Construcción automática de reglas de inversión utilizando programación genética

Mario Alberto Llorente Lopez

PFC – Ingeniería en Informática

24 de Enero de 2012

Contenido

- Introducción
  - Objetivos
  - Planificación
- Algoritmos genéticos y mercados financieros
  - Algoritmos genéticos
  - Mercados financieros
  - Algoritmo desarrollado
- Desarrollo de la aplicación
  - Análisis y especificación
  - Diseño
  - Implementación
- Experimentos y conclusiones
  - Experimentación
  - Conclusiones

Introducción

Objetivos

Planificación

Objetivos

- Diseñar algoritmo de programación genética
- Desarrollar una aplicación para utilizar el algoritmo
- Realizar experimentos para determinar los mejores parámetros del algoritmo

Planificación

Tarea	Tiempo (horas)
Estudio de la documentación de programación genética	100
Estudio del lenguaje C++, preparación del entorno de programación	40
Estudio de mercados financieros, obtención de datos de históricos de precios	40
Reuniones con el director del proyecto	20
Diseño del algoritmo	100
Implementación y testeo del algoritmo	140
Análisis y especificación	100
Diseño	40
Implementación	200
Experimentación	60
Redacción de la documentación	120

- Duración: 960 h.

Algoritmos Genéticos y Mercados Financieros

Introducción a la programación genética

Nociones sobre los mercados de acciones

Estructura del algoritmo implementado

Algoritmos Genéticos

- Familia de algoritmos de IA
- Espacio de búsqueda inabordable con búsqueda exhaustiva
- Decisiones importantes
  - Representación de las soluciones
    - Cadenas de objetos (normalmente bits)
  - Generación población inicial
  - Operadores y sus probabilidades
  - Función evaluación o de *fitness*
  - Criterio finalización

Algoritmos Genéticos

- Pasos
  - Crear una población de soluciones aleatoria
  - Aplicar operadores sobre los individuos un cierto número de veces
    - Operadores imprescindibles: cruce y reproducción
    - La población siempre cuenta con los mismos individuos
    - Se remplazan los individuos de peor calidad
  - Acabar cuando la población no mejora

Programación Genética

- Tipo de algoritmo genético
- Representa a los individuos mediante árboles
  - No hay limitación en la longitud de los individuos
- Cada nodo del árbol es una función
- Operadores
  - Principales
    - Reproducción
    - Cruce
  - Secundarios
    - Mutación
    - Permutación
    - Edición
    - Encapsulación
    - Destrucción

Mercados Financieros

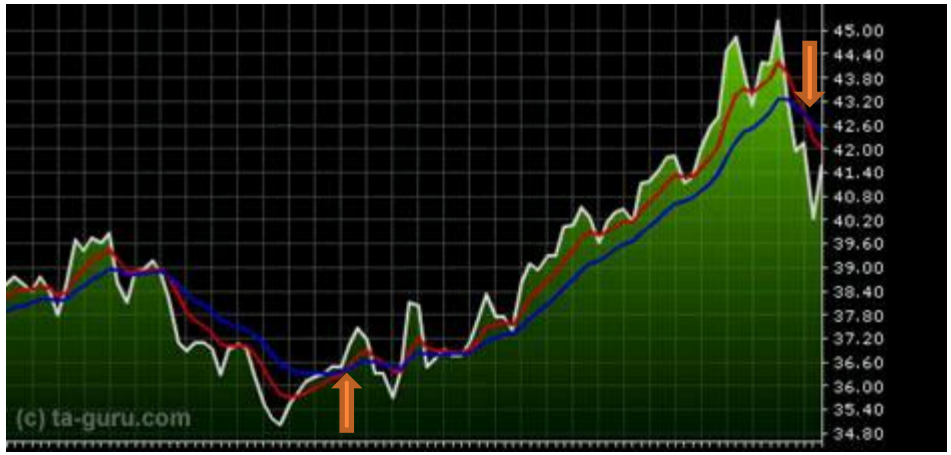
- Intercambio de activos entre inversores
- Precio regulado según oferta y demanda
- Mercado de acciones
  - Cotizan participaciones de empresas grandes

Mercados Financieros

- Estrategias inversión
  - Activas
    - Operaciones frecuentes
    - Costes por transacción influyen considerablemente
    - Inversión en empresas concretas
  - Pasivas
    - Largo plazo
    - Inversión en un conjunto de empresas o en un índice
    - Históricamente mejor que las activas
    - La más conocida es *buy and hold*

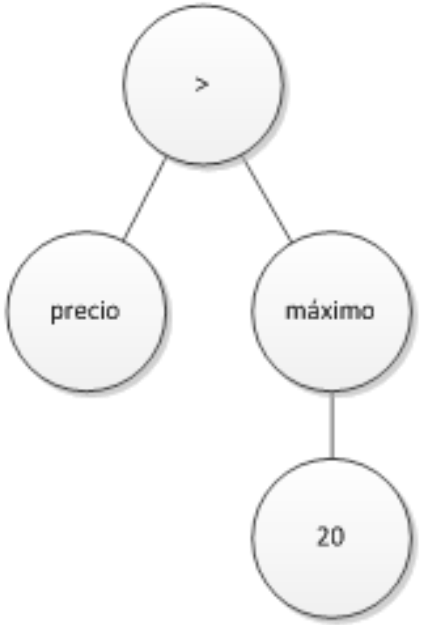
Mercados Financieros

- Criterios para invertir
  - Análisis técnico
    - Basado en tres principios:
      - Toda la información incluida en el precio
      - El precio se mueve en tendencias
      - La historia se repite
    - Indicadores cuantitativos sobre el estado de la tendencia
    - Ejemplo con dos medias móviles (corto y largo plazo):



Algoritmo Desarrollado

- Tipos de nodos
  - Indicadores de análisis técnico
  - Constantes
  - Operadores lógicos y aritméticos
- Ejemplo de regla:



Algoritmo Desarrollado

- Tipos de datos
  - Booleano: condición
  - Entero: número de días
  - Decimal: precio (normalizado)
- Los árboles resultantes han de ser correctos
  - El nodo raíz ha de devolver un valor booleano
  - Cada nodo devuelve un tipo de datos predefinido
  - Cada nodo acepta uno o más tipos de datos en sus descendientes
  - Ventaja: acota el espacio de búsqueda
  - Inconveniente: más tiempo al aplicar operadores

Algoritmo Desarrollado

- Basado en Allen y Karjalainen
  1. Crear una población de reglas aleatoriamente
  2. Guardar la mejor regla en el periodo de entrenamiento como mejor regla
  3. Aplicar operadores sobre la población de reglas
  4. Aplicar la mejor regla en el periodo de entrenamiento al de validación
  5. Si la regla anterior supera a la mejor regla guardada en el periodo de validación, sustituirla
  6. Ejecutar 3, 4 y 5 (llamados generación) hasta que no haya mejora durante un cierto número de generaciones o hayan pasado un cierto número total

Algoritmo Desarrollado

- Estrategia de generación con dos periodos: entrenamiento y validación
  - Ayuda a eliminar el sobreentrenamiento
- Función de *fitness*
  - Beneficio absoluto obtenido en el periodo, contando los costes por cada transacción
  - Cada regla se evalúa para cada día de mercado
  - $fitness = \sum_{t=1}^{T}(I(t) \cdot \ln\frac{P_t}{P_{t-1}}) + n \cdot \ln\frac{1-c}{1+c}$
    - T: días del periodo; P_t: precio del día *t*; n: número de transacciones en el periodo; c: coste por transacción (tanto por uno)
    - I(t) = 1 si en el día *t* la regla indica estar dentro del mercado
    - I(t) = 0 si en el día *t* la regla indica estar fuera del mercado
- 3 operadores
  - Reproducción (implícito)
  - Cruce
  - Mutación

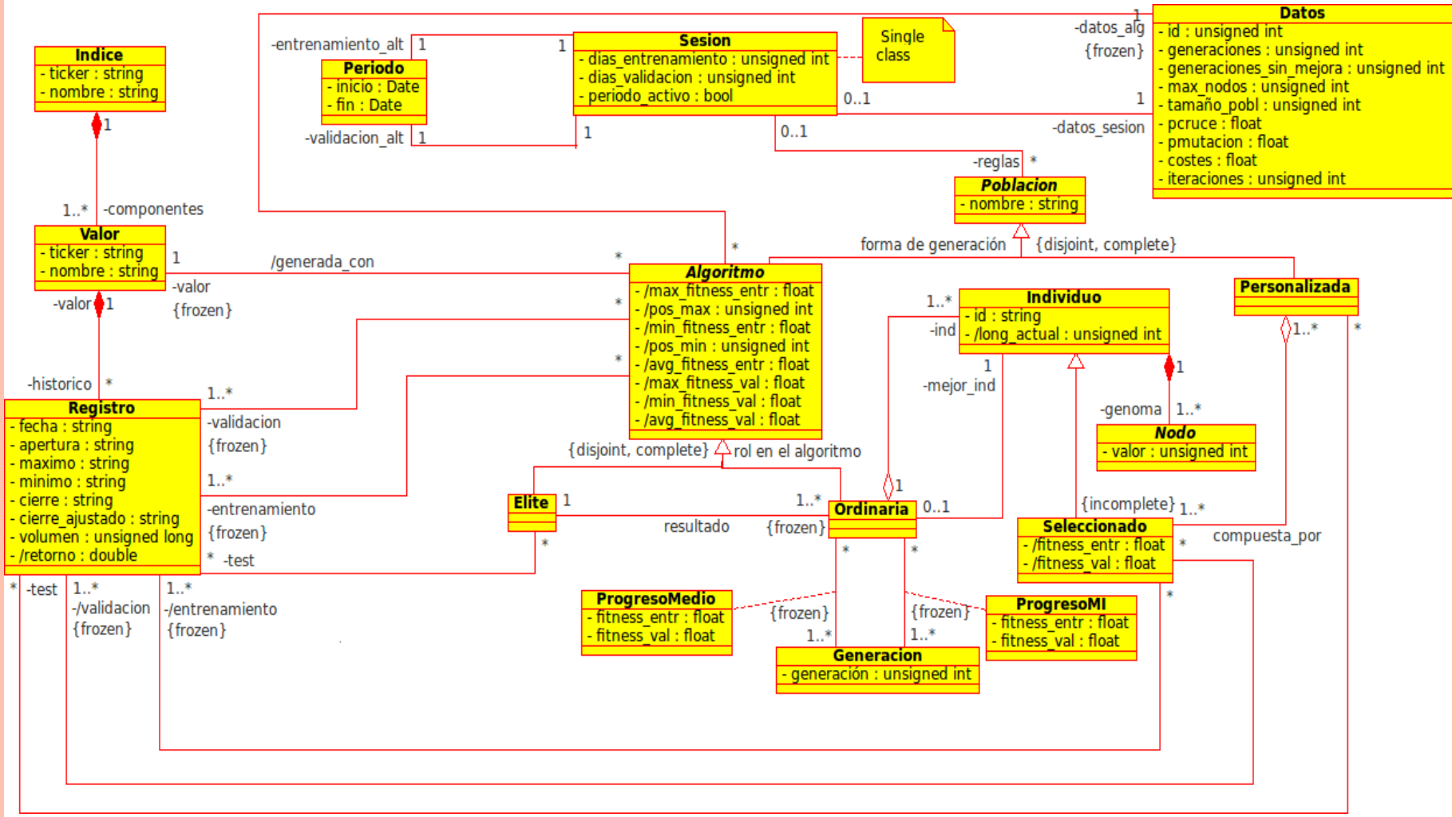
DESARROLLO DE LA APLICACIÓN

Especificación

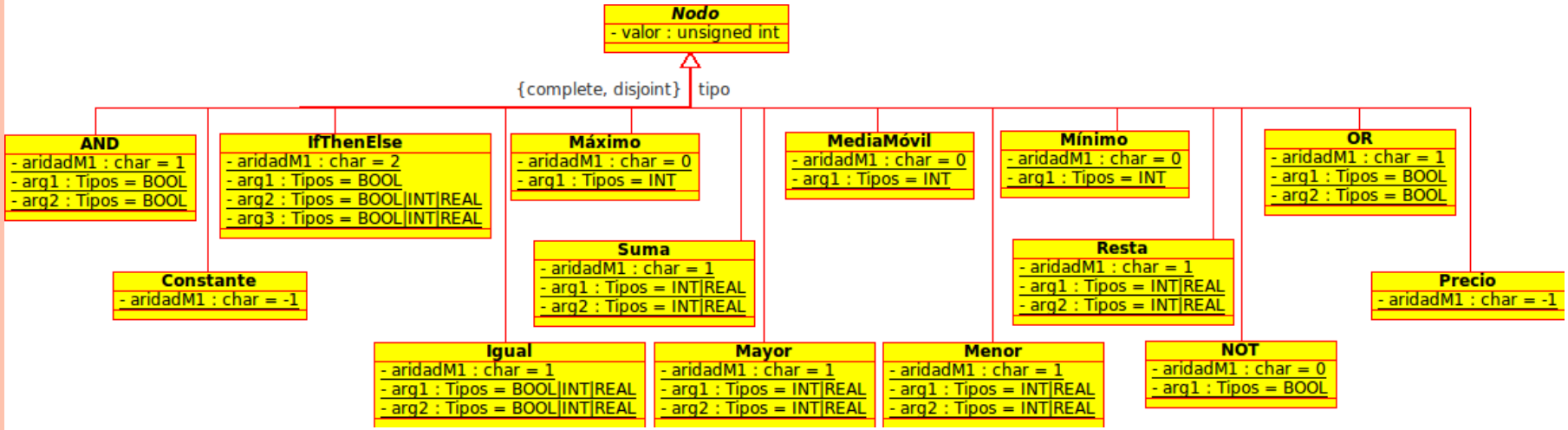
Diseño

Implementación

ESPECIFICACIÓN



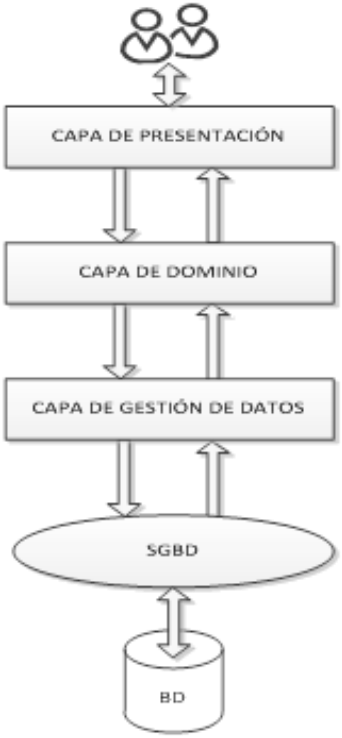
Especificación



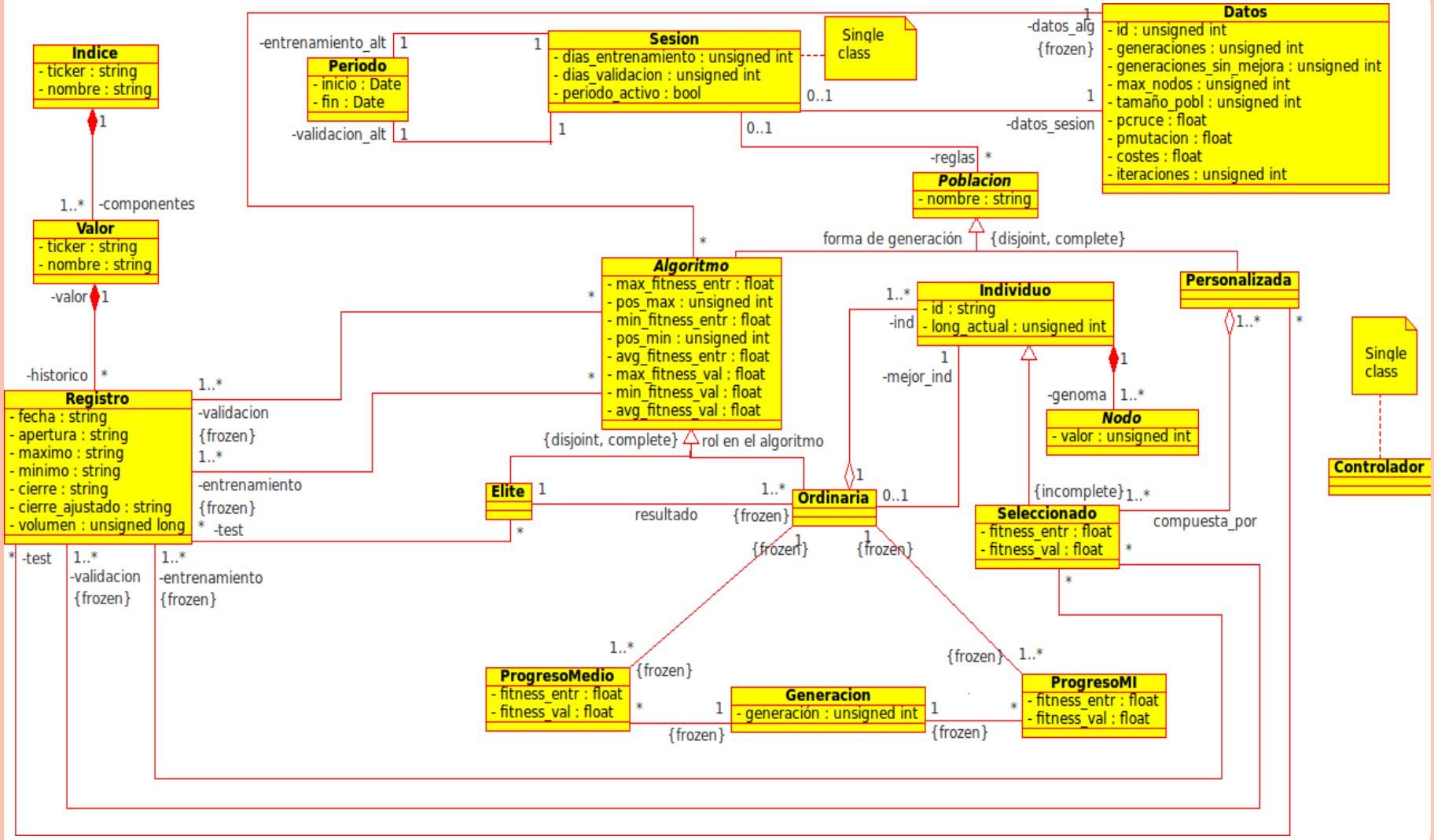
- Método adoptado del artículo de Keith y Martin

Diseño

- Arquitectura 3 capas
  - Separación lógica de las funcionalidades
  - Facilita la extensión y comprensión del sistema
  - Cada capa dispone de un controlador



Diseño


Indice
- ticker : string
- nombre : string
Valor
- ticker : string
- nombre : string
Registro
- fecha : string
- apertura : string
- maximo : string
- minimo : string
- cierre : string
- cierre_ajustado : string
- volumen : unsigned long
Periodo
- inicio : Date
- fin : Date
Sesion
- dias_entrenamiento : unsigned int
- dias_validacion : unsigned int
- periodo_activo : bool
Single class
Datos
- id : unsigned int
- generaciones : unsigned int
- generaciones_sin_mejora : unsigned int
- max_nodos : unsigned int
- tamaño_pobl : unsigned int
- pcruce : float
- pmutacion : float
- costes : float
- iteraciones : unsigned int
Poblacion
- nombre : string
Algoritmo
- max_fitness_entr : float
- pos_max : unsigned int
- min_fitness_entr : float
- pos_min : unsigned int
- avg_fitness_entr : float
- max_fitness_val : float
- min_fitness_val : float
- avg_fitness_val : float
Individuo
- id : string
- long_actual : unsigned int
Personalizada
Nodo
- valor : unsigned int
Seleccionado
- fitness_entr : float
- fitness_val : float
Elite
Ordinaria
ProgresoMedio
- fitness_entr : float
- fitness_val : float
Generacion
- generación : unsigned int
ProgresoMI
- fitness_entr : float
- fitness_val : float
Single class
Controlador
-componentes
-valor
-historico
-validacion
-entrenamiento
-test
-entrenamiento_alt
-validacion_alt
-datos_alg
-datos_sesion
-reglas
forma de generación
{disjoint, complete}
rol en el algoritmo
-ind
-mejor_ind
-genoma
{incomplete}
compuesta_por
resultado
{frozen}

Implementación

- Lenguajes de programación
  - C++ con Qt
  - SQL
- Base de datos
  - MySQL
- Herramientas
  - NetBeans 6.9.1
  - MySQL WorkBench 5.2
  - Boost

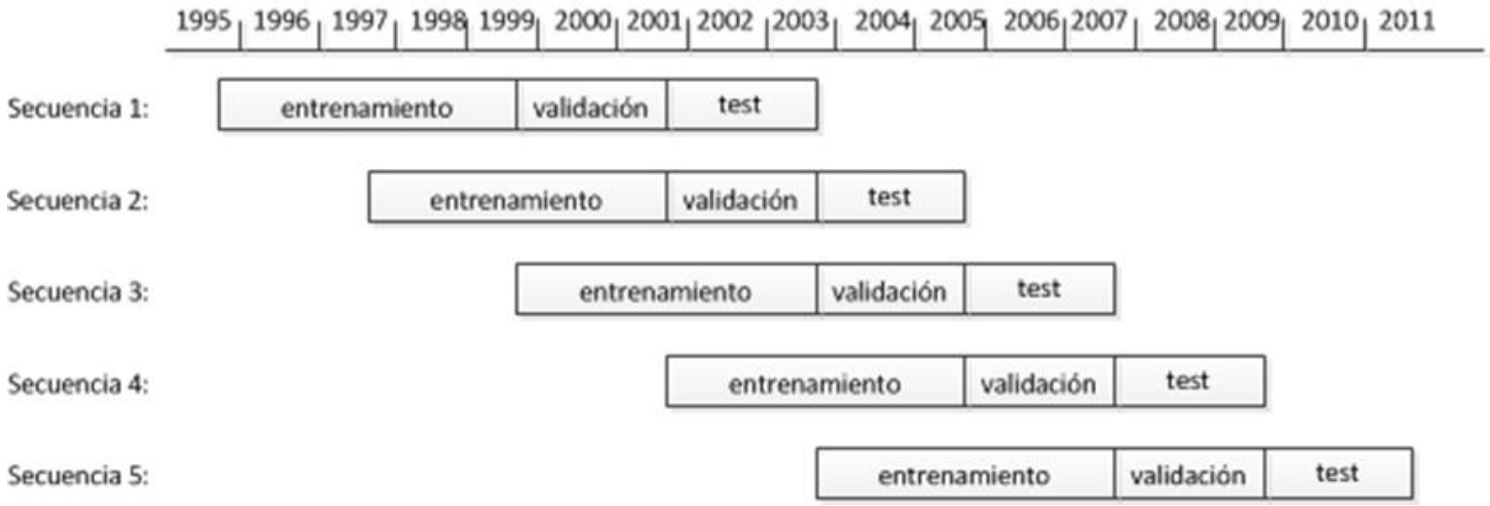
EXPERIMENTOS Y CONCLUSIONES

Experimentación

Conclusiones

Experimentación

- Objetivos
  - Comportamiento según los costes por transacción
  - Influencia de la tendencia en el resultado
  - Comprobar si se puede superar a *buy and hold*
- Método *rolling forward:*



Experimentación

- Parámetros usados:

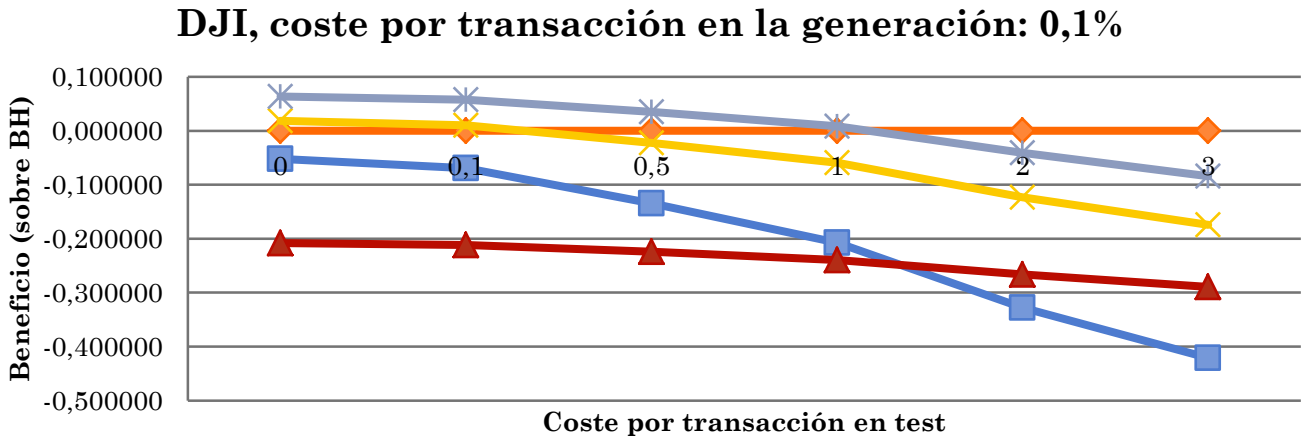
Probabilidad de cruce	90%
Probabilidad de mutación	5%
Iteraciones	50
Tamaño de la población de individuos	200
Máximo número de generaciones	100
Número de generaciones sin obtener mejora	50
Número máximo de nodos por individuo	80

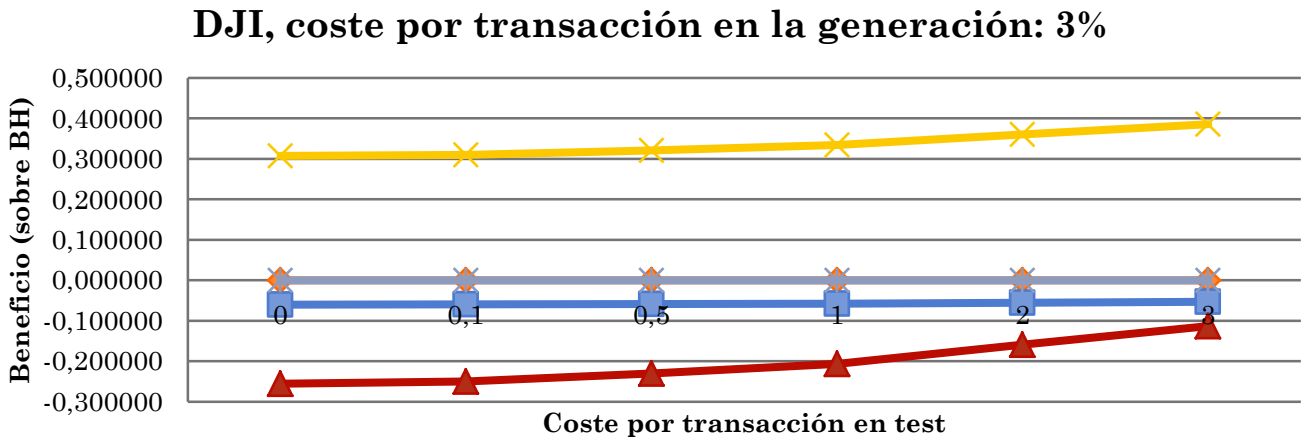
Experimentación

- Resultados
  - Comportamiento lógico de las reglas
    - Reglas adaptadas al coste con el que se generan
    - Resultados similares con iguales parámetros
    - Menos transacciones con mayores costes en la generación
  - Costes en la generación altos
  - Probable correlación entre la tendencia y los resultados
  - Posibilidad de mejorar el comportamiento de *buy and hold* con la tendencia adecuada

Experimentación

- Muestra de resultados:





Experimentación

- Conclusiones

Secuencia	Entrenamiento	Validación	Test	Resultado
1	Alcista muy fuerte	Mayormente bajista	Mayormente bajista	Siempre igual que BH
2	Mixta (alcista – bajista)	Mixta (bajista – alcista)	Alcista	Siempre peor que BH
3	Mayormente bajista	Alcista	Alcista (muy fuerte en IBEX)	Siempre peor que BH (el peor)
4	Mixta (bajista – alcista x 2)	Alcista (muy fuerte en IBEX)	Mixta (bajista muy fuerte – alcista)	Siempre mejor que BH
5	Alcista	Mixta (bajista muy fuerte x 2 – alcista)	Alcista en DJI Plana en IBEX	Mejor o igual que BH en DJI Ligeramente peor que BH en IBEX

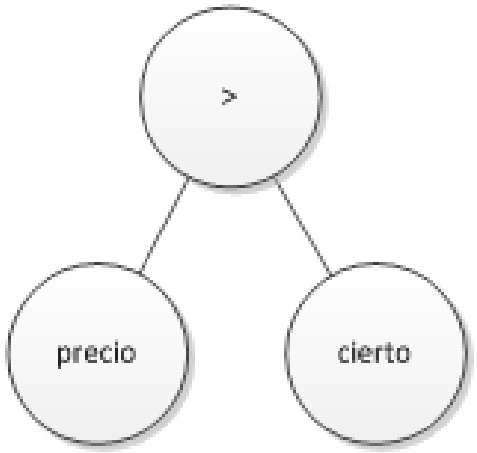
Conclusiones

- Expansión de conocimientos sobre algoritmos genéticos y sobre finanzas
- Diseño de un algoritmo a partir de otros trabajos e ideas propias
- Adquisición de habilidad de programación en C++ y Qt
- Desarrollo de una aplicación mediante ingeniería del software
- Introducción a la investigación en IA
- Redacción de una extensa documentación

Gracias por su atención

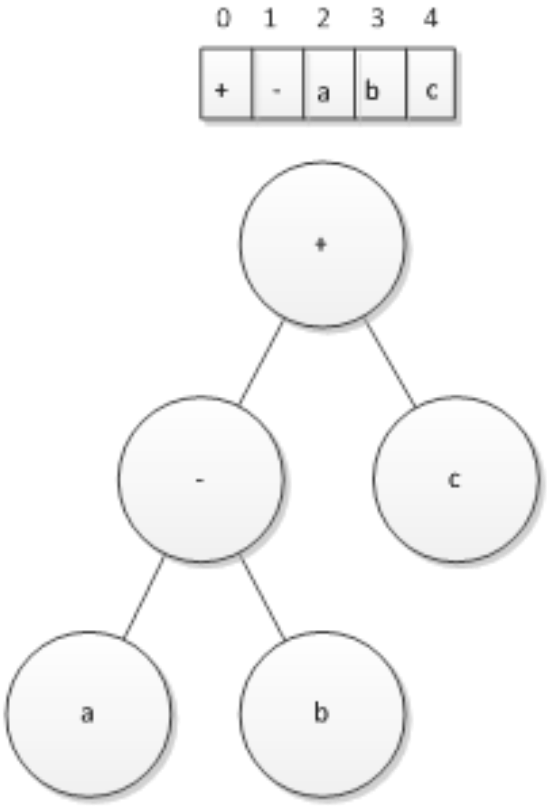
Algoritmo Desarrollado

- Ejemplo de regla ambigua e incorrecta:



Algoritmo Desarrollado

- Ejemplo de representación *prefix*:

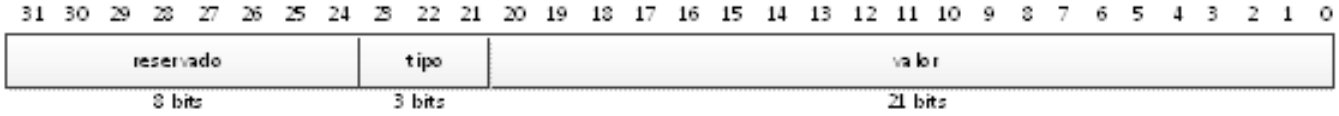


Algoritmo Desarrollado

- Tipos de nodos
  - Indicadores de análisis técnico
    - Precio
    - Máximo
    - Mínimo
    - Media móvil
  - Constantes (booleanas, enteras, decimales)
  - Operadores lógicos y aritméticos
    - AND
    - OR
    - NOT
    - Mayor
    - Menor
    - Igual
    - Suma
    - Resta
    - If – then – else

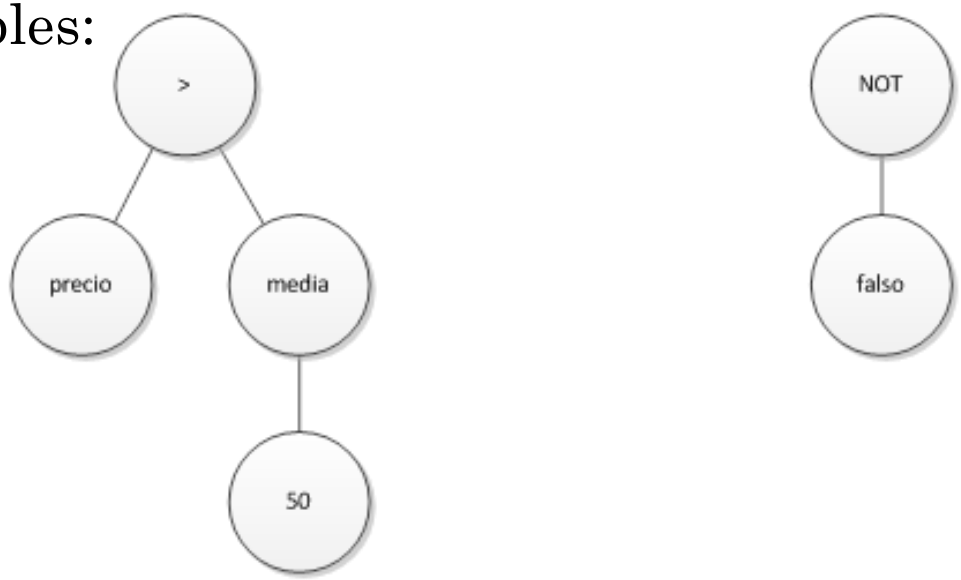
Algoritmo Desarrollado

- Representación de la información de un nodo:
  - 4 bytes (número entero)
  - "tipo": tipo de datos que contiene el nodo
  - "valor"
    - Booleano: falso = 0; cierto !=0
    - Entero
      - Rango: +1.048.575 … -1.048.576
    - Decimal
      - Coma fija (4 bits parte entera, 16 bits parte decimal)
      - Rango: +15,99998 … -16,99998
      - Precisión 1,526 $\cdot 10^{-5}$



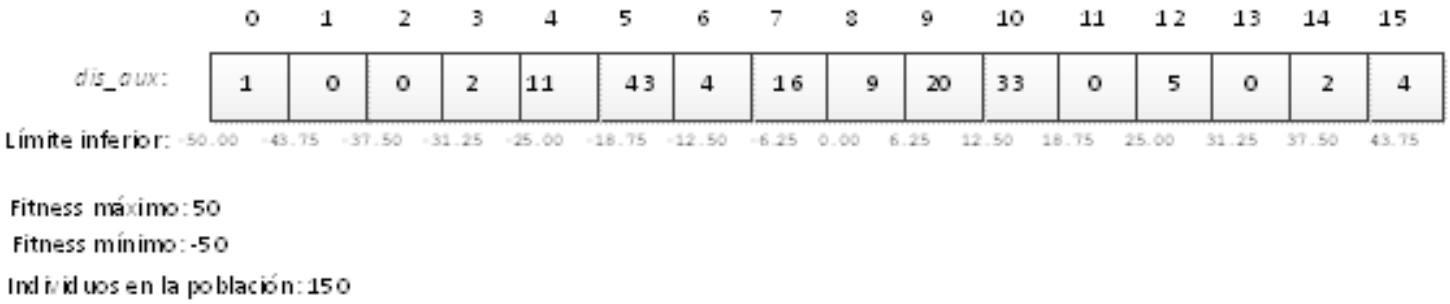
Algoritmo Desarrollado

- Cruce
  - Descendiente mejor siempre remplaza un progenitor
  - La probabilidad de que un progenitor sea remplazado es proporcional a la diferencia de *fitness* con el otro progenitor
  - Es posible que algunos individuos sean incompatibles:



Algoritmo Desarrollado

- Estructura para la selección:

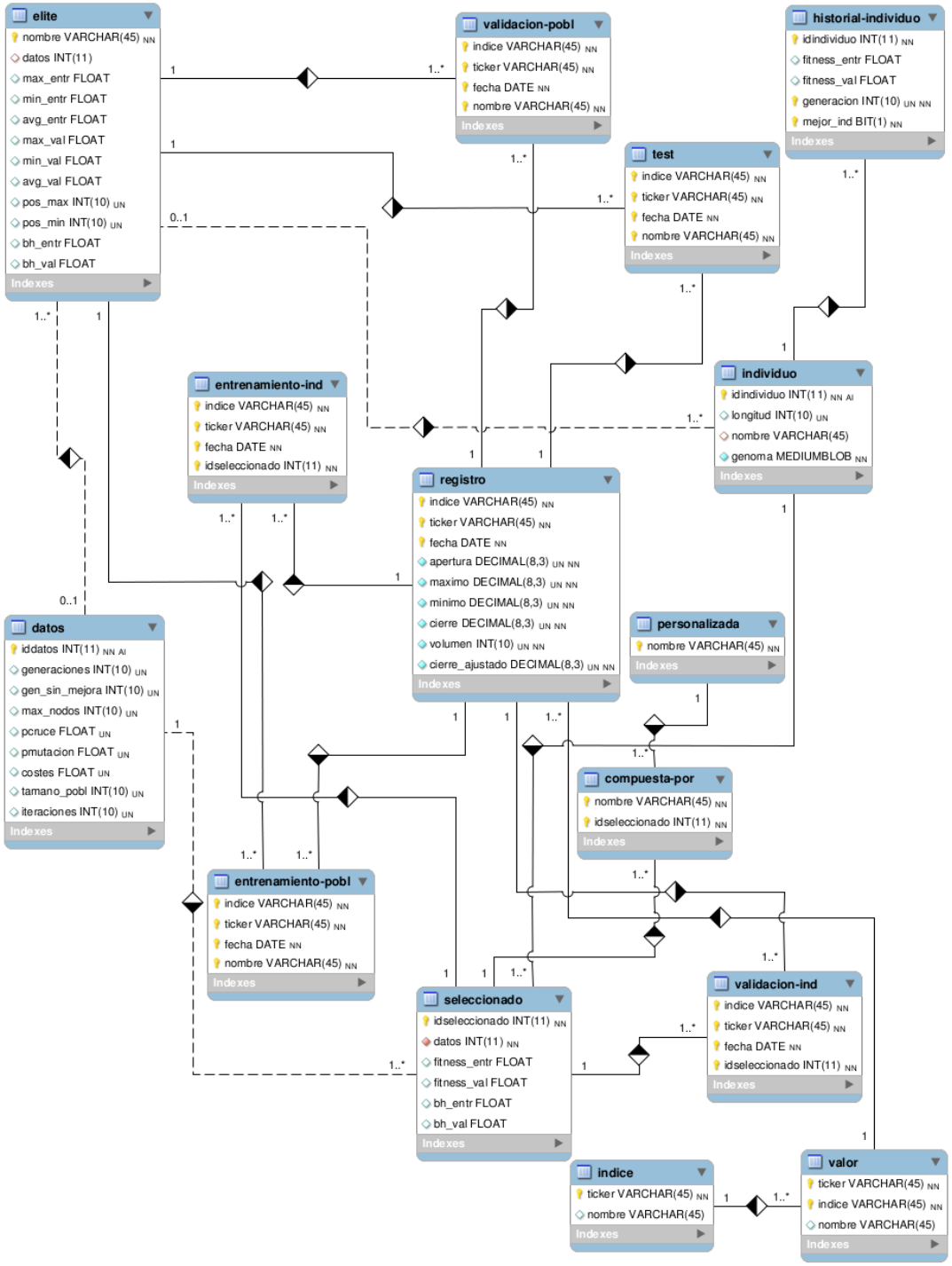


- Probabilidad de elegir una posición:

$$\text{adj\_fitness}_i = \frac{\text{p\_base}_i \cdot \text{num\_ind}_i}{\sum_{j=1}^{\text{num\_pos}} \text{p\_base}_j \cdot \text{num\_ind}_j}$$

- Posición de un individuo:

$$\text{posición} = \left\lfloor \frac{\text{num\_pos} \cdot (\text{fitness}_i - \text{min\_fitness})}{\text{max\_fitness} - \text{min\_fitness}} \right\rfloor$$


elite
nombre VARCHAR(45) NN
datos INT(11)
max_entr FLOAT
min_entr FLOAT
avg_entr FLOAT
max_val FLOAT
min_val FLOAT
avg_val FLOAT
pos_max INT(10) UN
pos_min INT(10) UN
bh_entr FLOAT
bh_val FLOAT
Indexes
validacion-pobl
indice VARCHAR(45) NN
ticker VARCHAR(45) NN
fecha DATE NN
nombre VARCHAR(45) NN
Indexes
historial-individuo
idindividuo INT(11) NN
fitness_entr FLOAT
fitness_val FLOAT
generacion INT(10) UN NN
mejor_ind BIT(1) NN
Indexes
test
indice VARCHAR(45) NN
ticker VARCHAR(45) NN
fecha DATE NN
nombre VARCHAR(45) NN
Indexes
entrenamiento-ind
indice VARCHAR(45) NN
ticker VARCHAR(45) NN
fecha DATE NN
idseleccionado INT(11) NN
Indexes
individuo
idindividuo INT(11) NN AI
longitud INT(10) UN
nombre VARCHAR(45)
genoma MEDIUMBLOB NN
Indexes
registro
indice VARCHAR(45) NN
ticker VARCHAR(45) NN
fecha DATE NN
apertura DECIMAL(8,3) UN NN
maximo DECIMAL(8,3) UN NN
minimo DECIMAL(8,3) UN NN
cierre DECIMAL(8,3) UN NN
volumen INT(10) UN NN
cierre_ajustado DECIMAL(8,3) UN NN
Indexes
datos
iddatos INT(11) NN AI
generaciones INT(10) UN
gen_sin_mejora INT(10) UN
max_nodos INT(10) UN
pcruce FLOAT UN
pmutacion FLOAT UN
costes FLOAT UN
tamano_pobl INT(10) UN
iteraciones INT(10) UN
Indexes
personalizada
nombre VARCHAR(45) NN
Indexes
compuesta-por
nombre VARCHAR(45) NN
idseleccionado INT(11) NN
Indexes
entrenamiento-pobl
indice VARCHAR(45) NN
ticker VARCHAR(45) NN
fecha DATE NN
nombre VARCHAR(45) NN
Indexes
seleccionado
idseleccionado INT(11) NN
datos INT(11) NN
fitness_entr FLOAT
fitness_val FLOAT
bh_entr FLOAT
bh_val FLOAT
Indexes
validacion-ind
indice VARCHAR(45) NN
ticker VARCHAR(45) NN
fecha DATE NN
idseleccionado INT(11) NN
Indexes
indice
ticker VARCHAR(45) NN
nombre VARCHAR(45)
Indexes
valor
ticker VARCHAR(45) NN
indice VARCHAR(45) NN
nombre VARCHAR(45)
Indexes