

SIMULANDO GERAÇÕES SOBREPOSTAS E NÃO-SOBREPOSTAS.

Rafael Keith Ono, Adam Taiti Harth Utsunomiya, Michele Porto Pires, Ricardo da Fonseca. Laboratório de Computação Científica aplicada a Zootecnia (Lucca-Z) - Faculdade de zootecnia – Campus Experimental de Dracena

Introdução

Experimentos em melhoramento animal são difíceis de serem conduzidos e mantidos devido aos altos custos de manutenção e longos períodos para obtenção de resultados. A simulação de dados pode ser uma solução para esses problemas. Dessa forma, a elaboração de bons algoritmos e sua implementação são tarefas importantes para os melhoristas.

Na área de melhoramento animal, os estudos dos algoritmos estão focados principalmente na avaliação genética e em procedimentos relacionados. Uma vez que a demanda é menor por recursos computacionais e cada estudo exige dados particulares, os algoritmos para simulação de dados não são, de modo geral, discutidos.

Em geral, algoritmos para simular populações evoluindo por meio de gerações sobrepostas e não-sobrepostas sob acasalamento ao acaso são necessários para estudos em melhoramento animal. Eles são a base para o desenvolvimento de esquemas mais complexos para a evolução de populações. Por essa razão, bons algoritmos para executar essas tarefas devem melhorar a eficiência do software e a qualidade dos dados simulados.

Assim, o objetivo desse trabalho é desenvolver algoritmos para a geração de populações evoluindo de acordo com gerações sobrepostas e não-sobrepostas.

Algoritmo

Deve-se assumir que a população base já existe e os indivíduos estão armazenados em um vetor (**pop**) e que o primeiro elemento está na posição 0. O algoritmo para se criar a população base é mostrado por Pires *et al.* (2005). Os indivíduos são identificados pelo tempo em que são criados. Deste modo, o primeiro indivíduo (na posição 0) tem identificação (*id*) um, o segundo 2 e assim por diante. Também é assumido que os indivíduos devem ser distinguidos entre machos e fêmeas, para uma geração, o número de machos e de fêmeas são iguais.

Geração não-sobreposta. Fornecido o número de machos (*nm*), o número de fêmeas por macho (*nf*) e o número de descendentes por cruzamento (*no*), é possível simular uma população evoluindo em *g* gerações não-sobrepostas sob acasalamento ao acaso seguindo os passos abaixo:

1. Crie um vetor (**controle**);
 1. O vetor será usado para armazenar a identificação evitando que o mesmo indivíduo seja usado na mesma geração. O vetor será visitado a cada momento que um indivíduo for escolhido para ser pai.
2. Crie um vetor de controle individual de machos (**controle_m**);
3. Crie um vetor de controle individual de fêmeas (**controle_f**);

No caso da geração não-sobreposta, os vetores **controle_m** e **controle_f** são designados somente com o propósito de distinguir machos de fêmeas, contudo, no caso de geração sobreposta, eles possuem uma função adicional, que será explicada posteriormente.
4. Para cada geração (*g_i*) até *g*;

$g_i = 0, 1, 2, \dots, g$. Admite-se 0 para a população base, 1 para a primeira geração depois da população base e assim por diante.

4.1 Para cada indivíduo macho até nm ;

4.1.1 Gere um número aleatório ($n1$), de uma distribuição uniforme no intervalo $[g_i (nm \times nf \times no)/2, tamanho\ do\ vetor\ controle_m - 1]$;

O número aleatório representa a posição no vetor **controle_m** que contém o indivíduo selecionado para ser o pai. Para a primeira geração, g_i é 0 e o número aleatório deve ser gerado no intervalo $[0, tamanho\ do\ vetor\ controle_m - 1]$. Para a segunda geração, g_i é 1 e o número aleatório deve ser gerado $[(nm \times nf \times no)/2, tamanho\ do\ vetor\ controle_m - 1]$, portanto, os indivíduos da população base não serão selecionados para ser pais da geração 2, desde que $(nm \times nf \times no)/2$ é a posição no vetor **controle_m** do primeiro indivíduo macho da geração 1.

4.1.2 Se o *id* do indivíduo escolhido estiver no vetor **controle**, execute o passo 4.1.1 novamente; senão, coloque o *id* no **controle**;

Este passo previne que o indivíduo seja usado como pai mais de uma vez por geração. Um procedimento de busca é necessário nesse ponto.

4.1.3 Coloque o *id* no vetor **controle_m**;

4.1.4 Para cada indivíduo fêmea até nf ;

4.1.4.1 Gere um número aleatório ($n2$), de uma distribuição uniforme do intervalo $[g_i (nm \times nf \times no)/2, tamanho\ do\ controle_f - 1]$;

O número aleatório representa a posição no vetor **controle_f** que contém o indivíduo selecionado para ser a mãe.

4.1.4.2 Se o *id* do indivíduo escolhido estiver no vetor **controle**, execute o passo novamente; senão, coloque o *id* no vetor **controle**;

4.1.4.3 Para cada novo indivíduo até no ;

4.1.4.3.1 Use um procedimento para gerar um novo indivíduo com os pais **controle_m**[$n1$] e **controle_f**[$n2$];

O algoritmo para simular a zigotogênese é apresentado por Vaz *et al.* (2005) e um procedimento similar é apresentado por Fonseca (2003).

4.1.4.3.2 Crie um contador (c) e inicialize com 0;

4.1.4.3.3 Se c for menor que $no/2$, coloque o *id* do indivíduo no vetor **controle_m**; senão, coloque o *id* do indivíduo no vetor **controle_f**;

Este passo garante que metade dos descendentes serão machos e a outra metade serão fêmeas.

4.1.4.3.4 Coloque o indivíduo no vetor **pop**;

4.2 Apague todos os registros do vetor **controle**;

Geração sobreposta. Fornecido o número de machos (nm), o número de fêmeas por macho (nf) e o número de descendentes por cruzamento (no), é possível simular uma população evoluindo em g gerações sobrepostas sob acasalamento ao acaso. O algoritmo é similar ao de gerações não-sobrepostas, com pequenas mudanças no intervalo para gerar números aleatórios.

O passo 4.1.1 deve ser modificado para: Gere um número aleatório ($n1$), de uma distribuição uniforme do intervalo $[0, tamanho\ do\ vetor\ controle_m - 1]$;

Porque na geração sobreposta indivíduos de outra geração podem ser pais, o intervalo do número aleatório começa com 0 para ajustar indivíduos de gerações anteriores.

O passo 4.1.4.1 deve ser reescrito como: Gere um número aleatório ($n2$), de uma distribuição uniforme do intervalo $[0, \text{tamanho do vetor } \textit{controle_f} - 1]$;

Os vetores *controle_m* e *controle_f* do algoritmo de gerações sobrepostas possuem a função adicional para evitar que indivíduos machos de gerações anteriores sejam utilizados como fêmeas em gerações seguintes.

Ilustração

Para demonstrar a performance dos dois algoritmos, eles foram codificados em C++, usando o compilador G++ no sistema operacional SUSE Linux 9.3. Os procedimentos necessários para gerar a população base e a zigotogênese são os mesmos para ambos, assim, diferenças em performance são essencialmente devido à diferenças nos algoritmos. O computador usado tem a seguinte configuração: processador 2.8 Ghz, 512 MB de memória RAM. Os resultados são representados pela Tabela 1.

Tabela 1. Avaliação das diferentes performances do tempo de execução (minutos,segundos) e quantidade de memória RAM utilizada (%) dos algoritmos para gerar 20 gerações sobrepostas e não-sobrepostas com 20.000 indivíduos.

<i>Gerações não-sobrepostas</i>		
Algoritmo	Tempo de execução	Memória usada
Geração sobreposta	2,32	23,7
Geração não-sobreposta	15,39	23,7

Os dois algoritmos tiveram a mesma porcentagem de memória RAM utilizada no final de vinte gerações. Esse resultado era esperado já que os algoritmos utilizavam vetores e variáveis para armazenar dados iguais e foram manipulados da mesma forma. De qualquer modo, o algoritmo de geração sobreposta foi mais rápido que o de geração não-sobreposta. No caso da geração não-sobreposta, a quantidade de cálculos é maior do que nas gerações sobrepostas. Nos passos 4.1.1 e 4.1.4.1 do algoritmo de gerações não-sobrepostas, a expressão $g_i (nm \times nf \times no)/2$ deve ser adicionada a cada número aleatório gerado, embora isso não seja necessário para os mesmos passos do algoritmo de geração sobreposta. Por exemplo, a expressão será adicionada pelo menos $nm \times nf$ (10.000) vezes por geração, se nenhum número amostrado já está no *controle* (passos 4.1.2 e 4.1.4.2). No final do programa (20 gerações), a implementação do algoritmo de gerações não-sobrepostas executou pelo menos 200.000 cálculos a mais do que a outra implementação. Além disso, para que o algoritmo de gerações não-sobrepostas encontre um indivíduo, o processador precisa “caminhar” mais nos vetores *vm* e *vf* para cada nova geração, desde que os indivíduos antigos não podem ser usados como pais. Por exemplo, para simular uma segunda geração o processador “caminhará” 10.000 elementos (indivíduos da população-base) para procurar o primeiro candidato a pai (primeiro indivíduo da geração 1). Para cada geração o “caminhar” aumenta pela quantidade $(nm \times nf \times no)/2$ para *vm* e *vf*. Na nonagésima geração o “caminhar” deve ser de 1.900.000 indivíduos. No caso de gerações sobrepostas, os “caminhares” são mais variáveis, desde que os indivíduos da primeira geração podem ser escolhidos mais de uma vez. Na média, o processador “caminha” muito menos.

Referências Bibliográficas

Fonseca, R (2003) *Doctor Thesis*, Universidade Federal de Viçosa, Viçosa – MG - Brasil.

Pires, M.P., Ono, R.K., Vaz, R.I., Utsunomyia, A.T.H. and Fonseca, R. (2005) *Proc. 51º Congresso Brasileiro de Genética*: cd-rom.

Vaz, R.I., Utsunomyia, A.T.H., Pires, M.P., Ono, R.K. and Fonseca, R (2005) *Proc. 42ª Reunião Anual da Sociedade Brasileira de Zootecnia*: cd-rom.

Bolsa: FAPESP