Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación

T E S I S

Aplicación web para el cálculo del Máximo Conjunto Independiente en grafos

Presenta: *José Domingo Valdez González*

Para obtener el grado de: *Ingeniero en Ciencias de la Computación*

Director: *Dr. Pedro Bello López*

Puebla, Pue, diciembre 2025

Agradecimientos

Agradecimiento a la Vicerrectoría de Investigación y Estudios de Posgrado de la Benemérita Universidad Autónoma de Puebla por el apoyo económico brindado para desarrollar este trabajo de tesis dentro del proyecto VIEP 2025: aplicando ciencia de datos para analizar cadenas genómicas. Agradecimiento a mis padres por el apoyo incondicional desde siempre y de diversas maneras. Agradecimiento a mis maestros de la Facultad, muy en especial mi asesor en este trabajo, Dr. Pedro Bello López, por todo el apoyo brindado.

CONTENIDO

1. Introducción 3
1.1 Problema del Máximo Conjunto Independiente 4
1.2 Objetivo general y objetivos específicos del proyecto 8
1.3 Metodología 8
1.4 Trabajos relacionados 10
2. Algoritmos para hallar un Conjunto Independiente Máximo 13
2.1 Teoría básica de grafos 13
2.2 Algoritmos para conjuntos independientes 17
2.3 Aplicaciones del MIS 21
3. Algoritmo implementado para el MIS 23
3.1 Pseudocódigo del algoritmo aleatorio 24
3.2 Pseudocódigo del algoritmo para el MIS 25
3.3 Pseudocódigo para el MIS exacto 27
4. Sistema para el cálculo del MIS 30
4.1 Herramientas utilizadas para el desarrollo 30
4.2 Diseño y funciones de la interfaz de usuario 33
4.5 Pruebas con grafos aleatorios 41
5. Conclusiones y perspectivas 46
Referencias 47

1. INTRODUCCIÓN

Un grafo [1] es una estructura matemática utilizada para representar relaciones entre objetos o entidades. Un grafo (Figura 1) consta de un conjunto de nodos (también llamados vértices) y un conjunto de aristas (también llamados arcos) que conectan los nodos. Los grafos se utilizan para modelar una variedad de situaciones en la vida real donde existen conexiones o relaciones entre elementos [2, 3].

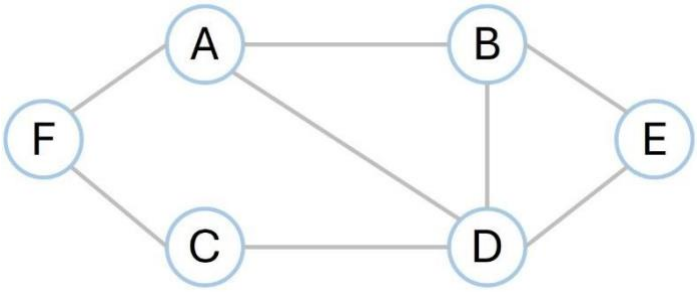


Figura 1. Un Grafo con 6 nodos y 8 aristas

Los grafos y su teoría asociada es una herramienta versátil y ampliamente aplicable que desempeña un papel fundamental en la resolución de problemas en una amplia gama de campos, desde la computación y las matemáticas hasta la biología, la logística, la ingeniería y las ciencias sociales. Su capacidad para representar y analizar relaciones y estructuras complejas la convierte en una herramienta esencial en la resolución de problemas en general [4]. Un ejemplo de problemas que se plantean con grafos es el caso de los conjuntos independientes.

Un conjunto independiente en un grafo es un grupo de vértices que no están conectados entre sí mediante aristas. En otras palabras, ningún par de vértices dentro del conjunto tiene una relación directa. Este concepto es importante en teoría de grafos porque permite identificar subconjuntos de nodos que no se influencian o no tienen conexión directa. En la Figura 2 indicamos un ejemplo

donde los nodos {A, E} y {B, C} forman conjuntos independientes de tamaño dos, sin embargo, hay más conjuntos independientes de tamaño 0, de tamaño 1, 2, 3, etc. En este trabajo se obtiene el conjunto independiente de mayor tamaño.

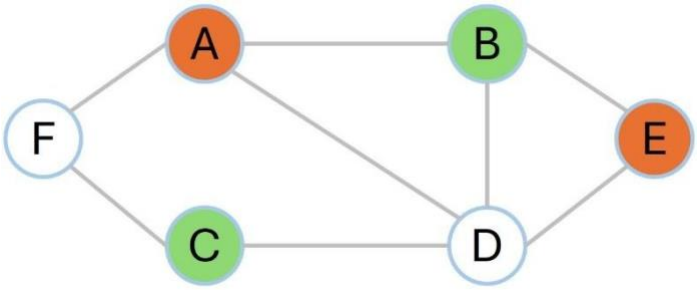


Figura 2. Ejemplo de dos conjuntos independientes (A, B) y (B, C)

1.1 Problema del Máximo Conjunto Independiente

Aquí planteamos el problema de determinar el Máximo Conjunto Independiente (MIS - por sus siglas en inglés, Maximum Independent Set). Un conjunto independiente en un grafo es un conjunto de vértices tal que ningún par de vértices en ese conjunto está conectado por una arista, es decir, no hay aristas entre los vértices dentro del conjunto.

El concepto de Máximo Conjunto Independiente tiene diversas aplicaciones en múltiples disciplinas, destacándose principalmente en la teoría de grafos, optimización y ciencias de la computación. En teoría de grafos, se utiliza para resolver problemas como la optimización de redes de comunicación, asignación de recursos o la planificación de tareas simultáneas sin interferencias. En redes sociales, permite identificar grupos de usuarios sin interacciones directas, facilitando la segmentación y análisis de comunidades. En biología computacional, ayuda a estudiar redes de proteínas y sistemas metabólicos al identificar componentes funcionales independientes. Además, en áreas como la computación distribuida, permite la asignación eficiente de tareas a nodos de un sistema sin conflictos. Así, su aplicabilidad abarca desde la ingeniería y el análisis

de redes hasta la biología y la planificación de proyectos, convirtiéndolo en una herramienta clave para la resolución de problemas complejos en diversas áreas [4].

El máximo conjunto independiente es el conjunto de vértices independiente más grande posible en un grafo, es decir, es el conjunto independiente con el mayor número de vértices. Un MIS no necesariamente es único; es posible que haya varios conjuntos independientes de tamaño máximo en un mismo grafo. En la Figura 1 los nodos A, C, E y B, D, F forman dos MIS y se colorea solo uno de los MIS para ejemplificar [5, 6].

El MIS tiene diversas aplicaciones importantes en áreas como redes de comunicación, optimización de recursos y planificación de tareas. En redes inalámbricas y de sensores, el MIS se utiliza para seleccionar nodos de transmisión sin interferencias, optimizando el uso del espectro. En problemas de asignación de recursos o tareas, el MIS ayuda a evitar conflictos al seleccionar subconjuntos de tareas o recursos que no se solapan. Además, en la programación de proyectos y sistemas de control, el MIS permite organizar y asignar tareas de manera eficiente, sin superposiciones. También se emplea en el coloreado de grafos, sistemas de recomendación, y optimización del tráfico, donde ayuda a seleccionar elementos clave de forma no conflictiva [1, 3].

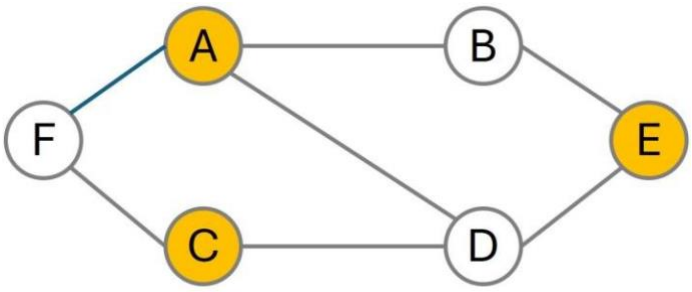


Figura 3. Los nodos A, E y C forman un MIS.

El Máximo Conjunto Independiente en grafos es un problema NP-duro, lo que significa que encontrar una solución exacta en tiempo polinómico no es posible para casos generales, a menos que se demuestre que P = NP. Un problema es NP-duro si, al menos, es tan difícil como cualquier problema en NP, pero no necesariamente pertenece a NP. Es decir, si pudieras resolver un problema NP-duro en tiempo polinómico, entonces podrías resolver todos los problemas en NP en tiempo polinómico. Los problemas NP-duros no tienen que ser decidibles (es decir, puede que no se pueda determinar si una solución es válida en tiempo finito). Existen varios algoritmos exactos y aproximados para resolver el problema, con diferentes enfoques dependiendo de las características del grafo.

Existen diferentes propuestas algorítmicas para tratar el problema del MIS, algunos de los más generales se describen a continuación.

Algoritmo de Fuerza Bruta

La técnica de fuerza bruta es un algoritmo exacto que evalúa todos los subconjuntos de vértices posibles en un grafo y verifica si son independientes. La complejidad de este enfoque es exponencial (O(2n)), por lo que solo es viable en grafos pequeños. En este algoritmo se generan todos los subconjuntos posibles de vértices, se verificar si el subconjunto es independiente (sin aristas entre los vértices del subconjunto) y se selecciona el subconjunto independiente más grande [7].

Programación Dinámica

Esta técnica de fuerza bruta se aplica principalmente sobre grafos tipo árboles y grafos bipartitos, se pueden usar algoritmos de programación dinámica que mejoran significativamente el tiempo de resolución. Este enfoque divide el problema en subproblemas más pequeños y los resuelve recursivamente [8].

Ramificación y Poda

Este también es un enfoque de fuerza bruta que utiliza una estrategia de búsqueda que evalúa las soluciones de forma eficiente, podando ramas que no pueden llevar a una mejor solución que la actual [9, 15, 16].

Algoritmos de Aproximación (heurísticas)

Algoritmo Greedy.

El algoritmo codicioso selecciona los vértices en función de algún criterio (por ejemplo, el vértice de menor grado) y los añade al conjunto independiente, eliminando los vértices adyacentes. Aunque no garantiza una solución óptima, es rápido y adecuado para grafos grandes. Primero ordena los vértices según algún criterio (por ejemplo, grado de los vértices), seleccion el primer vértice y lo añade al conjunto independiente, elimina los vértices adyacentes y este proceso se repite hasta que no queden vértices disponibles [10, 12, 14].

Búsqueda Local, Simulated Annealing, Algoritmos Genéticos

Los algoritmos heurísticos son útiles cuando se requiere una solución rápida para grafos grandes. Métodos como búsqueda local, simulated annealing y algoritmos genéticos permiten encontrar buenas soluciones aproximadas al MIS mediante exploración iterativa del espacio de soluciones [11, 12, 13].

Existen otras propuestas algorítmicas para grafos especiales como los Grafos Bipartitos donde el MIS se puede encontrar de manera eficiente, ya que cualquier conjunto independiente máximo es un conjunto de vértices de una de las dos particiones. Esto se puede resolver en tiempo polinómico. Y en los Grafos Planos donde existen algoritmos específicos que mejoran la eficiencia de encontrar el MIS, aprovechando las restricciones de planitud.

En este proyecto de tesis se revisaron las diversas propuestas algorítmicas para crear la aplicación en web que permita interactuar a los usuarios y calcular el MIS en grafos sin restricciones.

1.2 Objetivo general y objetivos específicos del proyecto.

Objetivo general

Desarrollar un sistema computacional que sea visible a través de la web [17, 18] que permita obtener el Máximo Conjunto Independiente de un grafo, para obtener el MIS se aplicará un método heurístico para que el algoritmo utilice tiempo polinomial para hallar el MIS.

Objetivos Específicos

- Revisar la teoría de grafos.
- Revisar la teoría de los conjuntos independientes y el MIS en grafos.
- Determinar el algoritmo a utilizar para calcular el MIS.
- Diseñar e implementar el sistema web integrando el algoritmo diseñado.
- Realizar las pruebas necesarias.
- Crear un repositorio de casos de prueba para ejemplificar el sistema creado.

1.3 Metodología.

Para el desarrollo [19, 20] del trabajo se utilizó la metodología RAP (Rapid Application Development, por sus siglas en inglés) es un enfoque de desarrollo de

software que se centra en la rápida entrega de aplicaciones mediante la creación iterativa y el uso de prototipos. RAP tiene como objetivo reducir el tiempo necesario para desarrollar aplicaciones, enfocándose en la rapidez y flexibilidad, y asegurando que el software cumpla con las expectativas del usuario. A través de un ciclo continuo de retroalimentación, el desarrollo de la aplicación se va ajustando y mejorando para satisfacer las necesidades del cliente. El RAP se caracteriza por el desarrollo rápido, ciclos continuos y rápidos, usa el prototipado y pruebas a través de la interacción constante con el cliente.

El RAP se debe aplicar si los requisitos no están completamente claros al inicio, RAP permite que los usuarios proporcionen retroalimentación mientras se desarrolla el sistema; en aplicaciones que requieren una experiencia de usuario muy centrada en la interfaz, RAP facilita realizar ajustes continuos a la interfaz según las preferencias de los usuarios y en proyectos pequeños debido a que el RAP puede ser muy eficaz, ya que el tiempo de desarrollo acelerado es un beneficio importante.

Las etapas principales que se van a considerar con esta metodología son:

1. Planificación inicial: Se establecen los requisitos generales del sistema, se establecen la teoría a revisar de grafos, conjuntos independientes y del MIS.
2. Desarrollo del prototipo: Se realizar el diseño del algoritmo propuesto para determinar el MIS, se establece la entrada de los datos y en esta etapa también se prevé el diseño de la aplicación web, se crea un prototipo inicial del sistema que incluye las características más esenciales. Este prototipo es funcional y puede ser probado por los usuarios para obtener retroalimentación.
3. Revisión y retroalimentación del usuario: Los usuarios interactúan con el prototipo de la aplicación web y proporcionan comentarios sobre lo que les gusta, lo que no les gusta y qué cambios desean. En base a esta retroalimentación, se ajustan los requisitos y se mejoran las características del sistema.

4. Iteración y refinamiento: Se desarrollan nuevas versiones del prototipo web con características mejoradas o adicionales. Cada iteración del prototipo incluye modificaciones basadas en los comentarios del cliente y es probada nuevamente para obtener más retroalimentación.
5. Desarrollo final y despliegue: Una vez que el sistema alcanza un nivel satisfactorio, se procede a su finalización y despliegue en el entorno de producción a través de un servidor web y se genera un conjunto de instancias de prueba que se pueden ejecutar a través del sistema.

1.4 Trabajos relacionados

Se realizó una revisión (https://bibliotecas.buap.mx/portal/) de las tesis realizadas en la Facultad de Ciencias de la Computación que están relacionadas con el tema de conjuntos independientes, el MIS y los problemas en grafos, de esta revisión se encontró que la teoría de grafos se ha utilizado para resolver diferentes problemas como el coloreo de grafos, problemas de satisfactibilidad y conjuntos independientes entre otros, como ejemplo tenemos los siguientes trabajos.

- Tesis de Licenciatura: un algoritmo heurístico para el coloreo de grafos (Lilibeth Zuñiga Vargas). En esta tesis, se da una breve explicación de las definiciones básicas de la teoría de grafos, y procedimientos básicos y generales utilizados en algoritmos para el coloreo de grafos, ya que son fundamentales para la comprensión de problemas típicos en la teoría de grafos. El motivo de esta tesis es proponer un algoritmo heurístico, para calcular el número cromático de un grafo de forma eficiente, siendo esté el problema medular en el área de Coloreo de grafos.
- Tesis de Licenciatura: Framework educativo para la modulación en la resolución la aplicación de algoritmos con grafos (Lima Estrada, Efraín De Jesús), "Se propone una herramienta de software con la finalidad de resolver problemas sobre caminos más cortos con grafos dirigidos, grafos

no dirigidos y árboles n-arios mediante algoritmos básicos y con implementación de técnicas de inteligencia... artificial, permitiendo la interacción con el usuario mediante una interfaz gráfica de fácil uso, facilitándole al usuario ingresar el grafo mediante un lienzo de dibujo para su posterior procesamiento.

- Tesis de Licenciatura: Grafos para procesamiento morfológico (Gallegos Ávila, Pedro Adair). En un grafo los objetos son representados mediante los vértices; mientras que las relaciones entre los objetos son descritas como aristas. En la década de 1960 fue que se dieron los primeros pasos en el estudio de imágenes; en un principio de forma... el capítulo 2, iniciamos con los elementos básicos de la teoría de grafos. Posteriormente, presentamos los operadores erosión y dilatación en un grafo generados por la función de vecindades NG(v).
- Tesis de Licenciatura: análisis del cómputo exhaustivo de conjuntos independientes (Juan Antares Perdomo Flández). En este trabajo es un método que busca englobar todas las anteriores soluciones, es decir un algoritmo capaz de resolver, para cualquier grafo simple (tipos de grafos que no presentan aristas múltiples, bucles o aristas dirigidas), el conteo de todos los conjuntos independientes lo que a la vez contiene a los conjuntos maximales y máximos. Adicionalmente se añade el muestreo de los conjuntos de forma ordenada y clasificados por tamaños.
- Tesis de Licenciatura: Sistema web para el conteo eficiente de Conjuntos Independientes en estructuras jerárquicas (Árboles). Morales Alcántara, Luis David. Los conjuntos independientes son fundamentales en combinatoria, y tienen aplicaciones en: visión por computadora, reconocimiento de patrones, planificación, entre otras. Los conjuntos independientes se utilizan también en física estadística, donde... el problema es un caso especial del modelo núcleo-duro (hard-core).
- Tesis de Licenciatura: Algoritmo para determinar la k-coloración de un grafo. García Limón, Olga Lidia. En la teoría de grafos, la coloración de grafos es un caso especial de etiquetado de grafos; es una asignación de etiquetas llamadas colores a los nodos o vértices del grafo. Es decir, una coloración de los vértices de un grafo es una asignación tal que ningún vértice adyacente comparta el mismo color. Si en la coloración se usan k (k=1,2, …, n) colores

distintos diremos que es una k-coloración. Una coloración siempre es posible, dado que podemos asignar a cada vértice del grafo un color diferente si fuera necesario, por ejemplo, para grafos completos. Si existe una k-coloración de G se dice que el grafo G es k-coloreable. El valor mínimo k para el que un grafo G es k-coloreable se denomina número cromático de G, y se designa por x(G). De esta forma la coloración de los vértices se basa en encontrar grupos de vértices en el grafo que no sean adyacentes entre sí para así poder asignarles el mismo color. Este proyecto propone diseñar e implementar un algoritmo para determinar con cuantos colores se puede colorear un grafo. El sistema debe estar en línea para que personas interesadas puedan realizar pruebas del kcoloreo de grafos".

Existen diversas tesis de licenciatura en computación desarrolladas respecto a la teoría de grafos y sus aplicaciones sin embargo de la investigación realizada en los repositorios institucionales de tesis y en internet no se encontró algún trabajo de tesis con nuestra propuesta de modelar el problema del Máximo Conjunto Independiente sobre grafos

2. ALGORITMOS PARA HALLAR UN CONJUNTO INDEPENDIENTE MÁXIMO

2.1 Teoría básica de grafos

La teoría de grafos constituye una de las áreas fundamentales dentro de las matemáticas discretas y la informática teórica. Un grafo se define como una estructura compuesta por un conjunto finito de vértices (o nodos) y un conjunto de aristas (o enlaces) que conectan pares de vértices. Formalmente, un grafo se denota como $G = (V, E)$, donde V representa el conjunto de vértices y E el conjunto de aristas. Cada arista puede representarse como un par ordenado (u, v), con $u, v \in V$ [2].

Los grafos pueden clasificarse según distintas características. Si las aristas poseen dirección, se denominan grafos dirigidos; si no la tienen, grafos no dirigidos. Además, los grafos pueden ser ponderados cuando las aristas tienen un valor asociado, o no ponderados si todas las conexiones se consideran equivalentes. Estas estructuras se utilizan ampliamente en problemas de optimización, redes, análisis de datos y modelado de sistemas complejos [3].

Grafos dirigidos (Digrafos)

Un grafo dirigido (conocido también como digrafo) como se muestra en la Figura 2.1 es un tipo de grafo en el cual las aristas que conectan los vértices tienen una dirección hacia un vértice en específico. Cada arista se representa como un par ordenado (u, v), donde la conexión va desde el vértice u hacia el vértice v.

Este tipo de grafo es fundamental en la representación de redes de flujo, diagramas de dependencia, rutas con sentido único (como calles de una ciudad) o modelos de relaciones jerárquicas.

Aplicaciones comunes:

- Modelado de procesos con secuencia lógica (flujos de trabajo, autómatas finitos).
- Redes de transporte (calles, rutas aéreas, logística).
- Análisis de dependencias en compiladores y sistemas de tareas.

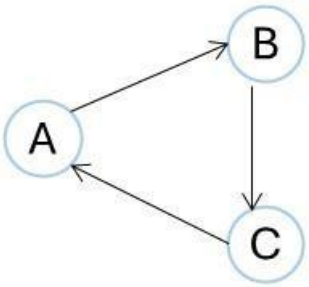


Figura 2.1. Grafo dirigido de 3 vértices con aristas que apuntan a desde un vértice a otro.

Grafos no dirigidos

En un grafo no dirigido (Figura 2.2), las aristas no tienen dirección; representan relaciones bidireccionales entre los vértices (Bondy & Murty, 2008). Matemáticamente, cada arista se expresa como un conjunto no ordenado $\{u, v\}$, lo que indica que la conexión entre u y v es mutua.

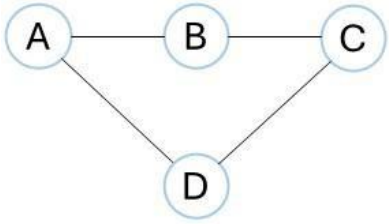


Figura 2.2. Ejemplo de grafo no dirigido

Aplicaciones comunes:

- Redes sociales, donde una amistad es recíproca.
- Sistemas de comunicación entre nodos interconectados.

- Problemas de optimización combinatoria, como el Máximo Conjunto Independiente (MIS), ya que las relaciones no dependen de una dirección.

Grafos ponderados

Un grafo ponderado (Figura 2.3) asigna un valor numérico (peso) a cada arista. Este peso puede representar distancia, costo, tiempo, capacidad o probabilidad, según el contexto del problema.

Formalmente, se define una función de ponderación $w\colon E \to \mathbb{R}$, que asigna un número real a cada arista $e \in E$.

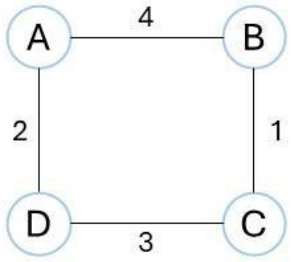


Figura 2.3. Ejemplo de grafo ponderado

Aplicaciones comunes:

- Rutas más cortas (algoritmos de Dijkstra o Bellman-Ford).
- Optimización de redes (redes eléctricas, transporte, comunicaciones).
- Análisis de costos y flujos en sistemas económicos o logísticos.

Grafos no ponderados

En los grafos no ponderados (Figura 2.4), todas las aristas se consideran equivalentes, es decir, no existe un peso asociado a las conexiones. Este tipo de grafo se usa cuando lo importante es la existencia o ausencia de una conexión, sin importar su intensidad, costo o distancia.

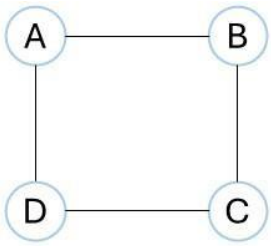


Figura 2.4. Ejemplo de grafo no ponderado

Aplicaciones comunes:

- Representación de relaciones simples, como amistades o enlaces binarios.
- Problemas combinatorios donde solo interesa la conectividad (coloración, independencia, emparejamiento).
- Modelos teóricos en investigación de estructuras de grafos.

Tabla 2.1. Resumen comparativo de los tipos de grafos

Tipo de grafo	Dirección	Peso	Ejemplo típico	Aplicación principal
Dirigido	Sí	Opcional	Redes de flujo	Modelado de procesos y dependencias
No dirigido	No	Opcional	Redes sociales	Relaciones simétricas
Ponderado	Puede tener o no dirección	Sí	Rutas más cortas	Optimización de caminos
No ponderado	Puede tener o no dirección	No	Conectividad simple	Análisis estructural

Un concepto importante en la teoría de grafos es el grado de un vértice, que representa el número de aristas incidentes en él. Este atributo permite definir

algoritmos basados en prioridades, como los que se aplican para hallar conjuntos independientes.

Un conjunto independiente $S \subseteq V$ se define como un subconjunto de vértices tal que ningún par de vértices dentro de S está conectado por una arista en E. Es decir, para todo par $(u, v) \in S$, $(u, v) \notin E$. Un conjunto independiente máximo (MIS, por sus siglas en inglés) es aquel conjunto independiente con la mayor cantidad posible de vértices. Encontrar este conjunto constituye un problema clásico en la teoría de grafos y se clasifica como NP-difícil, lo cual implica que no existe un algoritmo conocido que lo resuelva en tiempo polinomial para todos los casos [4].

El problema del MIS se relaciona estrechamente con otros problemas fundamentales en optimización combinatoria, como el problema del clique máximo, el problema del conjunto dominante mínimo y el problema de cobertura de vértices. En particular, el MIS es el complemento del problema del clique máximo: un conjunto independiente en un grafo (G) corresponde a un clique en su grafo complementario ($\bar{G}$).

2.2 Algoritmos para conjuntos independientes

Los algoritmos para encontrar conjuntos independientes se dividen principalmente en tres categorías: exactos, aproximados y heurísticos/metaheurísticos. Cada tipo difiere en precisión, tiempo de ejecución y aplicabilidad.

Algoritmos exactos.

Los algoritmos exactos buscan la solución óptima, es decir, el conjunto independiente máximo. Métodos clásicos incluyen la búsqueda exhaustiva, el retroceso (backtracking), y técnicas de poda como *branch and bound*. Aunque garantizan resultados exactos, su complejidad temporal crece exponencialmente con el número de vértices. En casos pequeños o medianos, estos métodos son adecuados, pero resultan inviables en grafos de gran tamaño [14].

Otra estrategia exacta consiste en modelar el problema mediante programación entera. En este enfoque, se asocia a cada vértice (v_i) una variable binaria (x_i) tal que (x_i = 1) si el vértice pertenece al conjunto independiente, y (*0*) en caso contrario. El modelo busca maximizar la suma de vértices seleccionados con restricciones que impiden incluir vértices adyacentes. Este método permite utilizar solvers de optimización como CPLEX o Gurobi, aunque su eficiencia depende del tamaño del grafo [12].

Algoritmos aproximados.

Dado que el problema es NP-difícil, los algoritmos aproximados ofrecen soluciones cercanas a la óptima en tiempos razonables. Estos algoritmos aplican estrategias de selección progresiva de vértices, priorizando aquellos con ciertas características. Un ejemplo clásico es el método *Greedy*, que selecciona en cada paso el vértice de menor grado, lo agrega al conjunto independiente y elimina del grafo al vértice y sus vecinos. Este proceso se repite hasta vaciar el conjunto de vértices disponibles. Aunque no garantiza la optimalidad, su tiempo de ejecución es polinomial, aproximadamente $O(n^2)$ [8].

Los métodos aproximados pueden incorporar ponderaciones para representar la importancia o el peso de cada vértice. En ese caso, el objetivo es maximizar la suma de los pesos del conjunto independiente. Estas variantes se conocen como problemas de conjunto independiente ponderado (Weighted Independent Set Problem, WISP) y tienen aplicaciones en planificación y optimización de recursos.

Algoritmos heurísticos y metaheurísticos.

Los métodos heurísticos no garantizan una solución óptima, pero pueden hallar soluciones de alta calidad con un costo computacional bajo. En particular, las heurísticas polinomiales se basan en reglas locales simples, como seleccionar vértices con grado mínimo o eliminar iterativamente vértices adyacentes a los ya seleccionados. Este tipo de heurística resulta especialmente útil cuando se requiere una respuesta rápida para grafos grandes.

Por otro lado, las metaheurísticas amplían estas ideas utilizando mecanismos de búsqueda más globales. Ejemplos comunes son el Recocido Simulado (Simulated

Annealing), los Algoritmos Genéticos, la Búsqueda Tabú, y las estrategias Híbridas. Estos métodos permiten escapar de óptimos locales mediante exploración controlada del espacio de soluciones [3].

2.3 Algoritmos para hallar el conjunto independiente máximo (MIS)

El cálculo del MIS ha sido objeto de extensos estudios en el campo de la computación teórica. A continuación, se describen los principales enfoques empleados para resolverlo.

1. Métodos exactos.

El enfoque exacto más directo es la búsqueda exhaustiva, en la que se exploran todas las combinaciones posibles de vértices para determinar cuál cumple con la independencia y tiene el mayor tamaño. Aunque su complejidad es $O(2^n)$, se usa como base para análisis teóricos y validación de otros métodos. El *branch and bound* mejora la búsqueda exhaustiva al limitar las combinaciones mediante cotas superiores e inferiores del tamaño máximo alcanzable.

La programación entera constituye otra alternativa exacta. En este modelo, las restricciones se formulan de modo que ningún par de vértices adyacentes pueda pertenecer simultáneamente al conjunto. Esto se expresa mediante una matriz de adyacencia y un conjunto de restricciones lineales. Sin embargo, el número de restricciones crece rápidamente con el tamaño del grafo, lo que limita su aplicabilidad práctica [12].

2. Métodos aproximados.

Los métodos aproximados aplican criterios determinísticos o probabilísticos para construir un conjunto independiente de forma iterativa. En el enfoque Greedy, se escoge un vértice con la característica deseada (por ejemplo, el menor grado o mayor peso relativo), se agrega al conjunto independiente y se eliminan

sus vecinos. El proceso se repite hasta no quedar vértices disponibles. Aunque no es óptimo, en grafos esparsos o de baja densidad produce resultados cercanos al máximo con bajo costo computacional [6].

3. Heurísticas polinomiales.

Las heurísticas polinomiales para el MIS combinan estrategias locales con decisiones adaptativas. Por ejemplo, una heurística típica selecciona un vértice aleatorio, lo incorpora al conjunto y elimina sus adyacentes. Posteriormente, evalúa los vértices restantes según una función de prioridad basada en su grado y la densidad local del grafo. Este proceso tiene un tiempo de ejecución entre $O(n)$ y $O(n^2)$, dependiendo de la implementación.

Otras heurísticas utilizan procedimientos de mejora local, donde a partir de un conjunto independiente inicial se buscan intercambios que aumenten su tamaño. Este enfoque se usa como punto de partida para metaheurísticas más avanzadas.

4. Metaheurísticas.

Entre las metaheurísticas más utilizadas se encuentran:

- **Recocido Simulado (Simulated Annealing):** inspirado en el proceso físico de enfriamiento de metales, este algoritmo explora el espacio de soluciones permitiendo aceptar ocasionalmente soluciones peores, con una probabilidad decreciente controlada por una temperatura. Esto permite escapar de óptimos locales y explorar regiones más amplias del espacio [9].
- **Algoritmos Genéticos:** basados en los principios de la evolución natural, representan posibles conjuntos independientes como cromosomas. Mediante operadores de selección, cruce y mutación, generan nuevas soluciones que evolucionan hacia mejores conjuntos. Este método ha mostrado buenos resultados en grafos de gran tamaño [7].
- **Búsqueda Tabú:** mantiene una lista de soluciones o movimientos prohibidos para evitar ciclos y mejorar la diversidad de la búsqueda.

Permite una exploración más sistemática y evita repetir configuraciones ya evaluadas [5].

- **Métodos híbridos:** combinan heurísticas rápidas con metaheurísticas para aprovechar la exploración local y global. Por ejemplo, se puede generar una solución inicial con Greedy y luego mejorarla mediante recocido simulado o búsqueda tabú.

En todos estos métodos, el balance entre precisión y tiempo de ejecución es clave. Las heurísticas polinomiales permiten resolver grafos grandes rápidamente, mientras que las metaheurísticas logran soluciones más cercanas al óptimo, aunque con mayor costo computacional.

2.3 Aplicaciones del MIS

El problema del conjunto independiente máximo tiene múltiples aplicaciones en diferentes áreas de la computación, la ingeniería y la biología computacional. A continuación, se describen algunas de las más relevantes.

1. Redes inalámbricas y comunicación.

En sistemas de comunicación, el MIS se aplica para optimizar el uso del espectro en redes inalámbricas. Los vértices representan estaciones o nodos, y las aristas representan interferencia entre ellos. Un conjunto independiente máximo corresponde al mayor número de estaciones que pueden operar simultáneamente sin interferirse [16].

2. Programación y planificación de tareas.

En la planificación de procesos o tareas con restricciones de conflicto, cada vértice representa una tarea y las aristas representan incompatibilidades. El conjunto independiente máximo representa el subconjunto más grande de tareas que pueden ejecutarse sin conflicto [10].

3. Análisis de redes sociales.

En análisis de redes, los vértices pueden representar individuos y las aristas relaciones entre ellos. El MIS permite identificar grupos de usuarios que no tienen conexiones directas, útil para segmentar comunidades o analizar estructuras sociales independientes [13].

4. Bioinformática y genómica.

El MIS se aplica en el análisis de secuencias genómicas, donde cada vértice representa una subsecuencia o fragmento de ADN, y las aristas indican solapamiento o dependencia. El conjunto independiente máximo identifica el mayor grupo de fragmentos que pueden analizarse simultáneamente sin redundancia, optimizando la reconstrucción de secuencias [1].

5. Optimización de recursos y diseño de redes.

En ingeniería, el MIS se emplea para seleccionar subconjuntos de elementos no conflictivos, como sensores o módulos, en un diseño donde ciertas conexiones no pueden coexistir. De esta manera, se optimiza el uso de recursos limitados y se maximiza la cobertura del sistema [11].

3. ALGORITMO IMPLEMENTADO PARA EL MIS

Como hemos descrito antes, un conjunto independiente es un subconjunto de vértices en un grafo donde ningún par de vértices está conectado por una arista. Esto significa que, si se seleccionan dos vértices cualesquiera de un conjunto independiente, estos no comparten una arista entre ellos. La idea es independiente porque la elección de un vértice en el conjunto no está conectada a ningún otro vértice del mismo conjunto. Un conjunto independiente se vuelve maximal (MIS), si cumple con la condición de que no hay ningún vértice fuera del conjunto independiente que pueda unirse a él sin romper la propiedad de independencia, es decir, que un conjunto independiente maximal jamás será un subconjunto de otro conjunto independiente. En un mismo grafo puede haber distintos conjuntos independientes maximales de diferente tamaño (diferente número de vértices).

De esta forma el conjunto independiente máximo es solamente el conjunto independiente maximal de mayor tamaño (mayor número de vértices). Básicamente, un conjunto independiente máximo es lo opuesto a un clique máximo.

Dado que el problema de encontrar el conjunto independiente máximo (CIM) es NP- completo, no se conoce un algoritmo eficiente para todos los casos. Sin embargo, un algoritmo común y fácil de implementar, aunque no garantiza la solución óptima, es el algoritmo greedy (voraz), que encuentra un conjunto maximal e independiente (pero no necesariamente el máximo), el cual podemos resumir como se describe a continuación:

```
Maximal_Indep  (V)
Inicio
        M= ∅
        Mientras el conjunto de vértices V no esté vacío
        Inicio
                Seleccionar un vértice v de V
                M = M ∪ { v }
                V = V / (N ( v ) ∪ { v })
        Fin_Mientras
        Devolver M
Fin
```

Donde *V* comienza siendo el conjunto de todos los vértices del grafo, y donde *M* comienza siendo un conjunto vacío al cual se le agrega un vértice v y todos aquellos vértices que no son sus vecinos.

3.1 Pseudocódigo del algoritmo aleatorio

El algoritmo propuesto para el desarrollo del sistema se plantea primero el pseudocódigo para generar un grafo aleatorio de la siguiente forma:

```
FUNCION generar_grafo_aleatorio(n, m)
INICIO
  maxAristas <- n × (n−1)/2
  SI (m > maxAristas) ENTONCES
    m <- maxAristas
  FIN_SI
  Dimension posibles []
  longitud <- 0
  PARA i <- 0 HASTA n - 1 HACER
    PARA j <- i + 1 HASTA n - 1 HACER
      longitud <- longitud +1
      posibles[longitud] <- i, $j];
      longitud <- longitud +1
      posibles[longitud] <- j
    FIN_PARA
  FIN_PARA
  RANDOM(posibles)
  Dimension elegidas[m+1]
  PARA i <- 0 HASTA m HACER
    elegidas = posibles[i]
  FIN_PARA
  Retornar elegidas
FIN_FUNCION
```

En la Figura 3.1 se muestra un ejemplo de la ejecución del sistema con la implementación del algoritmo aleatorio para generar el grafo aleatorio.

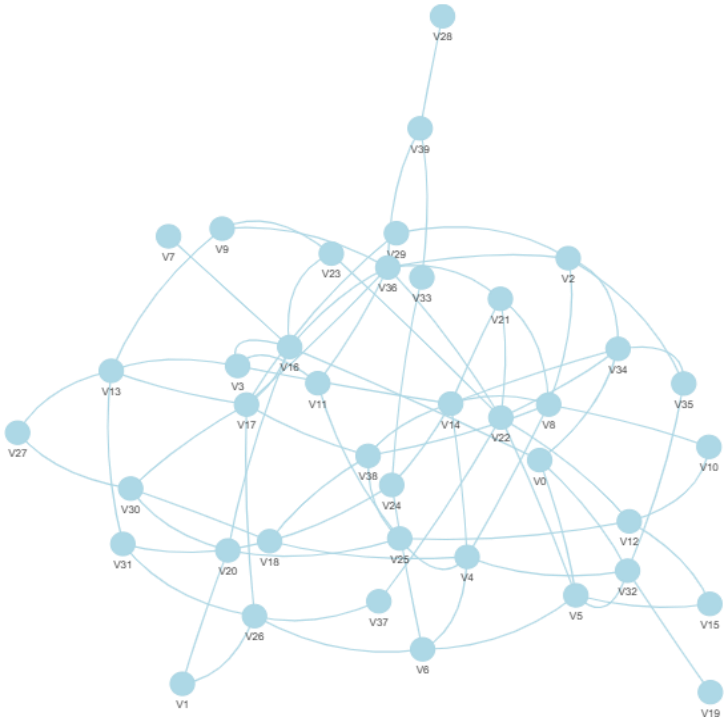


Figura 3.1 Ejemplo de la generación de un grafo aleatorio de 40 vértices y 80 aristas

3.2 Pseudocódigo del algoritmo para el MIS

A Continuación, se describe el pseudocódigo del algoritmo implementado para calcular el MIS Greedy.

```
FUNCION calcular_MIS(grafo)
INICIO
  n <- LEN(grafo)
  Dimension adj[n]
  PARA i <- 1 HASTA LEN(grafo) HACER
    u <- grafo[i][1]
    v <- grafo[i][2]
    adj[u] <- v
    adj[v] <- u
  FIN_PARA
  Dimension MIS []
  Dimension usados [n]
  PARA i <- 1 HASTA n HACER
    usados[i] <- 0
```

```
 FIN_PARA
 longitud <- 0
 PARA i <- 0 HASTA n - 1 HACER
   SI (usados[i]=0) ENTONCES
     longitud <- longitud + 1
     MIS[longitud] <- i
     PARA j <- 0 HASTA LEN(adj) HACER
       usados[adj[j]] <- 1
     FIN_PARA
   FIN_SI
 FIN_PARA
 cambios <- 1;
 longitud_mis <- 0
 MIENTRAS (cambios=1) HACER
   cambios <- 0
   PARA i <- 0 HASTA n -1 HACER
     bandera <- 0
     PARA j <- 1 HASTA LEN(MIS) HACER
       SI MIS[j]=i ENTONCES bandera <- 1
       FIN_SI
     FIN_PARA
     SI (NOT bandera) ENTONCES
       puedeAgregar = 1
       PARA k <- 1 HASTA LEN(adj) HACER
         bandera <- 0
         PARA j <- 1 HASTA LEN(MIS) HACER
           SI MIS[j]=adj[k] ENTONCES bandera <- 1
         FIN_PARA
         SI (bandera=1) ENTONCES
           puedeAgregar <- 0
           SALIR_CICLO
         FIN_SI
       FIN_SI
       SI (puedeAgregar=1) ENTONCES
         longitud_mis <- longitud_mis +1
         MIS[longitud] <- i
         cambios <- 1
         SALIR_CICLO
       FIN_SI
     FIN_SI
   FIN_PARA
 FIN_MIENTRAS
 Retornar MIS
FIN_FUNCION
```

En la Figura 3.2 se muestra el cálculo del algoritmo Greedy para el MIS implementando con el grafo aleatorio anteriormente generado.

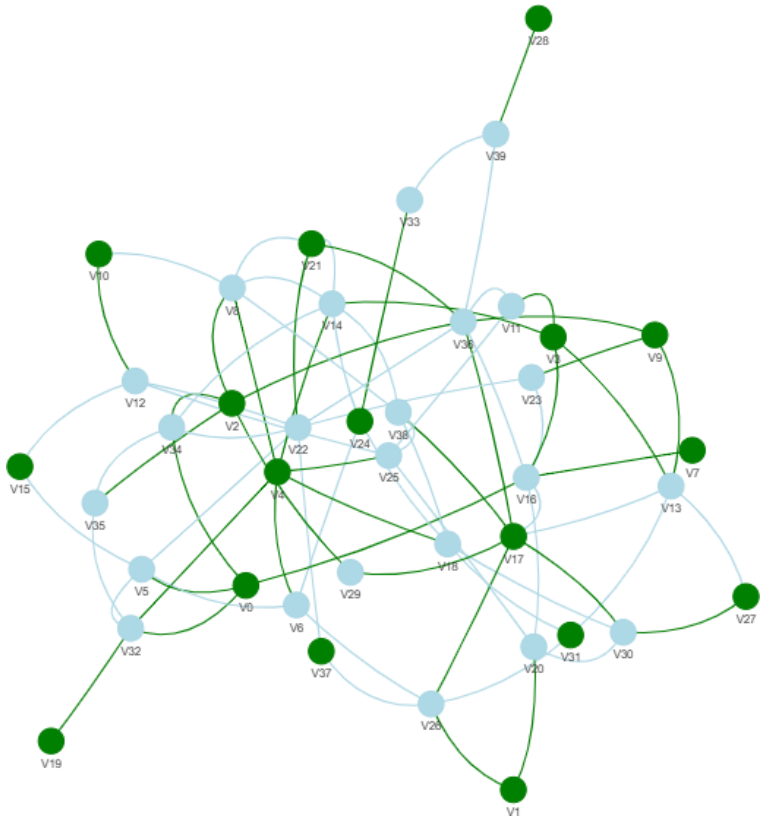


Figura 3.2. Cálculo del MIS con el Algoritmo Greedy, en color Verde se ilustra el MIS obtenido Tamaño del MIS: 17 Nodos: 0, 1, 2, 3, 4, 7, 9, 10, 15, 17, 19, 21, 24, 27, 28, 31, 37

3.3 Pseudocódigo para el MIS exacto

El cálculo de forma exacta del Máximo Conjunto Independiente (MIS) en un grafo ha sido motivo de estudios mediante diversas estrategias algorítmicas, todas con garantía de optimalidad, pero con complejidad exponencial en el peor caso debido a la naturaleza NP-completa del problema. Los métodos más básicos emplean búsqueda exhaustiva, evaluando todas las posibles combinaciones de vértices, mientras que los algoritmos de backtracking y Branch & Bound reducen significativamente el espacio de búsqueda mediante técnicas de poda basadas en límites superiores [25]. Entre los avances más relevantes se encuentran los

algoritmos propuestos por Robson [23, 24], que establecen algunos de los mejores tiempos teóricos conocidos para grafos generales. Otra estrategia ampliamente usada consiste en transformar el problema del MIS en uno equivalente de clique máximo en el grafo complemento, lo que posibilita aprovechar algoritmos altamente optimizados como los de Bron y Kerbosch (1973) o mejoras posteriores de Tomita, Tanaka y Takahashi [26]. Asimismo, formulaciones basadas en Programación Entera y en SAT solvers permiten resolver el MIS exacto a través de optimización y propagación de restricciones [22]. Finalmente, para grafos con propiedades estructurales particulares, como árboles, bipartitos o planares se dispone de algoritmos polinomiales fundamentados en programación dinámica o en reducciones eficientes [21], lo que mejora sustancialmente su tratabilidad computacional.

A Continuación, describimos el pseudocódigo utilizado para el cálculo del MIS de forma exacta usando la estrategia de backtracking, esto es útil para grafos de tamaño mediano.

```
FUNCION calcular_MIS_exacto(grafo:)
INICIO
  n <- LEN(grafo)
  Dimension adj[n]
  PARA i <- 1 HASTA LEN(grafo) HACER
    u <- grafo[i][1]
    v <- grafo[i][2]
    adj[u] <- v
    adj[v] <- u
  FIN_PARA
  Dimension maxMIS []
FIN_FUNCION

// Descripción de la función para backtraking

FUNCION backtrack(current, MIS, pos, adj, maxMIS, n)
INICIO
    SI (pos =n) ENTONCES
      SI LEN(MIS) > LEN(maxMIS) ENTONCES maxMIS <- $MIS
      FIN_SI
      Retornar
    FIN_SI
    SI (LEN(MIS) + (n - pos) <= LEN(maxMIS)) ENTONCES Retornar
    FIN_SI
    puedeAgregar <- 1
```

```
    PARA i <- 1 HASTA LEN(adj) HACER
      bandera <- 0
      PARA j <- 1 HASTA LEN(MIS) HACER
        SI MIS[j]=adj[i] ENTONCES bandera <- 1
       FIN_PARA
      SI (bandera=1) ENTONCES
        puedeAgregar <- 0
        SALIR_CICLO
      FIN_SI
    FIN_PARA
    SI (puedeAgregar=1) ENTONCES
      MIS_nuevo <- MIS
      MIS_nuevo[LEN(MIS_nuevo)+1] <- pos
      backtrack($MIS_nuevo, $MIS_nuevo, $pos + 1, $adj, $maxMIS, $n);
    FIN_SI
    backtrack(current, MIS, pos + 1, adj, maxMIS, n)
  FIN_PARA
  backtrack([], [], 0, adj, maxMIS, n)
  Retornar maxMIS
FIN_FUNCION
FIN
```

En la Figura 3.3 se muestra la aplicación del algoritmo bactraking para calcular el MIS exacto Con el mismo grafo de la Figura 3.1.

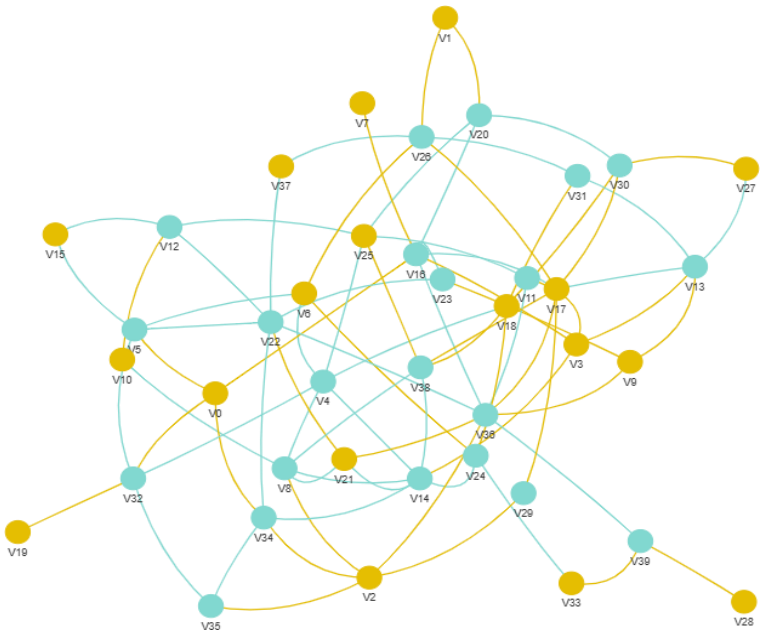


Figura 3.3. MIS exacto indicado en color amarillo. Tamaño del MIS: 18 Nodos: 0, 1, 2, 3, 6, 7, 9, 10, 15, 17, 18, 19, 21, 25, 27, 28, 33, 37

4. SISTEMA PARA EL CÁLCULO DEL MIS

A continuación, se describe las herramientas y tecnología utilizada en el desarrollo del sistema web para el cálculo del MIS.

4.1 Herramientas utilizadas para el desarrollo

Para el desarrollo del sistema destinado al cálculo del Máximo Conjunto Independiente (MIS), se emplearon diversos recursos de hardware y software que permitieron implementar, probar y visualizar el algoritmo heurístico propuesto dentro de un entorno funcional y accesible. A continuación, se describen las principales herramientas utilizadas.

Equipo de cómputo

El desarrollo se realizó en una computadora portátil HP Pavilion x360 Convertible 14-cd1xxx, con sistema operativo Windows 11 Home Single Language, versión 24H2.
Las especificaciones técnicas del equipo fueron las siguientes (Figura 4.1):

- Procesador: Intel® Core™ i3-8145U CPU @ 2.10 GHz (2.30 GHz máx.)
- Memoria RAM: 4 GB (3.83 GB utilizable)
- Almacenamiento: Disco duro de 466 GB (88 GB utilizados)
- Tarjeta gráfica: Intel® UHD Graphics 620 con 128 MB de memoria gráfica
- Tipo de sistema: 64 bits, procesador x64
- Compatibilidad táctil: Pantalla táctil con soporte para 10 puntos de contacto
- Sistema instalado el: 19 de junio de 2025
- Compilación del SO: 26100.4351

Figura 4.1. Captura de las especificaciones del dispositivo.

Este equipo ofreció el entorno adecuado para la programación, ejecución y prueba del sistema, garantizando una respuesta estable y un consumo moderado de recursos, compatible con los requerimientos del entorno WampServer. El sistema operativo Windows tuvo compatibilidad total con el servidor local Apache, PHP y MySQL.

WampServer

Se utilizó WampServer como el entorno de desarrollo web local. Este software integra en un mismo paquete los servicios de Apache Server, PHP y MySQL, lo que facilitó montar y ejecutar aplicaciones web de manera local sin necesidad de un servidor remoto. WampServer facilita la administración de bases de datos, la ejecución de scripts PHP y la gestión de sitios web en desarrollo, siendo una herramienta excelente para este tipo de proyectos de programación web.

Ventajas:

- Facilidad de instalación y configuración.
- Compatibilidad con múltiples versiones de PHP y Apache.
- Entorno ideal para pruebas de algoritmos antes de su despliegue final.

Lenguaje PHP

El lenguaje PHP (Hypertext Preprocessor) se empleó para implementar la lógica del sistema y procesar las operaciones asociadas al cálculo del Máximo Conjunto Independiente.

PHP es un lenguaje de programación del lado del servidor ampliamente usado en el desarrollo de aplicaciones web dinámicas. Su integración nativa con Apache dentro de WampServer permitió gestionar los procesos internos del sistema, realizar cálculos y generar los resultados de forma dinámica.

Además, su sintaxis sencilla y la disponibilidad de funciones matemáticas y de manejo de estructuras de datos lo convirtieron en una opción práctica el desarrollo de este proyecto.

CSS3 (Cascading Style Sheets)

Para el diseño visual del sistema, se utilizó CSS3, que permitió definir el estilo, la distribución y los elementos gráficos de la interfaz.
Gracias a CSS3, se logró un entorno visual sobrio, con una estructura simple para el usuario, que facilita la interpretación de los resultados del algoritmo MIS y mejora la experiencia de navegación dentro del sistema.

Vis.js

La biblioteca Vis.js se implementó para la visualización gráfica de los grafos generados durante la ejecución del algoritmo.
Vis.js es una herramienta de JavaScript especializada en representar nodos y aristas interactivas, ideal para proyectos de teoría de grafos. Su integración permitió mostrar los vértices, las conexiones y los subconjuntos independientes de manera visual e intuitiva, favoreciendo la comprensión de los resultados obtenidos por el sistema.

Entre sus principales ventajas destacan:

- Interactividad mediante arrastre, zoom y resaltado de nodos.
- Visualización dinámica de grafos con propiedades personalizables.
- Compatibilidad directa con HTML, CSS y PHP en entornos web locales.

Editor de código fuente

El código fuente se escribió en el editor Visual Studio Code, una herramienta desarrollada por Microsoft que permite la edición de código multiplataforma, la instalación de extensiones para depuración y control de versiones, así como la integración con sistemas de gestión de bases de datos y servidores locales.

4.2 Diseño y funciones de la interfaz de usuario

Aquí se describe el funcionamiento del sistema a través de pantallas para mostrar la funcionalidad en el cálculo del MIS.

4.2.1 Pantalla de Inicio

La interfaz del sistema para el cálculo del MIS presenta una pantalla inicial simple con las siguientes opciones en la parte superior en forma de pestañas: Inicio, Calcular MIS, MIS aleatorio y Contacto. Posteriormente se visualiza una breve explicación de lo que son los conjuntos independientes con la imagen de ejemplo sencillo de un grafo y del MIS coloreado en amarillo Figura 4.1.

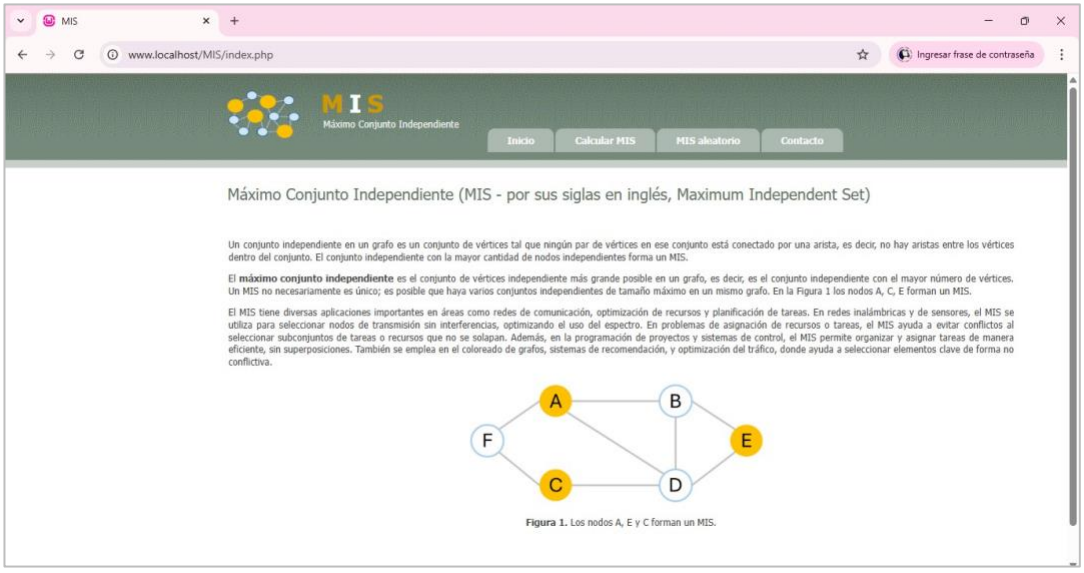


Figura 4.1 Interfaz inicial del sistema.

Esta pantalla inicial la podemos visualizar cada vez que se seleccione la pestaña "Inicio".

Pestaña Calcular MIS

La siguiente pestaña a la que se puede acceder es "Calcular MIS" la cual ofrece las siguientes acciones a realizar mediante los siguientes botones: Seleccionar archivo, Cargar Grafo, Guardar Grafo, MIS Greedy, MIS Exacto, Agregar Nodo y Agregar (Agregar arista).

Posteriormente debajo de estas posibles acciones se encuentra un espacio donde se puede visualizar de manera grafica la estructura del Grafo. Debajo de este espacio se encuentra información del grafo como Numero de nodos y Numero de aristas. Figura 4.2.

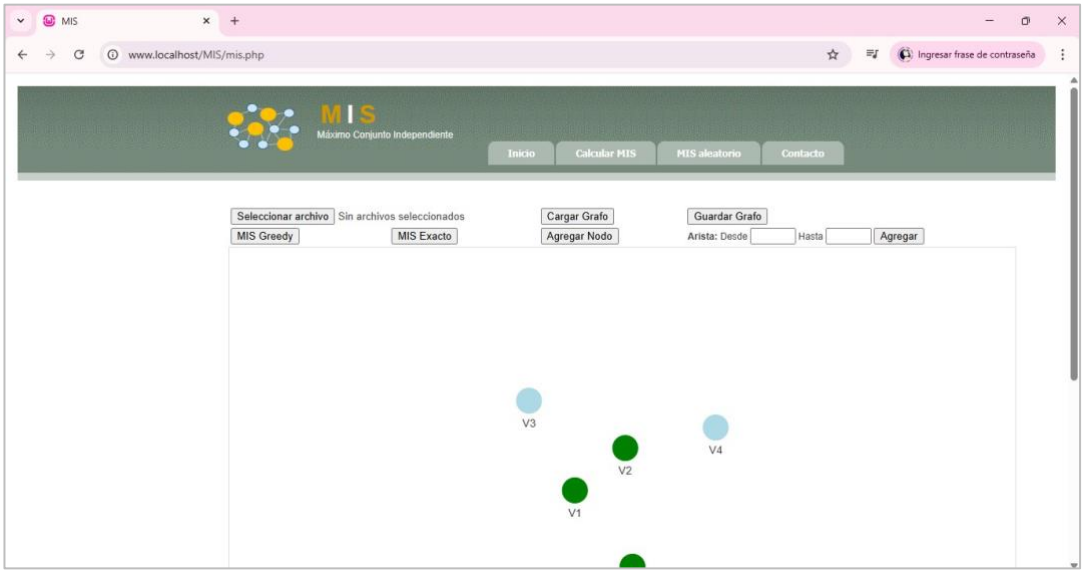


Figura 4.2 Interfaz de la pestaña **Calcular MIS**

Acontiniuacion se explica la función de cada opción de esta pantalla:

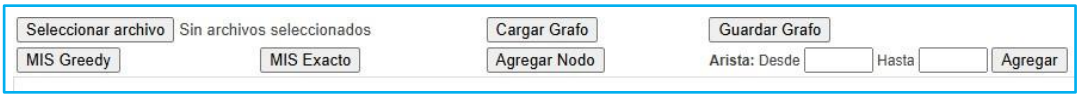


Figura 4.3. Opciones de la opción seleccionada

- Botón **Seleccionar archivo**. Permite seleccionar un grafo desde un archivo de texto en alguna ubicación específica Figura 4.3. El archivo de origen debe mantener una estructura de entrada Similar al de la figura 4.4. Es decir, al inicio de la línea el Nodo o vértice, espacio, símbolo de igual, símbolo de Mayor que, espacio, Corchete Izquierdo, Sucesión de vértices con los que tiene relación el vértice inicial separados por una coma y finalmente un corchete derecho. En el siguiente salto de línea se continua con los demás vértices.

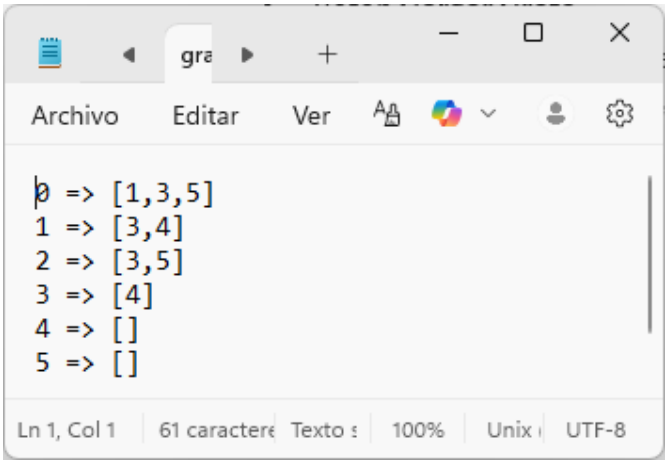


Figura 4.4 Ejemplo de estructura del archivo de entrada en el editor Block de Notas de Windows.

- Botón **Cargar Grafo**

  Este botón genera la imagen gráfica del grafo con sus vértices y aristas el grafo seleccionado previamente con el botón "Seleccionar archivo" Figura 4.5. Si se desea se puede arrastrar cada vértice para acomodar el grafo a conveniencia.

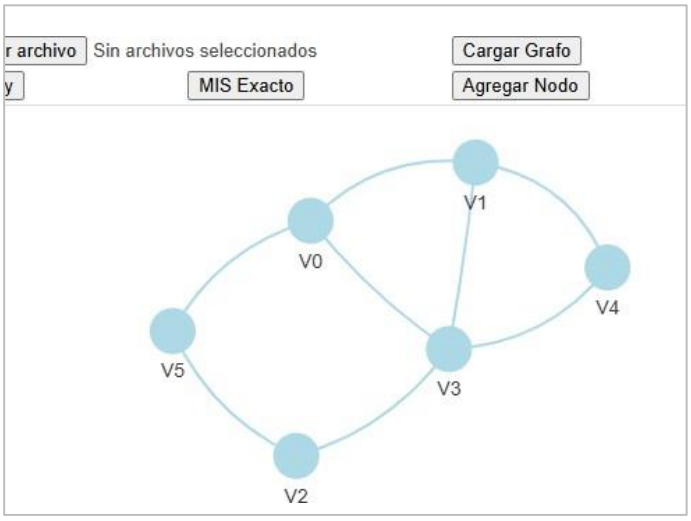


Figura 4.5 ejemplo de grafo generado desde un archivo.

- Botón **Guardar Grafo**

  El botón Guardar Grafo guarda el grafo actual en un nuevo archivo de texto de salida con el nombre grafo_modificado.txt y lo almacena en la carpeta de descargas del navegador. Figura 4.6.

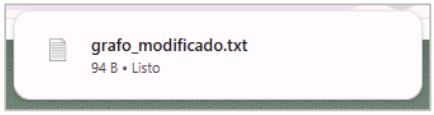


Figura 4.6. Notificación de archivo del grafo actual guardado en descargas.

- Botón **MIS Greedy**

  Este botón activa la resolución MIS mediante el algoritmo de Greedy y colorea en verde la solución obtenida. Figura 4.7.

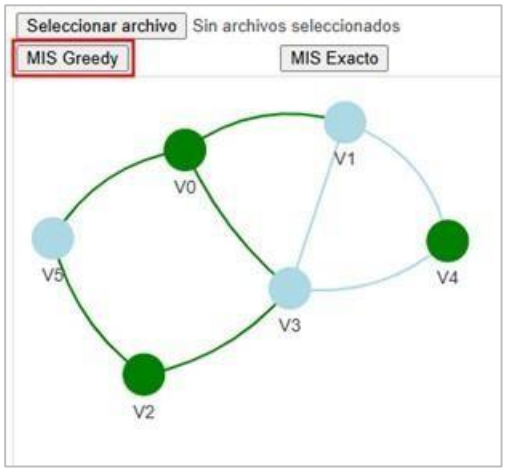


Figura 4.7 Ejemplo de solución del MIS Greedy.

- Botón **MIS Exacto**

  Este botón activa la resolución del MIS del grafo actual y lo colorea en amarillo el MIS. Figura 4.2.8.

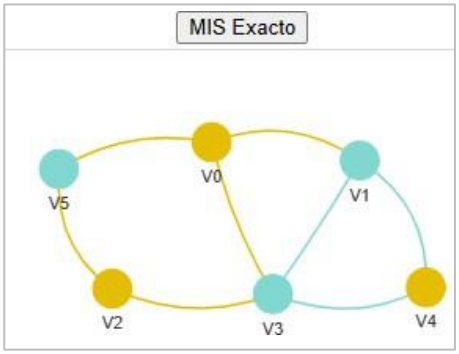


Figura 4.8. Ejemplo de solución del MIS Exacto.

- Botón **Agregar Nodo**

  Este botón agrega un nuevo vértice al grafo. Dicho vértice no tendrá relación con ningún otro Figura 4.9, a menos que se establezca una relación mediante el botón Agregar(arista) que se explica más adelante.

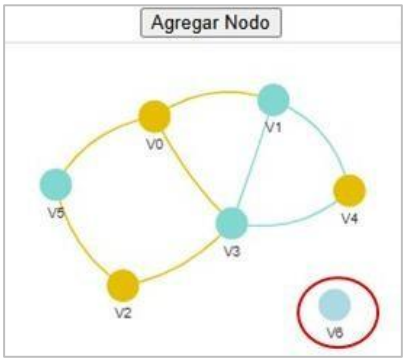


Figura 4.9. Nodo V6 agregado al grafo.

- Botón **Agregar(arista)** El botón Agregar establece una nueva arista desde un nodo origen hasta uno destino como se muestra en la Figura 4.10.

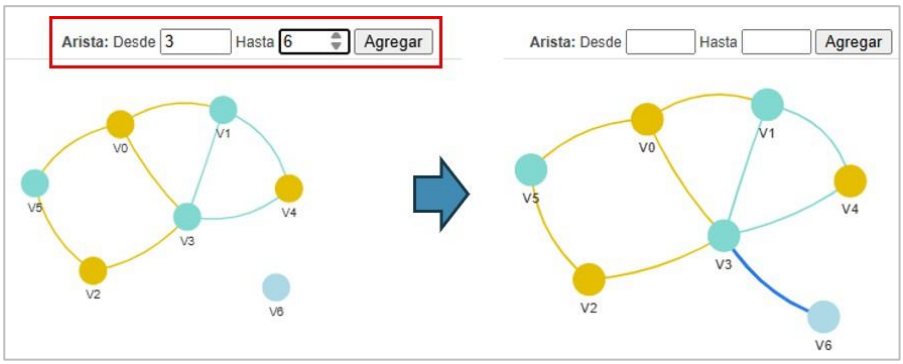


Figura 4.10. Se agrega una nueva arista desde el vértice 3 al vértice 6.

Pestaña MIS aleatorio.

La pestaña de MIS aleatorio (Figura 4.11), como en la descripción inicial se menciona, genera un grafo aleatorio especificándole el número de nodos y el umero de aristas deseados. Esta pantalla cuenta con cuatro botones: **Generar**, **Guardar .txt**, **MIS Greedy** y **MIS Exacto**.

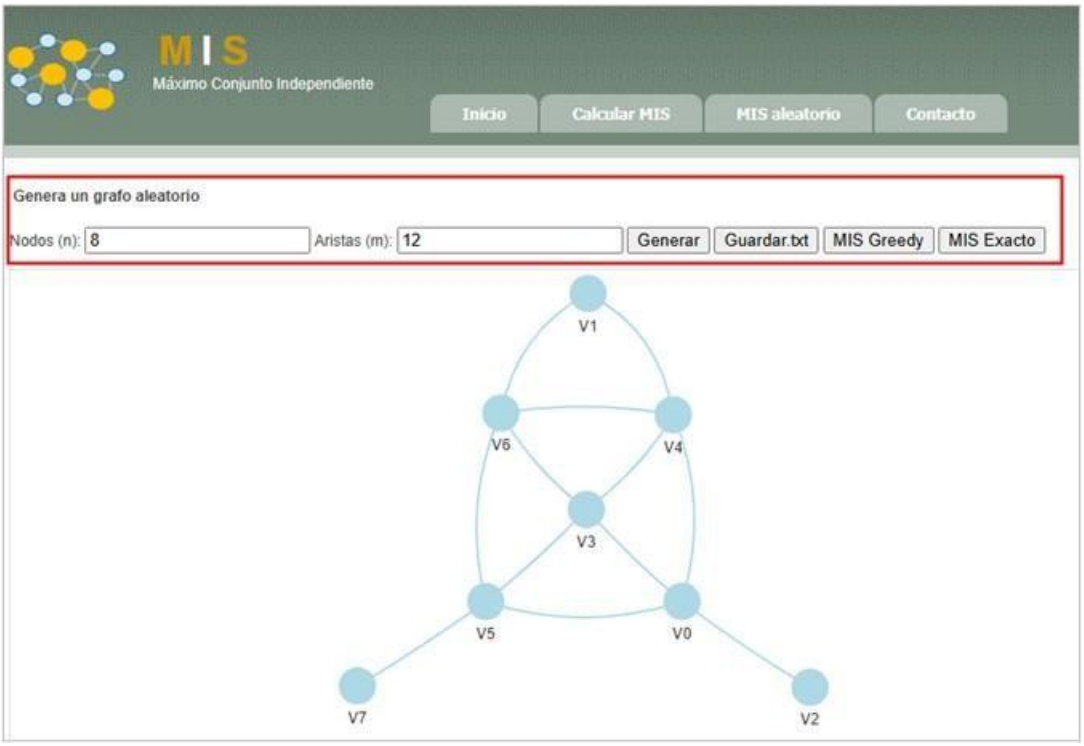


Figura 4.11. Generación aleatoria de un grafo de 8 vértices y 12 aristas

Generar un grafo aleatorio

Para dicho propósito es necesario seguir los siguientes pasos:

1- Ingresar el número de vértices deseado en la caja de texto **nodos(n)**.
2- Ingresar el número de aristas que relacionaran los vértices del grafo, en la caja de texto **Aristas(m)**.
3- Click en el botón **Generar** para visualizar el grafo.
4- Si se desea acomodar el grafo se puede manipular arrastrando los vértices a conveniencia. Cabe señalar que cada vez que se presione el botón **Generar**, se estará generando de manera aleatoria un nuevo grafo.

Guardar .txt

Para guardar el grafo en pantalla únicamente es necesario hacer click en el botón **Guardar .txt** (Figura 4.11) y el archivo se guardará en la ubicación de descarga.

Cálculo del MIS Greedy

Para obtener el **MIS Greedy** del grafo en pantalla únicamente se debe hacer click en dicho botón y aparecerá el MIS obtenido, aclarando que no siempre es efectivo para la obtención del Máximo Conjunto Independiente. El conjunto independiente encontrado se visualiza con los vértices coloreados en verde. Si se desea se puede acomodar el grafo obtenido arrastrando los vértices del mismo a conveniencia, como se muestra en la Figura 4.12.

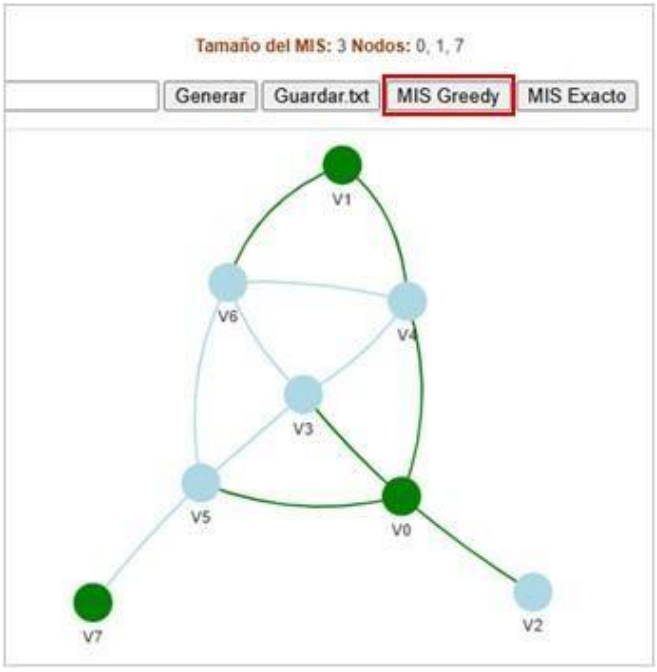


Figura 4.12. Se visualiza el cálculo del MIS mediante la técnica Greedy. MIS con 3 vértices {0,1,7}. Se puede observar de manera clara que este resultado efectivamente es un conjunto independiente, sin embargo, no es el Máximo que se pudiera obtener.

Cálculo del MIS exacto

Para calcular el Máximo Conjunto Independiente del grafo en pantalla es preciso dar click en el botón **MIS Exacto**. Dependiendo del tamaño del Grafo y el número de vértices, en un breve tiempo aparecerán coloreados los vértices que componen el Máximo Conjunto Independiente. Véase la Figura 4.13.

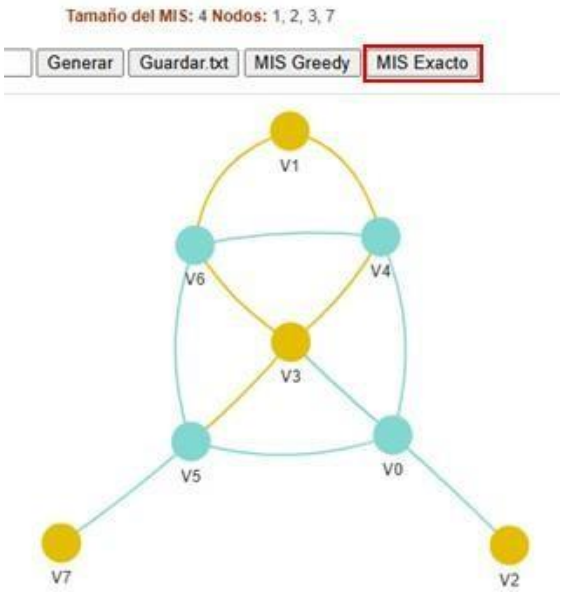
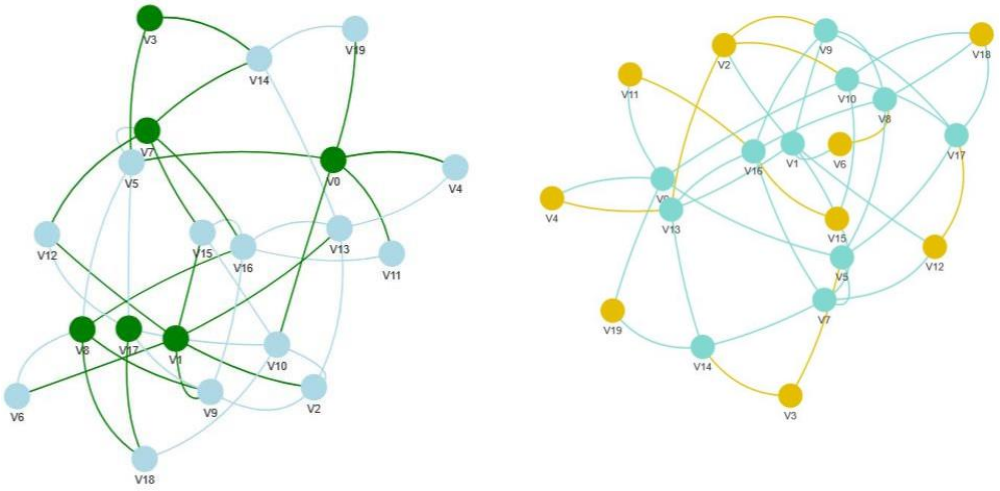


Figura 4.13 Se puede apreciar en este ejemplo el conjunto de los vértices que componen el MIS coloreados en amarillo, compuesto por los nodos {1,2,3,7}.

4.5 Pruebas con grafos aleatorios

El calculo del MIS en grafos medianamente grandes es difícil de procesar, si realizamos el calculo con el algoritmo Greedy se puede resolver, pero de forma exacta el tiempo de procesamiento se dispara y es debido a la naturaleza exponencial del problema. En la Figura 4.14 se muestra la ejecución del calculo del MIS de forma aleatoria para 20 nodos y 40 aristas, el tiempo de ejecución es inmediato en ambos casos.

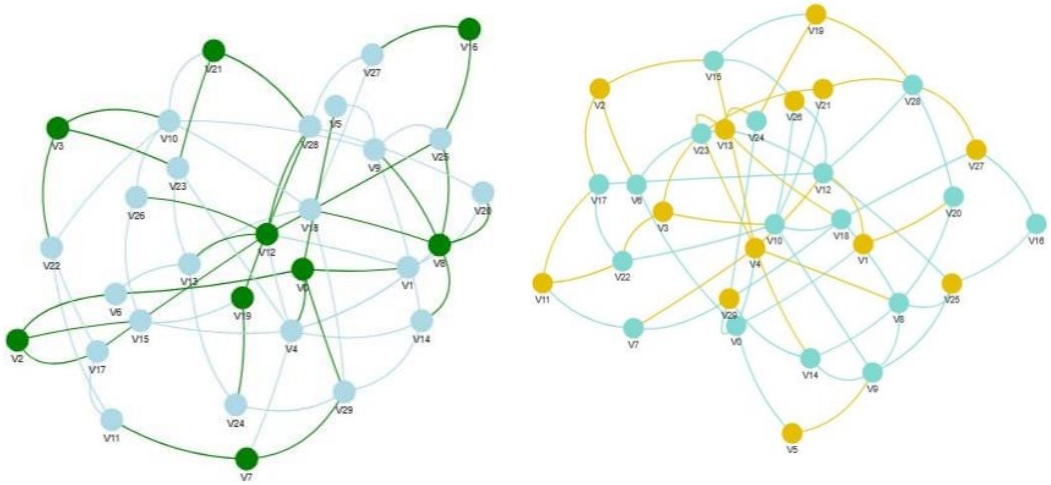


a) MIS Greedy - MIS: 6 Nodos: 0, 1, 3, 7, 8, 17

b) MIS Exacto - MIS: 9 Nodos: 2, 3, 4, 6, 11, 12, 15, 18, 19

Figura 4.14. MIS aleatorio con 20 nodos y 40 aristas

En la Figura 4.15 se genera un grafo aleatorio con 30 nodos y 60 aristas donde la ejecución en tiempo es también muy rápido en ambos casos.

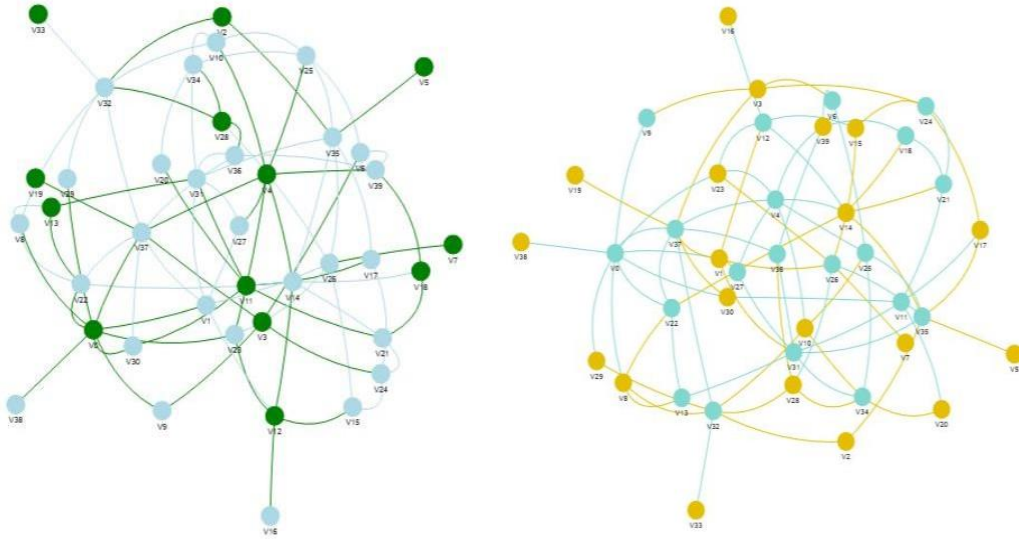


a) MIS Greedy - MIS: 9 Nodos: 0, 2, 3, 7, 8, 12, 16, 19, 21

b) MIS Exacto - MIS: 13 Nodos: 1, 2, 3, 4, 5, 11, 13, 19, 21, 25, 26, 27, 29

Figura 4.15. MIS aleatorio con 30 nodos y 60 aristas

En la Figura 4.16 se generó un grafo aleatorio con 40 nodos y 60 aristas, la generación del MIS con el algoritmo Greedy fue inmediato mientras que el cálculo del MIS exacta tardo alrededor de 10 segundos.

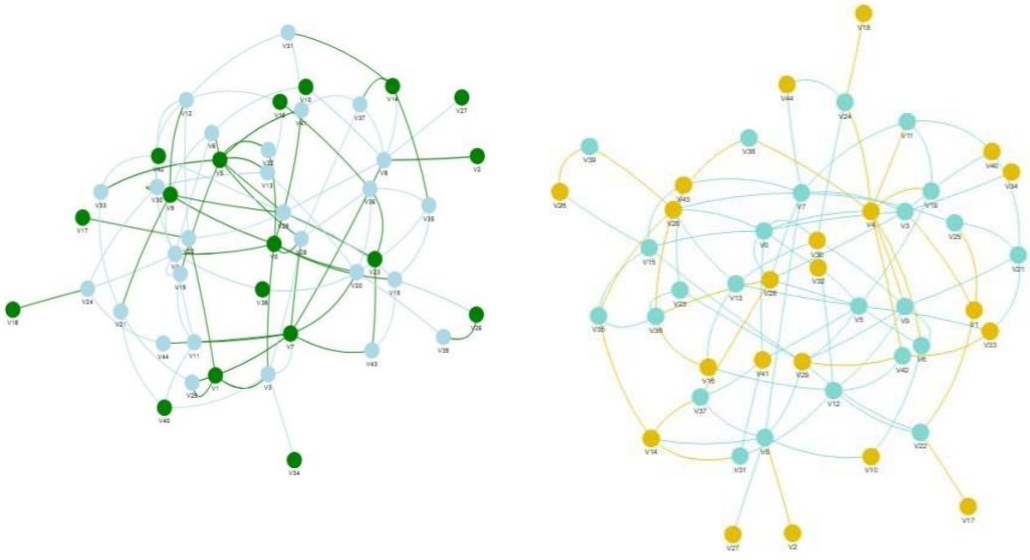


a) MIS Greedy - MIS: 13 Nodos: 0, 2, 3, 4, 5, 7, 11, 12, 13, 18, 19, 28, 33

b) MIS Exacto - MIS: 20 Nodos: 1, 2, 3, 5, 7, 8, 10, 14, 15, 16, 17, 19, 20, 23, 28, 29, 30, 33, 38, 39

Figura 4.16. MIS aleatorio con 40 nodos y 80 aristas

En la Figura 4.17 se muestra el cálculo del MIS con 45 nodos y 90 aristas, al realizar el MIS Greedy este se resolvió de forma inmediata, mientras que el MIS con el algoritmo exacto se tardó alrededor de 22 segundos, recuerde que una de las diferencias es que el MIS Greedy nos da una posible solución que no siempre es la mejor opción mientras que el algoritmo exacto nos da el MIS más grande de forma correcta.



a) MIS Greedy - MIS: 18 Nodos: 0, 1, 2, 5, 7, 9, 10, 14, 16, 17, 18, 23, 26, 27, 34, 38, 40, 42

c) MIS Exacto - MIS: 21 Nodos: 1, 2, 4, 10, 14, 16, 17, 18, 20, 26, 27, 28, 29, 30, 32, 33, 34, 40, 41, 43, 44

Figura 4.16. MIS aleatorio con 45 nodos y 90 aristas

Se intento realizar el cálculo del MIS con 50 nodos y 100 aristas y el algoritmo exacto no fue posible generase solo se obtuvieron resultados del Algoritmo Greedy.

Con el Algoritmo Greedy con 500 nodos y 500 aristas se obtuvo el grafo de la Figura 4.17 donde el MIS obtenido es de tamaño 273 que contiene los nodos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 35, 36, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54, 57, 58, 59, 60, 61, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 85, 86, 87, 89, 91, 92, 93, 94, 95, 97, 99, 101, 102, 103, 104, 105, 106, 108, 109, 110, 111, 113, 115, 116, 117, 118, 120, 122, 123, 124, 125, 126, 127, 128, 129, 130, 132, 133, 135, 137, 138, 139, 140, 141, 142, 143, 144, 145, 150, 151, 152, 153, 154, 156, 157, 158, 160, 161, 162, 163, 164, 166, 170, 171, 174, 177, 178, 179, 180, 182, 190, 191, 194, 195, 197, 198, 199, 201, 202, 203, 206, 209, 212, 213, 214, 217, 219, 220, 221, 222, 227, 228, 229, 231, 235, 240, 241, 244, 245, 246, 249, 250, 252, 253, 255, 258, 259, 268, 273, 276, 277, 278, 280, 282, 283, 284, 285, 290, 291, 292, 294, 295, 297, 298, 299, 300, 301, 306, 309, 310, 313, 315, 317, 318, 320, 323, 327, 332,

335, 336, 337, 339, 341, 344, 345, 346, 349, 352, 356, 357, 358, 361, 363, 364, 367, 370, 371, 373, 376, 378, 380, 382, 384, 386, 392, 397, 399, 402, 404, 405, 407, 409, 414, 417, 419, 420, 421, 423, 424, 431, 449, 450, 452, 456, 460, 465, 472, 477, 478, 483, 487, 488, 490, 493, 494, 495, 498**.**

Figura 4.17. MIS aleatorio con 500 nodos y 500 aristas

5. CONCLUSIONES Y PERSPECTIVAS

Este proyecto de tesis es una investigación aplicada donde se logran integrar los conocimientos y habilidades de la teoría de grafos, de matemáticas discretas, estructuras de datos, de desarrollo de software, de paradigmas de programación, de aplicaciones web y en general de la tecnología en computación que al integrarse en una aplicación disponible a través de internet facilita el uso a nivel global de personas interesadas en determinar el Máximo Conjunto Independiente, este tipo de problemas puede ser utilizado en diversas áreas y no solo en ciencias de la computación, por ejemplo en redes sociales, permite identificar grupos de usuarios sin interacciones directas, en biología ayuda a estudiar redes de proteínas y sistemas metabólicos al identificar componentes funcionales independientes.

El principal aporte es el desarrollo de un sistema en web que implementa un algoritmo para obtener el Máximo Conjunto Independiente modelado en los grafos, es decir, el presente trabajo contribuye a la ciencia básica en el área de la teoría de grafos y la optimización combinatoria, al abordar el problema del máximo conjunto independiente (MIS), uno de los problemas clásicos y fundamentales dentro de las ciencias de la computación.

En el trabajo desarrollado se obtuvo un prototipo funcional en web de fácil funcionalidad para los usuarios finales, el cual permitirá realizar pruebas con grafos de diferentes estructuras y obtener el Máximo Conjunto Independiente, generando además una vista del grafo coloreando los nodos que forman el MIS. Se realizaron diferentes ejemplos de diferentes estancias de grafos aleatorios para determinar el MIS usando un algoritmo rápido y un algoritmo exacto.

Como trabajo a futuro se está trabajando en un artículo de divulgación para dar a conocer esta herramienta que puede ser utilizada de carácter académico, además se podrían integrar otros algoritmos y comparar los resultados obtenidos y los tiempos de ejecución.

REFERENCIAS

[1] Cheng, Y., Yu, Y., & Tang, J. (2011). *Independent set based analysis of genomic sequences*. Bioinformatics Journal, 27(8), 1124–1132.

[2] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to Algorithms* (4th ed.). MIT Press.

[3] Eiben, A. E., & Smith, J. E. (2015). *Introduction to evolutionary computing* (2nd ed.). Springer.

[4] Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman.

[5] Glover, F., & Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers.

[6] Halldórsson, M. M., & Radhakrishnan, J. (1997). Greedy and local algorithms for the weighted independent set problem. *Information Processing Letters, 63*(1), 19–25.

[7] Holland, J. H. (1992). *Adaptation in natural and artificial systems*. MIT Press.

[8] Johnson, D. S., Papadimitriou, C. H., & Yannakakis, M. (2001). On generating all maximal independent sets. *Information Processing Letters, 27*(3), 119–123.

[9] Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*(4598), 671–680.

[10] Korte, B., & Vygen, J. (2018). *Combinatorial optimization: Theory and algorithms* (6th ed.). Springer.

[11] Mannino, C., Rossi, F., Smriglio, S., & Toth, P. (2019). *Optimization and decision science: Methodologies and applications*. Springer.

[12] Nemhauser, G. L., & Wolsey, L. A. (1999). *Integer and combinatorial optimization*. Wiley.

[13] Newman, M. (2018). *Networks*. Oxford University Press.

[14] Papadimitriou, C. H., & Steiglitz, K. (1998). *Combinatorial optimization: Algorithms and complexity*. Dover Publications.

[15] West, D. B. (2021). *Introduction to graph theory* (3rd ed.). Pearson.

[16] Wu, J., & Li, H. (1999). On calculating connected dominating set for efficient routing in ad hoc wireless networks. *Telecommunication Systems, 18*(1–3), 13–36.

[17] Abraham Gutiérrez, Ginés Bravo. (2005). PHP5, Alfaomega Ra-MA, México.

[18] Luke Welling, Laura Thomson. (2009). Desarrollo Web con PHP y MySQL Editorial Anaya.

[19] Braude (2003), Ingeniería de Software, una perspectiva orientada a objetos, Alfaomega, México.

[20] Ian Sommerville (2005). Ingeniería del software, Séptima edición, Pearson Educación. S.A. Madrid.

[21] Bodlaender, H. L. (1998). A partial k-arboretum of graphs with bounded treewidth. Theoretical Computer Science, 209(1–2), 1–45.

[22] Bomze, I. M., Budinich, M., Pardalos, P. M., & Pelillo, M. (1999). The maximum clique problem. In Handbook of Combinatorial Optimization (pp. 1–74). Springer.

[23] Robson, J. M. (1986). Algorithms for maximum independent sets. Journal of Algorithms, 7(3), 425–440.

[24] Robson, J. M. (2019). Worst-case optimal algorithms for maximum independent sets. Theoretical Computer Science, 812, 120–133.

[25] Tarjan, R. E., & Trojanowski, A. E. (1977). Finding a maximum independent set. SIAM Journal on Computing, 6(3), 537–546.

[26] Tomita, E., Tanaka, A., & Takahashi, H. (2006). The worst-case time complexity for generating all maximal cliques. Theoretical Computer Science, 363(1), 28–42.