UNIVERSIDAD SIMÓN BOLÍVAR
DPTO. COMPUTACIÓN Y TEC. DE INF.
CI-5437 INTELIGENCIA ARTIFICIAL I

Informe Proyecto 3

Estudiantes
LEONEL GUERRERO
EROS CEDEÑO

Profesor
CARLOS INFANTE

14 de julio de 2023

Contenido General

1. Introducción

La inteligencia artificial ha demostrado ser una herramienta poderosa para resolver problemas complejos en diversas áreas. Uno de los enfoques utilizados en la resolución de problemas es la modelación en Forma Normal Conjuntiva (CNF) y el uso de solucionadores SAT (Satisfiability) para encontrar soluciones a estos problemas. En este proyecto, nos enfocaremos en la modelación de problemas en CNF y en el uso de Glucose como SAT solver para resolverlos.

El objetivo principal del proyecto es aprender a modelar un problema en CNF y utilizar un SAT solver, en este caso Glucose, para encontrar soluciones eficientes y legibles. Un ejemplo de problema a resolver es la organización de un torneo, donde se deben asignar fechas y horas a los juegos, siguiendo ciertas reglas y restricciones. La modelación en CNF permite representar estas restricciones de manera eficiente y el SAT solver Glucose se encarga de encontrar una solución que cumpla con todas ellas.

A lo largo del proyecto, se modelara un problema en CNF y se evaluará la eficiencia de la traducción del problema y la solución proporcionada por Glucose. Este enfoque tiene aplicaciones en diversas áreas, como la planificación de eventos, la optimización de recursos y la toma de decisiones en general. Al finalizar el proyecto, se espera haber adquirido un conocimiento sólido sobre la modelación en CNF y el uso de SAT solvers para resolver problemas complejos en el ámbito de la inteligencia artificial.

2. Problema a resolver

En el contexto de la organización de un torneo, se requiere desarrollar un programa que encuentre una asignación de fecha y hora para los juegos que se llevarán a cabo. El objetivo es cumplir con las siguientes reglas:

1. Cada participante debe enfrentarse dos veces con todos los demás participantes, una vez como "visitantez otra como "local". Por lo tanto, si hay n equipos, cada equipo deberá jugar un total de $2 \times (n-1)$ partidos.
2. No puede haber dos juegos que se lleven a cabo simultáneamente.
3. Cada participante puede jugar como máximo una vez por día.
4. Un participante no puede jugar como "visitante.$^{\text{en}}$ días consecutivos ni como "local.$^{\text{en}}$ días consecutivos.
5. Todos los juegos deben comenzar en horas .$^{\text{en}}$ punto"(por ejemplo, 13:00:00 es una hora válida, pero 13:30:00 no lo es).
6. Los juegos deben tener lugar entre una fecha de inicio y una fecha de fin especificadas, y pueden tener lugar en dichas fechas.
7. Los juegos deben ocurrir dentro de un rango de horas especificado, que será el mismo para todos los días del torneo.
8. Para fines prácticos, se asume que todos los juegos tienen una duración de dos horas.

2.1. Formato de entrada

El sistema debe recibir un archivo JSON con el siguiente formato (se asume que siempre se proporcionará el formato correcto):

{
"*tournament_name*" : String. Nombre del torneo,
"*start_date*" : String. Fecha de inicio del torneo en formato ISO 8601,
"*end_date*" : String. Fecha de fin del torneo en formato ISO 8601,
"*start_time*" : String. Hora a partir de la cual pueden ocurrir los juegos en cada día, en formato ISO 8601
"*end_time*" : String. Hora hasta la cual pueden ocurrir los juegos en cada día, en formato ISO 8601,
"*participants*" : [String].Lista con los nombres de los participantes en el torneo
}

Se asume que todas las horas se presentan sin especificar la zona horaria, y por lo tanto se considera que la zona horaria es UTC.

3. Modelación del problema

El objetivo general es crear un transformador de sintaxis para el SAT Solver Glucose que genere una organización de los encuentros del torneo tales que se satisfagan las restricciones dadas y luego representar esta organización en un archivo ics de calendario para su mejor visualización. La figura 1 muestra un diagrama de flujo del programa cliente.

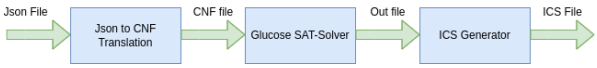
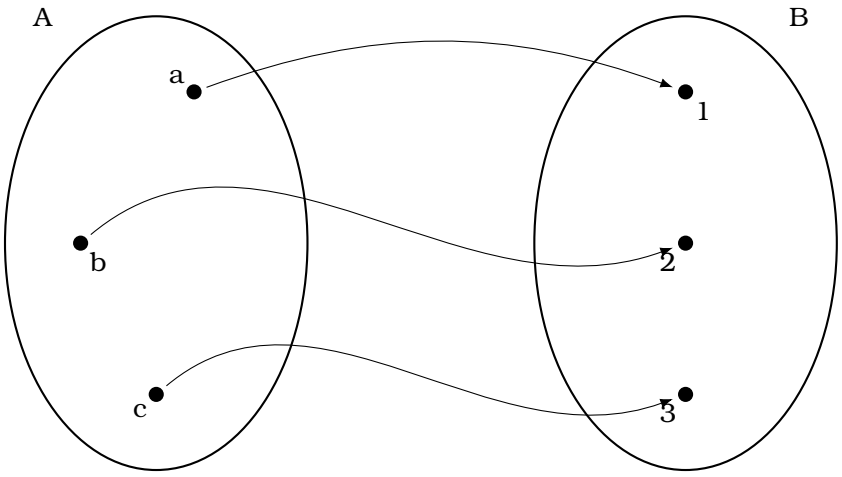


Figura 1: Diagrama de Flujo del programa

3.1. Variables y modelo matemático

El problema a resolver presenta 4 variables de interés. Estas son: Equipo Local, Equipo visitante, Hora y Fecha de disputa del encuentro. Por lo que las posibles combinaciones de estas 4 variables se pueden representar de manera única como tuplas de 4 elementos. Abstraído el problema de este modo, las variables quedan en un espacio 4-dimensional, por lo que es necesario crear una función biyectiva tal que dada una 4-tupla que represente la configuración de un encuentro especifico posible retorne un numero entero positivo menor o igual a la cardinalidad del conjunto de las variables (tuplas) en el espacio 4-dimensional. Por lo que se desea asignar a cada una de estas 4-tuplas un numero único que servirá como identificador de la variable dentro del archivo cnf que se genera. Se almacenan las 4-tuplas dentro de un vector y así el índice de la posición de una tupla dentro del vector sera su identificador único para poder representar las variables como números enteros únicos y aun ser capaces de retornar la transformación sin perder información. El vector de variables es el tipo abstracto de dato fundamental dentro de la solución del problema.

Por ejemplo, si se tiene un dominio del problema con únicamente 3 posibles encuentros entre equipos se puede representar la funcion de abstraccion de las variables con en el diagrama de Venn siguiente:



Donde

$$A = \{< E_i, E_j, F_k, H_t > | E_i, E_j \in Participantes \wedge E_i \neq E_j \wedge F_k \in Fecha \wedge H_t \in Hora\}$$

$$B = \{i \in \mathbb{Z}^+ \wedge i \leq |A|\}$$

Continuando con el análisis matemático, se define la función de evaluación

$$\alpha \colon B \to \mathbb{B}$$

Donde

$$\alpha(x) \equiv True \wedge x \in B$$

Indica que ese juego entre los dos participantes y a esa hora se debe efectuar.

3.2. Especificación del TAD

Para representar el problema y almacenar las variables definidas en el inciso anterior se plantea almacenar las variables en una matriz, sin embargo para aplicar la numeracion de las variables necesaria para la biyeccion se concatenan todas sus filas en un vector. Por simplicidad de la explicación se analiza su estructura matricial teniendo en cuenta que su conversión no afecta la capacidad de representación.

La definición de la matriz de vectores se hace de modo iterativo hasta formar una matriz de la siguiente forma:

$$V = \begin{bmatrix}
E_1E_2F_1H_1 & E_1E_2F_1H_2 & \dots & E_1E_2F_1H_t & E_1E_2F_2H_1 & E_1E_2F_2H_2 & \dots & E_1E_2F_kH_t \\
E_1E_3F_1H_1 & E_1E_3F_1H_2 & \dots & E_1E_3F_1H_t & E_1E_3F_2H_1 & E_1E_3F_2H_2 & \dots & E_1E_3F_kH_t \\
\vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots \\
E_1E_nF_1H_1 & E_1E_nF_1H_2 & \dots & E_1E_nF_1H_t & E_1E_nF_2H_1 & E_1E_nF_2H_2 & \dots & E_1E_nF_kH_t \\
E_2E_1F_1H_1 & E_2E_1F_1H_2 & \dots & E_2E_1F_1H_t & E_2E_1F_2H_1 & E_2E_1F_2H_2 & \dots & E_2E_1F_kH_t \\
E_2E_3F_1H_1 & E_2E_3F_1H_2 & \dots & E_2E_3F_1H_t & E_2E_3F_2H_1 & E_2E_3F_2H_2 & \dots & E_2E_3F_kH_t \\
\vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots \\
E_nE_1F_1H_1 & E_nE_1F_1H_2 & \dots & E_nE_1F_1H_t & E_nE_1F_2H_1 & E_nE_1F_2H_2 & \dots & E_nE_1F_kH_t \\
E_nE_2F_1H_1 & E_nE_2F_1H_2 & \dots & E_nE_2F_1H_t & E_nE_2F_2H_1 & E_nE_2F_2H_2 & \dots & E_nE_2F_kH_t \\
\vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots \\
E_nE_{n-1}F_1H_1 & E_nE_{n-1}F_1H_2 & \dots & E_nE_{n-1}F_1H_t & E_nE_{n-1}F_2H_1 & E_nE_{n-1}F_2H_2 & \dots & E_nE_{n-1}F_kH_t
\end{bmatrix}$$

Se enfatiza que la matriz de variables V presenta características favorables para la definición de las restricciones. Por ejemplo para la restricción 1 queda definida implícitamente

que dos equipos se enfrentan al menos dos veces dado que para todo par de equipos E_i y E_j estos aparecen en dos filas de la matriz. Para asegurar que como máximo se enfrenten dos veces basta con asegurar que en cada fila no existan dos variables evaluando a True al mismo tiempo. Para la segunda restricción el razonamiento es similar pero por columnas, dado que cada columna comparte el tiempo del juego basta con asegurar que por columna no evalúen a True dos o mas elementos simultáneamente. Para la restricción 3 se asegura que por día (una submatriz con las columnas restringidas en cierto rango del dia) una variable que contenga un equipo dado no sea satisfecha mas de una vez, esto es que no evalúen cierto en simultaneo ya sea que el equipo sea visitante o local.

De igual modo se hace con las restricciones restantes, se genera el cnf recorriendo la matriz de variables y se asegura que las variables no evaluaran True en simultaneo dependiendo del caso.

4. Implementación

El proyecto de generación de horarios para torneos se implementó utilizando el lenguaje de programación C++. A continuación, se presenta una descripción general de la implementación y las funciones principales utilizadas en el programa.

El programa consta de varias funciones que realizan diferentes tareas para lograr la generación de horarios de manera eficiente. Estas funciones se dividen en los siguientes bloques:

1. Función principal (main): La función principal del programa se encarga de coordinar todas las etapas del proceso de generación de horarios. Realiza las siguientes tareas:
   - Lee el archivo de entrada JSON que contiene la configuración del torneo, como las fechas, horas y participantes.
   - Realiza la modulación del problema en CNF mediante la creación de una abstracción de variables.
   - Crea el archivo CNF y lo llena con las restricciones correspondientes.
   - Ejecuta el SAT solver Glucose para encontrar una solución satisfactoria o determinar que el problema es insatisfactible.
   - Analiza la salida del SAT solver y genera el archivo de salida en formato ICS (iCalendar) que representa los horarios de los juegos.
2. Funciones auxiliares (utils.hpp): Estas funciones se utilizan para realizar tareas específicas, como la creación de eventos en el archivo ICS, la generación de vectores de tiempos posibles y la obtención de la abstracción de variables.

El código fuente completo así como las instrucciones de clonado, compilación y ejecución se encuentra disponible en el repositorio del proyecto.

5. Resultados

Se implemento un programa en Python que genera casos de prueba faciles y dificiles para el problema de la organización de un torneo. El programa genera varios casos de prueba variando el número de participantes, la cantidad de días del torneo y el rango de horas en el que pueden ocurrir los juegos. Para cada caso de prueba, se genera un archivo JSON con el formato de entrada especificado en la sección anterior. Los casos de prueba se generan de manera combinatoria, pero se garantiza que siempre se cumplen las reglas del problema.

En esta sección, se presentan los resultados obtenidos del proyecto de generación de horarios para torneos. Se realizaron pruebas utilizando diferentes casos de prueba, tanto casos fáciles como casos difíciles, y se registró el tiempo de ejecución para cada caso. A continuación, se muestra un análisis de los resultados obtenidos:

5.1. Casos de prueba fáciles

Se ejecutaron pruebas utilizando casos de prueba fáciles, los cuales presentan una menor cantidad de participantes y un número reducido de bloques de juegos por día. Estos casos son considerados más simples y se espera que el tiempo de ejecución sea muy bajo. En la tabla 1 se muestran los resultados obtenidos de las pruebas realizadas con casos de prueba fáciles.

Numero de Participantes	Dias del torneo	Horas disponibles	Tiempo de ejecucion
2	4	2	0.00
2	6	4	0.00
4	4	3	0.00
4	5	4	0.00
2	4	3	0.00
2	6	2	0.00
5	5	4	0.04
5	6	2	0.00
4	4	2	0.00
5	6	3	0.01
2	5	4	0.00
4	6	2	0.00
2	4	4	0.00
4	5	2	0.00
5	5	2	0.00
5	5	3	0.00
2	6	3	0.00
4	4	4	0.00
4	5	3	0.00
5	6	4	0.76
2	5	3	0.00
5	4	2	0.00
4	6	3	0.00
5	4	3	0.00
2	5	2	0.00
5	4	4	0.00
4	6	4	0.00

Cuadro 1: Casos de prueba fáciles

Como se puede observar en la tabla, la mayoría de los casos de prueba fáciles se ejecutaron en un tiempo muy bajo, cercano a cero. Esto indica que el algoritmo pudo generar los horarios de los juegos de manera eficiente para estos casos simples.

Sin embargo, hay un caso de prueba que se destaca por su mayor tiempo de ejecución: el que contiene 5 participantes, 6 dias para el torneo y 4 horas disponibles, con un tiempo de 0.76 segundos. Este caso pudo requerir más tiempo debido a una mayor complejidad en las restricciones o a una combinación particular de participantes y bloques de juegos.

En general, los casos de prueba fáciles demuestran que el algoritmo es eficiente y puede generar los horarios de los juegos de manera rápida y precisa para la mayoría de los casos simples.

5.2. Casos de prueba dificiles

También se realizaron pruebas utilizando casos de prueba difíciles, los cuales presentan una mayor cantidad de participantes y un mayor número de bloques de juegos por día. Estos

casos son más desafiantes y se espera que requieran más tiempo de ejecución. En la tabla 2 se muestran los resultados obtenidos de las pruebas realizadas con casos de prueba difíciles.

Numero de Participantes	Dias del torneo	Horas disponibles	Tiempo de ejecucion
8	9	4	237.80
6	8	4	73.30
8	8	4	10.52
7	8	4	37.96

Cuadro 2: Casos de prueba dificiles

En los casos de prueba difíciles, se puede observar un incremento significativo en el tiempo de ejecución en comparación con los casos fáciles. Esto se debe a la mayor complejidad de las restricciones y al mayor número de combinaciones posibles entre participantes y bloques de juegos.

En particular, el caso de prueba con 8, 9, 4 requirió el mayor tiempo de ejecución, con 237.80 segundos. Esto indica que el algoritmo tuvo que realizar un esfuerzo considerable para generar los horarios de los juegos en este caso específico.

En general, los casos de prueba difíciles demuestran que el algoritmo puede manejar escenarios más complejos, pero a costa de un mayor tiempo de ejecución. Estos casos pueden requerir optimizaciones adicionales para mejorar la eficiencia del algoritmo y reducir el tiempo de generación de horarios.

5.3. Análisis de resultados

El proyecto de generación de horarios para torneos ha demostrado ser eficiente en la mayoría de los casos de prueba fáciles, generando los horarios de los juegos en tiempos muy bajos. Esto demuestra la capacidad del algoritmo para manejar escenarios simples de manera rápida y precisa.

Sin embargo, los casos de prueba difíciles presentaron un mayor desafío, con tiempos de ejecución significativamente más altos. Esto indica que el algoritmo puede encontrar dificultades al manejar escenarios más complejos con una mayor cantidad de participantes y bloques de juegos.

Además, es importante destacar que los tiempos de ejecución pueden verse afectados de manera exponencial al realizar cambios pequeños en los parámetros del problema, como el número de participantes o el número de bloques de juegos por día. Esto significa que incluso un ligero aumento en la complejidad del problema puede tener un impacto significativo en el tiempo requerido para generar los horarios. Por lo tanto, se recomienda tener en cuenta esta escalabilidad al planificar torneos de mayor envergadura, y considerar posibles mejoras al algoritmo para manejar eficientemente casos más desafiantes. Esto puede verse reflejado sobre todo en los casos de prueba difíciles, donde se observa un incremento considerable en el tiempo de ejecución al cambiar el número de participantes y bloques de juegos por día.

6. Conclusiones

En este proyecto, se abordó el desafío de organizar un torneo mediante la asignación de fechas y horas a los juegos, siguiendo diversas reglas y restricciones. Para lograrlo, se utilizó la modelación en Forma Normal Conjuntiva (CNF) y se empleó el SAT solver Glucose para encontrar soluciones eficientes y legibles.

La modelación en CNF permitió representar de manera eficiente las reglas del torneo,

como la asignación de juegos entre participantes, la restricción de juegos simultáneos, la limitación de juegos por día y la secuencia adecuada de juegos como visitantes y locales. Esta representación en CNF facilitó la traducción del problema a una forma que Glucose pudiera entender y resolver.

Durante el desarrollo del proyecto, se observó que el enfoque de modelación en CNF y el uso de Glucose como SAT solver ofrecieron ventajas significativas. La capacidad de Glucose para encontrar soluciones satisfactorias en un tiempo razonable resultó ser una herramienta valiosa para resolver problemas complejos. Además, la legibilidad de las soluciones proporcionadas por Glucose permitió comprender fácilmente la asignación de fechas y horas a los juegos, lo cual facilitó su implementación práctica.

El proyecto también permitió adquirir conocimientos sólidos sobre la modelación en CNF y el uso de SAT solvers en la resolución de problemas complejos. Estos conocimientos son aplicables en diversas áreas, como la planificación de eventos, la optimización de recursos y la toma de decisiones en general. La capacidad de representar problemas en CNF y utilizar SAT solvers para encontrar soluciones eficientes brinda oportunidades para abordar desafíos en la inteligencia artificial de manera efectiva y escalable.

En resumen, este proyecto ha demostrado que la modelación en CNF y el uso de SAT solvers, como Glucose, son herramientas poderosas para resolver problemas complejos. La combinación de estos enfoques proporciona soluciones eficientes y legibles, permitiendo abordar desafíos en diversos campos. Como resultado, se ha obtenido un valioso conocimiento y experiencia en la modelación en CNF y el uso de SAT solvers, sentando las bases para futuros proyectos y aplicaciones en el campo de la inteligencia artificial.