SISTEMA IMUNOLÓGICO ARTIFICIAL APLICADO AO PROBLEMA GENERALIZADO DE ATRIBUIÇÃO

Tiago A. Almeida[1], Carlos H. Dias[2], Jurandy Almeida[3], Ricardo C. Silva[1], Akebo Yamakami[1]

[1] Faculdade de Engenharia Elétrica e de Computação - FEEC
Universidade Estadual de Campinas - UNICAMP
13083-970, Campinas, SP, Brasil
{tiago,rcoelhos,akebo}@dt.fee.unicamp.br

[2] Instituto de Matemática, Estatística e Computação Científica - IMECC
Universidade Estadual de Campinas - UNICAMP
13083-970, Campinas, SP, Brasil
diashenrique@ime.unicamp.br

[3] Instituto de Computação - IC
Universidade Estadual de Campinas - UNICAMP
13083-970, Campinas, SP, Brasil
jurandy.almeida@ic.unicamp.br

RESUMO

O Problema Generalizado de Atribuição é um problema clássico de otimização combinatória comumente utilizado para modelar uma grande variedade de aplicações reais. Este trabalho apresenta um novo paradigma de resolução baseado no estudo do sistema imunológico biológico. O método proposto foi avaliado usando as principais instâncias presentes na literatura. A análise dos resultados indica que a heurística proposta é eficiente na exploração do espaço de busca, uma vez que, foi encontrado um conjunto de soluções de alta qualidade e ao mesmo tempo o método conseguiu contornar a complexidade do problema.

PALAVRAS CHAVE. Problema Generalizado de Atribuição. Sistema Imunológico Artificial. Computação Evolutiva. Metaheurísticas.

ABSTRACT

The Generalized Assignment Problem is a classical combinatorial optimization problem generally used for modeling a variety of real world applications. This work presents a new paradigm for solving the problem based on studies of the biological immune system. The proposed method was evaluated using standard instances available in the literature. The results indicate that the proposed heuristic is very efficient on exploration the search space, yielding an effective solution and avoiding the complexity issue.

KEYWORDS. Generalized Assignment Problem. Artificial Immune System. Evolutionary Computation. Metaheuristics.

1. Introdução

O Problema Generalizado de Atribuição (PGA) é um dos mais representativos problemas de otimização combinatória. Sua resolução consiste em atribuir um conjunto de agentes para atender um conjunto de tarefas a um custo mínimo, sem sobrecarregar os agentes além da sua capacidade (Martello e Toth, 1981).

A característica NP-Difícil inerente ao PGA, torna-o de difícil tratamento até a otimalidade, pois um crescimento linear do número de variáveis do problema (agentes e tarefas) provoca um acréscimo exponencial nos recursos da máquina utilizada para resolvê-lo (Osman, 1995). Isso significa que, para instâncias suficientemente grandes, não há uma metodologia comprovadamente eficaz para resolvê-los até a solução ótima global. Nesse sentido, os ramos da pesquisa operacional e da inteligência artificial desenvolveram metodologias, denominadas metaheurísticas, que procuram tratar esses problemas de natureza combinatória a partir de uma exploração mais específica do espaço de busca, procurando soluções, quando não ótimas, ao menos de boa qualidade (Osman, 1995; Michalewicz, 1996).

Sistemas Imunológicos Artificiais (SIA) são técnicas de programação bio-inspirada que possuem alta capacidade de exploração do espaço de soluções. Devido a sua comprovada eficiência e facilidade de implementação, eles vêm sendo largamente utilizados para auxiliar na resolução de problemas combinatoriais (Dasgupta, 1998; Bäck *et al*., 2000).

Neste trabalho é proposto um SIA adaptado para a resolução do PGA. Essa proposta foi avaliada utilizando as principais instâncias encontradas na literatura. A análise dos resultados indica que o método é eficaz e eficiente na resolução do problema, pois além de encontrar soluções de alta qualidade, ele também é capaz de evoluir um grande conjunto de soluções em paralelo. Além disso, o algoritmo proposto encontrou uma quantidade elevada de boas soluções para cada instância testada. Esse aspecto é extremamente relevante, pois as características do problema e o elevado número de restrições fazem com que a dificuldade para encontrar uma solução factível, seja por si só, uma tarefa significativamente complexa.

O restante do texto está organizado da seguinte maneira. Na Seção 2 é introduzido o PGA e a sua formulação matemática. A Seção 3 apresenta os trabalhos relacionados encontrados na literatura. Na Seção 4, é proposto um SIA adaptado para a resolução do problema procurando contornar a questão da complexidade. Experimentos e resultados são demonstrados na Seção 5. Na Seção 6 são disponibilizadas informações sobre o *ToolBox* e o código-fonte do método proposto. Por fim, a Seção 7 apresenta as conclusões e possíveis extensões para este trabalho.

2. Apresentação do Problema

O Problema Generalizado de Atribuição consiste em encontrar uma maneira com mínimo custo (ou máximo benefício) de alocar *n* tarefas para *m* agentes, de modo que cada tarefa seja designada exatamente a um agente de acordo com a sua capacidade.

A formulação do PGA comumente encontrada na literatura é (Chu e Beasley, 1997):

$$Min \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \qquad (1)$$

s.a.

$$\sum_{i=1}^{n} r_{ij} x_{ij} \leq b_j \qquad \forall j \qquad 1 \leq j \leq m \qquad (2)$$

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad \forall i \qquad 1 \leq i \leq n \qquad (3)$$

$$x_{ij} \in \{0,1\} \qquad \begin{cases} \forall i & 1 \leq i \leq n \\ \forall j & 1 \leq j \leq m \end{cases} \qquad (4)$$

Onde:

x_{ij}: 1 se a tarefa *i* foi atribuída ao agente *j*, 0 caso contrário.
c_{ij}: custo associado à designação do agente *j* à tarefa *i.*
r_{ij}: quantidade de recursos consumida pelo agente *j* para a realização da tarefa *i.*
b_j: capacidade do agente *j.*
m: número de agentes.
n: número de tarefas.

No modelo matemático apresentado tem-se: a função objetivo (1) que procura minimizar a somatória dos custos das arestas pertencentes à solução (ou seja, quando $x_{ij} = 1$), a restrição de capacidade (2) que limita o montante de recursos que cada agente pode dispensar. A restrição de atribuição (3) que garante que cada tarefa terá um único agente para atendê-la e a condição de integralidade (4) que é responsável por transformar o PGA em um problema combinatório.

Uma solução é considerada infactível quando alguma tarefa não está sendo atendida ou quando algum agente está utilizando mais recursos do que pode oferecer. Normalmente, o número de soluções infactíveis é de ordem muito superior ao número de soluções factíveis, de modo que os algoritmos que procuram tratar esse problema gastam grande parte do tempo computacional determinando uma solução factível para posteriormente melhorar a sua qualidade (redução do custo) (Osman, 1995; Chu e Beasley, 1997).

O número de possíveis soluções para um PGA qualquer é m^n (Matello e Toth, 1981). As instâncias utilizadas neste trabalho variam de $m = 5$ e $n = 15$ ($\approx 3x10^{10}$ possíveis soluções) até $m = 10$ e $n = 60$ ($= 10^{60}$ possíveis soluções). Isso configura um enorme espaço de soluções, inviabilizando a resolução por algoritmos exatos e justificando a utilização de metaheurísticas.

3. Trabalhos Relacionados

Na literatura são encontradas diversas aplicações do PGA, por exemplo: distribuição de tarefas entre processadores de um sistema de computação paralela (Bokhari, 1987), alocação de salas de aula (Luan e Yao, 1996), planejamento de tarefas de um telescópio espacial (Nowakovski *et al.*, 1999), alocação de pacientes em vôos médicos (Ruland, 1999), carregamento de caminhões (Pigatti, 2003), recuperação de blocos de dados a partir de discos paralelos (Aerts *et al*., 2003), alocação de tarefas independentes em tempo real visando minimização de energia em sistemas heterogêneos (Yu e Prasanna, 2003), roteamento de veículos (Imai *et. al.*, 2007), balanceamento de sistema de redes P2P (Chen e Tsai, 2008), entre muitas outras. Uma revisão geral e bem detalhada das aplicações e técnicas de solução propostas para o PGA pode ser encontrada em Cattrysse e Van Wassenholve (1992), Osman (1995) e em Öncan (2007).

Os primeiros métodos exatos propostos para o PGA baseiam-se no algoritmo *branch-and-bound* (B&B) combinado com relaxação lagrangeana e heurísticas para a obtenção de limites (Fisher *et al.*, 1986). Posteriormente, Guignard e Rosenwein (1989) propuseram métodos baseados em heurísticas duais e em relaxações de programação linear para calcular limitantes superiores e inferiores. Abordagens mais recentes propõem a combinação do algoritmo B&B com o método de geração de colunas, a qual passou a ser conhecida como algoritmo *branch-and-price* (B&P). Savelsbergh (1997) propôs um algoritmo B&P para resolver uma reformulação do PGA como um problema de particionamento de conjuntos. Barnhart *et al.* (1998) usaram algoritmos B&P para a resolução de problema de programação inteira de grande porte. Outros trabalhos que adotaram com sucesso essas técnicas são: Narciso e Lorena (1999), Senne *et al.* (2004), Yagiura *et al.* (2004), Yagiura *et al.* (2006)**,** Jeet e Kutanoglu (2007) e Litvinchev e Rangel (2008).

Em relação aos métodos aproximados, em geral, métodos heurísticos, Martello e Toth (1981) apresentaram uma forma de solução dividida em duas fases: na primeira, as atribuições são calculadas de modo a satisfazer um certo critério (por exemplo, minimizar uma função de penalização), sendo, posteriormente, melhoradas numa segunda fase. Outra abordagem heurística dividida em duas partes foi desenvolvida por Amini e Racer (1994), na qual os autores fazem uma cuidadosa comparação de vários métodos até então propostos.

Metaheurísticas, como Busca Tabu (Laguna *et al.*, 1995), *Simulated Annealing* (Osman,

1995), Algoritmos Genéticos (Chu e Beasley, 1997), Sistemas Híbridos (Feltl e Raidl, 2004), Algoritmos de Otimização por Colônia de Formigas (Chen *et al.*, 2005) e Redes Neurais Artificiais (Monfared e Etemadi, 2006), também já foram propostas para resolver o PGA. Até a conclusão destre trabalho, não foi encontrada na literatura nenhuma proposta que explorasse as vantagens dos sistemas imunológicos artificiais para resolver o PGA.

4. Sistema Imunológico Artificial

Os Sistemas Imunológicos Artificiais (SIA), assim como outras técnicas inspiradas na natureza, tentam extrair idéias dos sistemas biológicos para desenvolver ferramentas que resolvam problemas computacionais. Essa técnica já vem sendo utilizada em diversas áreas, como reconhecimento de padrões, detecção de falhas e anomalias, segurança computacional, otimização, controle, robótica, *scheduling*, análise de dados, aprendizagem de máquina, entre outras (Dasgupta, 1998; De Castro, 2001).

A simulação de processos evolutivos naturais objetivando resolver problemas de explosão combinatória e multimodais demonstrou ser uma estratégia eficaz e robusta. A otimização do comportamento de um sistema obtida através da simulação de processos evolutivos representa uma abordagem poderosa para aprendizagem de máquina e estudo de fenômenos auto-organizáveis. A implementação computacional desses métodos de simulação da evolução, chamada computação evolutiva, possibilita a determinação de soluções para várias classes de problemas (Bäck *et al.*, 2000). Além dessas, outras linhas de pesquisa ainda mais recentes têm surgido como novos paradigmas de computação.

4.1. Algoritmo Proposto

Se considerarmos o PGA como um problema de otimização em grafos, uma possível solução factível corresponde a um conjunto de nós *n* (tarefas) conectados ao conjunto de nós *m* (agentes), sendo que as seguintes restrições devem ser satisfeitas:

- Cada nó de *n* deve estar conectado a exatamente um nó de *m*;
- A capacidade associada a cada nó de *m* não pode ser menor que a somatória dos recursos consumidos pelos nós de *n* conectados a ele.

Essas características levaram ao desenvolvimento de um SIA nos moldes propostos por Dasgupta, (1998) e De Castro (2001) para tentar encontrar um conjunto solução para o problema proposto. Tomando como base o algoritmo genético proposto por Chu e Beasley (1997), o repertório é formado por anticorpos (soluções factíveis ou infactíveis) que geram novos indivíduos herdeiros das características dos anticorpos-pai (clones) e evoluem por meio de operadores de maturação. Assim, a medida de afinidade (*fitness*) é utilizada de forma a privilegiar a reprodução dos indivíduos mais adaptados ao ambiente (seleção clonal). Para penalizar soluções infactíveis, foi utilizada uma medida de desafinidade, que mede o quanto uma solução é infactível. Dessa forma, com o passar de um número finito de gerações, espera-se obter um repertório de anticorpos com um alto grau de afinidade (soluções com baixo custo) e uma desafinidade igual a zero (soluções factíveis).

4.1.1. Estrutura

Representação do Anticorpo (Ab_i): cada anticorpo é representado por um vetor linha com *n* posições. Cada índice do vetor é associado a uma tarefa (número inteiro) e o seu valor corresponde ao agente que irá realizá-la.

Foram implementados três métodos distintos para a geração de anticorpos:

1. **Aleatório** – utiliza distribuição uniforme. Cada tarefa possui a mesma probabilidade de ser alocada a um agente;
2. **Roleta** – a distribuição de probabilidade é proporcional à capacidade do agente. Quanto maior for à capacidade do agente, maior será a probabilidade dele ser escolhido para realizar uma tarefa. O objetivo é reduzir a geração de soluções infactíveis;

Factível – somente anticorpos representando soluções factíveis são gerados. A estratégia adotada para obter um indivíduo factível é a seguinte:

1º) Para cada tarefa i:

- Ordenar os agentes que podem atender a tarefa *i* usando a menor quantidade de recursos r_{ij};
- Escolher aleatoriamente um dos *k* primeiros agentes ordenados e alocar a tarefa *i* para ser atendida por ele;
- Se a capacidade do agente *j* não for suficiente para atender a tarefa *i*, então escolher novamente, aleatoriamente, um dos *k* primeiros agentes ordenados exceto o agente *j*.

2º) Repetir este processo, até que todas as tarefas sejam atendidas.

Seja *p* a quantidade de anticorpos que constituí o repertório (**Ab**). Esse repertório pode ser representado por uma matriz *p* x *n*.

Inicialização: é feita por um dos três métodos de geração de anticorpos (aleatório, roleta, factível) a ser escolhido pelo usuário.

Clonagem: é feita gerando-se *z* clones idênticos de cada anticorpo. Esses anticorpos clonados irão formar a população de clones **C**.

Maturação: a população de clones **C** passa por um processo de maturação de afinidade (mutação) gerando uma nova população de clones maturados **C***. Essa mutação ocorre sorteando-se aleatoriamente *q* posições (tarefas) do vetor anticorpo e substituindo o conteúdo dos índices (agentes) por outros escolhidos aleatoriamente. A quantidade de índices escolhidos para maturação é inversamente proporcional à qualidade do anticorpo.

Medida de Afinidade (*fitness*): para medir a afinidade é utilizada a própria função objetivo do problema.

Medida de Desafinidade (*unfitness*): é calculada como a somatória das unidades excedidas de capacidade de cada agente, ou seja, representa o quanto uma solução é infactível. Assim, a desafinidade será nula se todos os agentes não excederem a sua capacidade.

Seleção: é inspirada nos princípios da seleção clonal (De Castro, 2001). O método utilizado é elitista (Michalewicz, 1996), que privilegia o indivíduo mais adaptado. Assim, o clone com maior afinidade (ou menor desafinidade) é selecionado entre os indivíduos do repertório de clones, comparado com o seu anticorpo-pai e apenas o melhor será preservado.

Diversidade: a chance de obter anticorpos com alto grau de afinidade aumenta em um conjunto de soluções diversificadas. Para estudar o comportamento da diversidade do repertório, foi utilizada a definição proposta por Almeida *et al.*, (2007):

Diversidade é a razão entre o número de indivíduos únicos (aqueles que não se repetem na população) e o tamanho da população. Na métrica de similaridade adotada, uma solução é dita igual à outra se as duas forem exatamente iguais em todos os índices do vetor.

Outras métricas de similaridade foram testadas, por exemplo, a porcentagem de tarefas executadas pelos mesmos agentes que se repetem entre uma solução e outra. Entretanto, todas apresentaram desempenho pior devido à alta sensibilidade entre a representação adotada e a qualidade da solução. Note que, a alteração de valor de uma única posição numa solução pode alterar muito a sua qualidade, podendo transformar uma boa solução factível em uma solução infactível.

Um procedimento de injeção de diversidade é utilizado sempre que a diversidade do repertório cai abaixo de um limiar *l*. Nesse caso, o repertório passa por um processo de supressão que retira

uma parcela de anticorpos que representam soluções similares, e completa-o com novos indivíduos criados pelo método de geração de anticorpos factíveis.

Busca Local: auxilia na redução da desafinidade (quando existir) e/ou na melhoria da afinidade de um anticorpo obtido por seleção clonal. A busca local é dividida em duas partes:

1ª Parte - Diminuir a Desafinidade:

Seja T_i o agente designado para a tarefa i em uma determinada solução. Então, para cada agente j, se o consumo excedeu a sua capacidade, isto é,

$$\sum_{i \in I,\, T_i = j} r_{ij} > b_j ,$$

é escolhida aleatoriamente uma tarefa q com $T_q = j$ e designada para o próximo agente (na ordem de $j + 1,..., m, 1,..., j - 1$), verificando-se a sua disponibilidade.

2ª Parte - Aumentar a Afinidade:

Para cada tarefa i, achar $\min_{j,\, j \in J} c_{ij}$ que satisfaz:

$$c_{ij} < c_{T_I i} \quad e \quad \left(\sum_{q \in I,\, T_q = j} r_{jq} \right) + r_{ij} \leq b_j$$

Se um determinado j pode ser localizado, designar a tarefa i do agente T_i para o agente j.

Esses procedimentos são realizados apenas uma vez para cada anticorpo-filho. Portanto, a busca local não necessariamente transforma soluções infactíveis em factíveis e nem tampouco localiza uma solução ótima, ela apenas proporciona a redução da infactibilidade das soluções e melhora o valor da função objetivo.

4.1.2. Pseudocódigo

A implementação computacional do método proposto está resumida pelo pseudocódigo apresentado no Quadro I.

5. Resultados Experimentais

Para avaliar o desempenho do algoritmo utilizamos 60 instâncias disponíveis na base de dados OR-Library[1], um repositório público de instâncias para uma grande quantidade de problemas de pesquisa operacional.

A implementação computacional foi feita em MatLab7® e todos os testes foram realizados em um microcomputador Atlhon® XP 2200 – 2.0 GHz e 512 MB de memória. Todas as instâncias testadas correspondem a problemas de maximização e, portanto, a medida de afinidade utilizada corresponde ao valor da própria função objetivo.

A Tabela 1 apresenta os parâmetros que foram utilizados para realizar os experimentos.

A maturação de cada clone é proporcional a sua afinidade. O número de tarefas que são alteradas em cada clone está descrita na Tabela 2, sendo que $\mathbf{Af_m}$ corresponde à afinidade do melhor anticorpo do repertório e $\mathbf{Af_c}$ à afinidade do clone a ser maturado.

Esses parâmetros foram empiricamente calibrados de forma a obter o melhor desempenho para as instâncias testadas. Note que, o tamanho do repertório, bem como o número de gerações, varia de instância para instância conforme a complexidade do problema (ver Tabela 2). Em relação à criação do repertório inicial, o Método de Roleta foi o que apresentou melhores resultados, uma vez que o método de geração de soluções factíveis consome muito tempo de processamento, e o método de soluções aleatórias freqüentemente cria anticorpos infactíveis de baixa qualidade. Dessa forma, optamos por aquele que oferece o melhor custo/benefício.

[1] http://people.brunel.ac.uk/ ~mastjjb/jeb/info.html

Tabela 1 **-** Parâmetros utilizados

Parâmetro	***Valor***
Tamanho da população (*p*)	Variável
Número de gerações (*ger*)	Variável
Número de clones gerados por anticorpo (*z*)	10
Limiar de diversidade (*l*)	70%
Quantidade de rodadas por experimento	10
Criação do repertório inicial (**Ab**)	Método da Roleta
Busca Local	Habilitada

```
Ab   ← inicializar o repertório com p anticorpos;
 f   ← afinidade(Ab);
 d   ← desafinidade(Ab);
geração ← 1;
enquanto geração ≤ gen faça
    para cada anticorpo i de Ab faça
        se d_i = 0 então
            C ← clonar(Ab_i,z);
        senão
            C ← criar_anticorpos(z);
        fim-se;
            f ← afinidade(C);
           C* ← maturar(C,f);
            f ← afinidade(C*);
            d ← desafinidade(C*);
      Ab_{w} ← selecionar(C*,f,d);
      Ab_{w} ← busca_local(Ab_{w});
        Ab_i ← selecionar (Ab_{w},Ab_i);
    fim-para;
    Atualizar f e d;
    se diversidade(Ab) < l então
            s ← Supressão(Ab) //aplica supressão e retorna o
número de soluções extraídas;
        Ab_{d} ← criar_anticorpos(s);
          Ab ← Ab ∪ Ab_{d}
    fim_se;
    geração ← geração + 1;
fim_enquanto;
```

Quadro I **–** Pseudocódigo do SIA

Tabela 2 **-** Maturação x Afinidade

Num. tarefas alteradas	*Afinidade do clone*
2	$\mathbf{Af_c} \geq 95\%\ \mathbf{Af_m}$
3	$95\%\ \mathbf{Af_m} > \mathbf{Af_c} \geq 90\%\ \mathbf{Af_m}$
4	$90\%\ \mathbf{Af_m} > \mathbf{Af_c} \geq 80\%\ \mathbf{Af_m}$
5	$80\%\ \mathbf{Af_m} > \mathbf{Af_c} \geq 70\%\ \mathbf{Af_m}$
6	$\mathbf{Af_c} < 70\%\ \mathbf{Af_m}$

A Tabela 3 apresenta o melhor resultado obtido em cada uma das 10 rodadas. As dimensões dos problemas-teste tratados variam de $m = 5$ e $n = 15$ ($\approx 3\text{x}10^{10}$ possíveis soluções) a $m = 10$ e $n = 60$ ($= 10^{60}$ possíveis soluções).

Tabela 3 **-** Resultados computacionais

Inst	**Ótimo**	**Melhor Solução encontrada em cada uma das 10 rodadas**										**Média**	**σ**
gap1-1	336	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	336,0	0,00
gap1-2	327	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	327,0	0,00
gap1-3	339	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	339,0	0,00
gap1-4	341	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	341,0	0,00
gap1-5	326	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	326,0	0,00
gap2-1	434	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	434,0	0,00
gap2-2	436	**o**	434	**o**	**o**	**o**	**o**	432	**o**	**o**	**o**	435,4	1,35
gap2-3	420	**o**	418	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	419,8	0,63
gap2-4	419	417	**o**	**o**	**o**	**o**	417	**o**	417	**o**	**o**	418,4	0,97
gap2-5	428	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	428,0	0,00
gap3-1	580	**o**	578	**o**	**o**	**o**	578	**o**	**o**	**o**	578	579,4	0,97
gap3-2	564	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	564,0	0,00
gap3-3	573	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	573,0	0,00
gap3-4	570	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	570,0	0,00
gap3-5	564	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	564,0	0,00
gap4-1	656	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	656,0	0,00
gap4-2	644	**o**	643	643	**o**	**o**	641	643	643	643	**o**	643,2	0,92
gap4-3	673	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	673,0	0,00
gap4-4	647	646	646	646	646	646	646	646	646	**o**	646	646,1	0,32
gap4-5	664	660	**o**	659	**o**	**o**	662	662	662	662	659	661,8	1,93
gap5-1	563	562	**o**	562	**o**	562	**o**	**o**	**o**	**o**	562	562,6	0,52
gap5-2	558	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	558,0	0,00
gap5-3	564	**o**	**o**	**o**	**o**	563	563	**o**	**o**	**o**	**o**	563,8	0,42
gap5-4	568	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	568,0	0,00
gap5-5	559	558	558	558	558	558	**o**	558	558	558	558	558,1	0,32
gap6-1	761	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	761,0	0,00
gap6-2	759	**o**	**o**	**o**	**o**	**o**	**o**	**o**	758	758	**o**	758,8	0,42
gap6-3	758	755	755	**o**	753	753	753	756	753	754	757	754,7	1,83
gap6-4	752	**o**	**o**	**o**	**o**	**o**	**o**	**o**	751	**o**	**o**	751,9	0,32
gap6-5	747	746	746	746	746	746	**o**	746	**o**	746	746	746,2	0,42
gap7-1	942	941	941	**o**	941	941	**o**	941	941	941	941	941,2	0,42
gap7-2	949	948	948	**o**	947	948	**o**	948	948	**o**	948	948,2	0,63
gap7-3	968	**o**	966	**o**	967	967	967	**o**	**o**	967	**o**	967,4	0,70
gap7-4	945	**o**	944	**o**	944	**o**	944	944	**o**	944	943	944,3	0,67
gap7-5	951	950	950	**o**	**o**	**o**	**o**	**o**	950	**o**	**o**	950,7	0,48

gap8-1	1133	1126	**o**	1129	1126	1126	1127	1127	1131	1131	1128	1128,4	2,50
gap8-2	1134	1125	1130	**o**	1126	1129	1133	1131	1131	1126	1126	1129,1	3,21
gap8-3	1141	1137	1138	**o**	1138	1138	**o**	1139	1137	1138	1139	1138,6	1,43
gap8-4	1117	1110	1114	**o**	1114	**o**	1110	1114	1111	1109	1110	1112,6	2,99
gap8-5	1127	**o**	1124	1125	1125	1125	**o**	**o**	1124	**o**	1124	1125,5	1,36
gap9-1	709	**o**	**o**	**o**	**o**	708	**o**	**o**	**o**	**o**	**o**	708,9	0,32
gap9-2	717	715	715	715	715	716	715	**o**	715	716	715	715,4	0,69
gap9-3	712	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	712,0	0,00
gap9-4	723	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	723,0	0,00
gap9-5	706	**o**	703	704	702	701	702	702	703	702	704	702,9	1,45
gap10-1	958	**o**	**o**	**o**	957	**o**	**o**	**o**	**o**	**o**	**o**	957,9	0,32
gap10-2	963	962	962	962	**o**	**o**	962	962	962	962	962	962,2	0,42
gap10-3	960	957	956	958	958	957	**o**	957	957	957	957	957,4	1,07
gap10-4	947	945	946	944	945	945	**o**	**o**	945	944	945	945,3	1,06
gap10-5	947	945	**o**	945	**o**	946	945	946	945	945	946	945,7	0,82
gap11-1	1139	1138	1138	1137	1137	1135	1136	**o**	**o**	1136	1138	1137,3	1,34
gap11-2	1178	1177	**o**	1177	1174	1175	1177	1176	1177	1175	1174	1176,0	1,41
gap11-3	1195	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	1195,0	0,00
gap11-4	1171	**o**	1170	**o**	1168	**o**	1169	1170	1169	1169	**o**	1169,9	1,10
gap11-5	1171	1168	1169	1168	1168	1169	1168	**o**	**o**	1169	1168	1168,9	1,19
gap12-1	1451	**o**	**o**	**o**	1450	1449	1449	1449	**o**	1450	1449	1450,0	0,94
gap12-2	1449	1447	1447	1447	1447	1447	**o**	1447	1448	1448	1447	1447,4	0,70
gap12-3	1433	1429	**o**	**o**	1429	**o**	1430	1430	1429	1430	1429	1430,5	1,78
gap12-4	1447	1444	**o**	1444	1446	1446	1446	1445	1444	1446	1444	1445,2	1,13
gap12-5	1446	1445	1444	1445	1445	1445	**o**	1445	1445	1445	1445	1445,0	0,47

o = solução ótima / **σ** = desvio padrão

Note que, em termos qualitativos, o método proposto obteve um excelente desempenho, uma vez que, este conseguiu encontrar a solução ótima para todas as instâncias utilizadas (Tabela 3). Entretanto, o maior destaque concentra-se nos termos quantitativos, pois o algoritmo foi capaz de evoluir um grande conjunto de soluções em paralelo, de forma que, no final do processo evolutivo uma quantidade elevada de boas soluções foi encontrada para cada instância testada além da solução ótima. Esse resultado pode ser explicado pela forte capacidade do SIA em manter os ótimos locais, devido à sua eficiência na manutenção da diversidade, não permitindo que todo o repertório de soluções convirja para um único ponto no espaço de busca. Nesse caso, a manutenção e a evolução de um conjunto de soluções de alta qualidade são aspectos importantes quando surge a possibilidade de que uma nova restrição venha a ser imposta posteriormente.

Para melhor visualização, a Tabela 4 apresenta os parâmetros variáveis utilizados para a resolução de cada problema-teste, bem como o tempo de processamento e a quantidade de soluções factíveis obtidas.

Note que, para uma grande parcela dos testes realizados, o método foi capaz de encontrar um conjunto solução final com cardinalidade igual ao número de anticorpos do repertório inicial. Em relação às exceções, a instância gap7-3 é aquela cuja proporção entre o tamanho do repertório e quantidade de soluções é menor. Nesse caso, o conjunto solução final possui cardinalidade 22% menor que o repertório inicial, pois, partindo de 300 anticorpos (pontos no espaço de busca), obteve-se 234 boas soluções.

Para verificar a representatividade de cada solução dentro do conjunto de soluções obtido para cada instância computamos uma matriz de distâncias *D*. Essa matriz é quadrada com ordem igual ao número de soluções. Cada posição D_{ij} corresponde quanto à solução *i* se difere da solução *j* em termos percentuais, ou seja, quantos agentes foram designados para realizar tarefas diferentes nas duas soluções. Em média, cada solução é, aproximadamente, 30% diferente das outras dentro do conjunto solução, ou seja, em média, seria necessário alocar 30% das tarefas

para outros agentes de forma a transformar uma solução *i* em uma solução *j*. Esse resultado comprova que além do método evoluir um conjunto amplo de soluções, grande parte dessas soluções é representativa. Para maiores detalhes, consulte o conjunto completo de resultados disponível em http://www.dt.fee.unicamp.br/~tiago/research.php.

O fato de evoluir um conjunto de soluções é interessante quando novas restrições podem ser apresentadas ao problema, possibilitando uma posterior tomada de decisão. Essa característica permite que um ou mais planos de contingência sejam especificados para atender eventuais fatores que podem ocorrer.

Ao compararmos o método proposto com outros algoritmos presentes na literatura, constatamos que o SIA consome mais tempo computacional, pois diferentemente de outras heurísticas, ele evolui várias "populações de soluções" em paralelo. Por outro lado, essa característica proporciona uma exploração mais eficiente do espaço de busca, permitindo que um conjunto amplo de soluções seja encontrado.

Tabela 4 **-** Parâmetros variáveis, tempo de processamento e número de soluções factíveis

Inst	**Num Ger**	**Tam Pop**	**Tempo (s)**	**Qt Sol**	**Inst**	**Num Ger**	**Tam Pop**	**Tempo (s)**	**Qt Sol**
gap1-1	100	50	15	48	gap6-2	300	100	92	91
gap1-2	100	50	50	50	gap6-3	300	100	100	100
gap1-3	100	50	18	50	gap6-4	300	100	98	94
gap1-4	100	50	16	50	gap6-5	300	100	94	100
gap1-5	100	50	14	48	gap7-1	500	300	481	248
gap2-1	100	50	15	50	gap7-2	500	300	440	277
gap2-2	100	100	28	98	gap7-3	500	300	424	234
gap2-3	100	100	26	99	gap7-4	500	300	559	283
gap2-4	100	100	28	84	gap7-5	500	300	480	240
gap2-5	100	50	10	44	gap8-1	500	100	224	100
gap3-1	200	100	51	96	gap8-2	500	100	207	100
gap3-2	200	100	52	99	gap8-3	500	100	207	100
gap3-3	200	100	50	99	gap8-4	500	100	213	100
gap3-4	200	100	49	97	gap8-5	500	100	197	100
gap3-5	200	100	51	94	gap9-1	700	100	232	100
gap4-1	200	100	50	87	gap9-2	700	100	233	96
gap4-2	200	100	57	98	gap9-3	700	100	221	89
gap4-3	200	100	50	98	gap9-4	700	100	196	98
gap4-4	200	100	57	94	gap9-5	700	100	275	100
gap4-5	200	100	49	100	gap10-1	700	100	220	97
gap5-1	300	100	79	96	gap10-2	700	100	240	100
gap5-2	300	100	96	85	gap10-3	700	100	222	97
gap5-3	300	100	82	100	gap10-4	700	100	274	100
gap5-4	300	100	80	99	gap10-5	700	100	267	100
gap5-5	300	100	101	83	gap11-1	1000	100	355	100
gap6-1	300	100	85	100	gap11-2	1000	100	329	98
gap6-2	300	100	92	91	gap11-3	1000	100	238	93
gap6-3	300	100	100	100	gap11-4	1000	100	320	99
gap6-4	300	100	98	94	gap11-5	1000	100	352	96
gap6-5	300	100	94	100	gap12-1	1000	100	330	100
gap7-1	500	300	481	248	gap12-2	1000	100	339	100
gap7-2	500	300	440	277	gap12-3	1000	100	420	100
gap7-3	500	300	424	234	gap12-4	1000	100	336	100
gap7-4	500	300	559	283	gap12-5	1000	100	353	100

Qt Sol –Quantidade de soluções factíveis obtidas no melhor resultado de 10 rodadas

6. *ToolBox* e Código Fonte

Um *toolbox* completo, contendo todas as funções, instâncias, resultados, manual em português e uma interface gráfica para execução do programa e *plotagem* das soluções, estão disponíveis gratuitamente em http://www.dt.fee.unicamp.br/~tiago/research.php.

7. Conclusões e Trabalhos Futuros

O Problema Generalizado de Atribuição (PGA) é um famoso problema de otimização combinatória utilizado em diversas aplicações práticas. A característica NP-Difícil faz com que a sua resolução seja um desafio complexo que pode tornar inviável a utilização de métodos de otimização clássicos, dependendo da quantidade de agentes e/ou tarefas envolvidos.

Atualmente, existe um elevado número de metaheurísticas que são utilizadas para resolver muitos tipos de problemas dessa natureza. A maioria delas procura simular processos evolutivos, como os Algoritmos Genéticos (AG) e os Sistemas Imunológicos Artificiais (SIA).

Neste trabalho foi proposto um SIA adaptado para a resolução do PGA. O método proposto provou ser altamente eficaz e eficiente, pois além de ter encontrado a solução ótima para todas as instâncias testadas, ele também conseguiu preservar um conjunto diversificado de boas soluções (ótimos locais). Essa característica permite que um plano de contingência seja adotado, possibilitando uma rápida tomada de decisão em caso de novas restrições serem incorporadas. Isso faz com que o sistema torne-se mais robusto e menos sensível a falhas. Adicionalmente, um *toolbox* completo (*open source*) foi disponibilizado gratuitamente.

Alternativamente, algumas técnicas encontradas mais recentemente na literatura podem ser sugeridas para aperfeiçoar o método proposto, tais como: Teoria da Rede Imunológica, Teoria do Próprio e Não Próprio, dentre outras. Além disso, uma possível hibridização do método com outras metaheurísticas, como Algoritmos Genéticos ou Algoritmos de Otimização por Colônia de Formigas, pode aumentar a capacidade de exploração do espaço de busca proporcionando aumento de desempenho.

Agradecimentos

Os autores são gratos à CNPq, CAPES e FAPESP pelo apoio financeiro à realização desta pesquisa.

Referências Bibliográficas

Aerts, J., Korst, J. e Spieksma, F. (2003), Approximation of a Retrieval Problem for Parallel Disks, *Proceedings of Italian Conference on Algorithms and Complexity*, 178-188.

Almeida, T.A., Yamakami, A. e Takahashi, M. T. (2007), Sistema Imunológico Artificial para Resolver o Problema da Árvore Geradora Mínima Fuzzy, *Pesq. Operacional*, 27, 131-154.

Amini, M.M. e Racer, M. (1994), A Rigorous Comparison of Alternative Solution Methods for the Generalized Assignment Problem, *Management Science*, 40, 868-890.

Bäck, T., Fogel, D.B. e Michalewicz, Z., *Evolutionary Computation 1. Basic Algorithms and Operators.* Institute of Physiscs Publishing (IOP), Bristol & Philadelphia, 2000.

Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P. e Vance, P.H. (1998), Branch-and-Price: Column Generation for Solving Huge Integer Programs, *Operations Research*, 46, 316-329.

Bokhari, S.H., *Assignment Problems in Parallel and Distributed Computing*. Kluwer Academic Publisher, Boston, USA, 1987.

Catrysse, D.C. e Van Hossenholve, L.N. (1992), A Survey of Algorithms for the Generalized Assignment Problem, *EJOR*, 60, 260-272.

Chen, Y.-F., Liu, Y.-S., Fan, J. e Zhao, J.-H. (2005), Niching Genetic and Ant Colony Optimization Algorithm for Generalized Assignment Problem, *Transactions of Beijing Institute of Technology*, 25, 490-494.

Chen, C. e Tsai, K.-C. (2008), The Server Reassignment Problem for Load Balancing in Structured P2P Systems, *IEEE Transactions on Parallel and Distributed Systems*, 19, 234-246.

Chu, P.C. e Beasley, J.E. (1997), A Genetic Algorithm for the Generalised Assignment Problem, *Computers Operacional Research*, 24, 17-23.
Dasgupta, D., *Artificial Immune Systems and Their Applications*, Springer, 1998.
De Castro, L.N., Engenharia Imunológica: Desenvolvimento e Aplicação de Ferramentas Computacionais Inspiradas em Sistemas Imunológicos Artificiais, Tese de Doutorado, Faculdade de Engenharia Elétrica e de Computação, Unicamp, Campinas, Brasil, 2001.
Feltl, H. e Raidl, G.R. (2004), An Improved Hybrid Genetic Algorithm for the Generalized Assignment Problem, *Proceedings of the ACM Symposium on Applied Computing,* 990-995.
Fisher, M.L., Jaikumar, R. e Van Wassenhove, L.N. (1986), A Multiplier Adjustment Method for the Generalized Assignment Problem, *Management Science*, 32, 1095-1103.
Guignard, M. e Rosenwein, M. (1989), An Improved Dual-Based Algorithm for the Generalized Assignment Problem, *Operations Research*, 37, 658-663.
Imai, A., Nishimura, E. e Current, J. (2007), A Lagrangian Relaxation-Based Heuristic for the Vehicle Routing with Full Container Load, *EJOR*, 176, 87-105.
Jeet, V. e Kutanoglu, E. (2007), Lagrangian Relaxation Guided Problem Space Search Heuristics for Generalized Assignment Problems, *EJOR*, 182, 1039-1056.
Laguna, M., Kelly, J.P., Gonzales-Velarde, J.L. e Glover, F. (1995), Tabu Search for the Multilevel Generalized Assignment Problem, *EJOR*, 42, 677–687.
Litvinchev, I.S. e Rangel, S. (2008), Comparison of Lagrangian Bounds for One Class of Generalized Assignment Problems, *Comp. Mathematics and Math. Physics*, 48, 739-746.
Luan, F. e Yao, X. (1996), Solving Real-World Lecture Room Assignment Problems by Genetic Algorithms. *Complex Systems*, 148-160.
Matello, S. e Toth, P. (1981), An Algorithm for the Generalized Assignment Problem, *Operational Research*, 81, 589-603.
Michalewicz Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, 1996.
Monfared, M.A.S. e Etemadi, M. (2006), Impact of Energy Function Structure on Solving Generalized Assignment Problem Using Hopfield Neural Network, *EJOR*, 168, 645-654.
Narciso, M.G. e Lorena, L.A.N. (1999), Lagrangean/Surrogate Relaxation for Generalized Assignment Problems, *EJOR*, 114, 165-177.
Nowakovski, J., Schwärzler, W. e Triesch, E. (1999), Using the Generalized Assignment Problem in Scheduling the ROSAT Space Telescope. *European Journal of Operational Research*, 112, 531-541.
Öncan, T. (2007), A Survey of the Generalized Assignment Problem and Its Application, *Information Systems and Operational Research*, 45, 123-141.
Osman, I.H. (1995), Heuristics for the Generalized Assignment Problem: Simulated Annealing and Tabu Search Approaches, *OR Spektrum*, 17, 211-225.
Pigatti, A.A., Modelos e Algoritmos para o Problema de Alocação Generalizada e Aplicações. *Dissertação de Mestrado*, Pontifícia Universidade Católica do Rio de Janeiro, 2003.
Ruland, K.S. (1999), A Model for Aeromedical Routing and Scheduling, *International Transactions in Operational Research*, 6, 57-73.
Savelsbergh, M.W.P. (1997), A Branch-and-Price Algorithm for the Generalized Assignment Problem, *Operations Research*, 45, 831-841.
Senne, E.L.F., Lorena, L.A.N. e Salomão, S.N.A. (2004), Uma Abordagem de Geração de Colunas para o Problema Generalizado de Atribuição, *Produção*, 4, 2950-2957.
Yagiura, M., Ibarakati, T. e Glover, F. (2004), An Ejection Chain Approach for the Generalized Assignment Problem, *INFORMS Journal on Computing*, 16, 133-151.
Yagiura, M., Ibarakati, T. e Glover, F. (2006), A Path Relinking Approach with Ejection Chains for the Generalized Assignment Problem, *EJOR*, 169, 548-569.
Yu, Y. e Prasanna, V.K. (2003), Resource Allocation for Independent Real-Time Tasks in Heterogeneous Systems for Energy Minimization, *Journal of Information Science and Engineering*, 19, 433-449.