Problema CalDep

Jean Carlos Lerma Rojas
Tecnología en Desarrollo de software
Universidad del Valle
Tuluá, Colombia
jean.lerma@correounivalle.edu.co

Juan Camilo Garcia Saenz
Tecnología en Desarrollo de software
Universidad del Valle
Tuluá, Colombia
juan.garcia.saenz@correounivalle.edu.co

Steven Giron Arcila
Tecnología en Desarrollo de software
Universidad del Valle
Tuluá, Colombia
steven.giron@correounivalle.edu.co

***Abstract*— This project addresses the challenge of creating efficient schedules for sports tournaments, incorporating real-world constraints. The focus is on applying programming concepts to analyze the problem, employing various programming techniques, selecting a programming language, and implementing data structures for different solution alternatives.**

The objective is to consider efficiency and proximity to the optimum solution in the analysis of these alternatives. The underlying issue revolves around the creation of schedules that adhere to basic requirements, such as an even number of teams, matches between home and away teams, and adherence to round-robin structures. Additional constraints include minimizing travel distances, avoiding consecutive home or away matches, and optimizing overall scheduling efficiency.

The formal definition of the problem involves creating valid schedules for tournaments with specific constraints on the size of tours and home stays while minimizing the total cost of tours. The distances between team locations are represented in a distance matrix, and the goal is to find a schedule matrix that satisfies the constraints.

Keywords—Round-Robin, optimization, efficiency, constraints, data structures, genetic algorithm.

I. Introducción

El presente proyecto aborda la compleja tarea de diseñar calendarios eficientes para torneos deportivos, confrontando a los estudiantes con desafíos prácticos y conceptos aprendidos en el curso. Se enfoca en el análisis de un problema real que involucra la programación de eventos deportivos, considerando aspectos de eficiencia y optimización.

La problemática central se centra en la creación de calendarios que satisfagan condiciones fundamentales, como un número par de equipos, la realización de partidos entre equipos locales y visitantes, y la adherencia a estructuras de tipo round-robin. Además, se incorporan restricciones prácticas, como la minimización de distancias de desplazamiento, la alternancia de partidos locales y visitantes, y la gestión de la dificultad de los encuentros para cada equipo.

La complejidad de la tarea radica en la búsqueda de soluciones que cumplan con todas estas restricciones y condiciones, especialmente cuando se trabaja con un número significativo de equipos, como es común en torneos nacionales o internacionales. La resolución eficiente de problemas de esta magnitud se convierte en un desafío computacional de gran interés, siendo este proyecto una oportunidad para aplicar y poner a prueba los conocimientos adquiridos en el curso.

El objetivo final es desarrollar soluciones prácticas y eficientes que puedan ser aplicadas para la solución de este tipo de problemas computacionales, considerando tanto la complejidad del problema como la viabilidad de implementación en entornos reales.

II. Análisis

A. *Algoritmo ingenuo*

El algoritmo empleado para generar el calendario de partidos opera mediante fuerza bruta, careciendo de estrategias de optimización específicas. Su ejecución puede elevarse en complejidad, especialmente al considerar los bucles anidados y su impacto en el rendimiento.

Complejidad Temporal: El método principal, createCalendarSolution(), incluye un bucle do-while que se ejecuta hasta que isValid() devuelve true. Dentro de este bucle, un for se repite calendarSolution.length veces. En cada iteración, se llama a generateRow(), que a su vez ejecuta un bucle for rowGenerated.length veces.

Esto se traduce en una complejidad temporal aproximada de $O(n^3)$, donde 'n' es el número de equipos. La naturaleza anidada de los bucles implica un aumento exponencial en el tiempo de ejecución a medida que crece 'n'.

Complejidad Espacial: La estructura de datos principal, calendarSolution, es una matriz 2D con dimensiones de 2 * (teams - 1) por teams. Esta matriz ocupa la mayor parte de la memoria, contribuyendo a una complejidad espacial de $O(n^2)$, donde 'n' representa el número de equipos.

B. *Algoritmo optimizado.*

Se eligió implementar un algoritmo genético en lugar de depender únicamente de un enfoque ingenuo para abordar el problema de la generación de un calendario de partidos. La decisión de utilizar un algoritmo genético se basa en la necesidad de buscar soluciones más eficientes y óptimas en un espacio de búsqueda complejo.

Las ventajas que ofrece el algoritmo genético para este contexto son:

- Exploración Efectiva del Espacio de Búsqueda: Al mantener una población diversa de posibles soluciones (calendarios de partidos), el algoritmo genético puede explorar una gama más amplia de opciones en paralelo.
- Explotación de Soluciones Prometedoras: A través de operadores genéticos como el cruce y la mutación, el algoritmo genético puede combinar y mejorar soluciones existentes, facilitando la identificación de soluciones más prometedoras.
- Iteración a lo Largo de Generaciones: La capacidad del algoritmo genético para evolucionar soluciones a lo largo de múltiples generaciones permite una mejora continua. Las soluciones menos aptas son eliminadas gradualmente, y las más aptas tienen la oportunidad de influir en generaciones futuras.
- Manejo de Restricciones y Condiciones Específicas: Al ajustar los operadores genéticos y la función de evaluación de aptitud, es posible incorporar de

manera efectiva las restricciones específicas del problema, como la limitación de ocurrencias consecutivas.

Ahora bien, se calcula la complejidad en dicha implementación del algoritmo genético.

Complejidad Temporal: El punto focal de la optimización recae en la función runGeneticAlgorithm(). En este método, un bucle for se ejecuta un número definido por max_generations. Dentro de este bucle, otro bucle for opera según la longitud de la matriz parents. Dentro de este último bucle, se invoca el método crossover(), que a su vez presenta dos bucles for anidados, ejecutándose según las longitudes de parent1 y parent2, respectivamente. En conjunto, la complejidad temporal del algoritmo se estima en aproximadamente $O(n^4)$, siendo 'n' el número de equipos.

Complejidad Espacial: La matriz populationCalendar destaca como el principal consumidor de memoria. Al ser una matriz 2D con dimensiones de POPULATION_SIZE por equipos, la complejidad espacial del algoritmo se estabiliza en $O(n^2)$, donde 'n' representa el número de equipos.

C. *Algoritmo para calcular el costo*

El servicio calculateTotalCost de la clase CalculateCost es el que se utiliza en los algoritmo para calcular el costo total de un calendario que contien partidos ida y vuelta. Este servicio recibe dos parametros, el primero es la matriz de distancias de tamaño n, donde n es el número de equipos, que representa el costo de viajar de una ciudad a otra y, el segundo es la matriz calendario de tamaño 2(n –1), donde n es el número de equipos, que representa las fechas de los partidos ida y vuelta.

La varible totalTournamentCost es donde se almacenara el total del costo calculado para todas las fechas del calendario y el arreglo teamCosts almacena el costo total para cada equipo.

El primer ciclo for del algoritmo recorre todos los equipos y la variable currentCity almacena la ciudad actual donde se encuentra el equipo.

El for interno recorre todos los partidos. La variable opponent almacena el oponente del equipo actual, el cual es llevado por el iterador team,en el partido actual y, la variable booleana isHomeGame indica si el partido es en casa o no.

Posteriormente el algoritmo verifica si el partido es en casa o no. Si no es en casa el equipo viaja a la ciudad del oponente y se añade el costo del viaje al costo total del equipo. Si es un partido en casa y el equipo no está ya en su ciudad, , el equipo vuelve a casa y se añade el costo del viaje al costo total del equipo.

Después de todos los partidos, si el equipo no está en su ciudad, vuelve a casa y se añade el costo del viaje al costo total del equipo. Finalmente, se añade el costo total del equipo al costo total del torneo.

Complejidad espacial del algoritmo: Para este algoritmo la complejidad espacial es proporcional al número de equipos, ya que se crea un array teamCosts para almacenar los costos de cada equipo. Por lo tanto, la complejidad espacial es O(n), donde n es el número de equipos.

Complejidad temporal del algoritmo: El algoritmo tiene dos bucles anidados. El bucle externo recorre todos los equipos, y el bucle interno recorre todos los partidos. Por lo tanto, la complejidad temporal es proporcional al número de equipos multiplicado por el número de partidos. esto se expresa como O(n*m), donde n es el número de equipos y m es el número de partidos que a su vez es igual a 2(n – 1) lo que da como resultado una complejidad temporal de $O(n^2)$.

III. Pruebas

En el desarrollo de software, la eficiencia y la precisión de los algoritmos son aspectos cruciales para garantizar un rendimiento óptimo de las aplicaciones. En este contexto, las pruebas de algoritmos desempeñan un papel fundamental al verificar y validar el comportamiento del código implementado. Por tal motivo se realizaron las siguientes pruebas para ver la funcionalidad del algoritmo tanto en tiempo de ejecución como en el costo/valor optimo total de las rutas de los equipos.

A. *Algoritmo Optimizado*

Minimo	Maximo	Equipos	Poblacion	Mutacion	Generacion	Tiempo(seg)	Costo
1	3	4	1000	20	1000	2,864	8276
						2,332	8290
						2,335	8276
						2,46	8290
						2,302	8276
						2,425	8276
						2,474	8276
						2,371	8276
						2,278	8276
						2,291	8413

Fig. 1. Variables y resultados utilizados primera prueba algoritmo optimizado (*figure caption*)

P. tiempo	P. Costo	V. Tiempo	D. Tiempo	C. V. Tiempo	V. Costo	D. Costo	C. V. Costo
2,4132	8292,5	0,02693136	0,164107769	0,00197899%	1643,85	40,54442008	0,48892879%

Fig. 2. Promedio y medidas de disperción primera prueba algoritmo optimizado (*figure caption*)

Minimo	Maximo	Equipos	Poblacion	Mutacion	Generacion	Tiempo(seg)	Costo
1	5	6	3000	20	1000	17,882	5510
						16,589	5198
						16,776	5738
						16,716	5695
						16,792	5584
						16,983	5545
						16,982	5283
						16,485	5555
						18,83	5552
						16,908	5478

Fig. 3. Variables y resultados utilizados segunda prueba algoritmo optimizado (*figure caption*)

P. tiempo	P. Costo	V. Tiempo	D. Tiempo	C. V. Tiempo	V. Costo	D. Costo	C. V. Costo
17,0943	5513,8	0,46552181	0,682291587	0,01237425%	24645,16	156,9877702	2,84717926

Fig. 4. Promedio y medidas de disperción segunda prueba algoritmo optimizado.

Minimo	Maximo	Equipos	Poblacion	Mutacion	Generacion	Tiempo(seg)	Costo
1	7	8	3000	20	1000	20,868	8825
						18,669	9018
						19,11	9043
						19,288	8871
						18,82	8604
						19,408	8834
						18,997	8868
						18,531	9263
						19,689	8915
						18,726	8994

Fig. 5. Variables y resultados utilizados tercera prueba algoritmo optimizado.

P. tiempo	P. Costo	V. Tiempo	D. Tiempo	C. V. Tiempo	V. Costo	D. Costo	C. V. Costo
19,2106	8923,5	0,42191364	0,649548797	0,00727908%	26914,25	164,0556308	1,83846731

Fig. 6. Promedio y medidas de disperción tercera prueba algoritmo optimizado.

Minimo	Maximo	Equipos	Poblacion	Mutacion	Generacion	Tiempo(seg)	Costo
1	9	10	3000	20	1000	48,614	15270
						48	14968
						49,805	14836
						48,79	15033
						47,876	15481
						48,101	14953
						47,55	15477
						46,693	15129
						46,862	14718
						47,54	14772

Fig. 7. Variables y resultados utilizados cuarta prueba algoritmo optimizado.

P. tiempo	P. Costo	V. Tiempo	D. Tiempo	C. V. Tiempo	V. Costo	D. Costo	C. V. Costo
47,9831	15063,7	0,76992349	0,877452842	0,00582495%	67058,01	258,955614	1,71907044

Fig. 8. Promedio y medidas de disperción cuarta prueba algoritmo optimizado.

Los resultados obtenidos a partir de las pruebas de rendimiento y precisión realizadas para el algoritmo optimizado revelan una notable estabilidad, respaldada por coeficientes de variación excepcionalmente bajos. Este hallazgo respalda la afirmación de que la precisión en el cálculo del mínimo costo para la creación de un calendario que cumple con las restricciones establecidas es altamente confiable. Sin embargo, es importante destacar que el tiempo de ejecución experimenta un aumento considerable al superar la cantidad de 8 equipos en la entrada del algoritmo.

Asimismo, se observa que a medida que el tamaño de la entrada, representado por el número de equipos, aumenta, la precisión se ve ligeramente afectada. Este inconveniente podría abordarse mediante el incremento de la variable de población, aunque esto conllevaría a un sacrificio en términos de tiempo de ejecución. En consecuencia, se plantea la necesidad de encontrar un equilibrio entre la precisión deseada y la eficiencia temporal, explorando posibles ajustes en los parámetros del algoritmo que permitan optimizar ambas facetas.

B. *Algoritmo Ingenuo*

Minimo	Maximo	Equipos	Tiempo(seg)	Costo	P. tiempo
1	3	4	0,709	10656	0,373
			0,307	11784	
			0,271	10815	
			0,278	11930	
			0,330	12275	
			0,267	8559	
			0,788	11617	
			0,256	9203	
			0,29	9933	
			0,234	9975	

Fig. 9. Variables y resultados utilizados primera prueba algoritmo ingenuo.

P. tiempo	P. Costo	V. Tiempo	D. Tiempo	C. V. Tiempo	V. Costo	D. Costo	C. V. Costo
0,373	10674,7	0,036189	0,190234066	0,00178210%	1395703	1181,398921	11,06727984%

Fig. 10. Promedio y medidas de disperción primera prueba algoritmo ingenuo.

Minimo	Maximo	Equipos	Tiempo(seg)	Costo
1	5	6	0,659	6889
			0,327	7544
			0,309	7223
			0,303	7272
			0,277	7103
			0,256	7339
			0,284	7670
			0,264	6910
			0,261	6659
			0,281	6426

Fig. 11. Variables y resultados utilizados primera prueba algoritmo ingenuo.

P. tiempo	P. Costo	V. Tiempo	D. Tiempo	C. V. Tiempo	V. Costo	D. Costo	C. V. Costo
0,3221	7103,5	0,01307149	0,114330617	0,00160950%	135313,5	367,8497655	5,17842987

Fig. 12. Promedio y medidas de disperción primera prueba algoritmo ingenuo.

Minimo	Maximo	Equipos	Tiempo(seg)	Costo
1	7	8	0,651	12611
			0,311	11406
			0,278	12343
			0,249	10102
			0,266	12609
			0,269	11762
			0,285	11188
			0,280	12719
			0,251	11169
			0,284	11926

Fig. 13. Variables y resultados utilizados primera prueba algoritmo ingenuo.

P. tiempo	P. Costo	V. Tiempo	D. Tiempo	C. V. Tiempo	V. Costo	D. Costo	C. V. Costo
0,3124	11783,5	0,01302684	0,114135183	0,00096860%	627735,5	792,2975767	6,72378815

Fig. 14. Promedio y medidas de disperción primera prueba algoritmo ingenuo.

Minimo	Maximo	Equipos	Tiempo(seg)	Costo
1	9	10	0,689	18765
			0,316	18461
			0,287	17283
			0,281	18552
			0,298	18078
			0,305	18300
			0,303	16500
			0,328	18827
			0,352	19262
			0,279	17217

Fig. 15. Variables y resultados utilizados primera prueba algoritmo ingenuo.

P. tiempo	P. Costo	V. Tiempo	D. Tiempo	C. V. Tiempo	V. Costo	D. Costo	C. V. Costo
0,3438	18124,5	0,01368896	0,116999829	0,00064553%	669728,3	818,3692626	4,51526532%

Fig. 16. Promedio y medidas de disperción primera prueba algoritmo ingenuo.

IV. Elección De La Mejor Alternativa

Para abordar la problemática del calendario, hemos decidido emplear un algoritmo optimizado que utiliza la estrategia de un algoritmo genético. Esta opción ha sido seleccionada debido a su capacidad para optimizar significativamente los costos de traslado. A pesar de que su complejidad temporal puede aumentar en función de la cantidad de datos de entrada, hemos determinado que el tiempo requerido para obtener una solución no es lo suficientemente alto como para descartar esta idea. En nuestra evaluación, priorizamos minimizar el costo de los traslados sobre el tiempo de respuesta, por lo que esta elección se basa en los resultados obtenidos en las pruebas anteriores detalladas en el punto III.

CONCLUSIONES

1. En relación al algoritmo genético, se ha notado que el tiempo de ejecución se ve notablemente afectado al aumentar el tamaño de la entrada, representado por el número de equipos en este caso. Además, al incrementar la variable POPULATION_SIZE, se logra mejorar la precisión para encontrar la solución óptima, pero se experimenta un aumento exponencial en el tiempo de ejecución. Esto se debe a que esta variable determina el número de soluciones consideradas al calcular el costo más bajo. Asimismo, se observó que al aumentar la variable MAX_GENERATIONS, se logra mejorar ligeramente la precisión sin afectar significativamente el tiempo de ejecución del algoritmo. En resumen, existe un equilibrio delicado entre precisión y eficiencia en la configuración de los parámetros del algoritmo genético, donde ajustar POPULATION_SIZE y MAX_GENERATIONS puede influir en la calidad de la solución y en el rendimiento temporal del algoritmo.

2. La capacidad del algoritmo optimizado para explorar y explotar eficientemente el espacio de búsqueda conduce a la convergencia hacia soluciones más óptimas en términos de costos. En el contexto de la generación de calendarios de partidos, la eficiencia del algoritmo óptimo en la búsqueda de soluciones de menor costo se revela como un elemento crucial. La notable diferencia en los costos promedio entre el algoritmo ingenuo y el óptimo resalta la capacidad de este último para generar soluciones de mayor calidad en términos de costos.

3. Una mejora significativa que podríamos implementar en la optimización del código se centra en reducir el tiempo de ejecución. Actualmente, nuestra implementación depende de un algoritmo ingenuo o de fuerza bruta para generar las posibles soluciones en el contexto del algoritmo genético que usamos para crear el calendario de partidos. Esta dependencia afecta notablemente la eficiencia general del algoritmo.

   Para mejorar esta situación, buscamos estrategias que nos permitan minimizar la dependencia del algoritmo ingenuo en la generación de soluciones iniciales dentro del algoritmo genético. La meta es disminuir la complejidad temporal y mejorar la eficacia general del proceso.

   Esta modificación resultaría en un algoritmo genético más eficiente y con una complejidad general más baja, lo que tendría un impacto positivo en el tiempo de ejecución y en la capacidad de generar soluciones óptimas para nuestro desafío de programación de calendarios de partidos.

REFERENCES

[1] Pablo Estevéz Valencia, “Optimización Mediante Algoritmos Genéticos” Anales del Instituto de Ingenieros De Chile, pp. 83–92, Agosto 1997.

[2] Marcos Gestal, Daniel Rivero, Juan Ramón Rabuñal, Julián Dorado, Alejandro Pazos, Introducción a los Algoritmos Genéticos y la Programación Genética, Universidade da Coruña, A Coruña, 2010.