```
% Cortador de caños mediante Algoritmos Genéticos: (rev. 20-04-08)
% ================================================

% Importaremos un archivo de texto con los datos separados por " ; " que en
% su primer fila contiene la longitud de los listones comprados, en nuestro
% caso Lmax.
% Las resultantes filas (no se sabe cuantas pueden ser) tienen dos numeros
% enteros, el primero es la cantidad de caños cortados necesarios de la
% longitud es especificada por el segundo numero.
% El nombre del archivo de entrada siempre sera: agXX.txt  (donde XX es un
% numero, por ej. ag01.txt)
% El archivo de salida a generar debera ser: agXXgdg.txt  (siendo GDG las
% iniciales del alumno).
% Este archivo de salida informara la Cantidad de tiras utilizadas y la
% suma total de los desperdicios.
%-------------------------------------------------------------------------
clear;
clc;
disp('Este es un programa para optimizar la compra de caños.')
disp('Por favor, verifique que el archivo con los datos se encuentre en el directorio de trabajo.')
disp(' ')
String=char(input ('Ingrese el nombre del archivo con datos entre apostrofes (comillas simples) = '));
disp(' ')
if (size(String)~=[1 8])
    disp('El nombre del archivo no es correcto.')
    beep;                                   %sonido de llamado de atencion
end
Archivo=dlmread(String,';');

%Comenzamos a acomodar los parametros como nos resulta util
Lmax=Archivo(1,1);
Longitudes=Archivo(2:end,2);
Cantidades=Archivo(2:end,1);

CantDeTirasTotales=sum(Cantidades);     %suma los elementos del vector

% Que hicimos hasta ahora?
% Importamos los datos del archivo y nos quedamos con:
%       Lmax:longitud estandar de las tiras de caños comprados
%       Longitudes:vector con las longitudes de los cortes solicitados
%       Cantidades:vector con las cantidades de los listones pedidos
%       CantDeTirasTotales:sumatoria de Cantidades

% Una vez hecho esto podemos liberar de la memoria los datos que no
% utilizaremos mas como ser la matriz archivo, esto se deberia hacer
% siempre que sea posible con todas las variables pero lo omitire para no
% dificultar la comprension del programa.  :-)

clear Archivo;


% Constantes Importantes:
```

```
 53 % ======================
 54 disp(' ')
 55 disp(' ')
 56 disp('Ahora podremos experimentar modificando la estructura ')
 57 disp('de funcionamiento de nuestro algoritmo:')
 58 disp(' ')
 59 P=input ('Ingrese la cantidad de Individuos de la Poblacion de posibles
soluciones (minimo 50)=')
 60 disp(' ')
 61 CantItera=input ('Ingrese la cantidad de Iteraciones o Repeticiones del
algoritmo (minimo 30)=')
 62 Iteracion=[1:CantItera];
 63
 64 m=0;                %mutaciones iniciales realizadas por iteracion
 65
 66
 67 % Generamos el vector "Largos" de [CantDeTirasTotales x 1]
 68
 69 Largos=zeros([CantDeTirasTotales 1]);      %inicializo un vector con cero
 70
 71 j=1;
 72 for i=1:CantDeTirasTotales
 73     contador=1;
 74     %carga del vector "Largos"
 75     while ((j<=CantDeTirasTotales)&(contador<=Cantidades(i)))
 76         Largos(j,1)=Longitudes(i);
 77         j=j+1;
 78         contador=contador+1;
 79     end
 80 end
 81 % Largos:longitudes de tiras cortadas de cada codificacion binaria de tira
 82
 83 LargosTot=sum(Largos);
 84
 85 % Generamos la matriz "Datos" de [CantDeTirasTotales x P]
 86 % INICIO DEL PRE PROSESAMIENTO IMPONIENDO LA PEOR CONDICION PARA LA
 87 % CANTIDAD DE TIRAS A EMPLEAR (CantDeTirasTotales)
 88
 89 Datos=randint(CantDeTirasTotales,P,[1,CantDeTirasTotales]);
 90 %generamos la poblacion inicial de soluciones, de manera RANDOM (Aleatoria)
 91
 92
 93 % ANALIZANDO EL PROGRAMA:
 94 % Por el momento veremos que en la matriz "Datos" nos quedaron varios
 95 % elementos sin usar (por ej. que el numero 13 de codificación binaria
 96 % nunca apareció, mientas que otras codificaciones aparecieron repetidas
 97 % veces), esto significa que esa codificacion binaria de tira no se empleo
 98 % en la solucion. Las que si aparecen, es lo que se aprovecho del liston
 99 % número tal (contenido de Datos[i][n]), de long Largos[i].
100 %
101 % Para buscar una solucion aceptable para nuestro problema, debemos evaluar
102 % la aptitud de cada individuo (solucion) de la poblacion. Para cuantificar
103 % la aptitud (fitness), emplearemos la siguiente funcion de evaluacion:
104 %
```

```
105 %    Aptitud = Cantidad de tiras empleadas + Sumatoria de desperdicios
106 %
107 %
108 % Nota: casualmente tenemos una variable TirasNoEmpleadas, ohhh!
109 % Otra mas... hablar de desperdicios es = que MaterialRestantePorTira
110
111 % Siempre procuraremos minimizar la funcion de evaluacion, ya cuanto mas
112 % bajo sea el valor numérico de nuestra función, menos cantidad de tiras
113 % empleamos, y menor es el desperdicio producido.
114
115 % La matriz "Valor" contiene la sumatoria de los largos de caño empleado
116 % por las soluciones.
117
118 % Ahora calcularemos el desperdicio que resulta de cada liston utilizado
119 % haciendo:
120 %
121 %  MaterialRestantePorTira = Desp = Lmax - Valor   (en las tiras empleadas)
122 %
123 % Notando que se pueden presentar las siguientes 3 alternativas:
124 % a)Desp>0   %desperdicio longitudinal real de la tira (situacion habitual)
125 % b)Desp<0   %situacion irrealizable, ya que gastamos mas longitud del
126              %liston que Lmax (penalizamos la aptitud a dicha solucion)
127 % c)Desp=0   %situacion optima, mayor aprovechamiento del liston (premiamos
128              %la aptitud de estas soluciones)
129
130 IteraActual=0;
131
132 for iteraciones=1:CantItera    %====== INICIO DEL LOOP ALGORITMICO ========
133 %cantidad de ciclos que se repite el algoritmo (minimo 30)
134
135 IteraActual=IteraActual+1;
136
137 % Debemos inicializar cada vez las 2 matrices para que levanten los datos
138 % de la nueva poblacion.
139 clear Valor;
140 Valor=zeros(CantDeTirasTotales,P);
141 % cada posicion sera el numero de tira al que pertenece el corte
142 % (total de tiras = CantDeTirasTotales)
143 % En la matriz "Valor" haremos la sumatoria de los largos de las soluciones
144
145 for j=1:P
146     for i=1:CantDeTirasTotales
147           Valor(Datos(i,j),j)=Valor(Datos(i,j),j)+Largos(i);
148           %sumatoria // Fila:Datos(i,j) Columna:j
149     end
150 end
151
152 clear TirasNoEmpleadas;
153 TirasNoEmpleadas=Valor<=0;
154 MaterialRestantePorTira=ones(CantDeTirasTotales,P)*Lmax.*(~TirasNoEmpleadas);
155 MaterialRestantePorTira=MaterialRestantePorTira-Valor;     %incrementamos
156
157 % Hasta aqui tenemos:
158 % - Datos: la poblacion actual
```

```
159 % - Largos: las longitudes de tiras cortadas de cada codificacion binaria
160 %           de tira
161 % - Valor: tengo la sumatoria en longitud, de los listones usados por ese
162 %          codigo binario
163 % - MaterialRestantePorTira: longitud sobrante de esa codificacion binaria
164 %                            de tira
165 % -------------------------------------------------------------------------
166 % Notar que si el material sobrante por tira es cero, puede darse porque no
167 % se empleo dicha codificacion, o porque se realizo una combinacion
168 % perfecta de cortes, de ser esta segunda premiaremos la aptitud de dicha
169 % combinacion, y en el caso de que sea negativa, debemos penalizar ya que
170 % estamos requiriendo un liston mas largo del que disponemos para hacer la
171 % combinacion de cortes.
172
173
174 % Ahora obtendremos la APTITUD de cada solucion, osea de la poblacion:
175 %    Aptitud = Cantidad de tiras empleadas + Sumatoria de desperdicios
176
177 CantidadDeTirasEmpleadas=CantDeTirasTotales*ones(1,P)-sum(TirasNoEmpleadas);
178
179 % La Sumatoria de desperdicios es mas compleja porque debemos analizar los
180 % 3 casos posibles, para arrojar un valor numerico para el valor de Aptitud
181
182 % a) Desp > 0
183 % desperdicio longitudinal real de la tira (situacion habitual)
184 DespA=sum(MaterialRestantePorTira.*(MaterialRestantePorTira>0));
185 DespA=DespA/Lmax;       %normalizamos con respecto a Lmax       MINIMIZAR
186
187 % b) Desp < 0
188 % situacion irrealizable, ya que gastamos mas longitud del liston que Lmax
189 %         (penalizamos la aptitud a dicha solucion) x10
190 DespB=sum(MaterialRestantePorTira.*(MaterialRestantePorTira<0));
191 DespB=DespB/Lmax;       %normalizamos con respecto a Lmax       MINIMIZAR
192
193 % c) Desp = 0
194 % situacion óptima, mayor aprovechamiento del liston (premiamos) x5
195 DespC=sum(MaterialRestantePorTira==0);                          %MAXIMIZAR
196
197 % Obtenemos la Sumatoria de desperdicios, recordar que menor valor numérico
198 % es mas Apto !!!      -->     MINIMIZAR
199 Desp=1*DespA+10*abs(DespB)+5*(CantDeTirasTotales-DespC);
200
201
202 % Contemplamos que la long de nuestra solucion sea la necesaria
203 SumValor=sum(Valor);
204
205 Aptitud=CantidadDeTirasEmpleadas+Desp;     %aptitud del conj. de soluciones
206
207 AptiProm=(sum(Aptitud))/P;           %aptitud promedio de esta la población
208
209 Evaluacion_AptiProm(iteraciones)=AptiProm;       %luego con un PLOT veremos
210 % la evolucion del conjunto de soluciones en funcion de las iteraciones
211
212
```

```
213 % ============================================================================
214 % Criterios de:   - Apareamiento (reproduccion binaria, 2 padres generan 2
215 % =============     hijos y 2 poco aptos fallecen)
216 %                 - Mutación (anomalía genética en la copia de genes a la
217 %                   siguiente generación de individuos)
218 %
219 % Ahora definiremos en base a la "aptitud" que posee cada individuo de la
220 % población, con respecto a la aptitud promedio que posee dicha población,
221 % quienes son las mejores soluciones (valor numérico de la aptitud baja) y
222 % quienes son las peores soluciones.
223 % Las mejores soluciones se Aparearan, para intercambiar material genético
224 % para intentar obtener una mejor solución, las peores soluciones tendrán
225 % la posibilidad de mutar para ver si se pueden encaminar hacia una mejor
226 % solución, mientras tanto la media de las soluciones pasara directamente
227 % a la siguiente generación.
228
229 % El umbral de apareo debe ser menor que la media, ya que menor aptitud es
230 % mejor. Estoy empleando un criterio de selección muy simple: EL ELITISTA.
231 if IteraActual<=(CantItera/2)
232     UmbralApareo=0.97*AptiProm;
233 else
234     UmbralApareo=0.98*AptiProm;
235 end
236 % Notar que soy mas ELITISTA en la selección de individuos, al comienzo de
237 % la evolución de mi población.
238
239 ProbaMuta=randsrc(1,1,[0 1;.99 .01]);
240 % Debemos ser cautelosos con la pobabilidad de Mutacion < 1/100
241
242 clear Indices_Para_Aparear;
243 Indices_Para_Aparear=find(Aptitud<=UmbralApareo);
244
245 % El apareamiento para realizar el intercambio genético, se producira a
246 % través de una mascara binaria generada de manera aleatoria, de esta
247 % manera podemos independizarnos la posición en la que se encuentran los
248 % genes, ya que si tenemos una secuencia de genes 1-2-3-4-5-6-7-8, podemos
249 % hacer que sólo se intercambien los genes 1 y 3, sin necesidad de
250 % intercambiar el gen 2 que se encuentra entre éstos.
251
252 Mask=randsrc(CantDeTirasTotales,1,[0:1]);
253 % generamos un vector mascara binaria, de manera RANDOM
254
255 Not_Mask=~Mask;                    %obtenemos una mascara complementaria
256
257 [a1,a2]=size(Indices_Para_Aparear);
258 % averiguamos cuantos individuos tenemos para aparear (a2)
259
260 Evaluacion_Apareos(iteraciones)=a2;
261 % luego con un HIST veremos la cantidad de apareamientos del conjunto
262 % de soluciones en funcion de las iteraciones
263
264
265 % Apareamiento: (si no hay al menos 2 en condiciones, no se puede aparear)
266 % =============
```

```
267
268 Datos=Datos([1:CantDeTirasTotales],[1:P]);  %PARCHE
269
270 if a2>1
271     ExtractA=Datos(:,[Indices_Para_Aparear]);
272     %extraemos diversas columnas de la matriz datos
273     ExtractB=ExtractA;
274     for a1=1:(a2)
275         %para trabajar con las mascaras
276         ExtractA(:,a1)=Not_Mask.*ExtractA(:,a1);
277         ExtractB(:,a1)=Mask.*ExtractB(:,a1);
278     end
279     acumulador=ExtractB(:,1);
280     %guardo la primer columna, el primer medio individuo
281     %desplazamos para hacer el intercambio de genes
282     for a1=1:(a2-1)
283         ExtractB(:,a1)=ExtractB(:,a1+1);
284     end
285     ExtractB(:,a2)=acumulador;
286     ExtractA=ExtractA+ExtractB;
287     %tenemos los nuevos individuos de la poblacion, que reemplazaran a
288     %alguno de las peores soluciones (mayor aptitud numerica en nuestro caso)
289
290
291     %Busqueda de los REEMPLAZOS:
292
293     Aptitud_Decreciente=sort(Aptitud*-1);
294     %multiplicamos por -1 porque SORT ordena en forma creciente
295     Aptitud_Decreciente=Aptitud_Decreciente*-1;
296
297     %en este instante dentro de "Aptitud_Decreciente", en sus primeras "a2"
298     %posiciones, tenemos las aptitudes de los peores individuos de la
299     %población (valor numérico mas grande), y guardaremos en
300     % "Indice_Para_Reemplazar" el indice que se corresponde en la matriz
301     % "Datos"
302
303     Flag=find(Aptitud>Aptitud_Decreciente(a2));
304     Flag1=find(Aptitud==Aptitud_Decreciente(a2));
305
306     [c1,c2]=size(Flag);
307
308     while c2<a2             %nos aseguramos de tener la cantidad necesaria
309         Flag=[Flag Flag1(c1)];
310         c1=c1+1;
311         c2=c2+1;
312     end
313
314     Indice_Para_Reemplazar=Flag(1:a2);
315     %tenemos los indices de los individuos que reemplazaremos
316
317
318     %REEMPLAZOS:
319     for i=1:a2
320         Datos(:,(Indice_Para_Reemplazar(i)))=ExtractA(:,i);
```

```
321     end                                 %lista la nueva generacion !!!
322
323
324 end
325
326         %-------------------------------------------------
327
328 if (1==ProbaMuta)
329     % con ROUND redondeo al entero mas cercano y con RANDN obtengo números
330     % entre -1 y 1
331     Datos(randsrc(1,1,[1:256]),randsrc(1,1,[1:P]))=randsrc(1,1,[1:256]);
332         m=1;                    %solo una mutación
333 end
334
335 Evaluacion_Mutaciones(iteraciones)=m;
336 % luego con un HIST veremos la cantidad de mutaciones del conjunto
337 % de soluciones en funcion de las iteraciones
338 m=0;
339
340
341 end     %======== FIN DEL LOOP ALGORITMICO ==========
342
343
344 %Gráficas de los indicadores de funcionamiento del algoritmo:
345 %-----------------------------------------------------------
346
347 subplot (3,1,1),plot(Iteracion,Evaluacion_AptiProm)
348 xlabel('Numero de iteracion del Algoritmo Genetico')
349 ylabel('Aptitud promedio de la poblacion')
350 title('Analisis de Convergencia (disminuir la aptitud = OK)')
351 grid on
352
353 subplot (3,1,3),bar(Iteracion,Evaluacion_Apareos)
354 xlim([0 Iteracion(end)])
355 xlabel('Numero de iteracion del Algoritmo Genetico')
356 ylabel('Individuos que Aparean')
357 grid on
358
359 subplot (3,1,2),bar(Iteracion,Evaluacion_Mutaciones)
360 xlim([0 Iteracion(end)])
361 xlabel('Numero de iteracion del Algoritmo Genetico')
362 ylabel('Mutaciones')
363 grid on
364
365
366 %Respuesta solicitada por el problema:
367 %-------------------------------------
368
369 clear i j;
370 % Liberamos de la memoria las variables "j" e "i" utilizada anteriormente,
371 % esto se deberia haber realizado siempre al dejar de utilizar cada
372 % variable que no necesitaramos mas, pero no se hizo para no complicar el
373 % entendimiento del codigo. Otro recordatorio.  :-)
374
```

```
375 j=min(Aptitud);
376 i=find(Aptitud==j);
377 %indice de la mejor solución de la ultima población(menor aptitud numérica)
378
379 String=[String(1:4) 'G' 'D' 'G' String(5:8)];
380 %formamos el nombre del archivo de salida con mis iniciales
381
382 diary(String);                              %Generamos el archivo de salida
383
384 diary on                                    %Inicio del archivo
385
386 disp('Desechos acumulados de caños (DespA) :')
387 disp(DespA(1,i(1)))
388
389 disp('Desechos irrealizables de caños (DespB) :')
390 disp(DespB(1,i(1)))
391
392 disp('Combinaciones Perfectas (DespC) :')
393 disp(DespC(1,i(1)))
394
395 disp('Tiras de caños comprados empleados:')
396 disp(CantidadDeTirasEmpleadas(1,i(1)))
397
398 diary off                                   %Fin del archivo
399
400 disp('NOTA: miranfo la matriz Datos, subindicada por i(1), junto con el vector
Largos, podemos ver como realizar los cortes.')
```