## Contents



```
function [ Cromosomas, Matriz_objetivos, orden_poblacion ] = NSGAII( poblacion,generaciones )
```

```
%=========================================================================
```

## NSGAII Introducción

```
%=========================================================================
% El NSGAII (Non-dominated Sorting Genetic Algorithm II) es un modelo de
% computación evolutiva multiobjetivo desarrollado por K Deb en 2012. El
% código a continuación se basa de manera estructural en el propuesto por
% Kanpur Genetic % Algorithm Labarotary y kindly, (información sobre el
% algoritmo original % puede ser consultada en:
% http://www.iitk.ac.in/kangal/) y estructurado por Aravind Seshadri,
% Copyright (c) 2009. Y presenta como diferencia que la cantidad de
% variables de decisión y sus límites inferiores y superiores no son
% ingresados manualmente, en vez de ello, la función denominada
% 'funcion_objetivo' es un archivo guia (script) editable para cambiar la
% configuración del modelo (en conjunto con los respectivos archivos .data)
% además, teniendo en cuenta la estructura del modelo, la generación de
% cromosomas y la descodificación se genera de % manera independiente, es
% decir, se crean variables donde se almacena los resultados de las
% funciones fitness. Por otra parte la funcion 'operador_genetico' se
% desarrolla con un tipo de cruce y mutación diferente al propuesto por
% Kanpur Genetic y, para ello, se proponen otras funciones para generar las
% nueva poblaciones.

% A continuación se presenta el acuerdo de propiedad intelectual original:
% This functions is based on evolutionary algorithm for finding the optimal
% solution for multiple objective i.e. pareto front for the objectives.
% Initially enter only the population size and the stoping criteria or
% the total number of generations after which the algorithm will
% automatically stopped.
%
% You will be asked to enter the number of objective functions, the number
% of decision variables and the range space for the decision variables.
% Also you will have to define your own objective funciton by editing the
% evaluate_objective() function. A sample objective function is described
% in evaluate_objective.m. Kindly make sure that the objective function
% which you define match the number of objectives that you have entered as
% well as the number of decision variables that you have entered. The
% decision variable space is continuous for this function, but the
% objective space may or may not be continuous.
%
% Original algorithm NSGA-II was developed by researchers in Kanpur Genetic
% Algorithm Labarotary and kindly visit their website for more information
% http://www.iitk.ac.in/kangal/


%  Copyright (c) 2009, Aravind Seshadri
%  All rights reserved.
%
%  Redistribution and use in source and binary forms, with or without
%  modification, are permitted provided that the following conditions are
%  met:
%
%     * Redistributions of source code must retain the above copyright
%       notice, this list of conditions and the following disclaimer.
%     * Redistributions in binary form must reproduce the above copyright
%       notice, this list of conditions and the following disclaimer in
%       the documentation and/or other materials provided with the distribution
```

```
%
%  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
%  AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
%  IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
%  ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
%  LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
%  CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
%  SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
%  INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
%  CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
%  ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
%  POSSIBILITY OF SUCH DAMAGE.
%=========================================================================
% la función NSGAII( poblacion,generaciones ) cuenta con dos parámetros de
% entrada. poblacion y generaciones, donde poblacion indica la cantidad de
% individuos (cromosomas) permitidos pro cada generacion. La variable
% generaciones indica el número de veces que evolucionará la poblacion. Las
% vaiables de salida son: Cromosomas (conjunto de individos pertenecientes
% a la última generación), Matriz_objetivos (valor de las funciones
% fitness) y orden_poblacion (ubicación de cada solución en el frente de
% pareto y su distancia de apilameinto).
%=========================================================================
```

```
%=========================================================================
% Verificación de parámetros del modelo
%=========================================================================
% A continuación se verifica que los parámetros de entrada cumplan con
% requisitos como tipo de variable y su magnitud.
if nargin < 2
    error('NSGA-II: Por favor, Ingrese el tamaño de la población y el número de generaciones como argumentos de entrada.');
end
% Tipo de argumentos (numéricos)
if isnumeric(poblacion) == 0 || isnumeric(generaciones) == 0
    error('Ambos argumentos de entrada Pobalación (pop) y Número de generaciones (gen) deben ser de tipo entero');
end
% El tamaño mínimo de la población debe ser de 20 individuos
if poblacion < 20
    error('El tamaño mínimo de la población debe ser de 20 individuos');
end
% La cantidad mínima de generaciones es de 5
if generaciones < 5
    error('La cantidad mínima de generaciones es de 5');
end

if isinteger(poblacion) == 0 || isinteger(generaciones)==0
    fprintf('Los valores deben ser enteros y, por tanto, para evitar errores son redondeados al entero inferior')
    % Verificar que las entradas son enteras, a partir de un redondeo
    poblacion = round(poblacion);
    generaciones = round(generaciones);
end
```

```
Error using NSGAII (line 87)
NSGA-II: Por favor, Ingrese el tamaño de la población y el número de generaciones como argumentos de entrada.
```

```
%=========================================================================
% Carga de la función objetivo
%=========================================================================
% La función denominada "funcion_objetivo", es un guión editable en el cual
% se definen las dos funciones objetivo bajo estudio, así mismo, la
% cantidad de variables y parámetros del modelo como: Número de periodos,
% cantidad de Lotes, Cantidad de productos, número de variables de
% decisión, cantidad de restricciones, etc. Teniendo en cuenta la
% estructura del modelo, los parámetros son obtenidos a partir de de unos
% datos almacenados en un archivo .data Para cambiar características como
```

```
% el rendimiento agrícola poe lote, el tamaño de cada lote, etc. Es
% necesario cambiar el conjunto de datos.
%=========================================================================
[ cant_objetivos, cant_variables, ...
    cant_periodos ,cant_productos, cant_lotes, precio_venta, ...
    rendimiento, areas, demanda,familia_botanica,familia_venta,Covkkp] ...
    = funcion_objetivo();
%=========================================================================
```

```
%=========================================================================
% Creación de los cromosomas iniciales
%=========================================================================
% La función denominada "inicializar_cromosomas", tiene como parámetros de
% entrada las características del modelo aproximado (con unidades de tiempo
% basadas en meses en vez de semanas como el modelo exacto) y la cantidad
% de individuos que conforman la población. Esta función carga datos
% relacionados con las características del cromosoma como cantidad de
% periodos y restricciones de siembra y demás, retorna los cromosomas y sus
% respectivas funciones objetivo. Para cambiar el modelo, es necesario
% modificar el archivo .data ya que este indica los periodos de siembra,
% duración del periodo de madurez y otros elementos que caracterizan el
% caso de estudio a trabajar.
%=========================================================================
[ Cromosomas,Matriz_objetivos ] = inicializar_cromosomas(poblacion, ...
    cant_objetivos, cant_periodos, cant_productos, cant_lotes,precio_venta, ...
    rendimiento, areas, demanda,familia_botanica,familia_venta,Covkkp);
%=========================================================================
```

```
%=========================================================================
% Determinar dominancia de la solución inicial
%=========================================================================
% Se ordena la población usando ordenar_poblacion. La función requiere como
% parámetro de entrada las soluciones generadas de manera aleatoria
% (cromosomas), el valor de sus funciones objetivo (funciones fitness o de
% ajuste) y la cantidad de objetivos. La función devuelve dos
% columnas para cada individuo que son el rango y la distancia de
% apilamiento correspondiente a su posición en el frente al que pertenecen.
% En este punto el rango y la distancia de apilación para cada
% cromosoma se almacena en una variable extra denominada orden_poblacion.
%=========================================================================
[ Cromosomas, Matriz_objetivos, orden_poblacion ] = ...
    ordenar_poblacion( Cromosomas, Matriz_objetivos,cant_objetivos );
%=========================================================================
```

```
%=========================================================================
% Comienza la evolución de las generaciones
%=========================================================================
% Se hace un recorrido para todas las generaciones
for gen = 1: generaciones
    %     de manera opcional, se grafican los valores de los individuos según
    %     ambas funciones de ajuste
    scatter(Matriz_objetivos(2,:),Matriz_objetivos(3,:),'filled')
    drawnow
    % Los padres son seleccionados para la reproducción para generar
    % descendencia a partir de un torneo tipo binario basado en en comparar la
    % función fitness (Matriz_objetivo, fila 1), de ahí se generan lo padres
    % para realizar los diversos crices.

    % se crea la cantidad de grupos a enfrentarse
    pool = round(poblacion/2);
    %     se define el tipo de torneo
    torneo = 2;
    %=====================================================================
    %    Selección de los padres por torneo
```

```
%=========================================================================
%     La selección de los padres de hace de manera aleatoria a partir del
%     conjunto total de soluciones (variable denominada 'Cromosomas'), cada
%     par de padres se enfrenta teniendo en cuenta dos parámetros: 1) el
%     valor del frente a cual pertenece cada individuo, seleccionando el
%     individuo de menor frente y, 2) el valor de la distancia de
%     apilamiento, en caso de pertenecer al mismo frente, como criterio de
%     selección está el individuo con mayor valor de distancia.
%=========================================================================
[Cromosomas_padres ,criterio_evaluacion_padres,objetivos_padres]= ...
    seleccion_por_torneo(Cromosomas, Matriz_objetivos, ...
    orden_poblacion, pool, torneo);
%=========================================================================
% Se ajusta la probabilidad de mutación, esto implica que del total de
% modificaciones genéticas un 'probabilidad_mutacion' no se formará
% mediante el cruce de padres, en vez de ello, un segmento de su
% información genética (para este caso la producción en un lote al azar) se
% vovlerá a computar
probabilidad_mutacion=0.10;
%=========================================================================
%    Creación de los hijos mediante los operadores genéticos
%=========================================================================
%     la función 'operador_genetico' crea hijos a partir de los padres
%     seleccionados, para ello existen dos posibildiades: 1) generar dos
%     hijos a partir del cruce -en un único punto- entre dos padres. 2) la
%     generación de 2 hijos a partir de la mutación en un segmento del
%     cromosoma de ambos padres. la función genera la solución y el
%     respectivo valor de las funciones de ajuste:
[Cromosomas_hijos, objetivos_hijos]= operador_genetico( ...
    cant_objetivos, cant_periodos, cant_lotes,Cromosomas_padres, ...
    probabilidad_mutacion,poblacion,cant_productos,Covkkp,...
    precio_venta,rendimiento,areas,demanda,familia_botanica,...
    familia_venta);
%=========================================================================
% Se crea una población intermedia, conformada por las soluciones padres y
% las soluciones hijas
% Soluciones
Cromosomas_intermedios=cat(2,Cromosomas_padres,Cromosomas_hijos);
%     valor de las funciones de ajuste
Matriz_objetivos_intermedio=cat(2,objetivos_padres,objetivos_hijos);
%=========================================================================
% Determinar dominancia de la solución intermedia
%=========================================================================
% Se ordena la población usando ordenar_poblacion. La función requiere como
% parámetro de entrada las soluciones generadas de manera aleatoria
% (cromosomas), el valor de sus funciones objetivo (funciones fitness o de
% ajuste) y la cantidad de objetivos. La función devuelve dos
% columnas para cada individuo que son el rango y la distancia de
% apilamiento correspondiente a su posición en el frente al que pertenecen.
% En este punto el rango y la distancia de apilación para cada
% cromosoma se almacena en una variable extra denominada orden_poblacion.
%=========================================================================
[ Cromosomas_intermedios, Matriz_objetivos_intermedio, ...
    orden_poblacion_intermedio ] = ordenar_poblacion(...
    Cromosomas_intermedios, Matriz_objetivos_intermedio,cant_objetivos );
%=========================================================================
% Se organizan las soluciones intermedias según el frente (de menor a
% mayor) y según su distancia de apilamiento (de mayor a menor para el caso
% de empatar en frente)
% Se crea una variable auxiliar para almacenar el orden
x=orden_poblacion_intermedio';
% Se crea una columna auxiliar con la posición de cada solución
x(:,3)=1:length(x);
% Se ordena la solución según frente (columna 1) y según distancia (columna
% 2)
orden=sortrows(x,[1,2],{'ascend','descend'});
% Se extrae el orden de las funciones y se almacena en la variable
% 'orden'
orden=orden(:,3)';
```

```
    % Se actualiza la población 'Cromosomas' según la población intermedia
    % 'Cromosomas_intermedios' para las posiciones 'orden' hasta un tamaño
    % máximo igual a la cantidad de individuos de la población 'poblacion'
    for ajuste=1:length(orden(1:poblacion))
        Cromosomas(:,(ajuste-1)*cant_lotes+1:(ajuste-1)*cant_lotes+cant_lotes)=...
            Cromosomas_intermedios(:,(orden(ajuste)-1)*cant_lotes+1:(orden(ajuste)-1)*cant_lotes+cant_lotes);
    end
    % Se actualiza el valor de las funciones objetivo de la nueva población
    Matriz_objetivos=Matriz_objetivos_intermedio(:,orden(1:poblacion));
    orden_poblacion=orden_poblacion_intermedio(:,orden(1:poblacion));
end
%=========================================================================
```

```
end
```

*Published with MATLAB® R2017a*