Algoritmo genético - Pseudocódigo

Fabricio Kassardjia - nusp:2234961

23 de setembro de 2019

1 Introdução

Pseudocódigo representando o algoritmo genético simples de forma a apresentar uma solução para modelo de Ising em uma matriz 10×10, com cada nó (spin) representado pelos possíveis valores -1 e 1.

Sendo $\Lambda = [-5, 5]^2$ as posições da matriz, e $\sigma_t \in \{-1, 1\}$ representando os valores de cada spin em determinada posição da matriz, e seja $(\sigma_t)_{t\in\Lambda} = \sigma$ um conjunto de valores da matriz.

A função de avaliação que deve ser minimizada é dada por $H(\sigma) = -\sum_{<t,t'>} \sigma_t\sigma_{t'}$ onde t e t' representam dois spins vizinhos.

2 Pseudocódigo

O código irá representar cada cromossomo da população como um vetor de valores inteiros de tamanho 100, representando assim a matriz 10×10, alinhando uma linha da matriz na sequencia da outra no vetor. Assim o nosso genótipo é uma cadeia de inteiros, onde o fenótipo será a matriz correspondente quebrando o vetor em linhas de tamanho 10.

Exemplo da relação genótipo x fenótipo para uma matriz 3×3:

$$\text{cromossomo} = [-1, 1, 1, 1, -1, 1, 1, 1, -1]$$

$$\text{fenótipo} = \begin{bmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{bmatrix}$$

Na listagem em pseudocódigo a seguir temos um SGA (*Simple Genetic Algorithm*), usando *crossover* em ponto único e mutação simples com seleção por roleta viciada.

A função *RAND()* gera um número real aleatório com distribuição $\sim U[0, 1)$ e a função *RANDI(*limite*)* gera um número inteiro aleatório com distribuição $\sim U[0, \text{limite}]$.

```
algoritmo "Genetic Algorithm"
var
  tamPopulacao, tamCromossomo, qtdGeracoes, geracao: inteiro
  linhas, colunas: inteiro
  // cada linha da matriz representa um individuo
  populacao: vetor[1..30,1..100] de inteiro
  // vetor contendo nova populacao
  novaPopulacao: vetor[1..30, 1..100] de inteiro
  avaliacao: vetor[1..30] de inteiro
  cromossomo, gene: inteiro
  probMutacao, probCross: real
  somaAvaliacao: inteiro
  melhor: vetor[1..100] de inteiro
  melhorAvaliacao: inteiro
  cromo1, cromo2: inteiro
  i, j: inteiro

// funcoes auxiliares
// seleciona um individuo baseado na avaliacao
// roleta viciada
funcao seleciona(): inteiro
var
  cromo, sorteio, soma: inteiro
inicio:
  // sorteia numero
  sorteio <- RANDI(somaAvaliacao)
  cromo <- 1
  soma <- avaliacao[cromo]
  enquanto soma < sorteio faca
    cromo <- cromo + 1
    soma <- soma + avaliacao[cromo]
  fimenquanto
  retorne cromo
fimfuncao

// calcula a avaliacao do cromossomo
funcao calculaAvaliacao(cromo: inteiro): inteiro
var
  i, j, soma, maximoNeg: inteiro
inicio
  soma <- 0
  para i de 1 ate linhas faca
    para j de 1 ate colunas faca
      // soma elo com proximo vizinho
      se i < linhas - 1 entao
        soma <- soma + populacao[cromo, colunas * i + j] * populacao[cromo,
    colunas * (i+1) + j]
      fimse
      // soma elo com vizinho de baixo
      se j < colunas - 1 entao
        soma <- soma + populacao[cromo, colunas * i + j] * populacao[cromo,
    colunas * i + j+1]
      fimse
    fimpara
  fimpara
  // calcula qual o valor maximo negativo
  maximoNeg <- 2 * linhas * colunas - linhas - colunas
  // dessa forma a avaliacao e sempre positiva para uso na selecao
  soma <- soma + maximoNeg
  retorne soma
fimfuncao

// executa o crossover de um ponto entre os pais (gera um filho)
procedimento crossover(pai1: inteiro, pai2: inteiro, filho: inteiro)
var
  ponto, i: inteiro
inicio:
```

```
66    ponto <- RANDI(tamCromossomo) + 1
67
68    se RAND() > probCross
69      // se nao deve fazer cross coloca no fim
70      // assim so copia o cromossomo
71      ponto = limite
72    fimse
73
74    se RAND() < 0.5 entao
75      // comeca pelo pai1
76      para i de 1 ate ponto faca
77        novaPopulacao[filho, i] <- populacao[pai1, i]
78      fimpara
79      para i de ponto + 1 ate tamCromossomo faca
80        novaPopulacao[filho, i] <- populacao[pai2, i]
81      fimpara
82    senao
83      // comeca pelo pai2
84      para i de 1 ate ponto faca
85      novaPopulacao[filho, i] <- populacao[pai2, i]
86      fimpara
87      para i de ponto + 1 ate tamCromossomo faca
88      novaPopulacao[filho, i] <- populacao[pai1, i]
89      fimpara
90    fimse
91  fimprocedimento
92
93  // faz mutacao no cromossomo com a prob definida
94  procedimento mutacao(filho: inteiro)
95  var
96    i: inteiro
97  inicio
98    para i de 1 ate tamCromossomo faca
99      se RAND() < probMutacao entao
100       // se deve fazer a mutacao no gene sorteia um novo
101       se RAND() < 0.5 entao
102         novaPopulacao[filho, i] <- 1
103       senao
104         novaPopulacao[filho, i] <- -1
105       fimse
106     fimse
107   fimpara
108 fimprocedimento
109
110 // copia cromossomo da populacao para o melhor
111 procedimento copiaMelhor(cromo: inteiro, cromoAval: inteiro)
112 var
113   i: inteiro
114 inicio:
115   para i de 1 ate tamCromossomo faca
116     melhor[i] <- populacao[cromo, i]
117   fimpara
118   melhorAvaliacao <- cromoAval
119 fimprocedimento
120
121 // inicio da rotina principal
122 inicio
123   tamPopulacao <- 30
124   tamCromossomo <- 100
125   linhas <- 10      // qtd de linhas na matriz
126   colunas <- 10     // qtd de colunas na matriz
127   qtdGeracoes <- 200    // qts geracoes serao testadas
128
129   probCross <- 0.80
130   probMutacao <- 0.02
131
132   somaAvaliacao <- 0
133   melhorAvaliacao <- 0
```

```
134
135   //inicia a populacao
136   para cromossomo de 1 ate tamPopulacao faca
137     para gene de 1 ate tamCromossomo faca
138       // atribui valor randomico 1 ou -1
139       se RAND() < 0.5 entao
140         populacao[cromossomo, gene] <- 1
141       senao
142         populacao[cromossomo, gene] <- -1
143       fimse
144     fimpara
145     avaliacao[cromossomo] <- calculaAvaliacao(cromossomo)
146     somaAvaliacao <- somaAvaliacao + avaliacao[cromossomo]
147     se avaliacao[cromossomo] > melhorAvaliacao entao
148       copiaMelhor(cromossomo, avaliacao[cromossomo])
149     fimse
150   fimpara
151
152   // roda as geracoes
153   para geracao de 1 ate qtdGeracoes faca
154     para cromossomo de 1 ate tamPopulacao faca
155       cromo1 <- seleciona()
156       cromo2 <- seleciona()
157       // crossover
158       crossover(cromo1, cromo2, cromossomo)
159       // mutacao
160       mutacao(cromossomo)
161     fimpara
162
163     // copia a nova populacao
164     somaAvaliaca <- 0
165     para cromossomo de 1 ate tamPopulacao faca
166       // faz a copia de cada individuo
167       para gene de 1 ate tamCromossomo faca
168         populacao[cromossomo, gene] <- novaPopulacao[cromossomo, gene]
169       fimpara
170       avaliacao[cromossomo] <- calculaAvaliacao(cromossomo)
171       somaAvaliacao <- somaAvaliacao + avaliacao[cromossomo]
172       se avaliacao[cromossomo] > melhorAvaliacao entao
173         copiaMelhor(cromossomo, avaliacao[cromossomo])
174       fimse
175     fimpara
176     //repete processo por n geracoes
177   fimpara
178
179   //imprime o melhor cromossomo
180   escreval ("Melhor cromossomo")
181   para i de 1 ate linhas
182     para j de 1 ate colunas
183       escreva ("| ", melhor[i * colunas + j], " |")
184     fimpara
185     // proxima linha
186     escreval ("")
187   fimpara
188   escreval ("Avaliacao: ", melhorAvaliacao)
189 fimalgoritmo
```