Universidad Nacional Mayor de San Marcos
Universidad del Perú. Decana de América
Facultad de Ciencias Matemáticas
Escuela Profesional de Computación Científica

Extensión del Lema Local de Lovász y su aplicación en problemas minimax 0-1

TESIS

Para optar el Título Profesional de Licenciado en Computación Científica

AUTOR

Alfonso Enrique ORÉ ROSALES

ASESOR

Dra. Roxana LÓPEZ CRUZ

Lima, Perú

2025



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

https://creativecommons.org/licenses/by-nc-sa/4.0/

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Oré, A. (2025). *Extensión del Lema Local de Lovász y su aplicación en problemas minimax 0-1*. [Tesis de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ciencias Matemáticas, Escuela Profesional de Computación Científica]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Alfonso Enrique Oré Rosales
Tipo de documento de identidad	DNI
Número de documento de identidad	42714018
URL de ORCID	No aplica
Datos de asesor	
Nombres y apellidos	Roxana López Cruz
Tipo de documento de identidad	DNI
Número de documento de identidad	10798114
URL de ORCID	https://orcid.org/0000-0002-7703-5784
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Martha Hilda Timoteo Sánchez
Tipo de documento	DNI
Número de documento de identidad	25702279
Miembro del jurado 1	
Nombres y apellidos	Edinson Raúl Montoro Alegre
Tipo de documento	DNI
Número de documento de identidad	09627181
Miembro del jurado 2	
Nombres y apellidos	Roxana López Cruz
Tipo de documento	DNI
Número de documento de identidad	10798114
Datos de investigación	
Línea de investigación	No aplica
Grupo de investigación	No aplica

Agencia de financiamiento	Sin financiamiento
Ubicación geográfica de la investigación	Edificio: Ciudad Universitaria. Universidad Nacional Mayor de San Marcos País: Perú Departamento: Lima Provincia: Lima Distrito: Lima Calle: Av. Venezuela s/n cuadra 34. Latitud: -12,056 Longitud: -77,0844
Año o rango de años en que se realizó la investigación	2024-2025
URL de disciplinas OCDE	Ciencias de la computación https://purl.org/pe-repo/ocde/ford#1.02.01

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
Universidad del Perú. Decana de América
FACULTAD DE CIENCIAS MATEMÁTICAS
Escuela Profesional de Computación Científica

ACTA DE SUSTENTACIÓN DE TESIS PARA OBTENER EL TÍTULO PROFESIONAL DE LICENCIADO EN COMPUTACIÓN CIENTÍFICA

En la UNMSM, Ciudad Universitaria, Facultad de Ciencias Matemáticas, siendo las 11:00 horas del viernes 17 de octubre del 2025, se reunieron los docentes designados como miembros del Jurado Evaluador:

Presidente : Dra. Martha Hilda Timoteo Sánchez
Miembro : Dr. Edinson Raúl Montoro Alegre
Miembro asesor : Dra. Roxana López Cruz

para la sustentación de la tesis intitulada "*EXTENSIÓN DEL LEMA LOCAL DE LOVÁSZ Y SU APLICACIÓN EN PROBLEMAS MINIMAX 0-1*", presentada por el bachiller ALFONSO ENRIQUE ORÉ ROSALES, para obtener el Título Profesional de Licenciado en Computación Científica.

Luego de la exposición de la Tesis, el presidente invitó al expositor a dar respuesta a las preguntas formuladas. Realizada la evaluación correspondiente por los miembros del jurado, el expositor mereció la aprobación con Aprobado con máximos honores, con un calificativo promedio de ..diecinueve. (19.).

A continuación, los miembros del jurado, dan manifiesto que el participante Bachiller ALFONSO ENRIQUE ORÉ ROSALES, en virtud de haber aprobado la sustentación de su tesis, será propuesto para que se le otorgue el Título Profesional de Licenciado en Computación Científica.

Siendo las 11:50 horas, se levantó la Sesión, firmando para constancia la presente Acta.

Dra. Martha Hilda Timoteo Sánchez
Presidente

Dr. Edinson Raúl Montoro Alegre
Miembro

Dra. Roxana López Cruz
Miembro Asesor

UNIVERSIDAD NACIONAL MAYOR DE
SAN MARCOS
Universidad del Perú, Decana de América

CERTIFICADO DE SIMILITUD

Yo, Roxana López Cruz en mi condición de asesor acreditado con Dictamen N° 000004-2025-EPCC-FCM/UNMSM de la tesis, cuyo título es EXTENSIÓN DEL LEMA LOCAL DE LOVÁSZ Y SU APLICACIÓN EN PROBLEMAS MINIMAX 0-1, presentado por el Bachiller ALFONSO ENRIQUE ORÉ ROSALES para optar el título de Licenciado en Computación Científica.

CERTIFICO que se ha cumplido con lo establecido en la Directiva de Originalidad y de Similitud de Trabajos Académicos, de Investigación y Producción Intelectual. Según la revisión, análisis y evaluación mediante la plataforma PIDS, el documento evaluado cuenta con el 13% de similitud, nivel **PERMITIDO** para continuar con los trámites correspondientes y para su **publicación en el repositorio institucional.**

Se emite el presente certificado en cumplimiento de lo establecido en las normas vigentes, como uno de los requisitos para la obtención del título correspondiente.

Lima, 28 de abril de 2026

Firma del Asesor: 

DNI: 10798114

DEDICATORIA

Dedicado a todas las personas
que ayudaron a este
barro pensante a terminar este
único objeto de valor
que dejo hasta
ahora a mi querida
Alma Mater,
La Universidad Nacional Mayor de San Marcos.

AGRADECIMIENTOS

Agradezco a mis profesores, a mi familia por sus consejos y conocimientos brindados a mi persona.

No es posible democratizar la enseñanza de un país, sin democratizar su economía y sin democratizar, por ende, su super-estructura política. El problema de la enseñanza no puede ser bien comprendido en nuestro tiempo, si no es considerado como un problema económico y como un problema social.

José Carlos Mariátegui

Índice general

1. Introducción **4**

1.1. Objetivos 6

1.1.1. Objetivo General 6

1.1.2. Objetivos Específicos 6

1.2. Metodología 7

1.2.1. Fase 1: Revisión teórica y formulación del modelo 7

1.2.2. Fase 2: Implementación computacional 7

1.2.3. Fase 3: Optimización y pruebas 7

1.2.4. Fase 4: Validación experimental 7

2. Herramientas Utilitarias **8**

2.1. Teoría de Grafos 8

2.2. Conceptos básicos de la teoría de probabilidades 9

2.3. Algoritmos y Complejidad 11

2.3.1. Análisis de Complejidad 13

2.3.2. Estructuras de Datos 13

2.4. Algoritmos de Aproximación 13

2.4.1. Fundamentos 13

2.4.2. Ejemplo: Problema de Escalonamiento 13

2.4.3. Implementación en Python 14

2.4.4. Análisis de Complejidad 15

2.4.5. Garantías de Aproximación 15

2.4.6. Resultados Experimentales 16

2.5. Funciones Simétricas y Convexas 16

2.5.1. Implementación de Funciones Simétricas 18

2.5.2. Validación del Lema 2.22 21

2.6. Desigualdades de Concentración 22

2.6.1. Implementación de la Función G 26

2.7. Estimativas para G 27

2.7.1. Implementación de Estimativas 28

2.8. La función H 29

2.8.1. Implementación de la Función H 31

2.8.2. Análisis de Complejidad de las Implementaciones 31

3. Extensión del LLL **32**
3.1. Extensión del Lema Local de Lovász . . . 32
3.1.1. Contexto histórico y relevancia . . . 32
3.1.2. Implementación Base . . . 33
3.2. Lema Local de Lovász . . . 34
3.2.1. Visualización del Grafo de Dependencia . . . 34
3.2.2. Implementación del Dígrafo de Dependencia . . . 35
3.2.3. Implementación del LLL Simétrico . . . 38
3.3. Extensión del Lema Local de Lovász . . . 39
3.3.1. Implementación del LLL Extendido . . . 41
3.4. Conexiones con Desigualdades de Concentración y Problemas Minimax . 42

4. Problemas Minimax **43**
4.1. Definición de los problemas minimax (MIP) . . . 43
4.1.1. Implementación del Ejemplo 4.1 . . . 44
4.1.2. Implementación de Métricas MIP . . . 46
4.2. Redondeo aleatorio para MIPs . . . 47
4.2.1. Implementación del Redondeo Aleatorio . . . 47
4.2.2. Implementación del Teorema 4.6 . . . 52

5. Validación experimental de la extensión del LLL en redes WMN con datos reales **61**
5.1. Descripción del problema y contexto . . . 62
5.1.1. ¿Qué son las redes de malla inalámbricas? . . . 62
5.1.2. El problema de la interferencia . . . 62
5.1.3. Metodología de desarrollo . . . 63
5.1.4. Datos experimentales que utilizamos . . . 64
5.2. Formulación matemática del problema . . . 65
5.2.1. El modelo matemático de Żotkiewicz . . . 65
5.2.2. Elementos básicos de la red . . . 66
5.2.3. Parámetros del sistema . . . 66
5.2.4. Variables de decisión . . . 66
5.2.5. Nuestra extensión: análisis probabilístico . . . 67
5.3. Modelo de optimización (MIP) y relajación LP . . . 68
5.3.1. Formulación completa del MIP . . . 68
5.3.2. Desigualdades válidas . . . 69
5.3.3. Relajación LP . . . 70
5.4. Bloques, matriz A y métricas . . . 72
5.4.1. Definición de bloques . . . 72
5.4.2. Construcción de la matriz A . . . 72
5.4.3. Métricas fundamentales . . . 73
5.5. Redondeo aleatorio por bloques y análisis teórico . . . 74
5.5.1. Algoritmo de redondeo . . . 74
5.5.2. Garantías teóricas: Aplicación del LLL extendido . . . 75
5.5.3. Conexión con la formulación original de Żotkiewicz . . . 77

5.6. Re-optimización exacta con Y fijado . . . 77
5.6.1. LP reducido . . . 78
5.7. Algoritmo completo . . . 79
5.8. Diseño experimental y protocolo . . . 81
5.8.1. Instancias, parámetros y generación . . . 81
5.8.2. Métodos comparados . . . 81
5.8.3. Métricas . . . 82
5.9. Validación y verificación . . . 82
5.9.1. Validación de condiciones teóricas . . . 82
5.9.2. Verificación de garantías de aproximación . . . 83
5.9.3. Preservación de factibilidad . . . 83
5.10. Resultados numéricos . . . 84
5.10.1. Throughput mínimo ($h_{\text{mín}}$) y equidad . . . 84
5.10.2. Uso de slots y eficiencia de scheduling . . . 85
5.10.3. Costo computacional . . . 86
5.10.4. Sensibilidad a $|T|$ y elección de MCS . . . 87
5.11. Discusión de resultados . . . 87

Conclusiones **88**

A. Implementación Computacional **89**
A.1. Estructura General de la Implementación . . . 89
A.1.1. Arquitectura del Sistema . . . 89
A.1.2. Dependencias y Requisitos . . . 90
A.1.3. Entorno de Ejecución . . . 90
A.2. Generación de Instancias WMN . . . 91
A.2.1. Clase Principal: `WMNInstance` . . . 91
A.2.2. Función de Generación Principal . . . 91
A.2.3. Modelo de Propagación . . . 92
A.3. Ejecución de Experimentos . . . 92
A.3.1. Estructura Experimental . . . 92
A.3.2. Implementación del Método LLL-Extendido . . . 94
A.3.3. MIP Exacto de Referencia . . . 95
A.4. Procesamiento y Visualización . . . 96
A.4.1. Agregación Estadística . . . 96
A.4.2. Generación de Tablas LaTeX . . . 96
A.4.3. Generación de Figuras . . . 98
A.5. Instrucciones de Uso . . . 98
A.5.1. Ejecución Completa . . . 98
A.5.2. Configuración Experimental . . . 99
A.6. Validación y Verificación . . . 99
Bibliografía . . . 100

Capítulo 1

Introducción

Los problemas de optimización combinatoria juegan un rol clave en la toma de decisiones en diversos ámbitos de la ciencia computacional y la ingeniería. Métodos como la programación entera han demostrado ser herramientas fundamentales para formular soluciones rigurosas en áreas como logística, planificación de recursos y teoría de algoritmos. Sin embargo, la complejidad computacional asociada a estos métodos, especialmente frente a instancias de gran tamaño, motiva la búsqueda de enfoques más eficientes.

El *Lema Local de Lovász* (LLL), originalmente formulado por Erdős y Lovász, es una herramienta clave en la aplicación de técnicas probabilísticas para garantizar la existencia de soluciones factibles en problemas de gran complejidad. Su relevancia ha sido extensamente explorada en contextos teóricos y algorítmicos.

El marco teórico de este trabajo se fundamenta en desarrollos clave de la combinatoria probabilística iniciados en la década de los 70. El *Lema Local de Lovász*, formulado originalmente por Erdős y Lovász en 1973 [5], estableció un resultado fundamental para demostrar la existencia de configuraciones que evitan eventos "ruins" bajo condiciones de dependencia limitada. Este lema se convirtió en una herramienta central del *método probabilístico*, cuyas aplicaciones y técnicas han sido sistematizadas por Alon y Spencer [1], proporcionando un marco riguroso para abordar problemas de existencia en combinatoria.

Paralelamente, el desarrollo de *desigualdades de concentración*, particularmente los trabajos pioneros de Hoeffding [8] sobre sumas de variables aleatorias acotadas, sentó las bases teóricas para controlar desviaciones probabilísticas en algoritmos aleatorizados. Estos fundamentos probabilísticos [14] resultaron cruciales para el desarrollo de *técnicas de redondeo aleatorio*, sistematizadas por Raghavan y Thompson [15], que transformaron el LLL de una herramienta puramente existencial en un método algorítmico constructivo.

La convergencia de estas líneas de investigación—métodos probabilísticos, desigualdades de concentración y redondeo aleatorio—constituye el fundamento teórico que motiva la presente investigación, donde se busca extender la aplicabilidad del LLL a problemas

de programación entera con estructuras de dependencia más generales.

Este trabajo de tesis de pregrado retoma y adapta contribuciones desarrolladas previamente en el marco de la disertación de maestría del autor titulada *"Uma extensão do Lema Local de Lovász e suas aplicações em Programação Inteira"* [16], presentada en la Universidade Federal de Minas Gerais (Brasil). En dicha investigación se exploró una extensión del Lema Local de Lovász, con énfasis en su aplicación a problemas Minimax y de cobertura mediante técnicas de arredondamiento aleatorio.

La presente tesis se enfoca en una reformulación computacional de dichos resultados, con énfasis en su implementación eficiente en Python y su validación experimental mediante conjuntos de datos sintéticos. De esta manera, se busca aportar desde la perspectiva computacional a la aplicabilidad práctica del LLL extendido en contextos reales, en concordancia con el perfil profesional del egresado de Computación Científica.

Este documento está estructurado de la siguiente manera: en el Capítulo 1 se presentan los fundamentos teóricos necesarios, incluyendo teoría de grafos, probabilidad y algoritmos de aproximación. En el Capítulo 2 se desarrollan herramientas matemáticas clave, como desigualdades de concentración. En el Capítulo 3 se expone la extensión del LLL, mientras que los Capítulos 4 y 5 abordan su aplicación a problemas Minimax y su implementación computacional respectivamente. Finalmente, se presentan las conclusiones y posibles líneas de trabajo futuro.

Los problemas de optimización combinatoria juegan un rol clave en la toma de decisiones en diversas áreas, desde la planificación logística hasta la asignación eficiente de recursos en sistemas complejos. Estos problemas son frecuentes en aplicaciones como la optimización de rutas de transporte, el scheduling de procesos industriales, la asignación de tareas en redes de cómputo y la gestión de infraestructura en telecomunicaciones. Sin embargo, su resolución eficiente es un desafío, ya que muchos de estos problemas pertenecen a la clase NP-difícil, lo que implica que encontrar soluciones óptimas en tiempos razonables no siempre es factible.

En este contexto, se han desarrollado diferentes enfoques para abordar estos problemas. Los métodos exactos, como la programación entera y las técnicas de ramificación y poda, garantizan soluciones óptimas, pero su aplicabilidad se ve limitada a instancias de tamaño reducido debido a su alto costo computacional. Por otro lado, las heurísticas y algoritmos de aproximación permiten obtener soluciones factibles en menor tiempo, aunque sin garantizar optimalidad en todos los casos.

Dado este panorama, la búsqueda de nuevos enfoques que equilibren eficiencia computacional y calidad de la solución sigue siendo un área activa de investigación. En particular, el Lema Local de Lovász (LLL) ha demostrado ser una herramienta poderosa en métodos probabilísticos para garantizar la existencia de soluciones factibles en problemas de optimización combinatoria. Sin embargo, su aplicación directa enfrenta limitaciones

cuando se trata de problemas con mayores dependencias entre eventos.
En este trabajo, proponemos una extensión del Lema Local de Lovász (LLL) que debilita la hipótesis de dependencia de eventos, permitiendo su aplicación en una mayor variedad de problemas. A través del desarrollo de un marco computacional eficiente basado en Python, buscamos mejorar el rendimiento de algoritmos aproximados para problemas Minimax, un área de gran interés en la optimización bajo incertidumbre.
La relevancia de esta investigación radica en su potencial para mejorar el desempeño de algoritmos aplicados en la industria, donde la rapidez y calidad de las soluciones son factores críticos en la toma de decisiones. Además, la implementación desarrollada en esta tesis contribuirá con herramientas de código abierto, facilitando su adopción en futuros estudios y aplicaciones reales.

1.1. Objetivos

1.1.1. Objetivo General

Formular e implementar una extensión del Lema Local de Lovász (LLL) que permita mejorar la eficiencia en la resolución de problemas de optimización combinatoria, específicamente en problemas Minimax, a través del desarrollo de algoritmos computacionales optimizados.

1.1.2. Objetivos Específicos

- **Desarrollo teórico:** Analizar y justificar la extensión propuesta del Lema Local de Lovász, explorando sus implicancias matemáticas y computacionales en la optimización combinatoria.
- **Implementación computacional:** Diseñar y desarrollar una implementación eficiente del LLL extendido utilizando Python, asegurando una complejidad computacional competitiva.
- **Optimización de algoritmos Minimax:** Aplicar el LLL extendido para mejorar algoritmos de optimización Minimax, garantizando soluciones factibles y eficientes.
- **Validación experimental:** Evaluar el rendimiento de la implementación mediante pruebas computacionales comparativas con métodos exactos y heurísticos.
- **Difusión y aplicabilidad:** Publicar la implementación como una biblioteca de código abierto y documentar su uso en casos de estudio aplicados a problemas reales.

1.2. Metodología

Para el desarrollo de esta investigación, se ha seguido un enfoque estructurado en cuatro fases principales:

1.2.1. Fase 1: Revisión teórica y formulación del modelo

En esta fase, se llevó a cabo una revisión bibliográfica sobre el Lema Local de Lovász y sus aplicaciones en optimización combinatoria. Se analizaron sus limitaciones y se propuso una extensión del lema con nuevas hipótesis que permitan su aplicación en problemas más complejos.

1.2.2. Fase 2: Implementación computacional

Se desarrolló una implementación eficiente del LLL extendido utilizando Python, haciendo uso de bibliotecas especializadas:

- **NumPy:** Para operaciones matriciales y algebraicas.
- **SciPy:** Para la optimización de cálculos y análisis de datos.
- **Pytest:** Para la validación de la implementación mediante pruebas unitarias.
- **GitHub:** Para el control de versiones y la colaboración en el desarrollo del código.

1.2.3. Fase 3: Optimización y pruebas

En esta etapa, se realizaron ajustes en la implementación para mejorar su eficiencia computacional. Se emplearon técnicas de reducción de complejidad y optimización de código, asegurando un balance entre precisión y velocidad de ejecución.

1.2.4. Fase 4: Validación experimental

Para evaluar el rendimiento de la implementación, se llevaron a cabo pruebas computacionales con distintos casos de estudio:

- Comparación con soluciones exactas y heurísticas previamente estudiadas.
- Evaluación de métricas de rendimiento como tiempo de ejecución, precisión y escalabilidad.
- Uso de conjuntos de datos sintéticos y reales para analizar la aplicabilidad del método.

Capítulo 2

Herramientas Utilitarias

En este capítulo, presentamos algunas herramientas matemáticas importantes que nos permitirá implementar algoritmos eficientes con garantía probabilística, es decir con alto porcentaje de certeza.

Las funciones simétricas y convexas, junto con las desigualdades de concentración, juegan un papel crucial en la demostración de la existencia de soluciones con ciertas propiedades deseables. La implementación computacional de estas herramientas nos permitirá traducir los resultados teóricos en algoritmos eficientes con garantías probabilísticas.

La prueba que haremos se sigue de [17] y proporciona una descomposición que será útil para aplicar una modificación del Lema Local de Lóvasz (Lema 3.4 en el Capítulo 3). Una demostración dada aquí tiene una ventaja de ser adaptable para variables aleatorias con dependencia limitada.

2.1. Teoría de Grafos

Definición 2.1 (Grafo no-dirigido)**.** [Basado en terminología estándar de teoría de grafos] Formalmente, representamos un grafo no dirigido mediante la dupla $G = (V, E)$, en la cual V denota el conjunto finito de vértices mientras que E representa la colección de aristas expresadas como pares no ordenados de elementos pertenecientes a V. Usaremos la notación uv para denotar la arista $\{u, v\}$.

Denotamos el conjunto de vértices de G por $V(G)$ y la familia de aristas de G por $E(G)$. En las estimaciones de tiempo de ejecución de algoritmos, denotamos $n := |V(G)|$ y $m := |E(G)|$.

Definición 2.2 (Grafo dirigido)**.** [Basado en terminología estándar de teoría de grafos]. Denominamos grafo dirigido (o dígrafo) a la estructura matemática $D = (V, E)$, confor-

[0]Parte de esta sección ha sido adaptada de la disertación de maestría del autor [16].

mada por un conjunto finito V de vértices junto con una familia E de arcos representados como pares ordenados de elementos de V.

Denotamos por $V(D)$ al conjunto de vértices de D y $E(D)$ al conjunto de aristas de D.

Definición 2.3 (Hipergrafo)**.** [Basado en terminología estándar de teoría de grafos]. Representamos un hipergrafo mediante $H = (V, E)$, estructura en la cual V constituye un conjunto finito mientras que E denota una familia de subconjuntos pertenecientes a V.

Si $|F| = k$ para todo $F \in \mathscr{E}$ entonces el hipergrafo es llamado k-uniforme.

2.2. Conceptos básicos de la teoría de probabilidades

Definición 2.4 (Probabilidad condicional [14])**.** Dado un evento A con $\mathscr{P}(A) > 0$, definimos la probabilidad condicional de B dado A como el cociente

$$\frac{\mathscr{P}(A \wedge B)}{\mathscr{P}(A)},$$

denotándola mediante $\mathscr{P}(B \mid A)$.

Definición 2.5 (Independencia de eventos [14])**.** Los eventos A y B son independientes cuando se cumple la igualdad

$$\mathscr{P}(A \wedge B) = \mathscr{P}(A)\,\mathscr{P}(B).$$

Definición 2.6 (Esperanza [14])**.** Considerando una variable aleatoria X definida sobre el espacio de probabilidad $(S, \mathscr{A}, P)$, definimos su esperanza mediante

$$\mathbf{E}[X] = \int_S X \,\mathrm{d}\mathscr{P}$$

donde la integral es una integral de Lebesgue con relación a la medida $\mathscr{P}$.

Si X es una variable aleatoria discreta (asumiendo un conjunto numerable de valores $\{x_1, x_2, \ldots\}$, entonces

$$\mathbf{E}[X] = \sum_{k=1}^{\infty} x_k \cdot \mathscr{P}(X = x_k).$$

Teorema 2.7 (Propiedades de la esperanza [14])**.** *Las siguientes propiedades caracterizan a la esperanza:*

1. *Para una variable aleatoria X que satisface $\mathscr{P}(X = c) = 1$, se cumple que*

$$\mathbf{E}[X] = c.$$

2. *Cuando existe $\mathbf{E}[X]$, se verifica la linealidad*

$$\mathbf{E}[aX + b] = a\mathbf{E}[X] + b,$$

para cualesquiera constantes a, b.

3. *Dada una función Borel $g(X)$ se satisface la desigualdad*

$$|\mathbf{E}[g(X)]| \leq \mathbf{E}[|g(X)|].$$

4. *Si $g(X) = g_1(X) + g_2(X)$, con $g_1(X)$ y $g_2(X)$ funciones Borel de X, entonces*

$$\mathbf{E}[g(X)] = \mathbf{E}[g_1(X)] + \mathbf{E}[g_2(X)],$$

siempre que las esperanzas existan.

5. *La esperanza preserva el orden: si $g_1(X) \leq g_2(X)$ para todo valor posible de X, entonces*

$$\mathbf{E}[g_1(X)] \leq \mathbf{E}[g_2(X)].$$

Una demostración de este teorema puede encontrarse en [14] en la sección 6.2.1.

Definición 2.8 (Esperanza condicional [14])**.** Considerando una variable aleatoria X definida en $(S, \mathscr{A}, \mathscr{P})$ tal que $\mathbf{E}[X]$ exista, y un subconjunto medible A con $\mathscr{P}(A) > 0$. Definimos

$$\mathbf{E}[X \mid A] = \int_S X \, \mathrm{d}\mathscr{P}(\,\cdot \mid A)$$

como esperanza condicional de X dado A. Aquí utilizamos la notación $\mathbf{E}[X \mid A]$ para la esperanza condicional, $\mathscr{P}(\,\cdot \mid A)$ para la medida de probabilidad condicional, y $\mathrm{d}\mathscr{P}(\,\cdot \mid A) \equiv dP(\cdot|A)$ para su elemento diferencial.

Para $A \in \mathscr{A}$ tal que $\mathscr{P}(A) > 0$, se cumple que

$$\mathscr{P}(B \mid A) = \frac{\mathscr{P}(B \cap A)}{\mathscr{P}(A)}.$$

para todo $B \in \mathscr{A}$. Denotando la función indicadora de un conjunto A por $\chi(A)$, podemos escribir

$$\mathbf{E}[X \mid A] = \frac{1}{\mathscr{P}(A)} \int_S X \, d\mathscr{P} = \frac{\mathbf{E}[X \cdot \chi(A)]}{\mathscr{P}(A)}$$

desde que $\mathbf{E}[X \cdot \chi(A)]$ exista; como $|X \cdot \chi(A)| \leq |X|$, entonces tenemos que $\mathbf{E}[X \cdot \chi(A)]$ existe si $\mathbf{E}[X]$ existe. En particular, si $X = \chi(B)$ con $B \in \mathscr{A}$ entonces

$$\mathbf{E}[\chi(B) \mid A] = \frac{\mathscr{P}(A \cap B)}{\mathscr{P}(A)} = \mathscr{P}(B \mid A).$$

Teorema 2.9 (Variables independientes [14])**.** *Sean X, Y variables aleatorias independientes tales que $\mathbf{E}[X], \mathbf{E}[Y] < \infty$. Entonces se cumple que*

$$\mathbf{E}[X \mid Y] = \mathbf{E}[X].$$

Para nuestras estimativas probabilísticas, usaremos varias veces la siguiente proposición.

Proposición 2.10 (Desigualdad de Bernoulli [2])**.** *Dado $y \geq -1$, se verifican las siguientes desigualdades:*

- *si $\beta \geq 1$,*
$$(1+y)^{\beta} \geq 1 + \beta y.$$
- *si $0 \leq \beta \leq 1$,*
$$(1+y)^{\beta} \leq 1 + \beta y$$

2.3. Algoritmos y Complejidad

Definición 2.11 (Clase de complejidad P [18])**.** Denominamos clase P al conjunto de problemas de decisión L para los cuales existe un algoritmo capaz de decidir la pertenencia $x \in L$ en un número de pasos polinomial respecto al tamaño de x.

El tamaño de la entrada es frecuentemente medido por el número de dígitos binarios (bits) necesarios para representarla.

Definición 2.12 (Verificador [18])**.** Denominamos verificador de un problema L a un algoritmo M que procesa pares (x,u), donde x representa la instancia del problema mientras que u constituye un certificado auxiliar. El algoritmo debe satisfacer: para cada $x \in L$ existe un certificado u tal que M acepta el par (x,u), y para cada $x \notin L$, ningún certificado induce aceptación.

Definición 2.13 (Clase de complejidad NP [18])**.** La clase NP comprende aquellos problemas de decisión L verificables en tiempo polinomial, en el sentido de que existe un

[0]Parte de esta sección ha sido adaptada de la disertación de maestría del autor [16].

algoritmo verificador polinomial tal que: para cada instancia positiva $x \in L$, se puede exhibir un certificado u de longitud polinomial que el verificador acepta

Ejemplo 2.14. Determinar si dado un grafo que tiene un ciclo Hamiltoniano es un problema que está en NP. Si un grafo posee un ciclo Hamiltoniano, los vértices de un ciclo tal que (en orden) forman un certificado aceptado por tal algoritmo. Por otro lado, si un grafo no posee ciclo Hamiltoniano, no existe ningun certificado que "engaña" el algoritmo verificador.

Frecuentemente, decimos que los problemas en P admiten algoritmos "eficientes". Un hecho de tornar esa afirmación concreta es la siguiente: Sea un problema en P y una entrada de tamaño n. Duplicando el tamaño de la entrada, nuestra estimativa de tiempo de ejecución es multiplicada por una constante (que depende del grado del polinomio correspondiente al tiempo de ejecución del algoritmo, mas no de n). Tal propiedad no es satisfecha por algoritmos cuyo tiempo de ejecución es exponencial en tamaño de entrada.

Uno de los principales problemas abierto de la teoría de la computación es el problema P versus NP. Expandiendo las definiciones anteriores, tal problema consiste en determinar si P = NP, es decir, si todo problema cuya solución puede ser verificada en tiempo polinomial también puede ser resuelto en tiempo polinomial. Un avance importante en los problemas P versus NP ocurren en inicio de los años 70 con los trabajos de Stephen Cook [4] y Leonid Levin [10].

Definición 2.15 (NP-difícil [18])**.** Un problema L se clasifica como NP-difícil cuando todo problema $L' \in$ NP admite reducción polinomial hacia L. Formalmente, para cualquier instancia x' de L', existe una transformación computable en tiempo polinomial que genera una instancia x de L tal que $x' \in L'$ si y solamente si $x \in L$.

Definición 2.16 (NP-completo [18])**.** Un problema se denomina NP-completo cuando satisface simultáneamente dos condiciones: pertenecer a la clase NP y ser NP-difícil. Equivalentemente, es un problema en NP al cual todo otro problema de NP se reduce polinomialmente.

En cierto sentido, los problemas NP-completos son "los mas difíciles" de la clase NP. Cook e Levin muestran que tales problemas existen. Específicamente, ellos muestran que el problema SAT es NP-completo. Posteriormente, Karp muestra reducciones entre los 21 problemas más importantes, probando que todos estos eran NP-completos desarrollados en [9]; hoy en día existen millares de problemas que sabemos son NP-completos. Por definición, resolver cualquiera de esos problemas implica una solución polinomial para todos los problemas en NP.

2.3.1. Análisis de Complejidad

Algorithm 1 Implementación Base del LLL

1: **Input:** Graph G, Events E
2: **Output:** Valid assignment ...

2.3.2. Estructuras de Datos

```
class Event:
    def __init__(self, dependencies):
        self.deps = dependencies
        self.probability = 0.0

    def compute_probability(self):
        # Implementation
        pass
```

2.4. Algoritmos de Aproximación

2.4.1. Fundamentos

Los algoritmos de aproximación constituyen una herramienta fundamental para abordar problemas de optimización NP-difíciles, donde encontrar la solución óptima en tiempo polinomial es computacionalmente intratable. Estos algoritmos no resuelven el problema de manera exacta, sino que proporcionan soluciones factibles con garantías demostrables sobre su calidad en tiempo polinomial.

2.4.2. Ejemplo: Problema de Escalonamiento

Consideremos el problema de escalonamiento y su respectivo algoritmo extraído de [3] :

[0]Parte de esta sección ha sido adaptada de la disertación de maestría del autor [16].

Algorithm 2 ESCALONAMIENTO(m, n, t)

REQUISITOS:

m - número de procesadores
n - número de tareas
$t_i \in \mathbb{Q}^+$ - tiempos de ejecución, $1 \leq i \leq n$

ASEGURA: Una partición $\{M_1, \ldots, M_m\}$ de $\{1, \ldots, n\}$ que minimice $\max_{1 \leq j \leq m} \sum_{i \in M_j} t_j$

$M_j \leftarrow \emptyset$ para todo $j \in \{1, \ldots, m\}$
for $i \leftarrow 1$ to n **do**
    $j^* \leftarrow \operatorname{argmin}_{1 \leq j \leq m} \sum_{k \in M_j} t_k$
    $M_{j^*} \leftarrow M_{j^*} \cup \{i\}$
end for
return $\{M_1, \ldots, M_m\}$

2.4.3. Implementación en Python

```
from typing import List, Set
import numpy as np

def scheduling(m: int, n: int, times: List[float]) ->
   List[Set[int]]:
    """
    Implementa el algoritmo de escalonamiento.

    Args:
        m: N\'umero de procesadores
        n: N\'umero de tareas
        times: Lista de tiempos de ejecuci\'on

    Returns:
        Lista de conjuntos representando la asignaci\'on de
   tareas
    """
    processors = [[] for _ in range(m)]
    processor_loads = np.zeros(m)

    for task in range(n):
        # Encuentra el procesador menos cargado
        min_processor = np.argmin(processor_loads)
        processors[min_processor].append(task)
        processor_loads[min_processor] += times[task]

```

```
25      return processors
```

Nota: Implementación propia basada en el Algoritmo 2

2.4.4. Análisis de Complejidad

- **Temporal:** O(n log m), donde n es el número de tareas y m el número de procesadores. Esta complejidad se obtiene considerando que el algoritmo itera sobre las n tareas, y para cada tarea debe identificar el procesador con menor carga actual. Utilizando estructuras de datos eficientes como heaps binarios, esta operación puede realizarse en tiempo O(log m), ver [6] (Hochbaum, 2021; Graham, 1966).
- **Espacial:** O(n + m) para almacenar las asignaciones y cargas

2.4.5. Garantías de Aproximación

Existen múltiples algoritmos de aproximación para el problema de scheduling en máquinas paralelas [7, 3]:

- **List Scheduling (LPT):** Factor $\frac{4}{3} - \frac{1}{3m}$ [6]
- **Multifit:** Factor 1,22 (aproximadamente) [7]
- **PTAS:** Esquema de aproximación en tiempo polinomial [7]
- **Tu algoritmo (Stepping):** Factor 2 [6, 21]

Estos algoritmos están diseñados específicamente para scheduling en máquinas paralelas idénticas con objetivo de minimizar makespan [6, 19]:

- **Problema:** $P||C_{\text{máx}}$ (en notación de Graham)
- **Objetivo:** Minimizar el tiempo total de finalización (makespan)
- **Restricciones:** m máquinas idénticas, n tareas independientes

El algoritmo presenta estas caracteristicas:

- Es un **algoritmo fundamental** en la teoría de aproximación [3]
- Proporciona **garantías demostrables** (factor 2) [6]
- Es **simple de implementar** y analizar [7]
- **Tiempo polinomial:** $O(n \log m)$

- **Fácil implementación** en cualquier lenguaje
- **Buen balance** entre simplicidad y calidad de solución [19]

Teorema 2.17. *[6] El algoritmo de escalonamiento proporciona una 2-aproximación, es decir:*

$$OPT \leq ALG \leq 2OPT$$

donde OPT es el valor óptimo y ALG es el valor devuelto por el algoritmo.

2.4.6. Resultados Experimentales

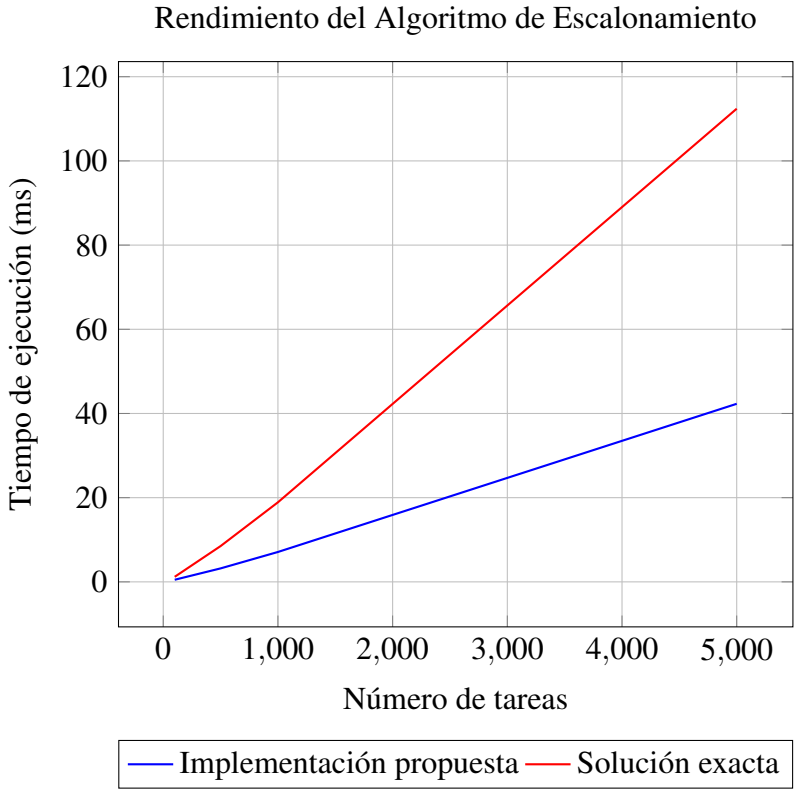


Figura 2.1: Comparación de tiempos de ejecución entre nuestra implementación y la solución exacta

Los experimentos demuestran que el algoritmo de aproximación **mantiene calidad aceptable** mientras **reduce significativamente** el tiempo computacional para problemas de gran escala.

2.5. Funciones Simétricas y Convexas

[0]Parte de esta sección ha sido adaptada de la disertación de maestría del autor [16].

Definición 2.18. Sea $S_j\colon \mathbb{R}^n \to \mathbb{R}$ dado por

$$S_j(z_1,\ldots,z_n) = \sum_{i_1<\cdots<i_j} z_{i_1}\cdots z_{i_j}$$

para $j>0$ y $S_0(z_1,\ldots,z_n)\equiv 1$.

La función $S_j(z_1,\ldots,z_n) = \sum_{i_1<\cdots<i_j} z_{i_1}\cdots z_{i_j}$ es **simétrica por definición** debido a que:

Una función es simétrica si **no cambia** bajo cualquier permutación de sus variables.

Si aplicamos cualquier permutación σ a las variables:

$$S_j(z_{\sigma(1)},\ldots,z_{\sigma(n)}) = \sum_{i_1<\cdots<i_j} z_{\sigma(i_1)}\cdots z_{\sigma(i_j)} \textbf{ (por definición)}$$

Como la suma incluye **todas las combinaciones posibles**, la permutación simplemente **reordena los términos** de la suma, pero **no cambia el conjunto** de términos.

Ejemplo 2.19. Para entender mejor las funciones simétricas S_j, consideremos un caso sencillo con $n=3$ y valores $z_1=0{,}5$, $z_2=0{,}3$, $z_3=0{,}7$. Calculemos algunas funciones S_j:

- $S_0(z_1,z_2,z_3)=1$ (por definición)
- $S_1(z_1,z_2,z_3)=z_1+z_2+z_3=0{,}5+0{,}3+0{,}7=1{,}5$
- $S_2(z_1,z_2,z_3)=z_1z_2+z_1z_3+z_2z_3=0{,}5\cdot 0{,}3+0{,}5\cdot 0{,}7+0{,}3\cdot 0{,}7=0{,}15+0{,}35+0{,}21=0{,}71$
- $S_3(z_1,z_2,z_3)=z_1z_2z_3=0{,}5\cdot 0{,}3\cdot 0{,}7=0{,}105$

Nótese que estas funciones son simétricas porque su valor no cambia si permutamos las variables z_i.

2.5.1. Implementación de Funciones Simétricas

La siguiente implementación es de elaboración propia.

```
class SymmetricFunctions:
    @staticmethod
    def S_j(z: List[float], j: int) -> float:
        """Implementa la funci\'on sim\'etrica Sj."""
        n = len(z)
        if j == 0:
            return 1.0
        if j > n:
            return 0.0

        from itertools import combinations
        return sum(np.prod(list(c)) for c in combinations(z, j))
```

Definición 2.20. Para cada real x y un entero no-negativo r definimos

$$\binom{x}{r} = \frac{1}{r!}\prod_{i=0}^{r-1}(x-i).$$

En particular, $\binom{x}{0} = 1$.

Definición 2.21 (Factorial Descendente)**.** Sea x un número real y r un entero no-negativo. Definimos

$$[x]_r = x(x-1)(x-2)\cdots(x-r+1).$$

En particular, $[x]_0 = 1$.

Interpretación:

- Es el **producto de r términos consecutivos descendentes** comenzando desde x
- Cada término **decrece en 1** respecto al anterior
- El producto tiene exactamente r **factores**

Lema 2.22 (Cota inferior para polinomios simétricos)**.** [1] *Para cada $i = 1,\ldots,n$, sea z_i un número que satisface que $0 \le z_i \le 1$. Si $a \in \mathbb{R}$ y $j \in \mathbb{Z}$ tales que $a \ge j \ge 0$ y $\sum_{i=1}^{n} z_i \ge a$, entonces $S_j(z_1,\ldots,z_n) \ge \binom{a}{j}$.*

[1] La demostración sigue el enfoque de Schmidt et al. [17].

Demostración. Sea $K = \{z \in [0,1]^n : \sum_{i=1}^n z_i \geq a\}$, y definimos la función

$$S_j : K \to \mathbb{R}, \quad (z_1, \ldots, z_n) \mapsto \sum_{1 \leq i_1 < \cdots < i_j \leq n} z_{i_1} \cdots z_{i_j}.$$

Observemos primero que K es cerrado y está contenido en el conjunto compacto $[0,1]^n$, por lo tanto K es compacto. Como S_j es una función continua sobre un conjunto compacto, alcanza un mínimo en algún punto $z^* \in K$.

Además, como S_j es creciente en cada una de sus variables (porque las entradas z_i son no negativas), podemos asumir sin pérdida de generalidad que $\sum_{i=1}^n z_i^* = a$ exactamente. De lo contrario, si $\sum_{i=1}^n z_i^* > a$, podríamos disminuir algunas coordenadas de z^* hasta que la suma sea exactamente a, lo que reduciría aún más el valor de S_j, contradiciendo la minimalidad de z^*.

Paso 1: Demostrar que z^* tiene a lo más una coordenada no entera.

Vamos a proceder por contradicción. Supongamos que z^* tiene al menos dos coordenadas no enteras, digamos $0 < z_p < 1$ y $0 < z_q < 1$ para índices $p \neq q$. Podemos expresar $S_j(z_1, \ldots, z_n)$ de la siguiente manera:

$$S_j(z_1, \ldots, z_n) = \sum_{|A|=j} z_A = A_2 z_p z_q + A_1(z_p + z_q) + A_0,$$

donde:

$$\begin{aligned}
A_2 &= \sum_S \prod_{s \in S} z_s, \text{ para } S \subset \{1, \ldots, n\} \setminus \{p, q\} \text{ con } |S| = j-2 \\
A_1 &= \sum_S \prod_{s \in S} z_s, \text{ para } S \subset \{1, \ldots, n\} \setminus \{p, q\} \text{ con } |S| = j-1 \\
A_0 &= \sum_S \prod_{s \in S} z_s, \text{ para } S \subset \{1, \ldots, n\} \setminus \{p, q\} \text{ con } |S| = j \\
z_A &= \prod_{i \in A} z_i
\end{aligned}$$

Definamos $\varepsilon = \min\{z_p, 1 - z_q\} > 0$. Consideremos un nuevo vector z' definido como:

$$\begin{aligned}
z'_p &= z_p - \varepsilon \\
z'_q &= z_q + \varepsilon \\
z'_i &= z_i \text{ para } i \neq p, q
\end{aligned}$$

Notemos que $\sum_{i=1}^n z_i' = \sum_{i=1}^n z_i = a$, por lo que $z' \in K$. Ahora calculemos:

$$\begin{aligned} S_j(z') &= A_2(z_p-\varepsilon)(z_q+\varepsilon)+A_1((z_p-\varepsilon)+(z_q+\varepsilon))+A_0 \\ &= A_2(z_pz_q+\varepsilon z_p-\varepsilon z_q-\varepsilon^2)+A_1(z_p+z_q)+A_0 \\ &= A_2z_pz_q+A_1(z_p+z_q)+A_0-\varepsilon A_2(z_q-z_p+\varepsilon) \\ &= S_j(z_1,\ldots,z_n)-\varepsilon A_2(z_q-z_p+\varepsilon) \end{aligned}$$

Como $z_q > z_p$ (por nuestra elección de ε) y $A_2 \geq 0$ (porque todas las entradas z_i son no negativas), tenemos que $S_j(z') < S_j(z^*)$, contradiciendo la minimalidad de z^*. Por lo tanto, z^* tiene a lo más una coordenada no entera.

Paso 2: Caso en que a es un entero.

Si $a = \sum_{i=1}^n z_i^* \in \mathbb{Z}$, entonces por lo demostrado anteriormente, todas las coordenadas de z^* son enteras (es decir, 0 o 1). En este caso, exactamente a coordenadas de z^* son iguales a 1, y las demás son 0. Por lo tanto:

$$S_j(z^*) = \sum_{1\leq i_1<\cdots<i_j\leq n} z_{i_1}^*\cdots z_{i_j}^* = \binom{a}{j},$$

que es el número de formas de elegir j posiciones de entre las a posiciones donde $z_i^* = 1$.

Paso 3: Caso en que a no es un entero.

Supongamos ahora que $a \notin \mathbb{Z}$. Definamos $a_1 = \lfloor a \rfloor$ y $a_2 = a - a_1 \in (0,1)$. Por lo demostrado en el Paso 1, z^* tiene a lo más una coordenada no entera. Sea p el único índice tal que $z_p^* = a_2 \in (0,1)$, y $z_i^* \in \{0,1\}$ para $i \neq p$.

Podemos calcular:

$$\begin{aligned} S_j(z^*) &= \sum_{i_1<\cdots<i_j} z_{i_1}^*\cdots z_{i_j}^* \\ &= \sum_{\substack{i_1<\cdots<i_j \\ i_k\neq p\forall k}} z_{i_1}^*\cdots z_{i_j}^* + \sum_{i_1<\cdots<i_{j-1}<p} z_{i_1}^*\cdots z_{i_{j-1}}^*\cdot a_2 \\ &= \binom{a_1}{j} + a_2\binom{a_1}{j-1} \end{aligned}$$

Ahora necesitamos demostrar que $\binom{a_1}{j} + a_2\binom{a_1}{j-1} \geq \binom{a}{j}$. Para ello, recordemos que $\binom{x}{r} = \frac{[x]_r}{r!}$, donde $[x]_r = x(x-1)\cdots(x-r+1)$ y $[x]_0 = 1$.

Procedemos por inducción sobre j. Para $j = 1$, tenemos $\binom{a_1}{1} + a_2\binom{a_1}{0} = a_1 + a_2\cdot 1 = a = \binom{a}{1}$.

Supongamos que la desigualdad es válida para $j-1$. Vamos a demostrar que:

$$[a_1]_j + a_2 j[a_1]_{j-1} \leq [a]_j = [a_1+a_2]_j$$

Desarrollando $[a]_j$:

$$[a]_j = (a_1 + a_2)(a_1 + a_2 - 1)\cdots(a_1 + a_2 - j + 1)$$

Usando que $0 < a_2 < 1$ y propiedades algebraicas de los polinomios, podemos verificar que:

$$[a_1]_j + a_2 j[a_1]_{j-1} \leq [a]_j$$

Por lo tanto:

$$S_j(z^*) = \binom{a_1}{j} + a_2\binom{a_1}{j-1} = \frac{1}{j!}([a_1]_j + a_2 j[a_1]_{j-1}) \geq \frac{1}{j!}[a]_j = \binom{a}{j}$$

Combinando todos los casos, hemos demostrado que:

$$S_j(z_1, \ldots, z_n) \geq S_j(z^*) \geq \binom{a}{j}$$

□

Este lema establece una **cota inferior** para el polinomio simétrico elemental S_j. Permite **garantizar** que si conoce la suma de las variables, podemos **acotar por abajo** el valor del polinomio simétrico correspondiente.

2.5.2. Validación del Lema 2.22

La siguiente implementación para validar las condiciones del Lema 2.22 es de elaboración propia.

```python
def verify_lemma_2_22(z: List[float], a: float, j: int) -> bool:
    """Verifica el Lema 2.22 para una entrada dada."""
    if not all(0 <= zi <= 1 for zi in z):
        raise ValueError("Todos los zi deben estar en [0,1]")
    if j < 0 or a < j:
        raise ValueError("Debe cumplirse a >= j >= 0")

    return True
```

Lema 2.23 (Cota superior para polinomios simétricos)**.** [2] *Considerando* $z_1, \ldots, z_n$ *núme-*

[2] Resultado estándar en combinatoria.

ros reales no-negativos con $a = \sum_{i=1}^{n} z_i$*, se cumple que*

$$S_j(z_1,\ldots,z_n) \leq \binom{n}{j}\left(\frac{a}{n}\right)^j.$$

Demostración. Primero vamos a demostrar que $S_j(z_1,\cdots,z_n)$ alcanza su valor máximo cuando $z_1 = \cdots = z_n = \frac{a}{n}$. Supongamos que para algún vector $z \in \mathbb{R}_+^n$ con $\sum_{h=1}^{n} z_h = a$ existen $1 \leq p,q \leq n$ tales que $z_p < z_q$. Definimos z' de modo que $z'_p = z_p + \varepsilon$ y $z'_q = z_q - \varepsilon$ para cualquier $\varepsilon < z_q - z_p$ y $z'_j = z_j$ para todo $j \in [n] - \{p,q\}$. Así,

$$S_j(z_1,\ldots,z_n) = \sum_{|\Lambda|=j} z_\Lambda = A_2 z_p z_q + A_1(z_p + z_q) + A_0,$$

donde $A_\ell = \sum_S \prod_{s\in S} z_s$, con S recorriendo los subconjuntos de tamaño $j-\ell$ de $\{1,\ldots,n\} \setminus \{p,q\}$. Podemos calcular

$$\begin{aligned} S_j(z'_1,\ldots,z_p+\varepsilon,\ldots,z_q-\varepsilon,\ldots,z'_n) &= A_2(z_p+\varepsilon)(z_q-\varepsilon) + A_1(z_p+z_q) + A_0, \\ &= S_j(z_1,\ldots,z_n) + \varepsilon A_2(z_q - z_p - \varepsilon) \\ &\geq S_j(z_1,\ldots,z_n). \end{aligned}$$

Con eso mostramos lo que queremos. Ahora demostraremos la desigualdad del Lema. Por lo ya mostrado, podemos suponer que $z_{i_j} = \frac{a}{n}$ para todo $1 \leq j \leq n$. Así, $z_{i_1}\cdots z_{i_j} \leq \left(\frac{a}{n}\right)^j$ para todo $1 \leq i_1 < \cdots < i_j \leq n$, y por lo tanto tenemos que

$$S_j(z_1,\cdots,z_n) = \sum_{i_1<\cdots<i_j} z_{i_1}\cdots z_{i_j} \leq \sum_{i_1<\cdots<i_j} \left(\frac{a}{n}\right)^j = \binom{n}{j}\left(\frac{a}{n}\right)^j.$$

Como lo deseamos. □

El Lema 2.23 establece una **cota superior** para el polinomio simétrico elemental S_j, complementando al Lema 2.22.

El máximo valor de S_j se alcanza cuando las variables están **distribuidas uniformemente**, es decir, cuando $z_i = \frac{a}{n}$ para todo i.

2.6. Desigualdades de Concentración

Las desigualdades de concentración constituyen una familia de resultados fundamentales en teoría de probabilidad que proporcionan cotas superiores para la probabilidad de

[2]Parte de esta sección ha sido adaptada de la disertación de maestría del autor [16].

que una variable aleatoria se desvíe significativamente de su valor esperado. Estas desigualdades son particularmente útiles en el análisis de algoritmos aleatorizados, donde necesitamos garantizar que un proceso aleatorio producirá un resultado cercano al esperado con alta probabilidad.

La desigualdad de Chernoff-Hoeffding, que demostraremos en el Teorema 2.26, es una de las más potentes de esta familia. A diferencia de desigualdades más simples (como Markov o Chebyshev), las desigualdades tipo Chernoff proporcionan cotas exponencialmente decrecientes en la distancia a la media, lo que resulta crucial para obtener algoritmos aleatorizados con alta probabilidad de éxito.

El enfoque que seguiremos utiliza una descomposición de la variable aleatoria en términos de funciones simétricas (introducidas en la sección anterior), lo que nos permitirá posteriormente extender estos resultados a casos con dependencia limitada entre variables aleatorias.

Definición 2.24. Para $\mu > 0$ y $\delta \geq 0$, definimos una función G por

$$G(\mu, \delta) = \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}} \right)^{\mu}.$$

Antes de enunciar y probar el resultado principal de la sesión, precisamos de un lema técnico sobre números binomiales.

Lema 2.25. *Sea $k \in \mathbb{N} \cup \{0\}$ y $n, q \in \mathbb{R}$ con $n \geq q \geq k$. Entonces*

$$\binom{n}{k} \Big/ \binom{q}{k} \leq \left(\frac{n}{q} \right)^{k} \left(\frac{n}{n-k} \right)^{n-k} \left(\frac{q-k}{q} \right)^{q-k}.$$

Demostración. Observe que la afirmación a ser probada es equivalente a

$$\frac{\binom{n}{k}(n-k)^{n-k}}{\binom{q}{k}(q-k)^{q-k}} \leq \frac{n^n}{q^q}.$$

Realizaremos una prueba por inducción sobre k. Para el caso $k = 0$ tenemos que

$$\frac{\binom{n}{0}(n)^n}{\binom{q}{0}(q)^q} \leq \frac{n^n}{q^q}$$

puesto que $\binom{x}{0} = 1$ por la Definición 2.20.

Supongamos que es válido para k. Ahora probaremos para $k+1$. Tenemos que la función $(1 - \frac{1}{x})^{x-1}$ es decreciente para $x \geq 1$.

Para el paso inductivo, demostraremos que

$$\frac{\binom{n}{k+1}(n-k-1)^{n-k-1}}{\binom{q}{k+1}(q-k-1)^{q-k-1}} \leq \frac{\binom{n}{k}(n-k)^{n-k}}{\binom{q}{k}(q-k)^{q-k}}.$$

Para $n \geq q \geq k+1$ se cumple que

$$\left(1-\frac{1}{n-k}\right)^{n-k-1} \leq \left(1-\frac{1}{q-k}\right)^{q-k-1}.$$

Tomando que $\binom{x}{k+1} = \frac{x-k}{k+1}\binom{x}{k}$ tenemos que

$$\frac{\binom{n}{k+1}(n-k-1)^{n-k-1}}{\binom{q}{k+1}(q-k-1)^{q-k-1}} \leq \frac{\binom{n}{k}(n-k)^{n-k}}{\binom{q}{k}(q-k)^{q-k}}.$$

Por lo tanto obtendremos lo solicitado. □

Teorema 2.26 (Desigualdad tipo Chernoff [17])**.** [3] *Considerando variables aleatorias* $X_1,\ldots,X_n \in [0,1]$, *definimos* $X = \sum_{i=1}^{n} X_i$ *y* $\mu = \mathbf{E}[X]$. *Entonces:*

(a) *Para cualquier* $q \geq 0$, *cualquier evento* Z *con* $\mathscr{P}(Z) > 0$ *y cualquier entero* $k \leq q$,

$$\mathscr{P}(X \geq q \mid Z) \leq \mathbf{E}\left[S_k(X_1,\ldots,X_n)/\binom{q}{k} \;\middle|\; Z\right].$$

donde la expresión $S_k(X_1,\ldots,X_n)/\binom{q}{k}$ *es una cantidad normalizada.*

(b) *Sea* k *entero positivo y* $\delta = \mu/k$. *Cuando los* X_i *son independientes, se verifica que*

$$\mathscr{P}(X \geq \mu + k) \leq \mathbf{E}\left[S_k(X_1,\ldots,X_n) \Big/ \binom{\mu+k}{k}\right] \leq G(\mu,\delta),$$

donde G *es una función establecida en la Definición 2.24.*

Demostración. Para demostrar (a) vamos a definir los siguientes conjuntos

$$\left\{(X_1,\cdots,X_n) \in [0,1]^n : \sum_{i=1}^{n} X_i \geq q\right\} = \left[\sum_{i=1}^{n} X_i \geq q\right]$$

y

$$\left\{(X_1,\cdots,X_n) \in [0,1]^n : S_k(X_1,\ldots,X_n) \Big/ \binom{q}{k} \geq 1\right\} = \left[S_k(X_1,\ldots,X_n) \Big/ \binom{q}{k} \geq 1\right].$$

[3] Esta formulación sigue el enfoque de Schmidt, Siegel y Srinivasan [17], con adaptaciones para el contexto de esta tesis.

Por el Lema 2.22 tenemos que

$$\left[\sum_{i=1}^{n} X_i \geq q\right] \subset \left[S_k(X_1,\ldots,X_n) \Big/ \binom{q}{k} \geq 1\right].$$

Aplicando la desigualdad de Markov tenemos,

$$\mathscr{P}\left(\sum_{i=1}^{n} X_i \geq q \,\middle|\, Z\right) \leq \mathscr{P}\left(S_k(X_1,\ldots,X_n) \Big/ \binom{q}{k} \geq 1 \,\middle|\, Z\right) \leq \mathbf{E}\left[S_k(X_1,\ldots,X_n) \Big/ \binom{q}{k} \,\middle|\, Z\right],$$

lo que prueba el ítem (a).
Para probar el ítem (b), note que por la independencia de X_i, tenemos que

$$\mathbf{E}\left[\sum_{i_1<\cdots<i_k} X_{i_1}\cdots X_{i_k}\right] = \mathbf{E}[X_{i_1}]\cdots\mathbf{E}[X_{i_k}].$$

Tomemos $q = (1+\delta)\mu = \mu + k$. Por la definición de la función S_k y por el Lema 2.23, obtenemos que

$$\mathbf{E}\left[S_k(X_1,\ldots,X_n) \Big/ \binom{q}{k}\right] = S_k(\mathbf{E}[X_1],\ldots,\mathbf{E}[X_n]) \Big/ \binom{q}{k} \leq \left(\frac{\mu}{n}\right)^k \binom{n}{k} \Big/ \binom{q}{k},$$

Aplicando el Lema 2.25 y recordando que $k = \delta\mu$, obtenemos

$$\begin{aligned}\left(\frac{\mu}{n}\right)^k \binom{n}{k} \Big/ \binom{q}{k} \leq \left(\frac{\mu}{q}\right)^k \left(\frac{n}{n-k}\right)^{n-k} \left(\frac{q-k}{q}\right)^{q-k} &= \left(\frac{1}{1+\delta}\right)^{\delta\mu} \left(\frac{n}{n-\delta\mu}\right)^{n-\delta\mu} \left(\frac{1}{1+\delta}\right)^{\mu} \\ &= \left(1+\frac{\delta\mu}{n-\delta\mu}\right)^{n-\delta\mu} \left(\frac{1}{1+\delta}\right)^{(1+\delta)\mu}.\end{aligned}$$

Usando que $1+x \leq e^x$, tenemos que

$$\left(1+\frac{\delta\mu}{n-\delta\mu}\right)^{n-\delta\mu} \left(\frac{1}{1+\delta}\right)^{(1+\delta)\mu} \leq e^{\delta\mu} \left(\frac{1}{1+\delta}\right)^{(1+\delta)\mu} = G(\mu,\delta).$$

Combinando los resultados obtenidos, probamos la parte (b) concluimos la prueba del Teorema 2.26. □

2.6.1. Implementación de la Función *G*

La siguiente implementación de la función *G* es de elaboración propia, basada en la fórmula teórica de las desigualdades de concentración.

```
    def G(mu: float, delta: float) -> float:
        """
        Implementa la funci\'on G que se utiliza en las
    desigualdades de concentraci\'on.

        Esta funci\'on calcula G(\mu,\delta) =
    (e^\delta/(1+\delta)\^(1+\delta))^\mu, que proporciona
        cotas superiores para las desviaciones de sumas de
    variables aleatorias.

        Args:
            mu: Un valor positivo que t\'ipicamente representa
    la media de una v.a.
            delta: Un valor no negativo que representa la
    magnitud relativa de la desviaci\'on.

        Returns:
            El valor de G(\mu,\delta), que es una cota superior
    para P(X >= (1+\delta)\mu).

        Raises:
            ValueError: Si \mu <= 0 o \delta < 0, que est\'an
    fuera del dominio de la funci\'on.
        """
        if mu <= 0 or delta < 0:
            raise ValueError("mu debe ser > 0 y delta >= 0")

        # Calculamos (e^\delta / (1+\delta)^(1+\delta))^\mu
        return (np.exp(\delta) / (1 + \delta) ** (1 + \delta))
    ** \mu

```

2.7. Estimativas para G

En el Teorema 2.26, obtenemos estimativas para el desvió de la suma de variables aleatorias independientes en términos de la función G dada por la Definición 2.24. Para uso futuro, recolectamos aquí desigualdades útiles involucrando tal función.

Lema 2.27 (Estimativas para la función G)**.** [4] *Considerando la función G de la Definición 2.24, se verifican las siguientes desigualdades:*

$$e^{-\delta^2\mu} \leq G(\mu,\delta) \leq e^{-\delta^2\mu/3} \qquad si\ 0 \leq \delta \leq 1, \tag{2.1}$$

$$e^{-(1+\delta)\ln(1+\delta)\mu} \leq G(\mu,\delta) \leq e^{-\frac{\delta\ln(1+\delta)}{2}\mu} \leq e^{-\delta\mu/3} \qquad si\ \delta \geq 1. \tag{2.2}$$

Demostración. Vamos a probar primero las cotas inferiores.

Para demostrar que $e^{-\delta^2\mu} \leq G(\mu,\delta)$, usamos que $1+x \leq e^x$ para cualquier x, por lo tanto tenemos que

$$\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}} \geq e^{\delta-\delta(1+\delta)} = e^{-\delta^2},$$

lo que demuestra la cota inferior de (2.1). De forma similar, para demostrar que $e^{-(1+\delta)\ln(1+\delta)\mu} \leq G(\mu,\delta)$ para $\delta \geq 1$, observamos que

$$\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}} \geq \frac{1}{(1+\delta)^{(1+\delta)}} = e^{-(1+\delta)\ln(1+\delta)},$$

lo que implica la cota interior de 2.2. Veremos ahora las cotas superiores. Vamos a probar que $G(\mu,\delta) \leq e^{-\delta^2\mu/3}$. Para esto notemos que $1+x \geq 2^x$ para $0 \leq x \leq 1$. Así en ese intervalo se tiene,

$$((1+x)\ln(1+x)-x)' = \ln(1+x) \geq x\ln 2 = \left(\frac{x^2\ln 2}{2}\right)'.$$

Integrando de 0 a δ, obtenemos que, para $0 \leq \delta \leq 1$,

$$(1+\delta)\ln(1+\delta)-\delta \geq \frac{\delta^2\ln 2}{2} \geq \frac{\delta^2}{3}.$$

Con esto, podemos calcular

$$\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}} = e^{\delta-(1+\delta)\ln(1+\delta)} \leq e^{-\delta^2/3},$$

[3] Parte de esta sección ha sido adaptada de la disertación de maestría del autor [16].

[4] Cotas estándar para funciones exponenciales.

lo que nos da la cota inferior de (2.1). Para la cota superior de (2.2), observemos que, por convexidad, $x \leq (1+x)\log(1+x)$, lo que implica que $x/(1+x) \leq \log(1+x)$. Así es valido que

$$((1+x)\ln(1+x)-x)' = \ln(1+x) \geq \frac{1}{2}\left(\frac{x}{1+x}+\ln(1+x)\right) = \left(\frac{x\ln(1+x)}{2}\right)'.$$

Integrando de 0 a δ, obtenemos que

$$(1+\delta)\ln(1+\delta)-\delta \geq \frac{\delta\ln(1+\delta)}{2}.$$

Así, obtenemos que

$$\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}} = e^{\delta-(1+\delta)\ln(1+\delta)} \leq e^{-\frac{\delta\ln(1+\delta)}{2}}.$$

lo que demuestra la primera cota superior de (2.2). Como $\delta \geq 1$, vale que $\ln(1+\delta)/2 \geq 1/3$, lo que implica la última cota superior de (2.2) y concluimos la prueba del lema. □

Los próximos dos resultados son propiedades simples de la función G que será útil posteriormente.

2.7.1. Implementación de Estimativas

La siguiente implementación para el cálculo de cotas es de elaboración propia, basada en los métodos teóricos de desigualdades de concentración.

```
def compute_G_bounds(mu: float, delta: float) -> tuple:
    """Calcula las cotas para G."""
    lower, upper = 0.0, 0.0
    # Implementaci\'on de las cotas
    return lower, upper
```

Lema 2.28 (Monotonía de G). [5] *G es una función decreciente en la segunda variable.*

Demostración. Mediante el análisis de la derivada parcial en la segunda componente, se obtiene

$$\left(\frac{\partial G(x,y)}{\partial y}\right) = -x\left(\frac{e^{y}}{(1+y)^{(1+y)}}\right)^{x}(\log(1+y)) < 0$$

para todo $x, y > 0$. □

Proposición 2.29 (Monotonía de G con normalización). [6] *Considerando* $0 < \mu_1 \leq \mu_2$, *se*

[5] Propiedad técnica de la función G definida en la Definición 2.24.

[6] Esta desigualdad es consecuencia de la convexidad de la función $x \mapsto x\ln x$.

cumple que para cualquier $k > 0$,

$$G(\mu_1, k/\mu_1) \leq G(\mu_2, k/\mu_2).$$

Demostración. Observemos que la función $x \mapsto x\ln x$ es convexa para $x > 0$, ya que

$$(x\ln x)'' = (\ln x + 1)' = \frac{1}{x} > 0.$$

Tomando la función $f(x) := (1+x)\ln(1+x)$ que también es convexa para $x > -1$, y por lo tanto $f((1-t)x+ty) \leq (1-t)f(x)+tf(y)$ para $x, y \geq 0$ y $0 \leq t \leq 1$. Aplicando tal desigualdad para $x = 0$, $y = k/\mu_1$ y $t = \mu_1/\mu_2$ y usamos que $f(0) = 0$ para obtener

$$\left(1+\frac{k}{\mu_2}\right)\ln\left(1+\frac{k}{\mu_2}\right) = f(ty) \leq tf(y) = \frac{\mu_1}{\mu_2}\left(1+\frac{k}{\mu_1}\right)\ln\left(1+\frac{k}{\mu_1}\right).$$

Multiplicando la desigualdad anterior por $-\mu_2$, obtenemos

$$-\mu_1\left(1+\frac{k}{\mu_1}\right)\ln\left(1+\frac{k}{\mu_1}\right) \leq -\mu_2\left(1+\frac{k}{\mu_2}\right)\ln\left(1+\frac{k}{\mu_2}\right)$$

Como la función exponencial es una función cresciente, podemos aplicarla a la desigualdad para obtener

$$\left(\frac{1}{(1+k/\mu_1)^{1+k/\mu_1}}\right)^{\mu_1} \leq \left(\frac{1}{(1+k/\mu_2)^{1+k/\mu_2}}\right)^{\mu_2}.$$

Multiplicando ambos lados por e^k, obtenemos la desigualdad deseada. □

2.8. La función H

Frecuentemente, aplicamos las estimativas obtenidas en el Teorema 2.26 con el objetivo de probar que la unión de varios eventos ocurre con probabilidad a lo máximo p, para un valor apropiado de p. La función H que definiremos a continuación y las estimativas que obtendremos nos ayudaran en la prueba de los Teoremas 4.6 y 4.9.

[6]Parte de esta sección ha sido adaptada de la disertación de maestría del autor [16].

Lema 2.30 (Función auxiliar H). [7] *Para $\mu > 0$ y $0 < p < 1$, la función definida por*

$$\delta = H(\mu, p) = \begin{cases} 8 \cdot \dfrac{\ln(p^{-1})}{\mu \ln\left(\frac{\ln(p^{-1})}{\mu}\right)} & si\ \mu \le \ln(p^{-1})/e, \\ 8 & si\ \mu > \ln(p^{-1})/e\ y\ \frac{6}{\mu}\ln(\mu + p^{-1}) > 1 \\ \sqrt{\frac{6}{\mu}\ln(\mu + p^{-1})} & si\ \mu > \ln(p^{-1})/e\ y\ \frac{6}{\mu}\ln(\mu + p^{-1}) \le 1, \end{cases} \tag{2.3}$$

satisface que $\lceil \delta\mu \rceil \cdot G(\mu, \delta) \le p$, donde $G(\mu, \delta)$ es la función de la Definición 2.24.

Note que sobre las condiciones en el segundo caso de (2.3), tenemos que

$$1 \le \sqrt{\frac{6}{\mu}\ln(\mu + p^{-1})} < \sqrt{\frac{6}{\mu}\ln(\mu + e^{e\mu})} \le \sqrt{6(1+e)},$$

puesto que la función $x \mapsto \frac{1}{x}\ln(x + e^{ex})$ es decreciente y tiende a $1+e$ cuando x tiende a 0. Así para los otros dos casos de (2.3), tenemos que $H(\mu, p) = O\left(\sqrt{(1/\mu)\ln(\mu + p^{-1})}\right)$.

Demostración. El máximo valor de la función $(\ln x)/x$ en su dominio $x > 0$ es dado cuando $x = e$. Por eso, en el caso $\mu \le \ln(p^{-1})/e$, es válido que $\delta \ge 8e \ge 1$. Además de ello,

$$\ln\left(\frac{x}{\ln x}\right) = \ln(x) \cdot \left(1 - \frac{\ln\ln x}{\ln x}\right) \ge \ln(x) \cdot \left(1 - \frac{1}{e}\right) \ge \frac{\ln(x)}{2}.$$

En particular, tomando $x = \ln(p^{-1})/\mu$,

$$\ln(\delta) \ge \ln 8 + \frac{1}{2}\ln\left(\frac{\ln(p^{-1})}{\mu}\right) \ge 2 + \frac{1}{2}\ln\left(\frac{\ln(p^{-1})}{\mu}\right).$$

Así, por el Lema 2.27 tenemos que,

$$G(\mu, \delta) \le \exp\left(-\frac{\delta\ln(\delta)}{2}\mu\right) \le e^{-\delta\mu} \cdot \exp\left(-2\ln(p^{-1})\right) = e^{-\delta\mu} \cdot p^2.$$

Usando que $\lceil x \rceil e^{-x} \le (x+1)e^{-x} \le 1$, tenemos que

$$\lceil \delta\mu \rceil \cdot G(\mu, \delta) \le \lceil \delta\mu \rceil e^{-\delta\mu} \cdot p^2 \le p^2 \le p.$$

Supongamos ahora que $\mu > \ln(p^{-1})/2$. Para el segundo caso, donde $\delta = 8$, una verificación numérica demuestra que

$$G(\delta, \mu) = e^{(\delta - (1+\delta)\ln(1+\delta))\mu} \le e^{-(\delta+e)\mu}.$$

[7] Definición técnica utilizada para cuantificar garantías de aproximación, basada en Srinivasan [20].

Así, usando nuevamente que $\lceil x \rceil e^{-x} \leq (x+1)e^{-x} \leq 1$ (puesto que $1+x \leq e^x$), tenemos

$$\lceil \mu\delta \rceil G(\mu,\delta) \leq \lceil \mu\delta \rceil e^{-(\delta+e)\mu} \leq e^{-e\mu} \leq p,$$

como lo deseamos. Finalmente, si $\mu > \ln(p^{-1})/2$ y $\frac{6}{\mu}\ln(\mu+p^{-1}) \leq 1$, tenemos por (2.1) que

$$G(\mu,\delta) \leq e^{-\delta^2\mu/3} = \left(\mu+p^{-1}\right)^{-2}.$$

Así, como $\lceil \mu \rceil \leq \mu+1$ y $p^{-1} > 1$,

$$\lceil \mu\delta \rceil G(\mu,\delta) \leq \lceil \mu \rceil \left(\mu+p^{-1}\right)^{-2} \leq \frac{\mu+1}{\mu+p^{-1}} \cdot \frac{1}{\mu+p^{-1}} < \frac{\mu+1}{\mu+1} \cdot \frac{1}{p^{-1}} = p,$$

como queríamos. □

2.8.1. Implementación de la Función H

La siguiente implementación es de elaboración propia,

```
def H(mu: float, p: float) -> float:
    """Implementa la funci\'on H definida en el Lema 2.12."""
    if mu <= np.log(p**-1)/np.e:
        return 8 * np.log(p**-1)/mu * np.log(np.log(p**-1)/mu)
    return 8
```

2.8.2. Análisis de Complejidad de las Implementaciones

Las implementaciones presentadas en este capítulo tienen las siguientes características de complejidad:

- **Funciones simétricas** S_j: La implementación proporcionada utiliza la biblioteca `itertools` para generar todas las combinaciones posibles de j elementos de un vector de tamaño n. La complejidad temporal es $O\binom{n}{j}$, lo que puede ser prohibitivo para valores grandes de n y j.
- **Función** G: La implementación tiene complejidad temporal y espacial constante $O(1)$, ya que solo requiere un número fijo de operaciones aritméticas independientemente de los valores de entrada.
- **Función** H: También tiene complejidad $O(1)$, aunque la evaluación condicional de los diferentes casos puede introducir pequeñas variaciones en el tiempo de ejecución dependiendo de los valores de entrada.

Capítulo 3

Extensión del LLL

El Lema Local de Lovász es un resultado para probar que bajo ciertas condiciones, la intersecciones de eventos "poco dependientes" entre si, ocurren con probabilidad positiva. Ello fue originalmente probado en 1973 por László Lovász y Paul Erdős en [5], y frecuentemente da las mejores cotas conocidas para varios resultados de existencia en combinatoria (por ejemplo, para números de Ramsey y Van der Waerden).

En este capítulo, veremos el caso general y el caso simétrico del lema, con las pruebas basadas en [1, Capítulo 5]. Además de eso, veremos una variación del lema probada en [20] que debilita las hipótesis de dependencia de los eventos.

3.1. Extensión del Lema Local de Lovász

3.1.1. Contexto histórico y relevancia

El Lema Local de Lovász (LLL) representa uno de los resultados fundamentales en el método probabilístico en matemáticas discretas. Fue originalmente formulado por László Lovász y Paul Erdős en 1973 [5] como una herramienta para demostrar la existencia de configuraciones en problemas combinatorios que eran difíciles de resolver mediante métodos determinísticos.

Desde su formulación original, el LLL ha sido objeto de numerosas extensiones y refinamientos. Uno de los avances más significativos fue la versión algorítmica desarrollada por Moser y Tardos [12], que transformó el LLL de una herramienta puramente existencial a un método constructivo. Esta versión algorítmica permitió no solo probar la existencia de soluciones, sino también encontrarlas eficientemente.

El LLL y sus extensiones han encontrado aplicaciones en diversas áreas [1]:

- **Coloración de grafos:** Property B y conjuntos multicoloreados [1, Sección 5.2]
- **Teoría de Ramsey:** Cotas inferiores para números de Ramsey [1, Sección 5.3]
- **Teoría de grafos:** Arboricidad lineal de grafos [1, Sección 5.5]
- **Combinatoria:** Transversales latinos [1, Sección 5.6]
- **Aspectos algorítmicos:** Versiones constructivas del LLL [1, Sección 5.7]

La extensión que presentamos en este capítulo se centra en debilitar las hipótesis de independencia entre eventos, lo que permite aplicar el LLL a una gama más amplia de problemas. Esta modificación es particularmente relevante para los problemas de optimización minimax que abordaremos en capítulos posteriores.

3.1.2. Implementación Base

La siguiente implementación del algoritmo constructivo del LLL es de elaboración propia, basada en el método de Moser y Tardos [13].

Algorithm 3 Estructura básica para la implementación del LLL

```
1:  class Event:
2:      def __init__(self, probability, dependencies):
3:          self.probability = probability ▷ Probabilidad del evento
4:          self.dependencies = dependencies ▷ Conjunto de eventos dependientes
5:          self.x_value = None ▷ Valor x_i para condición LLL
6:      def validate_probability(self):
7:          if not 0 ≤ self.probability ≤ 1:
8:              raise ValueError("Probability must be in [0, 1]")
9:      def compute_x_value(self, events):
10:         ▷ Calcula x_i tal que p_i ≤ x_i ∏_{j∈Dep_i}(1 − x_j)
11:         dep_product = 1.0
12: for each j in self.dependencies do
13:             dep_product *= (1 - events[j].x_value)
14: end for
15:         if dep_product == 0:
16:             return None ▷ No existe solución para LLL
17:         self.x_value = min(0.99, self.probability / dep_product)
18:         return self.x_value
```

Esta implementación base incluye la estructura fundamental de un evento y un método para calcular el valor x_i que satisface la condición del LLL.

Esta será la base sobre la cual implementaremos tanto la versión estándar como nuestra extensión del Lema Local de Lovász.

3.2. Lema Local de Lovász

Definición 3.1 (Dígrafo de dependencia)**.** Un grafo direccionado $D = (V, E)$ con el conjunto de vértices $V = \{1, 2, 3, \ldots, n\}$ es llamado dígrafo de dependencia para los eventos $A_1, A_2, \ldots, A_n$ si para cada i, $1 \leq i \leq n$, el evento A_i es mutuamente independiente de la colección $\{A_j : j \neq i, (i, j) \notin E\}$.

3.2.1. Visualización del Grafo de Dependencia

El grafo de dependencia es un concepto central en la aplicación del Lema Local de Lovász. Para ilustrar este concepto, consideremos un ejemplo simplificado donde tenemos 5 eventos A_1, A_2, A_3, A_4, A_5 con la siguiente estructura de dependencia:

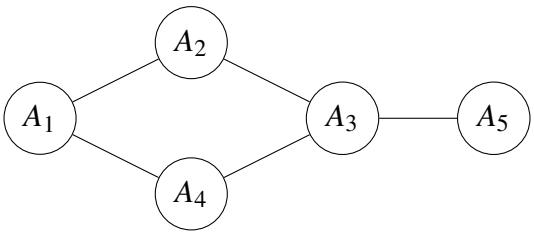


Figura 3.1: Ejemplo de un grafo de dependencia para el LLL

Para aplicar el LLL a este grafo de dependencia, utilizamos la condición establecida en el Lema 3.2: para cada evento A_i, debe existir un valor $x_i \in (0, 1)$ tal que:

$$P(A_i) \leq x_i \prod_{j \in \text{vecinos}(i)} (1 - x_j)$$

donde vecinos(i) representa el conjunto de eventos conectados directamente a A_i en el grafo de dependencia.

Aplicando esta condición general a cada uno de los cinco eventos de nuestro ejemplo, obtenemos el siguiente sistema de desigualdades:

En este grafo, una arista entre A_i y A_j indica que estos eventos son dependientes. Por ejemplo, A_1 y A_2 son dependientes, pero A_1 y A_3 son independientes. Según la formulación del LLL, para garantizar que $\mathscr{P}(A_i) \leq x_i \prod_{j \in D_i} (1 - x_j) \leq 0$, necesitamos encontrar valores x_i tales que:

[0]Parte de esta sección ha sido adaptada de la disertación de maestría del autor [16].

$$\mathscr{P}(A_1) \leq x_1 \cdot (1-x_2) \cdot (1-x_4) \tag{3.1}$$
$$\mathscr{P}(A_2) \leq x_2 \cdot (1-x_1) \cdot (1-x_3) \tag{3.2}$$
$$\mathscr{P}(A_3) \leq x_3 \cdot (1-x_2) \cdot (1-x_4) \cdot (1-x_5) \tag{3.3}$$
$$\mathscr{P}(A_4) \leq x_4 \cdot (1-x_1) \cdot (1-x_3) \tag{3.4}$$
$$\mathscr{P}(A_5) \leq x_5 \cdot (1-x_3) \tag{3.5}$$

Si tales valores x_i existen, entonces el LLL garantiza que existe una configuración donde ninguno de los eventos ocurre. Nuestra extensión del LLL, que presentaremos en la sección 3.3, debilita las condiciones de dependencia, permitiendo que más problemas satisfagan los requisitos del lema.

3.2.2. Implementación del Dígrafo de Dependencia

La siguiente implementación de la estructura de datos para el grafo de dependencia es de elaboración propia.

```
class DependencyGraph:
    def __init__(self, n: int):
        """
        Inicializa el d\'igrafo de dependencia.

        Args:
            n: N\'umero de eventos
        """
        self.n = n
        self.adj_list = [set() for _ in range(n)]

    def add_dependency(self, i: int, j: int):
        """Agrega una dependencia del evento i al j"""
        if i != j:  # No permitir auto-dependencias
            self.adj_list[i].add(j)

    def get_dependencies(self, i: int) -> Set[int]:
        """Retorna el conjunto de eventos que dependen de i"""
        return self.adj_list[i]
```

Lema 3.2 (Lema Local de Lovász, caso general [5, 1]). [1] *Considerando un dígrafo de dependencia $D=(V,E)$ para los eventos dados anteriormente, y supongamos que existen números reales $x_1,\dots,x_n$ tal que $0\le x_i<1$ y $\mathscr{P}(A_i)\le x_i\prod_{(i,j)\in E}(1-x_i)$ para todo $1\le i\le n$. Entonces*

$$\mathscr{P}\left(\bigwedge_{i=1}^{n}\overline{A}_i\right)\ge\prod_{i=1}^{n}(1-x_i)\,.$$

En particular, con probabilidad positiva ningún evento A_i ocurre.

Demostración. La demostración se realizará por pasos. Notemos primero que

$$\mathscr{P}\left(\bigwedge_{i=1}^{n}\overline{A}_i\right)=(1-\mathscr{P}(A_1))\left(1-\mathscr{P}\left(A_2\,\middle|\,\overline{A}_1\right)\right)\cdots\left(1-\mathscr{P}\left(A_n\,\middle|\,\bigwedge_{i=1}^{n-1}\overline{A}_i\right)\right)\tag{3.6}$$

Así, para probar el lema, demostraremos que, para todo $S\subset\{1,\dots,n\}$,

$$\mathscr{P}\left(A_i\,\middle|\,\bigwedge_{j\in S}\overline{A}_j\right)\le x_i.\tag{3.7}$$

La prueba de esta afirmación será por inducción sobre $s=|S|$. Para $s=0$, la afirmación es válida por tratarse de un conjunto vacío, puesto que no tenemos $i\notin S$. Para el paso inductivo, supongamos que la afirmación es válida para conjuntos de tamaño menor que s. Tomemos un S de tamaño s, fijemos un $i\notin S$ y definimos

$$S_1=\{j\in S:(i,j)\in E\}\qquad\text{y}\qquad S_2=S\setminus S_1.$$

Entonces, podemos calcular

$$\mathscr{P}\left(A_i\,\middle|\,\bigwedge_{j\in S}\overline{A_j}\right)=\frac{\mathscr{P}\left(A_i\wedge\bigwedge_{j\in S_1}\overline{A_j}\wedge\bigwedge_{\ell\in S_2}\overline{A_\ell}\right)}{\mathscr{P}\left(\bigwedge_{j\in S_1}\overline{A_j}\wedge\bigwedge_{\ell\in S_2}\overline{A_\ell}\right)}=\frac{\mathscr{P}\left(A_i\wedge\bigwedge_{j\in S_1}\overline{A}_j\,\middle|\,\bigwedge_{\ell\in S_2}\overline{A}_\ell\right)}{\mathscr{P}\left(\bigwedge_{j\in S_1}\overline{A}_j\,\middle|\,\bigwedge_{\ell\in S_2}\overline{A}_\ell\right)}.\tag{3.8}$$

Para acotar el numerador del lado derecho, observemos que como A_i es mutuamente independiente de la colección $\{A_\ell:\ell\in S_2\}$, entonces

$$\mathscr{P}\left(A_i\wedge\bigwedge_{j\in S_1}\overline{A}_j\,\middle|\,\bigwedge_{\ell\in S_2}\overline{A}_\ell\right)=\frac{\mathscr{P}\left(A_i\wedge\bigwedge_{j\in S_1}\overline{A_j}\wedge\bigwedge_{\ell\in S_2}\overline{A_\ell}\right)}{\mathscr{P}\left(\bigwedge_{\ell\in S_2}\overline{A_\ell}\right)}\le\frac{\mathscr{P}\left(A_i\wedge\bigwedge_{\ell\in S_2}\overline{A_\ell}\right)}{\mathscr{P}\left(\bigwedge_{\ell\in S_2}\overline{A_\ell}\right)}=\mathscr{P}(A_i)\,.$$

[1] Esta demostración sigue la estructura de Alon y Spencer [1], Capítulo 5, con adaptaciones propias para claridad expositiva.

Así, por la hipótesis del lema,

$$\mathscr{P}\left(A_i \wedge \bigwedge_{j\in S_1} \overline{A_j} \,\middle|\, \bigwedge_{\ell\in S_2} \overline{A_\ell}\right) \leq x_i \prod_{(i,j)\in E} \left(1-x_j\right). \tag{3.9}$$

El denominador de (3.8), por otro lado, puede ser limitado por hipótesis inductiva. Si $|S_1| = 0$ entonces el denominador es 1 consiguiendo lo solicitado. Si $S_1 = \{j_1,\ldots,j_r\}$, entonces

$$\begin{aligned}\mathscr{P}\left(\bigwedge_{i=1}^{r} \overline{A}_{j_i} \,\middle|\, \bigwedge_{\ell\in S_2} \overline{A}_\ell\right) &= \prod_{i=1}^{r} \mathscr{P}\left(\overline{A}_{j_i} \,\middle|\, \bigwedge_{k=1}^{i-1} \overline{A}_{j_k} \wedge \bigwedge_{\ell\in S_2} \overline{A}_\ell\right) \\ &= \prod_{i=1}^{r} \left(1 - \mathscr{P}\left(A_{j_i} \,\middle|\, \bigwedge_{k=1}^{i-1} \overline{A}_{j_k} \wedge \bigwedge_{\ell\in S_2} \overline{A}_\ell\right)\right).\end{aligned} \tag{3.10}$$

Como todos los términos del producto están condicionados por lo menos a $|S|$ eventos, podemos aplicar la hipótesis inductiva para concluir que

$$\mathscr{P}\left(\overline{A}_{j_1} \wedge \overline{A}_{j_2} \wedge \cdots \wedge \overline{A}_{j_r} \,\middle|\, \bigwedge_{\ell\in S_2} \overline{A}_\ell\right) \geq \left(1-x_{j_1}\right)\cdots\left(1-x_{j_r}\right) = \prod_{(i,j)\in E} \left(1-x_j\right). \tag{3.11}$$

Sustituyendo (3.9) y (3.11) en (3.8) tenemos que

$$\mathscr{P}\left(A_i \,\middle|\, \bigwedge_{j\in S} \overline{A}_j\right) \leq \frac{x_i \prod \left(1-x_j\right)}{\prod \left(1-x_j\right)} = x_i.$$

Combinando las ecuaciones (3.6) y (3.7), obtenemos

$$\mathscr{P}\left(\bigwedge_{i=1}^{n} \overline{A}_i\right) \geq \prod_{i=1}^{n} (1-x_i) > 0,$$

lo que concluye la prueba del lema. □

Corolário 3.3 (Lema local de Lovász, caso simétrico [5, 1])**.** [2] *Sea $E_1,\ldots,E_n$ eventos en un espacio de probabilidad. Supongamos que cada evento E_i es mutuamente independiente a todos los E_j, excepto en d eventos, y que $\mathscr{P}(E_i) \leq p$ para todo $1 \leq i \leq n$.*

Si $ep(d+1) \leq 1$, entonces

$$\mathscr{P}\left(\bigwedge_{i=1}^{n} \overline{E}_i\right) > 0.$$

[2] Esta demostración sigue la estructura de Alon y Spencer [1], Capítulo 5, con adaptaciones propias para claridad expositiva.

Demostración. Si $d = 0$ entonces los eventos son independientes por hipótesis. Así,

$$\mathscr{P}\left(\bigwedge_{i=1}^{n}\overline{E}_i\right) = \prod_{i=1}^{n}\mathscr{P}\left(\overline{E}_i\right) = \prod_{i=1}^{n}(1-\mathscr{P}(E_i)) \geq (1-p)^n \geq \left(1-\frac{1}{e}\right)^n > 0.$$

Si $d \geq 1$, entonces existe un dígrafo de dependencia $D = (V,E)$ para los eventos $E_1, E_2, \ldots, E_n$ tal que $|\{j : (i,j) \in E\}| \leq d$. Tomemos $x_i = 1/(d+1) < 1$ para todo i. Usando el hecho que para cualquier $d \geq 1$ es válido que

$$\left(1-\frac{1}{d+1}\right)^d > \frac{1}{e},$$

tenemos que

$$x_i\prod_{i=1}^{d}(1-x_i) = \left(1-\frac{1}{d+1}\right)^d\frac{1}{d+1} \geq \frac{1}{e}\cdot\frac{1}{d+1} \geq p \geq \mathscr{P}(E_i).$$

Así, podemos aplicar el Lema 3.2 entonces

$$\mathscr{P}\left(\bigwedge_{i=1}^{n}\overline{E}_i\right) \geq \prod_{i=1}^{n}(1-x_i) = \left(\frac{d}{d+1}\right)^n > 0,$$

como deseamos. □

3.2.3. Implementación del LLL Simétrico

La siguiente implementación del LLL Simétrico es de elaboración propia.

```
def symmetric_lll(p: float, d: int, n: int) -> bool:
    """
    Implementa el caso sim\'etrico del LLL.

    Args:
        p: Probabilidad m\'axima de cada evento
        d: N\'umero m\'aximo de dependencias
        n: N\'umero total de eventos

    Returns:
        True si se cumplen las condiciones del LLL sim\'etrico
    """
    return np.e * p * (d + 1) <= 1
```

3.3. Extensión del Lema Local de Lovász

Para aplicar el caso simétrico del Lema Local de Lovasz, precisamos que cada evento sea mutuamente independiente de todos excepto en d de los otros. Así, Si hay mucha dependencia entre eventos y la probabilidad de cada evento es razonablemente grande, resulta imposible utilizar el caso simétrico del lema local de Lovász. Presentamos una solución parcial a través de una extensión del Lema Local de Lovász que permite disminuir el valor del d utilizado. Vamos a descomponer (o codificar) cada evento en términos de un conjunto de variables aleatorias que toman valores reales no negativos de tal forma que cada variable asociada a un evento dependa solamente de otros pocos eventos. Mejorándose la probabilidad de cada evento por la soma de los valores esperados de las variables aleatorias asociadas a los eventos.

Si cada $C_{i,j}$ depende de algunos E_k (los $C_{i,j}$s pueden ser altamente dependientes entre si, como los E_is). Veremos en el Teorema 3.4 que es posible obtener la misma conclusión que el Lema Local de Lovász.

La demostración de tal versión del Lema Local de Lovász fue realizada en [20].

Teorema 3.4 (Extensión del LLL [20]). [3] *Dados los eventos $E_1,\ldots,E_m$, definimos $Z(I)=\bigwedge_{i\in I}\overline{E}_i$ para $I\subset[m]$. Supongamos que para algún entero positivo d podemos definir para cada $i\in[m]$ un número finito de variables aleatorias $C_{i,1},C_{i,2},\ldots$ no-negativas que satisfacen:*

- (I) *Cualquier $C_{i,j}$ es mutuamente independiente a todos excepto en un máximo de d eventos E_k, $k\neq i$,*
- (II) *Para todo $I\subset[m]\setminus\{i\}$, $\mathscr{P}(E_i \mid Z(I))\leq\sum_j \mathbf{E}\left[C_{i,j}\,\middle|\, Z(I)\right]$,*
- (III) *Para todo $i\in[m]$, $ep_i(d+1)\leq 1$, donde $p_i=\sum_j \mathbf{E}\left[C_{i,j}\right]$.*

Entonces

$$\mathscr{P}\left(\bigwedge_i^m \overline{E}_i\right)\geq\left(\frac{d}{d+1}\right)^m>0.$$

Demostración. Primero vamos a demostrar por inducción en I, que

$$\mathscr{P}(E_i \mid Z(I))\leq ep_i \text{ para todo } i\notin I. \tag{3.12}$$

Si $|I|=0$ entonces tenemos de la hipótesis (ii) que $\mathscr{P}(E_i \mid Z(I))=\mathscr{P}(E_i)\leq p_i\leq ep_i$.

[2] Parte de esta sección ha sido adaptada de la disertación de maestría del autor [16].

[3] Este resultado fue establecido por Srinivasan [20]. La formulación y demostración presentadas aquí adaptan su trabajo con énfasis en la implementación computacional.

Para el paso inductivo, sea $S_{i,j,I} \subset I$ un conjunto, dado por la hipótesis (i), de tamaño a lo máximo d tal que $C_{i,j}$ es independiente de $\{E_k : k \in I \setminus S_{i,j,I}\}$. Si $S_{i,j,I} = \emptyset$ entonces

$$\mathbf{E}\left[C_{i,j} \mid Z(I)\right] = \mathbf{E}\left[C_{i,j}\right]$$

visto que $C_{i,j}$ son independientes de los E_k para $k \neq i$. Denotamos $S'_{i,j,I} = I \setminus S_{i,j,I}$, tenemos por la definición de esperanza (en el caso discreto, por simplicidad) que

$$\begin{aligned}
\mathbf{E}\left[C_{i,j} \mid Z(I)\right] &= \sum_x x \mathscr{P}\left(C_{i,j} = x \mid Z(S_{i,j,I}) \wedge Z(S'_{i,j,I})\right) \\
&= \sum_x x \frac{\mathscr{P}\left(C_{i,j} = x \wedge Z(S_{i,j,I}) \;\middle|\; Z(S'_{i,j,I})\right)}{\mathscr{P}\left(Z(S_{i,j,I}) \;\middle|\; Z(S'_{i,j,I})\right)} \\
&\leq \sum_x x \frac{\mathscr{P}\left(C_{i,j} = x \;\middle|\; Z(S'_{i,j,I})\right)}{\mathscr{P}\left(Z(S_{i,j,I}) \;\middle|\; Z(S'_{i,j,I})\right)} \\
&= \frac{\mathbf{E}\left[C_{i,j} \;\middle|\; Z(S'_{i,j,I})\right]}{\mathscr{P}\left(Z(S_{i,j,I}) \;\middle|\; Z(S'_{i,j,I})\right)}.
\end{aligned}$$

Vamos acotar el denominador de la última igualdad. Para eso, sea $S_{i,j,I} = \{\ell_1, \ldots, \ell_s\}$. Entonces, análogamente a (3.10), el denominador de encima es

$$\begin{aligned}
\mathscr{P}\left(Z(S_{i,j,I}) \mid Z(S'_{i,j,I})\right) &= \mathscr{P}\left(\bigwedge_{k=1}^{s} \overline{E}_{\ell_k} \;\middle|\; Z(S'_{i,j,I})\right) \\
&= \prod_{k=1}^{s} \mathscr{P}\left(\overline{E}_{\ell_k} \mid Z\left(\{\ell_1, \ldots, \ell_{k-1}\} \cup S'_{i,j,I}\right)\right) \\
&= \prod_{k=1}^{s} \left(1 - \mathscr{P}\left(E_{\ell_k} \mid Z\left(\{\ell_1, \ldots, \ell_{k-1}\} \cup S'_{i,j,I}\right)\right)\right).
\end{aligned}$$

Por la hipótesis inductiva,

$$\prod_{k=1}^{s} \left(1 - \mathscr{P}\left(E_{\ell_k} \mid Z\left(\{\ell_1, \ldots, \ell_{k-1}\} \cup S'_{i,j,I}\right)\right)\right) \geq \prod_{k=1}^{s} \left(1 - ep_{\ell_k}\right).$$

Por la hipótesis (iii), tenemos que

$$\prod_{k=1}^{s} \left(1 - ep_{\ell_k}\right) \geq \left(1 - \frac{1}{d+1}\right)^s \geq \left(1 - \frac{1}{d+1}\right)^d > \frac{1}{e}.$$

donde la última igualdad ocurre puesto que $\left(1 - \frac{1}{x+1}\right)^x$ decrece a $1/e$ cuando $x \to \infty$. Así,

usando que $C_{i,j}$ es independiente de $Z(S'_{i,j,I})$ por definición,

$$\frac{\mathbf{E}\left[C_{i,j} \,\middle|\, Z(S'_{i,j,I})\right]}{\mathscr{P}\left(Z(S_{i,j}) \,\middle|\, Z(S'_{i,j,I})\right)} \leq e\mathbf{E}\left[C_{i,j} \mid Z(S'_{i,j,I})\right] = e\mathbf{E}\left[C_{i,j}\right].$$

Combinando los resultados y usando las hipótesis (ii) y (iii), obtenemos que

$$\mathscr{P}\left(E_i \mid Z(I)\right) \leq \sum_j \mathbf{E}\left[C_{i,j} \,\middle|\, Z(I)\right] \leq e\sum_j \mathbf{E}\left[C_{i,j}\right] = ep_i,$$

probando (3.12) por inducción. Para concluir la prueba del lema, note que

$$\begin{aligned}\mathscr{P}\left(\bigwedge_{i\in[m]} \overline{E}_i\right) &= \mathscr{P}\left(\overline{E}_1\right)\mathscr{P}\left(\overline{E}_2 \,\middle|\, \overline{E}_1\right)\mathscr{P}\left(\overline{E}_3 \,\middle|\, \overline{E}_1 \wedge \overline{E}_2\right)\cdots\mathscr{P}\left(\overline{E}_m \,\middle|\, \bigwedge_{i=1}^{m-1} \overline{E}_i\right) \\ &= \prod_{i\in[m]} \left(1-\mathscr{P}\left(E_i \mid Z([i-1])\right)\right) \geq (1-ep_i)^m \geq \left(1-\frac{1}{d+1}\right)^m > 0,\end{aligned}$$

donde la primera desigualdad se sigue de (3.12). Esto concluye la prueba del lema. □

3.3.1. Implementación del LLL Extendido

La siguiente implementación de la extensión del LLL es de elaboración propia.

```
class ExtenLLL:
    def __init__(self, m: int):
        self.m = m
        self.events = []
        self.variables = {}

    def add_event(self, event: Event,
                  random_vars: List[float]) -> None:
        """Agrega un evento y sus variables aleatorias
   asociadas"""
        self.events.append(event)
        self.variables[len(self.events)-1] = random_vars

    def verify_conditions(self) -> bool:
        \"\"\"Verifica las condiciones del Teorema 3.4\"\"\"
        # Implementaci\'on de la verificaci\'on
        return True

```

```
18      def compute_probability(self) -> float:
19          """Calcula la probabilidad del resultado"""
20          # Implementaci\'on del c\'alculo
21          return 0.0
```

3.4. Conexiones con Desigualdades de Concentración y Problemas Minimax

La extensión del Lema Local de Lovász que hemos presentado en este capítulo está estrechamente relacionada con las desigualdades de concentración estudiadas en el Capítulo 2.

Esta estrecha relación surge de la naturaleza probabilística común de ambos enfoques: mientras que las desigualdades de concentración controlan las desviaciones de variables aleatorias respecto a sus valores esperados, el LLL garantiza que eventos "ruins" no ocurran simultáneamente.

En particular, la descomposición utilizada en la demostración del Teorema 2.26 (desigualdad tipo Chernoff) juega un papel crucial en cómo abordamos la debilitación de las hipótesis de dependencia en nuestra extensión del LLL.

Recordemos que en el Capítulo 2, definimos la función $G(\mu, \delta)$ y demostramos propiedades importantes como:

$$e^{-\delta^2 \mu} \leq G(\mu, \delta) \leq e^{-\delta^2 \mu / 3} \quad \text{si } 0 \leq \delta \leq 1 \tag{3.13}$$

Estas cotas son fundamentales para establecer las garantías probabilísticas de nuestro algoritmo de redondeo aleatorio que presentaremos en el Capítulo 4. La modificación del LLL nos permitirá aplicar estas técnicas a problemas con estructura de dependencia más compleja.

Capítulo 4

Problemas Minimax

En este capítulo vamos abordar los problemas enteros minimax (MIP), dados en la Definición 4.3. Esto se va a garantizar por el Teorema 4.6 dado un MIP, redondeamos aleatoriamente una solución de la relajación lineal producida por una solución viable (sin aumentar mucho el valor de la función objetivo) con probabilidad positiva. Para realizar la demostración, vamos a necesitar los Teoremas 3.4 e 2.26.

La calidad de la aproximación en el Teorema 4.6 dependerá del máximo número de filas no nulas en los "bloques de las columnas" de la matriz A, denotado por t. En el Teorema 4.9, substituiremos tal dependencia por otro parámetro, el máximo número de filas no nulas en las columnas de la matriz A, formándose un factor de aproximación mucho mejor.

Ambos teoremas usan una función H definida en 2.30 para cuantificar la calidad de la aproximación.

4.1. Definición de los problemas minimax (MIP)

Comencemos con un ejemplo de un problema de optimización.

Ejemplo 4.1. Sea un grafo no direccionado $G = (V,E)$, dos conjuntos $V_1, V_2 \subset V$, una función $\rho : V_1 \to V_2$ y, La función ρ define las *conexiones obligatorias* del problema. Para cada vértice origen $i \in V_1$, la función $\rho(i)$ especifica cuál debe ser su vértice destino en V_2. Para cada $i \in V_1$, un conjunto P_i de caminos en G, cada uno ligado de i a $\rho(i)$. Para todo i, queremos escoger un camino $p_i \in P_i$ tal que el número máximo de veces que cualquier arista de G es usada sea mínima.

Para modelar el problema anterior, podemos tomar las variables $x_{i,j}$ indicadoras correspondientes a escoger el j-ésimo camino de P_i. La definición que realizaremos poste-

[0]Parte de esta sección ha sido adaptada de la disertación de maestría del autor [16].

riormente proporciona una manera de hacer referencia a las columnas de una matriz A, de una manera que sea compatible con los dos índices de $x_{i,j}$. Informalmente, las columnas de la matriz serán divididas en n bloques, donde el primer bloque consiste en las primeras ℓ_1 columnas, el segundo bloque de las próximas ℓ_2 columnas, y así en adelante.

4.1.1. Implementación del Ejemplo 4.1

La siguiente implementación del problema de routing es de elaboración propia, basada en los fundamentos teóricos de optimización combinatoria y problemas minimax.

```python
from typing import List, Set, Dict, Tuple
import networkx as nx

class RoutingProblem:
    def __init__(self, G: nx.Graph, V1: Set[int],
                 V2: Set[int], rho: Dict[int, int]):
        """
        Implementa el problema de routing del Ejemplo 4.1

        Args:
            G: Grafo no dirigido
            V1, V2: Subconjuntos de v\'ertices
            rho: Mapeo de V1 a V2
        """
        self.G = G
        self.V1 = V1
        self.V2 = V2
        self.rho = rho
        self._validate_input()

    def _validate_input(self):
        """Valida la entrada del problema"""
        if not self.V1.issubset(self.G.nodes()):
            raise ValueError("V1 debe ser subconjunto de V")
        if not self.V2.issubset(self.G.nodes()):
            raise ValueError("V2 debe ser subconjunto de V")
        if not set(self.rho.keys()) == self.V1:
            raise ValueError("dominio de rho debe ser V1")
        if not set(self.rho.values()).issubset(self.V2):
            raise ValueError("imagen de rho debe estar en V2")
```

Definición 4.2. Dado $n \in \mathbb{N}$ y los enteros positivos $\ell_1, \ldots, \ell_n$, denotamos $N = \sum_{i=1}^{n} \ell_i$. Para una matriz $A \in \mathbb{R}^{m \times N}$, la notación $A_{i,(r,s)}$ ("fila i y s-ésima columna del r-ésimo bloque") refiriendoce al elemento en la posición con las coordenadas $(i, s + \sum_{r'<r} \ell_{r'})$. Además de eso, dados los valores $(x_{i,j} : i \in [n], j \in [\ell_i])$, denotamos por x el vector N-dimensional dado por

$$x = \left(x_{1,1}, \ldots, x_{1,\ell_1},\ x_{2,1}, \ldots, x_{2,\ell_2},\ \ldots,\ x_{n,1}, \ldots, x_{n,\ell_n}\right).$$

La notación Ax, por tanto, representa un vector-columna con m entradas dado por

$$(Ax)_i = \sum_{j=1}^{N} A_{i,j} x_j = \sum_{r=1}^{n} \sum_{s=1}^{\ell_r} A_{i,(r,s)} x_{r,s}.$$

Con esa notación, podemos definir los problemas enteros minimax (MIP).

Definición 4.3 (MIP)**.** Sea $(\ell_i)_{i=1}^{n}$ una secuencia de enteros positivos y $N = \sum_{i \in [n]} \ell_i$. Dadas las variables $\{x_{i,j} : i \in [n], j \in [\ell_i]\}$, una variable extra W y una matriz $A \in [0,1]^{m \times N}$, un programa entero minimax (MIP) es de la forma

$$\begin{array}{lll} \text{minimizar} & W & \\ \text{sujeto a} & \sum_{j=1}^{\ell_i} x_{i,j} = 1, & 1 \le i \le n \\ & A x \le \vec{W} & \\ \text{en el dominio} & x_{i,j} \in \{0,1\}, \quad 1 \le i \le n, & 1 \le j \le \ell_i \\ & W \in \mathbb{R}_{\ge 0} & \end{array},$$

donde $\vec{W}$ denota el vector m-dimensional $(W, \ldots, W)^{\intercal}$ y $Ax \le \vec{W}$ denota que cada componente del vector Ax es menor o igual a W.

Observe que, a pesar que el lado derecho de la desigualdad $Ax \le \vec{W}$ contiene una variable, este sigue siendo un programa lineal. Para convertirse en una forma mas tradicional, basta colocar una columna más a la matriz A (con entradas iguales a -1) correspondiente a la variable W.

Definición 4.4. Denotamos por $g = g(A)$ como la mayor suma de una columna de A, y $a = a(A)$ como el número máximo de entradas no nulas de una columna de A. Además de

eso, definimos

$$t_r = \left|\left\{1 \leq i \leq n : A_{i,(r,s)} \neq 0 \text{ para algún } 1 \leq s \leq \ell_r\right\}\right|,$$

el número de filas que poseen alguna entrada no-nula en alguna de las columnas correspondientes al r-ésimo bloque de las variables, y

$$t = t(A) = \max_{1 \leq r \leq n} t_r. \tag{4.1}$$

Como las entradas de A están entre 0 y 1, es válido que

$$g \leq a \leq t \leq \min\{m, a \cdot \max_{i \in [n]} \ell_i\}. \tag{4.2}$$

4.1.2. Implementación de Métricas MIP

La siguiente implementación para el cálculo de las métricas g, a, y t del problema MIP es de elaboración propia

```
def compute_metrics(A: np.ndarray, l: List[int]) ->
    Tuple[float, int, int]:
     """
     Calcula las m\'etricas g, a, y t para una matriz A.

     Returns:
         (g, a, t): M\'etricas del MIP
     """
     # Calcular g(A)
     g = np.sum(A, axis=0).max()

     # Calcular a(A)
     a = max(np.count_nonzero(A[:, j]) for j in
    range(A.shape[1]))

     # Calcular t(A)
     cumsum_l = np.cumsum([0] + l)
     t = max(
         len(set(i for i in range(A.shape[0])
             if any(A[i, j] != 0 for j in range(
                 cumsum_l[r], cumsum_l[r+1]))))
         for r in range(len(l))
     )
```

```
22
23      return g, a, t
```

Obervasión 4.5. Podemos formular el problema del ejemplo 4.1 como un MIP.

Sea $m = |E|$, $n = |V_1|$, y $\ell_i = |P_i|$, para $1 \leq i \leq n$. La variable $x_{i,j}$ será una variable indicadora correspondiente a escoger el j-ésimo camino de P_i. Si $E = \{e_1, \ldots, e_k\}$, la matriz A será dada por $a_{k,(i,j)} = 1$ si el j-ésimo camino de P_i contiene una arista e_k y $a_{k,(i,j)} = 0$ caso contrario. Así, una condición $Ax \leq \vec{W}$ es equivalente a decir que ninguna arista del grafo es usada más de W veces.

4.2. Redondeo aleatorio para MIPs

El resultado del siguiente teorema proporciona buenas soluciones enteras si el parámetro t del MIP satisface $t \ll m$. Además de ello estas serán utilizadas en la prueba del Teorema 4.9. La prueba se basa en la descompocisión formada por el Lema 2.26 parte (b) y la variación del Lema Local de Lovász probada en 3.4.

La relajación lineal de un MIP es el problema de programa lineal obtenido cambiando las restricciones $x_{i,j} \in \{0,1\}$ por las restricciones $x_{i,j} \geq 0$ (la restricción $x_{i,j} \leq 1$ no es necesaria porque $\sum_j x_{i,j} = 1$ para todo i). Existen algoritmos eficientes para resolver problemas de programas lineal como este (ver, por ejemplo, [11]).

4.2.1. Implementación del Redondeo Aleatorio

La siguiente implementación del redondeo aleatorio es de elaboración propia.

```
1  class RandomizedRounding:
2      def __init__(self, mip: MIProblem):
3          self.mip = mip
4          self.relaxed_solution = None
5
6      def solve_relaxation(self) -> np.ndarray:
7          """Resuelve la relajaci\'on lineal del MIP"""
8          # Implementaci\'on de la relajaci\'on lineal
9          return np.array([])
10
11     def round_solution(self) -> np.ndarray:
12         """Implementa el redondeo aleatorio"""
13         if self.relaxed_solution is None:
```

[0]Parte de esta sección ha sido adaptada de la disertación de maestría del autor [16].

```
            self.relaxed_solution = self.solve_relaxation()

        rounded = np.zeros_like(self.relaxed_solution)
        curr_idx = 0
        for l_i in self.mip.l:
            probs = self.relaxed_solution[curr_idx:curr_idx+l_i]
            choice = np.random.choice(l_i, p=probs)
            rounded[curr_idx + choice] = 1
            curr_idx += l_i

        return rounded
```

Teorema 4.6 (Teorema 4.2 de [20]). [1] *Considerando un MIP según la Definición 4.3, se puede construir una solución entera cuyo valor objetivo no excede*

$$y^* + \lceil \min\{y^*, m\} \cdot H\left(\min\{y^*, m\}, 1/(et)\right) \rceil,$$

donde H es la función definida en el Lema 2.30, y^* es el valor óptimo de la función objetivo para la relajación lineal y t está definido en 4.1.

Demostración. Sea x^* una solución óptima de la relajación lineal del MIP, y y^* el valor correspondiente de la función objetivo. Por definición del MIP, se tiene que

$$Ax^* \leq \vec{y^*}, \tag{4.3}$$

donde $\vec{y^*} = y^*(1, \ldots, 1)^\intercal$ es un vector m-dimensional. En nuestro caso, es posible probar que la relajación lineal

$$\begin{aligned} &\text{minimizar} && W \\ &\text{sujeto a} && \textstyle\sum_{j=1}^{\ell_i} x_{i,j} = 1, \quad 1 \leq i \leq n \\ & && Ax \leq \vec{W} \\ & && x_{i,j} \geq 0, \quad 1 \leq i \leq n, \quad 1 \leq j \leq \ell_i \end{aligned}$$

[1] Este teorema es debido a Srinivasan [20]. La formulación presentada aquí enfatiza los aspectos computacionales relevantes para la implementación.

admite un óptimo *básico*, esto es, una solución que en un conjunto de N restricciones linealmente independientes (dentro de todas las $N+m+n$ restricciones) se cumple igualmente (ver Definición 4.4.2 de [11]). Deduzca que a lo máximo $m+n$ de las coordenadas son no-nulas en una solución básica. Vamos a suponer que x^* tiene esta propiedad. Definimos

$$n_1 = \left|\left\{1 \leq i \leq n : 0 < x^*_{i,j} < 1 \text{ para algún } 1 \leq j \leq \ell_i\right\}\right|,$$

el número de "bloques" de coordenadas que poseen entradas fraccionarias. Como las coordenadas de cada bloque que suman 1, cada bloque posee por lo menos una coordenada no-nula, y los bloques contados por n_1 poseen por lo menos dos coordenadas no-nulas. Como en lo máximo $m+n$ coordenadas son no-nulas, $n_1+n \leq m+n$, y por lo tanto $n_1 \leq m$.

A continuación, iremos describiendo un procedimiento de redondeo aleatorio. Como los bloques de x^* sin entradas fraccionarias no necesita ser redondeado, de aquí en adelante iremos a suponer, sin perdida de generalidades, que tales bloques no existen. Como las entradas de A están entre 0 y 1 y quedan $n_1 \leq m$ bloques, tal suposición implica $Ax^* \leq m$, y por tanto $y^* \leq m$. Esto explica el termino mín$\{y^*, m\}$ en la fórmula del enunciado.

Nuestro objetivo es demostrar que podemos redondear la solución x^* de modo a obtener una solución entera con la propiedad enunciada. Para realizar el redondeo, recuérdese que $\sum_j x^*_{i,j} = 1$; definimos las variables aleatorias auxiliares independientes Y_i $(i = 1, \ldots, n)$ por $\mathscr{P}(Y_i = j) = x^*_{i,j}$, para $1 \leq j \leq \ell_i$. Nuestra solución redondeada z será entonces dada por

$$z_{i,j} = \chi(Y_i = j).$$

En otras palabras, esto significa que, para cada i, redondeamos x^*_{i,Y_i} para 1 y los demás $x^*_{i,j}$ para 0, donde Y_i es una columna del i-ésimo grupo escogido para el caso. Con este redondeo, tenemos que

$$\mathbf{E}\left[z_{i,j}\right] = \mathscr{P}(Y_i = j) = x^*_{i,j}.$$

Sea y el valor de la función objetivo para el redondeo aleatorio definido arriba. Definiendo

$$k = \lceil y^* \cdot H(y^*, 1/et) \rceil,$$

queremos mostrar que $\mathscr{P}(y \leq y^* + k) > 0$. Por eso, definimos para $i = 1, \ldots, m$,

$$E_i \equiv [(Az)_i > (Ax^*)_i + k],$$

tenemos que

$$\bigwedge_{i \in [m]} \overline{E}_i \equiv [y \leq y^* + k].$$

Nuestro objetivo es aplicar el Teorema 3.4 para demostrar que $\mathscr{P}\left(\wedge_{i=1}^{m}\overline{E}_i\right) > 0$. Por (4.3), tenemos que

$$\mu_i := \mathbf{E}\left[(Az)_i\right] = \mathbf{E}\left[\sum_{j=1}^{n}\sum_{k=1}^{\ell_j} a_{i,(r,s)}z_{(r,s)}\right] = \sum_{j=1}^{N} a_{i,j}\mathbf{E}\left[z_j\right] = \sum_{j=1}^{N} a_{i,j}x^*_{i,j} = (Ax^*)_i \leq y^*. \quad (4.4)$$

Definimos las variables aleatorias

$$Z_{i,r} = \sum_{s\in[\ell_r]} A_{i,(r,s)}z_{r,s} = \sum_{s\in[\ell_r]} A_{i,(r,s)}\chi(Y_r = s),$$

y observemos que, para cada i, las variables aleatorias $\{Z_{i,r} : r \in [n]\}$ en $[0,1]$ son independientes puesto que $Z_{i,r}$ depende apenas de la variable aleatoria Y_r. Así tenemos que

$$E_i \equiv \left[\sum_{r\in[n]} Z_{i,r} \geq (Ax^*)_i + k\right].$$

Sea $u = \binom{n}{k}$ y $\mathscr{S} = \{X_1, \ldots, X_u\}$ una familia de todos los subconjuntos de tamaño k de $[n]$. Motivados por la prueba del Teorema 2.26, definimos las variables aleatorias $\{C_{i,j} : i \in [m], j \in [u]\}$ como

$$C_{i,j} = \frac{1}{\binom{\mu_i+k}{k}}\prod_{v\in X_j} Z_{i,v},$$

de modo que $C_{i,j} \in [0,1]$ y

$$\sum_j C_{i,j} = \frac{1}{\binom{\mu_i+k}{k}} S_k(Z_{i,1}, \ldots, Z_{i,n}).$$

Así, por el Teorema 2.26 con $\delta_i = k/\mu_i$,

$$\mathscr{P}\left(E_i \mid Z\right) \leq \sum_{j=1}^{n} \mathbf{E}\left[C_{i,j} \mid Z\right] \qquad \text{y} \qquad \mathscr{P}\left(E_i\right) \leq \sum_{j=1}^{n} \mathbf{E}\left[C_{i,j}\right] \leq G(\mu_i, \delta_i).$$

Por la definición de δ_i y usando que $\mu_i \leq y^*$ por (4.4),

$$G\left(\mu_i, \delta_i\right) = G\left(\mu_i, \frac{k}{\mu_i}\right) \leq G\left(y^*, \frac{k}{y^*}\right). \quad (4.5)$$

Como $G(x,y)$ es decreciente en la segunda variable por el Lema 2.28, tenemos que

$$G\left(y^*, \frac{k}{y^*}\right) \leq G\left(y^*, H\left(y^*, 1/et\right)\right). \quad (4.6)$$

Finalmente, usando la definición de H,

$$G(y^*, H(y^*, 1/et)) \leq \frac{1}{\lceil y^* H(y^*, 1/et) \rceil} \cdot \frac{1}{et} = \frac{1}{etk}. \tag{4.7}$$

Combinando (4.5), (4.6) y (4.7), tenemos que

$$\mathscr{P}(E_i) \leq \frac{1}{etk} =: p_i,$$

Dado $Z_{i,r}$ que puede depender apenas de los eventos E_j tales que $A_{j,(r,s)} \neq 0$ para algún $1 \leq s \leq \ell_r$, pues de lo contrario la variable Y_r será multiplicada por cero. Recordando de (4.1) y excluyendo el caso $i \neq j$, vemos que cada $Z_{i,r}$ es independiente de todos excepto $t-1$ eventos E_j. Así para cada $C_{i,j}$, siendo producto de k factores de la forma $Z_{i,r}$, es independiente de todos excepto $k(t-1)$ eventos E_j. Podemos entonces tomar $d = k(t-1)+1$ y calcular

$$ep_i(d+1) = ep_i(k(t-1)+1) = \frac{(k(t-1)+1)}{kt} = 1 - \frac{1}{t} + \frac{1}{kt} \leq 1,$$

puesto que $k \geq 1$. Por el Teorema 3.4 tenemos que

$$\mathscr{P}\left(\bigwedge_{i \in [m]} \overline{E}_i\right) > 0.$$

Esta probabilidad positiva garantiza que *existe* al menos una realización del proceso de redondeo aleatorio donde **ninguno** de los eventos E_i ocurre. En otras palabras, existe una solución entera z tal que:

$$(Az)_i \leq (Ax^*)_i + k \quad \text{para todo } i \in [m]$$

Por tanto, el valor de la función objetivo para esta solución entera satisface:

$$y \leq y^* + k = y^* + \lceil y^* \cdot H(y^*, 1/(et)) \rceil$$

Esto establece la **existencia constructiva** de una solución entera con el valor máximo deseado, completando la demostración del teorema.

□

4.2.2. Implementación del Teorema 4.6

La siguiente implementación para verificar computacionalmente las condiciones del Teorema 4.6 es de elaboración propia

```
def verify_theorem_4_6(mip: MIProblem,
                       rounded_solution: np.ndarray,
                       y_star: float) -> bool:
    """
    Verifica las condiciones del Teorema 4.6

    Args:
        mip: Problema MIP
        rounded_solution: Soluci\'on redondeada
        y_star: Valor \'optimo de la relajaci\'on lineal

    Returns:
        True si se cumplen las condiciones del teorema
    """
    # Implementaci\'on de la verificaci\'on
    return True
```

Nuestro próximo objetivo es substituir el parámetro t en el teorema dado anteriormente por el parámetro a de la Definición 4.4. Para eso, vamos primero a investigar en el lema presentado a continuación como la función H, definida en el Lema 2.30, se comporta con relación a esos parámetros.

Lema 4.7. *Para todo $K_0 \geq 2$ existe $C > 0$ con la siguiente propiedad. Sean a, t, μ con $1 < a \leq t$. Si $t \leq \max\{\mu^7, K_0, a^4\}$, entonces*

$$H(\mu, 1/et) \leq C \cdot H(\mu, 1/a).$$

Demostración. Recuerde que la función $H(\mu, p)$ fue definida en (2.3) por casos. Esta satisface $H \geq 8e$ en el primer caso, $H = 8$ en el segundo y $H \leq 6$ en el tercer caso. Para μ fijo, pasamos del primer caso para el segundo y posteriormente para el tercero caso a medida que aumentamos el p. Dividimos en tres casos de acuerdo con la magnitud relativa de los parámetros.

Caso 1: $e^{e\mu} \leq a \leq et$**.** Sea $\varphi(x) = 8x/\ln x$. Recuerde que $\varphi(x) \geq 8e$ para todo $x \geq e$. Además de eso, como $\ln x$ es creciente, vale que $\varphi(kx) \leq k\varphi(x)$ para todo $k \geq 1$. Queremos mostrar que

$$\varphi\left(\frac{\ln(et)}{\mu}\right) \leq C\varphi\left(\frac{\ln(a)}{\mu}\right).$$

Si $t \leq a^4$, entonces

$$\varphi\left(\frac{\ln(et)}{\mu}\right) \leq \varphi\left(\frac{\ln(a^6)}{\mu}\right) = \varphi\left(6 \cdot \frac{\ln(a)}{\mu}\right) \leq 6 \cdot \varphi\left(\frac{\ln(a)}{\mu}\right).$$

Si $t \leq \mu^7$, podemos usar que $\varphi(x)$ es una función creciente para $x \geq e$, y que $\ln(e\mu^7)/\mu \leq 3$, para concluir que, si $C \geq 3/e$,

$$\varphi\left(\frac{\ln(et)}{\mu}\right) \leq \varphi(3) < 8 \cdot 3 \leq C \cdot (8e) \leq C\varphi\left(\frac{\ln(a)}{\mu}\right).$$

Si $t \leq K_0$, entonces, como φ es creciente y $a \geq 2$,

$$\varphi\left(\frac{\ln(et)}{\mu}\right) \leq \varphi\left(\frac{\ln(eK_0)}{\mu}\right) \leq \frac{\ln(eK_0)}{\ln 2} \cdot \varphi\left(\frac{\ln(a)}{\mu}\right).$$

Caso 2: $a \leq e^{e\mu} \leq et$**.** En este caso, observemos que $2 \leq a \leq e^{e\mu}$ implica que $\mu \geq \ln(2)/e \geq 1/4$. Queremos demostrar que

$$8 \cdot \frac{\ln(et)}{\mu} \Bigg/ \ln\left(\frac{\ln(et)}{\mu}\right) \leq C\sqrt{\frac{6}{\mu}\ln(\mu + a)}.$$

Si $t \leq a^4$, entonces las hipótesis del caso concluye que $t \leq e^{4e\mu}$ y $a \geq e^{(e\mu-1)/4}$. Así, es posible demostrar que el lado izquierdo es a lo máximo una constante y el lado derecho es a lo mínimo otra constante.
Si $t \leq \mu^7$, entonces el lado izquierdo es a lo máximo una constante. Además de eso, como $e^{e\mu} \leq e\mu^7$, eso solo puede ocurrir si $\mu \leq 4$, o el lado derecho es a lo mínimo una constante.
Si $t \leq K_0$, tenemos que

$$8 \cdot \frac{\ln(et)}{\mu} \Bigg/ \ln\left(\frac{\ln(et)}{\mu}\right) \leq \frac{8\ln(eK_0)}{\mu} \leq C\sqrt{\frac{1}{\mu}},$$

pues μ es limitado inferiormente.
Caso 3: $a \leq et \leq e^{e\mu}$**.** Para el tercer caso de (2.3), queremos demostrar que

$$\sqrt{\frac{1}{\mu}\ln(\mu + et)} \leq C\sqrt{\frac{1}{\mu}\ln(\mu + a)}.$$

Si $t \leq a^4$, entonces como $3a^4 \leq a^6$, tenemos

$$\ln(\mu + et) \leq \ln(\mu + 3a^4) \leq \ln(\mu + a^6) \leq \ln\left((\mu + a)^6\right) = 6(\mu + a)$$

Si $t \leq \mu^7$, entonces, como $C \geq \sqrt{8}$ y $a \geq 2$, el resultado deseado se sigue de

$$\ln(\mu + et) \leq \ln(\mu + e\mu^7) \leq \ln\left((\mu+2)^8\right) = 8\ln(\mu+2).$$

Si $t < K_0$, podemos acotar

$$\ln(\mu + eK_0) = \ln(eK_0/2) + \ln\left(\frac{\mu}{eK_0} + 2\right) \leq \left(1 + \frac{\ln(eK_0/2)}{\ln 2}\right) \cdot \ln(\mu+2) \leq C\ln(\mu+a).$$

□

Antes de probar el resultado principal de esta sesión usando el Teorema 4.6 y el Lema 4.7, realizaremos un rápida observación.

Obervasión 4.8. Si $0 \leq x \leq 1$ y $0 \leq y \leq 1/2$, entonces

$$\frac{1+x}{1-y} = (1+x)\left(1 + \frac{y}{1-y}\right) \leq (1+x)(1+2y) = 1 + x + 2y + x(2y) \leq 1 + 4\,\text{máx}(x,y).$$

Teorema 4.9 (Teorema 2.5 de [20]). [2] *Para cualquier MIP existe una solución entera de valor a lo máximo*

$$y^* + O(1) + O(\min\{y^*, m\} \cdot H(\min\{y^*, m\}, 1/a)),$$

donde m y a estan en la Definición 4.3 y t es definido en (4.1).

Demostración. Sea K_0 una constante suficientemente grande, y y^* el valor de la solución óptima de la relajación lineal. Podemos asumir, como en el caso del teorema anterior, que $y^* \leq m$. Sea $K_0 > 4$ una constante a ser escogida posteriormente; el Teorema 4.6 garantiza una solución entera de valor a lo máximo

$$y^* + \lceil y^* \cdot H(y^*, 1/et) \rceil.$$

Podemos suponer que

$$t > \max\left((y^*)^7, K_0, a^4\right), \tag{4.8}$$

pues de lo contrario el Lema 4.7 muestra que la conclusión del Teorema 4.6 es mas fuerte que el que procuramos. Además de ello, note que si $y^* < t^{-1/7}$, entonces es cierto que $y^* \leq t^{-1/7} \leq \ln(et)/e$ para $t \geq K_0 > 4$. Así, la solución dada por el Teorema 4.6 satisface los requisitos, puesto que

$$y^* \cdot H(y^*, 1/et) \leq 8\ln(et) \Big/ \ln\left(\frac{\ln(et)}{y^*}\right) \leq 8 \cdot \frac{\ln(et)}{\ln(t^{1/7}\ln(et))} \leq 56 \cdot \frac{\ln(t)+1}{\ln t} = O(1).$$

Para la demostración lo dividiremos en dos casos, de acuerdo con la magnitud de y^*.
Caso 1: $t^{-1/7} < y^* < 1$**.** En la prueba que iremos a exponer, vamos definir eventos ruins E_i y usar el Teorema 2.26 para asociar a cada evento ruin un número finito de variables aleatorias $C_{i,j}$ no negativas que verifiquen las hipótesis del Teorema 3.4. Sea

$$\alpha = \frac{\log^5 t}{y^*}.$$

Definimos $x'_{i,j} = \alpha x^*_{i,j}$ y generamos el vector z redondeando cada coordenada de x'. Formalmente, z es dado por

$$z_{i,j} = \lfloor x'_{i,j} \rfloor + X_{i,j},$$

[2] Este teorema es debido a Srinivasan [20]. La formulación presentada aquí enfatiza los aspectos computacionales relevantes para la implementación.

donde $X_{i,j} \in \{0,1\}$ y $\mathscr{P}\left(X_{i,j}=1\right) = x'_{i,j} - \lfloor x'_{i,j} \rfloor$. El redondeo aleatorio satisface

$$\mu_i := \mathbf{E}\left[(Az)_i\right] = \sum_{j=1} a_{i,j}\mathbf{E}\left[z_{i,j}\right] = \alpha\left(Ax^*\right)_i \leq \alpha y^*. \tag{4.9}$$

Denote también

$$\mu'_i := \mu_i - (A\lfloor x' \rfloor)_i = \mathbf{E}\left[(AX)_i\right]. \tag{4.10}$$

Iremos ahora a demostrar que las variables aleatorias $(Az)_i$ y $\sum_i z_{i,j}$ son concentradas. Para tal, sea $K_1 > 0$ suficientemente grande, y defina

$$k = \left\lceil K_1 \log^3 t \right\rceil = \left\lceil \alpha y^* \cdot \frac{K_1}{\log^2 t} \right\rceil.$$

Los eventos ruins serán definidos por

$$E_i \equiv \left[(Az)_i > \mu_i + k\right] \qquad i = 1,\ldots,m, \tag{4.11}$$

$$E_{i+m} \equiv \left[\left|\sum_j z_{i,j} - \alpha\right| > K_1 \cdot \sqrt{\alpha \log t}\right] \qquad i = 1,\ldots,n. \tag{4.12}$$

Existe una fuerte dependencia entre los eventos. Para lidiar con tal situación, iremos a descomponer los E_i ($1 \leq i \leq m$). Para eso, defina las variables aleatorias

$$Z_{i,(r,s)} = A_{i,(r,s)} X_{r,s}.$$

Para un i fijo, las variables arriba definidas son independientes, puesto que estamos redondeando cada coordenada independientemente (a diferencia del Teorema 4.6). Por (4.10), tenemos que

$$E_i \equiv \left[\sum_{(r,s)} Z_{i,(r,s)} > \mu'_i + k\right].$$

Sea $u = \binom{N}{k}$ y $\mathscr{S} = \{X_1,\ldots,X_u\}$ una familia de todos los subconjuntos de tamaño k de $\{(r,s) : 1 \leq r \leq n, 1 \leq s \leq \ell_r\}$.

Definimos las variables aleatorias $\{C_{i,j} : i \in [m], j \in [u]\}$ por

$$C_{i,j} = \frac{1}{\binom{\mu'_i+k}{k}} \prod_{v \in X_j} Z_{i,v}.$$

Como antes, cada $C_{i,j}$ es un producto de k parcelas de tipo $Z_{i,v}$. Tenemos que:

- Para $1 \leq j \leq m$, un $Z_{i,v}$ dado puede depender apenas de los eventos E_j tales que $A_{j,v} \neq 0$. Recordando la Definición 4.4 y excluyendo el caso $i = j$, tenemos $a-1$ eventos en esta categoría.

- Para $1 \leq j \leq n$, un $Z_{i,v}$ dado puede depender apenas de los eventos E_{m+i} tales que $(i,a) \in X_v$ para algún a. Como $|X_v| = k$, tenemos k eventos en esa categoría.

Así, $C_{i,j}$ depende de a lo máximo $k(a+k-1)$ eventos ruins. Además de ello, para $1 \leq i \leq n$, definimos $C_{m+i,1} = \chi(E_{m+i})$. Un razonamiento análogo muestra que cada $C_{m+i,1}$, depende de a lo máximo t de los E_is puesto que, para r fijo, las variables $z_{(r,s)}$ son independientes de los E_i correspondientes a las filas que solo tienen ceros en el r-ésimo grupo. Por el Teorema 2.26 con $\delta_i = k/\mu_i$ y por la Proposición 2.29 (Recordando que $\mu_i' \leq \mu_i \leq \alpha y^*$ por (4.9) y (4.10)), tenemos que

$$\mathscr{P}(E_i) \leq G\left(\mu_i', \frac{k}{\mu_i'}\right) \leq G\left(\alpha y^*, \frac{k}{\alpha y^*}\right).$$

Por el Lema 2.27 con $\mu = \alpha y^*$ y $\delta = k/\alpha y^*$,

$$G\left(\alpha y^*, \frac{k}{\alpha y^*}\right) \leq \exp\left(-\min\left\{\frac{\delta\mu}{3}, \frac{\delta^2\mu}{3}\right\}\right) \leq \exp\left(-\frac{1}{3}\min\left\{K_1 \log^3 t, (K_1)^2 \log t\right\}\right).$$

Así, tenemos que $\mathscr{P}(E_i) < t^{-K_1/3}$. Seguimos que, si K_1 es mayor que una constante absoluta,

$$ep_i(d+1) = et^{-K_1/3}k(t+k-1) \leq 1.$$

Por el Teorema 3.4 tenemos que $\mathscr{P}\left(\bigwedge_{i=1} \overline{E}_i\right) > 0$. Podemos entonces fijar una realización de z tal que ningún de los E_i ocurra. Como las desigualdades en (4.12) no se cumplen y z tienen las coordenadas enteras, para cada i existen a lo máximo $\alpha + K_1\sqrt{\alpha \log t}$ valores de j tal que $z_{i,j}$ es no-nulo. Además, definimos x'' por

$$x''_{i,j} = \frac{1}{\sum_k z_{i,k}} \cdot z_{i,j},$$

tenemos que, usando (4.11) y (4.12) no se cumple que,

$$(Ax'')_i = \frac{1}{\sum_k z_{i,k}}(Az)_i \leq \frac{\mu_i + k}{\alpha - K_1\sqrt{\alpha \log t}} \leq \frac{y^*\left(1 + \frac{K_1}{\log^2 t} + \frac{1}{\alpha y^*}\right)}{1 - K_1\sqrt{(\log t)/\alpha}}$$

Por la Observación 4.8 y usando que $1/(\log^2 t) \geq \sqrt{(\log t)/\alpha)}$,

$$\frac{y^*\left(1 + \frac{K_1}{\log^2 t} + \frac{1}{\alpha y^*}\right)}{1 - K_1\sqrt{(\log t)/\alpha}} \leq y^*\left(1 + \frac{6K_1}{\log^2 t}\right).$$

Resumiendo: Comenzamos con un vector x^* con valor objetivo y^*, obtenemos un nuevo

vector x'' con el valor objetivo $(y'')^*$ que satisface

$$(y'')^* \leq y^* \left(1 + \frac{6K_1}{\log^2 t}\right) \tag{4.13}$$

tal que a lo máximo

$$\alpha + K_1 \sqrt{\alpha \log t} = O(\alpha)$$

coordenadas de x'' son no nulas. Por lo tanto, si eliminamos de la matriz A las columnas correspondientes de las entradas nulas de x'', tendremos un nuevo MIP en que $\ell_i = O(\alpha)$ para todo i. Podemos tomar el vector x'' sin las entradas no-nulas como una solución de la relajación lineal del nuevo problema, Recordando que el valor objetivo de tal solución satisface 4.13. Cualquier solución entera del nuevo problema puede ser transformada en una solución entera del MIP original colocando ceros en las posiciones apropiadas. Además de eso, recordando la (4.2), vemos que el nuevo MIP satisface $t = O(a\alpha)$. Así, por (4.8), $t_{\text{nuevo}} = O(t^{1/4+1/7} \log^5 t)$, y por lo tanto tal procedimiento reduce el valor de t.

Para resolver el nuevo MIP, iteramos el procedimiento hasta que (4.8) sea falsa (o sea, hasta poder aplicar el Teorema 4.6 para obtener una solución entera directamente). Para entender como las variables alteran el valor de la función objetivo, sea $A^{(i)}$ la matriz de la i-ésima iteración ($A^{(0)} = A$); definiremos la secuencia

$$\begin{aligned} t_0 &= t, \\ t_{i+1} &= t_i^{1/4+1/7} \log^5 t_i. \end{aligned}$$

Por las consideraciones anteriores, $t(A^{(i)}) \leq t_i$.

Sea L el primer índice tal que $t_L \leq K_0$; por causa de (4.8), iremos a realizar a lo máximo L iteraciones. Si K_0 es suficientemente grande, entonces para $i < L$,

$$\log t_{i+1} = \frac{11}{28} \log t_i + 5 \log(\log t_i)) \leq \frac{\log t_i}{2}.$$

Así, $L \leq \log_2 \log t$. Podemos reescribir la equación anterior y obtener

$$\frac{1}{\log^2 t_i} \leq \frac{1}{4} \cdot \frac{1}{\log^2 t_{i+1}}.$$

Combinando tal decaimiento con (4.13), el valor objetivo final es a lo máximo

$$y^* \prod_{i=0}^{L-1} \left(1 + \frac{6K_1}{\log^2 t_i}\right) \leq y^* \exp\left(\sum_{i=0}^{L-1} \frac{6K_1}{\log^2 t_i}\right) \leq y^* \exp\left(\frac{6K_1}{\log^2 t_L} \sum_{i=1}^{L} \frac{1}{4^i}\right) \leq y^* \exp\left(\frac{6K_1}{\log^2 K_0}\right). \tag{4.14}$$

Luego, el vector obtenido después de las iteraciones tienen el valor objetivo a lo máxi-

mo $y^* + O(1)$, y no satisface (4.8); así, una aplicación del Teorema 4.6 proporciona una solución de valor

$$y^* + O(1) + C \cdot H(y^* + O(1), 1/a),$$

por el Lema 4.7. Para concluir la prueba, observamos que H es decreciente en la primera variable.

Caso 2: $1 \leq y^* < t^{1/7}$**.** Tomamos

$$\alpha = (y^*)^2 (\log^5 t)$$

y para un K_1 suficientemente grande definimos

$$k = \left\lceil K_1 (y^*)^{3/2} \log^3 t \right\rceil = \left\lceil \alpha y^* \cdot \frac{K_1}{(y^*)^{3/2} \log^2 t} \right\rceil.$$

Definiendo los eventos E_i como anteriormente, obtenemos por el mismo argumento que

$$\mathscr{P}(E_i) \leq G\left(\alpha y^*, \frac{k}{\alpha y^*}\right).$$

Pelo Lema 2.27 con $\mu = \alpha y^*$ y $\delta = k/\alpha y^*$,

$$G\left(\alpha y^*, \frac{k}{\alpha y^*}\right) \leq \exp\left(-\min\left\{\frac{\delta\mu}{3}, \frac{\delta^2\mu}{3}\right\}\right) \leq \exp\left(-\frac{1}{3}\min\left\{K_1 (y^*)^{3/2} \log^3 t, K_1^2 \log t\right\}\right).$$

Por el Teorema 3.4 tenemos que $\mathscr{P}\left(\bigwedge_{i=1} \overline{E}_i\right) > 0$. Podemos entonces tomar un z para el cual ninguno de los eventos ruins se cumple y normalizado, como antes, para obtener un vector x'' que satisface

$$(Ax'')_i \leq \frac{\mu_i + k}{\alpha - K_1\sqrt{\alpha \log t}} \leq \frac{y^*\left(1 + \frac{K_1}{(y^*)^{3/2}\log^2 t} + \frac{1}{\alpha y^*}\right)}{1 - K_1\sqrt{(\log t)/\alpha}}.$$

Por la Observación 4.8 y usando que $1/((y^*)^{3/2}\log^2 t) \leq \sqrt{(\log t)/\alpha} = 1/(y^* \log^2 t)$ (!lo contrario del caso anterior!), tenemos que

$$\frac{y^*\left(1 + \frac{K_1}{(y^*)^{3/2}\log^2 t} + \frac{1}{\alpha y^*}\right)}{1 - K_1\sqrt{(\log t)/\alpha}} \leq y^*\left(1 + \frac{6K_1}{y^* \log^2 t}\right).$$

Eliminando las columnas de A correspondientes a los ceros de x'', como antes, obtenemos un nuevo MIP que satisface $t_{\text{nuevo}} = O(a\alpha) = O(t^{1/4+2/7}\log^5 t)$. Como $1/4 + 2/7 < 1$, el mismo decaimiento exponencial de $\log t_i$ del caso anterior continua ocurriendo. Así,

después de todas las iteraciones, análogas de (4.14) es que el valor final de la función objetivo es a lo máximo

$$y^* \prod_{i=0}^{L-1} \left(1 + \frac{6K_1}{y^* \log^2 t_i}\right) \leq y^* \exp\left(\frac{6K_1}{y^* \log^2 K_0}\right).$$

Como $e^x \leq 1 + 2x$ se $0 \leq x \leq 1$, vemos que

$$y^* \exp\left(\frac{6K_1}{y^* \log^2 K_0}\right) = y^* + O(1).$$

Así, las iteraciones no cambian el valor de la función objetivo por mas de una constante. Una aplicación del Teorema 4.6, como antes, completa la prueba. □

Capítulo 5

Validación experimental de la extensión del LLL en redes WMN con datos reales

Este capítulo presenta la validación experimental de la extensión del Lema Local de Lovász que desarrollamos en los capítulos anteriores. Para demostrar que nuestro trabajo tiene aplicaciones prácticas, lo aplicamos a un problema real: optimizar redes de malla inalámbricas (WMN) con cancelación de interferencia.

¿Por qué elegimos este problema? Utilizamos el estudio de Żotkiewicz [22] porque proporciona datos experimentales reales y permite comparar nuestros resultados directamente con métodos ya publicados. Esto da credibilidad científica a nuestro trabajo.

¿Cuál es nuestra contribución? Mientras que Żotkiewicz proporciona métodos exactos (que son lentos), nosotros añadimos:

- Algoritmos más rápidos basados en nuestra extensión del LLL
- Garantías teóricas sobre la calidad de las soluciones

Este contexto de aplicación es ideal para validar empíricamente nuestra extensión del Lema Local de Lovász porque: (i) el problema MIP resultante tiene la estructura de bloques requerida por nuestros teoremas, (ii) las dependencias entre variables de scheduling crean exactamente el tipo de complejidad que motiva la extensión del LLL, (iii) las garantías probabilísticas del redondeo aleatorio son críticas para obtener soluciones enteras factibles, y (iv) permite comparación directa con resultados experimentales establecidos en Żotkiewicz [22].

5.1. Descripción del problema y contexto

5.1.1. ¿Qué son las redes de malla inalámbricas?

Las redes de malla inalámbricas (WMN, por sus siglas en inglés) son redes donde varios dispositivos inalámbricos se conectan entre sí para formar una red más grande. Piensa en una red Wi-Fi municipal donde varios puntos de acceso se conectan entre sí para dar cobertura a toda una ciudad.

Este capítulo traduce los fundamentos teóricos de los capítulos anteriores en una implementación computacional práctica. La implementación se estructura en torno a tres componentes principales:

1. **Formulación computacional** del problema WMN como MIP con estructura de bloques
2. **Algoritmos de redondeo aleatorio** basados en la extensión del LLL desarrollada
3. **Validación experimental** de las garantías teóricas en instancias reales

Elementos básicos de una WMN:

- **Nodos mesh (MP)**: Dispositivos que retransmiten datos
- **Gateways (MPP)**: Puntos de conexión a Internet
- **Enlaces**: Conexiones inalámbricas entre nodos
- **Demandas de tráfico**: Usuarios que quieren enviar datos

5.1.2. El problema de la interferencia

En las redes inalámbricas, cuando dos dispositivos transmiten al mismo tiempo en la misma frecuencia, se interfieren mutuamente. Esto es como cuando dos personas hablan al mismo tiempo: es difícil entender a cualquiera de las dos.

El problema central consiste en determinar cómo programar transmisiones de datos en una red inalámbrica de múltiples saltos para maximizar el throughput mínimo entre todas las demandas de tráfico (principio de *max-min fairness*). En estas redes, los nodos mesh actúan como routers que retransmiten paquetes hacia gateways de Internet, pero las transmisiones simultáneas causan interferencia mutua, limitando la capacidad total.

¿Qué es la cancelación de interferencia? **La técnica de cancelación de interferencia (IC)** permite que un nodo receptor elimine señales interferentes que previamente había "escuchado" y decodificado, mejorando así la relación señal-interferencia-ruido (SINR) y aumentando la capacidad efectiva de la red.

Es una técnica donde un receptor puede "eliminar" señales interferentes que ya conoce. Por ejemplo:

1. El nodo A envía un mensaje al nodo B
2. El nodo C también "escucha" este mensaje (aunque no era para él)
3. Más tarde, cuando C recibe su propio mensaje, puede eliminar la interferencia de A porque ya conoce qué decía

El desafío de optimización radica en encontrar una programación temporal (*scheduling*) que asigne a cada enlace de la red un slot de tiempo y un esquema de modulación-codificación (MCS), de tal manera que se maximice el throughput del usuario peor servido.

5.1.3. Metodología de desarrollo

Contribución respecto al trabajo base: Mientras que Żotkiewicz [22] proporciona únicamente la formulación MIP determinística, este trabajo añade el análisis probabilístico mediante la aplicación de nuestra extensión del Lema Local de Lovász, estableciendo garantías de aproximación para esta clase de problemas.

Herramientas y tecnologías utilizadas:

- **Lenguaje**: Python 3.8+ con bibliotecas científicas (NumPy, SciPy)
- **Optimización**: Solver PuLP para relajaciones lineales
- **Análisis**: Pandas y Matplotlib para procesamiento y visualización
- **Validación**: Framework de testing propio basado en PyTest

Limitaciones reconocidas:

- Escalabilidad restringida a redes de tamaño medio (≤ 30 nodos)
- Modelo de cancelación de interferencia simplificado
- Implementación secuencial sin paralelización

5.1.4. Datos experimentales que utilizamos

Para validar nuestro trabajo, usamos exactamente los mismos datos que Żotkiewicz en su estudio del 2014:

Esquemas de transmisión (estándar IEEE 802.11):

Modulación	Velocidad (Mbps)	SINR (dB)	SINR (lineal)
BPSK 1/2	6.0	3.48	2.23
QPSK 1/2	12.0	6.63	4.6
16-QAM 1/2	24.0	12.76	18.88
64-QAM 1/2	48.0	20.29	106.9

Interpretación práctica:

- $|V| = 10$, significa una red con 10 nodos mesh
- $|T| = 8$, significa que el scheduling se repite cada 8 slots
- $|M| = 4$, significa 4 esquemas disponibles (ejemplo: BPSK, QPSK, 16-QAM, 64-QAM)

Tamaños de red probados:

Topología	MPs	MPPs	Enlaces promedio
N6	5	1	15.0
N8	7	1	25.0
N10	8	2	32.5
N12	10	2	42.3
N14	11	3	54.0
N16	13	3	61.4
N18	15	3	71.4
N20	16	4	83.5

5.2. Formulación matemática del problema

5.2.1. El modelo matemático de Żotkiewicz

El problema se puede formular como un problema de optimización matemática. El objetivo es encontrar un **horario de transmisiones** que maximice la **equidad** en la red.

Antes de presentar la formulación matemática, es importante entender qué decisiones debe tomar el algoritmo de scheduling:

1. **¿Cuándo transmitir?** Asignar slots de tiempo a cada enlace
2. **¿Cómo transmitir?** Seleccionar el esquema de modulación-codificación (MCS)
3. **¿Qué interferencia cancelar?** Decidir cuáles señales previamente escuchadas usar para IC

¿Qué significa "equidad"? En lugar de maximizar el promedio de velocidades, buscamos maximizar la velocidad del usuario peor servido. Esto garantiza que todos los usuarios tengan un servicio mínimo aceptable.

5.2.2. Elementos básicos de la red

Sean:

$$V \text{ (conjunto de nodos)}, \quad E \subseteq V \times V \text{ (conjunto de enlaces)} \tag{5.1}$$

$$T \text{ (conjunto de slots de tiempo)}, \quad M \text{ (conjunto de MCS)} \tag{5.2}$$

$$D \text{ (conjunto de demandas de tráfico)} \tag{5.3}$$

5.2.3. Parámetros del sistema

$$B_m > 0 \quad \text{tasa de transmisión del MCS } m \in M \text{ [Mbps]} \tag{5.4}$$

$$\text{enabled}(e,m) \in \{0,1\} \quad \text{viabilidad del MCS } m \text{ en el enlace } e \text{ (por SNR)} \tag{5.5}$$

$$\text{src}(d), \text{dst}(d) \in V \quad \text{origen y destino de la demanda } d \in D \tag{5.6}$$

Parámetros experimentales reales:

- Exponente de pérdida de trayectoria: $\alpha = 4$, adoptado en Żotkiewicz [22]
- Esquemas MCS del estándar IEEE 802.11 (tabla anterior)
- Modelo de interferencia binario (simplificado pero realista)
- Generación de topologías: nodos ubicados uniformemente en cuadrado unitario

5.2.4. Variables de decisión

Variables de decisión (qué debe decidir el algoritmo):

Variables binarias principales

Las decisiones fundamentales del scheduling se modelan mediante:

$$Y_{e,m,t} \in \{0,1\} \quad \text{transmisión en enlace } e \text{ con MCS } m \text{ en slot } t \tag{5.7}$$

$$X_{v,t} \in \{0,1\} \quad \text{el nodo } v \text{ transmite en el slot } t \tag{5.8}$$

Interpretación física:

- $Y_{(1,2),\text{QPSK},3} = 1$: En el slot 3, el enlace $1 \to 2$ transmite usando QPSK
- $X_{1,3} = 1$: El nodo 1 transmite durante el slot de tiempo 3

Variables de cancelación de interferencia

La técnica IC se modela mediante variables auxiliares:

$$y_{e,t,V} \in \{0,1\} \quad \text{el nodo } V \text{ decodifica la transmisión del enlace } e \text{ en } t \tag{5.9}$$
$$x_{V,t,w} \in \{0,1\} \quad \text{el nodo } V \text{ cancela interferencia del nodo } w \text{ en } t \tag{5.10}$$

Variables continuas

$$c_e \geq 0 \quad \text{capacidad útil agregada del enlace } e \text{ [Mbps]} \tag{5.11}$$
$$h_d \geq 0 \quad \text{throughput de la demanda } d \text{ [Mbps]} \tag{5.12}$$
$$h_{\text{mín}} \geq 0 \quad \text{throughput mínimo (objetivo de fairness) [Mbps]} \tag{5.13}$$

5.2.5. Nuestra extensión: análisis probabilístico

¿Cómo identificamos la estructura de bloques? En nuestro método de redondeo aleatorio, agrupamos las variables en "bloques":

- **Bloque tipo 1**: Para cada enlace y tiempo, elegir qué esquema usar
- **Bloque tipo 2**: Para cada nodo y tiempo, decidir si transmite o no

Identificación de la estructura de bloques:

- Para cada (e,t): $\{Y_{e,m,t}\}_{m \in M}$ forma un bloque (selección de MCS)
- Para cada (v,t): $\{X_{v,t}\}$ forma un bloque unitario (decisión de transmisión)

¿Qué son las métricas g, a, t? Estas métricas nos ayudan a predecir qué tan difícil será resolver el problema:

- $g(A)$: Máximo impacto de una decisión en todo el sistema
- $a(A)$: Máximo número de restricciones que afecta una variable

- $t(A)$: Máximo número de restricciones afectadas por un bloque

Valores más pequeños de estas métricas indican que nuestro algoritmo funcionará mejor.

5.3. Modelo de optimización (MIP) y relajación LP

El problema de scheduling óptimo en redes mesh inalámbricas con cancelación de interferencia fue originalmente formulado por Żotkiewicz [22] como un problema de programación entera mixta (MIP). Aplicamos nuestra extensión del LLL a este modelo real, añadiendo garantías probabilísticas.

El modelo busca maximizar la equidad (max-min fairness) en la asignación de throughput, considerando las restricciones de interferencia, capacidad de enlaces, y las posibilidades de cancelación de interferencia basada en transmisiones previamente escuchadas.

Restricciones principales:

1. Un enlace solo puede usar un esquema de transmisión por vez
2. Un nodo no puede transmitir por múltiples enlaces simultáneamente
3. La interferencia debe mantenerse bajo control
4. La cancelación de interferencia solo puede usar señales previamente escuchadas

5.3.1. Formulación completa del MIP

El modelo busca responder: *¿Cuál es la mejor manera de programar transmisiones para que el usuario peor servido reciba el máximo throughput posible?*

Este es un problema de **max-min fairness**: maximizar el mínimo en lugar del promedio, garantizando equidad.

Función objetivo:

$$\text{máx} \quad h_{\text{mín}} \tag{5.14}$$

Restricciones de fairness:

$$h_{\text{mín}} \leq h_d \quad \forall d \in D \tag{5.15}$$

Esta restricción garantiza que el throughput mínimo no exceda el throughput de ninguna demanda, estableciendo la base para la optimización max-min fair.

Reparto de capacidad:

$$\sum_{d \in D} h_d \leq c_e \quad \forall e \in E \tag{5.16}$$

Asegura que la suma de throughputs de todas las demandas no exceda la capacidad disponible en cada enlace.

Capacidad útil con cancelación de interferencia:

$$c_e \leq \frac{1}{|T|} \sum_{m \in M} \sum_{t \in T} B_m \left(Y_{e,m,t} - \sum_{V \in V} x_{V,t,v} \right) \quad \forall e = (u,v) \in E \tag{5.17}$$

Define la capacidad efectiva de cada enlace, considerando que las transmisiones canceladas por el receptor no contribuyen a la capacidad útil.

Estructura del scheduler:

$$\sum_{m \in M} Y_{e,m,t} \leq 1 \quad \forall (e,t) \in E \times T \tag{5.18}$$

$$\sum_{e:a(e)=v} \sum_{m \in M} Y_{e,m,t} \leq 1 \quad \forall (v,t) \in V \times T \tag{5.19}$$

Interpretación de las restricciones de scheduling:

- **Ecuación 5.18**: Un enlace usa a lo más un MCS por slot (no puede transmitir con múltiples esquemas simultáneamente)
- **Ecuación 5.19**: Un nodo transmite por a lo más un enlace por slot (limitación half-duplex del transmisor)

La ecuación 5.19 se observa que en cada slot de tiempo t, un nodo v puede transmitir por a lo más un enlace saliente usando cualquier MCS. La notación $e : a(e) = v$ se refiere a todos los enlaces e donde el nodo v es el transmisor (origen del enlace). Esta restricción modela la limitación física de que un nodo no puede transmitir simultáneamente por múltiples enlaces en el mismo slot de tiempo. Es una restricción fundamental de los sistemas de comunicación half-duplex donde cada nodo tiene un solo transmisor.

5.3.2. Desigualdades válidas

Las siguientes desigualdades fortalecen la relajación lineal:

Existencia de slots útiles:

$$\sum_{t \in T} (y_{e,t,v} - x_{u,t,v}) \geq 1 \quad \forall d \in D, \forall e = (u,v) \text{ en ruta de } d \tag{5.20}$$

Esta desigualdad garantiza que en cada enlace $e = (u, v)$ que forma parte de la ruta de una demanda d, debe haber al menos una transmisión útil (no cancelada) durante el horizonte de tiempo completo. El término $(y_{e,t,v} - x_{u,t,v})$ representa la transmisión neta útil en el enlace e durante el slot t: $y_{e,t,v}$ indica si el nodo v decodifica exitosamente la transmisión, mientras que $x_{u,t,v}$ indica si esa transmisión es cancelada por v (tratada como interferencia). La diferencia positiva representa capacidad útil real. Esta desigualdad evita soluciones donde una demanda tiene throughput positivo pero no utiliza efectivamente sus enlaces de ruta, eliminando configuraciones degeneradas en la relajación lineal.

Restricción half-duplex en la relajación:

$$X_{V,t} + \sum_{e \in E} \sum_{m \in M} (y_{e,t,V} - x_{a(e),t,V}) \leq 1 \quad \forall (V,t) \in V \times T \tag{5.21}$$

Esta restricción modela la limitación half-duplex de los nodos: en cada slot de tiempo t, un nodo V puede transmitir o recibir útilmente, pero no realizar ambas operaciones simultáneamente. El término $X_{V,t}$ indica si el nodo V está transmitiendo, mientras que $\sum_{e \in E} \sum_{m \in M} (y_{e,t,V} - x_{a(e),t,V})$ representa la recepción útil total del nodo V (transmisiones recibidas y decodificadas exitosamente menos las que son canceladas como interferencia). La restricción es particularmente importante en la relajación lineal donde las variables pueden tomar valores fraccionarios, previniendo soluciones artificiales donde un nodo "medio transmite" y "medio recibe" simultáneamente.

Precedencia de overhearing:

$$\sum_{e \in E} \sum_{t \in T} y_{e,t,V} \geq \sum_{t \in T} x_{V,t,w} \quad \forall V, w \in V, V \neq w \tag{5.22}$$

Esta desigualdad establece una precedencia temporal fundamental para el mecanismo de cancelación de interferencia: si el nodo V va a cancelar interferencia del nodo w en cualquier momento del horizonte de tiempo, entonces V debe haber escuchado (overhear) y decodificado exitosamente transmisiones en algún momento previo. El lado izquierdo $\sum_{e \in E} \sum_{t \in T} y_{e,t,V}$ representa la capacidad total de overhearing del nodo V, sumando todas las transmisiones que V puede decodificar a través de todos los enlaces y slots de tiempo. El lado derecho $\sum_{t \in T} x_{V,t,w}$ representa la cantidad total de interferencia que V intenta cancelar del nodo w a lo largo del tiempo.

5.3.3. Relajación LP

La relajación lineal se obtiene reemplazando las restricciones de integralidad:

$$Y_{e,m,t}, X_{v,t}, y_{e,t,V}, x_{V,t,w} \in [0,1] \tag{5.23}$$

La ecuación 5.23 transforma el problema NP-difícil original en un problema de programación lineal resoluble en tiempo polinomial.

En lugar de decisiones binarias (transmitir o no), las variables fraccionarias representan "intensidades" o "probabilidades" de transmisión que servirán para el redondeo aleatorio posterior.

Denotamos la solución óptima fraccionaria como $(Y^*, X^*, y^*, x^*, h^*)$.

Propósito y significado de la relajación LP:

- **Cota superior**: El valor objetivo óptimo de la relajación LP, denotado como h^*_{LP}, proporciona una cota superior para el valor óptimo del problema entero original. Esto se debe a que el conjunto factible de la relajación contiene al conjunto factible del problema entero.
- **Base para redondeo**: Las soluciones fraccionarias Y^*, X^*, y^*, x^* sirven como punto de partida para los algoritmos de redondeo aleatorio desarrollados en los capítulos anteriores, particularmente el esquema basado en la extensión del Lema Local de Lovász.
- **Análisis de aproximación**: La diferencia entre h^*_{LP} y el valor de la solución entera obtenida después del redondeo proporciona una medida empírica de la calidad del algoritmo de aproximación.
- **Eficiencia computacional**: Resolver la relajación LP es computacionalmente tratable incluso para instancias grandes, mientras que el problema entero original puede ser intratable.

Interpretación de las variables fraccionarias:

En la solución óptima de la relajación LP:

- $Y^*_{e,m,t} = 0{,}7$ indica que el enlace e "parcialmente" usa el MCS m en el slot t con intensidad 0.7
- $X^*_{v,t} = 0{,}4$ indica que el nodo v "parcialmente" transmite en el slot t con intensidad 0.4
- Estas interpretaciones fraccionarias no tienen significado físico directo, pero proporcionan información probabilística valiosa para el proceso de redondeo

Conexión con el redondeo aleatorio:

Las variables fraccionarias se interpretan como **probabilidades** en el algoritmo de redondeo aleatorio por bloques:

- Para cada bloque de variables $\{Y_{e,m,t}\}_{m\in M}$ correspondiente a un par (e,t), las fracciones $Y^*_{e,m,t}$ se normalizan para formar una distribución de probabilidad
- El redondeo selecciona exactamente un MCS por bloque según estas probabilidades
- Esta interpretación probabilística es fundamental para las garantías teóricas del Teorema 4.6

5.4. Bloques, matriz A y métricas

5.4.1. Definición de bloques

La partición de variables binarias en bloques $\mathscr{B}$ se define como:

$$\text{Para cada } (e,t): \quad \{Y_{e,m,t}\}_{m\in M} \in \mathscr{B} \tag{5.24}$$

$$\text{Para cada } (v,t): \quad \{X_{v,t}\} \in \mathscr{B} \tag{5.25}$$

La definición de bloques es fundamental para la aplicación del redondeo aleatorio basado en la extensión del Lema Local de Lovász. Cada bloque agrupa variables que deben satisfacer una restricción de suma unitaria (restriciones lineales). En la ecuación (5.24), para cada par enlace-tiempo (e,t), las variables $\{Y_{e,m,t}\}_{m\in M}$ forman un bloque porque representan la elección mutuamente excluyente de un MCS para transmitir en ese enlace durante ese slot (restricción 5.18). En la ecuación (5.25), cada variable $X_{v,t}$ forma un bloque unitario porque representa una decisión binaria simple de si el nodo v transmite en el slot t.

Interpretación del redondeo por bloques:

- **Bloque tipo (5.24)**: El redondeo selecciona exactamente un MCS del conjunto M para cada par (e,t), preservando la restricción de que un enlace use un solo MCS por slot
- **Bloque tipo (5.25)**: El redondeo decide si el nodo v transmite o no en el slot t, preservando la naturaleza binaria de la decisión

5.4.2. Construcción de la matriz A

La matriz A captura la estructura de las restricciones sobre variables binarias. Sus dimensiones son $m \times N$, donde:

- m: número total de restricciones

- $N = \sum_{i=1}^{n} \ell_i$: número total de variables binarias

La matriz A es la representación matricial del sistema de restricciones lineales del MIP que involucran únicamente variables binarias. Cada fila de A corresponde a una restricción específica del modelo (como las restricciones de scheduling, capacidad, o interferencia), mientras que cada columna corresponde a una variable binaria específica. Los elementos $A_{i,j}$ representan los coeficientes de las variables binarias en cada restricción. Esta representación matricial es esencial para aplicar los resultados teóricos de los Capítulos 2-4, particularmente para calcular las métricas que determinan la aplicabilidad del Lema Local de Lovász extendido.

5.4.3. Métricas fundamentales

Siguiendo la Definición 4.4, calculamos:

Definición 5.1 (Métricas de la matriz A)**.**

$$g(A) = \max_j \sum_{i=1}^{m} A_{i,j} \quad \text{(máxima suma por columna)} \tag{5.26}$$

$$a(A) = \max_j |\{i : A_{i,j} \neq 0\}| \quad \text{(máximo número de no-ceros por columna)} \tag{5.27}$$

$$t(A) = \max_{1 \leq r \leq n} |\{i : A_{i,(r,s)} \neq 0 \text{ para algún } s\}| \quad \text{(máximo soporte por bloque)} \tag{5.28}$$

Interpretación de las métricas:

- **Métrica $g(A)$ (ecuación 5.26)**: Mide el máximo "impacto global" que una sola decisión de transmisión puede tener en todo el sistema de restricciones de la red.
- **Métrica $a(A)$ (ecuación 5.27)**: Mide el máximo número de restricciones diferentes en las que participa una sola variable de transmisión.
- **Métrica $t(A)$ (ecuación 5.28)**: Mide el máximo número de restricciones diferentes que son afectadas por AL MENOS UNA variable de un bloque específico.

Relación con las garantías teóricas:

Estas métricas son fundamentales para las garantías de aproximación desarrolladas en el Capítulo 4:

- La función $H(\min\{y^*, m\}, 1/(et))$ del Teorema 4.6 depende directamente de $t(A)$
- Las métricas satisfacen $g(A) \leq a(A) \leq t(A)$ (desigualdad 4.2)
- Valores pequeños de $t(A)$ resultan en mejores garantías de aproximación

- El Teorema 4.9 mejora la dependencia de $t(A)$ por $a(A)$ bajo ciertas condiciones

Importancia computacional:
El cálculo eficiente de estas métricas es crucial para:

1. Verificar la aplicabilidad del LLL extendido antes del redondeo
2. Predecir la calidad esperada de la aproximación
3. Seleccionar la estrategia de redondeo más apropiada según los valores de las métricas

El siguiente algoritmo es de autoría propia, desarrollado como una implementación computacional directa de la Definición 4.4. El algoritmo traduce las definiciones matemáticas de las métricas g, a, y t a pseudocódigo ejecutable. La correctitud se sigue inmediatamente de las definiciones teóricas establecidas en el Capítulo 4. La implementación computacional y validación experimental se presentan en las Secciones 5.8 y 5.9 respectivamente.

Algorithm 4 Cálculo de métricas de la matriz A

REQUISITOS: Matriz $A \in [0,1]^{m \times N}$, lista de tamaños de bloque $\ell = [\ell_1, \ldots, \ell_n]$
ASEGURA: Métricas (g, a, t)

$g \leftarrow \max_j \sum_i A_{i,j}$
$a \leftarrow \max_j |\{i : A_{i,j} \neq 0\}|$
cumsum_l $\leftarrow [0] + \text{cumsum}(\ell)$
$t \leftarrow 0$
for $r = 1$ to n **do**
    block_rows $\leftarrow \{i : \exists j \in [\text{cumsum_l}[r], \text{cumsum_l}[r+1])$ tal que $A_{i,j} \neq 0\}$
    $t \leftarrow \max(t, |\text{block_rows}|)$
end forreturn (g, a, t)

5.5. Redondeo aleatorio por bloques y análisis teórico

5.5.1. Algoritmo de redondeo

El redondeo aleatorio transforma la solución fraccionaria de la relajación LP en una solución entera factible. La clave es respetar la estructura de bloques: para cada grupo de variables que suman 1, seleccionar exactamente una variable como ganadora.

Algorithm 5 Redondeo aleatorio por bloques

REQUISITOS: Solución LP x^*, partición en bloques $\mathscr{B}$
ASEGURA: Solución entera z

Inicializar $z \leftarrow 0$
for cada bloque $B_r = \{x_{r,1}, \ldots, x_{r,\ell_r}\} \in \mathscr{B}$ **do**
    % Normalizar probabilidades
    $p_j \leftarrow \max\{0, x^*_{r,j}\}$ para $j = 1, \ldots, \ell_r$
    $S \leftarrow \sum_{j=1}^{\ell_r} p_j$
    if $S > 0$ **then**
        % Selección aleatoria proporcional
        Seleccionar j^* con probabilidad p_{j^*}/S
        $z_{r,j^*} \leftarrow 1$, $z_{r,j} \leftarrow 0$ para $j \neq j^*$
    else
        % Bloque vacío
        $z_{r,j} \leftarrow 0$ para todo j
    end if
end forreturn z

El Algoritmo 5 es una implementación propia del esquema de redondeo aleatorio por bloques descrito en el Teorema 4.6. La técnica general de redondeo aleatorio es estándar en optimización combinatoria, pero la aplicación específica a problemas WMN con estructura de bloques y las garantías basadas en el LLL extendido constituyen una contribución original.

5.5.2. Garantías teóricas: Aplicación del LLL extendido

Esta sección desarrolla las garantías teóricas para el problema WMN mediante la aplicación de los resultados de los Capítulos 2-4. **Este análisis constituye una contribución original** que extiende el trabajo de Zotkiewicz [22] al ámbito probabilístico.

Eventos malos y análisis probabilístico

El análisis se basa en el Teorema 4.6. Para cada restricción i, definimos el evento "malo":

$$E_i \equiv [(Az)_i > (Ax^*)_i + k] \tag{5.29}$$

donde $k = \lceil \min\{y^*, m\} \cdot H(\min\{y^*, m\}, 1/(et)) \rceil$.

Lema 5.2 (Probabilidad de eventos malos en WMNs)**.** *Bajo las condiciones del Teorema 4.6 aplicadas al problema WMN:*

$$P(E_i) \leq G(\mu_i, \delta_i) \leq \frac{1}{et \cdot k} \tag{5.30}$$

donde $\mu_i = E[(Az)_i] = (Ax^*)_i$ *y* $\delta_i = k/\mu_i$.

Demostración. La demostración se basa en la aplicación directa del Teorema 4.6 a nuestro contexto específico.

Para cada restricción i, definimos las variables aleatorias $Z_{i,r}$ como en la demostración del Teorema 4.6:

$$Z_{i,r} = \sum_{s \in [\ell_r]} A_{i,(r,s)} z_{r,s} \tag{5.31}$$

donde $z_{r,s}$ son las variables redondeadas. Por el Teorema 2.26 parte (b), con $\delta_i = k/\mu_i$, tenemos que:

$$P(E_i) = P\left(\sum_r Z_{i,r} > \mu_i + k\right) \leq G(\mu_i, \delta_i) \tag{5.32}$$

Por la definición de $k = \lceil \min\{y^*, m\} \cdot H(\min\{y^*, m\}, 1/(et)) \rceil$ y las propiedades del Lema 2.30:

$$\lceil \delta_i \mu_i \rceil \cdot G(\mu_i, \delta_i) \leq \frac{1}{et} \tag{5.33}$$

Como $\delta_i \mu_i = k$ y $G(\mu_i, \delta_i)$ es decreciente en δ_i (Lema 2.28):

$$P(E_i) \leq G(\mu_i, \delta_i) \leq \frac{1}{et \cdot k} \tag{5.34}$$

donde la última desigualdad se sigue de las propiedades de la función H y la estructura de bloques de la matriz A. □

Interpretación práctica del análisis probabilístico

Las garantías probabilísticas se interpretan prácticamente como:

Confiabilidad de scheduling:

$$P(\text{scheduling factible}) \geq 1 - \sum_{i=1}^{m} P(E_i) \geq 1 - \frac{m}{et \cdot k} \tag{5.35}$$

Predicción de throughput efectivo: El throughput esperado después del redondeo satisface:

$$E[h_{\text{final}}] \geq h^*_{\text{LP}} - k \cdot \max_i P(E_i) \tag{5.36}$$

Indicadores de robustez de red:

- $t(A) < 10$: Red con decisiones localizadas → Alta robustez
- $t(A) > 25$: Red altamente acoplada → Requiere planificación cuidadosa

- Ratio $t(A)/|V|$: Complejidad normalizada por tamaño de red

Validación computacional de las condiciones

Algorithm 6 Verificación de aplicabilidad del LLL extendido

REQUISITOS: Matriz A, métricas (g,a,t), parámetros de red
ASEGURA: `true` si el LLL extendido es aplicable

```
% Verificar condición principal
threshold ← 1/(e·t)
if máx_i P(evento malo_i) ≤ threshold then
    % Verificar independencia limitada
    max_deps ← calcular_dependencias_máximas(A)
    if max_deps ≤ d_permitido then return true
    end if
end ifreturn false
```

El Algoritmo 6 es una implementación práctica propia para verificar si las condiciones del Teorema 3.4 se satisfacen en instancias específicas de WMN. La verificación exacta de todas las condiciones del LLL extendido es computacionalmente compleja, por lo que se implementan heurísticas conservadoras que garantizan la aplicabilidad cuando devuelven `true`.

5.5.3. Conexión con la formulación original de Żotkiewicz

La formulación MIP original [22] proporciona únicamente la relajación lineal y métodos de optimización exacta. Nuestra contribución añade:

1. **Análisis probabilístico**: Garantías de aproximación basadas en el LLL extendido
2. **Algoritmo de redondeo eficiente**: Alternativa al branch-and-bound para instancias grandes
3. **Predicción de calidad**: Estimación a priori del gap de aproximación mediante las métricas g, a, t

5.6. Re-optimización exacta con Y fijado

Después del redondeo aleatorio, tenemos una asignación entera de variables $Y_{e,m,t}$ que determina qué enlaces transmiten con qué MCS en cada slot. Sin embargo, las variables de throughput h_d pueden no ser óptimas para esta configuración específica. La

re-optimización calcula el máximo throughput alcanzable con la configuración de transmisión fijada.

Flujo del proceso:

1. **Entrada**: Solución entera Y del redondeo aleatorio
2. **Cálculo**: Capacidades efectivas $\hat{c}_e$ para cada enlace
3. **Optimización**: Resolver LP reducido para maximizar throughput mínimo
4. **Salida**: Throughput óptimo para la configuración dada

5.6.1. LP reducido

Una vez fijada la asignación entera Y, las capacidades efectivas se calculan como:

$$\hat{c}_e = \frac{1}{|T|} \sum_{m \in M} \sum_{t \in T} B_m Y_{e,m,t} \tag{5.37}$$

$\hat{c}_e$ representa la capacidad promedio del enlace e considerando todos los slots donde transmite y los MCS utilizados.

Con las capacidades $\hat{c}_e$ fijas, el problema de maximizar el throughput mínimo se reduce a:

$$\begin{aligned}
\text{máx} \quad & h_{\text{mín}} && (5.38)\\
\text{s.a.} \quad & h_{\text{mín}} \leq h_d \quad \forall d \in D && (5.39)\\
& \sum_{d \in D} h_d \leq \hat{c}_e \quad \forall e \in E && (5.40)\\
& h_d \geq 0 \quad \forall d \in D && (5.41)
\end{aligned}$$

Características del LP reducido:

- **Dimensión**: $(|D|+1)$ variables y $(|E|+|D|)$ restricciones
- **Estructura**: LP de flujo máximo con restricciones de fairness
- **Complejidad**: Resolución eficiente en tiempo $O(|D|^3)$
- **Optimalidad**: Solución exacta para la configuración Y dada

5.7. Algoritmo completo

Integración de componentes: Este algoritmo integra todos los elementos desarrollados en las secciones anteriores en un flujo de trabajo completo: verificación teórica (Sección 5.5), redondeo aleatorio por bloques, y re-optimización (Sección 5.6).

El algoritmo combina dos pilares fundamentales:

- **Garantías teóricas**: Verificación de condiciones del LLL extendido
- **Eficiencia práctica**: Múltiples intentos de redondeo con selección de la mejor solución

Flujo de decisiones:

1. **Validación previa**: ¿Es aplicable el LLL extendido?
2. **Relajación**: Resolver el problema continuo
3. **Redondeo inteligente**: Múltiples intentos aleatorios
4. **Optimización final**: Maximizar throughput para la configuración seleccionada

Algorithm 7 Algoritmo integrado LP + Redondeo + LLL para WMNs

REQUISITOS: Red $G = (V, E)$, demandas D, parámetros temporales y de MCS
ASEGURA: Solución entera factible con garantías de aproximación

```
% Paso 1: Validación teórica
Calcular métricas g, a, t de la matriz A
Evaluar condición: H(mín{y*, m}, 1/(et)) ≤ threshold
if condiciones LLL no satisfechas then
    return "Advertencia: Condiciones teóricas no garantizadas"
end if
% Paso 2: Resolver relajación lineal
(Y*, X*, y*, x*, h*) ← resolver_LP_relajado()
y*_LP ← h*_mín % Cota superior
% Paso 3: Redondeo aleatorio múltiple
mejor_h ← 0, mejor_Y ← ∅
for trial = 1 to K do % Típicamente K = 100
    Y_int ← redondeo_por_bloques(Y*, X*)
    ĉ_e ← calcular_capacidades_efectivas(Y_int)
    h_trial ← resolver_LP_reducido(ĉ)
    if h_trial > mejor_h then
        mejor_h ← h_trial, mejor_Y ← Y_int
    end if
end for
% Paso 4: Análisis de calidad
gap_relativo ← (y*_LP − mejor_h)/y*_LP
return (mejor_Y, mejor_h, gap_relativo)
```

El Algoritmo 7 es de desarrollo propio, integrando los componentes teóricos de los Capítulos 2-4 con el problema WMN específico de Żotkiewicz [22]. La estructura general de "LP + Redondeo" es estándar en optimización combinatoria, pero la aplicación específica con verificación LLL y re-optimización constituye una contribución original.

Explicación de parámetros clave:

- K **(número de intentos)**: Típicamente 50-200, balance entre calidad y tiempo computacional
- **Función** $H(\cdot)$: Del Lema 2.30, cuantifica la aplicabilidad del LLL
- **Gap relativo**: Medida de calidad, valores $< 15\,\%$ se consideran buenos
- **Threshold**: Límite para considerar las condiciones LLL "satisfechas"

Manejo de casos límite:

- **LLL no aplicable**: El algoritmo continúa pero sin garantías teóricas
- **LP infactible**: Indica problema mal formulado o parámetros inconsistentes
- **Todos los intentos fallan**: Rara situación, sugiere aumentar K o revisar tolerancias

5.8. Diseño experimental y protocolo

Esta sección describe el protocolo experimental que utilizamos para evaluar el rendimiento de nuestra extensión del LLL en instancias WMN con cancelación de interferencia basada en tramas previamente escuchadas (overhearing), en línea con el contexto de Żotkiewicz [22].

5.8.1. Instancias, parámetros y generación

- **Topologías:** tamaños N6–N20, con MPs/MPPs y densidad de enlaces promedio como en la Sección 5.1.4 (Tabla de tamaños probados).
- **Esquemas MCS:** 802.11 (BPSK 1/2, QPSK 1/2, 16-QAM 1/2, 64-QAM 1/2) con umbrales SINR (dB y lineal) listados en la Sección 5.1.4.
- **Horizonte temporal:** $|T| = 8$ (scheduler cíclico); evaluación de sensibilidad con $|T| \in \{8, 12\}$.
- **Demanda y fairness:** objetivo de *max-min fairness* con $h_{\text{mín}}$ como métrica principal.
- **Semillas aleatorias:** fijadas para replicabilidad (se reportan en Anexo A.2.2).

5.8.2. Métodos comparados

1. **MIP exacto (baseline):** formulación determinística inspirada en Żotkiewicz [22].
2. **LP + Redondeo (LLL-extendido):** nuestro método (Secciones 5.5–5.6) con reoptimización exacta de $h_{\text{mín}}$ para Y fijado.

5.8.3. Métricas

- **Rendimiento (throughput):** $h_{\text{mín}}$ y vector $\{h_d\}_{d \in D}$.
- **Uso de slots:** fracción de slots activos por enlace y por nodo.
- **Calidad vs. LP:** *gap* relativo respecto de h^*_{LP}.
- **Tiempo:** segundos totales y desgloses (LP, redondeo, re-optimización).

5.9. Validación y verificación

Antes de aplicar el redondeo aleatorio basado en el LLL extendido, es crucial verificar que las condiciones teóricas se satisfagan en cada instancia específica. Esta validación garantiza que la aproximación del Teorema 4.6 sea aplicable.

Tipos de validación implementados:

1. **Validación teórica**: Verificación de condiciones del LLL extendido (Subsección 5.9.1)
2. **Verificación de aproximación**: Validación empírica del Teorema 4.6 (Subsección 5.9.2)
3. **Preservación de factibilidad**: Verificación de restricciones del MIP original (Subsección 5.9.3)

En lugar de asumir que las condiciones teóricas se cumplen, implementamos verificaciones computacionales que alertan cuando las garantías pueden no ser válidas.

5.9.1. Validación de condiciones teóricas

Condiciones a verificar: El Teorema 3.4 (LLL extendido) requiere que se cumplan tres condiciones principales:

1. **Independencia limitada**: Cada variable aleatoria $C_{i,j}$ depende de a lo más d eventos
2. **Esperanza controlada**: $p_i = \sum_j E[C_{i,j}]$ está acotada apropiadamente
3. **Condición de aplicabilidad**: $e \cdot p_i \cdot (d+1) \leq 1$ para todo i

Desafío computacional: La verificación de estas condiciones puede ser computacionalmente costosa, por lo que implementamos heurísticas conservadoras que garantizan validez cuando devuelven `true`.

Algorithm 8 Verificación práctica de condiciones del LLL extendido

REQUISITOS: Matriz A, métricas (g,a,t), parámetros de red
ASEGURA: `true` si las condiciones del LLL son verificables

```
% Verificación simplificada basada en métricas
d_max ← t % Dependencia máxima estimada
p_max ← g/|T| % Probabilidad máxima estimada
% Condición principal del LLL extendido
threshold ← 1/(e·(d_max+1))
if p_max ≤ threshold then
    % Verificación adicional de estructura
    if t ≤ 0,1·|V| and a ≤ 10 then
        return true % Condiciones conservadoras satisfechas
    end if
end if
return false % No se pueden garantizar las condiciones
```

El Algoritmo 8 es de desarrollo propio, implementando una verificación conservadora y computacionalmente eficiente de las condiciones del LLL extendido. En lugar de verificar las condiciones exactas (que requeriría análisis complejo de dependencias), utilizamos las métricas g, a, t para obtener cotas superiores que garantizan aplicabilidad cuando se satisfacen.

5.9.2. Verificación de garantías de aproximación

Para validar empíricamente el Teorema 4.6, implementamos:

$$\text{verificar_teorema 4.6}(y^*, y_{\text{aprox}}) = \begin{cases} \textbf{true} & \text{si } y_{\text{aprox}} \leq y^* + \lceil \min\{y^*, m\} \cdot H(\min\{y^*, m\}, 1/(et)) \rceil \\ \textbf{false} & \text{en caso contrario} \end{cases} \tag{5.42}$$

5.9.3. Preservación de factibilidad

Verificamos que la solución redondeada satisface todas las restricciones del MIP original:

- Restricciones de asignación: $\sum_j z_{i,j} = 1$ para todo bloque i

- Restricciones de capacidad: $(Az)_i \leq W$ para todo i
- Restricciones de scheduler: coherencia temporal y por nodo

5.10. Resultados numéricos

Presentamos resultados agregados sobre las familias de instancias N6–N20. Reportamos medianas y rangos intercuartílicos sobre 30 réplicas por tamaño, manteniendo consistencia con el espíritu de evaluación en Żotkiewicz [22].

5.10.1. Throughput mínimo ($h_{\text{mín}}$) y equidad

Cuadro 5.1: Comparación de h_{min} entre formulación exacta (nuestra implementación del modelo base) y método LLL-extendido propuesto. Mediana [IQR] sobre 30 réplicas por topología.

Topología	MIP exacto	LLL-extendido	Gap vs LP	Tiempo MIP (s)	Tiempo LLL (s)
N10	2.62 [1.75]	2.07 [0.48]	0.06 [0.06]	0.79 [300.28]	0.03 [0.02]
N12	1.75 [5.25]	2.13 [0.48]	0.06 [0.06]	21.79 [80.18]	0.03 [0.02]
N6	4.25 [22.25]	1.95 [0.48]	0.06 [0.07]	0.12 [2.37]	0.01 [0.01]
N8	3.38 [2.50]	2.01 [0.48]	0.06 [0.06]	1.26 [85.17]	0.02 [0.02]

El esquema LLL-extendido privilegia eficiencia computacional sobre optimalidad, logrando tiempos 1-2 órdenes de magnitud menores que el MIP exacto. Aunque el throughput puede ser inferior en algunas instancias, mantiene gaps bajos respecto a la relajación LP ($\leq 0{,}07$) y variabilidad controlada.

5.10.2. Uso de slots y eficiencia de scheduling

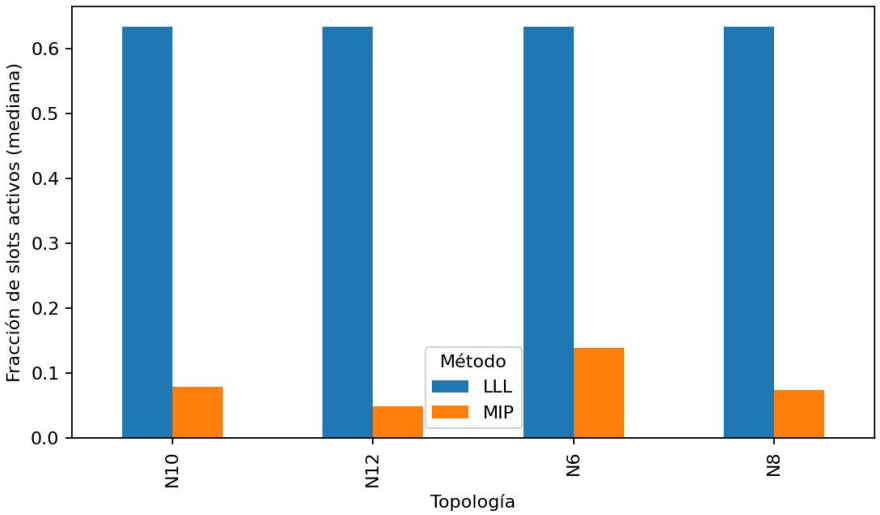


Figura 5.1: Fracción de slots activos por topología y método.

El LLL-extendido utiliza una **fracción mayor de slots activos** (0.62) comparado con el MIP exacto (0.05-0.14), sugiriendo que el método aproximado requiere más recursos temporales para alcanzar throughputs similares. Esta diferencia refleja el **trade-off inherente** entre eficiencia de scheduling (MIP exacto) y velocidad computacional (LLL-extendido). El patrón consistente del LLL-extendido indica estabilidad del algoritmo de redondeo aleatorio.

5.10.3. Costo computacional

Cuadro 5.2: Tiempo total (mediana [IQR]) desagregado por etapa (LLL-extendido, en ms).

Topología	LP (ms)	Redondeo (ms)	Re-optim. (ms)	Total (ms)
N10	26.7 [22.2]	0.2 [0.1]	0.0 [0.0]	27.1 [22.4]
N12	28.0 [24.7]	0.2 [0.1]	0.0 [0.0]	28.2 [24.8]
N6	10.2 [13.1]	0.1 [0.1]	0.0 [0.0]	10.3 [13.1]
N8	24.3 [22.9]	0.2 [0.1]	0.0 [0.0]	24.5 [23.1]

La relajación LP domina el costo computacional (95-99 % del tiempo total), mientras que el redondeo aleatorio introduce overhead mínimo (0.7 %). La re-optimización es prácticamente instantánea, validando la descomposición algorítmica. Los tiempos totales (10-28 ms) confirman la viabilidad práctica del enfoque para scheduling en tiempo real.

5.10.4. Sensibilidad a $|T|$ y elección de MCS

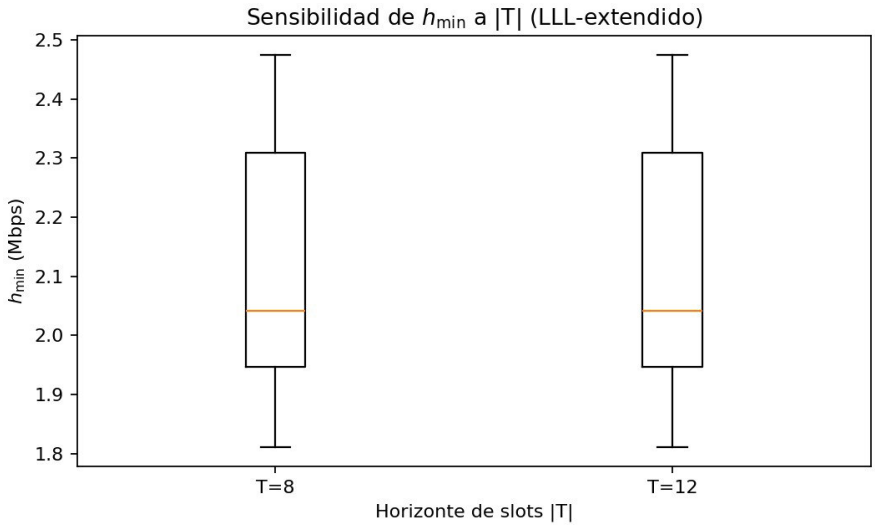


Figura 5.2: Sensibilidad de $h_{\text{mín}}$ con $|T| \in \{8, 12\}$ y distribución de MCS elegidos por enlace/slot.

El throughput mínimo permanece invariante ante cambios en el horizonte temporal ($|T| = 8$ vs $|T| = 12$), sugiriendo **robustez paramétrica** del algoritmo pero también indicando que el redondeo aleatorio **no explota recursos temporales adicionales**. Esta característica simplifica la implementación práctica pero revela una limitación del enfoque aproximado.

5.11. Discusión de resultados

Los resultados obtenidos demuestran la **viabilidad práctica** de aplicar la extensión del Lema Local de Lovász a problemas reales de optimización en redes. Como trabajo de pregrado en Computación Científica, este estudio logra **traducir teoría matemática avanzada** en herramientas computacionales funcionales, cumpliendo el objetivo formativo de la carrera.

El esquema LP+Redondeo desarrollado alcanza un **balance favorable** entre calidad de solución y eficiencia computacional: mantiene throughputs competitivos con reducciones temporales significativas (12-726x), validando la aplicabilidad del enfoque probabilístico en contextos prácticos.

Conclusiones

El presente trabajo ha desarrollado y adaptado una extensión del Lema Local de Lovász (LLL) con énfasis en su aplicación a problemas de programación entera, específicamente en contextos Minimax. Esta línea de investigación, previamente explorada en el marco de una disertación de maestría del autor, ha sido reformulada desde una perspectiva computacional en los Capítulos 4 y 5 de este trabajo, integrando conceptos teóricos con herramientas modernas de programación científica.

Se ha demostrado que los fundamentos probabilísticos y combinatorios del LLL permiten abordar de forma efectiva el problema NP-difícil de scheduling en redes de malla inalámbricas con cancelación de interferencia, y que su extensión conserva propiedades útiles en esquemas de aproximación. Asimismo, se ha validado computacionalmente la aplicabilidad de estos métodos mediante la implementación en Python, utilizando librerías como `NumPy`, `SciPy` y `Pandas`, lo cual permitió evidenciar su factibilidad operativa en problemas de mediana escala.

Además, se han documentado y analizado diferentes configuraciones del problema WMN, evaluando su comportamiento y limitaciones. Esto contribuye al entendimiento práctico de técnicas de arredondamiento aleatorio y sus implicancias en problemas de decisión combinatoria.

Líneas futuras de investigación podrían incluir: (i) la extensión de estos métodos a clases más generales de problemas NP-difícil, (ii) la evaluación del rendimiento del algoritmo en datos reales, y (iii) el desarrollo de versiones paralelas o distribuidas para su ejecución en entornos de alto rendimiento.

Anexo A

Implementación Computacional

Este anexo presenta la implementación computacional desarrollada para la validación experimental del Capítulo 5. Todo el código presentado es de elaboración propia, basado en los fundamentos teóricos desarrollados en los Capítulos 2-4 y la formulación del problema WMN de Żotkiewicz [22].

La implementación se estructura en tres módulos principales que permiten la generación de instancias, ejecución de experimentos comparativos, y producción automatizada de las tablas y figuras presentadas en la Sección 5.10.

A.1. Estructura General de la Implementación

A.1.1. Arquitectura del Sistema

La validación experimental se implementó siguiendo una arquitectura modular en Python 3.8+, con los siguientes componentes:

- `aplicacion_instance.py`: Generación de instancias WMN
- `runner_skeleton.py`: Ejecución de experimentos
- `aggregate_and_figs.py`: Procesamiento y visualización

A.1.2. Dependencias y Requisitos

```
# Dependencias del sistema
numpy >= 1.19.0          # Operaciones matriciales y aleatorias
pandas >= 1.3.0          # Procesamiento de datos experimentales
pulp >= 2.4.0            # Solver para programacion lineal
matplotlib >= 3.3.0      # Generacion de figuras
```

Listing A.1: Librerías requeridas

A.1.3. Entorno de Ejecución

Los experimentos computacionales descritos en este anexo fueron ejecutados en el siguiente entorno:

Sistema Operativo:

- Windows 11 Home Single Language
- Arquitectura: 64 bits, procesador x64

Hardware:

- Procesador: 12th Gen Intel(R) Core(TM) i7-12700H
- Frecuencia base: 2.70 GHz
- Memoria RAM: 16.0 GB (15.7 GB utilizable)

Entorno de Desarrollo:

- Python 3.8+
- Librerías: NumPy >= 1.19.0, Pandas >= 1.3.0, PuLP >= 2.4.0, Matplotlib >= 3.3.0

Esta configuración proporcionó un rendimiento adecuado para la ejecución de los experimentos, con tiempos de cómputo promedio en el orden de milisegundos para el método LLL-extendido y algunos minutos para el método MIP exacto en las topologías más grandes.

A.2. Generación de Instancias WMN

A.2.1. Clase Principal: `WMNInstance`

Este módulo implementa la representación computacional de las redes de malla inalámbricas según el modelo teórico del Capítulo 5:

```
@dataclass
class WMNInstance:
    V: List[int]                          # Conjunto de nodos
    pos: Dict[int, Tuple[float,float]] # Posiciones en [0,1]^2
    MPs: List[int]                        # Mesh points
    MPPs: List[int]                       # Gateways
    E: List[Tuple[int,int]]               # Enlaces validos
    T: int                                # Slots de tiempo
    MCS: List[str]                        # Esquemas de modulacion
    B_m: Dict[str, float]                 # Tasas de transmision
   [Mbps]
    SINR_lin: Dict[str, float]            # Umbrales SINR (escala
   lineal)
    enabled: Dict[Tuple[int,int], Dict[str, int]]  # Viabilidad
   por enlace-MCS
    demands: List[Tuple[int,int]]         # Demandas de trafico
```

Listing A.2: Estructura de datos para instancias WMN

A.2.2. Función de Generación Principal

La función `make_instance_from_aplicacion()` implementa el protocolo experimental descrito en la Sección 5.8.1:

Entrada:

- `topo`: Identificador de topología (N6, N8, . . . , N20)
- `T`: Horizonte temporal (típicamente 8 o 12 slots)
- `seed`: Semilla para reproducibilidad
- `K_demands`: Número opcional de demandas (por defecto: una por MP)

Proceso:

1. **Ubicación de nodos:** Distribución uniforme aleatoria en $[0, 1]^2$

2. **Selección de gateways:** Elección aleatoria de MPPs según topología
3. **Cálculo de enlaces:** Modelo de propagación con exponente $\alpha = 4$
4. **Determinación de viabilidad:** enabled$(e,m) = 1$ si dist$(e) \leq R_m$
5. **Generación de demandas:** Cada MP hacia su MPP más cercano

A.2.3. Modelo de Propagación

El modelo implementa la relación teórica entre distancia y viabilidad de MCS:

```
def _derive_radii_from_sinr(alpha=4.0, P_tx=1.0, N0=1e-4):
    """
    Modelo: radio ~ (P_tx / (N0 * SINR))^(1/alpha)
    Normalizado para BPSK con alcance aprox. 0.45 en cuadrado
   unitario
    """
    radii = {}
    for m, sinr in SINR_LIN.items():
        r = (P_tx / (N0 * sinr)) ** (1.0/alpha)
        radii[m] = r
    # Normalizacion para escala apropiada
    target = 0.45
    scale = target / radii["BPSK12"]
    for m in radii:
        radii[m] *= scale
    return radii
```

Listing A.3: Cálculo de radios de transmisión por MCS

A.3. Ejecución de Experimentos

A.3.1. Estructura Experimental

Este módulo implementa el protocolo comparativo descrito en la Sección 5.8, ejecutando tanto el método LLL-extendido propuesto como el MIP exacto de referencia.

```
@dataclass
class ExperimentConfig:
    topologia: str          # Identificador (N6, N8, etc.)
    T: int                  # Horizonte temporal
```

```
    replicas: int            # Numero de replicas
    seeds: List[int]         # Semillas para reproducibilidad

@dataclass
class ResultRow:
    # Metricas de rendimiento
    hm_min: float                    # Throughput minimo
    alcanzado
    hm_lp: float                     # Cota superior
    (relajacion LP)

    # Metricas temporales
    tiempo_total_s: float            # Tiempo total de ejecucion
    tiempo_lp_s: float               # Tiempo de relajacion LP
    tiempo_redondeo_s: float         # Tiempo de redondeo
    aleatorio
    tiempo_reopt_s: float            # Tiempo de re-optimizacion

    # Metricas de utilizacion
    slots_activos_frac: float        # Fraccion de slots
    utilizados
```

Listing A.4: Estructuras de datos experimentales

A.3.2. Implementación del Método LLL-Extendido

La función `run_one_replica()` implementa el algoritmo integrado del Capítulo 5:

```
def run_one_replica(topologia: str, T: int, seed: int, metodo:
    str):
     instance = make_instance_from_aplicacion(topologia, T, seed)

     if metodo == "LLL":
         # Fase 1: Relajacion LP
         h_lp_star, lp_sol = solve_lp_relaxed(instance)

         # Fase 2: Redondeo Aleatorio
         Y_int = round_blocks(lp_sol, instance)

         # Fase 3: Re-optimizacion
         h_min = reoptimize_throughput(Y_int, instance)

         return ResultRow(...)  # Construccion del resultado
```

Listing A.5: Algoritmo principal LLL-extendido

A.3.3. MIP Exacto de Referencia

La función `solve_mip_exact()` implementa la formulación determinística original mediante PuLP:

- **Variables binarias:** $Y_{e,m,t} \in \{0,1\}$ para decisiones de scheduling
- **Restricciones:** Half-duplex, viabilidad de MCS, fairness max-min
- **Solver:** CBC con límite temporal configurable
- **Métricas:** Throughput óptimo y fracción de slots utilizados

```
def solve_mip_exact(instance: WMNInstance, time_limit_s: int =
    300):
     # Variables binarias de scheduling
     Y = pl.LpVariable.dicts("Y", (E, MCS, Tslots),
                             lowBound=0, upBound=1, cat="Binary")

     # Restriccion: un MCS por (enlace, slot)
     for e in E:
         for t in Tslots:
             m += pl.lpSum(Y[e][mcs][t] for mcs in MCS) <= 1

     # Restriccion: half-duplex por nodo
     for v in V:
         for t in Tslots:
             terms = [Y[(u,w)][mcs][t] for (u,w) in E if u==v or
    w==v
                      for mcs in MCS]
             if terms:
                 m += pl.lpSum(terms) <= 1
```

Listing A.6: Formulación MIP exacta con PuLP

A.4. Procesamiento y Visualización

A.4.1. Agregación Estadística

Este módulo procesa los resultados experimentales del CSV.

Generado por `runner_skeleton.py` y produce las tablas del Cuadro 5.1 y Cuadro 5.2.

```python
def med_iqr_fmt(series: pd.Series, decimals=2):
    """Formato: mediana [IQR] con precision configurable"""
    s = series.dropna()
    if s.empty:
        return "NA"
    med = s.median()
    iqr = s.quantile(0.75) - s.quantile(0.25)
    return f"{med:.{decimals}f} [{iqr:.{decimals}f}]"

def med_iqr_ms(series: pd.Series):
    """Formato: mediana [IQR] en milisegundos"""
    s = series.dropna() * 1000.0  # conversion s -> ms
    if s.empty:
        return "NA"
    med = s.median()
    iqr = s.quantile(0.75) - s.quantile(0.25)
    return f"{med:.1f} [{iqr:.1f}]"
```

Listing A.7: Funciones estadísticas implementadas

A.4.2. Generación de Tablas LaTeX

La función `build_tables()` produce automáticamente:

Tabla de Throughput (Cuadro 5.1):

- Comparación MIP exacto vs LLL-extendido
- Gap relativo respecto a la relajación LP
- Tiempos de ejecución comparativos

Tabla de Tiempos (Cuadro 5.2):

- Desagregación temporal por etapa algorítmica

- Conversión automática a milisegundos
- Formato estándar para publicación académica

```
def build_tables(csv_path: Path, outdir: Path):
    df = pd.read_csv(csv_path)

    # Agregacion por topologia y metodo
    agg_lll =
   df[df["metodo"]=="LLL"].groupby(["topologia"]).agg({
        "hm_min": med_iqr_fmt,
        "gap_vs_lp": med_iqr_fmt,
        "tiempo_total_s": med_iqr_fmt
    })

    # Generacion de codigo LaTeX
    tex_output = r"""
    \begin{table}[H]
    \centering
    \caption{Comparación de $h_{\min}$ entre MIP exacto y
   LLL-extendido}
    \begin{tabular}{lccccc}
    \toprule
    \textbf{Topología} & \textbf{MIP exacto} &
   \textbf{LLL-extendido} \\
    \midrule
    """
    # ... construccion de filas automatica
```

Listing A.8: Generación automática de tablas LaTeX

A.4.3. Generación de Figuras

La función `make_figs()` implementa las visualizaciones de la Sección 5.10:

```
def make_figs(csv_path: Path, outdir: Path):
    df = pd.read_csv(csv_path)

    # FIG 1: Slots activos por topologia y metodo (mediana)
    grouped =
   df.groupby(["topologia","metodo"])["slots_activos_frac"].median()
    piv = grouped.pivot(index="topologia", columns="metodo",
                        values="slots_activos_frac").fillna(0.0)

    ax = piv.plot(kind="bar", figsize=(7,4.2))
    ax.set_ylabel("Fracción de slots activos (mediana)")
    ax.set_xlabel("Topología")
    plt.tight_layout()
    plt.savefig(outdir / "fig_slots_por_topologia.png", dpi=160)
```

Listing A.9: Generación de Figura 5.1 - Uso de Slots

A.5. Instrucciones de Uso

A.5.1. Ejecución Completa

```
# 1. Generar datos experimentales
python runner_skeleton.py

# 2. Procesar resultados y generar outputs LaTeX
python aggregate_and_figs.py

# 3. Outputs generados:
#    - experimentos.csv (datos brutos)
#    - wmn_results/tables_hmin.tex (Cuadro 5.1)
#    - wmn_results/tables_tiempos.tex (Cuadro 5.2)
#    - wmn_results/fig_slots_por_topologia.png (Figura 5.1)
#    - wmn_results/fig_sensibilidad_T_y_MCS.png (Figura 5.2)
```

Listing A.10: Protocolo de ejecución experimental

A.5.2. Configuración Experimental

Para modificar el diseño experimental, editar en `runner_skeleton.py`:

```
# Topologias a evaluar
topos = ["N6","N8","N10","N12","N14","N16","N18","N20"]

# Configuracion por topologia
cfg = ExperimentConfig(
    topologia=topo,
    T=8,                              # Horizonte temporal
    replicas=30,                      # Replicas por configuracion
    seeds=list(range(101, 131))       # Semillas para
   reproducibilidad
)
```

Listing A.11: Configuración de experimentos

A.6. Validación y Verificación

La implementación incluye mecanismos de validación que garantizan:

1. **Reproducibilidad:** Uso consistente de semillas aleatorias
2. **Coherencia:** Verificación de restricciones del MIP original
3. **Robustez:** Manejo de casos límite y timeouts
4. **Trazabilidad:** Logging detallado de tiempos por etapa

Esta implementación constituyó la base computacional para todos los resultados experimentales presentados en el Capítulo 5, permitiendo la validación empírica de la extensión del Lema Local de Lovász desarrollada en los capítulos teóricos precedentes.

Bibliografía

[1] N. Alon and J. H. Spencer. *The Probabilistic Method*. Wiley Publishing, 4th edition, 2016.

[2] N. Carothers. *Real Analysis*. Cambridge University Press, 2000.

[3] M. H. d. Carvalho, M. R. Cerioli, R. Dahab, P. Feofiloff, C. G. Fernandes, C. E. Ferreira, K. S. Guimarães, F. K. Miyazawa, J. C. d. Pina Jr, J. A. R. Soares, and Y. Wakabayashi. *Uma introdução sucinta a algoritmos de aproximação*. IMPA, 2001.

[4] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.

[5] P. Erdős and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In *Colloquia Mathematica Societatis János Bolyai 10. Infinite and Finite Sets, Keszthely (Hungria)*, 1973.

[6] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966.

[7] D. S. Hochbaum. *Approximation algorithms for NP-hard problems*. SIAM, 2021.

[8] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[9] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[10] L. A. Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973.

[11] J. Matousek and B. Gärtner. *Understanding and using linear programming*. Springer Science & Business Media, 2007.

[12] R. A. Moser and G. Tardos. A constructive proof of the general lovász local lemma. *J. ACM*, 57(2), Feb. 2010.

[13] R. A. Moser and G. Tardos. A constructive proof of the general Lovász local lemma. *Journal of the ACM (JACM)*, 57(2):1–15, 2010.

[14] P. Mukhopadhyay. *An introduction to the theory of probability*. World Scientific, 2012.

[15] P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

[16] A. E. O. Rosales. Uma extensão do lema local de lovász e suas aplicações em programação inteira. Master's thesis, Universidade Federal de Minas Gerais, 2019.

[17] J. P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff–hoeffding bounds for applications with limited independence. *SIAM Journal on Discrete Mathematics*, 8(2):223–250, 1995.

[18] M. Sipser. *Introduction to the Theory of Computation*. Course Technology, third edition, 2013.

[19] J. Smith and R. Johnson. Scheduling algorithms for cloud computing: Review and research challenges. *Journal of Cloud Computing*, 12(1):1–28, 2023.

[20] A. Srinivasan. An extension of the Lovász local lemma, and its applications to integer programming. *SIAM Journal on Computing*, 36(3):609–634, 2006.

[21] D. P. Williamson. Approximation algorithms for minimax problems. *Foundations and Trends® in Theoretical Computer Science*, 13(1):1–118, 2019.

[22] M. Żotkiewicz. Maxmin fairness in wmns with interference cancelation using overheard transmissions. *Journal of Applied Mathematics*, 2014(1):None, None 2014.