Memoria Práctica 3

GRUPO 14

Jose Manuel Perez Zamorano

Ederson Funes Castillo

## Implementación

El proyecto está estructurado en 3 partes: el modelo, la vista y el controlador.

La vista, llama al controlador para pasarle los parámetros del algoritmo y el controlador crea

un objeto de tipo algoritmo genético, lo ejecuta y le devuelve los resultados a la vista.

* 1. Estructura de paquetes

En los paquetes del controlador y la vista están respectivamente la clase controlador y la clase interfaz.

En el paquete útil están los tipos enumerados, la clase correspondiente a los datos que le pasamos a la interfaz para que forme la gráfica y una clase Funciones que tiene la función para calcular la aptitud del cromosoma.

En el paquete modelo están Cromosoma, problema, los diferentes tipos de selección y la clase algoritmo genético.

Para esta práctica intentamos reutilizar todo lo posible cosas de la anterior, pero la parte del cromosoma al final no la hemos podido generalizar y utilizamos el cromosoma correspondiente a este problema para casi todo.

* 1. Estructura de clases

La estructura del cromosoma ha cambiado, teniendo ahora un objeto tipo Nodo, que es lo que representa al árbol y el genotipo.

El nodo es un objeto cuyos atributos son a su vez otros nodos (hijo izquierdo, hijo derecho, hijo condicional y un nodo que apunta al padre en caso de que no sea la raíz), el terminal y la función, que es de tipo TipoOperacion. También cuenta con funciones para mutar parte del árbol (mutafuncion(), mutaTerminal() y permuta()) y funciones utilizadas para el cruce (terminalRandom() y funcionRandom()).

El TipoOperacion es un enumerado que sirve para distinguir las distintas operaciones que pueden aparecer en un nodo del árbol.

* 1. Cruces

**Cruce Normal**: Cruce en el que se escogen 2 nodos aleatorios de ambos árboles, y se intercambian uno por el otro. La peculiaridad de este cruce es que está diseñado para que no sea posible intercambiar un árbol entero con cualquier otra cosa (Con árbol entero nos referimos al árbol completo de un cromosoma).

**Cruce de Funciones**: Ídem al normal, solo que solo intercambian nodos que sean de tipo función.

**Cruce de Terminales**: Ídem al normal, solo que solo intercambian nodos que sean de tipo terminal.

**Cruce Propio**: Combina los cruces de función y terminal a la vez.

* 1. Mutaciones

**Mutación de Terminal**: Escoge un terminal al azar del árbol y lo sustituye por otros del conjunto de terminales.

**Mutación de Funciones**: Escoge una función al azar del árbol, y solo lo sustituye aquellas que tenga la misma aridad.

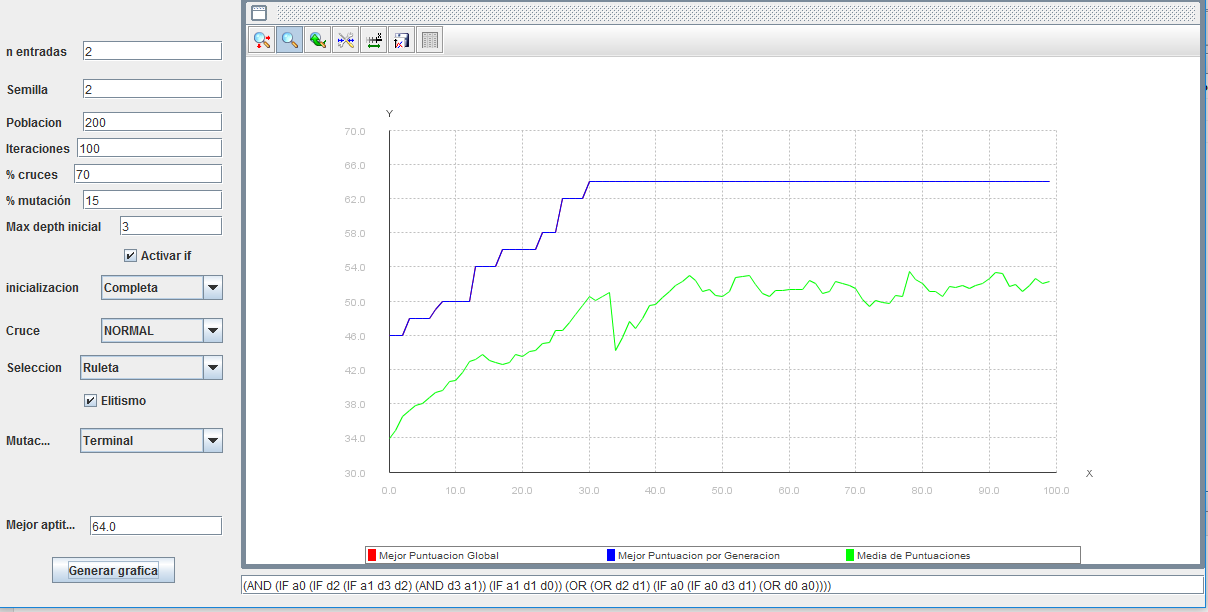
**Permutación**: Escoge una función (OR / AND) al azar del árbol e intercambia los hijos.

1. Graficas
   1. Multiplexor 6 entradas

Tras varias simulaciones, hemos visto que los parámetros que generan mejores resultados son:

* Población: 200
* Iteraciones: 100
* % cruces: 70
* % Mutación: 15
* Max. Profundidad Inicial: 3
* IF Activado
* Inicialización Completa
* Cruce Normal
* Selección por Ruleta
* Elitismo Activado
* Mutación de Terminales

1. GRAFICA 1

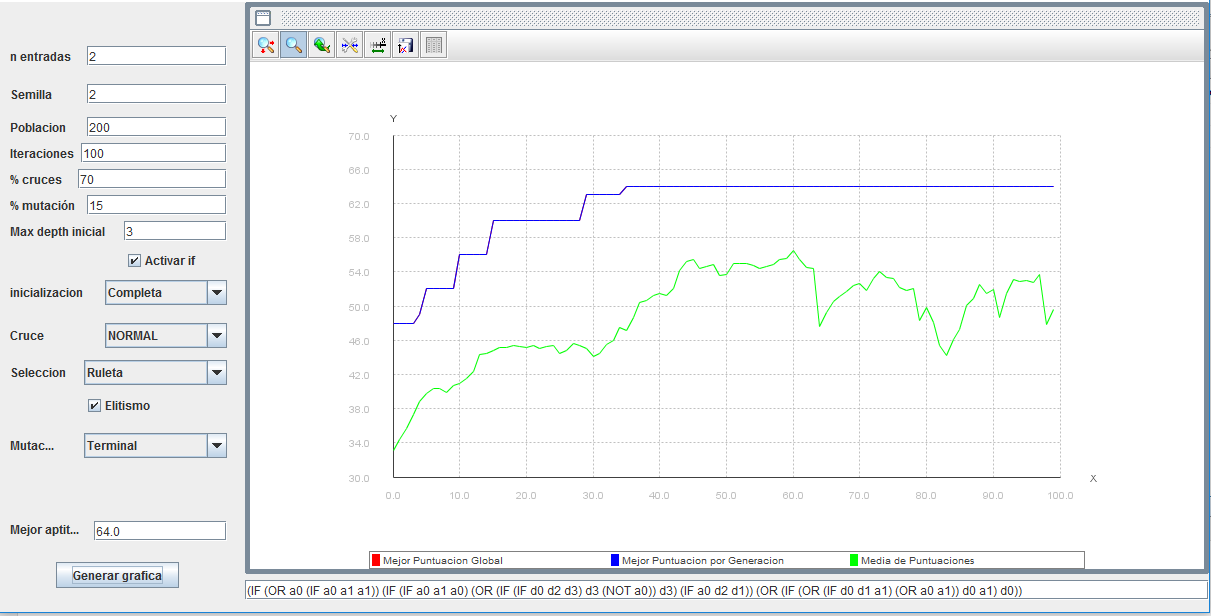


**Fenotipo:**

(AND (IF A0 (IF D2 (IF A1 D3 D2)(AND D3 A1))(IF A1 D1 D0))(OR ( OR D2 D1) (IF A0 ( IF A0 D3 D1) (OR D0 A0))))

**Aptitud**: 64

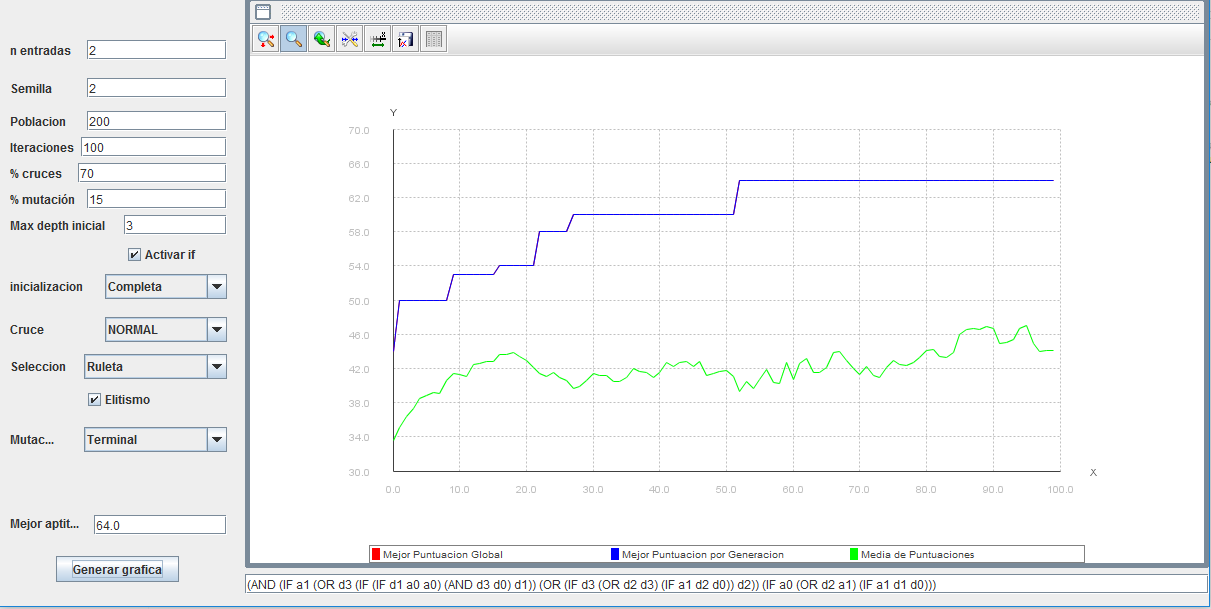
1. GRAFICA 2



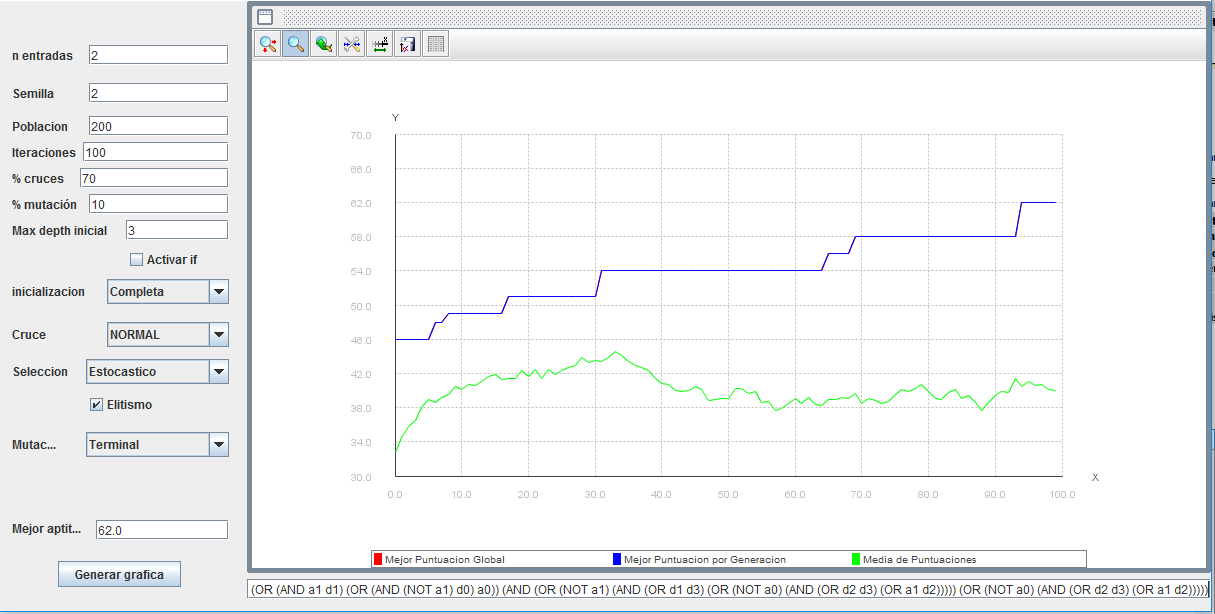
**Fenotipo:** (IF(OR A0 (IF A0 A1 A1))(IF(IF A0 A1 A0)(OR(IF(IF D0 D2 D3) D3 (NOT A0)) D3) (IF A0 D2 D1))(OR(IF(OR(IF( D0 D1 A1)(OR A0 A1)) D0 A1) D0))

**Aptitud**: 64

1. GRAFICA 3 / 4 / 5 (Esta sin IF)







* 1. Multiplexor 11 entradas



**Fenotipo**: (IF (AND a1 a1) (OR (IF (AND (IF d2 d1 a2) d1) (IF (AND a1 d7) (IF d0 d3 d2) d1) a2) (OR (IF (AND a0 d6) (OR d4 d6) a2) d2)) (OR (IF a2 d1 d0) (IF (OR (IF a2 d1 d4) (AND a2 d5)) (AND a0 a0) (AND d5 d4))))

Aptitud: 1600

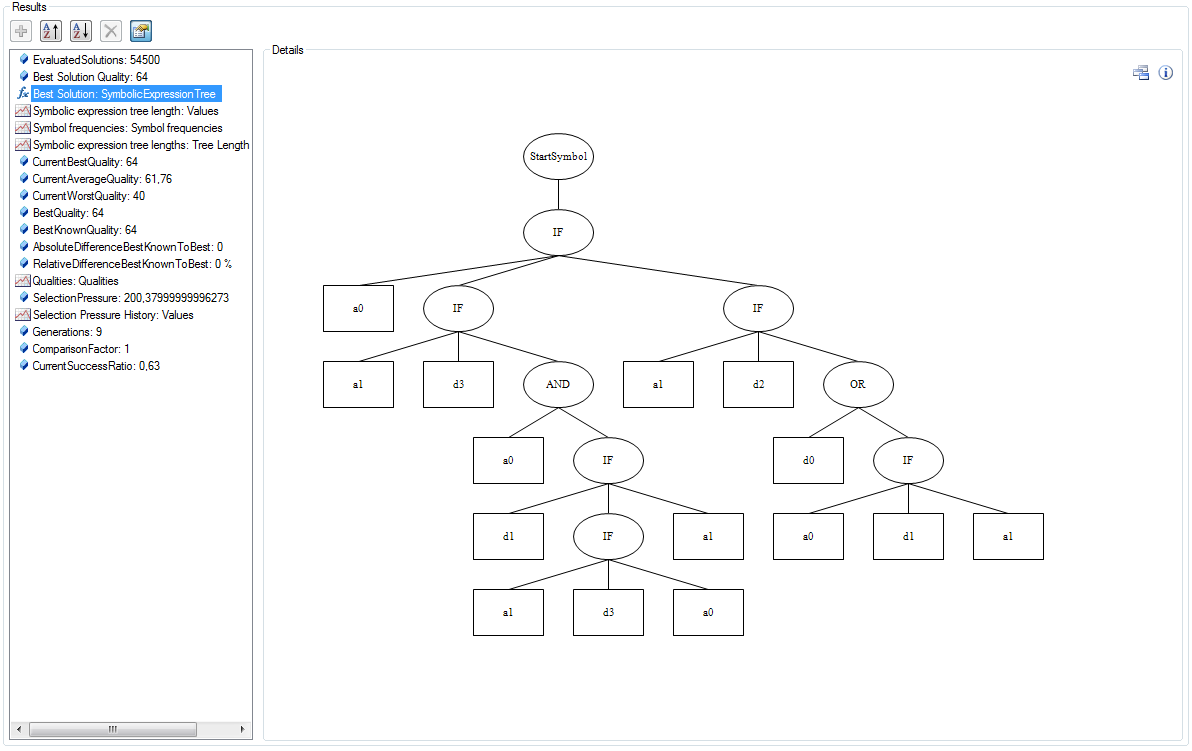


1. Heuristic Lab

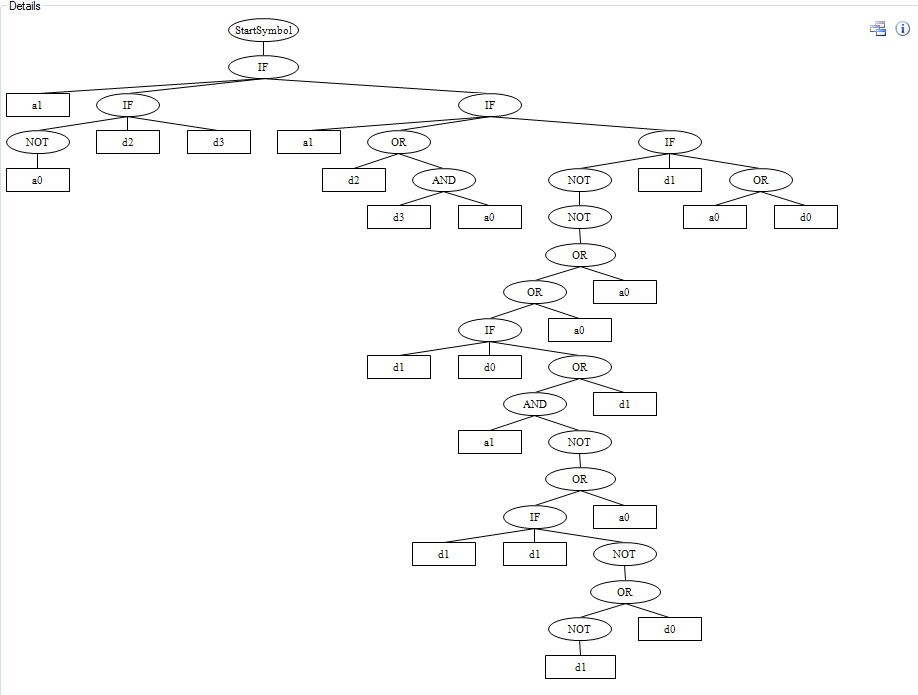
Realizamos distintas pruebas en Heuristic Lab para el problema del Multiplexor (6 y 11).

Para las pruebas del Multiplexor de 6 entradas no cambiamos ningún parámetro de los que venían por defecto (probabilidad de mutación, tamaño de la población inicial, elite, etc.) ya que con estos parámetros ya se conseguían individuos de aptitud 64 (máxima aptitud posible).

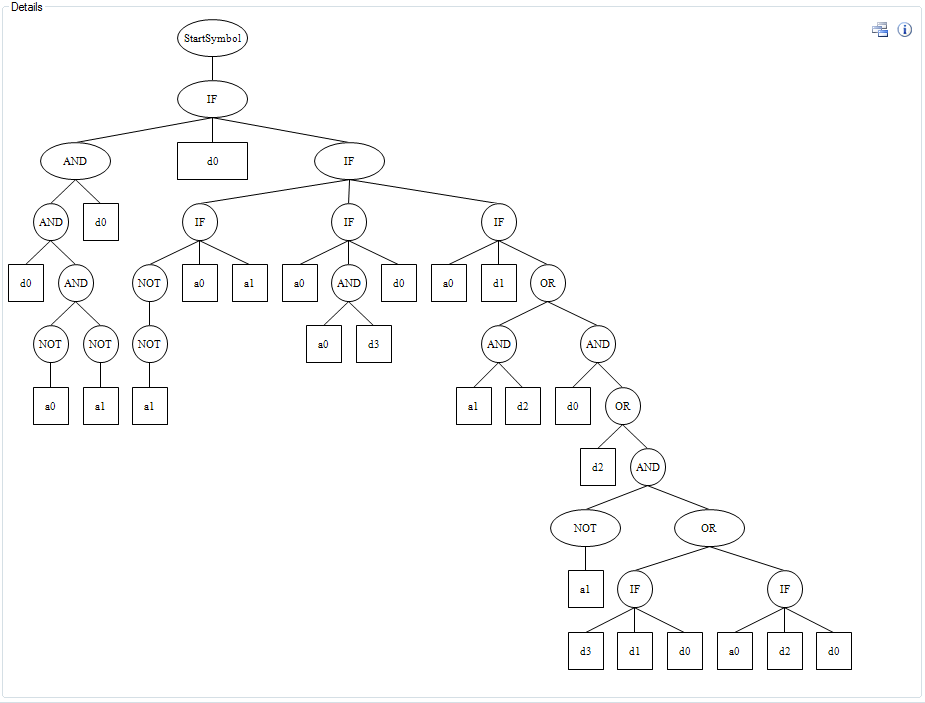
* Primera Simulación:



* 2º Simulación

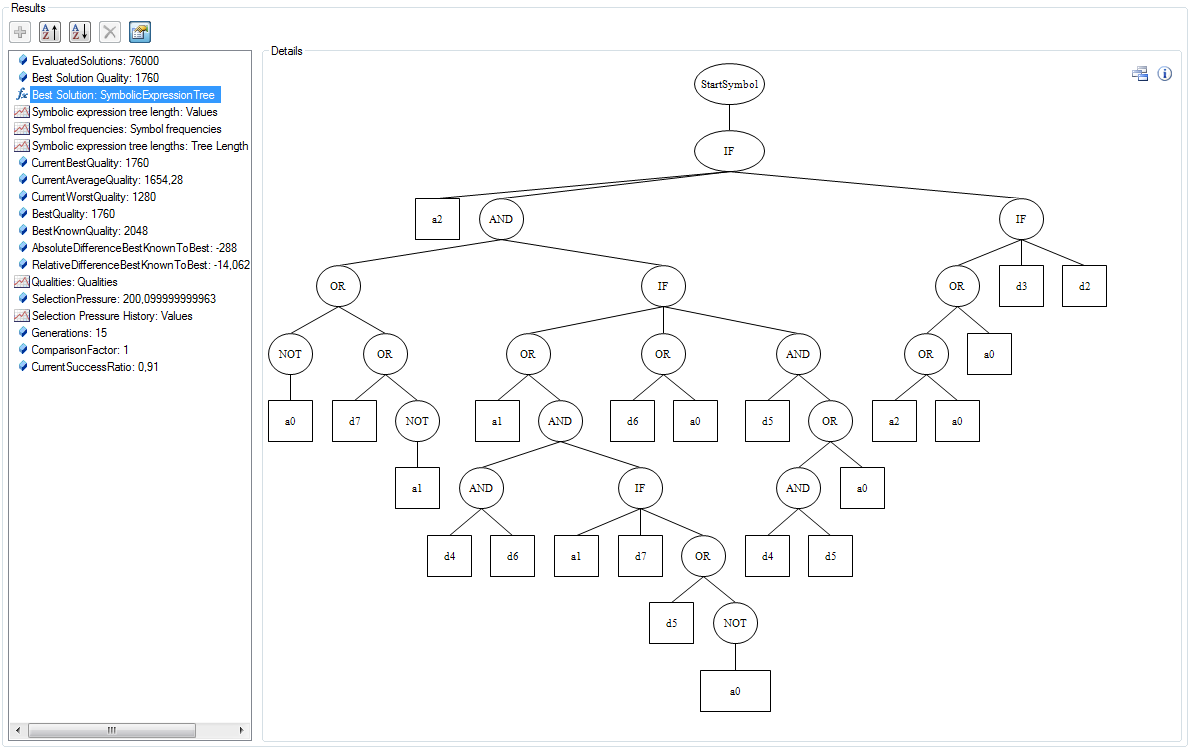


* 3º Simulación

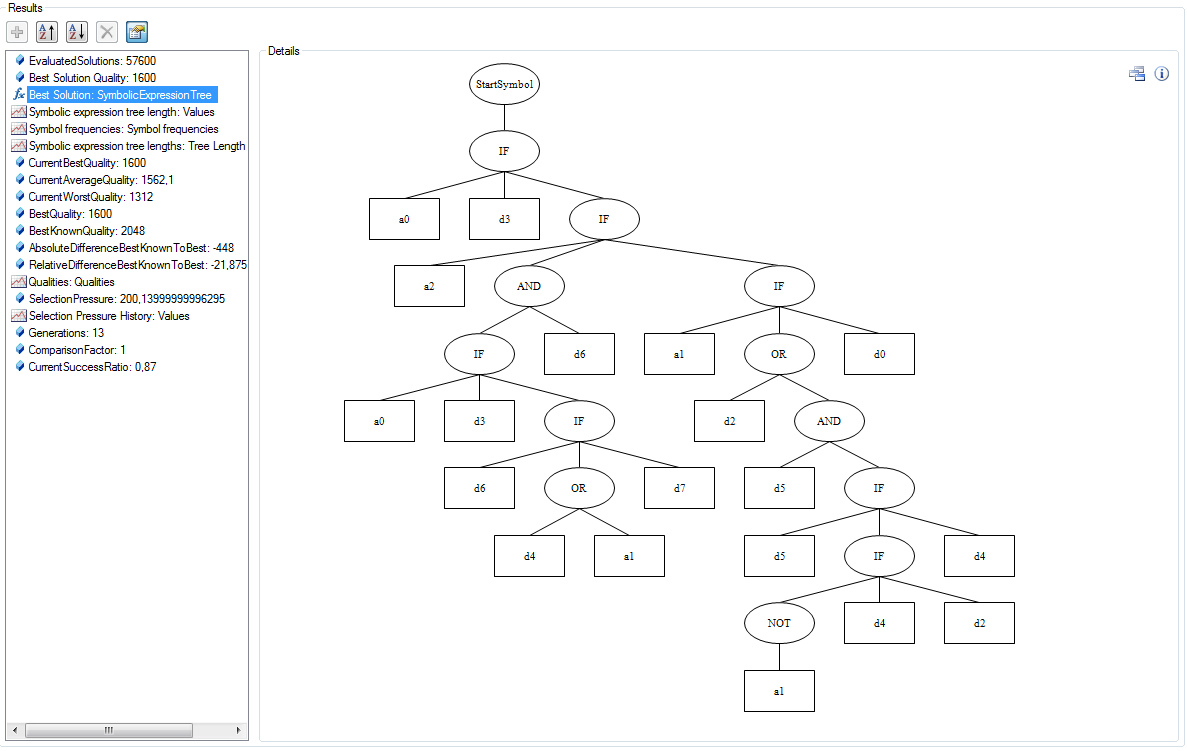


Para el Multiplexor de 11 entradas decidimos hacer cambios en los parámetros para ver las distintas aptitudes que se podían obtener

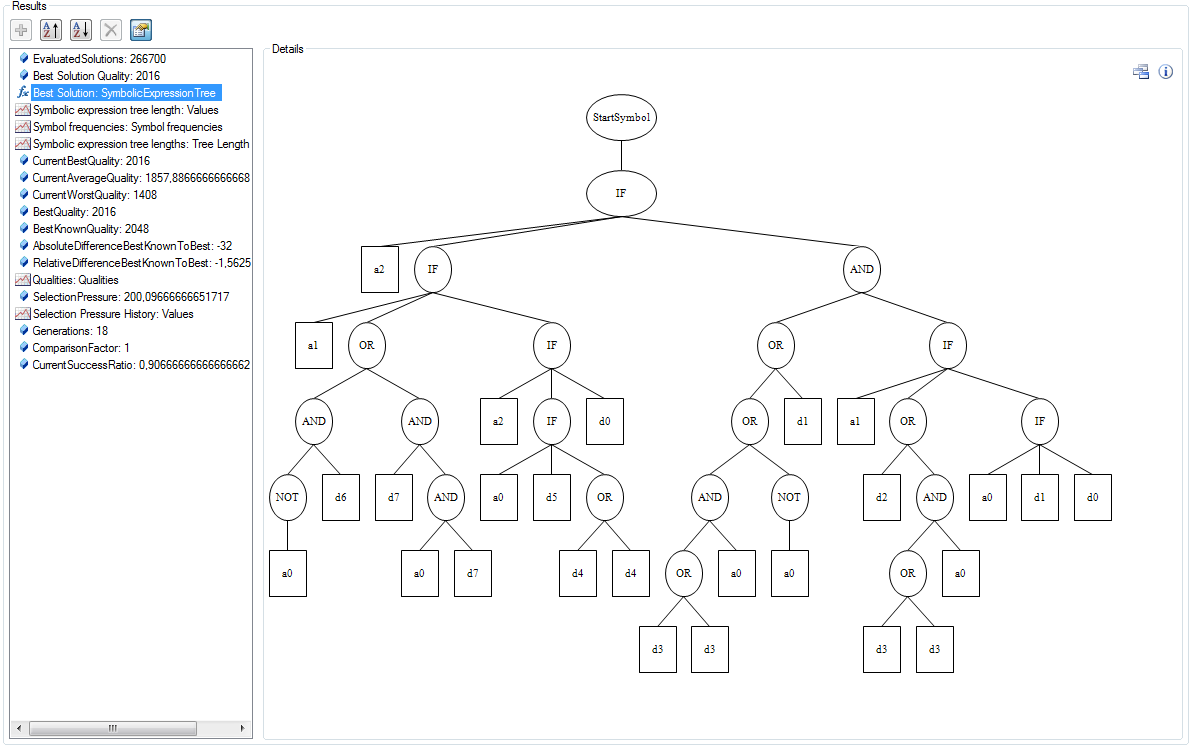
* 1º Simulación (Parámetros Predefinidos) – Mejor Fenotipo = 1760



* 2º Simulación (%Mutación al 50%) – Mejor Fenotipo = 1600



* 3º Simulación (%Mutación al 50% y 300 Individuos Iniciales) – Mejor Fenotipo = 2016



3.3 Bloating

Debido al cruce entre cromosomas, algunos árboles terminaban con una profundidad mucho mayor a la de antes del cruce. Esto creaba individuos gigantes que no necesariamente mejoraban su aptitud. Por ello decidimos incluir un pequeño control del bloating siguiendo el método Tarpeian.

El método Tarpeian consiste en encontrar aquellos cromosomas cuyos arboles tenga una profundidad que supere la media de profundidades de la población, y según una probabilidad 1/n, estos individuos son “eliminados” (se les asigna una aptitud muy baja, por ejemplo 0). Tras esto solo cabe esperar a que el método de selección escogido se encargue de eliminarlos definitivamente de la población, en nuestro caso lo que hacemos es cambiar el individuo por uno nuevo generado aleatoriamente con la misma profundidad máxima que la población inicial.