solver*, foi implementada uma heurística gulosa para o MCP, baseada na clássica heurística de Chvátal para o SCP, cujo pseudocódigo consta na Figura 3.

Figura 3 – Adaptação da heurística de Chvátal para a variante do MCP.

```
disponível ← x_j, ∀ j = 1, ..., n
col ← 0
enquanto disponível < > vazio ou col < d faça
    obter a coluna x_j que cobre a maior quantidade de linhas da matriz A.
    solução ← x_j
    disponível ← disponível – {x_j,}
    col ← col +1
fim-do-enquanto
```

Fonte: Elaboração do autor.

Para o caso da variante do MCP em análise, em que todas as colunas têm o mesmo custo, o procedimento míope consiste em designar para a solução aquelas colunas que cobrirem a maior quantidade de linhas da matriz A, respeitando a restrição de alocar no máximo d colunas.

1 As instâncias geradas estão disponíveis para *download* no sítio http://baprata.blogspot.com/2011/05/mcp.html.

Para cada instância, foram feitas vinte execuções do algoritmo PSO, sendo que tais resultados são ilustrados na Tabela 2. A primeira coluna da tabela apresenta a identificação da instância. A segunda e a terceira colunas apresentam, respectivamente, o melhor e o pior resultados obtidos pela heurística. A quarta e a quinta coluna apresentam, respectivamente, a média e o desvio padrão dos resultados obtidos pelo algoritmo, para as vinte replicações efetuadas. Na sexta coluna, apresenta-se o número de vezes que o algoritmo encontrou a solução ótima da instância. Na sétima coluna, são ilustrados os tempos computacionais médios, para as vinte replicações do algoritmo. Por fim, na oitava coluna, estão os resultados da heurística gulosa.

Como o algoritmo guloso apresentado é determinístico, só foi efetuada uma rodada do mesmo, para cada instância. Diante do fato da precisão adotada no presente artigo (trabalha-se com tempos inteiros, ou seja, com precisão de segundos), os tempos computacionais da heurística gulosa não foram reportados, pois foram todos iguais a 0s.

Com base nos resultados apresentados, pode-se ressaltar que, apesar das soluções iniciais geradas serem de baixa qualidade e da ausência de heurísticas para o reparo de soluções inviáveis, o PSO conseguiu obter bons resultados em tempo computacional admissível. Para cinco das dez instâncias consideradas, as soluções ótimas puderam ser obtidas. Tal fato corrobora a hipótese de que os princípios da meta-heurística PSO são adequados para a resolução de problemas de otimização binária, como é o caso do MCP.

Tabela 2 – Resultados dos experimentos computacionais.

Instância	Ótimo global	Melhor solução	Pior solução	Média	Desvio padrão	# soluções ótimas em 20 rodadas	$t_{médio}$(s)	Solução *greedy*
MCP01	48	48	55	50,10	1,76	4	23,71	70
MCP02	91	92	96	93,90	1,22	0	23,78	97
MCP03	37	37	44	40,35	1,77	2	23,74	57
MCP04	40	42	46	44,10	1,70	0	23,77	65
MCP05	44	44	50	46,45	1,60	4	23,94	65
MCP06	65	65	74	70,15	3,35	5	23,85	82
MCP07	41	41	48	43,70	2,85	9	23,86	57
MCP08	0	4	8	5,85	1,39	0	23,79	26
MCP09	52	54	58	55,65	1,62	0	23,60	73
MCP10	15	19	24	21,20	1,03	0	23,82	36

Fonte: Elaboração do autor.

A estratégia de gerar soluções de forma aleatória, sem levar em consideração as características do problema, não se mostrou a mais eficiente para o MCP. As maiores discrepâncias entre os resultados da heurística e as soluções ótimas das instâncias foram encontradas nos casos em que as soluções iniciais eram de qualidade muito baixa, acarretando que a heurística ficasse confinada em ótimos locais.

Devido à restrição (2) do MCP, o vetor x, que representa as colunas que fazem parte da solução do problema, é bastante esparso. Os diversos ótimos locais são separados por uma grande distância de Hamming, sendo rodeados por diversas regiões de inviabilidade. Deste modo, o PSO concentra sua busca em regiões de atração de ótimos locais, tendo dificuldade de migrar para o ótimo global. Esta dificuldade seria bastante reduzida com a inclusão de uma heurística de reparos.

A qualidade de uma solução para o MCP é obtida através do cálculo do percentual de cobertura *p*, determinado pela seguinte expressão, na qual *n* é o número de linhas da matriz *A* e *z* é o valor da função objetivo obtida, a qual representa o número de linhas descobertas:

$$p = \frac{(m - z)}{n} \tag{9}$$

Assim, calculou-se o percentual de cobertura *p*, obtido pelo método exato, pelo PSO (considerado em relação ao melhor resultado, ao pior resultado e à média dos resultados) e pela heurística *greedy*, para as instâncias analisadas.

Para avaliação da eficácia do PSO e da heurística gulosa, calculou-se o Desvio Percentual Relativo (DPR), o qual consiste na diferença entre a solução obtida e a solução ótima.

Tabela 3 – Comparação entre os resultados das heurísticas e as soluções ótimas.

Instância	Ótimo global	$P_{ótimo}$ (%)	P_{melhor} (%)	$P_{média}$ (%)	P_{pior} (%)	P_{greedy} (%)	DPR_{melhor} (%)	$DPR_{média}$ (%)	DPR_{pior} (%)	DPR_{greedy} (%)
MCP01	48	76,00	76,00	74,95	72,50	65,00	0,00	1,05	3,50	11,00
MCP02	91	54,50	54,00	53,05	52,00	51,50	0,50	1,45	2,50	3,00
MCP03	37	81,50	81,50	79,83	78,00	71,50	0,00	1,67	3,50	10,00
MCP04	40	80,00	79,00	77,95	77,00	67,50	1,00	2,05	3,00	12,50
MCP05	44	78,00	78,00	76,78	75,00	67,50	0,00	1,23	3,00	10,50
MCP06	65	67,50	67,50	64,93	75,00	59,00	0,00	2,58	4,50	8,50
MCP07	41	79,50	79,50	78,15	63,00	71,50	0,00	1,35	3,50	8,00
MCP08	0	100,00	98,00	97,08	76,00	87,00	2,00	2,93	4,00	13,00
MCP09	52	74,00	73,00	72,18	71,00	63,50	1,00	1,83	3,00	10,50
MCP10	15	92,50	90,50	89,40	88,00	82,00	2,00	3,10	4,50	10,50

Fonte: Elaboração do autor.

É pertinente destacar que, para o caso do PSO, os desvios obtidos foram pequenos, ainda que a heurística usasse tempo de processamento bastante inferior ao requerido para a obtenção das soluções ótimas. A heurística gulosa ofertou alguns resultados com desvios superiores a 10%, o que mostra que não provê resultados de alta qualidade.

Para uma análise global dos algoritmos, calculou-se o Desvio Percentual Relativo Médio (DPRM), o qual é obtido pela média aritmética dos DPR's obtidos para cada instância. Na tabela 4 são apresentados os valores de DPRM obtidos para o PSO (considerando o melhor resultado, o pior resultado e a média dos resultados) e para a heurística *greedy*. Os resultados da Tabela 4 corroboram a hipótese de que o PSO se mostrou eficaz para a resolução do MCP.

Tabela 4 – Análise dos DPRM's dos resultados das heurísticas.

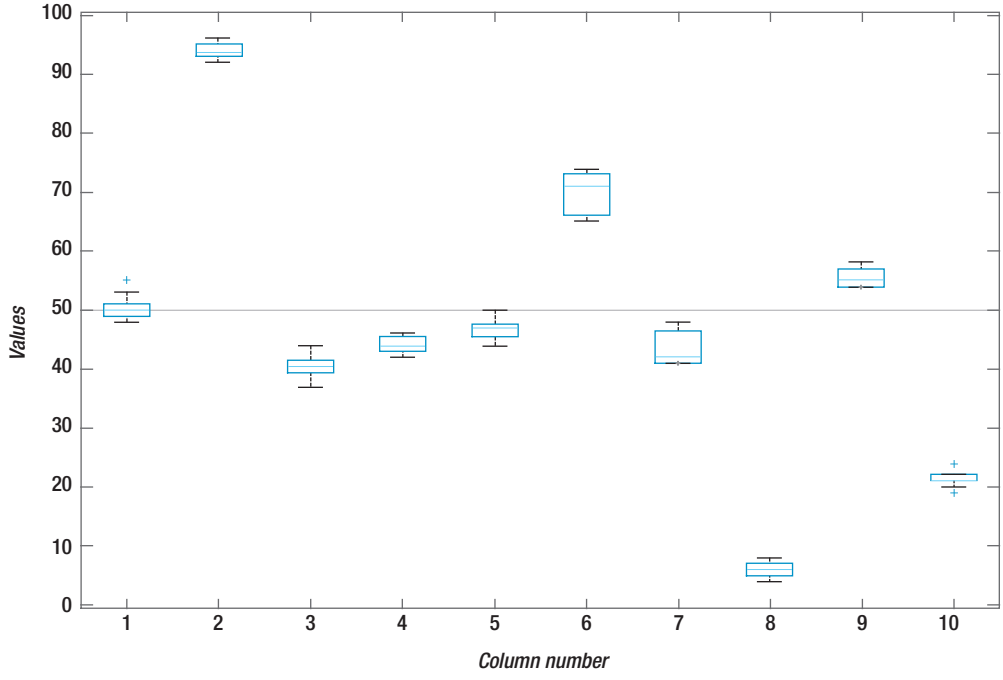
$DPRM_{melhor}$ (%)	$DPRM_{média}$ (%)	$DPRM_{pior}$ (%)	$DPRM_{greedy}$ (%)
0,65	1,92	3,50	9,75

Fonte: Elaboração do autor.

Para uma melhor compreensão do comportamento do algoritmo PSO em cada instância, foram concebidos os diagramas de caixas (*boxplots*), cuja variável de interesse era o valor encontrado pela heurística em cada rodada do algoritmo.

Para as dez instâncias analisadas, percebe-se que a dispersão entre os resultados obtidos e o resultado médio da heurística, foi mais acentuada para as instâncias 6 e 7. Nas oito demais instâncias, pode-se observar que o algoritmo se portou com estabilidade. Com relação à presença de observações aberrantes (*outliers*), pode-se destacar que em apenas três das duzentas replicações realizadas, ocorreram valores aberrantes (uma vez para a instância 1 e duas vezes para a instância 10). Tal resultado também aponta para a estabilidade do algoritmo proposto.

Figura 4 – Diagramas de caixas com os resultados do PSO para as instâncias testadas.



5. CONSIDERAÇÕES FINAIS

Este artigo teve como objetivo reportar a elaboração de um algoritmo PSO binário, com função de penalidade, para o *Maximum Covering Problem*. Este é um dos primeiros trabalhos a apresentar uma aplicação de PSO em problemas de otimização combinatória binária e, no melhor conhecimento do autor, este é o primeiro artigo a reportar uma aplicação da meta-heurística supracitada na resolução do *Maximum Covering Problem*.

A heurística proposta foi testada em um conjunto de dez instâncias de médio porte, as quais foram geradas aleatoriamente e solucionadas de forma exata, por meio do *software* LINGO. O autor se dispõe a fornecer os conjuntos de dados gerados para outros pesquisadores.

A experiência computacional realizada mostrou que o algoritmo desenvolvido pôde obter soluções de boa qualidade, com baixo custo computacional. O processo de aprendizagem coletiva das partículas permitiu que a população de soluções geradas percorresse, com eficácia e eficiência, o espaço de soluções do problema.

Como sugestões para futuros estudos, para melhoria da eficácia e da eficiência da abordagem em foco, podem ser citadas: incorporação de uma heurística para o reparo de soluções inviáveis, geração de população inicial eficiente e hibridização da heurística PSO, com um procedimento de busca local.

Também é pertinente propor testes com instâncias de maior porte, sejam elas teóricas ou mesmo de caráter prático, oriundas, por exemplo, de problemas de transporte público (*crew scheduling*). O autor está desenvolvendo uma aplicação do PSO para o *Vehicle and Crew Scheduling* Problem (VCSP).

6. REFERÊNCIAS BIBLIOGRÁFICAS

ARAKAKI, R. G. I.; LORENA, L. A. N. A constructive Genetic Algorithm for the Maximal Covering Location Problem. *In*: **4th Metaheuristics International Conference**, Porto, 2006.

BANKS, A.; VINCENT, J.; ANYAKOHA, C. A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. **Natural Computing**; 7: 109–124, 2007.

CURTIN, K. M.; QIU, F.; HAYSLETT-MCCALL, K.; BRAY, T. M. **Integrating GIS and Maximal Covering Models to Determine Optimal Police Patrol Areas**. USA: University of Dallas, 2005.

DRÉO, J.; PÉTROWKSI, A.; SIARRY, P.; TAILLARD, E. **Metaheuristics for hard optimization**. Springler: Heidelberg, 2006.

GOLDBARG, M. C.; LUNA, H. P. L. **Otimização combinatória e programação linear – modelos e algoritmos**. Rio de Janeiro: Elsevier, 2005.

KENNEDY, J.; EBERHART, R. C. Particle swarm optimization. *In*: **Proceedings of the IEEE International Conference on Neural Networks**, Piscataway, 1995a.

KENNEDY, J.; EBERHART, R. C. A new optimizer using particle swarm theory. *In*: **Proceedings of the 6th International Symposium on Micromachine and Human Science**, Nagoya, 1995b.

KENNEDY, J.; EBERHART, R. C. **A discrete binary version of the particle swarm algorithm**. *In*: Conference on Systems, Man and Cybernetics, Hyatt, 1997.

LEVINE, D. Application of a hybrid genetic algorithm to airline crew scheduling. **Computers and Operations Research**, 23, p. 547–558, 1996.

PARK, T.; RYU, K. R. Crew pairing optimization by a genetic algorithm with unexpressed genes. **Journal of Intelligent Manufacturing**, 17: 375–383, 2006.

RESENDE, M. G. C. Computing approximate solutions of the maximum covering problem with GRASP. **Journal of Heuristics**, 4: 161–177, 1998.

TASGETIREN, M. F.; LIANG, Y. C. A binary Particle Swarm Optimization Algorithm for the lot sizing problem. **Journal of Economic and Social Research**; 5: 1–20, 2003.

YANG, S. Y.; WANG, M.; JIAO, L. C. A quantum particle swarm optimization. *In*: **Proceedings of the 2004 IEEE congress on evolutionary computation**, Portland, 2004.