Algoritmo de Colônia de Formigas e Sistema Imunológico Artificial aplicado ao Problema de Designação Generalizada

Article · January 2004

CITATIONS
0

READS
541

5 authors, including:

Tiago A. Almeida
Universidade Federal de São Carlos
69 PUBLICATIONS **903** CITATIONS
SEE PROFILE

Fabio Luiz Usberti
University of Campinas
28 PUBLICATIONS **188** CITATIONS
SEE PROFILE

Algoritmo de Colônia de Formigas e Sistema Imunológico Artificial aplicado ao Problema de Designação Generalizada

Almeida, T.A.; Costa, J.C.; Dias, C.H.; Ferreira, H.M.; Usberti, F.L.

1. INTRODUÇÃO

Esse trabalho tem por objetivo o estudo de técnicas computacionais inspiradas na natureza que tratam de problemas de otimização discreta. Os algoritmos de colônias de formigas (ACO) e sistemas imunológicos artificiais (SIA) são duas dessas técnicas, possuindo uma excelente capacidade de exploração do espaço de soluções de problemas combinatoriais. Muitos desses problemas são de difícil tratamento até a otimalidade, de modo que um acréscimo linear do número de variáveis faz crescer exponencialmente a memória e/ou tempo de processamento necessários para resolver o problema.

O GAP, problema de designação generalizada, foi o problema utilizado nesse trabalho. Trata-se de designar um conjunto de agentes para atender um conjunto de tarefas a um custo mínimo, sem sobrecarregar os agentes além da sua capacidade.

Nesse trabalho foram implementados algoritmos ACO e SIA específicos para o GAP, a partir de onde foram efetuados testes com instâncias da literatura. A análise dos resultados sugere que ambas as técnicas foram excelentes na obtenção de soluções de boa qualidade. Verificou-se que o ACO obteve soluções de boa qualidade em frações de segundo para algumas instâncias. O SIA, por sua vez, conseguiu explorar o espaço de busca, encontrando soluções de boa qualidade e mantendo a diversidade do repertório de soluções obtidas.

2. PROBLEMA DE DESIGNAÇÃO GENERALIZADA (GAP)

2.1. Definição

O problema de designação generalizada (generalised assignment problem - GAP) e um problema de otimização combinatorial NP-Difícil muito conhecido na literatura e que envolve encontrar o mínimo custo (ou máximo benefício) de alocar *n* tarefas para *m* agentes de modo que cada trabalho seja designado exatamente a um agente, sujeito à capacidade desse último.

A característica NP-Difícil, inerente ao GAP, torna-o de difícil tratamento até a otimalidade, pois um crescimento linear do número de variáveis do problema (agentes e trabalhos) provoca um acréscimo exponencial nos recursos da máquina utilizada para resolvê-lo, ou seja, tempo de processamento e/ou memória. Problemas dessa natureza ainda são considerados intratáveis pela literatura, ou seja, para instâncias suficientemente grande, 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 meta-heurísticas, que procuram tratar esses problemas difíceis de natureza combinatorial a partir da exploração "inteligente" do espaço de busca, procurando soluções, quando não ótimas, ao menos de boa qualidade.

2.2. Representação em Grafo

O problema GAP pode ser representado facilmente com uma estrutura de grafo, ou seja, um conjunto de nós (agentes e tarefas) e um conjunto de arestas (as possíveis designações agentes-tarefas). A Figura 1 ilustra essa idéia, onde se pode observar que o conjunto de nós foi particionado em dois subconjuntos, os nós agentes e os nós tarefas, de modo que nós pertencentes ao mesmo subconjunto não apresentam adjacência. Essa característica é a que define um grafo bipartido, muito apropriado para representar um GAP. Para cada aresta do grafo se associam dois pesos distintos: custo (ou benefício) e recurso. Enquanto o custo é parâmetro importante para a função objetivo do problema, o recurso é fundamental para a restrição de capacidade do problema.

Uma solução GAP é considerada infactível quando as tarefas designadas a um (ou mais) agente(s) estão consumindo recursos aquém da capacidade do(s) mesmo(s). Normalmente, o número de soluções infactíveis de um GAP é de ordem muito superior ao número de soluções factíveis, de modo que os algoritmos que procuram tratar esse problema dispensam grande parte do tempo computacional para determinar uma solução factível para posterior melhoria de sua qualidade (redução do custo).

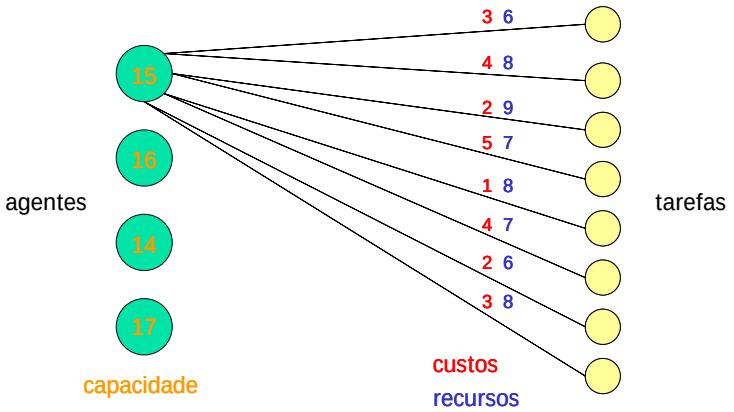

Figura 1. Representação do GAP a partir da estrutura de um grafo.

Retomando a definição do problema, sabe-se que um agente pode ser designado para qualquer tarefa. Nesse sentido, um nó agente é adjacente a todos os nós tarefas, o que caracteriza um

grafo bipartido completo (Figura 2a) que é a melhor representação em grafo para o GAP. Uma solução do GAP terá sempre a forma apresentada pela Figura 2b, ou seja, uma única aresta incidindo cada nó tarefa. Se tal solução é factível ou não, dependerá da verificação da capacidade de cada agente. É possível observar que nem todos os agentes são solicitados (como o terceiro de cima para baixo da Figura 2b), de modo que aqueles que custam mais para a função objetivo serão utilizados somente quando necessários.

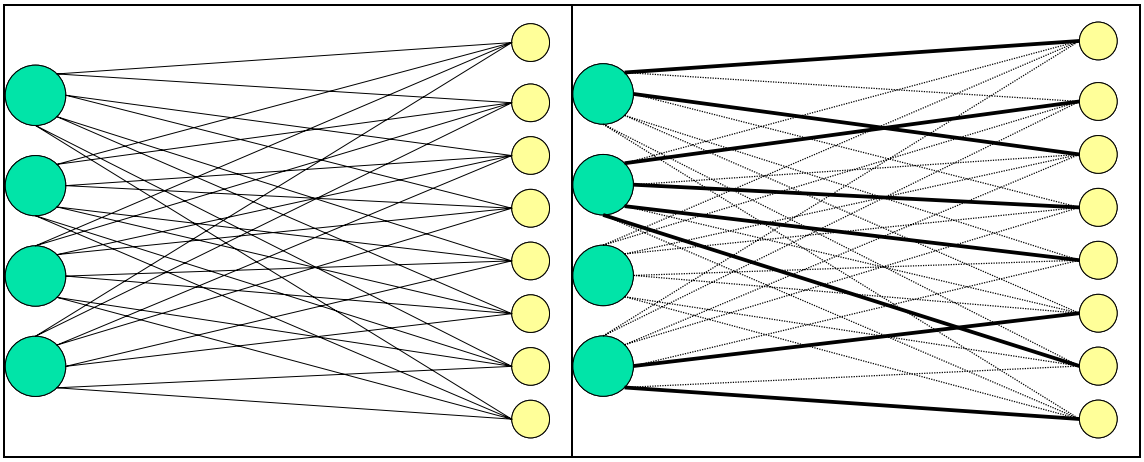

Figura 2. À esquerda (a) um grafo bipartido completo; à direita (b) uma solução típica do GAP (arestas em destaque pertencem à solução).

O grafo da Figura 2a auxilia na enumeração do número de possíveis soluções de uma instância GAP. Cada nó tarefa precisa ser designado a um nó agente, ou seja, é necessário escolher uma aresta para cada nó tarefa. Como o número de arestas para cada nó tarefa corresponde ao numero de agentes, tem-se que o número de possíveis solução par um GAP qualquer é m^n. No exemplo da Figura 2a (uma instância pequena comparada às da literatura), temos $m = 4$ e $n = 8$, logo $4^8 = 65536$ possíveis soluções. As instâncias utilizadas nesse trabalho vão de $m = 5$ e $n = 15$ ($\approx 3\text{x}10^{10}$ possíveis soluções) até $m = 10$ e $n = 60$ ($= 1\text{x}10^{60}$ possíveis soluções), ou seja, um espaço de soluções gigantesco, justificando a utilização de meta-heurísticas.

2.3. Modelo matemático

Apesar do GAP ser um problema NP-completo, é possível encontrar soluções ótimas para instâncias pequenas. Para isso, formula-se o GAP como um problema de programação linear inteira e utiliza-se um pacote computacional de otimização disponível no mercado (CPLEX, XPRESS, LINDO, dentre outros solvers). O modelo mais usual do GAP é apresentado a seguir.

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

s.a.

$$\sum_{i=1}^{N} a_{ij} x_{ij} \leq b_j \qquad \forall j \qquad 1 \leq j \leq M \qquad (2)$$

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

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

Onde:

x_{ij}: 1 se o trabalho i foi designado ao agente j, 0 caso contrário.

c_{ij}: custo associado à designação do agente *j* à tarefa *i*.

a_{ij}: recurso associado à designação do agente *j* à 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 $x_{ij} = 1$). Se um modelo de maximização for desejável, basta inverter o sinal dos custos. A restrição de capacidade (2) limita o montante de recursos que cada agente pode dispensar. A restrição de designação (3) garante que cada tarefa terá um único agente para atendê-la. Finalmente, a restrição binária (4), responsável por transformar o GAP em um problema combinatorial, limita dois valores para a variável x_{ij}, um ou zero.

3. ALGORITMO DE COLÔNIA DE FORMIGA

O algoritmo de colônia de formigas artificial ou *ant colony systems* (ACS) é inspirado pelo comportamento de colônias de formigas reais, em particular, pelo seu comportamento na procura de alimento. Uma das idéias centrais do ACS, proposto originalmente por Dorigo *et al.* (1991), é a comunicação indireta baseado em trilhas (caminhos ou trajetos) de feromônios entre uma colônia de agentes denominadas formigas (*ants*). As trilhas de feromônios são um tipo de informação numérica distribuída que é modificada pelas formigas para refletir sua experiência quando da resolução de um problema em particular (Stützle & Dorigo,1999).

3.1. Otimização por colônia de formigas (*ant systems*)

Cada criatura viva tem a missão de preservar sua espécie sob efeito dinâmico de uma variedade de mudanças ambientais. É conhecido que muitos grupos de criaturas estão aptas a resolução de problemas através de atividades exercidas de forma cooperativa. Neste contexto, um sistema distribuído pode melhorar seu desempenho geral através da interação entre diversos agentes autonômos. Um exemplo deste tipo de resolução de problema por cooperação é a utilização de uma meta-heurística denominada ACS (Kubo & Kakazu, 1993).

3.2. Inspiração biológica

As formigas reais são capazes de encontrar o caminho mais curto para uma fonte de alimento do formigueiro sem a utilização de dados visuais. Enquanto caminham, as formigas depositam no solo uma substância denominada de feromônio (designação genérica de substâncias secretadas pelas formigas que servem de meio de comunicação entre elas), e tem seu deslocamento baseado em trilhas de feromônios previamente depositados por outras formigas. Estas trilhas de feromônios pode ser observadas por outras formigas e motivá-las em seguir determinado caminho, isto é, um movimento aleatório das formigas segue com maior probabilidade uma trilha de feromônio. Esta é uma maneira de como as trilhas são reforçadas e mais e mais formigas tendem a seguir aquela trilha. Uma formiga trabalha da seguinte forma: Primeiro, quando as formigas chegam a um ponto de decisão em que elas tem que decidir mover-se à direita ou à esquerda, as formigas selecionam aleatoriamente o próximo caminho e depositam feromônio no solo, sem ter a noção de qual é a melhor escolha. Depois de um pequeno período de tempo a diferença entre a quantidade de feromônio entre dois caminhos é suficientemente grande que influencia a decisão de novas formigas estão no impasse da tomada de decisão por qual caminho seguir. Neste caso, as novas formigas escolhem o caminho com maior quantidade de feromônio. Consequentemente, as formigas podem cheirar o feromônio e escolher, com dada probabilidade, os caminhos marcados com concentrações mais acentuadas de feromônios. Em síntese, este princípio da natureza pode ser útil na configuração de ACS para resolução de problemas do tipo caixeiro viajante, escalonamento, roteamento e GAP.

3.3. Característica e potencialidades do ACS

O ACS é uma meta-heurística da inteligência coletiva (*swarm intelligence*) baseada em uma população de agentes (formigas) que possui as seguintes características:

- É um algoritmo não-determinístico baseado em mecanismos presentes na natureza, isto é, ele é baseado no comportamento de formigas para a determinação de caminhos através de suas colônias para procura eficiente de fontes de comida;
- É um algoritmo paralelo e adaptativo, pois uma população de agentes movê-se simultaneamente, de forma independente e sem um supervisor (não há um controle ou supervisão central);
- É um algoritmo cooperativo, pois cada agente (formiga) escolhe um caminho com base na informação (trilhas de feromônios) depositadas por outros agentes que tenham selecionado previamente o mesmo caminho. Este comportamento cooperativo tem ingredientes de autocátalise (catálise provocada por feromônios que se formam no próprio sistema reativo),

isto é, o ACS providencia uma realimentação positiva, desde que a probabilidade de um agente escolher o caminho aumenta com o número de agentes que escolheu previamente aquele caminho.

3.4. Colônia de Formigas aplicado ao GAP

Um procedimento de otimização baseado em colônia de formigas é uma meta-heurística baseada em uma população de agentes (formigas) que faz uso de mecanismos de adaptação, cooperação e paralelismo visando a obtenção de um procedimento para resolução de problemas de otimização combinatória. O GAP possui uma estrutura propícia para ser resolvido por meio do algoritmo de colônia de formigas.

Heurísticas baseadas em colônias de formigas para o GAP foram utilizadas por Lourenço e Serra(1998) e mais recentemente por Randall(2004). Lourenço e Serra(1998) propõe uma heurística usando Max-Min. Randall(2004), faz uma análise desta abordagem levando em conta a estrutura especial do problema.

A heurísticas baseadas em colônias de formigas se mostraram promissoras e competitivas com outras apresentadas na literatura para resolução, porém esta abordagem não é a melhor para a resolução do problema. Além disso, a aplicação a este problema é um relevante instrumento de estudo.

3.5. Implementação do GAP utilizando Colônia de Formigas (Randall, 2004)

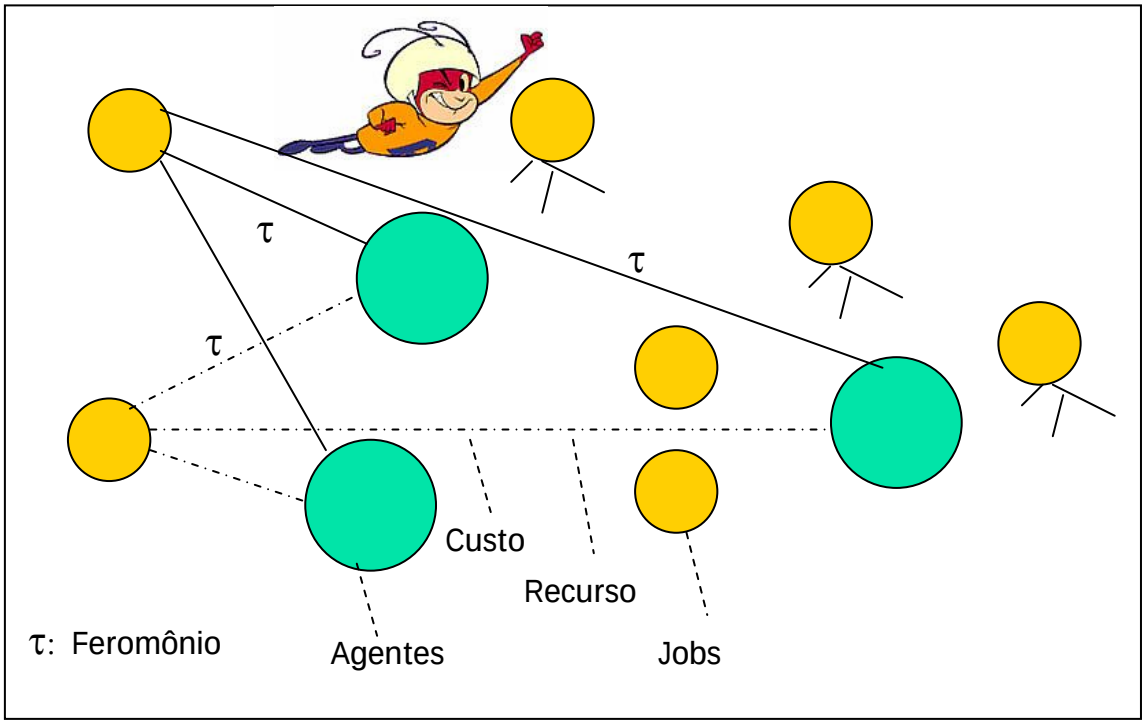

Figura 3: Diagrama do problema GAP.

Na Figura 3 todos os jobs estão conectados a todos os agentes, porém nem os jobs nem os agentes estão conectados entre si. Os feromônios estão nas arestas que ligam os jobs aos agentes. A própria definição do problema GAP faz com que cada job tenha um custo e consuma um recurso ao ser alocado a um determinado agente, sendo que o recurso total de cada agente é limitado.

```
Pseudo-código:
    Para cada iteração
        Para cada formiga
            job = escolheTrabalho();                          (p1)
            agente = escolheAgente(job);                      (p2)

            Se solução tornou-se infactível
                Reinicializa a formiga;                       (p3)
            Senão
                Decai feromônio da aresta job-agente escolhida;   (p4)
                Se todos os jobs foram alocados
                    Busca Local;                                  (p5)
                    Se a solução corrente é a melhor até agora
                        Decai a qtd de feromônio de todas as arestas;   (p6)
                            Aumenta a qtd de feromônio das
                                        arestas da solução corrente;    (p7)
                            Reinicializa a formiga;                     (p8)
```

Figura 4. Pseudo-código do ACO aplicado ao GAP.

A Figura 4 descreve o algoritmo sugerido por Randall (2004) cujas funções são descritas a seguir:

- p1 - escolheTrabalho( ): determina qual o job que vai ser alocado. O job que consome mais recursos é alocado primeiro, conforme a fórmula I.

$$job = máx\ (\ \Sigma_{agente}\ custo(agente, job\_não\_alocado)) \qquad (I)$$

- p2 - escolheAgente(job): escolhe o agente para alocar ao *job*.

Inicialmente sorteia-se um número q,

Se $q < q_0$ tem-se:

$$agente = \text{aresta que maximiza } \{ \tau(agente,job)\ [\eta(agente,job)]^{\beta} \} \qquad (II\text{-}a)$$

Caso contrário:

$$agente = \text{roleta com prob. } P(i,j) = \{\tau(i,j)\ [\eta(i,j)]^{\beta}\ /\ \{\Sigma\ \tau(i,j)\ [\eta(i,j)]^{\beta}\}\} \qquad (II\text{-}b)$$

Sendo:

i: agente que não faz com que a solução seja infactível

$\eta(i, j)$: custo **ou** recurso consumido ao alocar o job j ao agente i.

q_0: probabilidade de selecionar agente deterministicamente (ver **item 3.6**).

β: constante

Pela fórmula (II), quanto maior a quantidade de feromônio $\tau(i, j)$, maior a chance da aresta agente i – job j ser escolhida. E quanto menor $\eta(i, j)$, independente se η é custo ou recurso consumido do agente, maior a probabilidade da escolha da aresta.

Quando η é recurso consumido, a preocupação é em obter uma solução factível, mesmo com um custo mais alto. Fazendo η igual a custo objetiva a obtenção de uma solução com menor custo, assumindo que a factibilidade não é tão difícil para àquela instância do problema. A alternância entre as duas estratégias é comentada no item 3.6 (Medidas Heurísticas para obtenção de soluções factíveis).

Observe que a quantidade de feromônio $\tau(i, j)$ já é uma medida de factibilidade da solução e do seu custo. E o uso $[\eta(i,j)]^{\beta}$ vem reforçar a meta-heurística usando a estrutura do problema.

- p3 - reinicialização

Se durante a execução do algoritmo, uma formiga não consegue uma solução factível, ela é reinicializada. Devido isto, observa-se que no formigueiro existem formigas de idades diferentes, sendo que algumas formigas têm poucos agentes alocados aos jobs enquanto outras formigas estão prestes a completar a solução.

- p4 – decaimento da quantidade de feromônio

Ao alocar um agente a um job, diminui-se a quantidade de feromônio associada àquela aresta para diminuir a probabilidade da próxima formiga alocar o mesmo agente ao mesmo job, conseguindo-se, assim, uma maior diversidade das soluções.

Quando todos os jobs foram alocados, executa-se uma busca local **(p5)** para melhorar o custo total da solução encontrada (explicada no **item 3.7**), decai o feromônio de todas as arestas **(p6)** e aumenta a quantidade de feromônio das arestas pertencentes a solução encontrada **(p7)**, para aumentar a chance das demais formigas escolherem as mesmas arestas da melhor solução encontrada até o momento. Depois a formiga é reinicializada **(p8)** pois o papel daquela formiga já foi cumprido.

3.6. Medidas Heurísticas para obtenção de soluções factíveis

Alguns problemas de otimização como o GAP necessitam de especial atenção pelo fato de possuírem restrições, no caso, a capacidade dos agentes. Por isso, durante o processo de busca por soluções deve-se levar em conta tanto às restrições de capacidade quanto o custo.

A equação que governa as escolhas durante a construção de soluções no ACO tem com componente a medida heurística, em vários problemas esta medida está diretamente relacionada ao custo, porém deve-se ater ao fato das peculiaridades de cada problema para que a decisão da medida heurística seja favorável.

No caso do GAP, verificamos que a escolha do termo heurístico baseado apenas no custo gerava poucas soluções factíveis e em alguns casos nenhuma solução factível foi encontrada.

Por outro lado, utilizando-se como medida heurística as restrições de capacidades dos agentes foi possível gerar soluções factíveis, embora de baixa qualidade.

Diante disso, usamos uma estratégia adaptativa que alternava o uso dos termos heurísticos baseados no custo e no recurso. Para cada solução factível encontrada, aumentava a probabilidade de fazer η igual ao custo. De maneira análoga, para cada solução infactível encontrada, diminuía essa probabilidade.

3.7. Busca Local

As soluções produzidas pelo ACO são melhoradas utilizando-se procedimentos de busca local, cujos operadores pesquisam na vizinhança da solução corrente por uma outra solução de melhor qualidade. Se esta for encontrada, realiza-se a operação e a busca continua sobre essa nova solução, com o processo se repetindo até que não seja possível melhorar a solução corrente.

Para o problema GAP encontramos dois operadores de busca local que tem se mostrados promissores ao problema. Estes operadores são descritos a seguir.

Swap: Escolha duas tarefas, os agentes que estão alocados a atendê-las são trocados.

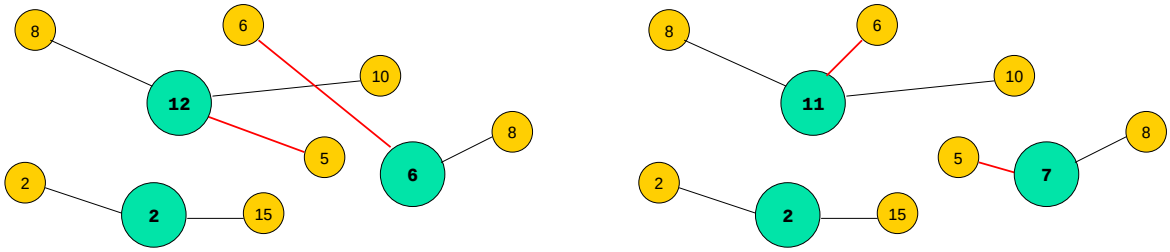

Figura 5. Busca local “Swap” entre duas tarefas.

Change: Escolha uma tarefa e troque o seu agente por um outro.

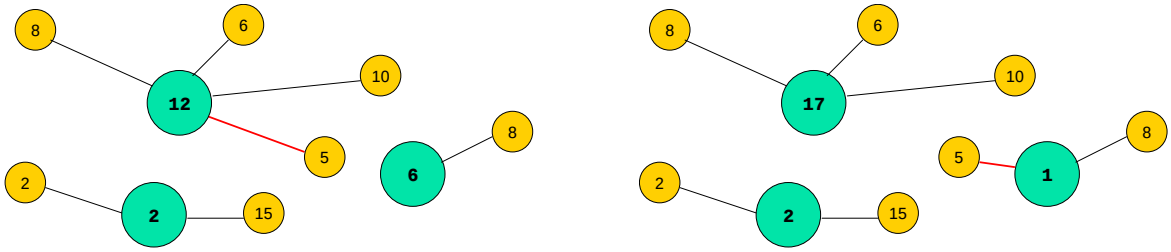

Figura 6. Busca local “Change” sobre uma tarefa.

A busca local inicia depois que o algoritmo ACO gera uma solução factível, da seguinte forma:

- Operador Change: verifica todos os possíveis movimentos e armazena o melhor;
- Operador Swap: verifica todos os possíveis movimentos e armazena o melhor;
- Verifica se a solução corrente é inferior a uma das dessas duas soluções. Em caso afirmativo, realiza-se a troca pela melhor solução, caso contrário, as soluções da busca local são descartadas.

3.8. Testes realizados

Para avaliar o desempenho do ACO foram utilizadas 60 instâncias do GAP disponíveis na base de dados OR-Library (Operational Research Library - http://people.brunel.ac.uk/~mastjjb/jeb/orlib/gapinfo.html). Essas instâncias foram criadas no intuito de serem maximizadas.

Os seguintes valores para os parâmetros foram utilizados:

- Probabilidade para seleção determinística do agente $q_0 = 0,9$
- Taxa de decaimento de feromônio: $\rho = 0,9$
- Expoente de reforço do termo heurístico: $\beta = -2$
- Número de formigas: $m = 10$
- Número de iterações: 10.000
- Inicialização da matriz de feromônio: $\varphi(i,j) = -c(i,j)$ (maximização)
- Termo heurístico 1: $\eta_1(i,j) = c(i,j)$ (melhora a função objetivo)

- Termo heurístico 2: $\eta_2(i,j) = a(i,j)$ (melhora a factibilidade)
- Probabilidade inicial de seleção do termo heurístico 1: $p = 0,1$
- Gradiente da probabilidade de seleção do termo heurístico 1: $\Delta p = 0,02$ (adaptativo)

3.9. Resultados Experimentais

A seguir na Tabela 1, são apresentados os resultados da aplicação do ACO para as instâncias gap1 ao gap12 (60 instâncias ao todo). A tabela apresenta as soluções ótimas das instâncias, informação extraída da literatura e obtida por técnicas de otimização linear inteira ("branch and bound"). Também são apresentadas as melhores soluções (maior custo) encontradas em cada uma das dez repetições efetuadas por instância. Por fim, são mostradas as médias das melhores soluções e os desvios percentuais dessas médias com relação à solução ótima. Uma célula da tabela preenchida com a letra "o" implica que o algoritmo obteve a solução de valor máximo (ótimo) para essa instância.

Pode-se afirmar que, em geral, o algoritmo obteve um desempenho muito bom. De uma amostra de 600 repetições o algoritmo atingiu a solução ótima do problema em 384, o que representa 64%. O maior desvio percentual com relação à solução ótima ocorreu no gap8-1, com 0,432%, um valor relativamente pequeno. O número de instâncias onde o algoritmo atingiu a solução ótima em todas as repetições foi 32 de 60, ou seja, aproximadamente 53,3%.

Considerando as instâncias com até 40 tarefas, o desempenho do algoritmo ACO foi ainda mais surpreendente, onde foi possível contabilizar 326 soluções ótimas de um total de 350, representando 93,1% de ótimos.

Já para as instâncias com mais de 40 tarefas, o desempenho já foi um pouco inferior, porém apresentando soluções de qualidade aceitável. Foram encontradas 58 soluções ótimas de 250 repetições, o que representa 23,2%. Esse número não é desprezível, pois deve-se levar em conta a grandeza do espaço de busca, que é gigantesco para essas instâncias.

A questão da aleatoriedade dos procedimentos de escolha dos agentes ou do termo heurístico utilizado não exerceu forte influência na qualidade das soluções, pois percebe-se que as dez repetições possuem um mesmo padrão característico.

O tempo de processamento não foi exposto, pois se trata de uma informação de pouco validade devido às diferenças na implementação, compilação e hardware. No entanto, pode-se afirmar que, para um PC com processamento de 2GHz, o tempo estimado para efetuar um teste completo para todas as 60 instâncias é de 5 minutos, sendo que esse tempo não é igualmente distribuído, pois as instâncias maiores levam mais tempo que as menores para serem processadas.

Tabela 1. Resultados da aplicação do ACO ao GAP.

Dados			Melhores Soluções Encontradas em 10 Testes										Estatísticas	
Instância	**Classe**	**Sol Otima[1]**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**Média**	**Desvio (%)**
gap1-1	c515	336	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	336.0	0.000
gap1-2	c515	327	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	327.0	0.000
gap1-3	c515	339	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	339.0	0.000
gap1-4	c515	341	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	341.0	0.000
gap1-5	c515	326	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	326.0	0.000
gap2-1	c520	434	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	434.0	0.000
gap2-2	c520	436	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	436.0	0.000
gap2-3	c520	420	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	420.0	0.000
gap2-4	c520	419	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	419.0	0.000
gap2-5	c520	428	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	428.0	0.000
gap3-1	c525	580	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	580.0	0.000
gap3-2	c525	564	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	564.0	0.000
gap3-3	c525	573	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	573.0	0.000
gap3-4	c525	570	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	570.0	0.000
gap3-5	c525	564	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	564.0	0.000
gap4-1	c530	656	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	656.0	0.000
gap4-2	c530	644	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	644.0	0.000
gap4-3	c530	673	**o**	**o**	**o**	**o**	**o**	671	**o**	**o**	**o**	**o**	672.8	0.030
gap4-4	c530	647	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	647.0	0.000
gap4-5	c530	664	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	664.0	0.000
gap5-1	c824	563	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	563.0	0.000
gap5-2	c824	558	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	558.0	0.000
gap5-3	c824	564	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	564.0	0.000
gap5-4	c824	568	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	568.0	0.000
gap5-5	c824	559	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	559.0	0.000
gap6-1	c832	761	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	761.0	0.000
gap6-2	c832	759	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	759.0	0.000
gap6-3	c832	758	757	**o**	**o**	757	**o**	**o**	**o**	757	**o**	757	757.6	0.053
gap6-4	c832	752	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	752.0	0.000
gap6-5	c832	747	746	**o**	**o**	**o**	**o**	**o**	**o**	746	**o**	**o**	746.8	0.027
gap7-1	c840	942	941	941	941	941	**o**	941	**o**	**o**	941	941	941.3	0.074
gap7-2	c840	949	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	949.0	0.000
gap7-3	c840	968	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	968.0	0.000
gap7-4	c840	945	**o**	939	939	939	942	**o**	941	939	940	941	941.0	0.423
gap7-5	c840	951	**o**	**o**	**o**	950	**o**	950	**o**	**o**	**o**	**o**	950.8	0.021
gap8-1	c848	1133	1129	1128	1130	1127	1129	1129	1126	1126	1128	1129	1128.1	0.432
gap8-2	c848	1134	1130	1131	1130	1130	1133	1133	1128	1132	1128	1130	1130.5	0.309
gap8-3	c848	1141	1138	1137	1139	1137	1139	1140	1137	1137	1140	1139	1138.3	0.237
gap8-4	c848	1117	1113	1113	1113	1112	1114	1112	1114	1112	1112	1112	1112.7	0.385
gap8-5	c848	1127	1123	1123	1123	1122	1122	1124	1123	1123	1124	1122	1122.9	0.364
gap9-1	c1030	709	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	709.0	0.000
gap9-2	c1030	717	715	715	715	715	715	715	715	**o**	715	715	715.2	0.251
gap9-3	c1030	712	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	712.0	0.000
gap9-4	c1030	723	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	722	**o**	722.9	0.014
gap9-5	c1030	706	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	**o**	706.0	0.000
gap10-1	c1040	958	**o**	956	956	956	956	956	955	957	957	**o**	956.5	0.157
gap10-2	c1040	963	962	959	959	960	960	**o**	960	959	960	962	960.4	0.270
gap10-3	c1040	960	958	958	958	957	957	956	958	957	958	956	957.3	0.281
gap10-4	c1040	947	945	945	944	**o**	944	944	944	945	945	944	944.7	0.243
gap10-5	c1040	947	945	944	944	945	944	944	943	943	945	944	944.1	0.306
gap11-1	c1050	1139	1137	1136	1138	1136	1136	1137	1136	1138	1137	1136	1136.7	0.202
gap11-2	c1050	1178	**o**	1176	1177	**o**	1176	1175	**o**	1176	1176	1177	1176.7	0.110
gap11-3	c1050	1195	**o**	1194	**o**	1194	**o**	**o**	**o**	**o**	**o**	**o**	1194.8	0.017
gap11-4	c1050	1171	1168	1168	**o**	1170	1170	**o**	1167	1169	1168	1169	1169.1	0.162
gap11-5	c1050	1171	1168	1169	1168	1170	1170	1169	1170	1168	1168	1168	1168.8	0.188
gap12-1	c1060	1451	1449	1448	1448	1448	1449	1450	1449	1449	1448	1448	1448.6	0.165
gap12-2	c1060	1449	1448	1446	1446	1447	**o**	1445	1445	1448	1447	1446	1446.7	0.159
gap12-3	c1060	1433	1429	1431	1430	1431	1431	1432	1430	1430	1431	1432	1430.7	0.161
gap12-4	c1060	1447	1443	1442	1444	1443	1443	1444	1446	1444	1444	1443	1443.6	0.235
gap12-5	c1060	1446	1444	1443	1444	1442	1443	1442	1443	1444	1444	1443	1443.2	0.194

3.10. Termo Heurístico Adaptativo

Como foi abordado anteriormente, foi utilizado um critério adaptativo para adequar o termo heurístico com o histórico de obtenção de soluções do algoritmo, ou seja, caso o ACO esteja encontrando muitas soluções factíveis, a probabilidade de escolher o termo heurístico baseado no custo aumenta, caso contrário, diminui em contrapartida com o termo heurístico baseado no recurso, que por sua vez aumenta.

Para ilustrar o funcionamento desse critério adaptativo, são exibidas as Figuras 7 e 8, que mostram a variação da probabilidade de selecionar o termo heurístico baseado no custo (***p***), o custo da melhor solução e das soluções atuais ao longo das iterações do algoritmo para a instância gap12-1. Essa instância foi selecionada por pertencer à classe mais difícil dentre todo o conjunto.

Na Figura 7a pode-se perceber que foi possível encontrar soluções factíveis logo no início do algoritmo, o que acarretou no crescimento linear de *p* até estabilizar-se dinamicamente na faixa de 0,6. Na Figura 7b é possível notar que o ACO consegue encontrar soluções de custos bastante distintos (linha contínua). A melhor solução (linha pontilhada) rapidamente convergiu para 1447. Vale notar que o registro da melhor solução inclui a melhoria proveniente da busca local, enquanto que os registros das soluções atuais não.

Na Figura 8a o valor de *p* foi inicializado em 0,9, priorizando mais o aspecto da função objetivo do que a restrição de capacidade. Ainda assim, o ACO conseguiu também encontrar rapidamente uma solução factível. Novamente o *p* ficou flutuando ao redor do valor 0,6, o que sugere que esse é um valor adequado para *p*, nas implementações em que o termo heurístico é adotado de maneira não adaptativa.

Pela Figura 8b não foi possível notar diferenças significativas em inicializar *p* em 0,1 ou 0,9. Desse modo, sugere-se apenas que o algoritmo seja executado um número suficiente de iterações para garantir a estabilização de *p*. Nos exemplos mostrados, isso ocorreu antes da iteração 1500. Ainda assim, nos testes computacionais foram executadas 10000 iterações devido ao tempo computacional reduzido que permite explorar melhor o espaço de soluções.

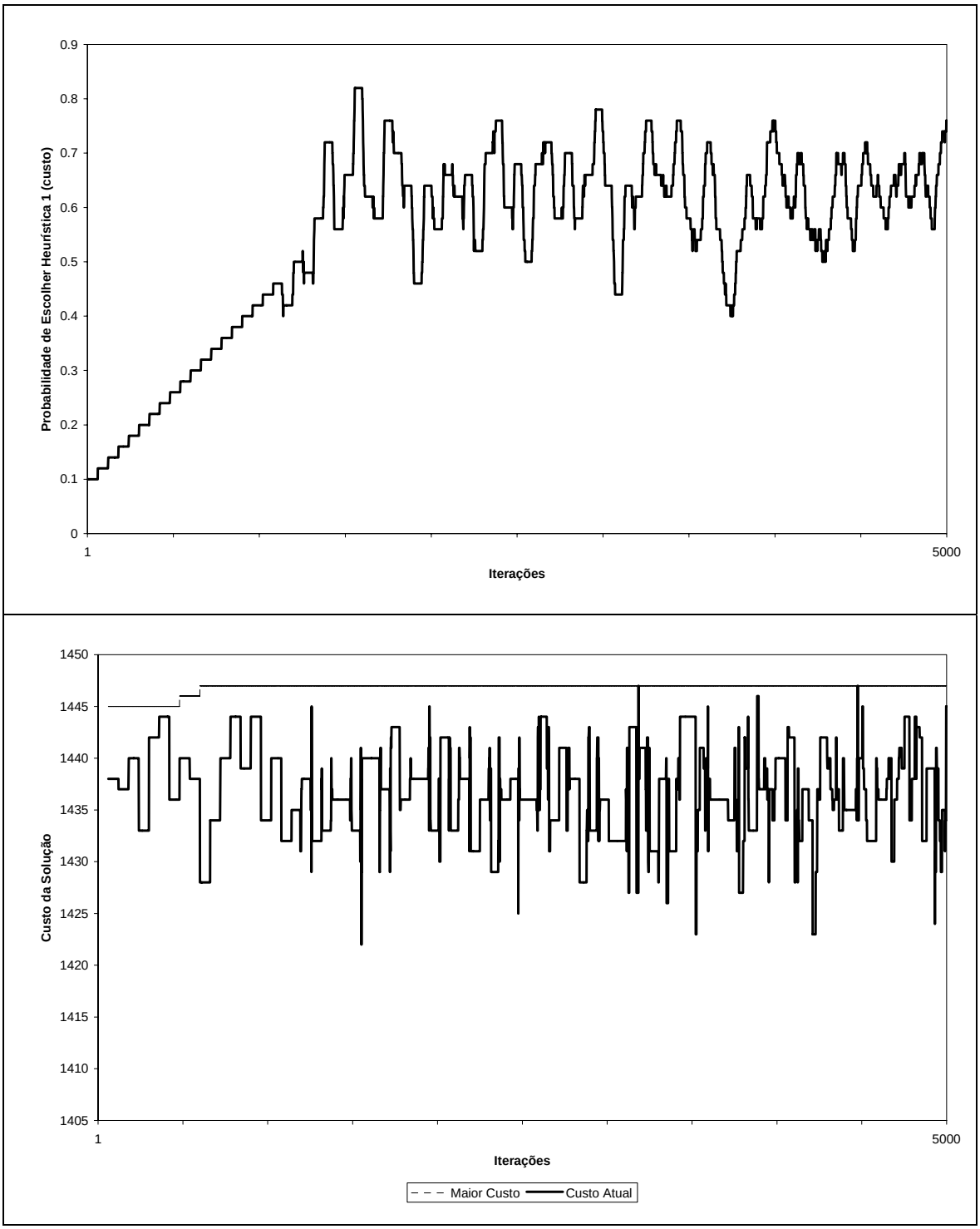

Figura 7. (a) Variação de p (acima), inicializado em 0,1, e (b) custos da solução atual e melhor solução (abaixo) ao longo das iterações.

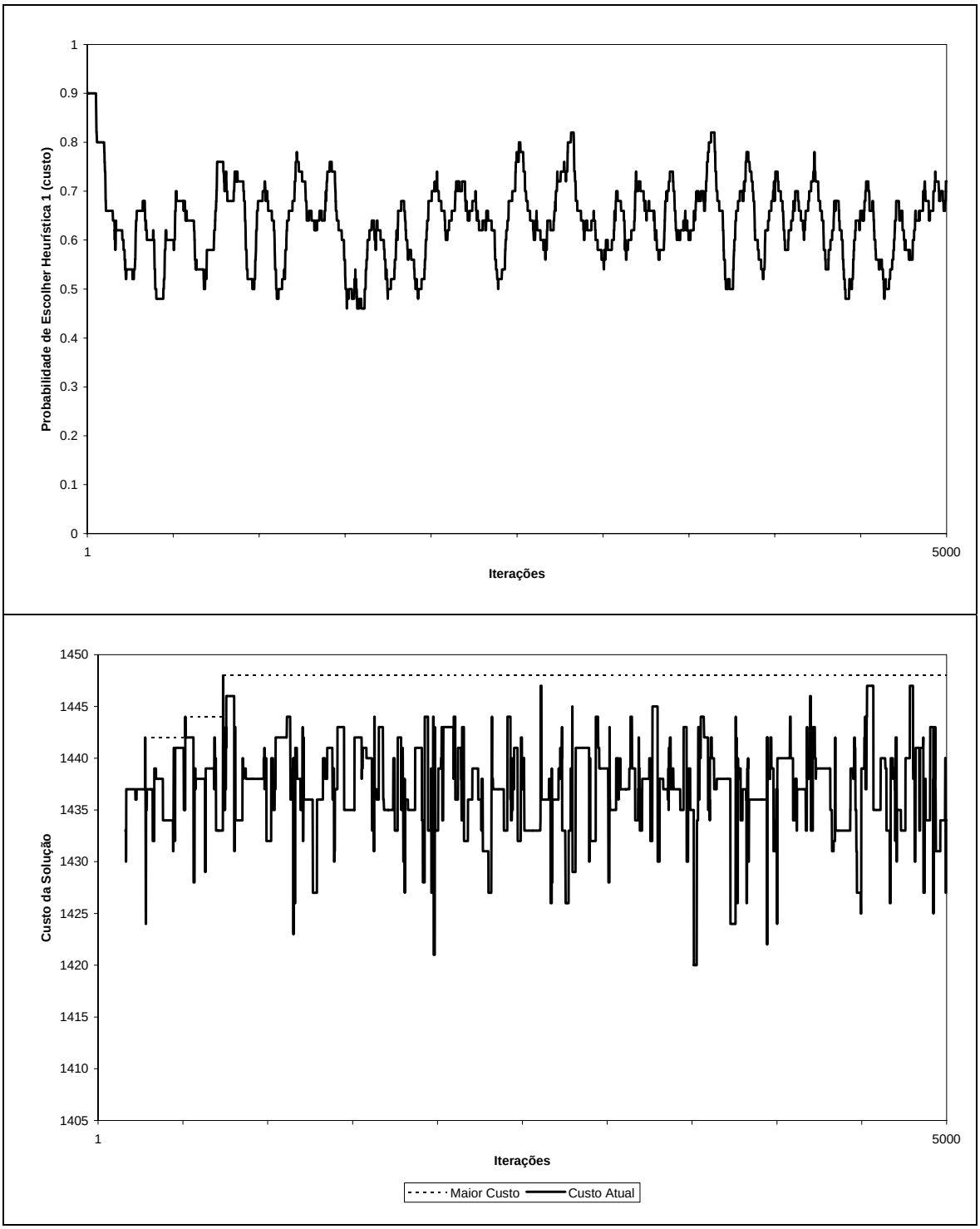

Figura 8. (a) Variação de p (acima), inicializado em 0,9, e (b) custos da solução atual e melhor solução (abaixo) ao longo das iterações.

4. SISTEMA IMUNOLÓGICO ARTIFICIAL

4.1. Introdução

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-organizados. 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.*, 2000a,b). Além dessas, outras linhas de pesquisa ainda mais recentes têm surgido como novos paradigmas de computação.

Os sistemas imunológicos artificiais (SIA) (Dasgupta, 1998a), que surgiram a partir de tentativas de modelar e aplicar princípios imunológicos no desenvolvimento de novas ferramentas computacionais, já vêm sendo utilizados em diversas áreas, como reconhecimento de padrões, detecção de falhas e anomalias, segurança computacional, otimização, controle, robótica, *scheduling*, análise de dados, aprendizagem de máquina, dentre outras, como pode ser encontrado em Dasgupta, (1998a,b), Bäck *et al.*, (2000a,b), Timmis (2000) e em De Castro (2001).

Nas áreas de engenharia e computação, tem surgido um forte interesse pelo estudo dos sistemas imunológicos devido, principalmente, à sua capacidade de processamento de informação. Sob uma perspectiva de engenharia, existem diversas características do sistema imunológico (SI) que podem ser destacadas:

- Unicidade: cada animal possui seu próprio sistema imunológico, com suas capacidades e vulnerabilidades particulares;

- Reconhecimento de padrões internos e externos ao sistema: as células e moléculas que não pertencem ao organismo são reconhecidas e eliminadas pelo SI;

- Detecção de anomalia: o SI pode detectar e reagir a agentes patogênicos (causadores de anomalias) a que o organismo nunca havia sido exposto anteriormente;

- Detecção imperfeita (tolerância a ruídos): um reconhecimento perfeito não é necessário para que o SI reaja contra um elemento causador de patologia (patógeno);

- Diversidade: existe uma quantidade limitada de células e moléculas no SI que são utilizadas para se obter o reconhecimento de um número praticamente infinito de elementos, incluindo aqueles sintetizados em laboratório;

- Aprendizagem por reforço: a cada encontro com o mesmo patógeno, o sistema imunológico melhora a qualidade de sua resposta; e

- Memória: os componentes do SI bem sucedidos no reconhecimento e combate às patologias são armazenados para uma resposta futura mais intensa e efetiva.

4.2. Definição

Segundo Dasgupta (1998a), os sistemas imunológicos artificiais são mecanismos computacionais compostos por metodologias inteligentes, inspiradas no sistema imunológico biológico, para a solução de problemas do mundo real.

4.3. Princípios Fundamentais do Sistema Imunológico

Para facilitar a compreensão da estrutura do sistema imunológico artificial serão apresentados, de forma sucinta, o funcionamento do sistema imunológico nos animais vertebrados e os princípios da seleção clonal. Informações referentes aos elementos que constituem o sistema,

bem como definições sobre os termos biológicos utilizados podem ser encontrados nas referências: De Castro (2001), Dreher (1995), Janeway *et al*. (2000) e Timmis (2000).

4.3.1. Composição e Funcionamento do Sistema Imunológico

Quando um sistema (corpo) detecta a presença de um antígeno (patógeno), ocorrem duas formas de respostas imunológicas dadas pelos seguintes sistemas (Figura 9):

- Sistema Imune Inato: é a primeira linha de defesa do organismo. Ele é formado por células fagocitárias, como macrófagos e os neutrófilos, além de fatores solúveis como o complemento e algumas enzimas. As células do sistema imune inato desempenham um papel crucial na iniciação e posterior direcionamento das respostas imunes adaptativas, principalmente devido ao fato de que as respostas adaptativas demoram período de tempo (da ordem de dias) para exercer seus efeitos. Portanto, a resposta inata apresenta um papel muito importante no controle das infecções durante esse tempo;

- Sistema Imune Adaptativo: os linfócitos (produzidos na medula óssea e no timo) são as principais células que compõe esse sistema, presentes apenas nos animais vertebrados, eles desempenham um papel crucial no desencadeamento e posterior regulação das respostas imunes inatas. Cada "linfócito virgem" que penetra na corrente circulatória é portador de receptores de um antígeno com uma única especificidade. Os linfócitos sofrem, então, um processo parecido com a seleção natural durante a vida do indivíduo: somente aqueles que encontram um antígeno com o qual seu receptor pode interagir serão ativados para proliferar e se diferenciar em células efetoras. Após a ligação do anticorpo de superfície ao antígeno, a célula é ativada para proliferar e produzir uma numerosa prole, conhecida como clone. Essas células secretam anticorpos com uma especificidade idêntica à do receptor de superfície. Esse princípio recebeu o nome de teoria da seleção clonal, e constitui a parte central da imunidade adaptativa.

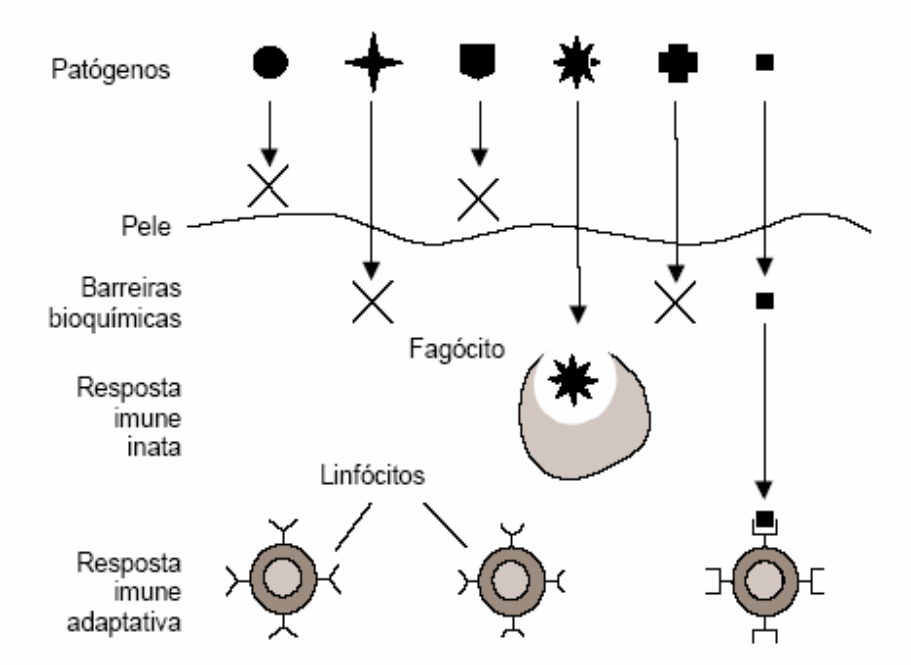

Figura 9. Estrutura multicamadas do sistema imunológico.

4.4. Princípios da Seleção Clonal

Uma vez que cada célula apresenta um padrão (forma) distinto de receptor antigênico, o número de linfócitos que pode se ligar a um determinado antígeno é restrito. A fim de produzir células

efetoras específicas em quantidade suficiente para combater uma infecção, um linfócito ativado deve se proliferar antes que sua prole se diferencie em células efetoras.

O princípio da seleção clonal estabelece que apenas aquela célula capaz de reconhecer um determinado estímulo antigênico irá se proliferar, sendo, portanto, selecionada em detrimento das outras.

Quando um animal é exposto a um antígeno, uma subpopulação de linfócitos responde através da produção de anticorpos. Cada célula secreta um único tipo de anticorpo, que é relativamente específico para o antígeno. Através da ligação do antígeno com o receptor do linfócito e, dado um segundo sinal (ou sinal co-estimulatório) de células acessórias como a célula T_H, um antígeno estimula o linfócito a se proliferar (dividir) e transformar-se em uma célula terminal capaz de secretar anticorpos em altas taxas. Estas células são chamadas de plasmócitos.

Os linfócitos, além de se proliferar e diferenciar em plasmócitos, também podem se diferenciar em linfócitos de memória, caracterizadas por longos períodos de vida. Esses linfócitos de memória provavelmente não produzem anticorpos, mas quando re-expostas ao mesmo estímulo antigênico começam a se diferenciar em plasmócitos capazes de produzir anticorpos pré-selecionados pelo antígeno específico que estimulou a resposta primária.

4.5. Implementação do Sistema Imunológico Artificial (SIA)

Considerando o problema do *GAP* como um problema de otimização em grafos, uma possível solução factível corresponde a um conjunto de nós N_1 (tarefas) conectados ao conjunto de nós N_2 (agentes), sendo que as seguintes restrições devem ser satisfeitas:

- Cada nó de N_1 deve estar conectado a exatamente um nó de N_2;
- A capacidade associada a cada nó de N_2 não pode ser menor que a somatória dos recursos consumidos pelos nós de N_1 conectados a ele.

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

4.5.1. A Estrutura e o Algoritmo

Considere os seguintes dados do problema:

- *nagentes* = número de agentes disponíveis para a execução das tarefas;
- *ntarefas* = número de tarefas a serem realizadas;
- $capacidade_j$ = capacidade total do agente *j*;
- $recurso_{ij}$ = quantidade de recurso consumido do agente *j* para a realização da tarefa *i*;
- $custo_{ij}$ = custo para a realização da tarefa *i* quando realizada pelo agente *j*;

Para facilitar a compreensão do algoritmo, considere o diagrama de blocos da Figura 10.

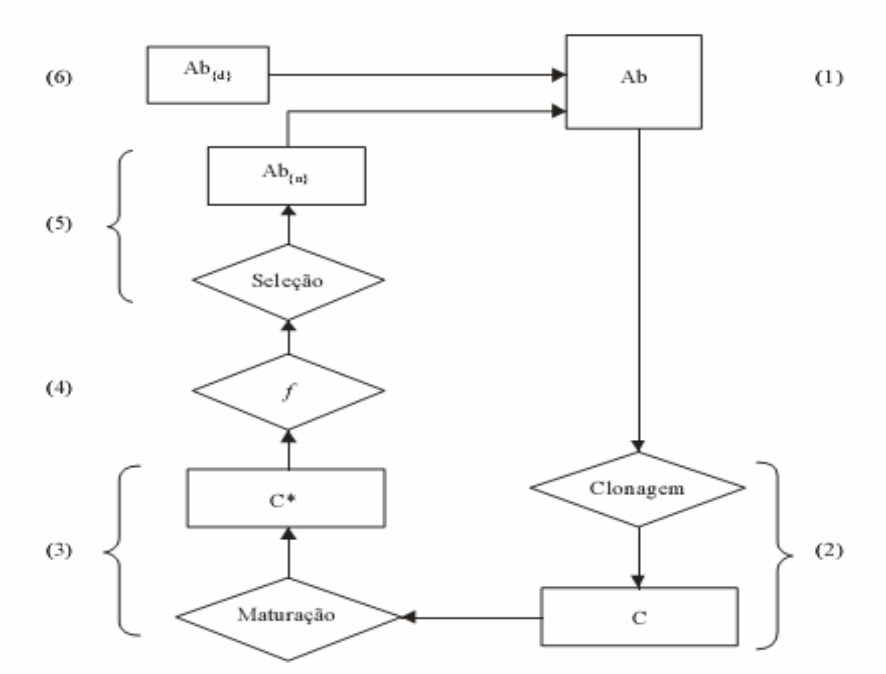

Figura 10. Diagrama de blocos do SIA

Representação do Anticorpo (Ab*i*): cada anticorpo é representado por um vetor-linha com *ntarefas* posições. Cada índice do vetor é associado a uma tarefa e o valor associado a cada posição do vetor corresponde ao agente que realizará tal tarefa. A Figura 11 mostra um exemplo de um anticorpo representando uma solução para o problema do *GAP* com 10 tarefas e 4 agentes. Pelo vetor-solução apresentado, as tarefas 4 e 5 foram alocadas ao agente 1, as tarefas 3, 6 e 10 ao agente 2, tarefas 1, 8 e 9 ao agente 3 e as 2 e 7 alocadas ao agente 4.

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

Figura 11. Exemplo de um anticorpo representando uma solução

Três métodos distintos para geração de anticorpos foram implementados:

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 evitar a geração de soluções infactíveis;
3. **Factível** – somente anticorpos representando soluções factíveis são gerados. A estratégia utilizada para montar uma solução factível foi a seguinte:
   1º Para cada tarefa:
   - Escolher os *k* menores consumos de recursos. Sendo *k* proporcional ao número de agentes disponíveis.
   - Escolher aleatoriamente um consumo dos *k* melhores e alocar a tarefa ao agente que corresponde a este consumo.

   2º Repetir este processo, até que todos os agentes tenham no mínimo uma tarefa.

Seja *p* a quantidade de anticorpos que constituí o repertório (**Ab)**, teremos uma matriz *p* x *ntarefas* que representará tal repertório.

1) Inicialização: seja *p* o tamanho do repertório desejado, a sua inicialização é feita por um dos três métodos de geração de anticorpos (aleatório, roleta, factível), conforme apresentado, a critério do usuário.

2) Clonagem: a clonagem do repertório é feita gerando-se *n* clones idênticos de cada anticorpo. Esses anticorpos clonados irão formar a população **C** de clones.

3) Maturação: a população **C** de clones sofre um processo de maturação de afinidade (mutação) gerando uma nova população **C*** de clones maturados. Essa mutação ocorre sorteando-se aleatoriamente *q* posições (tarefas) do vetor anticorpo e substituindo o conteúdo dos bits (agentes) por outros escolhidos aleatoriamente. A quantidade de bits escolhidos para maturação é inversamente proporcional à afinidade do anticorpo. Isto implica que quanto maior for a afinidade do anticorpo (melhor solução) menos bits serão maturados.

4) Medida de Afinidade (*fitness*): para medir a afinidade utilizamos a função objetivo, mesma formulação de Chu & Beasley (1997). No caso do problema de minimização, a afinidade de uma solução será o inverso multiplicativo do valor da função objetivo para esta solução. No problema de maximização o valor da afinidade de uma solução será o próprio valor da função objetivo.

5) Medida de Desafinidade (*unfitness*): para medir a desafinidade também seguimos a mesma formulação de Chu & Beasley (1997). A desafinidade de uma solução é calculada como a somatória das unidades excedidas de capacidade de cada agente, ou seja, a desafinidade é o quanto uma solução é infactível. Assim, se todos os agentes não possuem suas capacidade excedidas a desafinidade será nula.

6) Seleção: o processo de seleção dos clones é inspirado nos princípios da seleção clonal. Portanto, o método utilizado é elitista (Michalewicz, 1996), pois privilegia o clone com maior grau de afinidade. Assim, o clone com menor desafinidade e maior afinidade é selecionado entre os indivíduos do repertório de clones e comparado com o seu anticorpo-pai, sendo que apenas o melhor indivíduo será preservado na população de anticorpos.

7) 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 definida uma medida:

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 todas as posições do vetor. Outras métricas de similaridade foram testadas, por exemplo, uma porcentagem dos bits que 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 um único bit numa solução altera muito a sua qualidade, podendo transformar uma boa solução factível em uma solução infactível.

Uma possível injeção de diversidade pode ocorrer quando a diversidade do repertório ficar abaixo de um limiar. Neste caso, o repertório passa por um processo de supressão que retira uma parcela de anticorpos que representam soluções iguais e completa-o com novos indivíduos.

8) Busca Local: a utilização da busca local auxilia na redução da desafinidade (quando existir) e na melhoria da afinidade de um anticorpo obtido por seleção clonal. A busca local foi implementada seguindo a mesma formulação de Chu & Beasley (1997). Ela esta dividida em duas partes:

1º Parte - *Diminuir a Desafinidade:*

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

$$\sum_{j \in J,\, T_j = i} r_{ij} > b_i,$$

escolhe-se aleatoriamente uma tarefa q com $T_q = i$ e a designa para o próximo agente (na ordem de $i + 1,..., m, 1,..., i - 1$), verificando a disponibilidade do próximo agente.

2º Parte - *Aumentar a Afinidade:*

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

$$c_{ij} < c_{T_j\, j} \quad e \quad \left(\sum_{q\in J\,,T_q=i} r_{iq} \right) + r_{ij} \le b_i$$

Se um determinado *i* pode ser localizado, designar a tarefa *j* do agente *Tj* para o agente *i.*
Estes procedimentos devem ser 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.

4.5.2. Pseudocódigo

A implementação computacional do SIA pode ser feita seguindo o pseudocódigo descrito no Figura 11.

```
Ab    ← inicializar o repertório de anticorpos com tamanho p;
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,nclones);
          senão
            C   ← criar_anticorpos(nclones);
        fim-se;
            f   ← afinidade(C);
            C*  ← maturar(C,f);
            f   ← afinidade(C*);
            d   ← desafinidade(C*);
          Ab_{n} ← selecionar(C*,f,d);
          Ab_{n} ← busca_local(Ab_{n});
            Ab_i ← selecionar (Ab_{n},Ab_i);
    fim-para;
    Atualizar f e d;
    se diversidade(Ab) < l então
        n       ← Supressão(Ab) //aplica supressão e retorna o
número de soluções extraídas;
        Ab_{d}  ← criar_anticorpos(n);
        Ab      ← Ab ∪ Ab_{d}
    fim_se;
    geração ← geração + 1;
fim_enquanto;
```

Figura 11. Pseudocódigo do Sistema Imunológico Artificial

4.6. Resultados Experimentais

Para avaliar o desempenho do SIA foram utilizadas 60 instâncias do problema do GAP disponíveis na base de dados OR-Library (*Operational Research Library* - http://people.brunel.ac.uk/~mastjjb/jeb/orlib/gapinfo.html). Esta base, trata-se de um repositório público de instâncias para uma grande quantidade de problemas de otimização.

Os seguintes parâmetros foram utilizados para realizar os experimentos:

- O tamanho da população e o número de gerações variaram conforme a complexidade do problema;
- Quantidade de rodadas por experimento: 10;
- Número de clones gerados por anticorpo: 10;
- Limiar de diversidade para aplicação de supressão: 70%;
- Busca Local habilitada;
- Repertório Inicial gerado pelo método da Roleta;
- A maturação do clone é proporcional a sua afinidade. Seja $\mathbf{Af_m}$ a afinidade do melhor anticorpo do repertório e $\mathbf{Af_c}$ a afinidade do clone a ser maturado, a quantidade de bits do clone que será alterada é igual a:
    - 2, se $\mathbf{Af_c} \geq 95\%\ \mathbf{Af_m}$;
    - 3, se $\mathbf{Af_c} \geq 90\%\ \mathbf{Af_m}$;
    - 4, se $\mathbf{Af_c} \geq 80\%\ \mathbf{Af_m}$;
    - 5, se $\mathbf{Af_c} \geq 70\%\ \mathbf{Af_m}$;
    - 6, se $\mathbf{Af_c} < 70\%\ \mathbf{Af_m}$;

Esses parâmetros foram calibrados empiricamente de forma a obter o melhor desempenho possível.

Quanto à implementação dos algoritmos e à realização dos testes foi utilizada a ferramenta MatLab7® em um microcomputador Atlhon® XP 2200 – 2.0 GHz e 512 MB de memória.

Todas as instâncias testadas são problemas de maximização e, portanto, a medida de afinidade utilizada corresponde ao valor da própria função objetivo.

A Tabela 2 apresenta os melhores resultados obtidos para as 60 instâncias testadas.

Tabela 2. Resultados computacionais obtidos pelo SIA.

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**	**646**	**646**	646,0	0,00
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**	**558**	**558**	**558**	**558**	**558**	558,0	0,00
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	**757**	753	753	753	756	753	754	754	754,3	1,42
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	**1130**	1129	1126	1126	1127	1127	1125	1126	1128	1127,0	1,56
gap8-2	1134	1125	1130	**1133**	1126	1129	1126	1131	1125	1126	1126	1127,7	2,83
gap8-3	1141	1137	1138	**1139**	1138	1138	**1139**	1137	1137	1138	1137	1137,8	0,79
gap8-4	1117	1110	1109	**1114**	1108	**1114**	1110	1109	1111	1109	1110	1110,4	2,07
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	715	715	**716**	715	715	715	715,1	0,32
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**	**962**	**962**	**962**	**962**	**962**	**962**	**962**	962,0	0,00
gap10-3	960	957	956	956	956	957	**958**	957	957	957	957	956,8	0,63
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	**1170**	1167	**1170**	1168	**1170**	1169	1167	1169	1169	**1170**	1168,9	1,20
gap11-5	1171	1168	1168	1168	1168	1168	1168	**1169**	**1169**	1168	1168	1168,2	0,42
gap12-1	1451	**1450**	**1450**	**1450**	1448	1449	1449	1449	**1450**	1449	1449	1449,3	0,67
gap12-2	1449	1447	1447	1447	1447	1447	**o**	1447	1448	1448	1447	1447,4	0,70
gap12-3	1433	1429	**1430**	**1430**	1429	**1430**	**1430**	**1430**	1429	1428	1429	1429,4	0,70
gap12-4	1447	1444	**1446**	1444	1443	1443	1443	1445	1444	1443	1444	1443,9	0,99
gap12-5	1446	1445	1444	1445	1445	1445	**o**	1445	1445	1445	1445	1445,0	0,47

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

A solução ótima foi encontrada para 45 problemas, dos 60 testados. Para as outras 15 instâncias em que o ótimo não foi obtido, a melhor solução encontrada pelo SIA está muito próxima da ótima. Note que, os piores resultados foram obtidos para as instâncias gap8-1 e gap8-4, cujos resultados estão apenas a 0,27% da solução ótima.

Em termos comparativos, o SIA teve desempenho inferior ao Algoritmo Genético (AG) proposto por Chu & Beasley (1997) em relação aos resultados obtidos, pois este conseguiu encontrar a solução ótima para todas as instâncias. Entretanto, a grande vantagem do SIA está na capacidade de evoluir um conjunto solução, de tal forma que, no final do processo evolutivo uma grande quantidade de boas soluções é encontrada. Este resultado pode ser explicado pela grande capacidade do SIA de manter os ótimos locais, devido à sua eficiência na manutenção da diversidade, sem fazer com que todo o repertório de soluções convirja para um único ponto. A manutenção de um conjunto de soluções é extremamente importante quando há a possibilidade de que uma nova restrição venha a ser imposta posteriormente.

A Tabela 3 apresenta os parâmetros variáveis utilizados para a resolução de cada problema, bem como o tempo de processamento e a quantidade de soluções factíveis obtidas.

Tabela 3. Parâmetros variáveis, número de soluções factíveis e tempo de processamento.

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

Tempo e Quantidade de Soluções Factíveis obtidas no melhor resultado de 10 rodadas

Note que, para uma grande parte dos testes realizados, o SIA foi capaz de encontrar um conjunto solução final com cardinalidade igual ao número de anticorpos do repertório inicial. Quando isso não acontece, o caso em que a proporção entre o tamanho do repertório e quantidade de soluções é menor ocorre para a instância gap7-3, no qual o conjunto solução final

possui cardinalidade 22% menor que o repertório inicial, neste caso, partindo de 300 anticorpos (pontos no espaço de busca), obteve-se 234 boas soluções.

Para verificar a representatividade de cada solução dentro do conjunto solução obtido para cada instância, foi calculada uma matriz de distância entre as soluções. Essa matriz D_{ij} é quadrada com ordem igual ao número de soluções. Cada posição (i,j) corresponde o quanto a solução i difere da solução j, em termos percentuais, ou seja, quantos agentes foram alocados para realizar tarefas diferentes nas duas soluções. Em média, para todas as instâncias, cada solução é aproximadamente 30% diferente das outras dentro de um conjunto solução, ou seja, em média, seria necessário alocar 30% das tarefas para outros agentes de forma a "converter" uma solução i em uma solução j. Isto demonstra que além do SIA evoluir um conjunto amplo de soluções, todas as soluções do conjunto são representativas e distantes umas das outras.

O fato de evoluir um conjunto de soluções é interessante quando o problema pode apresentar mais restrições posteriormente, possibilitando que uma tomada de decisão possa ser adotada com urgência. Esta característica permite que um ou mais planos de contingência sejam especificados para atender eventuais problemas que podem vir a ocorrer.

5. CONCLUSÕES

O algoritmo ACO mostrou-se eficaz no tratamento do GAP, problema de natureza combinatorial explosiva, de difícil tratamento até a otimalidade. Foram obtidas soluções ótimas em grande parte dos testes efetuados e mesmo quando a solução ótima não é atingida, verifica-se um baixo desvio percentual com relação ao ótimo. A abordagem adaptativa para a utilização de diferentes termos heurísticos ao longo das iterações do ACO mostrou-se em uma técnica eficaz, que ajusta o algoritmo de acordo com a dificuldade ou não de encontrar uma solução factível. O tempo computacional do algoritmo foi relativamente pequeno, considerando um PC de 2 GHz, sendo inferior a 30 segundos para as maiores instâncias e praticamente instantâneo para as instância de menor porte.

Em relação ao desempenho do SIA, concluímos que apesar dos resultados obtidos para algumas instâncias serem ligeiramente inferior aos encontrados pelo método proposto por Chu & Beasley (1997), o fato do SIA conseguir preservar um conjunto diversificado de boas soluções (ótimos locais) permite que um plano de contingência seja proposto, possibilitando uma rápida tomada de decisão em caso de adição de novas restrições. Esta característica, de uma forma geral, faz com que o sistema torne-se mais robusto e menos sensível a falhas.

6. REFERÊNCIAS BIBLIOGRÁFICAS

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

(2) Bäck, T., Fogel, D.B. & Michalewicz, Z. (2000b), *Evolutionary Computation 2 Advanced Algorithms and Operators*, Institute of Physiscs Publishing, Bristol, UK.

(3) Chu, P.C. & Beasley, J.E. (1997), *A Genetic Algorithm for the Generalised Assignment Problem*, Computers Operacional Research, Elsevier Science, UK.

(4) Dasgupta, D. (1998a). *Artificial Immune Systems and Their Applications.* Springer-Verlag, Berlin, Germany.

(5) Dasgupta, D. (1998b). An Overview of Artificial Immune Systems and Their Applications, *Artificial Immune Systems and Their Applications.* Springer-Verlag, Berlin, Germany.

(6) De Castro, L.N. (2001). *Engenharia Imunológica: Desenvolvimento e Aplicação de Ferramentas Computacionais Inspiradas em Sistemas Imunológicos Artificiais*. Tese de Doutorado, Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e Computação.

(7) Dorigo, M.; Maniezzo, V. & Colorni, A. (1991) - *Positive feedback as a search strategy*, Technical Report 91-016, Dipartim ento di Elettronica e Informazione, Politecnico di Milano, Italy.

(8) Janeway, C.A., Travers P., Walport, M. & Capra, J.D. (2000). *Imunobiologia: O Sistema Imunológico na Saúde e na Doença.* 4 ed, Artmet, Porto Alegre, RS, Brasil.

(9) Kubo, M. & Kakazu, Y. (1993). *Simulating a competition for foods between ant colonies as a coordinated model of autonomous agents*. Proceedings of International Conference on Systems, Man, and Cybernetics, La Touquet, France, Vol. 5, p. 142-148.

(10) Lourenço, H. R., & Serra, D. (1998) – *Adaptive Approach Heuristics for The Generalized Assignment Problem*.

(11) Randall, M.(2004) – *Heuristics for Ant Colony Optimisation using the Generalised Assignment Problem*. IEEE.

(12) Stützle, T. & Dorigo, M. (1999) - *New ideas in optimization*, D. Corne, M. Dorigo & F. Glover (editors), McGraw-Hill.

(13) Timmis, J. (2000). Artificial Immune Systems: A Novel Data Analysis Technique Inspired by the Immune Network Theory. Tese de Doutorado, University of Whales, Department of Computer Science, Aberystwyth, Ceredigion, Wales.