Algoritmo híbrido y evolutivo para la generación de un Ultimate Team en FIFA 19

20210836 *Piero Marcelo Pastor Pacheco*

20200445 *Aarón Ulises Santillán Huamán*

29 de junio de 2024

Resumen

El siguiente trabajo tiene como objetivo implementar un algoritmo genético para la obtención de un Ultimate Team en el videojuego FIFA 2019. El foco de esta investigación es obtener el mejor equipo que puede obtener un jugador con un presupuesto determinado, con este fin se irán minimizando factores negativos como los gastos y las edades, y maximizando otros positivos como la media en posición que jugará, potencial y química de todo el equipo en la adaptación. Para ello, se propuso una función objetivo en el cual se abarcan las variables anteriormente mencionadas y se adapta un algoritmo genético de Bin Packing para obtener un grupo de jugadores que satisfaga ser el mejor equipo. La meta es demostrar la efectividad de este algoritmo y su viabilidad en el videojuego FIFA, el cual puede ser mejorado.

1. Introdución

Actualmente, los videojuegos se han convertido en una de las grandes industrias del mundo moderno, los cuales han evolucionado enormemente para dar al jugador una experiencia única; desde juegos de aventura hasta de deportes, estos logran transportar al usuario a experiencias inolvidables y divertidas, por lo que muchos deciden invertir su dinero en estos, según Redes Sociales (2024) " Se trata de una industria que ha experimentado cambios significativos y que se estima que tiene unos ingresos superiores a los 180.000 millones de dólares en 2022". Tratándose de esta manera de una de las industrias que más mueven dinero en torno a la tecnología la cual se encuentra en un gran crecimiento. Dentro de este sector se encuentra FIFA 19, el cual es uno de los videojuegos más famosos en cuanto al fútbol, el cual trata de implementar una experiencia inmersiva no solo en jugar un partido de fútbol, sino también en el manejo de un equipo, teniendo un presupuesto con el cual se puede contratar jugadores y obtener la victoria, obteniendo de esta manera una experiencia más inmersiva al mundo del fútbol. De modo que nace la pregunta ¿Cómo puede un algoritmo genético permitir obtener los mejores jugadores para un Ultimate Team en el videojuego FIFA 2019?

Normalmente para la elaboración de un equipo para algún deporte, existe siempre la limitante del presupuesto. El algoritmo que se presentará a continuación trata de bajo un presupuesto por más mínimo que sea, conseguir generar un buen equipo de fútbol. Esto se llevará a cabo maximizando la media en las posiciones de jugadores, potencial de juego, y química; mientras que se buscará minimizar el precio para siempre ahorrar lo mayor posible, y la edad de los jugadores teniendo preferencia por los más jóvenes.

Además de solucionar este problema, se buscará comparar un algoritmo híbrido que sea la combinación de un genético evolutivo y un GRASP (Greedy Randomized Adaptive Search Procedure), con un algoritmo genético evolutivo puro.

Los resultados nos ofrecerán una comparación para saber si vale la pena el costo computacional extra de un GRASP dentro de un evolutivo, o si la diferencia de resultados es mínima y no es necesario; considerándose así un desperdicio la implementación del híbrido.

2. Trabajos relacionados

2.1. Heurísticas de agrupación híbridas eficientes para el problema de empacado de objetos en contenedores

El artículo aborda el clásico problema de empaquetado (Bin Packing Problem), pero mediante un enfoque del uso de un algoritmo genético híbrido de agrupación denominado HGGA-BP, el cual está inspirado en el esquema de representación de grupos de Falkenauer.
En el caso de este artículo, se tiene la restricción del peso de cada uno de los contenedores empleados para guardar paquetes de determinado peso, en nuestro caso, se tendrá el presupuesto (peso de dinero no podemos sobrepasar al igual que los contenedores.
Asimismo, emplea el siguiente pseudocódigo en la implementación:

```
1  Inicio
2     Inicializar parámetros: max_gen, L₂, gen₁, gen₂
3     Generar la población inicial con PI_D-A
4     mientras generación < max_gen y mejor_solución > L₂
5        Seleccionar individuos a cruzar en proporción a su aptitud
6        Aplicar Cruzamiento_BFD y generar nuevos_individuos
7        Sustituir los peores individuos por los nuevos_individuos
8        Aplicar Mutación_RV a los peores individuos
9        si generación > gen₁
10          Clonar los mejores individuos de la población
11          Aplicar Mutación_RpP a los individuos clonados
12          Sustituir los peores individuos por los individuos clonados
13       fin si
14       Aplicar Eliminación_por_Sustitución a individuos repetidos
15       si generación > gen₂
16          Seleccionar los mejores individuos para cruzar
17          Aplicar Cruzamiento_RpP para generar nuevos_individuos
18          Sustituir los peores individuos por los nuevos_individuos
19       fin si
20       Registrar la mejor_solución
21    fin mientras
22 fin procedimiento
```

Figura 1: Algoritmo genético híbrido de agrupación para BPP: HGGA-BP.

En esta imagen, se puede visualizar el algoritmo empleado para el artículo como también las operaciones genéticas para grupos como el cruzamiento, mutación y eliminación, los cuales son de gran utilidad al obtener las mejores soluciones.
La conclusiones de este trabajo es que el tiempo de ejecución del algoritmo propuesto es muy corto, la cual presenta una calidad de soluciones similar a los algoritmos del estado del arte.

2.2. Implementación de un algoritmo memético para optimizar la carga de hornos para la producción de sanitarios

El presente trabajo tiene como objetivo la implementación de un algoritmo memético para optimizar la carga de hornos de cerámicos frente a la continua competencia entre las empresas de este sector. En este se plantea la comparación entre este algoritmo y uno genético para el problema de la selección de piezas en la producción de sanitarios, el cual tiene restricciones como el peso máximo de las vagonetas, volumen máximo del horno, etc.
Para este, se plantea el siguiente pseudocódigo:

```
Algoritmo_Memetico (poblacionInicial)
Inicio
 1   t ← Inicializar temporizador
 2   generacion ← 0
 3   sinMejora ← 0
 4   pop ← poblacionInicial
 5   mejorSol ← obtenerMejor(pop)
 6   Repetir
 7       nuevaPoblacion ← GenerarNuevaPoblacion(pop)
 8       pop ← ActualizarPoblacion(pop, nuevaPoblacion)
 9       mejorSolActual ← obtenerMejor(pop)
10       Si fitness(mejorSol) >= fitness(mejorSolActual) entonces
11             sinMejora ← sinMejora+1
12       Si no
13             mejorSol ← mejorSolActual
14             sinMejora ← 0
15       Fin Si
16       Si (sinMejora = MaxSinMejora) entonces
17             pop ← RestaurarPoblacion(pop)
18       Fin Si
19       generacion ← generacion +1
20   Hasta generacion > MaxGeneraciones o t > tiempoLímite
21   Retornar(mejorSol)
Fin
```

Figura 2: Pseudocódigo del algoritmo memético.

Este trabajo plantea la generación de una población inicial usando el algoritmo GRASP, el cual aportará en gran medida al algoritmo memético en la obtención del mejor resultado, en el cual al ser calibrado se obtiene un alpha de valor 0.4. Las conclusiones que este trabajo presenta, es que el algoritmo memético logra dar mejores soluciones que el algoritmo genético, no obstante, no es lo suficientemente significativa, por lo que se considera que ambas logran dar soluciones igualmente buenas.

2.3. A distributed Chromosome genetic algorithm for bin-packing

Este trabajo plantea a los algoritmos genéticos como un enfoque muy bueno en la resolución de problemas de optimización, en el cual se detalla el desarrollo de un cromosoma GA distribuido para reducir el tiempo de ejecución del GARP el cual tiene como fin el resolver el problema de empaquetamiento de cilindros de construcción creados por la Universidad de Louisville en contenedores de Sinterstation 2000. Para esto, se utilizan algoritmos genéticos distribuidos con el fin de reducir el tiempo de ejecución, para ello, se tiene en cuenta la sobrecarga de comunicación así como la sincronización. Se debe de tener conocimiento, que el empaquetado se realiza en contenedores tridimensionales, en el cual el algoritmo requiere un uso computacional intensivo y más cuando se trata de objetos definidos por archivos de modelo sólido o estereolitografía, para lo cual utilizan un gran cantidad de parámetros al ser analizadas. En la conclusión demuestran que el uso de un cromosoma GA distribuido disminuye el tiempo de ejecución, no obstante los resultados obtenidos del algoritmo son buenos, aún así, los autores plantean que una mejora se haría mediante una configuración del hardware.

3. Metodología

3.1. Descripción del problema

El artículo busca solucionar el problema de la generación de equipos de fútbol, pero al mismo tiempo busca realizar un análisis a los resultados de dos tipos de algoritmos metaheurísticos.

El algoritmo híbrido buscará mejorar mediante la selección, la media y química del equipo; minimizando gastos para que así el usuario pueda seleccionar más adelante a un jugador de élite de su gusto (en caso el algoritmo no lo haya colocado ya en su equipo). El GRASP se verá reflejado en la generación de una población inicial, que si bien puede ser randonomica, esta siempre buscará maximizar el fitness tanto del jugador individual como del equipo general. El resto del algoritmo evolutivo mantiene los estándares comunes, implementando una función que elimina las aberraciones, otra que se encarga de los clones, y una última que con cada generación limita la población disminuyéndola en un porcentaje; si bien esta última selección es randonomica, siempre da preferencia a eliminar a la población con peor fitness.
El algoritmo evolutivo puro, trabaja de la misma manera que el híbrido en el ámbito que corresponde a un algoritmo genético. La única diferencia es la forma en que se genera la población inicial, este algoritmo la genera randonomicamente, solo tiene como restricción el no crear aberraciones.

3.2. Entrada y Salida

Ambos algoritmos recibirán un grupo de jugadores del juego, aquí se encontrará por jugador los siguientes valores:

Atributo	Tipo de dato	Descripción
id	Integer	ID del jugador
nombre	String	Nombre del jugador
nacionalidad	String	Nacionalidad del jugador
posicion	String	Posición principal del jugador
media	Integer	Media general del jugador
potencial	Integer	Potencial de juego del jugador
club	String	Club del jugador
valor	Double	Valor en M€
mediasPos	Map¡String, Integer¿	Media de juego en cada posición del campo

Cuadro 1: Atributos de un jugador.

Se utilizaron varias posiciones, para que si una posición ya está tomada, pero si este jugador se desenvuelve igual de bien en otra; pueda ser enviado a esa.
La entrada por parte del usuario es únicamente que formación se utilizará y que presupuesto es el que posee para poder armar su equipo.
La salida es el equipo formado en cada posición y el fitness que generó este individuo en la población.

3.3. Constantes

Las constantes seleccionadas se consiguieron tras una calibración que evaluaba el fitness promedio entre ejecuciones. Este proceso se explicará a mayor detalle en la parte de experimentación. Es importante hacer inciso en que las únicas constantes que han sido calibradas son las que corresponden a porcentajes de la data; tanto $POBINICIAL$, como $GENERACIONES$ no han sido calibrados, ya que, únicamente son valores de inicio o topes.

3.3.1. Metaheurístico Híbrido

Las constantes utilizadas para maximizar el fitness en el algoritmo híbrido entre evolutivo y GRASP son:

Constante	Valor	Descripción
Tcasamiento	0.7	Que porcentaje de la población serán padres
Tmutacion	0.0	Que porcentaje de los padres mutarán
Tmuerte	0.1	Que porcentaje de la población se eliminará
Pcasamiento	0.4	Que porcentaje le corresponderá del padre al nuevo cromosoma
POBINICIAL	50	De que tamaño será la población que se generará
GENERACIONES	100	Cantidad de generaciones máximas que realizará el algoritmo
ALPHA	0.3	Porcentaje que se tomará para optimizar los resultados del GRASP

Cuadro 2: Constantes del híbrido

3.3.2. Genético Evolutivo

Las constantes utilizadas para maximizar el fitness en el algoritmo evolutivo puro son:

Constante	Valor	Descripción
Tcasamiento	0.7	Que porcentaje de la población serán padres
Tmutacion	0.0	Que porcentaje de los padres mutarán
Tmuerte	0.1	Que porcentaje de la población se eliminará
Pcasamiento	0.8	Que porcentaje le corresponderá del padre al nuevo cromosoma
POBINICIAL	50	De que tamaño será la población que se generará
GENERACIONES	100	Cantidad de generaciones máximas que realizará el algoritmo

Cuadro 3: Constantes del evolutivo puro

3.4. Algoritmos empleados

A continuación se tendrán los pseudocódigos de cada función que tenga alta relevancia empleada para los algoritmo, además de la programación metódica que demostrará el correcto funcionamiento del fitness.

3.4.1. Metaheurístico Híbrido

Metaheurístico Híbrido Algoritmo para encontrar el mejor equipo en el videojuego FIFA 19

```
1: procedure HIBRIDO(jugadores, n, presupuesto, posiciones, quimica)  ▷ Halla el mejor equipo en base al presupuesto
2:     GRASP(jugadores, poblacion, n, presupuesto, posiciones, quimica, POBINICIAL)
3:     i ← 0
4:     while i ≠ GENERACIONES ∧ tampoblacion > 1 do
5:         SELECCION(poblacion, padres, jugadores, n, posiciones, quimica)
6:         CASAMIENTO(poblacion, padres)
7:         MUTACION(poblacion, padres, jugadores)
8:         ELIMINARABERRACIONES(poblacion, jugadores, presupuesto)
9:         ELIMINARCLONES(poblacion)
10:        DISMINUIRPOBLACION(poblacion, jugadores, n, posiciones, quimica)
11:        i = i + 1
12:    end while
13:    SELECCION(poblacion, padres, jugadores, n, posiciones, quimica)
14:    return RANDOM(padres)  ▷ Retorna el mejor equipo
15: end procedure
```

FitnessIndividuo Algoritmo para encontrar el mejor equipo en el videojuego FIFA 19

```
1: procedure FitnessIndividuo(equipo, posiciones, quimica)  ▷ Halla el fitness de todo el equipo
2:     fo_max ← Media(equipo[0]) * Potencial(equipo[0])
3:     fo_min ← Valor(equipo[0]) * Edad(equipo[0])
4:     i ← 1
5:     while i < N_PLAYERS do
6:         fo_max ← fo_max + MediaPosicion(equipo[i]) * Potencial(equipo[i])
7:         fo_min ← fo_min + Valor(equipo[i]) * Edad(equipo[i])
8:         i = i + 1
9:     end while
10:    i ← 0
11:    QuimParcial ← 0
12:    while i < N_CHEM do
13:        j ← 0
14:        while j < N_CHEM do
15:            if quimica[i][j] ≠ −1 then
16:                index ← quimica[i][j]
17:                Cant_Relaciones ← Cant_Relaciones + 1
18:                if Nacionalidad(equipo[i]) = Nacionalidad(equipo[index]) ∧ Club(equipo[i]) =
   Club(equipo[index]) then
19:                    QuimParcial ← QuimParcial + 200
20:                else
21:                    if Nacionalidad(equipo[i]) = Nacionalidad(equipo[index]) ∨
   Club(equipo[i]) = Club(equipo[index]) then
22:                        QuimParcial ← QuimParcial + 100
23:                    else
24:                        QuimParcial ← QuimParcial + 60
25:                    end if
26:                end if
27:            else
28:                break
29:            end if
30:            j = j + 1
31:        end while
32:        i = i + 1
33:    end while
34:    QuimParcial ← QuimParcial ÷ (Cant_Relaciones × 100)
35:    if QuimParcial > 1 then
36:        QuimParcial ← 1
37:    end if
38:    return (fo_max × QuimParcial) ÷ (fo_min × PER_MIN)
39: end procedure
```

3.4.2. GRASP

Metaheurístico GRASP Algoritmo para elaborar la población inicial del híbrido

```
1: procedure GRASP(jugadores, poblacion, n, presupuesto, posiciones, quimica, requerido)  ▷ Halla una población inicial aceptable
2:     i ← 0
3:     while i ≠ ITERACIONES do
4:         foparcial ← CONSTRUCCION(mediasPosicion, pobParcial, n, presupuesto, posiciones, quimica)
5:         foparcial ← foparcial* HALLARQUIMICA(pobParcial, jugadores, quimica)  ▷ Halla la química de todo el equipo, ya que, la construcción no puede hacerlo por si sola
6:         ACTUALIZARMEJORES(poblacion, pobParcial, mejoresFo, foParcial, requerido)  ▷ Actualiza la población inicial con el nuevo individuo conseguido
7:         i ← i + 1
8:     end while
9: end procedure
```

Metaheurístico GRASP Construcción de la población inicial

```
1: procedure CONSTRUCCION(MediaDePosicion, candidato, n, presupuesto, posiciones, chem_pos)
2:     fitness ← 0
3:     borrados ← lista vacía
                                                  ▷ Análisis exclusivo de arqueros
4:     while true do
5:         beta ← Mejor(MediaDePosicion["GK"])
6:         tau ← 0
7:         maxrcl ← beta − ALPHA × (beta − tau)
8:         indmax ← beta − maxrcl
9:         inda ← (si indmax > 0) ? aleatorio % indmax : 0
10:        inda ← inda + maxrcl
11:        JugadorActual ← MediaDePosicion[posiciones[0]]
12:        if presupuesto ≥ VALOR(JugadorActual) then
13:            presupuesto ← presupuesto − VALOR(JugadorActual)
14:            candidato[0] ← JugadorActual
15:            fitness ← fitness + FITNESS(JugadorActual)
16:            break
17:        end if
18:    end while
19:
20:    for i ← 1 to N_PLAYERS − 1 do
21:        beta ← Mejor(MediaDePosicion[Posiciones[i]])
22:        tau ← 0
23:        maxrcl ← beta − ALPHA × (beta − tau)
24:        indmax ← beta − maxrcl
25:        inda ← (si indmax > 0) ? aleatorio % indmax : 0
26:        inda ← inda + maxrcl
27:        JugadorActual ← MediaDePosicion[posiciones[i]]
28:        if presupuesto ≥ VALOR(JugadorActual) ∧ Posicion(MediaDePosicion[Posiciones[i]]) ≠ "GK" then
29:            presupuesto ← presupuesto − VALOR(JugadorActual)
30:            candidato[i] ← JugadorActual
31:            fitness ← fitness + FITNESS(JugadorActual)
32:        else
33:            i ← i − 1                          ▷ Tiene que volver a analizar la posición
34:        end if
35:        BORRAR(JugadorSeleccionado)
36:    end for
37:    return fitness
38: end procedure
```

3.4.3. Genético Evolutivo

Metaheurístico GRASP Algoritmo para encontrar el mejor equipo en el videojuego FIFA 19

```
1: procedure HIBRIDO(jugadores, n, presupuesto, posiciones, quimica)  ▷ Halla el mejor equipo en base al presupuesto
2:     GenerarPoblacion(jugadores, poblacion, n, presupuesto, posiciones, quimica, POBINICIAL)
3:     i ← 0
4:     while i ≠ GENERACIONES ∧ tampoblacion > 1 do
5:         Seleccion(poblacion, padres, jugadores, n, posiciones, quimica)
6:         Casamiento(poblacion, padres)
7:         Mutacion(poblacion, padres, jugadores)
8:         EliminarAberraciones(poblacion, jugadores, presupuesto)
9:         EliminarClones(poblacion)
10:        DisminuirPoblacion(poblacion, jugadores, n, posiciones, quimica)
11:        i ← i + 1
12:    end while
13:    Seleccion(poblacion, padres, jugadores, n, posiciones, quimica)
14:    return Random(padres)                                  ▷ Retorna el mejor equipo
15: end procedure
```

3.4.4. Programación Metódica Fitness

G: $(\forall\ i : 1 \leq i < n : Posicion(equipo[i]) \neq "GK")$

$\{Pre : N = 11 \wedge equipo \neq \emptyset\ \wedge\ posiciones \neq \emptyset\ \wedge\ quimica \neq \emptyset\ \wedge\ Posicion(equipo[0]) = "GK" \wedge\ \mathbf{G}\}$

proc $HallarFitness$(**value** $equipo$: **array** **of** $Equipo$;
value $posiciones$: **array** **of** $string$;
value $quimica$: **array** **of array** **of** $integer$);
result $fitness$: $double$);

$fo\_max := Media(equipo[0]) \times Potencial(equipo[0])$

$fo\_min := Valor(equipo[0]) \times Edad(equipo[0])$

$i := 0$

$\{invariant : fo\_max > 0 \wedge fo\_min > 0\}$

$\{bound : N - 1\}$

do $i < N \rightarrow fo\_max := fo\_max + Media(equipo[i]) \times Potencial(equipo[i]), fo\_min := fo\_min +$
$Valor(equipo[i]) \times Edad(equipo[i]), i := i + 1$

od

$i, QuimParc, Cant\_Relaciones := 0$

$\{invariant : 0 \leq QuimicaParc\}$

$\{bound : N - 2\}$

do $i < N - 1 \rightarrow j := 0,$
$\{bound : N - 2\}$
do $j < N - 1 \rightarrow$
if $quimica[i][j] \neq -1 \rightarrow index := quimica[i][j], Cant\_Relaciones :=$
$Cant\_Relaciones + 1,$
if $Nacionalidad(equipo[i]) = Nacionalidad(equipo[index]) \wedge$
$Club(equipo[i]) = Club(equipo[index]) \rightarrow QuimParc :=$
$QuimParc + 200$
$\square\ Nacionalidad(equipo[i]) = Nacionalidad(equipo[index]) \vee$

$Club(equipo[i]) = Club(equipo[index]) \rightarrow QuimParc := QuimParc + 100$

$\square$ **True** $\rightarrow QuimParc := QuimParc + 65$

fi

$\square\ quimica[i][j] = -1 \rightarrow j := N - 1$

fi, $j := j + 1$

od, $i := i + 1$

od

$\{postcondition : 65 \leq QuimicaParc\}$

$QuimicaParc := QuimicaParc \div (Cant\_Relaciones \times 100)$

if $QuimcaParc > 1 \rightarrow QuimParc := 1$ **fi**

$fitness := (fo\_max \times QuimicaParc) \div (fo\_min \times PER\_MIN)$

$\{Post : fitness > 0\}$

Otra forma de entenderlo es:

G: $(\forall\ i : 1 \leq i < n : Posicion(equipo[i]) \neq "GK")$

Q: $\{N = 11 \wedge equipo \neq \emptyset\ \wedge\ posiciones \neq \emptyset\ \wedge\ quimica \neq \emptyset\ \wedge\ Posicion(equipo[0]) = "GK" \wedge\ \mathbf{G}\}$

R: $\{fitness > 0\}$

Q $\implies wp(HallarFitness, \mathbf{R})$

4. Experimentación y Resultados

4.1. Calibración

Para la calibración de constantes se utilizaron varias ejecuciones las cuales se fueron almacenando en archivos ".csv". Y de esta forma poder tener almacenado el histórico de cada valor que se cambiaba para tener un máximo preciso. Dichos archivos se encuentran en el repositorio del proyecto.

4.2. Comparación

Para la evaluación de los algoritmos y conocer tanto sus resultados independientes, como una comparación que nos pueda indicar que proceso es el que proporciona mejores resultados; se realizará una ejecución con la cual podamos tener 100 fitness resultado de cada uno de los algoritmos para posteriormente tener gráficas estadísticas y así poder sacar conclusiones. Estas gráficas se consiguieron utilizando el lenguaje de programación Python.

4.2.1. Gráfico de Dispersión

Se utilizó este gráfico para poder conocer a simple vista utilizando puntos el algoritmo que tiende a dar mejores resultados con más frecuencias, incluyendo los picos de cada uno.

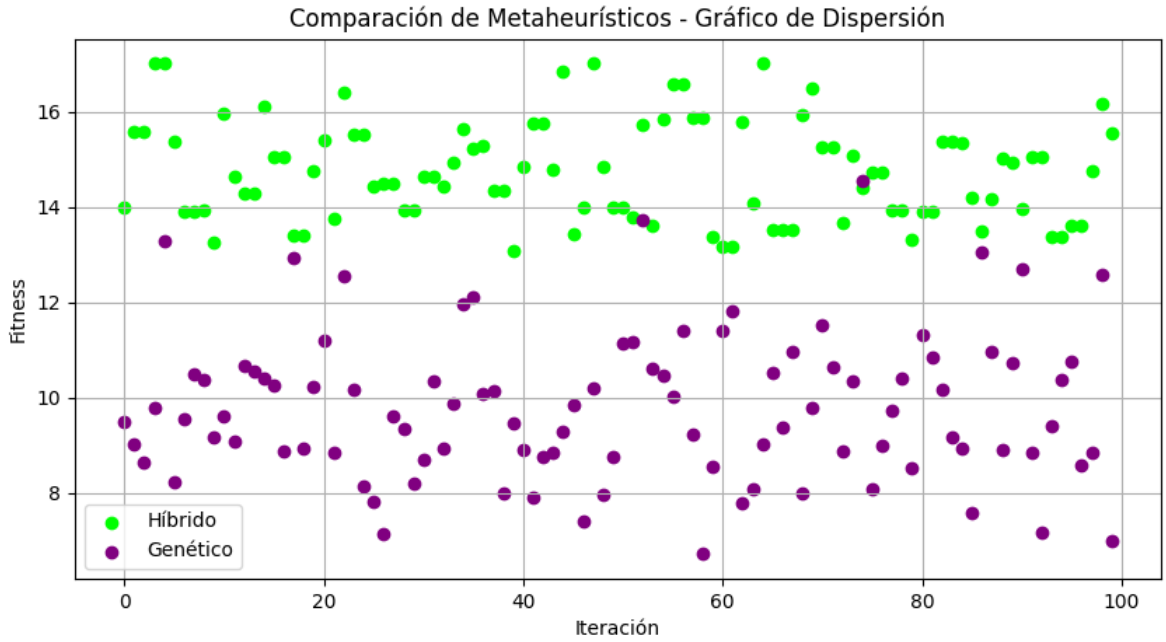

Figura 3: Gráfico de dispersión - Autoría propia

4.2.2. Gráfico Lineal

El gráfico lineal resultó útil para unir todos los puntos del gráfico de dispersión y poder conocer como eran las variaciones del fitness entre ejecuciones.

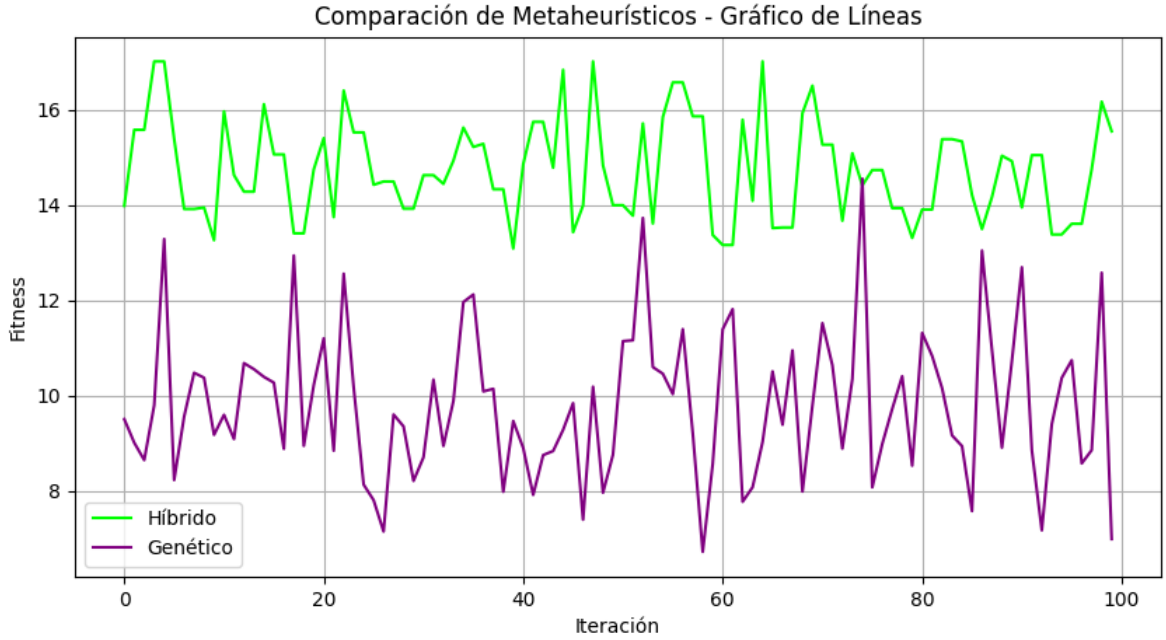

Figura 4: Gráfico lineal - Autoría propia

4.2.3. Gráfico BoxPlot

Con este gráfico podemos analizar los percentiles más importantes de los resultados, así como saber que datos atípicos existen por parte de cada algoritmo.

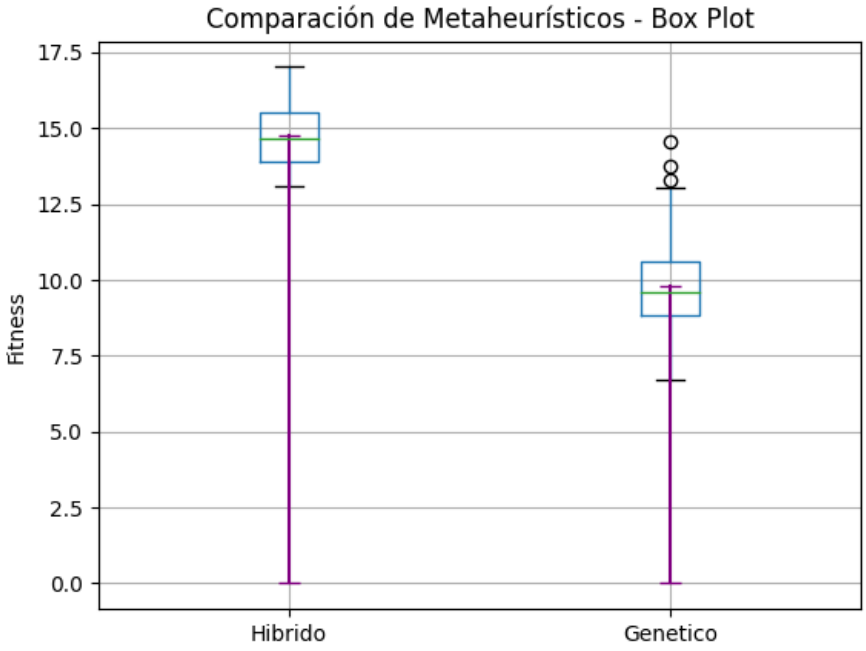

Figura 5: Gráfico BoxPlot - Autoría propia

4.2.4. Gráfico Histograma

Este cuarto gráfico es útil para saber un promedio aproximado del resultado que consiguieron cada una de las iteraciones, además de que tan probable es que consigan datos muy altos (atípicos).
Este último análisis es el más importante junto con el BoxPlot porque nos permiten de manera más directa analizar los promedios y que porcentaje es la probabilidad que se consigan resultados excelentes. Además de poder analizar en que algoritmo los resultados son muy variados, es decir, que pueden dispararse tanto como para resultados esperados, como a los peores.

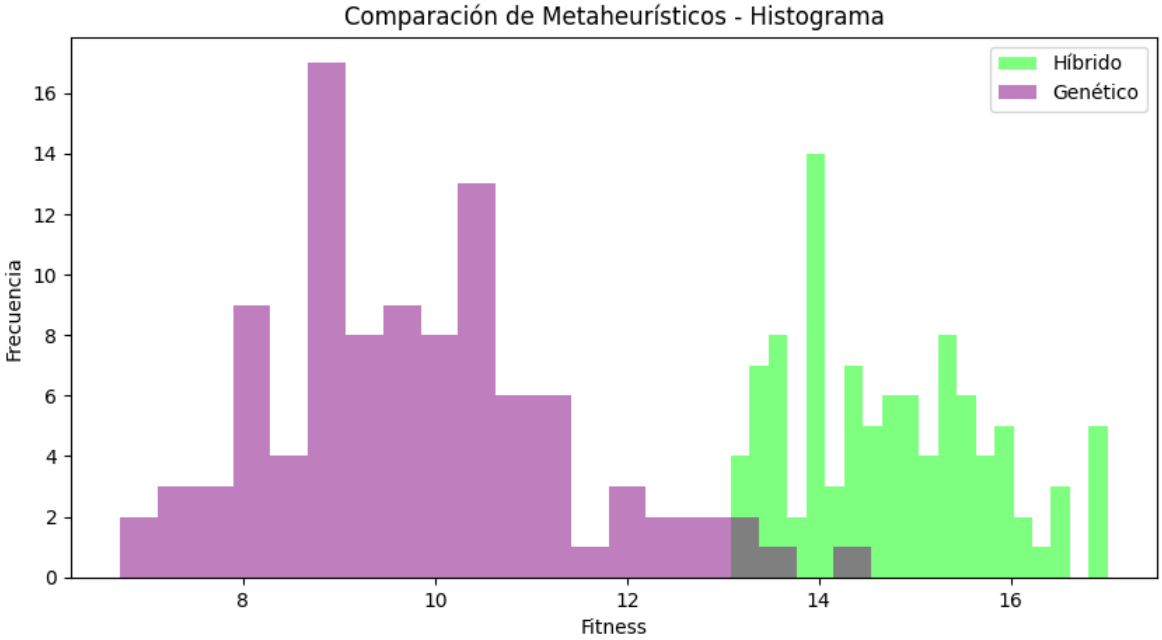

Figura 6: Gráfico Histograma - Autoría propia

5. Conclusión

De acuerdo con lo señalado y los resultados mostrados, se puede concluir principalmente que ambos algoritmos logran dar una respuesta ante la generación de un Ultimate Team para un jugador que cuenta con un presupuesto determinado. No obstante, según las gráficas se puede visualizar que mediante el algoritmo meta heurístico híbrido se logra obtener mejores resultados, ya que los resultados en 100 iteraciones suelen rondar entre los 13 y 17 de fitness promedio, mientras que el algoritmo genético evolutivo suele originar fitness entre 7 y 13, y aunque en pocas ocasiones se logra obtener fitness de 14, estos son valores atípicos.
En la prueba de calibraciones de las tasas para ambos algoritmos se pudo observar que es también el algoritmo meta heurístico híbrido el que logra dar un resultado en el menor tiempo, ya que el tiempo promedio de salida rondaba en 1 minuto con 30 segundos. Por otro lado, el algoritmo genético evolutivo daba una respuesta promedia de 10 minutos, llegando incluso a 1 hora con ciertas tasas.
Esto puede estar estrechamente relacionado con cómo está construido el algoritmo, ya que para ambos algoritmos hay una fase de eliminación de aberraciones al generar población, y el híbrido al utilizar GRASP suele tener poblaciones que no requieran tomar el proceso de eliminación ya que no son aberradas. Por otro lado, el algoritmo genético evolutivo al generar poblaciones aleatorias suele tener un mayor uso de esta función, incrementando así el uso de esta función tanto en la generación de poblaciones como en las posteriores funciones como el eliminado por menor fitness y una última eliminación de aberraciones.
De este modo, se puede concluir que el uso de un algoritmo meta heurístico híbrido logra emitir mejores y más rápidos resultados en cuanto a la generación del Ultimate Team de FIFA 19, los cuales pueden ayudar al jugador en cuanto a la decisión de generación de equipos al momento de jugar.

6. Sugerencia de trabajos futuros

Principalmente, una sugerencia debería ser tener un periodo de calibración de constantes más extenso, ya que aunque las tasas tanto de casamiento, mutación, eliminación y probabilidad de casamiento duraron 15 horas en ser calibradas, se puede mejorar la precisión en estas constantes si el periodo fuera más extenso. Asimismo, mediante estas calibraciones se encontró que la mejor tasa de mutación es de 0. Lo cual para futuros trabajos puede ser eliminado de las funciones ya que posiblemente la mutación al ser aleatoria esté generando peores resultados.
Otra sugerencia a tomar está en cuanto a los datos, este trabajo fue realizado para FIFA 19, un juego que ya lleva varios años desde su lanzamiento, en el cual ya hay múltiples juegos más actuales, se puede sugerir tomar los datos más actuales ya que estos pueden ser de mayor beneficio ya que es donde se encuentra actualmente la comunidad.
Por último, este trabajo tomó en cuenta un .ªlpha"de 0.4 para la implementación del GRASP, esto debido al trabajo relacionado "Implementación de un algoritmo memético para optimizar la carga de hornos para la producción de productos sanitarios". No obstante, esta constante es un valor que debe ser calibrado para futuros trabajos.

7. Implicancias éticas

Las implicaciones éticas giran en torno a la forma en cómo se desarrolla el juego en la actualidad. En la que se contempla tres factores principales:

7.1. Equidad y Competitividad

Si se comparte o vende el algoritmo, esto podría afectar seriamente a las partidas en línea con las que cuenta FIFA 19, en las que algunos jugadores pueden considerar injusto el uso de un algoritmo para obtener el Ultimate Team ya que el juego actualmente se basa en la experiencia manual. Esto podría solucionarse al no dar directamente el algoritmo a la comunidad sino a la empresa del juego "EA Sports", los cuales pueden implementarlo como una función dentro del juego para que cualquier jugador pueda utilizarlo si lo desea.

7.2. Economía

Esto ya que el algoritmo al encontrar a los mejores jugadores podrían solo concentrarse en comprar y vender jugadores en específico, desequilibrando el mercado dentro del juego. Esto se podría solucionar al afectar un poco el algoritmo para que logre emitir una solución promedio, que no siempre sea la mejor pero tampoco la peor, dando al jugador la perspectiva de que es una máquina de sugerencia, más para la decisión de formar el Ultimate Team es el criterio propio.

7.3. Transparencia

Si se comparte o vende el algoritmo, se debe ser transparente con los jugadores y la empresa de cómo realmente funciona, ya que algunos jugadores podrían tomarlo como si fuera trampa, y si la empresa determina que efectivamente es trampa podría llegar a convertirse en un software ilegal, lo cual podría escalar a términos legales. Por ello, siempre se debe ser transparente y honesto en cuanto a cómo funcionan los algoritmos.

8. Repositorio de GitHub

Repositorio - Algoritmo híbrido y evolutivo para la generación de un Ultimate Team en FIFA 19

9. Declaración de contribución de cada integrante

9.1. Aporte de Alumno Santillán Huamán, Aarón Ulises - 20200445

Realizó la introducción del informe, así como sintetizar los trabajos relacionados para obtener tasas y constantes iniciales para la calibración de los algoritmos.
Además, se encargó de la búsqueda y calibración de las constantes de tasa y probabilidad utilizadas en los algoritmos. Asimismo, realizó las diapositivas para la exposición en aula.
Por último, redactó las conclusiones del informe e Implicancias éticas de este.

9.2. Aporte de Alumno Pastor Pacheco, Piero Marcelo - 20210836

Realizó los algoritmos, además de sus respectivos pseudocódigos y la programación metódica para el fitness.
De igual manera realizó la experimentación y comparación entre ambos algoritmos, esto utilizando gráficas estadísticas.
Por último redactó el reporte en Latex.

10. Referencias

Cruz-Reyes, L., C, M. Q., Alvim, A. C. F., Huacuja, H. J. F., S, C. G., Torres-Jiménez, J. (2012). Heurísticas de agrupación híbridas eficientes para el problema de empacado de objetos en contenedores. Redalyc.org. https://www.redalyc.org/articulo.oa?id=61524403009

Bankinter. (12 de enero de 2024). Estudio mercado videojuegos: ¿Cuánto dinero mueve? (infografía). Bankinter. https://www.bankinter.com/blog/finanzas-personales/mercado-videojuegos-dinero-estudio-espana-mundo

Malgrejo P. Melgarejo N. (2019, 28 octubre). Implementación de un algoritmo memético para optimizar la carga de hornos para la producción de sanitarios. http://hdl.handle.net/20.500.12404/15273

Lewis, J. E., Ragade, R. K., Kumar, A., Biles, W. E. (2005). A distributed chromosome genetic algorithm for bin-packing. Robotics And Computer-integrated Manufacturing, 21(4-5), 486-495. https://doi.org/10.1016/j.rcim.2004.11.017