Comparação de performance entre Algoritmos Genéticos e Colonia de Formigas para a solução do Problema do Caixeiro Viajante Assimétrico

Killdary A. Santana[1], Rafael H. Bordini[2], Flávio Rech Wagner[1], Jomi F. Hübner[3]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{nedel,flavio}@inf.ufrgs.br

Abstract. *This meta-paper describes the style to be used in articles and short papers for SBC conferences. For papers in English, you should add just an abstract while for the papers in Portuguese, we also ask for an abstract in Portuguese ("resumo"). In both cases, abstracts should not have more than 10 lines and must be in the first page of the paper.*

Resumo. *Um dos objetivos do presente trabalho é a comparação entre o Algoritmo Genético(AG) e o Algoritmo Colônia de Formigas(ACF), como método de solução ótima para resolver o problema de roteirização de robôs autônomos. O problema das rotas é modelado em cima do Problema do Caixeiro Viajante Assimétrico. Também se estuda possível adaptação dos algoritmos em uma plataforma embarcada para que o robô por se mesmo encontrar a melhor rota. Conclusão ainda não realizada.*

Introdução

Com o avanço da tecnologia os robôs se tornaram mais eficazes na execução de diversas tarefas, podendo executar funções que antes eram consideradas impossíveis para uma máquina, deixando de ser um componente caro e acessível apenas para grandes indústrias e passando a atuar nas mais diversas aplicações que antes eram executadas por humanos, impulsionando assim o desenvolvimento de robôs autônomos.

A finalidade da pesquisa de robôs móveis com autonomia é construir máquinas para realizar tarefas com precisão e capazes de tomar decisões adequadas frente a uma situação inesperada.

Uma das maiores problemáticas em robôs móveis se dá no planejamento de sua trajetória e tem recebido grande atenção por parte dos pesquisadores, pois seu desenvolvimento está diretamente relacionado com a maior autonomia dos robôs. A complexidade do problema de planejamento do movimento tem motivado o desenvolvimento dos mais diversos algoritmos. Tal complexidade advém da necessidade de integrar a navegação do robô com o sensoriamento, a eficiência e o planejamento de rotas.

O problema de roteirização de robôs pode ser modelado através do Problema do Caixeiro Viajante(PCV). Este trabalho tem como objetivo aplicar 2 algoritmos diferentes para achar uma solução próxima da ideal e comparar o resultado e a performance entre as duas soluções estudadas. As duas soluções a serem implementadas serão algoritmos genéticos e colonia de formigas.

Dentre os objetivos específicos tem-se:

1. Investigar sobre os 2 métodos utilizados para o problema do caixeiro viajante;
2. Comparar o desempenho de operadores de recombinação dos algoritmos genéticos;
3. Comparar o desempenho entre as técnicas de IA escolhidas;
4. Verificar possibilidades de melhorias e adaptação parar sistemas embarcados.

Na próxima seção será descrito o PCV e sua variação assíncrona. Na seção 3 será abordado a fundamentação do problema e das soluções a serem implementadas. A seção 4 abordará os métodos de implementação e suas metodologias. Na seção 5 será exposto os resultados obtidos oriundos da seção 4, enquanto a solução será apresentada na seção 6.

Problema do Caixeiro Viajante

O Problema do Caixeiro Viajante (PCV) tem sido muito utilizado no experimento de diversos métodos de otimização por ser, principalmente, um problema de fácil descrição e compreensão, mas de grande dificuldade de solução, uma vez que é NP(Non-Deterministic Polynomial time). O PCV determina que um vendedor tem N cidades no qual o mesmo deveria visitar todas as cidades, sem repetir nenhuma, e voltar para a cidade de partida de modo que o custo da viagem seja mínimo [**?**]. O cálculo de rotas mínimas em robôs se assemelha ao PCV, sendo o robô o vendedor, as cidades os pontos objetivos que o robô deve visitar e as arestas a distância que o robô deve percorrer. O problema do caixeiro é um clássico exemplo de problema de otimização combinatória, no qual o número de soluções possíveis é representado na equação 1.

$$R(n) = (n-1)! \tag{1}$$

Onde:

- n: é o numero de cidades a serem visitadas;

O PVC pode ser classificado em simétrico, onde os custos de um caixeiro ir da cidade A para a cidade B e vice-versa são os mesmos, ou assimétrico, onde os custos de movimentação de uma cidade A para uma cidade B são diferentes de ir da cidade B para a cidade A. Para este trabalho será focado no problema **assimétrico**.

FORMULAÇÃO MATEMÁTICA PARA O PCV ASSIMÉTRICO

Seja o grafo G(N, A) onde N representa o conjunto ($|N| = n$) e ao conjunto de arestas. Seja, uma matriz simétrica com custos ou distâncias mínimas entre os nós da rede considerando ainda que $c_{ij} = +\infty \forall i \in N$. A matriz $X[x_{ij}]$ é composta pelas variáveis de decisão do problema

$$x_{ij} = \begin{cases} 1 & \text{, se: o arco } a_{ij} \in \text{rota} \\ 0 & \text{, se: o arco } a_{ij} \notin \text{rota} \end{cases} \tag{2}$$

Desta forma, a formulação de Programação Linear Inteira para o problema, devido a Golden et al., de 1977, pode ser escrita como:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{3}$$

Sujeito a:

$$\sum_{i=1}^{n} x_{ij} \qquad j = 1, 2, \ldots, n \tag{4}$$

$$\sum_{j=1}^{n} x_{ij} \qquad i = 1, 2, \ldots, n \tag{5}$$

$$X = x_{ij} \in S \tag{6}$$

$$x_{ij} = 0 \qquad \text{ou} \qquad x_{ij} = 1 \tag{7}$$

Os dois primeiros grupos de restrições (4) e (5) garantem que exatamente um arco (i, j) tem origem cada nó i da rota e exatamente um arco(i, j) é direcionado para um nó j da rota. A penúltima restrição (6) contém um subconjunto S que pode ser qualquer conjunto de restrições que impeça a formação de sub-rotas. Estas restrições são chamadas restrições de quebra de sub-rotas e podem ser, entre outras:

$$S = \left\{ (x_{ij}) \geq 1 \text{,para todo conjunto próprio não vazio } Q \text{ de } N \right\} \tag{8}$$

$$S = \left\{ (x_{ij}) \leq |R| - 1 \text{, para todo subconjunto não vazio } R \text{ de \{ 2,..., n \}} \right\} \tag{9}$$

$$S = \left\{ (x_{ij}) : y_i - y_j + n x_{ij} \leq (n-1) \text{ , para } 2 \leq i \neq j \leq n \text{ para alguns números reais } y_i \right\} \tag{10}$$

Fundamentação Teórica

Dentre as técnicas aplicadas ao PCV foram escolhidos os algoritmos genéticos e colonia de formigas, que serão abordadas a seguir.

Algoritmo Genético

Essencialmente, Algoritmos Genéticos são métodos de busca e otimização que tem sua inspiração nos conceitos da teoria de seleção natural das espécies proposta por Darwin (Goldberg, 1989) (Koza, 1992) (Mitchell, 1994) (Back, 1996) (Fogel et al., 1966).

Os sistemas desenvolvidos a partir deste princípio são utilizados para procurar soluções de problemas complexos ou com espaço de soluções muito grande (espaço de busca), o que os tornam problemas de difícil modelagem e solução quando se aplicam métodos de otimização convencionais.

Estes algoritmos são inspirados nos processos genéticos de organismos biológicos para procurar soluções ótimas ou sub-ótimas. Para tanto, procede-se da seguinte maneira: codifica-se cada possível solução de um problema em uma estrutura chamada de ”cromossomo”, que é composta por uma cadeia de bits ou símbolos. Estes cromossomos representam indivíduos, que são evoluídos ao longo de várias gerações, de forma similar aos seres vivos, de acordo com os princípios de seleção natural e sobrevivência dos mais aptos, descritos pela primeira vez por Charles Darwin em seu livro ”A Origem das Espécies”. Emulando estes processos, os algoritmos genéticos são capazes de ”evoluir” soluções de problemas do mundo real.

Os cromossomos, ou indivíduos, são então submetidos a um processo evolucionário que envolve avaliação, seleção, recombinação (crossover) e mutação. Após vários ciclos de evolução a população deverá conter indivíduos mais aptos. Os algoritmos genéticos utilizam uma analogia direta deste fenômeno de evolução na natureza, onde cada indivíduo representa uma possível solução para um problema dado. A cada indivíduo atribui-se um valor de avaliação: sua aptidão, que indica quanto à solução representada por este indivíduo é boa em relação às outras soluções da população. Desta maneira, o termo População refere-se ao conjunto de todas as soluções com as quais trabalha o sistema. Aos indivíduos mais adaptados é dada uma probabilidade maior de se reproduzirem mediante cruzamentos com outros indivíduos da população, produzindo descendentes com características de ambas as partes. A mutação também tem um papel significativo, ao introduzir na população novos indivíduos gerados de maneira aleatória. O processo de evolução começa com a criação aleatória dos indivíduos que formarão a população inicial. A partir de um processo de seleção baseado na aptidão de cada indivíduo, são escolhidos indivíduos para a fase de reprodução que cria novas soluções utilizando-se, para isto, um conjunto de operadores genéticos. Deste modo, a aptidão do indivíduo determina o seu grau de sobrevivência e, assim, a possibilidade de que o cromossomo possa fazer parte das gerações seguintes. O procedimento básico de um algoritmo genético é resumido na Figura 10 (Davis, 1996).

Algoritmo Genetico 2

O Algoritmo Genético (AG) constitui um modelo matemático que simula a teoria da evolução Darwiniana, no qual existe inicialmente um conjunto de indivíduos (população inicial), que representam possíveis soluções para um determinado problema e a cada iteração, chamada de geração em AG. Os indivíduos são avaliados em relação ao seu nível de adaptabilidade com o meio externo e os mais aptos são selecionados para gerar descendentes.

Uma nova população é gerada através da aplica¸c?ao de operadores genéticos, como recombinação e mutação de genes. Este processo é realizado até um determinado número de gerações e o indivíduo mais apto encontrado é dito ser a solução do problema (LOPES, 2006).

O algoritmo genético foi apresentado inicialmente por John Holland em seu trabalho intitulado de?Adaptation in Natural and Artificial Systems?em 1975, com o objetivo de formalizar matematicamente e explicar os processos de adapta¸c?ao de processos naturais e desenvolver sistemas artificiais que mantenham os mecanismos originais encontrados em sistemas naturais (IYODA, 2000).

Em AG os indivíduos são representados por cromossomos e cada símbolo do cromossomo é chamado de gene. Os genes por sua vez armazenam informações, os alelos.

Os cromossomos são geralmente implementados como uma cadeia de bits, comumente utiliza-se um vetor como estrutura de armazenamento. O nível de adaptabilidade do indivíduo em relação ao meio é calculado com base na função objetivo, que nos casos mais simples é própria função que se quer maximizar. Em AG esta função é denominada função de fitness.

O procedimento básico de um algoritmo genético é resumido no código abaixo:

```
begin
  t = 0;
  inicia P(t);
  avaliar P(t);
  while ãno çãcondio de parada do
    t = t + 1;
    selecione P(t) a partir de P(t ? 1);
    altere P(t);
    avalie P(t);
  end
end
```

Para determinar o final da evolução pode-se fixar o número de gerações, o número de indivíduos criados, ou ainda condicionar o algoritmo à obtenção de uma solução satisfatória, isto é, quando atingir um ponto ótimo. Outras condições para a parada incluem o tempo de processamento e o grau de similaridade entre os elementos numa população (convergência).

O AG possui um processo evolutivo composto pelas seguintes etapas:

1. Avaliação: análise da aptidão dos indivíduos (soluções) para verificar a sua resposta ao problema;
2. Seleção: seleção dos indivíduos para reprodução. São selecionados os indivíduos mais aptos para a solução;
3. Cruzamento: os indivíduos selecionados são cruzados para geração de novos indivíduos;
4. Mutação: características de indivíduos selecionados são alteradas para dar variedade à população;
5. Atualização: os indivíduos gerados são inseridos na população para próxima geração;
6. Finalização: verificação se a condição de parada do algoritmo foi atingida e o algoritmo é encerrado em caso positivo ou retorna a etapa de avaliação.

Colônia de formigas

A otimização por colônia de formigas (Ant Colony Optimization - ACO) é uma metaheur??stica recente para a solução de problemas combinatórios. Ela é inspirada no comportamento de formigas na busca de alimentos. Quando 2uma 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.

A meta-heurística 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 meta-heurísticas 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 determinísticoa.

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 pratica- mente inexistene, já que nenhuma solução é usada como ponto de partida para uma exploração mais intensa na sua vizinhança por soluções melho- res. Para piorar, as solução geradas por uma construção completamente gulosa são em geral sub-ótimas. 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 meta-heurística GRASP. Embora essa modificação produza um efeito de diversificação, aumentando substancialmente a quantidade de soluções geradas, ela ainda continua sem um efeito de intensificação.

A meta-heurística 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 itera- tivamente adicionando componentes de solução a uma solução parcial levando 3em consideração informação heur??stica 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 probabil??stico ba- seada na quantidade de feromônio depositada nas arestas que ligam a sua posição atual até posições ainda não visitadas e na informação heur??stica 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 metaheur??stica ACO. Como caracterizada por Dorigo, seu pseudo-código pode ser formulado como mostra o codigo

abaixo.

```
Iniciar parametros, iniciar rastros de feromonio

Agendar Atividadesçã
  Construo de Solucoes
  Acoes Globais [Opcional]
  Atualizacoo dos Feromonios
Fim do Agendamento
```

Como já dissemos, ACO é uma metaheur??stica baseada na construção de soluções. ACO também é uma metaheur??stica 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 heur??stica. É importante notar que não é necessário que as formigas caminhem em sin- cronia na construção da solução, o que facilita implementações concorrentes e ass??ncronas 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 ideia 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.

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.

Subsections

The subsection titles must be in boldface, 12pt, flush left.

Figures and Captions

Figure and table captions should be centered if less than one line (Figure 1), otherwise justified and indented by 0.8cm on both margins, as shown in Figure 2. The caption font must be Helvetica, 10 point, boldface, with 6 points of space before and after each caption.

Figure 1. A typical figure

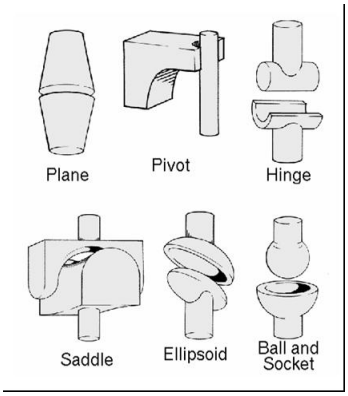

Figure 2. This figure is an example of a figure caption taking more than one line and justified considering margins mentioned in Section 6.

In tables, try to avoid the use of colored or shaded backgrounds, and avoid thick, doubled, or unnecessary framing lines. When reporting empirical data, do not use more decimal digits than warranted by their precision and reproducibility. Table caption must be placed before the table (see Table 1) and the font used must also be Helvetica, 10 point, boldface, with 6 points of space before and after each caption.

Table 1. Variables to be considered on the evaluation of interaction techniques

	Chessboard top view	Chessboard perspective view
Selection with side movements	6.02 ± 5.22	7.01±6.84
Selection with in-depth movements	6.29±4.99	12.22±11.33
Manipulation with side movements	4.66± 4.94	3.47±2.20
Manipulation with in-depth movements	5.71 ±4.55	5.37 ±3.28

Images

All images and illustrations should be in black-and-white, or gray tones, excepting for the papers that will be electronically available (on CD-ROMs, internet, etc.). The image resolution on paper should be about 600 dpi for black-and-white images, and 150-300 dpi for grayscale images. Do not include images with excessive resolution, as they may take hours to print, without any visible difference in the result.

References

Bibliographic references must be unambiguous and uniform. We recommend giving the author names references in brackets, e.g. [Knuth 1984], [Boulic and Renault 1991], and [Smith and Jones 1999].

The references must be listed using 12 point font size, with 6 points of space before each reference. The first line of each reference should not be indented, while the subsequent should be indented by 0.5 cm.

References

Boulic, R. and Renault, O. (1991). 3d hierarchies for animation. In Magnenat-Thalmann, N. and Thalmann, D., editors, *New Trends in Animation and Visualization*. John Wiley & Sons ltd.

Knuth, D. E. (1984). *The TEX Book*. Addison-Wesley, 15th edition.

Smith, A. and Jones, B. (1999). On the complexity of computing. In Smith-Jones, A. B., editor, *Advances in Computer Science*, pages 555–566. Publishing Press.