PEV: Práctica 3

Jorge Vieira Luna

José Miguel Tajuelo Garrigós

29-5-2017

Representación

Hemos creado una población de individuos cuyo cromosoma contiene un único gen árbol. Dicho árbol ha sido realizado mediante una implementación propia, empleando los nodos hoja (Terminales) para representar las diferentes entradas, y los no terminales para representar las distintas puertas lógicas.

Tenemos implementadas las puertas AND, OR, IF, NOT y hemos incluido como extra la XOR, pudiéndose activar y desactivar al gusto cada una de ellas desde la interfaz, para comprobar qué combinación da mejores resultados.

Así mismo, hemos generalizado el problema para que el usuario pueda elegir el tamaño del multiplexor en función del número de entradas de datos del mismo (2,4, 8, 16…).

También hemos generalizado el uso de las puertas lógicas para que puedan tener un número de entradas entre 2 y el definido por el usuario. Las que lo permiten: (AND, OR, XOR, DCe3, DCe4). Las puertas DCe3 y DCe4 (propias) evalúan condicionales consecutivamente de 3 y 4 entradas respectivamente.

En cuanto a la generación de población inicial, tenemos los tres métodos vistos en clase: Creciente, completa y ramped & half.

Como la práctica exige, también permitimos especificar la profundidad máxima del árbol.

Funcionalidades incluidas en la práctica:

Selección:

- Selección por ruleta
- Selección por torneo determinista
- Selección por torneo no determinista
- Selección por método estocástico universal

Cruce:

- Cruce por permutación (no forzada)

Mutación:

- Mutación terminal
- Mutación funcional
- Mutación de árbol
- Mutación múltiple (propia)
- OPM (propia)

Uso de élite

Contractividad:

- No
- Actualizando población
- Sin actualizar población

Irradiar periodico

Generación de población inicial:

- Creciente
- Completa
- Ramped&Half

AntiBloating:

- None
- Naive (funciona como naive y para decidir torneos según la k especificada)
- Tarpeian
- Covariant Parsimony Preasure (CPP o “bien fundamentada”)
- Advanced (propio)

Función de aptitud

La función que utilizamos para evaluar nuestros individuos es la siguiente:
Si el multiplexor tiene 4 entradas de datos y dos de selección, seis en total, tenemos 2^6 = 64 posibilidades que evaluar. Para cada una de ellas comprobaremos que la salida del multiplexor es la adecuada. Por ejemplo, si las dos entradas de selección son 10, en la salida del multiplexor ha de estar lo mismo que en la entrada de datos 3.

Mejoras del algoritmo evolutivo:

Irradiar periódico: Método especial que muta agresivamente de forma periódica la población al detectar estancamiento)
Contractividad: No considerar generaciones transcurridas las que no producen una mejora en el fitness del mejor absoluto (manteniendo o sin mantener la población generada en cada generación).

Operadores propios:

Mutación múltiple: Realiza aleatoriamente una de las mutaciones estándar (terminal, funcional y de árbol).
OPM: Genera n individuos por cada operador de mutación estándar (terminal, funcional y de árbol) y se queda con el mejor.

Mecanismos de control de bloating:

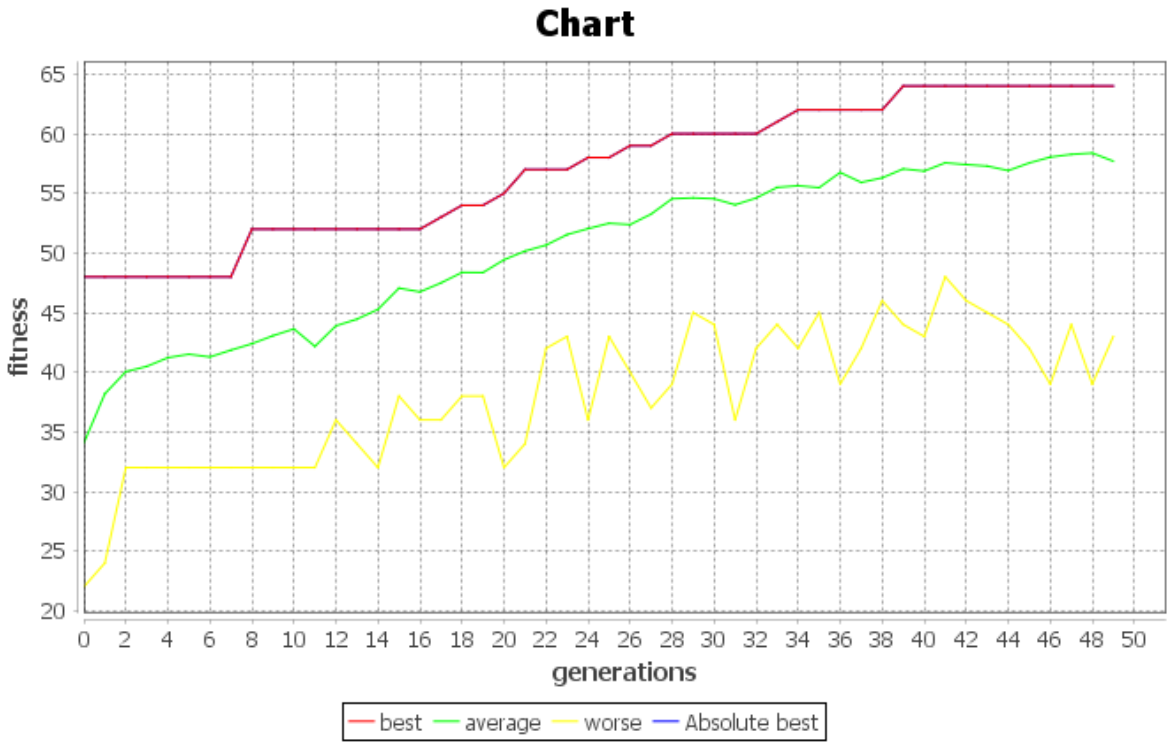
Se han implementado todos los mecanismos vistos en clase, así como uno propio, penalizando el fitness alcanzado según el ratio de superación del límite de nodos.

Advanced (control de bloating propio): Calcula un ratio matemáticamente (utilizando potencias y divisiones), aplicando una penalización a cada individuo relativa a en qué medida exceda un máximo de nodos (determinado por el máximo número de nodos asignable a un árbol con la profundidad máxima).

Parte 1 y 2: Multiplexores de 4 y 8 entradas de datos

Las cuatro ejecuciones mostradas a continuación se producen de forma rápida, produciéndose una convergencia en el mejor fitness del problema o, en su defecto, en un resultado muy próximo a este. En ninguna de las ejecuciones hemos empleado control de bloating, ya que, si bien reduce ampliamente la extensión de los códigos resultantes, producen una convergencia prematura en problemas de gran complejidad.

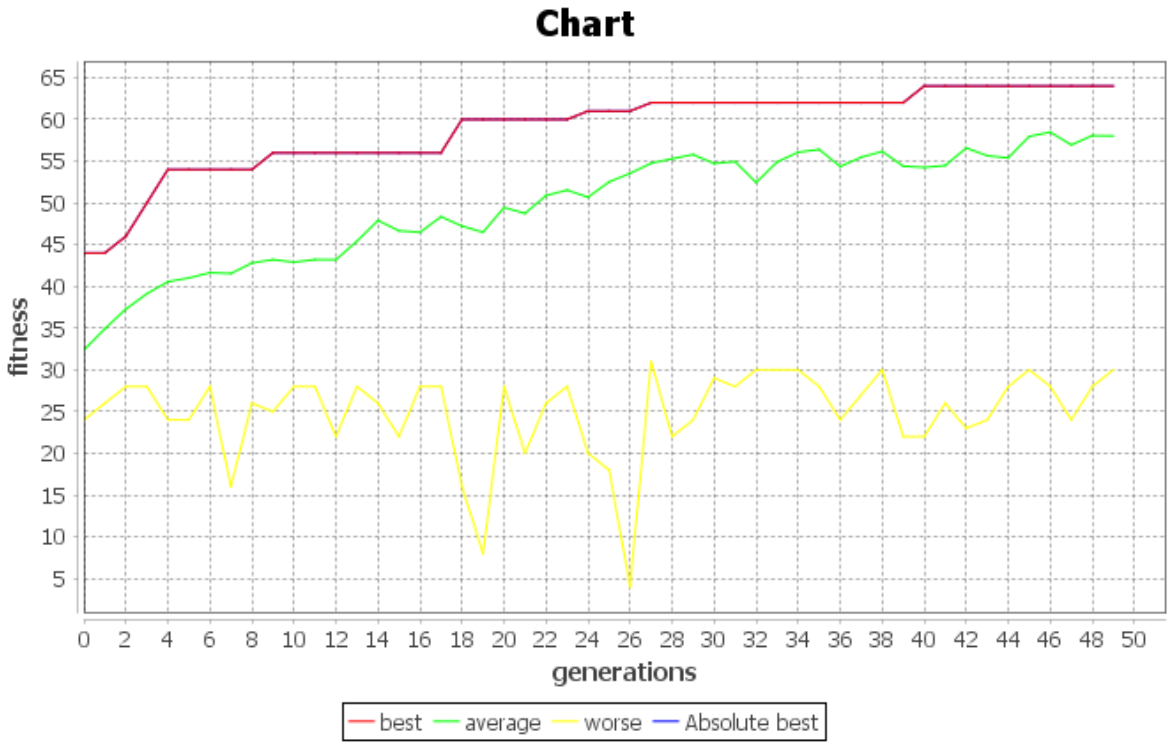
Multiplexor de 4 entradas con puerta IF



entradas máximas por puerta: 3
profundidad máxima: 4
inicialización: completa
puertas lógicas: AND OR NOT IF
antibloating: no
población: 100
generaciones: 50
selección: Torneo determinista (3)
cruce: Permutación no forzada (60%)
mutación: terminal (25%)
elite: 5%
contractividad: no

IF(IF(OR(NOT(0),AND(1,2)),OR(NOT(1),NOT(0)),AND(IF(2,3,5))),AND(OR(OR(3,3),NOT(1)),OR(AND(0,4),OR(1,2))),IF(OR(AND(0,4),OR(0,2)),IF(NOT(1),AND(0,4),AND(0,5));AND(IF(1,2,2)OR(3,2))))

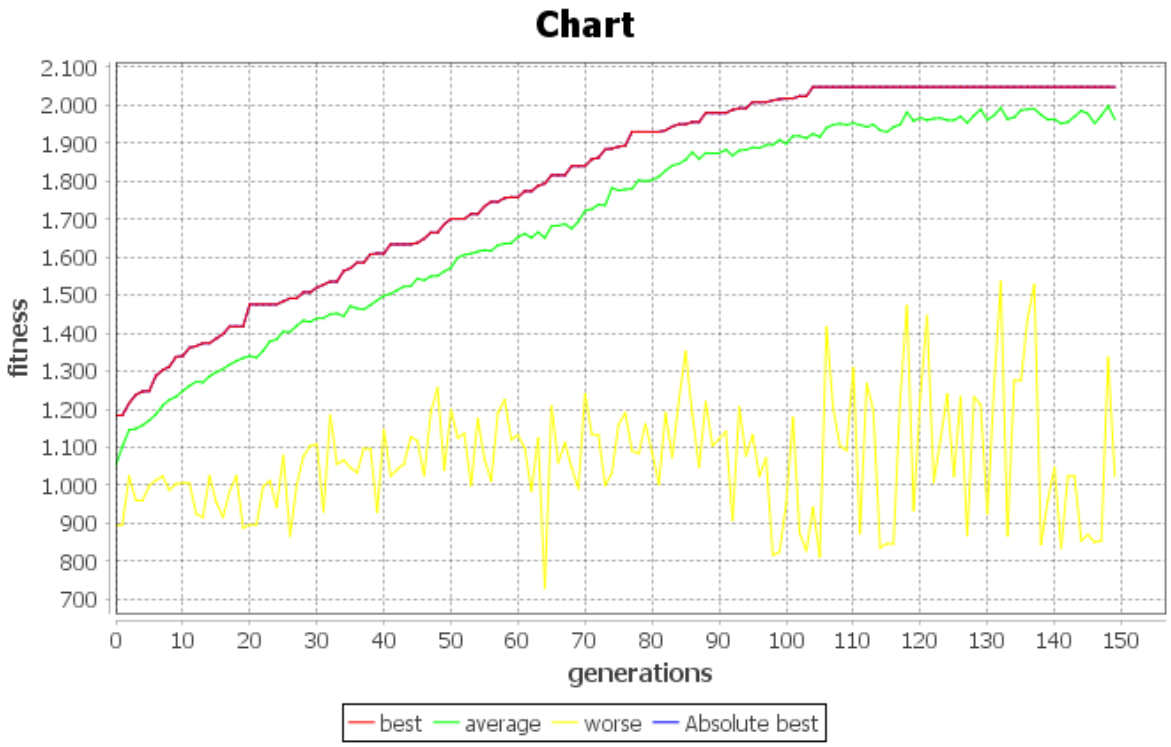
Multiplexor de 4 entradas sin puerta IF



entradas máximas por puerta: 3
profundidad máxima: 4
inicialización: completa
puertas lógicas: AND OR NOT XOR
antibloating: no
población: 100
generaciones: 50
selección: Torneo determinista (3)
cruce: Permutación no forzada (60%)
mutación: terminal (25%)
elite: 5%
contractividad: no

XOR(AND(NOT(NOT(0)),AND(NOT(1),NOT(4),XOR(XOR(NOT(3),AND(2,0,1),XOR(4,1,4)),XOR(NOT(OR(4,1,4)),AND(3,4,0),XOR(XOR(NOT(3),AND(3,0,1),XOR(4,3,1)),XOR(NOT(2),AND(1,5,0),OR(1,3,3)),NOT(AND(4,4,4)))),NOT(AND(4,4,4)))),XOR(NOT(0),XOR(AND(2,2,0),AND(1,2,4),AND(NOT(NOT(3)),AND(NOT(5),NOT(0),XOR(AND(5,1,5),AND(1,5,4),AND(NOT(NOT(0)),AND(2,0,1),XOR(NOT(5),XOR(AND(5,2,5),AND(1,2,4),AND(4,4,1)),AND(2,0,4))))),XOR(NOT(AND(4,1,1)),XOR(AND(5,2,4),AND(1,2,4),AND(5,4,0)),XOR(4,0,3)))),AND(1,3,3))),XOR(XOR(NOT(3),AND(3,0,1),XOR(4,1,1)),XOR(NOT(2),AND(1,5,0),OR(1,3,3)),NOT(AND(4,4,4))),OR(NOT(OR(4,1,4)),NOT(OR(0,1,1)),NOT(NOT(2))))

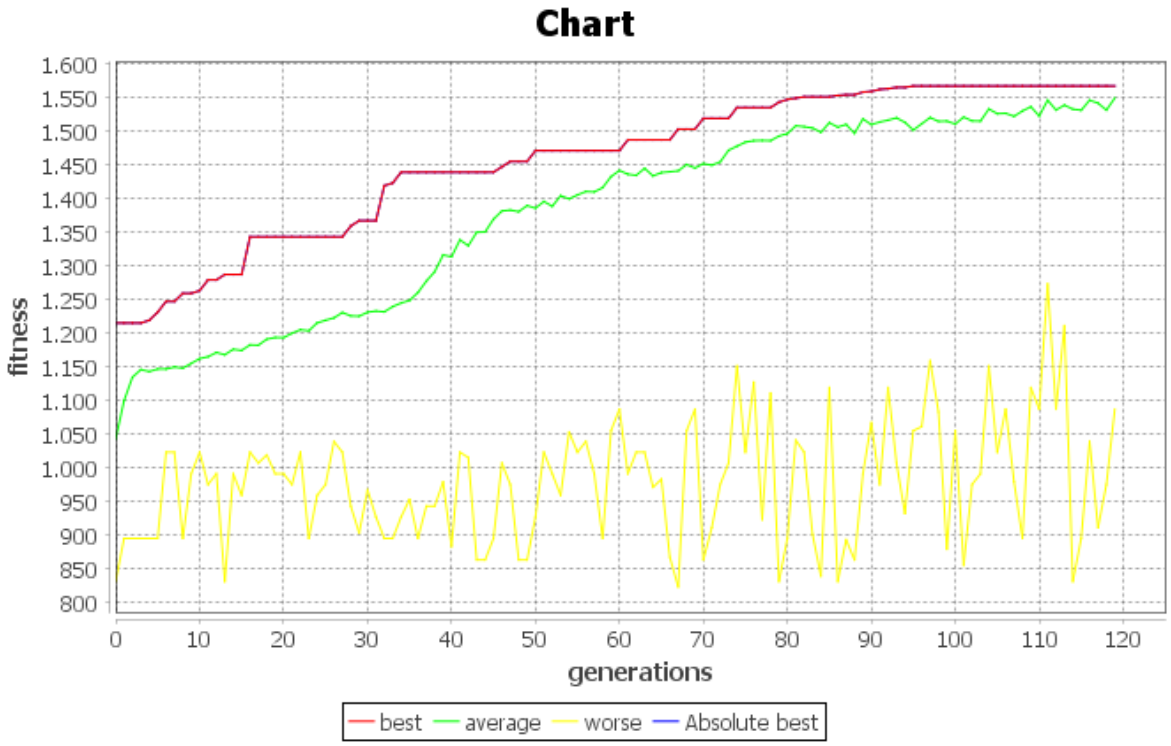
Multiplexor de 8 entradas con puerta IF



entradas máximas por puerta: 3
profundidad máxima: 8
inicialización: ramped&half
puertas lógicas: AND OR NOT IF XOR
antibloating: no
población: 100
generaciones: 50
selección: Torneo determinista (3)
cruce: Permutación no forzada (90%)
mutación: función (25%)
elite: 5%
contractividad: no

IF(NOT(NOT(NOT(IF(4,1,1)))),XOR(IF(IF(IF(10,6,1),OR(5,5,6),OR(5,7,5)),OR(2,0,7),IF(AND(8,1,3),OR(0,0,8),OR(0,7,2))),IF(IF(IF(IF(OR(4,10,3),IF(2,4,3),IF(IF(9,7,3),NOT(9),NOT(4))),IF(NOT(1),OR(1,4,3),XOR(6,5,10)),AND(IF(5,1,5),AND(2,6,1),IF(10,5,10))),AND(0,5,2),NOT(0)),AND(XOR(5,8,0),OR(5,9,10),IF(3,4,0)),IF(NOT(2),OR(0,5,3),IF(0,8,4))),OR(2,1,7)),IF(NOT(OR(NOT(9),IF(4,2,2),IF(0,2,9))),OR(AND(AND(4,10,10),IF(1,1,2),IF(2,9,2)),OR(NOT(0),IF(2,4,3),OR(2,1,7)),NOT(IF(6,0,0))),AND(OR(AND(IF(IF(AND(4,10,2),IF(2,4,3),OR(IF(0,2,3),NOT(9),NOT(2))),IF(NOT(1),XOR(3,4,5),OR(6,5,10)),XOR(IF(9,7,3),AND(8,1,1),IF(10,5,0))),XOR(0,5,2),NOT(0)),XOR(6,2,6),AND(2,7,8)),OR(AND(6,7,10),NOT(2),IF(0,10,6)),OR(XOR(XOR(XOR(OR(0,6,1),AND(9,5,6),OR(5,7,1)),OR(2,1,7),OR(OR(8,1,9),XOR(0,0,8),OR(0,7,2))),IF(OR(IF(AND(IF(4,10,2),IF(2,4,2),XOR(XOR(AND(5,2,10),XOR(1,1,0),IF(2,9,2)),OR(NOT(0),OR(6,10,3),XOR(2,1,7)),NOT(0))),IF(NOT(1),XOR(1,4,2),XOR(10,5,10)),IF(IF(0,1,0),XOR(8,6,1),OR(10,5,10))),AND(0,5,2),NOT(0)),AND(XOR(5,0,0),OR(5,9,10),OR(OR(6,8,0),OR(5,9,10),IF(3,4,0))),IF(NOT(2),IF(0,5,3),OR(8,8,5))),OR(3,1,7)),OR(7,5,10),AND(8,1,10)))))

Multiplexor de 8 entradas sin puerta IF



entradas máximas por puerta: 3
profundidad máxima: 15
inicialización: ramped&half
puertas lógicas: AND OR NOT XOR
antibloating: no
población: 200
generaciones: 150
selección: Torneo determinista (3)
cruce: Permutación no forzada (90%)
mutación: múltiple (40%)
elite: 5%
contractividad: no

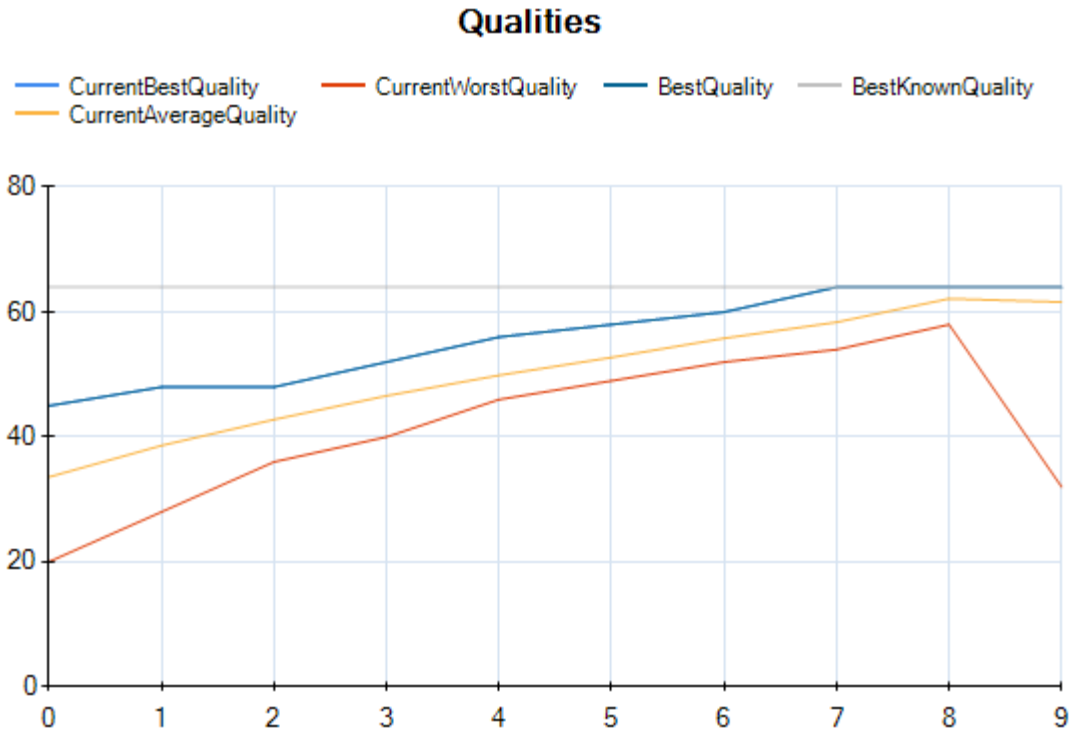
OR(AND(OR(10,AND(XOR(OR(AND(OR(AND(AND(AND(XOR(XOR(0,7),10),5),10),OR(4,1)),0),8),XOR(OR(OR(OR(10,6),AND(2,OR(OR(OR(0,0),AND(AND(OR(AND(AND(OR(XOR(2,4),OR(AND(XOR(AND(XOR(5,9),3),2),OR(10,8)),8)),XOR(5,3)),OR(AND(9,AND(2,10)),XOR(XOR(1,3),XOR(AND(0,AND(XOR(3,3),7)),4)))),XOR(5,7)),AND(OR(AND(XOR(OR(XOR(2,4),OR(AND(OR(AND(XOR(5,7),3),7),OR(10,8)),1)),XOR(1,3)),OR(XOR(3,AND(2,10)),XOR(XOR(1,3),XOR(AND(10,OR(9,AND(1,1))),4)))),AND(OR(AND(XOR(10,2),AND(0,10)),5),0)),OR(OR(2,6),AND(2,OR(OR(OR(0,7),XOR(0,AND(AND(2,7),10))),3))))),7)),0))),10),2)),1),XOR(1,0))),2),XOR(AND(0,3),XOR(AND(0,AND(OR(AND(XOR(OR(AND(OR(AND(XOR(AND(9,AND(10,10)),2),OR(4,0)),8),8),XOR(OR(XOR(5,7),XOR(1,7)),2)),1),XOR(1,0)),OR(AND(AND(OR(XOR(XOR(5,7),8),0),9),XOR(OR(9,AND(1,8)),AND(OR(AND(2,3),AND(OR(OR(9,4),3),2)),7))),AND(XOR(0,7),1))),7)),3)))

Parte 3: Uso de HeuristicLab para resolver los multiplexores de 4 y 8 entradas

Parámetros por defecto de HeuristicLab (para ambos multiplexores):

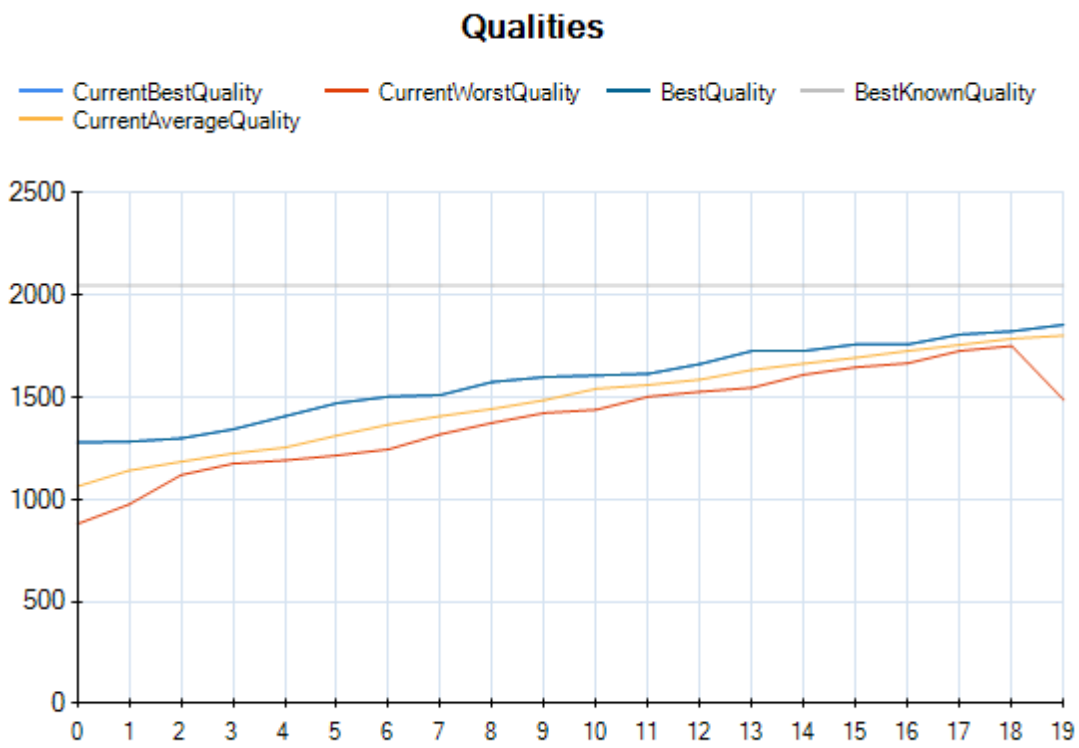
- Tamaño de población: 100
- Generaciones: 50
- Crossover: Permutación de árboles (90%)
- Mutación: Mezcla de reemplazar rama, borrar rama, mutar nodo función, mutar nodo entrada y mutar todos los nodos del árbol, al 25%
- Elite: 1 individuo

Multiplexor de 2 selectores y 4 entradas de datos



Con 65700 soluciones evaluadas. De la alta cantidad de soluciones evaluadas se deduce que el planteamiento de HeuristicLab es considerar la iteración transcurrida únicamente si se produce mejora (Para población 100 generaciones 50, como ellos, nosotros evaluaríamos 500 soluciones, es decir, 131 veces menos). Por eso también, nuestra evaluación es mucho más rápida.

Multiplexor de 3 selectores y 8 entradas de datos



Con 78300 soluciones evaluadas. Pese a la enorme cantidad de soluciones probadas, se estanca en un mejor absoluto de 1856, ya que la presión selectiva llega a 200% (La ejecución finaliza cuando esto ocurre). Nosotros llegamos en nuestro ejemplo al óptimo, 2048, en aproximadamente 110 generaciones. Son más que las que aparecen en la gráfica de HeuristicLab, pero nosotros para 110 generaciones con población 200 evaluamos 110*200 = 22000 individuos, casi 4 veces menos.