Otimização por Colônia de Formigas (*Ant Colony Optimization - ACO*)

Eros Moreira de Carvalho
Gabriel Silva Ramos
{emc06,gsr04}@c3sl.ufpr.br

11 de Junho de 2007

Resumo

Neste artigo, apresentamos a metaheurística *Ant Colony Optimization* (ACO) e algums dos algoritmos que a implementam, em especial, o *Ant System* (AS), *Ant Colony System* (ACS) e *MAX-MIN Ant System* (MMAS). Aplicaremos esses algoritmos ao problema do caixeiro viajante, usando algumas bases da TSPLIB. Nosso objetivo, além de acentuar como a metaheurística ACO obtém um bom equilíbrio dos conceitos de diversificação e intensificação, é evidenciar, pelo resultados empíricos apresentados, que o ACO é uma boa metaheurística para a solução de problemas combinatórios e o MMAS a sua implementação de melhor desempenho.

1 Introdução

Problemas de otimização combinatória são difíceis de resolver, isto é, o seu custo computacional cresce exponecialmente com o aumento da entrada. É o caso, por exemplo, do problema do caixeiro viajante (*Traveling Salesman Problem* - TSP). Imagine um conjunto de cidades completamente conectadas entre si, isto é, para cada par de cidades, há uma estrada que as liga. O problema consiste, então, em encontrar o menor caminho para percorrer todas as cidades uma única vez. O conjunto de todos os caminhos possíveis definem o espaço de busca para este problema. Para conjuntos de poucas cidades, até cinco ou seis, podemos testar todas as possibilidades para encontrar o menor caminho. Mas o problema se torna computacionalmente intratável para conjuntos maiores de cidades. Tal fato despertou a necessidade de se elaborar estratégias computacionalmente baratas, mas que encontrassem soluções

ótimas ou próximas delas para esse tipo de problema. Essa é a idéia principal por trás das metaheurísticas. São estratégias para explorar o espaço de busca de maneira eficiente, encontrando soluções ótimas, próximas da melhor solução, mas sem ter de explorar o espaço de busca exaustivamente.

Há basicamente duas maneiras de abordar um problema combinatorial de maneira computacionalmente viável: uma é a construção passo a passo de uma solução usando algum tipo de informação heurística ou obtida por experiência para produzir uma solução ótima. Outra abordagem é partir de uma solução pronta e modificar alguns dos seus elementos para obter outras soluções melhores. Algoritmos baseados na primeira abordagem são chamados de *algoritmos de construção de solução*, ao passo que aqueles baseados na segunda abordagem são chamados de *algoritmos de busca local*. Uma metaheurística, enquanto estratégia mais geral para obter soluções ótimas a um custo razoável, pode se basear em uma das duas abordagens ou em ambas conjuntamente. ACO, como veremos, é primordiamlmente uma metaheurística baseada na construção de solução, embora ela possa ser combinada também com buscas locais.

Dois conceitos importantes para o estudo de metaheurísticas são o de *intensificação* e *diversificação*. A intensificação é a exploração mais exaustivia de uma região do espaço de busca onde se espera encontrar boas soluções, ao passo que a diversificação é a mudança do foco da busca para regiões ainda não exploradas [2]. Uma boa metaheurística deve obter um equilíbrio entre intensificação e diversificação. Ao longo do artigo, tentaremos mostrar como a metaheurística ACO obtém os efeitos de intensificação e diversificação no processo de busca de uma boa solução e quais dos seus componentes são responsáveis por estes efeitos.

O presente artigo tem a seguinte estrutura. Na próxima seção, apresentaremos a metaheurística ACO e três algoritmos que visam implementá-la, AS, ACS e MMAS, tentando sempre focar os elementos responsáveis pelos efeitos de intensificação e diversificação. Em seguida, explicitamos as buscas locais usadas em conjunto com ACO em nossos testes. Em uma outra seção, explicitamos alguns detalhes da implementação que fizemos. Em seguida, apresentamos os testes, algumas aplicações para o ACO e as nossas conclusões.

2 Otimização por Colônia de Formigas

A otimização por colônia de formigas (*Ant Colony Optimization* - ACO) é uma metaheurística recente para a solução de problemas combinatórios. Ela é inspirada no comportamento de formigas na busca de alimentos. Quando

uma formiga precisa decidir para onde ir, ela usa informação proveniente de feromônio previamente depositado por outras formigas que passaram por aquele local. A direção que tiver maior depósito de feromônio será escolhida pela formiga. Por este processo de busca, formigas são capazes de encontrar o menor caminho de uma fonte de comida para o seu ninho [3].

A metaheurística ACO está baseada em um processo de construção de solução e sua principal inovação é usar formigas artificiais que, ao percorrer um caminho, depositam uma certa quantidade de feromônio, que, então, irá influenciar a decisão das formigas que vierem em seguida. A inovação do processo de construção de solução do ACO ficará mais clara se a compararmos com outros processos de construção. A construção de solução mais simples e empregada em muitas outras metaheurísticas para produzir soluções iniciais a um custo pequeno é a construção completamente gulosa. Para o problema do caixeiro viajante, ela funciona assim. Dado um conjunto de N cidades, escolhe-se uma delas e a coloca na solução. Avalia-se, então, qual das cidades restantes está mais próxima daquela e ela é adicionada também na solução. Esse processo é repetido N vezes, até que se tenha um caminho completo, isto é, uma solução para o problema do caixeiro viajante. Repare, no entanto, que se tivermos N cidades, teremos apenas N soluções gulosas distintas. Uma vez escolhida a cidade inicial, a construção da solução gulosa torna-se completamente determinística.

O processo de construção de solução completamente guloso tem um efeito de diversificação e intensificação fracos. O número de soluções geradas é muito pequeno, o que limita a diversificação. E a intensificação é praticamente inexistene, já que nenhuma solução é usada como ponto de partida para uma exploração mais intensa na sua vizinhança por soluções melhores. Para piorar, as solução geradas por uma construção completamente gulosa são em geral sub-ótimas[5]. Uma idéia para contornar parcialmente esse problema é introduzir um elemento aleatório na construção gulosa. Por exemplo, para cada passo da construção, pode-se pegar as 4 cidades mais próximas à última cidade colocada na solução e fazer um sorteio entre elas para decidir qual será colocada no caminho em construção. Essa modificação da construção gulosa, com elementos aleatórios, é usada, por exemplo, pela metaheurística GRASP [1]. Embora essa modificação produza um efeito de diversificação, aumentando substancialmente a quantidade de soluções geradas, ela ainda continua sem um efeito de intensificação.

A metaheurística ACO pode ser considerada uma melhoria do processo de construção de solução, buscando obter, pela influência dos feromônios, um efeito tanto de diversificação quanto de intensificação. O processo de construção de solução no ACO é estocástico. Ele constroi uma solução iterativamente adicionando componentes de solução a uma solução parcial levando

em consideração informação heurística e informação de feromônio, que muda dinamicamente para refletir a experiência da formiga [5]. Quando a formiga dá um passo na construção da solução, ela faz um cálculo probabilístico baseada na quantidade de feromônio depositada nas arestas que ligam a sua posição atual até posições ainda não visitadas e na informação heurística relacionada a estas arestas. O feromônio depositado tem um efeito de diversificação, ao possibilitar um número muito maior de soluções do que uma estratégia puramente gulosa. Mas o feromônio também tem um efeito de intensificação, ao refletir a experiência das formigas, pois componentes que foram largamente usados no passado em boas soluções terão preferência pelas formigas que estão construindo uma solução.

Em seguida, passaremos a expor a estrutura geral da metaheurística ACO. Como caracterizada por Dorigo[5], seu pseudo-código pode ser formulado como mostra a figura 1.

```
Iniciar parâmetros, iniciar rastros de feromônio

Agendar Atividades
    Construção de Soluções
    Ações Globais [Opcional]
    Atualização dos Feromônios
Fim do Agendamento
```

Figura 1: Pseudo-código da metaheurística ACO

Como já dissemos, ACO é uma metaheurística baseada na construção de soluções. ACO também é uma metaheurística baseada em população, no caso, na cooperação entre formigas. Como podemos ver pelo pseudo-código, o ACO é dividido basicamente em três etapas. Na primeira fase, as formigas da colônia constroem passo a passo uma solução, aplicando uma decisão local estocástica com base na informação de feromônio e na informação heurística. É importante notar que não é necessário que as formigas caminhem em sincronia na construção da solução, o que facilita implementações concorrentes e assíncronas do ACO[5]. Enquanto caminha, construindo uma solução, a formiga avalia a solução parcial e, dependendo do algoritmo ACO, pode vir a depositar feromônio do último componente visitado, cooperando, assim, com as formigas que vierem em seguida por aquele caminho.

Depois que uma formiga completa uma solução, algumas ações globais podem ser realizadas sobre essa solução. Essa fase é opcional. A idéia é que um agente com conhecimento global realize sobre uma solução ações que formigas individuais não teriam condições de fazer. Por exemplo, um agente global pode observar o caminho de cada formiga e resolver depositar

feromônio extra nos componentes usados pela formiga que construiu a melhor solução[5]. Outro procedimento muito comum usado nesta fase é a aplicação de uma busca local sobre as soluções construídas na primeira etapa, obtendo assim um efeito maior de intensificação. De acordo com Stützle, usar a busca local nesta etapa é uma forma de hibridizar o ACO e as melhores implementações do ACO geralmente se valem desse recurso[6].

Por fim, temos a etapa de atualização dos feromônios. Essa etapa envolve tanto o depósito de feromônio, quanto a evaporação de feromônio. Vimos que as formigas podem depositar feromônio após cada passo da construção, o que acontece na fase de construção da solução. No entanto, a formiga também pode depositar feromônio após construir uma solução completa, o que pode ser feito nesta terceira fase do algoritmo. Além disso, para evitar uma convergência muito precoce do ACO, ou que ele fique preso em um ótimo local, o feromônio depositado deve evaporar ao longo do tempo. Essa atividade também é realizada neste etapa e através dela obtemos um efeito de diversificação.

Em seguida, iremos apresentar três variações do ACO. Como ficará claro adiante, a principal diferença entre eles geralmente recai sobre quando fazem o incremento do feromônio e como o fazem. Era de se esperar que assim fosse, pois esses são justamente os procedimentos responsáveis por incorporar ao processo de busca a experiência acumulada pelas formigas.

2.1 Problema do Caixeiro Viajante - TSP

Como os testes comparativos foram feitos com o problema do Caixeiro Viajante, é oportuno que o apresentemos agora de uma maneira mais formal. O problema, como já dissemos, consiste em encontrar o menor caminho para percorrer um conjunto de cidades que estão completamente conectadas entre si. Neste trabalho, trabalhamos apenas com a versão simétrica do problema. Cada cidade é associada a um nó do *grafo de construção* das formigas. A distância entre uma cidade i e j é chamada de d_{ij}. Assim, uma instância do TSP é um grafo (N, E), onde N é o conjunto de cidades e E o conjunto de arestas entre as cidades.

2.2 *Ant System* - AS

Ant System (AS) foi o primeiro exemplo de algoritmo ACO, desenvolvido por por Marco Dorigo, Vitorio Maniezzo e Alberto Colorni em 1996[4]. Embora o seu desempenho não seja suficiente para competir com os melhores algoritmos propostos para resolver o TSP, ele é usado como ponto de partida para a formulação de outros algoritmos ACO[5].

Na fase de construção, cada formiga escolhe para qual cidade ir com uma probabilidade que é uma função da distância da cidade e da quantidade de feromônio presente na aresta que conecta a essa cidade. Para não permitir passos inválidos, a formiga tem uma memória de quais cidades ela já visiou. Assim, enquanto ela constrói uma solução, ela é forçada a não visitar uma mesma cidade duas vezes até que a solução esteja completa. Quando a formiga termina um caminho completo, ela deposita uma certa quantidade de feromônio em cada aresta (i,j) que ela visitou[4]. No AS, não há depósito de feromônio durante a fase de construção da solução. Ao dar um passo, a formiga calcula a probabilidade de todas as cidades que ela ainda não visitou e escolhe ir para aquela de maior probabilidade. Essa probabilidade é definida pela fórmula:

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}$$

onde α e β são parâmetros para indicar respectivamente a importância do feromônio e da informação heurística. τ_{ij} é a quantidade de feromônio na aresta ij, η_{ij} é o inverso de d_{ij} e N_i^k o conjunto das cidades ainda não visitadas pela formiga k.

AS não realiza nenhuma ação global, nem busca local. Assim, a próxima fase, após a construção da solução, é a atualização do feromônio, que envolve tanto o incremento do feromônio, quanto a sua evaporação e é realizado por todas as formigas. A atualização é definida pela fórmula:

$$\tau_{ij} = (1 - \rho).\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k$$

onde ρ é um parâmetro para regular a taxa de evaporação do feromônio, m o número de formigas e $\Delta\tau_{ij}^k$ é definida por:

$$\begin{cases} \frac{1}{L^k}, & \text{se aresta } ij \text{ faz parte do caminho da formiga k} \\ 0, & \text{caso contrário} \end{cases}$$

onde L^k é o custo do caminho percorrido pela formiga k.

Como podemos observar, há vários parâmetros que devem ser ajustados para se obter um bom desempenho com AS. Um parâmetro para controlar a importância do feromônio, outro para a informação heurística, um terceiro para regular a taxa de evaporação e um quarto para indicar o tamanho da população de formigas. O ajuste correto desses parâmetros é fundamental para se obter um bom equilíbrio entre intensificação e diversificação com o AS. ρ, por exemplo, deve ser ajustado para um valor menor que 1, pois, do contrário, haverá uma acumulação ilimitada do feromônio, prejudicando a

diversificação[4]. Dorigo sugere um valor em torno de 0.5. α deve ser maior que 0. Quando o seu valor é alto, isso significa que o feromônio é muito importante e as formigas tendem a escolher arestas escolhidas por outras formigas no passado, o que tem um efeito de intensificação. Se o valor for baixo, a construção se comporta de maneira semelhante a uma construção gulosa aleatória[4]. No entanto, valores altos demais podem provocar a estagnação precoce do algoritmo. Dorigo propõe que α deve ficar em torno de 1. Um valor muito baixo de β provocaria estagnação precoce do algoritmo e um valor alto demais o aproxima de uma construção gulosa. O ajuste deste parâmetro é importante para se obter uma boa diversificação. Dorigo propõe que β fique em torno de 5. Geralmente, $m = n$, isto é, a quantidade de formigas igual a quantidade de cidades. Por fim, Dorigo propõe inicializar o feromônio de todas as arestas por $\frac{1}{n.L_{nn}}$, onde L_{nn} é o custo de uma construção puramente gulosa.

2.3 *Ant Colony System* - ACS

Ant Colony System (ACS) foi proposto como uma melhoria de AS, em 1997, por Marco Dorigo e Luca Gambardella[3].

No ACS existe um parâmetro adicional q_0 que define a probabilidade de ser utilizada uma nova expressão para a decisão de qual aresta cada formiga deve seguir na fase de construção. A cada iteração é sorteado um número aleatório e caso este seja menor que q_0, a regra de decisão utilizada é:

$$p_{ij}^k = \tau_{ij}\eta_{ij}^\beta$$

caso contrário, a regra de decisão utilizada é a mesma usada no algoritmo AS. Com valores de q_0 mais próximos de 1, tem-se uma priorização da intensificação, enquanto que com valores de q_0 mais próximos de 0, prioriza-se a diversificação, por conta da regra de decisão do AS ser utilizada com maior frequência.

Outra diferença do ACS com relação ao AS é que, a cada iteração da fase de construção as formigas atualizam o feromônio da aresta pela qual acabaram de atravessar, de acordo com a expressão:

$$\tau_{ij} = (1 - \varphi).\tau_{ij} + \varphi.\tau_0$$

onde φ é um parâmetro definido entre 0 e 1, e τ_0 é o valor inicial dos feromônios das arestas. A esta ação dá-se o nome de *atualização local* de feromônio, tendo sido inserida no algoritmo para contra-balancear a regra de decisão um tanto quanto gulosa mencionada acima.

Além da atualização local, o ACS compreende uma atualização global de feromônio, a qual é executada a cada fase de atualização. Esta atualização é feita somente pela formiga que construiu a melhor solução até o momento. Todas as arestas são atualizadas segundo a expressão:

$$\tau_{ij} = (1 - \rho).\tau_{ij} + \rho.\Delta\tau_{ij}^{melhor}$$

Com relação aos melhores parâmetros, Dorigo propõe que m seja igual a 10, baseado em observações de experimentos [3]. Já os valores de α e ρ devem ser iguais a 0.1, e β deve ser igual a 2, fazendo com que a construção gulosa seja priorizada. Ainda, para favorecer a intensificação, Dorigo sugere que q_0 deve valer 0.9. Por fim, também foi sugerido que a inicialização dos feromônios das arestas siga a expressão $\frac{1}{n.L_{nn}}$.

2.4 *MAX-MIN Ant System* - MMAS

Observando a convergência precoce do AS, Thomas Stützle e Holger Hoos propuseram o *MAX-MIN Ant System* (MMAS) em 1999[6].

A etapa de construção é praticamente idêntica a do AS, usando a mesma fórmula para calcular a probabilidade de cada aresta que a formiga ainda pode tomar. As modificações mais substanciais dizem respeito à taxa de atualização do feromônio e a sua limitação a certos valores máximo e mínimo. Como no AS, a atualização do feromônio é feita apenas quando termina a fase de construção, mas diferentemente do AS, a atualização é feita apenas pela melhor formiga. A fórmula para a atualização do feromônio é idêntica a fórmula de atualização global do ACS:

$$\tau_{ij} = (1 - \rho).\tau_{ij} + \Delta\tau_{ij}^{melhor}$$

No entanto, há uma diferença. No ACS, a melhor formiga é a formiga que construiu a melhor solução até o momento, enquanto no MMAS, a melhor formiga pode ser tanto a melhor formiga de uma iteração do algoritmo, quanto da formiga que tem a melhor solução até o momento. Em geral, MMAS usa a melhor formiga até o momento, mas há implementações que usam uma estratégia dinâmica, alternando entre uma e outra formiga ao longo do processo de busca[6]. De acordo com Stützle, o uso da melhor formiga de uma iteração aumenta o efeito de intensificação sobre as melhores soluções encontradas ao longo do processo de busca e, ao mesmo tempo, contribui para o efeito de diversificação se comparado ao uso da melhor formiga até o momento.

Outra diferença do MMAS em relação ao AS é que existem limites superior e inferior para a taxa de feromônio. Ao aplicar o incremento e a

evaporação sobre uma aresta, seu valor não pode ultrapassar τ_{max}, nem ser inferior a τ_{min}. Esses limites foram introduzidos para evitar a convergência precoce do algoritmo. Se a discrepância entre as taxas de feromônios de duas arestas começa a ficar muito grande, uma formiga vai acabar optando sempre pela aresta de maior taxa, reforçando ainda mais essa aresta quando fizer o seu depósito de feromônio. Esse processo pode levar a uma estagnação muito rápida do algoritmo. A proposta de impor limites superior e inferior é evitar que essa discrepância fique muito grande, diminuindo, assim, as chances de estagnação. Pode-se dizer que ela tem um efeito de diversificação. Uma dificuldade que surge é saber quais os valores apropriados para τ_{min} e τ_{max}. Stützle define τ_{max} por:

$$\frac{1}{\rho.L^*}$$

onde L^* seria o tamanho do caminho ótimo. Quando não se tem o conhecimento deste valor, pode-se aproximar dele por L_{bs}, isto é, pelo tamanho do melhor caminho encontrado até o momento. Stützle propõe também inicializar $\tau_0 = \tau_{max}$, usando L_{nn} na fórmula do τ_{max}. O objetivo desta inicialização é aumentar a diversificação no início do processo de busca[6].

Quanto ao valor de τ_{min}, não há completo acordo sobre como determiná-lo. Dorigo diz que ele geralmente é determinado empiricamente[5]. Stützle, no entanto, tem uma proposta analítica para determiná-lo[6]:

$$\tau_{min} = \frac{\tau_{max}.(1 - P_{dec})}{k.P_{dec}}$$

onde k é o número de escolhas que a formiga ainda pode fazer e $P_{dec} = \sqrt[n-1]{P_{best}}$, onde P_{best} é a probabilidade de uma formiga construir o melhor caminho até agora e n o número de passos no caminho.

Com relação aos melhores parâmetros, Stützle propõe uma taxa de evaporação baixa, com $\rho = 0.02$, privilegiando uma intensificação maior. Os valores de α e β, segundo os seus experimentos[6], devem ser ajustados respectivamente em 1 e 2.

2.5 *Comparativo*

Como foi dito anteriormente, as principais diferenças entre AS, ACS e MMAS recaem sobre a etapa de atualização do feromônio. Ela pode diferir de duas maneiras: i) quando é feita e ii) por quais formigas ela é feita. Essa diferenças estão expostas na tabela 1.

O uso da melhor formiga em ACS e MMAS tem por objetivo a intensificação da busca. Usar sempre a melhor formiga até o momento pode fazer

Alg.	Expressão	Realizado
AS	$\tau_{ij} = (1-\rho).\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$	por todas as formigas ao completar o caminho
ACS	$\tau_{ij} = (1-\varphi).\tau_{ij} + \varphi.\tau_0$	por todas as formigas em cada passo da construção
ACS	$\tau_{ij} = (1-\rho).\tau_{ij} + \rho.\Delta\tau_{ij}^{melhor}$	apenas pela melhor formiga até agora ao completar o caminho
MMAS	$\tau_{ij} = (1-\rho).\tau_{ij} + \Delta\tau_{ij}^{melhor}$	apenas pela melhor formiga da iteração (ou melhor até o momento) ao completar o caminho

Tabela 1: Comparativo da atualização de feromônios

com que o algoritmo fique preso a um ótimo local. Em razão disso, Stützle sugere que o MMAS use preferencialmente, ou na maioria das iterações, a melhor formiga da iteração, obtendo, assim, tanto um efeito de intensificação quanto de diversificação, já que, a cada iteração, teremos uma diferente melhor solução. O problema da estagnação precoce é resolvido no MMAS pela imposição de limites superior e inferior à taxa de feromônio.

3 Busca Local

Nesta seção serão apresentadas as estratégias de busca local utilizadas na hibridização dos algoritmos de ACO presentes neste trabalho.

3.1 *First Improvement*

A estratégia *First Improvement* (*FI*) consiste de uma busca local simples (conhecida também como subida de encosta - *hill climbing*) de primeira melhoria, ou seja, a partir de uma solução qualquer, escolhe-se dois nós vizinhos pertencentes ao caminho e troca-se a posição dos mesmos. Se houver uma melhora no custo, adota-se esta nova solução, caso contrário, desfaz-se a troca e escolhe-se outro par de nós vizinhos. Repete-se esta operação até que não seja mais possível executar uma troca que melhore o custo. O resultado é uma solução ótima local.

3.2 *2-opt*

Esta estratégia tem o mesmo princípio da *FI*, porém a estrutura de vizinhança é a seguinte: escolhe-se duas arestas quaisquer do caminho da solução de entrada, remove-as da solução, inverte-se um dos caminhos parciais resultantes e reconectam-se os caminhos.

3.3 *3-opt*

Já a estratégia *3-opt* é semelhante à *2-opt*, só que neste caso, três arestas são removidas. Após isso, os três caminhos resultantes são reconectados, possivelmente invertendo um ou mais deles.

4 Detalhes da Implementação

Os algoritmos AS, ACS e MMAS foram implementados em linguagem C. Usamos uma matriz $N \times N$ para representar o grafo (N,E) com os valores das distâncias entre uma cidade e outra e uma segunda matriz $N \times N$ para representar o grafo (N,E) com os valores das taxas de feromônio depositado em cada aresta, isto é, entre o caminho de uma cidade e outra. Cada formiga artificial é uma estrutura de dados com dois vetores de tamanho N e uma variável inteira. Um vetor é usado para guardar a solução em construção e o segundo para marcar as cidades ainda não visitadas pela formiga. A variável inteira retém o custo da solução construída pela formiga.

5 Testes

Com o objetivo de comparar os três algoritmos do ACO mencionados anteriormente foi executado um conjunto de testes. Para este conjunto, foi determinado um espectro de combinações dos melhores parâmetros encontrados na literatura junto com outros que se mostraram eficazes, além de variar o uso de estratégias de busca local entre as seguintes: *2-opt*, *3-opt*, *First Improvement*, ou ainda, sem busca local. Cada uma das implementações dos algoritmos foi executada 10 vezes para cada uma das 108 combinações de parâmetros, totalizando 3240 testes. O critério de parada utilizado foi o tempo limite de execução de 10 segundos por teste. A instância utilizada foi a **pcb3038**, da *TSPLIB*, cujo caminho ótimo tem custo 137694. Os testes foram feitos em um computador Intel Pentium 4 $3.2GHz$, com $3GB$ de memória RAM. A tabela 2 mostra os resultados dos testes.

Alg.	Média	σ	Melhor	Pior	Tempo Méd.
AS	158469.9	19366.7	141109	239852	7.06
ACS	154165.0	13257.9	139512	178748	7.88
MMAS	162705.5	27080.5	138924	256076	8.27

Tabela 2: Comparação entre os algoritmos AS, ACS e MMAS.

Analisando os resultados das médias e desvios padrão, vemos que o ACS obteve o comportamento mais regular ao longo do tempo, além de ter alcançado a melhor média. Porém, os melhores caminhos (menores custos) foram obtidos com execuções do MMAS, apesar de que, na média, esse algoritmo teve um comportamento mais variável que os outros. Com isso, o *MAX-MIN Ant System* foi eleito como o algoritmo mais promissor dentre os analisados.

5.1 Análise da variação de parâmetros

A seguir, analisaremos o comportamento da variação de alguns parâmetros utilizados na comparação anterior, usando-se o algoritmo MMAS. A figura 2 mostra o custo das melhores soluções encontradas ao longo do tempo para as quatro estratégias de busca local mencionadas. Observa-se claramente a diferença drástica entre as buscas locais *2-opt* e *3-opt* em relação à *First Improvement*, e destas em relação à execução sem busca local alguma. Isso deve-se ao fato de que as buscas *k-opt* são muito eficazes em encontrar ótimos locais, o que faz com que nas próximas fases de atualização, os feromônios de arestas melhores sejam incrementados, melhorando consequentemente as próximas construções de soluções, e assim sucessivamente.

Na figura 3 analisa-se a variação do custo das melhores soluções com os seguintes valores para o parâmetro α: 0.5, 1 e 2. Fica clara a melhora da convergência a medida que aumenta-se a importância das taxas de feromônio (parâmetro α) das arestas do grafo de construção em contraste com a importância da informação heurística das mesmas (parâmetro β).

Já na figura 4 apresenta-se novamente a variação do custo ao longo do tempo, porém, neste gráfico é o parâmetro β que tem seu valor variando entre 2, 5 e 8. Estas diferenças não surtiram muito efeito na convergência do algoritmo. Tem-se apenas uma pequena melhora a medida que aumenta-se a importância da informação heurística, o que faz com que as formigas tenham maior chance de escolherem por arestas de menor comprimento, ou seja, tenham um comportamento mais guloso.

Quanto à evaporação dos feromônios, de acordo com o gráfico da figura

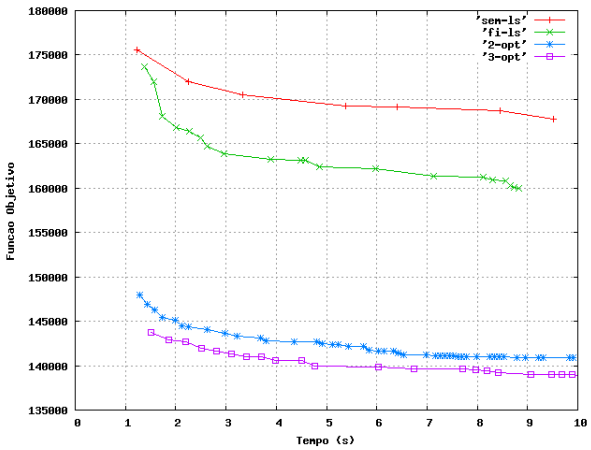

Figura 2: Variação da estratégia de Busca Local

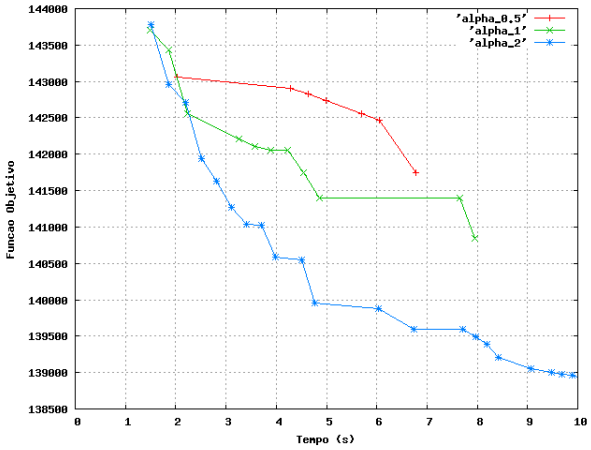

Figura 3: Variação do parâmetro α

5 pode-se notar que a medida que a taxa de persistência dos feromônios aumenta, tem-se uma convergência mais rápida para melhores soluções, o que confirma o efeito da diversificação proveniente da evaporação.

5.2 Testes com instâncias menores

Com o melhor algoritmo identificado e seus melhores parâmetros escolhidos, foram feitos testes com as instâncias pequenas que haviam sido propostas para serem analisadas neste trabalho. As quatro instâncias presentes nos testes são as seguintes: **dantzig42**, **fri26**, **gr48** e **hk48**, todas da *TSPLIB*. Os testes compreendem 10 execuções para cada estratégia de busca local,

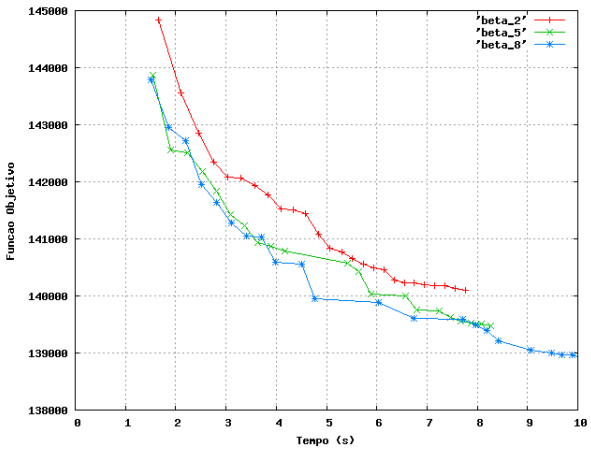

Figura 4: Variação do parâmetro β

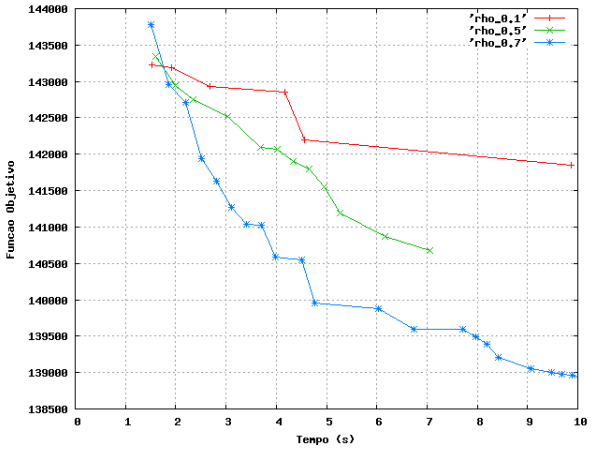

Figura 5: Variação do parâmetro ρ

totalizando 40 testes por instância. O computador utilizado foi o mesmo da bateria de testes apresentada anteriormente. Um resumo dos resultados pode ser visualizado na tabela 3. Nota-se que os resultados foram muito bons, atingindo o ótimo global em quase todas as execuções.

Observa-se na tabela 4 que o MMAS alcançou os ótimos globais destas instâncias em um tempo médio muito reduzido. Nas execuções com a busca local 3-opt e em uma delas com a 2-opt o ótimo global foi atingido na primeira iteração do algoritmo, o que justifica o tempo igual a 0.00.

Base	Média	σ	Melhor	Pior	Ótimo	Tempo Méd.
dantzig42	699	0	699	699	699	0.66
fri26	937	0	937	937	937	0.008
gr48	5051.1	7.4	5046	5069	5046	2.09
hk48	11461	0	11461	11461	11461	0.84

Tabela 3: Resultados dos testes com instâncias propostas

Base	Médias de Tempo			
	Sem busca	FI	2-opt	3-opt
dantzig42	2.48	0.14	0.04	0.00
fri26	0.02	0.01	0.00	0.00
gr48	5.40	2.90	0.04	0.00
hk48	2.24	1.03	0.09	0.00

Tabela 4: Médias de Tempo do MMAS para as buscas locais

6 Aplicações

Os algoritmos da metaheurística ACO podem ser aplicados na resolução de uma vasta gama de problemas de otimização combinatória. Dentre estes, os mais comuns são: o problema do caixeiro viajante (mencionado anteriormente), problemas de roteamento em redes de comunicação, problemas de roteamento de veículos e o problema da atribuição quadrática (QAP). Enfim, qualquer problema cujos componentes de solução possam ser modelado com um grafo completamente conectado pode ser resolvido com grandes chances de sucesso pelos algoritmos de ACO.

7 Conclusões

Neste trabalho foram analisados três algoritmos da metaheurística ACO, sendo que o algoritmo *MAX-MIN Ant System* se mostrou o melhor dentre eles por apresentar melhor convergência ao ótimo global das instâncias do problema do caixeiro viajante analisadas.

A adição de busca local como forma de hibridização do ACO mostrou-se muito importante e praticamente essencial para a obtenção de bons resultados [5]. Neste ponto dá-se destaque para a busca local *3-opt*, a qual mostrou-se como a mais eficaz em encontrar ótimos locais dentre as estratégias ana-

lisadas.

Além disso, notou-se que quando leva-se em conta o L_{nn} na inicialização dos feromônios, obtém-se soluções melhores na fase de construção, e consequentemente levando à obtenção de melhorias no custo mais rapidamente.

Por fim, concluiu-se que o uso de conhecimento adquirido durante a busca para gerar novas soluções é uma idéia promissora para algoritmos baseados na construção de soluções, podendo ser incorporada, com um certo trabalho, em outras metaheurísticas.

Referências

[1] D. Andrade and M. Resende. Grasp with evolutionary path-relinking. *AT&T Labs Research Technical Report*, TD-6XPTS7, janeiro 2007.

[2] C. Blum and A. Roli. Metaheuristics in combinatorial optimizaiton: Overview and conceptual comparation. *ACM Computing Surveys*, 35(3):268–308, 2003.

[3] M. Dorigo and L.M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation, Vol.1, No.1*, pages 1–24, 1997.

[4] M. Dorigo, V. Maniezzo, and Colorni A. Ant system: Optimization by a colony of cooperatings agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26(1):1–13, 1996.

[5] M. Dorigo and T. Stützle. Ant colony optimization metaheuristic: Algorithms, applications, and advance, 2002.

[6] T. Stützle. Max-min ant system. *Preprint submitted to Elsevier Science*, pages 1–26, 1999.