Football Eleven

Yeray Li, Sergio Hernández, David Laguillo y Ander Amigorena

***Resumen*—Este proyecto aborda el problema de formar alineaciones óptimas de 11 jugadores de fútbol a partir de un conjunto de 492 jugadores de La Liga de la temporada 2023-2024, respetando un presupuesto máximo determinado. Cada jugador está caracterizado por su ID, nombre, equipo, posición, valoración (rating) y coste. El objetivo es maximizar el rendimiento del equipo mientras se cumplen las restricciones de presupuesto y posiciones. Se exploran métodos heurísticos para identificar soluciones óptimas y se presentan los resultados obtenidos.**

I. Introducción

La optimización de alineaciones de fútbol es un problema combinatorio con aplicaciones prácticas en analítica deportiva y toma de decisiones estratégicas. Dado el desafío de seleccionar 11 jugadores de un conjunto amplio respetando restricciones posicionales y presupuestarias, este problema es de naturaleza NP-hard, lo que exige el uso de métodos heurísticos para obtener soluciones viables en tiempo razonable.

II. El Problema

El problema planteado consiste en seleccionar 11 jugadores para formar un equipo de fútbol, a partir de un conjunto de jugadores disponibles. Cada jugador tiene asociados dos parámetros relevantes: una calificación que refleja su habilidad y un coste que contribuye al total del presupuesto. Además, es necesario que se respeten las posiciones en el campo, como portero, defensa, mediocentro y delantero, y se debe asegurar que la distribución de estos jugadores sea adecuada.

El objetivo final es encontrar una combinación de jugadores que maximice el rendimiento del equipo, dado por las calificaciones individuales, mientras se minimizan las penalizaciones por presupuesto excedido y las posiciones mal distribuidas.

La representación de la solución es clave para abordar este problema de optimización. La solución se puede representar como un conjunto de jugadores seleccionados, denotado como S, tal que:

$$S = \{i_1, i_2, \ldots, i_{11}\}$$

Donde cada i_j es un jugador seleccionado de la lista de jugadores disponibles. En este conjunto de 11 jugadores, deben cumplirse las siguientes condiciones:

- Los jugadores deben cubrir las posiciones en el campo de manera adecuada. Por ejemplo, debe haber un portero, varios defensas, mediocentros y delanteros, respetando las limitaciones del número de jugadores por posición.
- El total del presupuesto utilizado por los jugadores seleccionados debe ser menor o igual al presupuesto máximo disponible.
- Las calificaciones de los jugadores seleccionados deben maximizarse para mejorar el rendimiento global del equipo.

En esta representación, cada jugador tiene una calificación R_i y un coste asociado que contribuye al presupuesto total. La selección de los jugadores está sujeta a las restricciones de presupuesto y de distribución de posiciones en el campo.

La función objetivo se utiliza para evaluar la calidad de la solución S. En este caso, la función objetivo busca maximizar el rendimiento global del equipo, penalizando las soluciones que exceden el presupuesto o que tienen una distribución inadecuada de jugadores en las diferentes posiciones.

La fórmula general de la función objetivo es:

$$\text{objective\_function}(S) = \left(\sum_{i \in S} R_i\right) \times \text{penalización\_n\_posiciones} + \text{penalización\_n\_presupuesto}$$

Donde:

- $\sum_{i \in S} R_i$ es la suma de las calificaciones de los jugadores seleccionados. Esto refleja el rendimiento global del equipo.
- penalización_n_posiciones es una penalización asociada con la distribución de jugadores en las distintas posiciones. Si la distribución no es equilibrada, esta penalización aumenta.
- penalización_n_presupuesto es una penalización que se aplica si el presupuesto total supera el presupuesto máximo permitido.

La penalización por distribución de jugadores en las posiciones penalización_n_posiciones se calcula teniendo en cuenta las restricciones de cada posición:

- **Porteros (G)**: Se penaliza si no hay exactamente un portero seleccionado. Si hay más de uno o ninguno, se aplica una penalización reduciendo la puntuación final al multiplicarla por $0{,}1$.
- **Defensores (D)**: Se penaliza si el número de defensores está fuera del rango [3, 5]. Si hay menos de 3 o más de 5 defensores, la penalización será más severa, multiplicando la puntuación por $0{,}3$.
- **Mediocampistas (M)**: Se penaliza si el número de mediocampistas está fuera del rango [2, 5]. Al igual que con los defensores, si el número de mediocampistas es insuficiente o excesivo, la penalización es del $0{,}3$.
- **Delanteros (F)**: Se penaliza si el número de delanteros está fuera del rango [1, 5]. Si hay menos de 1 o más de 5, se aplica la penalización multiplicando la puntuación por $0{,}3$.

La penalización por presupuesto se calcula de la siguiente forma:

$$\text{penalización\_n\_presupuesto} = \begin{cases} 0 & \text{si suma\_presupuesto} \leq \text{max\_presupuesto} \\ 100 - \left(\frac{\text{suma\_presupuesto} \times 100}{\text{max\_presupuesto}}\right) & \text{si suma\_presupuesto} > \text{max\_presupuesto} \end{cases}$$

Donde suma_presupuesto es la suma total del coste de los jugadores seleccionados y max_presupuesto es el presupuesto máximo disponible. Si el presupuesto se excede, la penalización es mayor cuanto más se sobrepase el presupuesto.

Además de la función objetivo, también se definen tres funciones de vecindario que permiten generar nuevas soluciones a partir de una solución inicial. Cada una de estas funciones busca obtener una solución vecina que pueda mejorar el rendimiento global del equipo.

III. Propuesta de implementación

Para resolver el problema de optimización de asignación de jugadores en Football Eleven, se proponen tres algoritmos diferentes: Algoritmo Genético y Simulated Annealing, Algoritmo Genético y NSGA-II. A continuación, se detallan estos algoritmos.

III-A. Simulated Annealing

El algoritmo de Simulated Annealing es una técnica de optimización inspirada en el proceso en el cual un material se calienta hasta una temperatura elevada y luego se enfría lentamente para alcanzar un estado de baja energía; el algoritmo explora el espacio de soluciones y utiliza un parámetro de temperatura para controlar la aceptación de soluciones peores.

A continuación se presenta el pseudocódigo del Simulated Annealing:

Algorithm 1 Algoritmo de Simulated Annealing

```
Inicializar la solución actual y la mejor solución con la solución inicial
for por cada iteración do
    Generar vecindarios de la solución actual
    for por cada solución vecina do
        Calcular la diferencia de calidad entre la solución vecina y la actual
        if la solución vecina es mejor o la probabilidad de aceptación es alta then
            Aceptar la solución vecina como la solución actual
            if la solución actual es mejor que la mejor solución encontrada then
                Actualizar la mejor solución
            end if
        end if
    end for
    Reducir la temperatura
    if no se encontró mejora en la solución actual then
        Terminar el proceso
    end if
end for
Devolver la mejor solución encontrada
```

A continuación se detallan las funciones clave del algoritmo:

- **Generación de Vecindarios**: Dependiendo del parámetro $generation\_type$, el algoritmo genera un vecindario de soluciones utilizando dos enfoques posibles: `neighbourhood_same_position` o `all_random_neighborhood`. La primera genera vecinos intercambiando jugadores dentro de la misma posición, mientras que la segunda genera vecinos aleatorios en el espacio de soluciones.
- **Función de Energía**: La función de energía se calcula como la diferencia en el valor de la función objetivo entre la nueva solución y la actual. Si esta diferencia es positiva, se acepta la nueva solución sin importar la temperatura. Si la diferencia es negativa, la nueva solución se acepta con una probabilidad $P = \exp(\frac{energy}{temp})$, donde $temp$ es la temperatura actual del sistema.
- **Enfriamiento**: La temperatura se reduce en cada iteración utilizando la tasa de enfriamiento $cool\_rate$, lo que reduce la probabilidad de aceptar soluciones peores con el tiempo.

III-B. Algoritmo Genético

Para el algoritmo genético, en cada iteración, el algoritmo selecciona un conjunto de soluciones (individuos) de la población, las cruza para generar nuevas soluciones y, finalmente, las muta para introducir variación. Después, las soluciones resultantes se combinan con las soluciones anteriores y se seleccionan las mejores para continuar el proceso.

A continuación se presenta el pseudocódigo del algoritmo genético propuesto:

Algorithm 2 Algoritmo Genético

```
Inicializar población aleatoria pop de tamaño MAX_POPULATION
for generación = 1 hasta gen do
    Seleccionar mejores soluciones de pop
    Crear descendientes mediante cruce y mutación
    Combinar padres e hijos y seleccionar las mejores soluciones
end for
Devolver la mejor solución de pop
```

A continuación, se detallan las funciones clave del algoritmo genético:

- **Selección truncada** (`truncation_selection`): Esta función selecciona los k mejores individuos de la población. Se utiliza una selección truncada para limitar el número de soluciones que pasan a la siguiente generación.
- **Cruce de un solo punto** (`simple_crossover`): En el algoritmo se utiliza el operador de cruce simple, en el cual se combinan dos soluciones parentales para crear un nuevo individuo. En este caso, se separan los jugadores de ambas soluciones, se elige un portero aleatorio y luego se mezclan los jugadores restantes de ambas soluciones para formar el nuevo equipo.
- **Mutación**: Después del cruce, las soluciones hijas pueden ser mutadas para introducir variación. La probabilidad de mutación es controlada por la constante $MUTANT\_PROB$. En este caso, la mutación consiste en realizar intercambios entre jugadores de la misma posición.

- **Reemplazo**: Una vez que se han generado los hijos, las soluciones actuales y los hijos generados se combinan en una nueva población. A continuación, se seleccionan las mejores soluciones entre la población original y la nueva para continuar el proceso.

III-C. Algoritmo NSGA-II

El algoritmo NSGA-II es un algoritmo evolutivo para la optimización multiobjetivo basado en la dominancia de Pareto y la distancia de aglomeración, que favorece la diversidad en la población. A continuación se describen sus componentes clave y el pseudocódigo:

- **Dominancia de Pareto**: Evalúa si una solución $sol1$ domina a otra $sol2$ en todos los objetivos, siendo estrictamente mejor en al menos uno.
- **Clasificación en Frentes**: Clasifica las soluciones según su dominancia. Las soluciones no dominadas forman el primer frente, y las demás se agrupan por dominación.
- **Distancia de Aglomeración**: Calcula la distancia de dispersión dentro de un frente, favoreciendo soluciones más diversas.
- **Selección de Padres**: Selecciona soluciones basadas en la dominancia de Pareto y la distancia de aglomeración.
- **Cruce y Mutación**: Realiza cruce entre padres y aplica mutación con una probabilidad $MUTANT\_PROB$.
- **Creación de Nueva Generación**: Se combinan las soluciones de padres e hijos, y las mejores soluciones se mantienen para la siguiente generación.

Algorithm 3 Algoritmo NSGA-II

```
Inicializar población de soluciones
Establecer parámetros de generaciones y probabilidad de mutación
while no se ha alcanzado el número máximo de generaciones do
    Clasificar las soluciones por dominancia de Pareto
    Calcular las distancias de aglomeración entre soluciones en cada frente
    Seleccionar padres para cruce según dominancia y diversidad
    Aplicar cruce entre pares de padres para generar hijos
    Aplicar mutación a los hijos con una probabilidad definida
    Combinar padres e hijos en una nueva población
    Reordenar las soluciones y seleccionar las mejores para la siguiente generación
end while
Devolver la mejor solución encontrada
```

IV. EXPERIMENTACIÓN

Para garantizar la validez de los resultados, cada algoritmo se ha ejecutado 5 veces con un máximo de 10^5 iteraciones. Como referencia, se ha utilizado un algoritmo aleatorio (baseline) cuyos resultados son:

- Media del fitness: -10.67
- Varianza del fitness: 2452.58

En cuanto a los resultados del Algoritmo Genético, se ha desarrollado una búsqueda en cuadrícula (*grid search*) sobre los hiperparámetros *Population Size* y *Mutation Rate* para identificar las configuraciones que maximizan el fitness. Los resultados se resumen a continuación:

Population Size	Mutation Rate	Mean Fitness	Variance Fitness
500	0.01	79.724	0.060024
500	0.05	80.058	0.046576
500	0.10	79.974	0.667304
750	0.01	80.652	0.073656
750	0.05	80.176	0.174344
750	0.10	80.360	0.162080
1000	0.01	80.754	0.061624
1000	0.05	80.108	0.606776
1000	0.10	80.548	0.082936

Cuadro I
RESULTADOS DE LAS CONFIGURACIONES DE PARÁMETROS EN EL ALGORITMO GENÉTICO

Los siguientes gráficos reflejan la media y varianza del fitness para las diferentes configuraciones probadas:

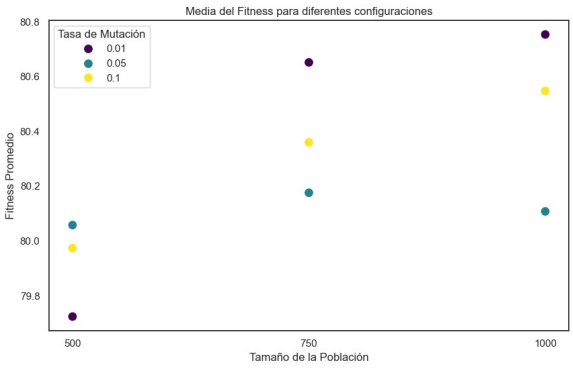

Figura 1. Media del Fitness con diferentes configuraciones del algoritmo genético.

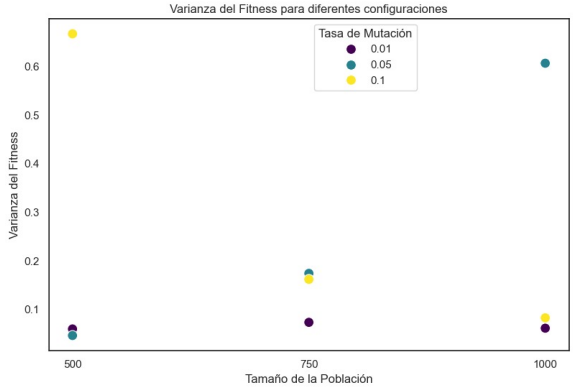

Figura 2. Varianza del Fitness con diferentes configuraciones del algoritmo genético.

La configuración óptima para este algoritmo es $PopulationSize = 1000$ y $MutationRate = 0{,}01$, obteniendo un fitness medio de 80.754 con una baja varianza de 0.062.

El algoritmo NSGA-II se evalúa en dos funciones objetivo: maximizar el rating de los jugadores y minimizar el coste total del equipo. Los resultados obtenidos son:

- **Fitness relacionado con el rating**: Media = -333.444, Varianza = 2609.33
- **Fitness relacionado con el presupuesto**: Media = 457,584,000€; Varianza = 2.61×10^{15} €

El algoritmo Simulated Annealing consiguió consistentemente un fitness de 81.33 en todas las ejecuciones, lo que demuestra una convergencia estable y efectiva hacia soluciones óptimas.

Probando con un algoritmo más simple como el Hill Climbing, con el enfoque del Best First, obtnemos el mismo fitness que con el algoritmo del Simulated Annealing; un fitness de 81.33 en todas las ejecuciones.

La siguiente tabla resume el rendimiento de los algoritmos evaluados. Excluimos el NSGA-II al ser un algoritmo multiobjetivo:

Algoritmo	Media del Fitness	Varianza del Fitness
Random Baseline	-10.67	2452.58
Genético	80.754	0.062
Simulated Annealing	81.33	0.0
Hill Climbing (Best First)	81.33	0.0

Cuadro II
COMPARATIVA DE LOS ALGORITMOS EN TÉRMINOS DE FITNESS.

Aquí, los gráficos relacionados con la media y varianza del fitness de los diferentes algoritmos:

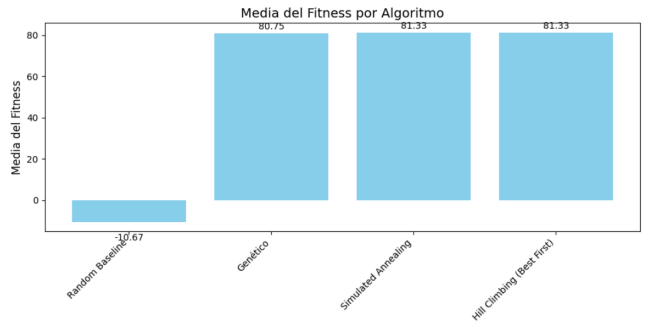

Figura 3. Media del Fitness de los diferentes algoritmos.

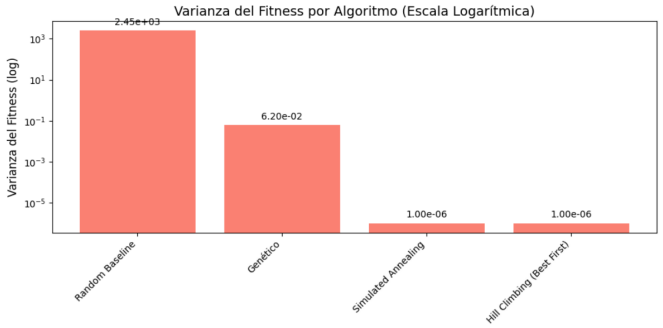

Figura 4. Varianza del Fitness de los diferentes algoritmos.

En cuanto al Random Baseline, la media indica un desempeño extremadamente bajo y no competitivo. El fitness negativo sugiere que este enfoque aleatorio produce soluciones muy deficientes. La varianza es extremadamente alta, lo cual denota una inconsistencia en los resultados esperada de un método sin optimización estructurada.

El Algoritmo Genético logra un desempeño notablemente superior al baseline, demostrando su capacidad de encontrar soluciones cercanas al óptimo. Sin embargo, no supera al Simulated Annealing ni al Hill Climbing, que consiguen una media del fitness ligeramente mayor (81.33). La varianza baja refleja estabilidad y confiabilidad en las soluciones que se producen.

El algoritmo Simulated Annealing, junto al Hill Climbing con Best First, logra el mejor resultado posible en términos de fitness medio. Su desempeño consistente en todas las ejecuciones indica que converge de manera efectiva hacia soluciones óptimas. La ausencia de varianza indica que el algoritmo encuentra siempre la misma solución óptima. Esto refuerza la idea de que es un método confiable y determinista para este problema específico.

V. CONCLUSIONES Y MEJORAS

Conclusiones

En términos generales, Simulated Annealing y Hill Climbing obtienen la mejor media del fitness (81.33), lo que les proclama como los algoritmos más efectivos para el problema. El Algoritmo Genético sigue de cerca, con una media de 80.754, lo que lo hace competitivo, especialmente considerando su flexibilidad para problemas más complejos. El Random Baseline es muy inferior, reforzando su rol únicamente como referencia.

Mejoras Propuestas

- **Hiperparametrización Avanzada:** Realizar búsquedas más exhaustivas en el espacio de hiperparámetros, incluyendo tasas de enfriamiento para Simulated Annealing y operadores de cruce más complejos para el Algoritmo Genético.
- **Considerar más factores del fútbol:** Incorporar dinámicas de equipo o lesiones para aproximar el modelo a escenarios del mundo real.
- **Híbrido entre Algoritmos:** Combinar los enfoques de Simulated Annealing y Algoritmo Genético para aprovechar la exploración global del segundo y la explotación local del primero.
- **Escalabilidad:** Probar los algoritmos con datasets más grandes para garantizar su rendimiento en problemas de mayor magnitud.