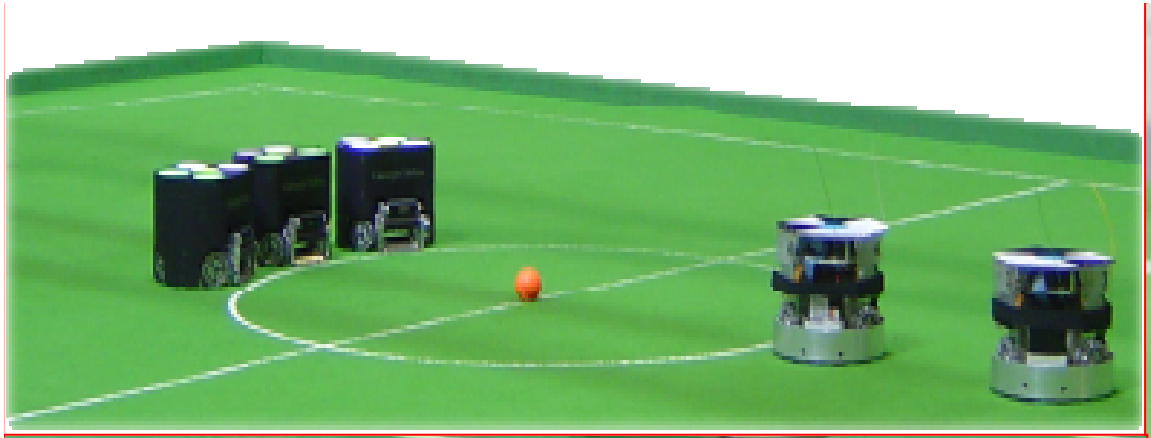
Programación Genética Aplicada a Robosoccer

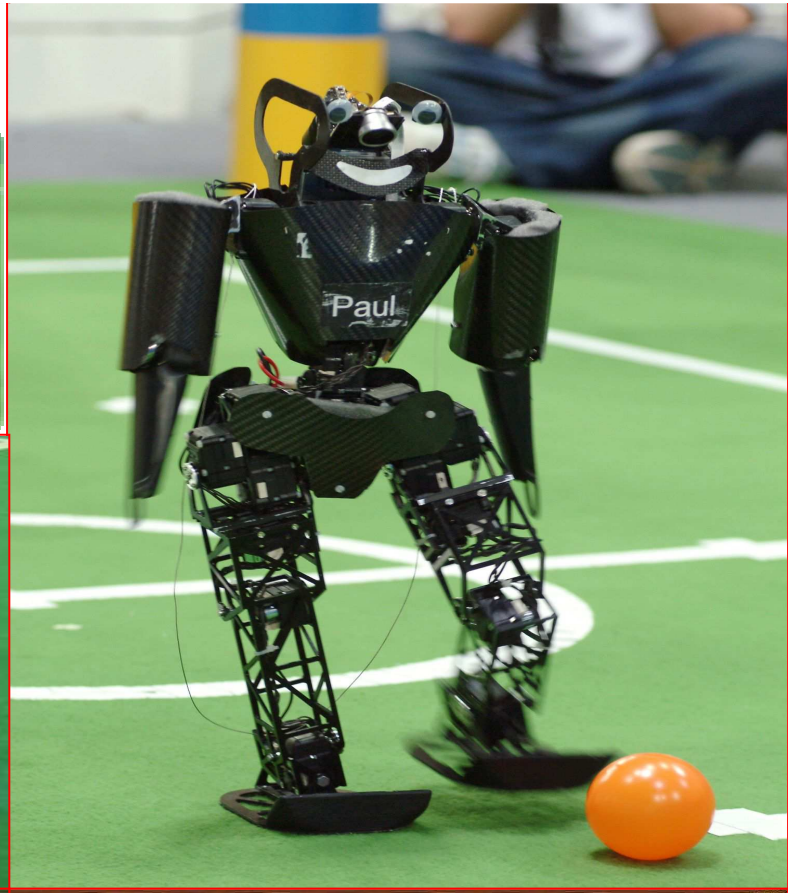
La Robocup / Robosoccer / Fútbol robótico

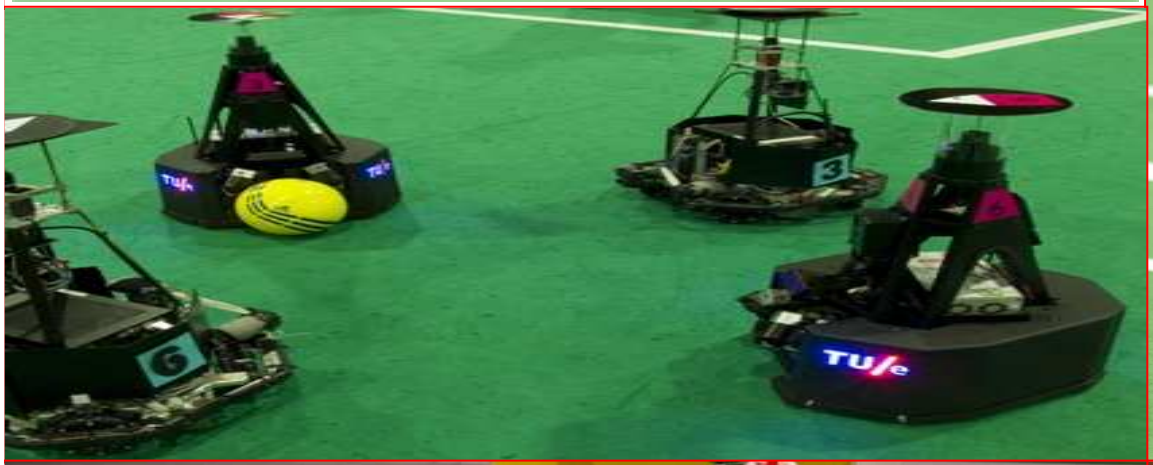
- Iniciativa internacional. www.robocup.org
- Promueve el estudio en los campos de la Robótica y de la IA.
- Contexto: Fútbol.
- Nace bajo el eslogan:
  - *By the year 2050, develop a team of fully autonomous humanoid robots that can win against the human world soccer champion team.*

Ligas Robocup

- Categorías.
  - Robots Pequeños, Robots Medianos, Robots con Patas, Robots humanoides, **Liga de Simulación**.
- Liga de Simulación.
  - Estudios a nivel de software.
  - Basada en SoccerServer: Servidor que establece el contexto de juego.
  - 22 clientes y un árbitro humano.
- Eventos.
  - Anuales Desde 1996




Paul


TU/e
TU/e
3
6


soccermonitor
Off
Quit
to: 0
play_on
2786
atu: 0

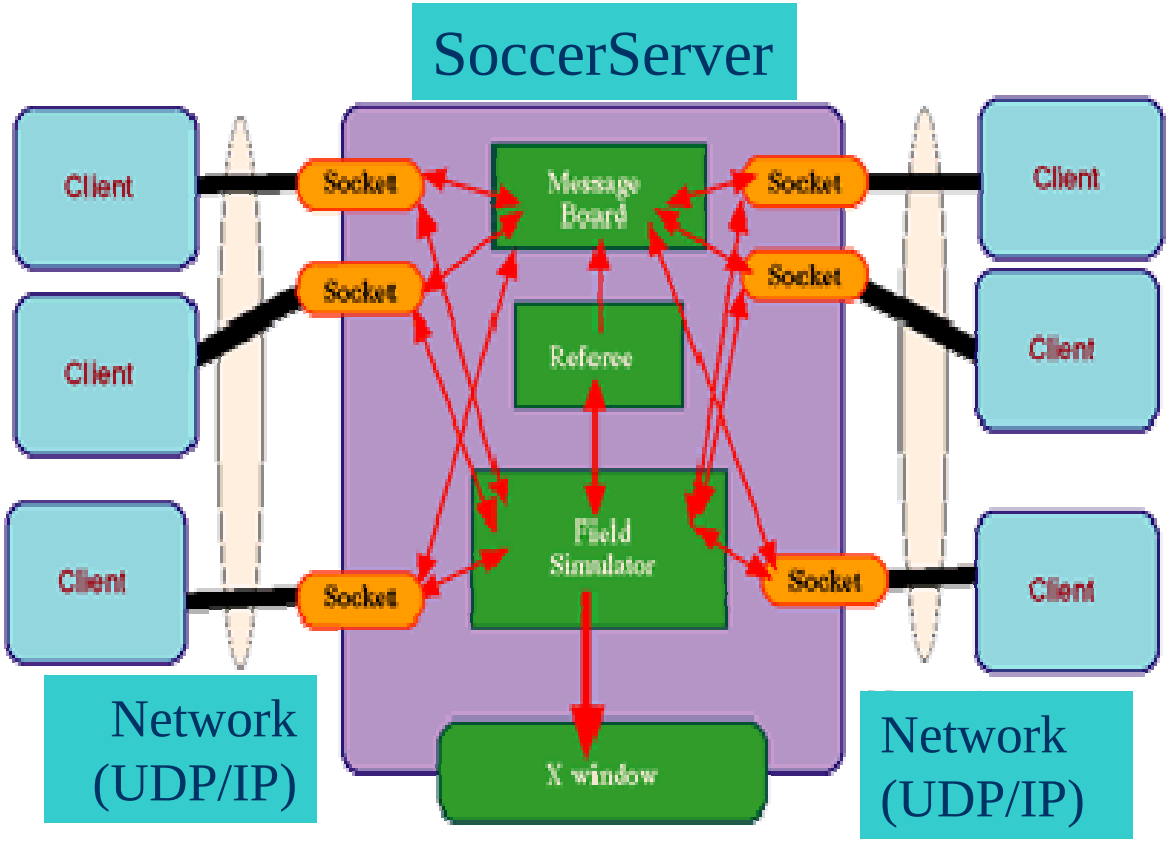
SoccerServer

Es un sistema que permite que agentes software autónomos jueguen un partido unos contra otros.

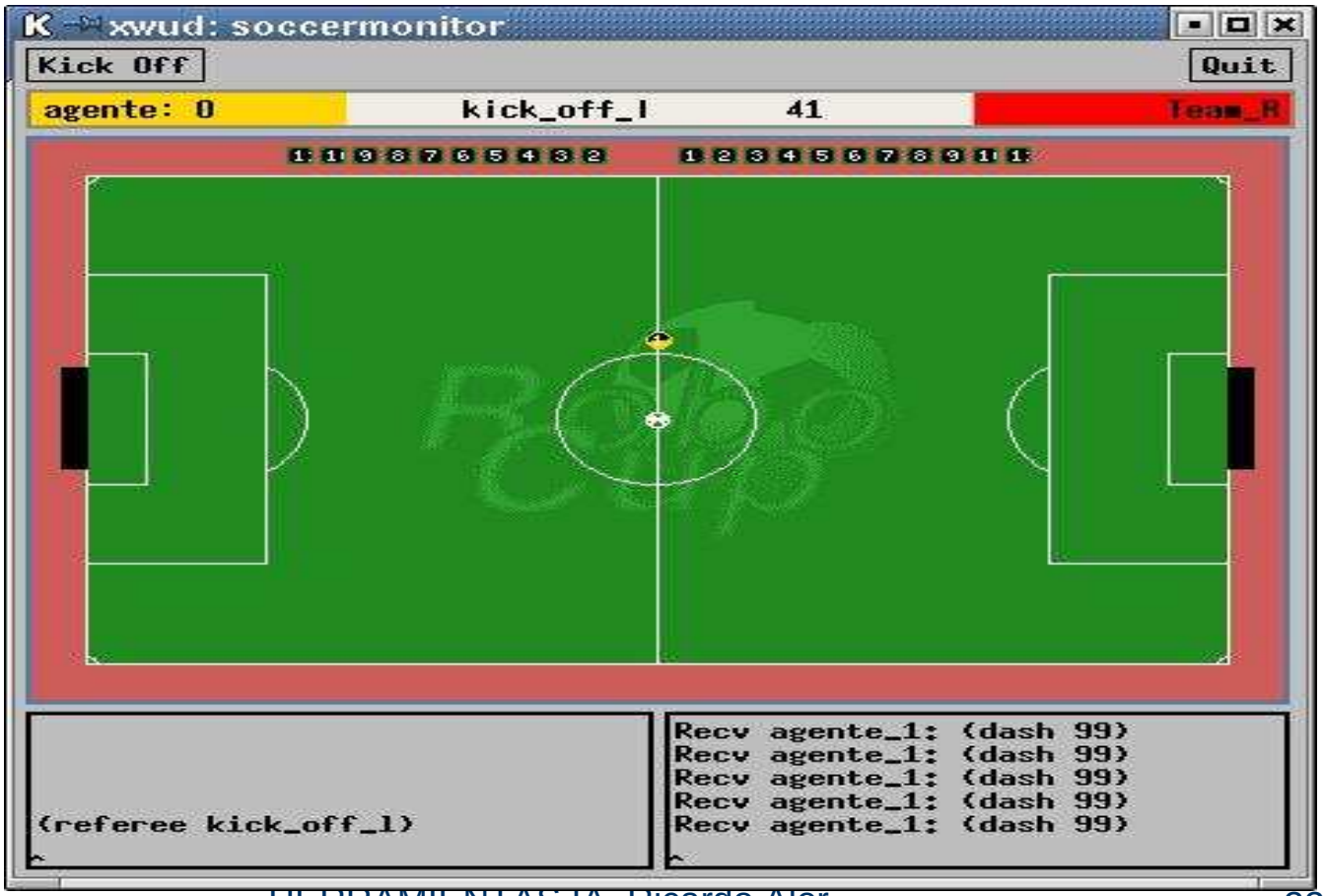
Contiene dos módulos:

- SoccerServer.
- SoccerMonitor.

SoccerServer. Arquitectura



SoccerMonitor



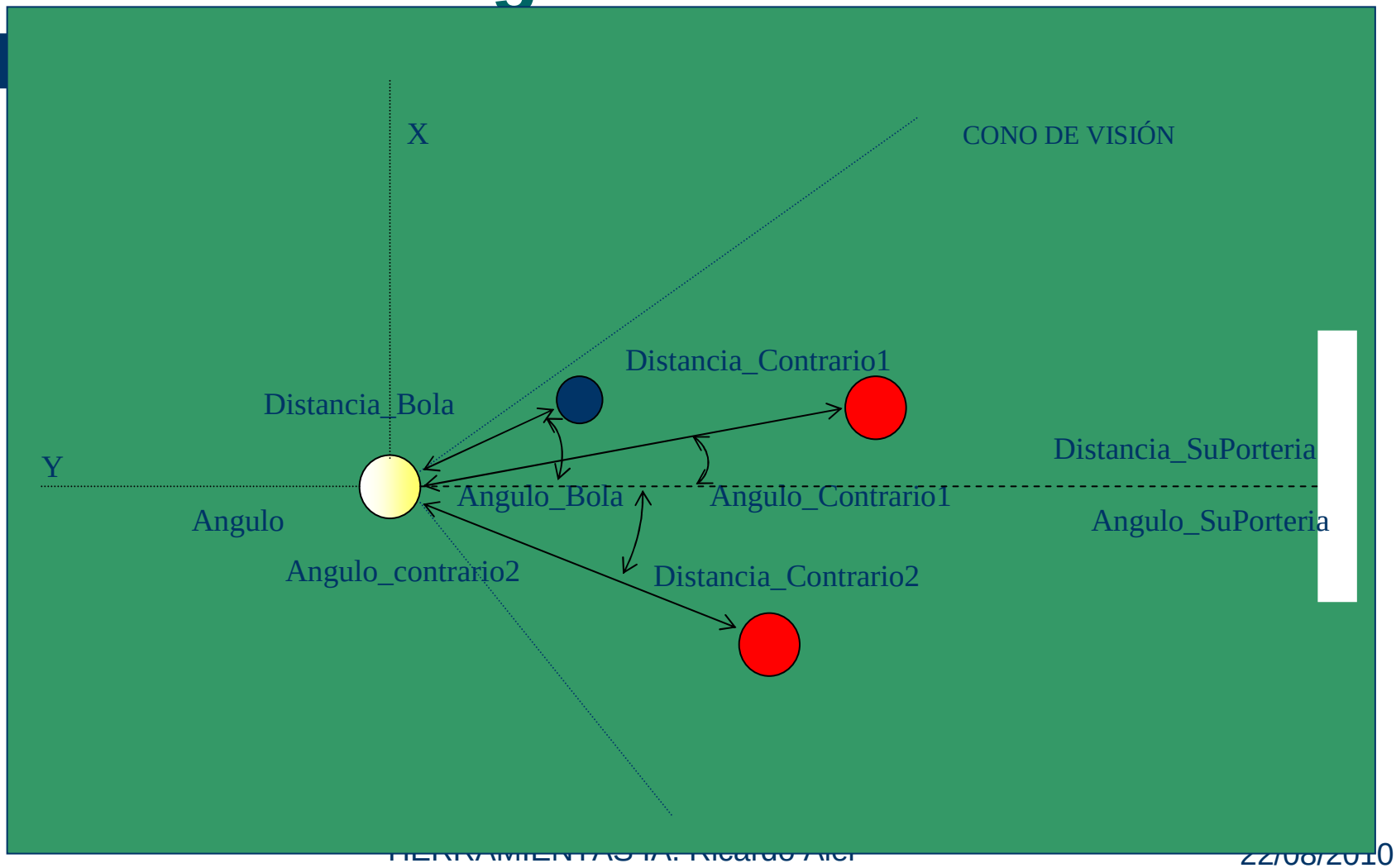
Características del Dominio RoboCup

- Acción estructurada en ciclos
- Adición por parte del simulador de ruido aleatorio en sensores y acciones
- Visión limitada (según la distancia)
- Hay viento, rozamiento, etc
- Acciones: turn(ángulo), kick(fuerza, ángulo), dash(fuerza), turn_neck(ángulo), yell(mensaje)
- Turn y dash son incompatibles
- Modelo de estamina (cansancio)

Características del Dominio RoboCup (II)

- Información de los sensores de los agentes basada en distancia y ángulos.
- Campo señalizado por 53 objetos (flags) estáticos, 4 líneas y dos porterías diseminados por el campo.
- No se reciben posiciones absolutas (X,Y) del resto de jugadores ni de uno mismo. Calcularlas a partir de la posición de las banderas (!)

Visión del Agente



Requisitos Robosoccer

- Para tener éxito en la competición hay que tener en cuenta:
  - Equipo de agentes cooperativos
  - En un domino “ruidoso” (sensores y acciones)
  - Física compleja
  - Comunicación limitada entre agentes

Aplicación de GP a Robosoccer

- Sean Luke. “Genetic Programming Produced Competitive Soccer Softbot **Teams** for Robocup97. Genetic Programming 1998”
- Problema: ¡tiempo!
  - No es (era) posible acelerar el SoccerServer
  - 100.000 evaluaciones estimadas (seguramente muchas más)
  - 1 evaluación = 1 partido (10 minutos)
  - ¡4 años de evolución!

Soluciones iniciales

- Reducir partidos a 20-60 segundos. Paralelización (cluster) a nivel de evaluaciones y de ejecuciones
- Simplificación del dominio: cálculo de posiciones absolutas (X,Y) de uno mismo y de los demás. No se usa comunicación

Codificación de los árboles

- Algoritmo programado a mano: "*Si se puede ver la pelota y está cerca, chuta. Si se puede ver pero no está cerca, avanza. En caso contrario, gira hasta verla*"
- Dos subárboles por jugador, devuelven vectores
  - Dirección y fuerza con la que chutar
  - Dirección y fuerza con la que avanzar
- Individuo = equipo. Posibilidades: heterogéneo / homogeneo

Funciones y terminales

- Se podrían utilizar las acciones de bajo nivel (dash, kick, etc), pero los primeros experimentos mostraron que era inviable
- Programación genética tipada (booleanos, enteros, vectores)

Función	D	Descripción
(home), (home-of i) (ball), (mate i), (goal)	V	Vectores a casa, pelota, compañero, portería
(blockgoal), (block-near-opp)	V	Vector al punto más cercano del segmento pelota-mi portería (o pelota-oponente)
(opp-closer), (mate-closer)	B	T si un oponente/compañero está más cerca de la pelota que yo
(inv v)	V	Rota ángulo 180 grados
(if-v b v1 v2)	V,K	Si b cierto, devuelve v1 si no, v2
(sight v)	V	Rotar v para ver la pelota
(ofme i), (ofhome i) (ofgoal i)	B	T si la pelota cerca de mi, de mi casa, de la portería
(far-mate i2), (mate-m i1 i2)	K	Posición del compañero más lejano que puede recibir la pelota con probabilidad i2/10, o del compañero i1
(kick-goal i)	K	Vector a la portería si chutar tendrá éxito con probabilidad i/10
(far-mate!)	K	Chutar al compañero más lejano. Si no, a la portería
(kick-clear)	K	Despeja pelota
(opponents-close i)	B	Devuelve T si un oponente está a i unidades de mi
(away-mates), (away-opps)	V	Centros de gravedad de compañeros y oponentes

Modificaciones al algoritmo

- GP tipada (2 tipos de datos: vector y booleano)
- Crossover homólogo (sólo cruza árboles de chutar con árboles de chutar. Idem con mover)
- Nuevo operador de cruce: intercambio de jugadores

Parámetros

- Poblaciones entre 100 y 400
- Posibilidad de estancamiento (convergencia prematura) -> Mutación 30%
- Selección: torneo. Cada emparejamiento corre en un procesador diferente DEC-alpha de 40 nodos, con lil-gp tipado y multi-hilo)
- Tiempo: semanas a meses

Función de *fitness*. Problemas

- Enfrentar al individuo con un buen equipo hecho a mano
- Problemas:
  - Oponente demasiado difícil al principio, es mejor que la dificultad mejore de manera gradual
  - Sobreadaptación a ese oponente en concreto

Función de *fitness*. Soluciones

- Que cada individuo compita con otros individuos en la población
- La dificultad evoluciona de manera gradual
- Se garantiza la generalidad

Función de *fitness*. Componentes

- Primera idea: combinar número de goles, tiempo de posesión, distancia de la pelota a la portería, número de pases con éxito, ... -> no se conseguía mejorar la *fitness* demasiado
- Mejor: utilizar diferencia en goles marcados. Parecería que las primeras generaciones quedarían 0-0, pero no era así. Eran equipos ofensivos y poco defensivos

Equipo que se envió a Robocup97

- Se puso a competir a los 40 mejores individuos evolucionados
- El ganador fue enviado a la competición
- Era un equipo homogéneo. Los autores creen que, con más tiempo de evolución, un equipo heterogéneo sería mejor
- Ganó a los dos primeros equipos y se llevaron el premio al mejor trabajo científico

Historias de las distintas evoluciones

- Al principio los jugadores se movían aleatoriamente, miraban a la pelota o perseguían compañeros
- Una vez apareció un equipo que huía de la pelota contra otro que huía del primer equipo. No duraron mucho
- En ocasiones un jugador iba a por la pelota y chutaba a la portería -> ganadores
- “Todos a por la pelota y chutar a portería”. Riesgo de máximo local.
- Se empiezan a desarrollar estrategias defensivas (quedarse cerca de la portería para obstaculizar disparos a larga distancia)
- Empiezan a aparecer pases, en lugar de chutar siempre a puerta

Jugadores aleatorios

Todos a por la pelota

Bloquear la portería

Distribución por el campo

Posibles mejoras

- Sean Luke. “Evolving Soccer Bots: A Retrospective. Japanese society for AI 1998”
- Poblaciones más grandes (mas exploración)
- La evaluación de los equipos era poco precisa -> hacerlos competir con más equipos y sacar la media
- Evolucionar individuos en lugar de equipos
- Las funciones demasiado orientadas y no había estado interno

Segundo intento

- Andre and Teller. “Evolving Team Darwin United. Robocup 98”
- “Because of the complexity of the soccer server domain, it is futile to try to learn intelligent behaviors straight from the primitives provided by the server”
- Objetivo: utilizar las primitivas básicas del juego (dash, kick, turn, ...)
- Cómo: utilizar una función de fitness más elaborada y realizar el aprendizaje por pasos

Funciones y terminales

- Entradas: X, Y, distancia a la pelota, ángulo de la pelota, distancia a portería, ángulo de portería, distancias y ángulos de los compañeros y oponentes, cambios en la distancia y ángulo de la pelota (**delta**)
- Constantes: valores reales
- Uso de arrays[10]: read(i), write(i,v)
- Cálculos: add, sub, mult, div, sin, cos, if-then-else
- Acciones: kick(a,b), turn(a), dash(a), grab
- ADFs: 8 ADFs de 2 argumentos

Codificación de los árboles

- Cada árbol es un equipo entero
- Se dispone de ADFs que pueden ser usadas por todos los miembros del equipo
- Están preprogramadas y son poco óptimas (2 horas de trabajo humano). Ej: correr y chutar la pelota
- Las ADF pueden evolucionar

Función de *fitness*. Componentes

1. Acercarse a la pelota
2. Chutar la pelota
3. Posición de la pelota en el campo
4. Estar “vivo”
5. Marcar gol
6. Ganar un partido

Secuencia de entrenamiento

1. Marcar en campo vacío en 30 segundos
2. Marcar contra un equipo de “postes chutantes”
3. Marcar contra el ganador de Robocup 1997 (Humboldt University Germany)
4. Jugar contra otros equipos que superaron los 3 puntos anteriores

Modificaciones al algoritmo

- Cruce homólogo: sólo se cruzan (crossover) subárboles del mismo tipo

Resultados Robocup 1998

- Grupo E:
  - (1) Miya2 (0-0)
  - (2) PasoTeam (0-5)
  - (3) Darwin United
  - (4) Ulm-Sparrow (1-0)
- Darwin United obtuvo 4 puntos. Se sitúa en la mitad de la tabla de 34 equipos participantes
- Resultado no espectacular, pero prueba la idea
- Hoy se dispone de 10 veces más potencia computacional (ley de Moore)

Conclusiones

- La liga de simulación de Robocup es un problema complejo
- Pero la PG parece poder obtener resultados comparables a algunos programados a mano
- Aunque es necesario darle alguna orientación al aprendizaje (mediante funciones potentes o mediante *fitness* orientadoras)
- Esto ocurre también con otros problemas reales