

# MUSSA

## Generador de Planes de Carrera Personalizados

Jennifer Andrea Woites

Tutores:  
Rosa Wachenchauzer  
Diego Essaya

# Agenda (Parte I)

- Motivación del trabajo
- Tecnologías y Herramientas
- MUSSA Web
- Algoritmos Analizados
- Arquitectura Elegida



## Agenda (Parte II)

- Programación Lineal Entera
- Algoritmo Branch & Cut
- Algoritmo PLE
- Algoritmo Greedy
- Pruebas



## Agenda (Parte III)

- Demo
- Mejoras Futuras
- Conclusiones



# Motivaciones

- Preferencias / Dificultades al elegir materias
- No todas las materias se dictan siempre
- Clasificación de Materias electivas
- Programa de Reinserción Académica



# Tecnologías y Herramientas



Google docs

# Tecnologías y Herramientas





# Tecnologías y Herramientas







# Tecnologías y Herramientas

SQLAlchemy



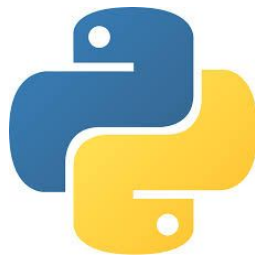
# Tecnologías y Herramientas



**CBC**



**redis**



**PuLP**



# MUSSA Web

- Vista Pública
- Usuario Alumno
- Usuario Administrador





# MUSSA Web: Vista Pública

- Búsqueda de Materias
- Búsqueda Docentes
- Links útiles / Comisiones Curriculares
- Encuestas



# MUSSA Web: Usuario Admin



- Horarios PDF
- Cursos
- Docentes

# Algoritmos Analizados

Greedy



Basados en  
Preferencias



Fuerza  
Bruta

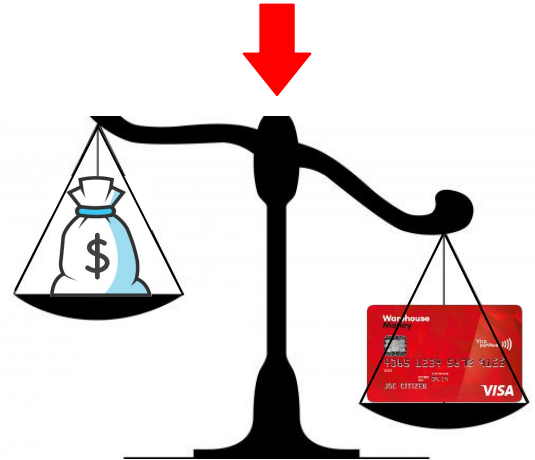


**SOLUTION**



PLE

# Algoritmos Basados en Preferencias



# Algoritmos Elegidos

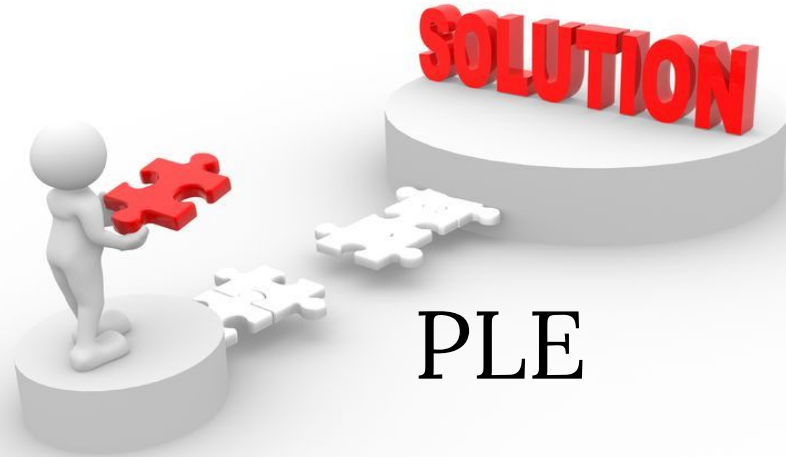
Greedy



Fuerza  
Bruta



Basados en  
Preferencias







# Programación Lineal Entera

Técnica matemática utilizada en la investigación de operaciones, que permite la optimización de una función objetivo a través de la aplicación de diversas restricciones a sus variables.



# Programación Lineal Entera

La solución óptima tiene sentido solamente si una parte o todas las variables de decisión toman valores restringidos a números enteros.



# Programación Lineal Entera

¿Son más fáciles de resolver que los continuos?

- Número de soluciones factibles a analizar finito (cuando el conjunto de oportunidades está acotado)
- Este número es suficientemente grande como para que resulte imposible su comparación.



# Programación Lineal Entera: Paso I

Comienzan su ejecución con la resolución del problema lineal asociado (PLA) consistente en eliminar las condiciones de integridad, obteniéndose en consecuencia un problema de programación lineal que puede ser resuelto mediante el algoritmo simplex.



# Programación Lineal Entera: Paso II

¿Verifica las condiciones de integridad?

- ¿SI? Es la solución al problema entero.
- ¿NO? Requiere utilizar otro método que permita resolver el problema entero (Ej: Branch & Cut)

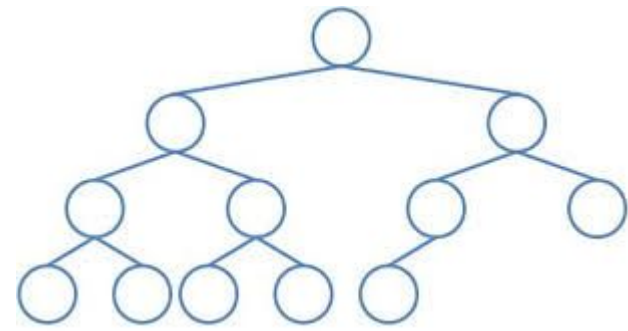


# Branch & Cut

El método Branch & Cut es un híbrido de:

- Branch & Bound (Ramificación y Acotamiento)
- Cutting Planes (Planos cortantes)

# Branch & Bound

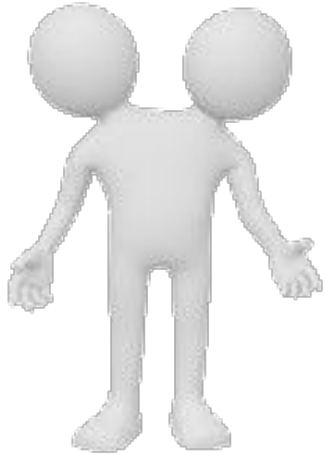


Genera un árbol de soluciones.  
El algoritmo se encarga de detectar qué ramificaciones no están siendo óptimas para podar esa rama del árbol.



# Branch & Bound

En cada rama, se resuelve el problema “relajado”, es decir, se quitan algunas restricciones para hacerlo más permisivo.



- Se divide el problema en dos versiones:
- 1)  $\text{variable} \geq \text{siguiente entero resultado}$
  - 2)  $\text{variable} \leq \text{anterior entero resultado}$





# Branch & Bound

Solución del problema original  $\rightarrow$  Solución del problema relajado.

Si la solución óptima del problema relajado es válida para el problema original, es la solución óptima para ese problema.



# Cutting Planes

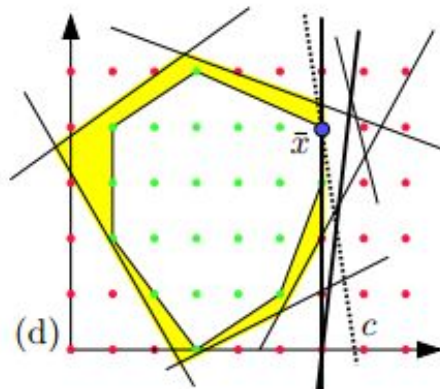
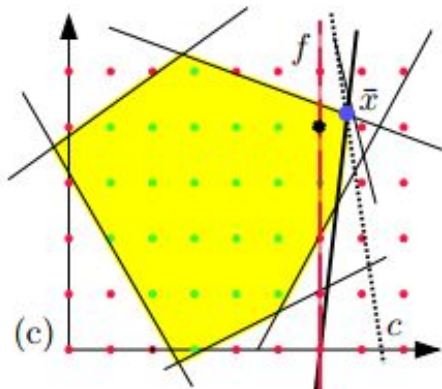
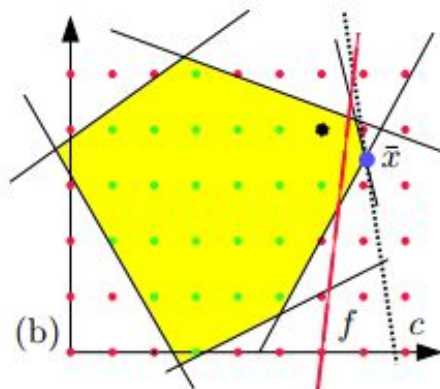
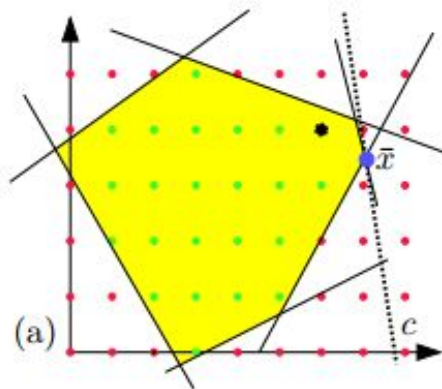
Resuelve programas lineales sin restricciones enteras usando algoritmos regulares simplificados. Cuando se obtiene una solución óptima que tiene un valor no entero (para una variable entera), se utiliza este algoritmo para encontrar una restricción lineal que sea satisfecha por los puntos factibles enteros pero violados por la solución actual.

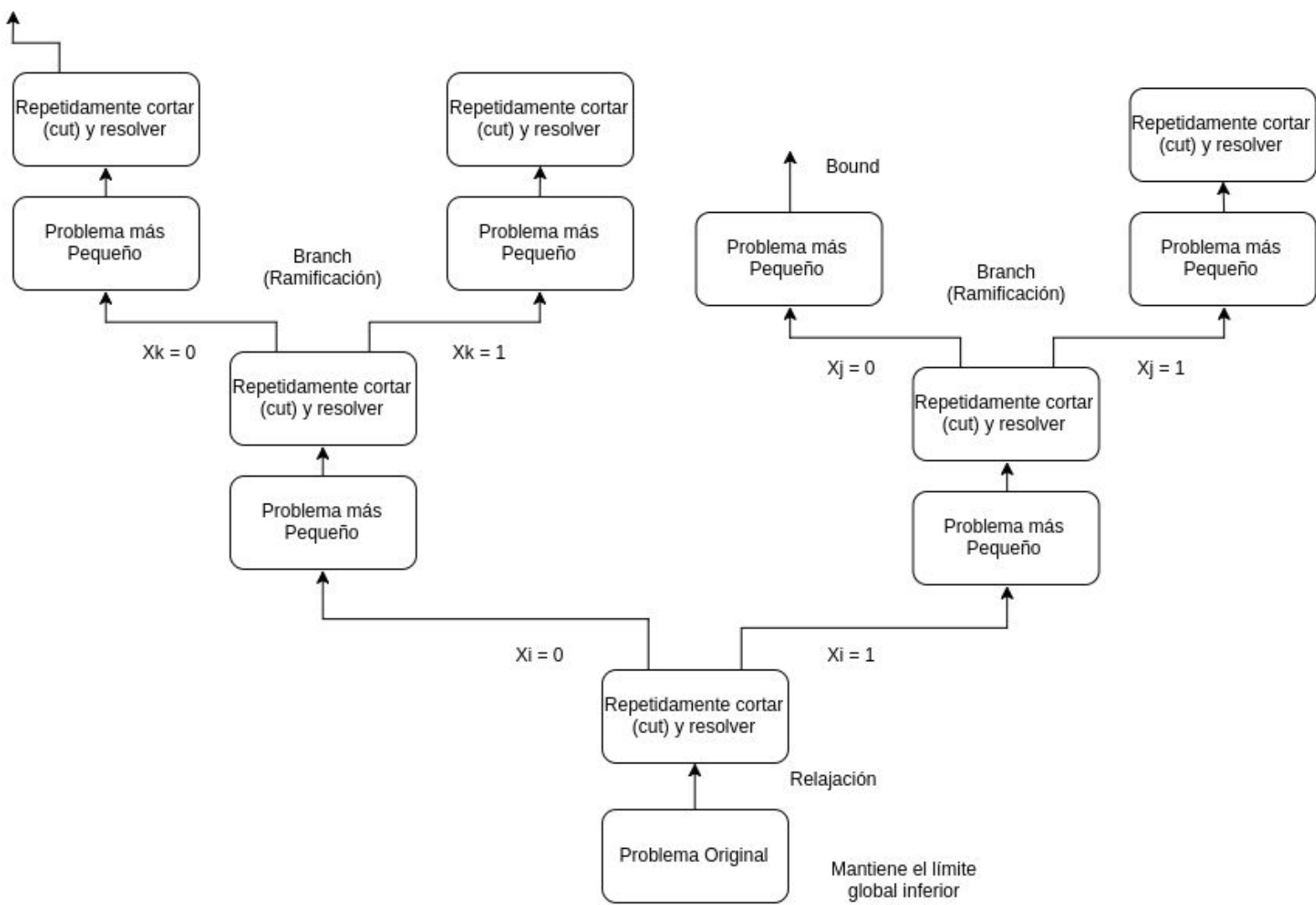


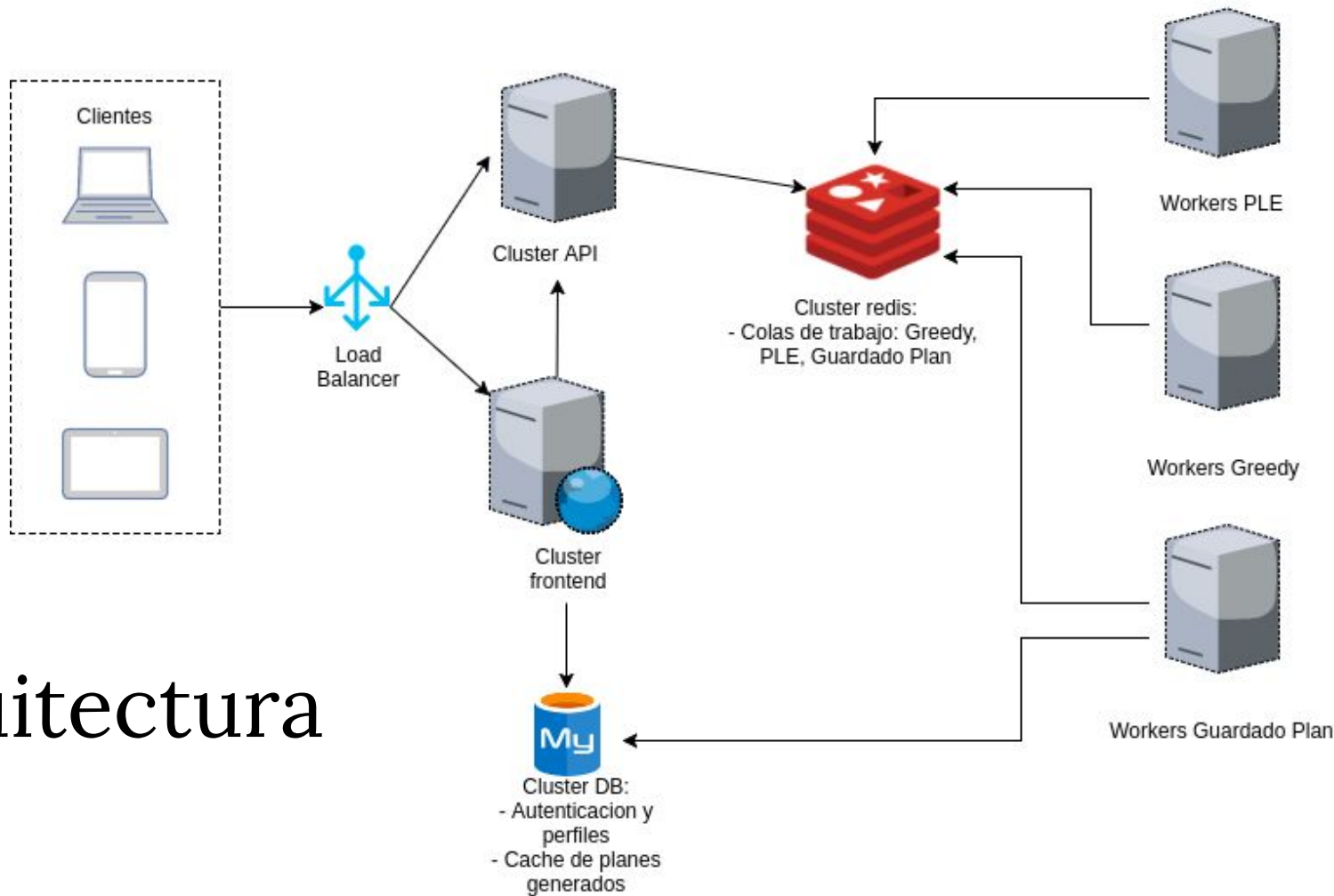
# Cutting Planes

Si se encuentra la desigualdad, se añade al programa lineal, que nos dará una nueva solución. Este proceso se repite hasta que se encuentra la solución entera (óptima) o no se encuentran más planos de corte. Cuando no se encuentran más planos, se procede nuevamente a la ramificación.

# Cutting Planes







# Arquitectura



# Configuración de Parámetros

- Inicio con plan completo y créditos según orientación y trabajo final elegidos
- Se eliminan las materias aprobadas
- Se eliminan las materias con final pendiente y se actualizan los cuatrimestres mínimos de las correlativas



# Configuración de Parámetros

- Separación de Materias del CBC (Ciclo Básico Común)
- Cursos elegidos por el usuario como materias obligatorias
- Verificación de puntajes y horarios de los cursos a agregar





# Configuración de Parámetros

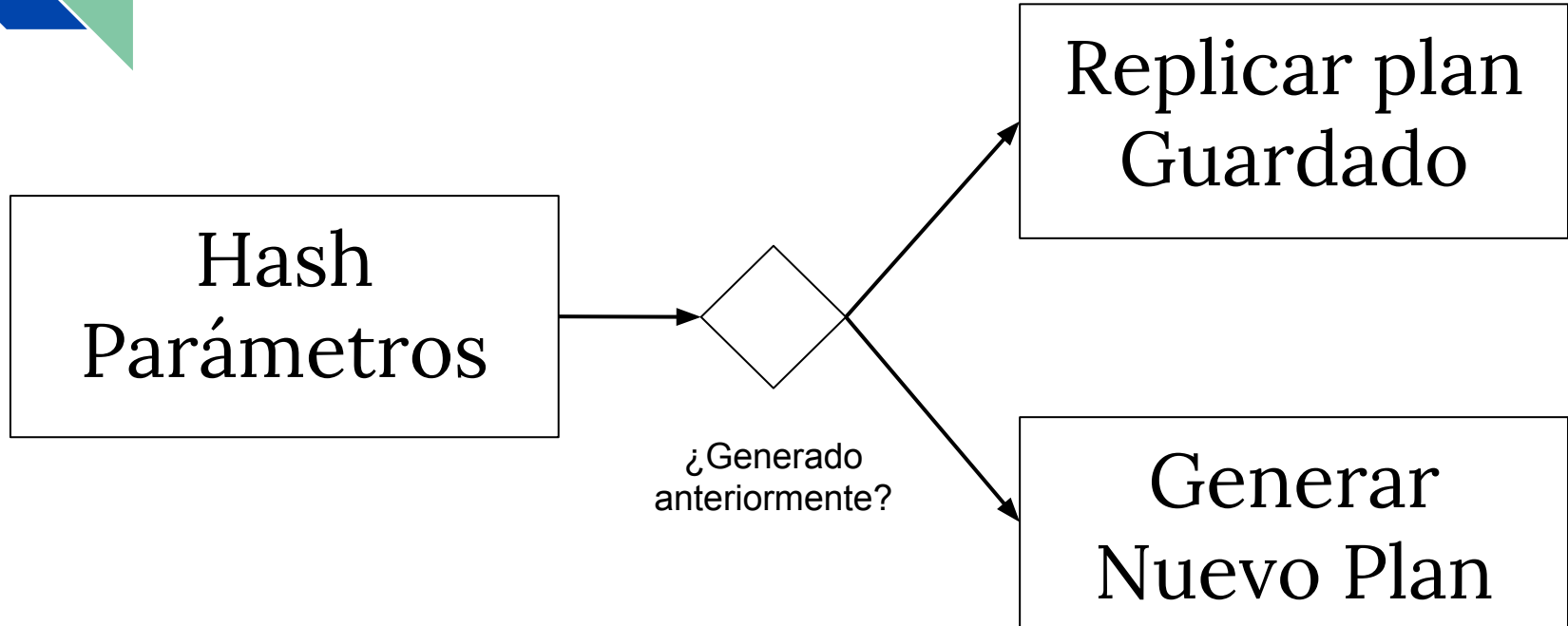
- Actualización de créditos electivas. Si se completaron o se hicieron materias extra, se eliminan las electivas disponibles
- Cálculo de créditos mínimos por temática
- Primer cuatrimestre del plan es 1º o 2º cuatrimestre



# Configuración de Parámetros

- ¿Se pueden cursar las materias obligatorias?
- Eliminar materias incompatibles
- ¿Las materias electivas son suficientes?
- Franjas mínima y máxima. Días de la semana

# Configuración de Parámetros



# Algoritmo PLE



- Código para PuLP → Modelado
- Función a minimizar: Total de Cuatrimestres
- Se arman ecuaciones para que cumplan las diversas restricciones. Por ejemplo:

Para toda  $i$  obligatoria

$$\sum_j Y_{ij} = 1; \forall j \in \{\text{cuatrimestres}\}$$


Para toda  $i$  electiva

$$\sum_j Y_{ij} \leq 1; \forall j \in \{\text{cuatrimestres}\}$$



# Algoritmo PLE: Optimizaciones

- Se eliminan las variables que se sabe que el valor es 0 y se las quita de las ecuaciones (se reducen las variables)
- Se eliminan las restricciones donde se tiene una variable de tipo entero como mayor o igual a 0 y como menor igual a INFINITO ya que son redundantes con la naturaleza de la variable



# Algoritmo PLE: Ejecutando el Algoritmo

- Se ejecuta el algoritmo con el solver CBC
- El solver CBC es multi thread y utiliza como método de resolución un algoritmo de Branch & Cut

# Algoritmo Greedy

- Se ordenan las materias: Obligatorias / Electivas primarias / Electivas Secundarias.
- Las electivas primarias son aquellas que aportan créditos a las temáticas elegidas (si no hay temáticas, todas las electivas son de este tipo).
- Las electivas secundarias son aquellas que no aportan créditos a las temáticas elegidas.





# Algoritmo Greedy

Las materias obligatorias se ordenan según:

- Menor horario de fin (Menor franja máxima / Menor cantidad de días / Menor cantidad de días con menor franja máxima)
- Mayor cantidad de materias liberadas
- Mayor cantidad de franjas ocupadas
- Mayor puntaje
- Mayor cantidad de créditos
- Menor código de materia





# Algoritmo Greedy

Las materias electivas se ordenan según:

- Menor horario de fin
- Mayor cantidad de créditos en temáticas
- Mayor cantidad de créditos
- Mayor cantidad de horas de cursada
- Mayor cantidad de materias liberadas
- Mayor puntaje
- Menor código de materia

# Algoritmo Greedy

Mientras que el cuatrimestre no esté completo:

- El algoritmo intenta agregar las materias en orden de aparición. Si la materia encaja con las colocadas anteriormente, la agrega. Sino la descarta e intenta con la próxima materia

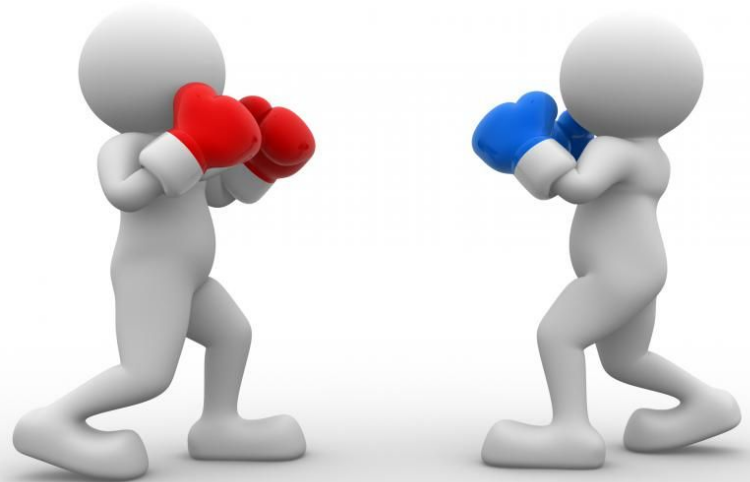


# Algoritmo Greedy

Si las combinaciones son pocas → Realiza Fuerza Bruta

Para ello, dada una materia, genera dos posibles combinaciones, una con la materia agregada y otra en la que se descarte la misma.

Luego elige la mejor combinación disponible.





# Algoritmo Greedy + Fuerza Bruta

La mejor combinación es elegida por:

- Mayor cantidad de materias
- Mayor cantidad de materias obligatorias que libera
- Mayor cantidad de créditos en temáticas cubiertos.
- Mayor cantidad de créditos en electivas.
- Mayor cantidad de créditos totales.
- Menor cantidad de horas de cursada.
- Menor cantidad de horas extra

# Pruebas

El algoritmo se probó en 2 máquinas con características diferentes:



Máquina 1:

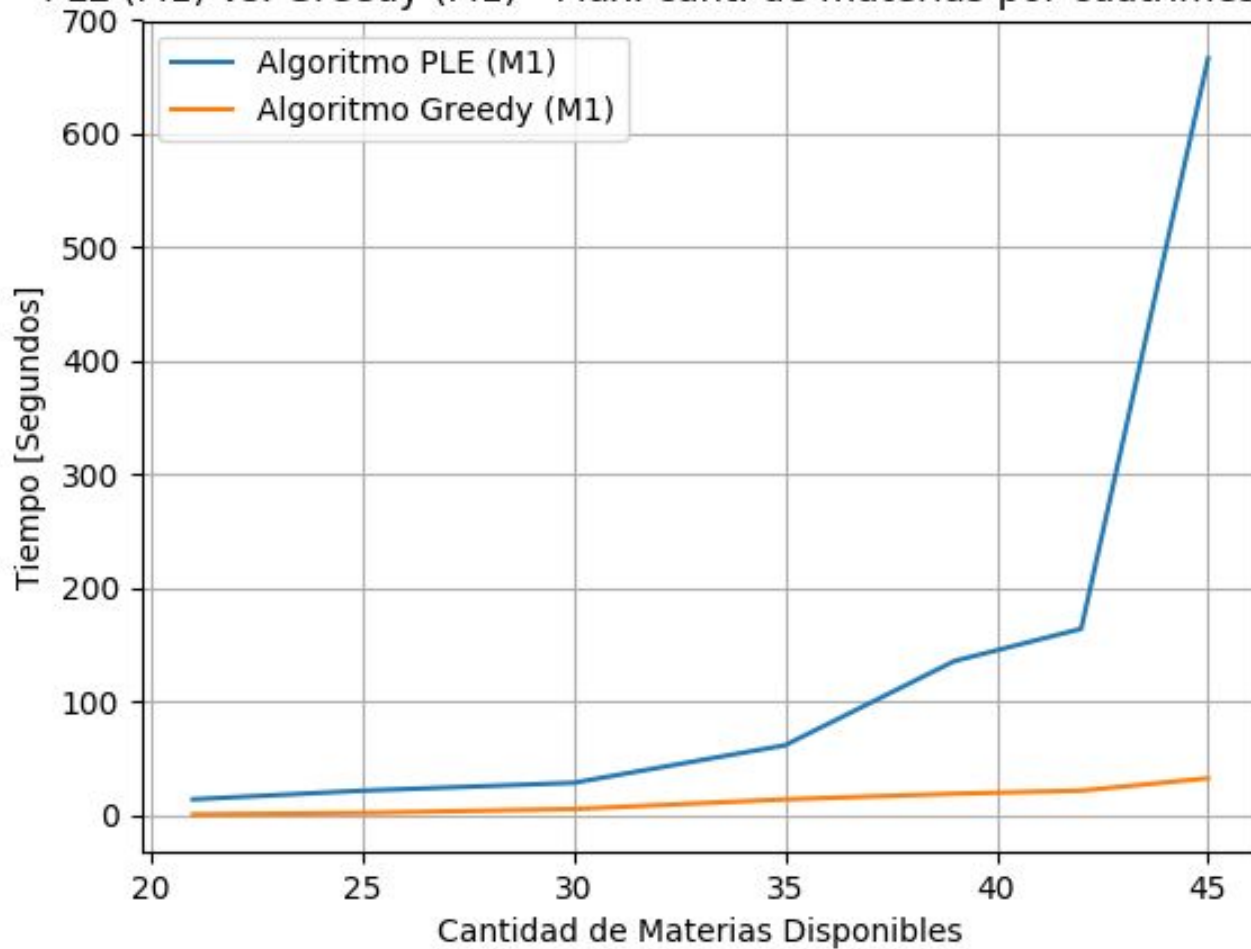
Intel Core I3-2330M CPU 2.20GHz  
10 GB RAM / 2 Core



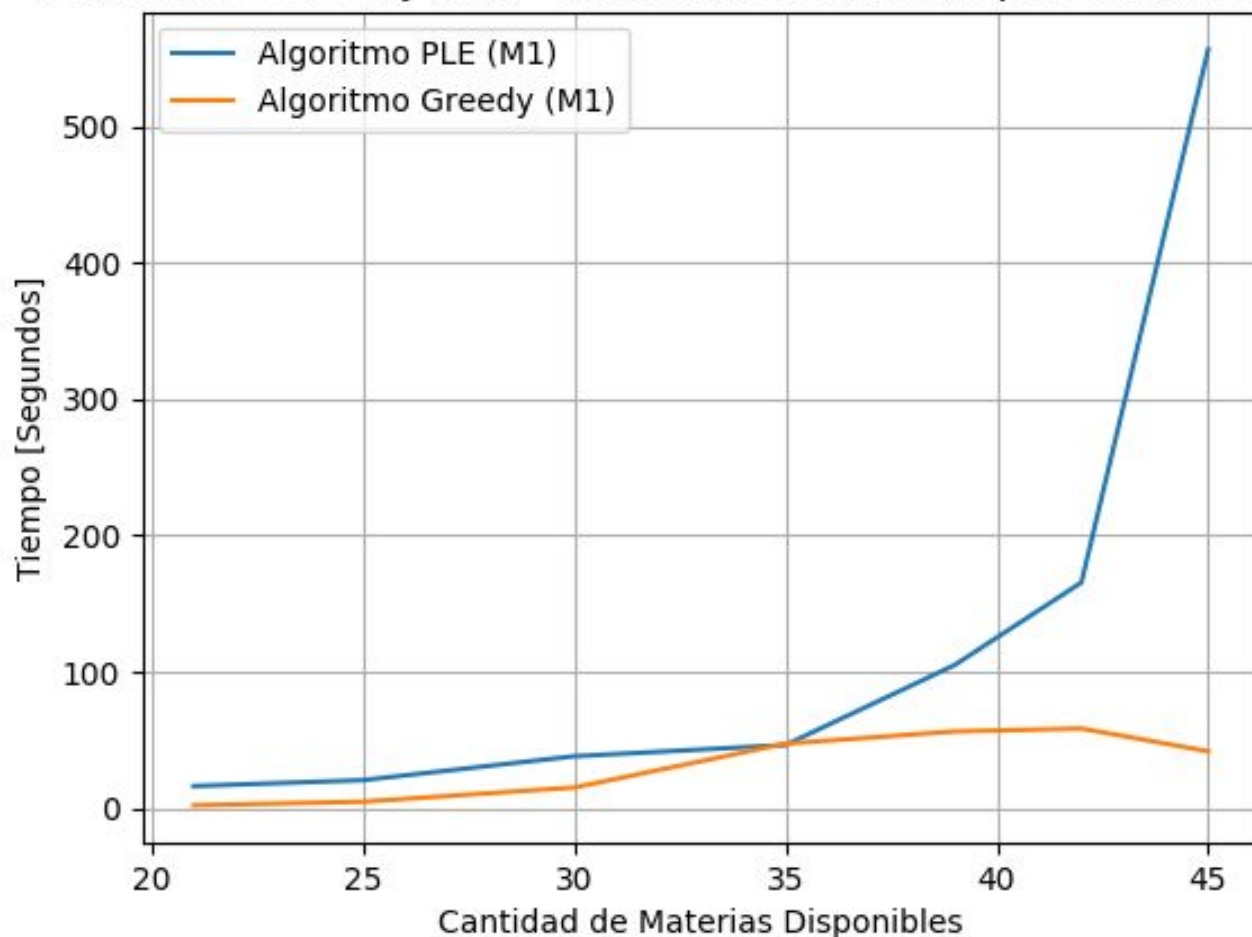
Máquina 2:

Intel Xeon Processor (Skylake) 2.10GHz  
16 GB RAM / 4 Core

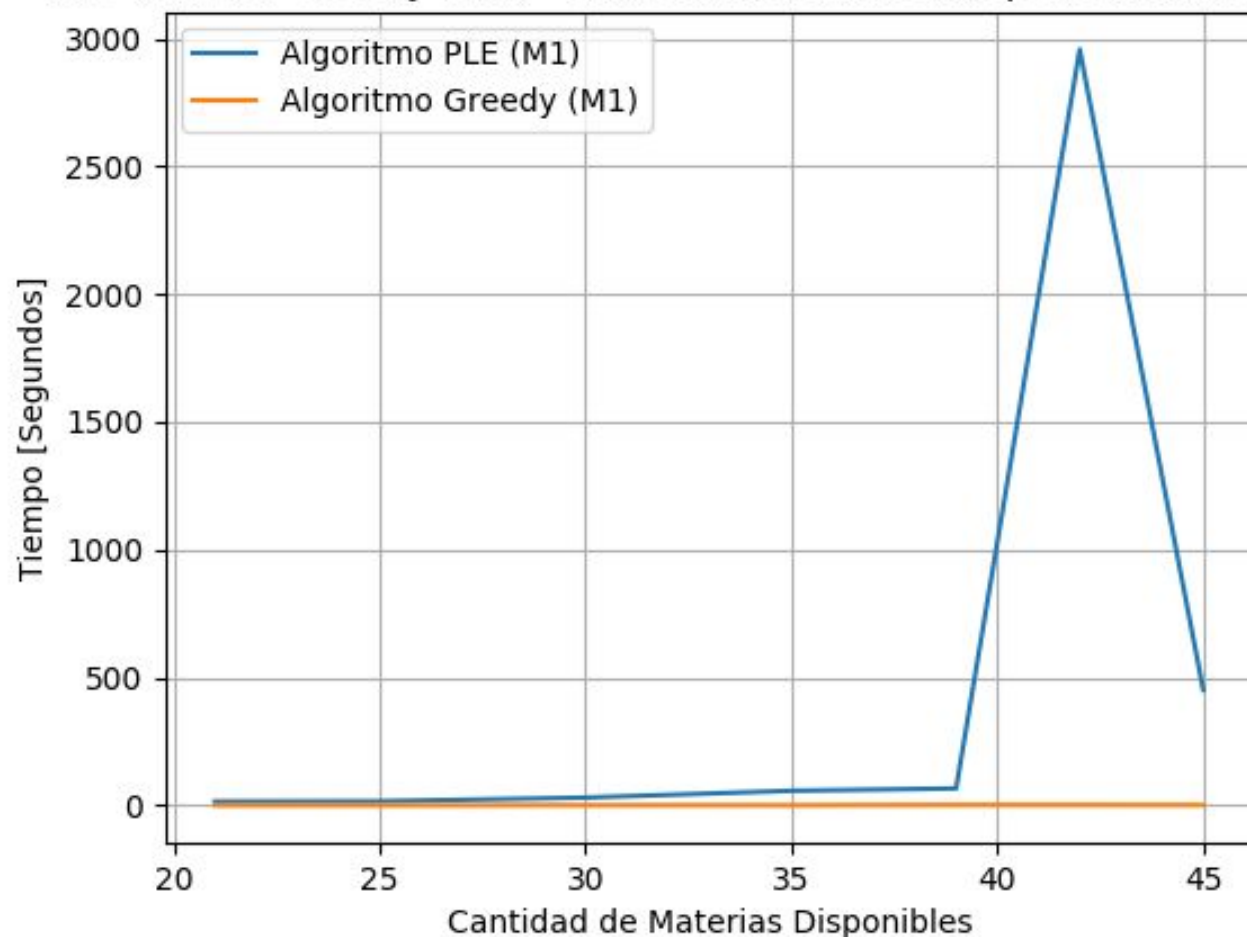
## PLE (M1) vs. Greedy (M1) - Máx. cant. de materias por cuatrimestre: 2



### PLE (M1) vs. Greedy (M1) - Máx. cant. de materias por cuatrimestre: 3

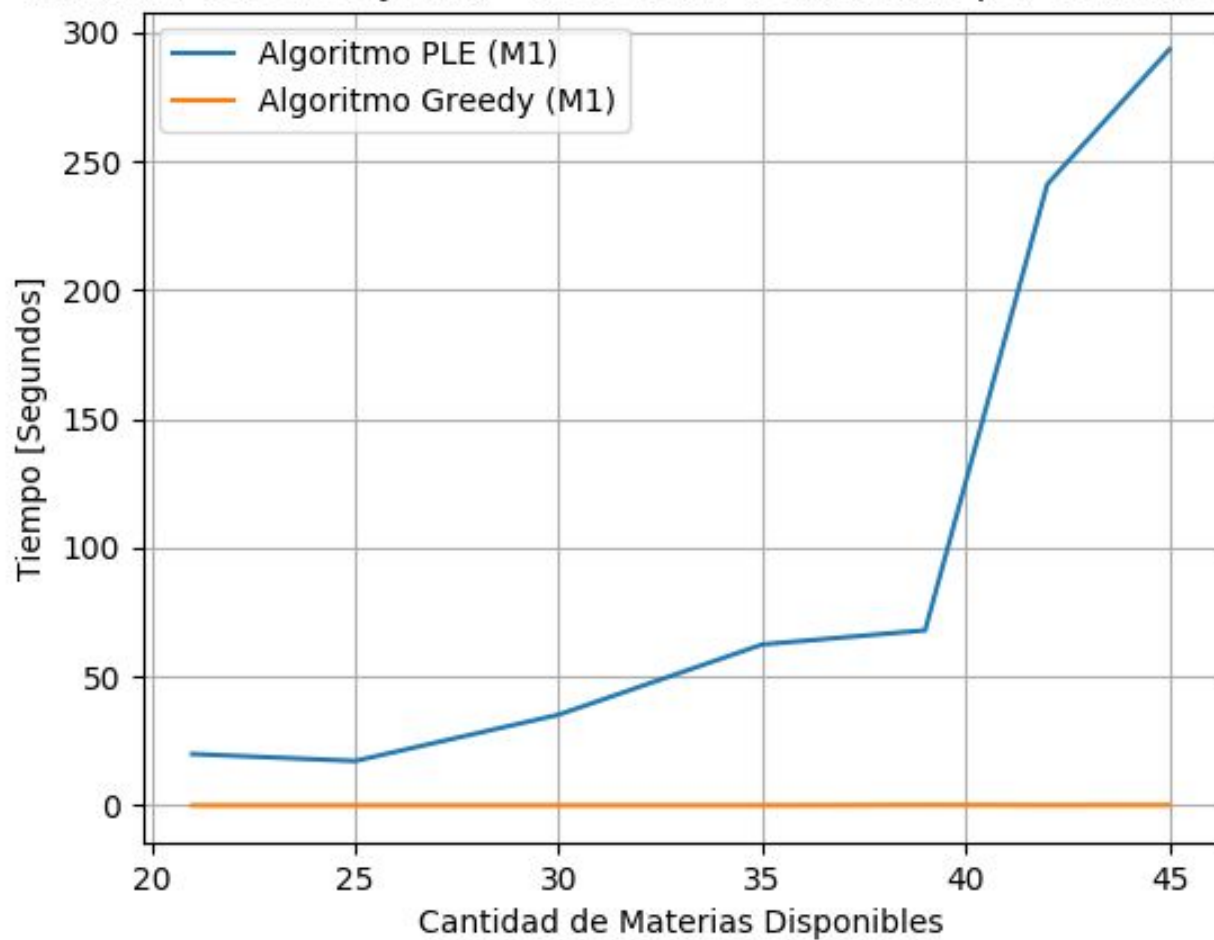


## PLE (M1) vs. Greedy (M1) - Máx. cant. de materias por cuatrimestre: 4

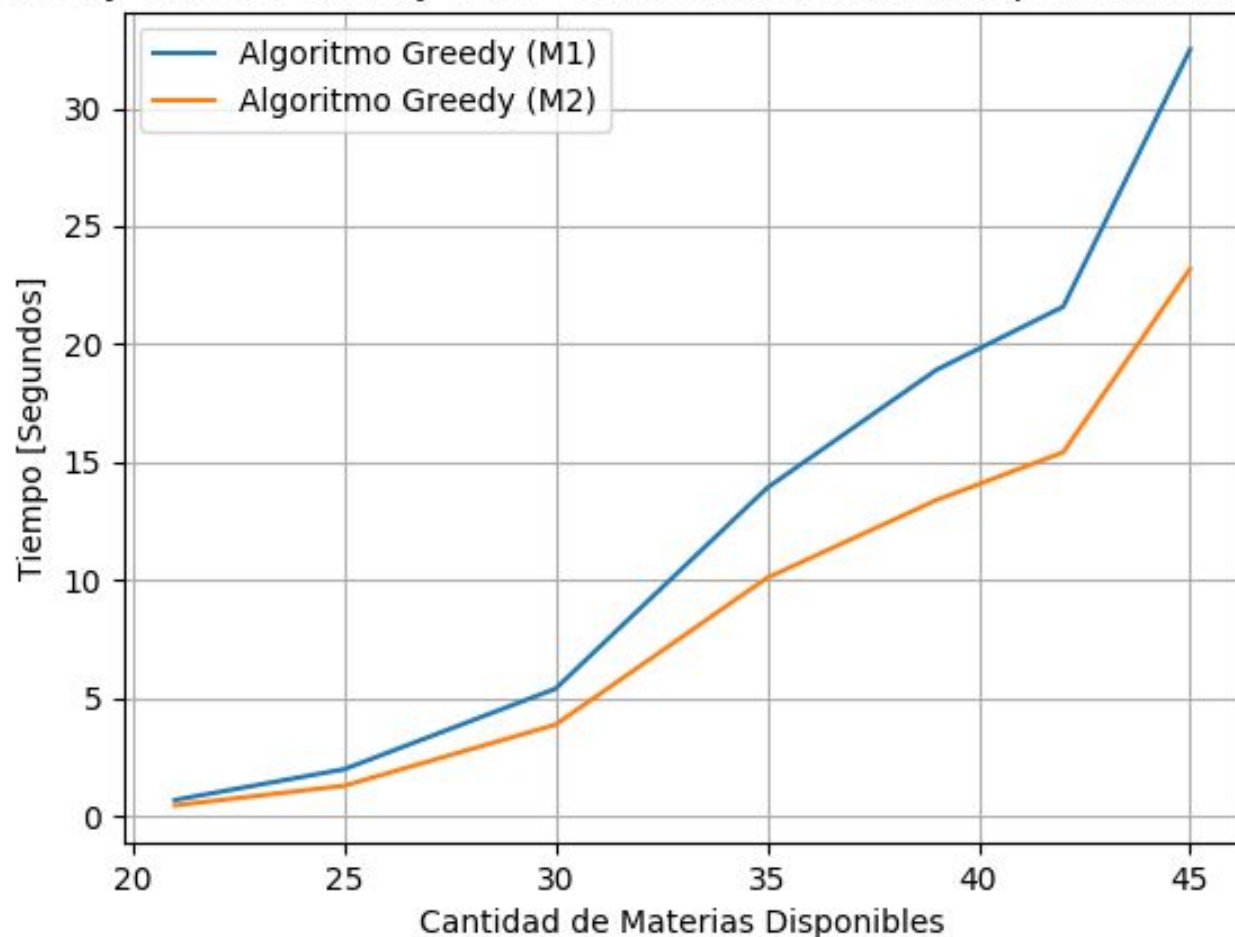




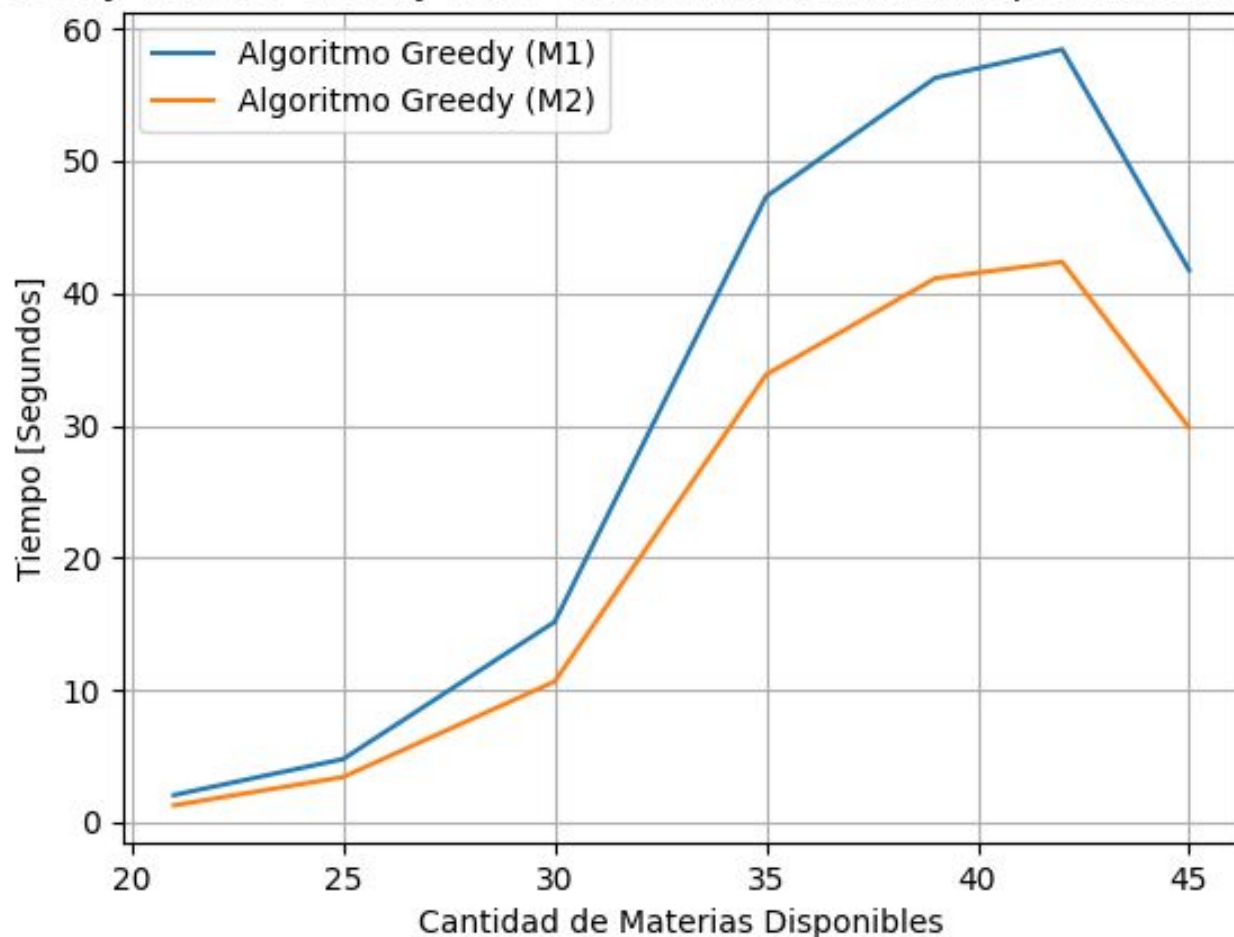
## PLE (M1) vs. Greedy (M1) - Máx. cant. de materias por cuatrimestre: 5



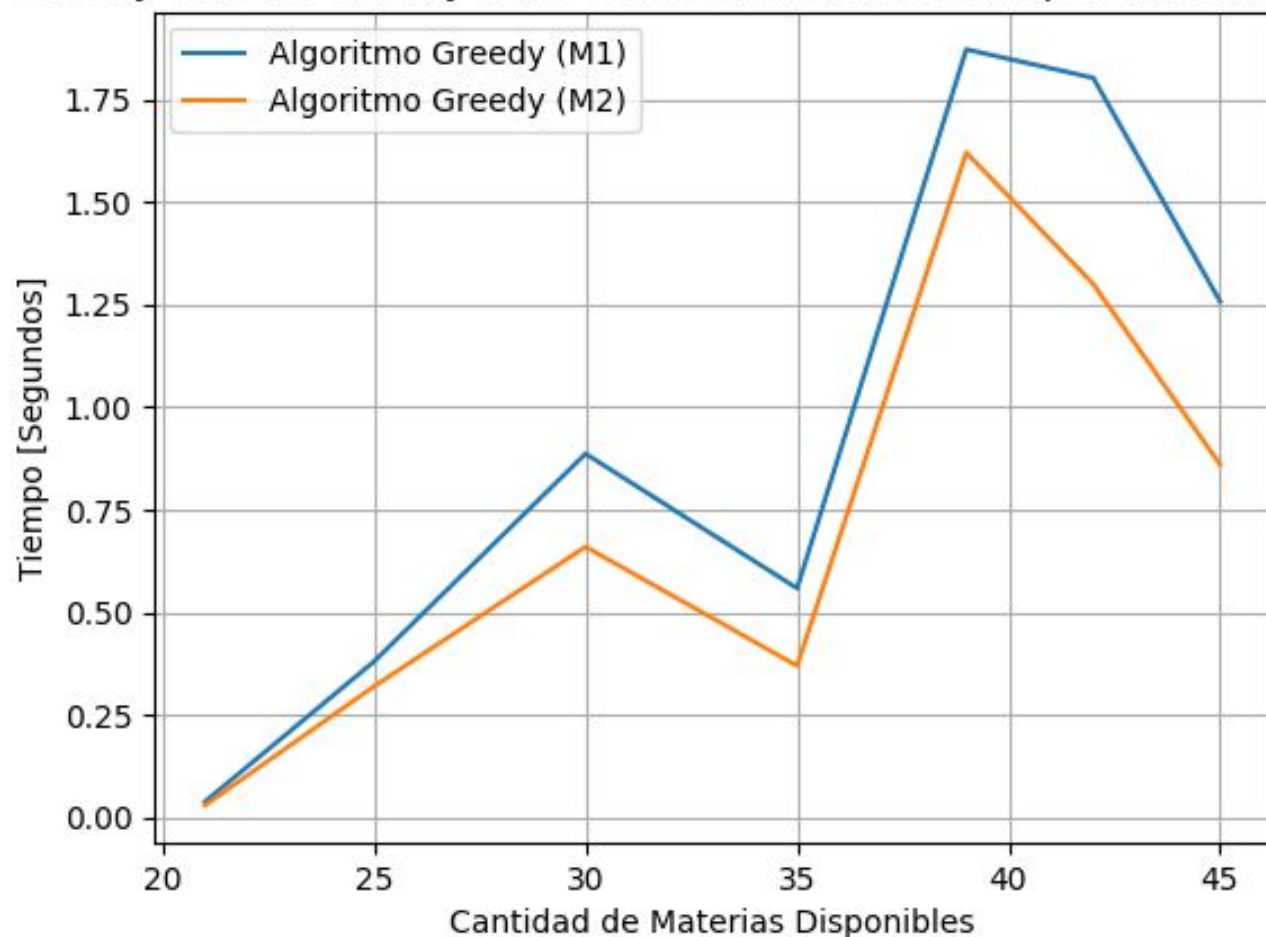
## Greedy (M1) vs. Greedy (M2) - Máx. cant. de materias por cuatrimestre: 2



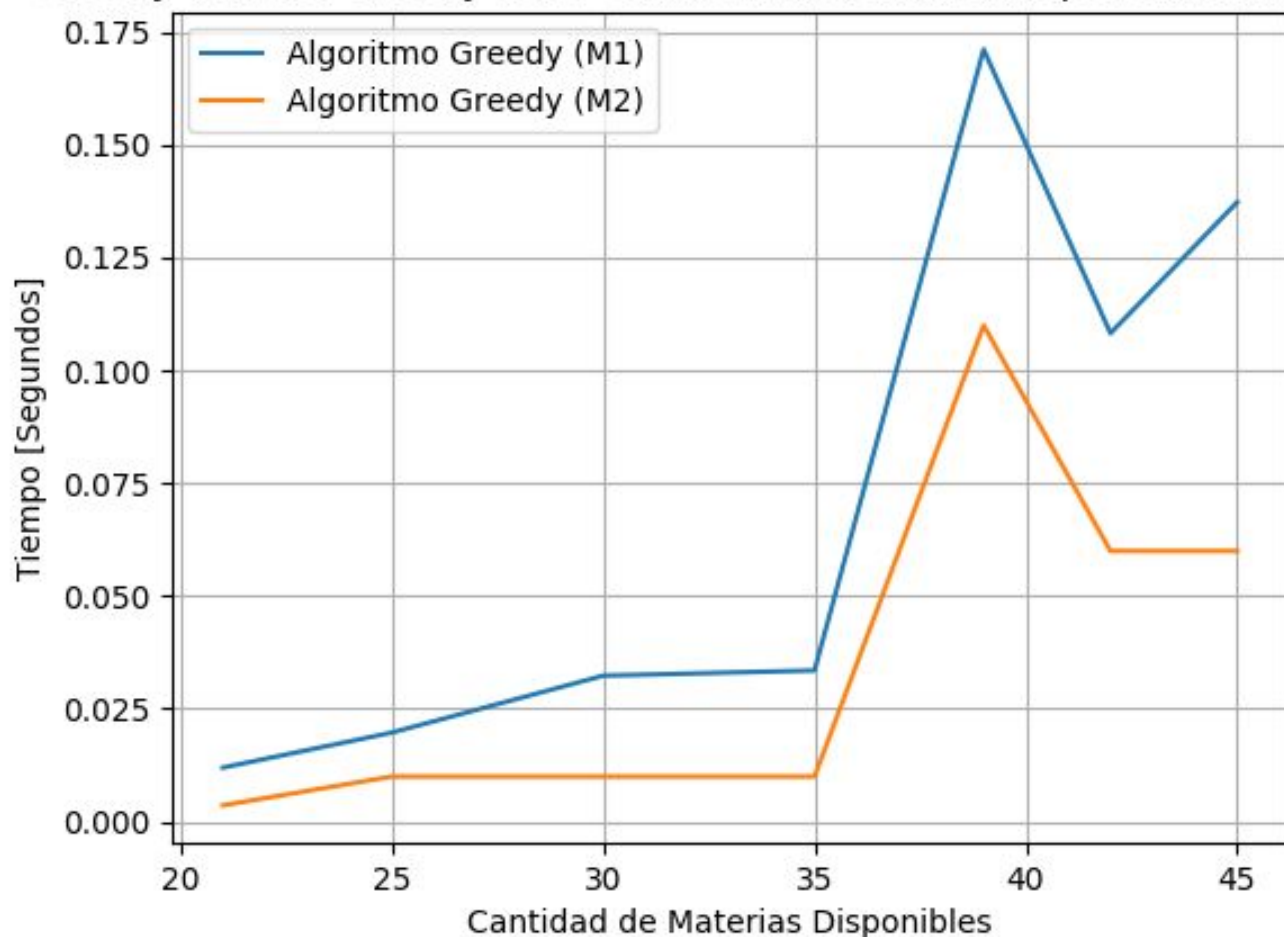
### Greedy (M1) vs. Greedy (M2) - Máx. cant. de materias por cuatrimestre: 3



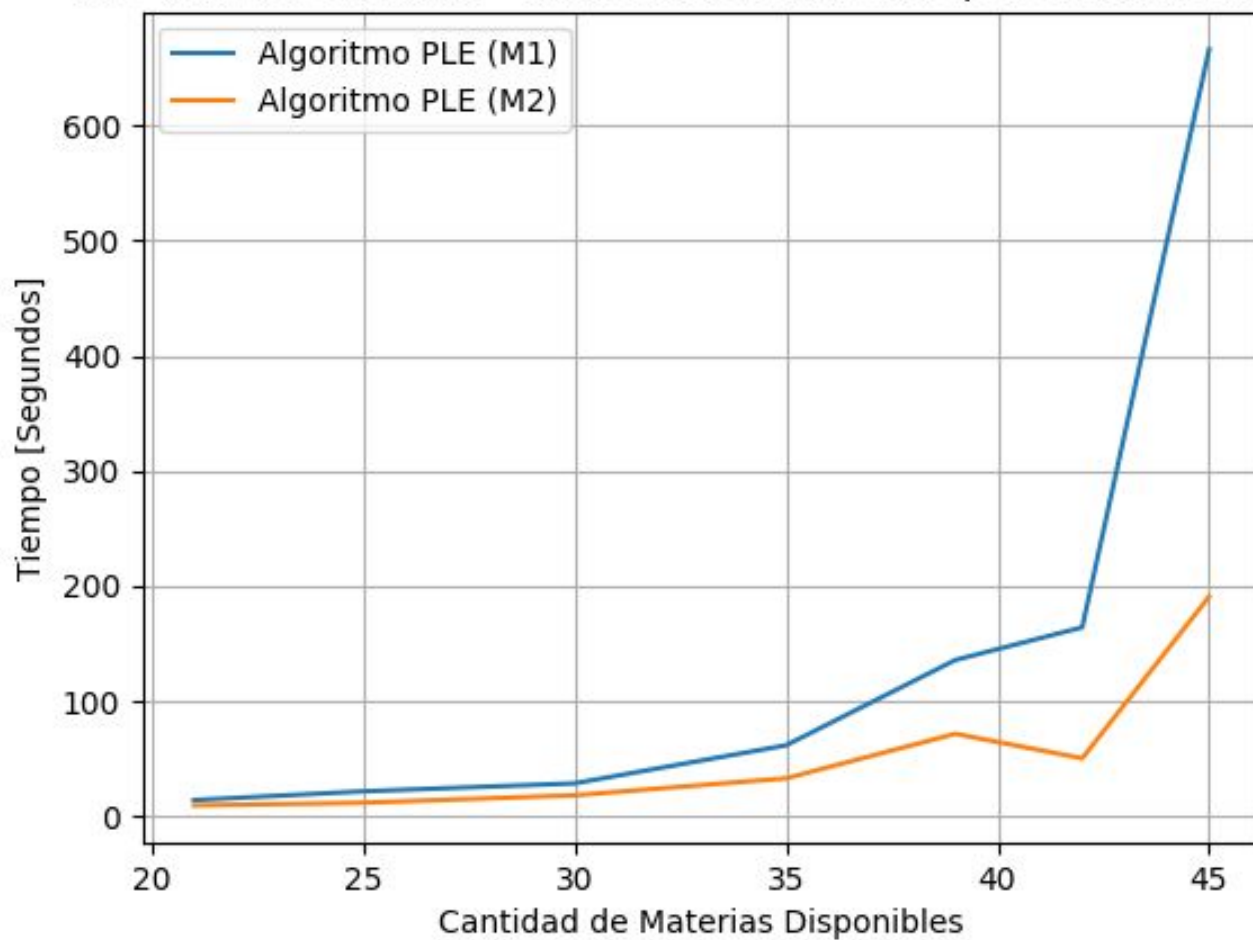
## Greedy (M1) vs. Greedy (M2) - Máx. cant. de materias por cuatrimestre: 4



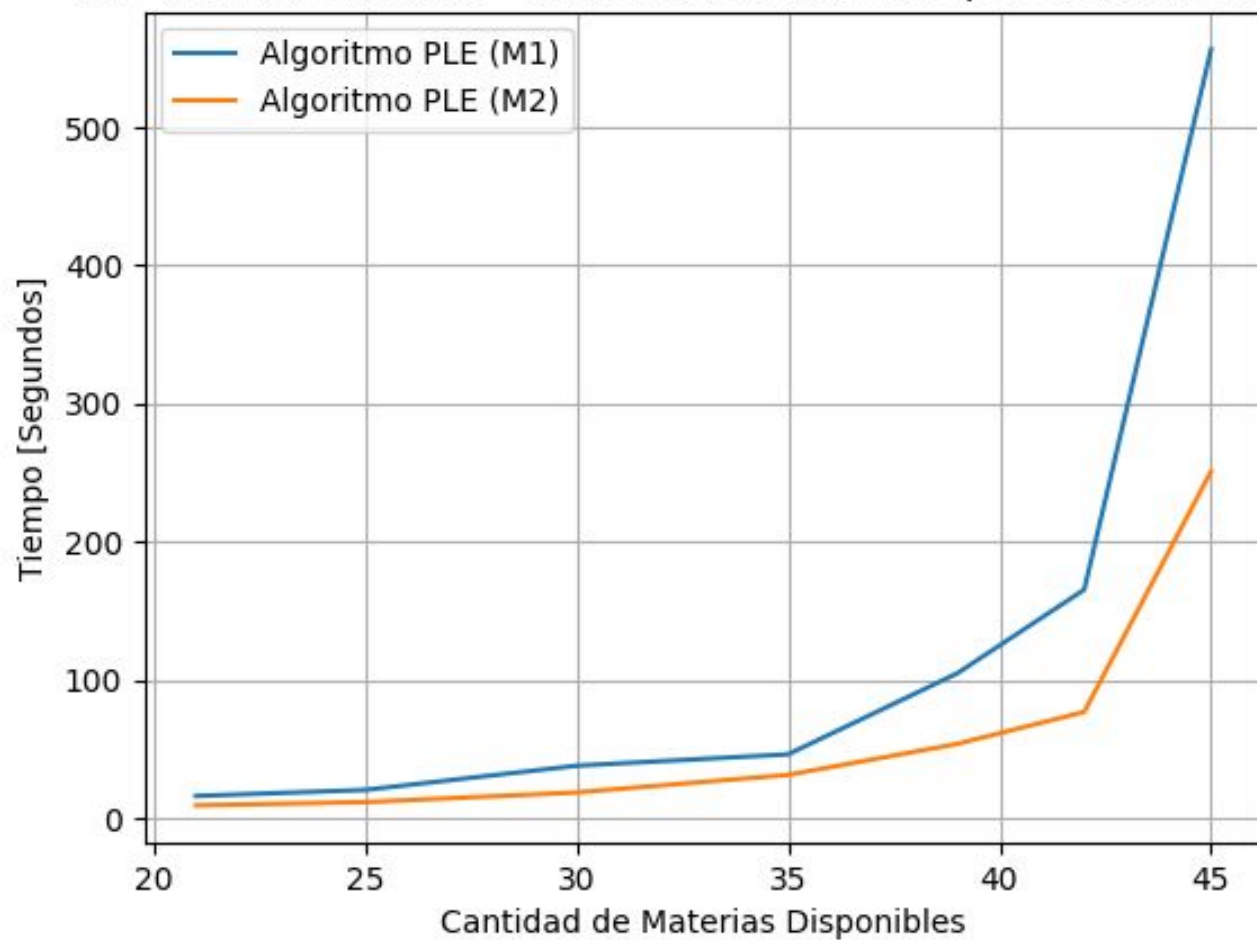
## Greedy (M1) vs. Greedy (M2) - Máx. cant. de materias por cuatrimestre: 5



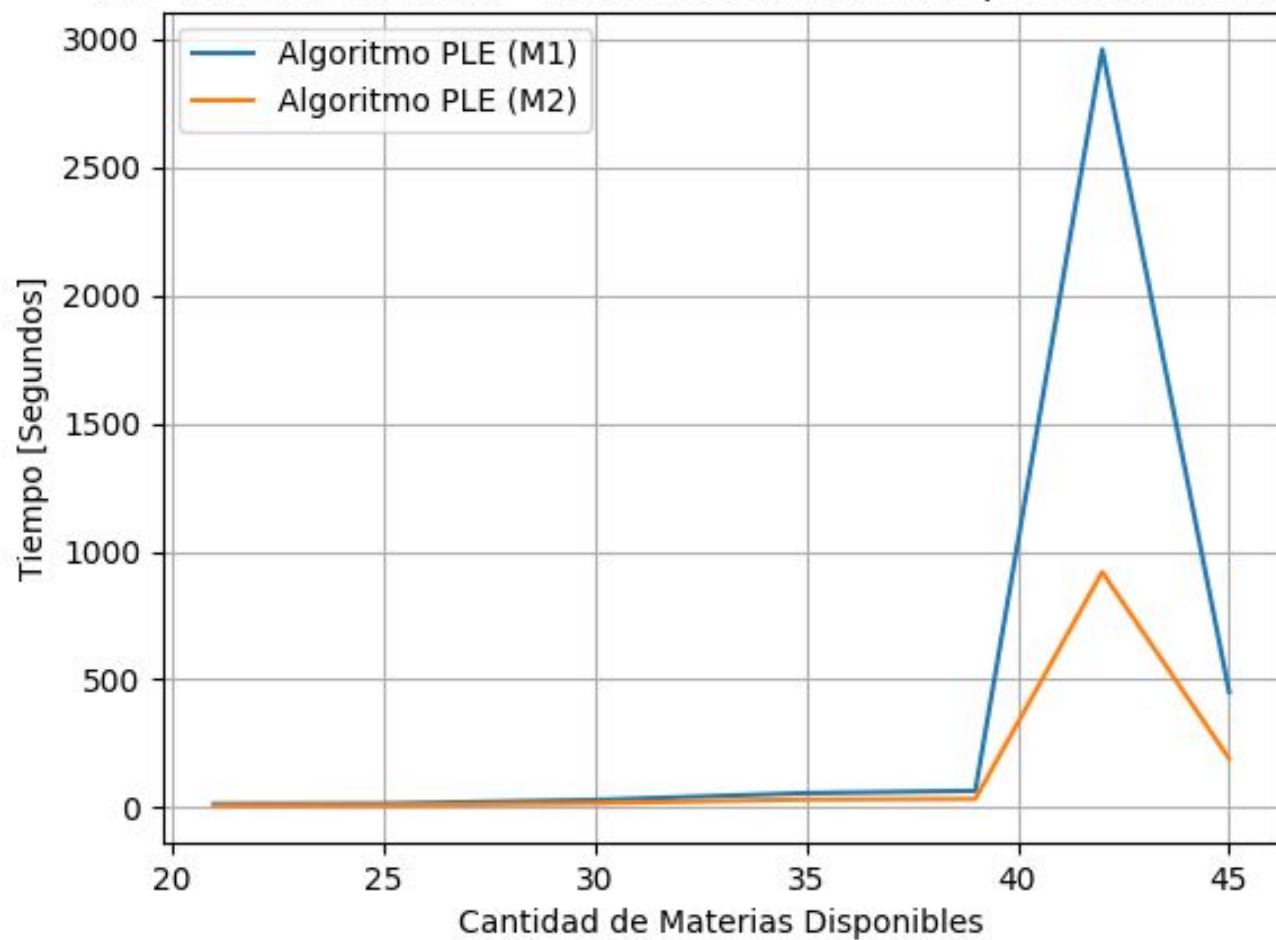
## PLE (M1) vs. PLE (M2) - Máx. cant. de materias por cuatrimestre: 2



### PLE (M1) vs. PLE (M2) - Máx. cant. de materias por cuatrimestre: 3

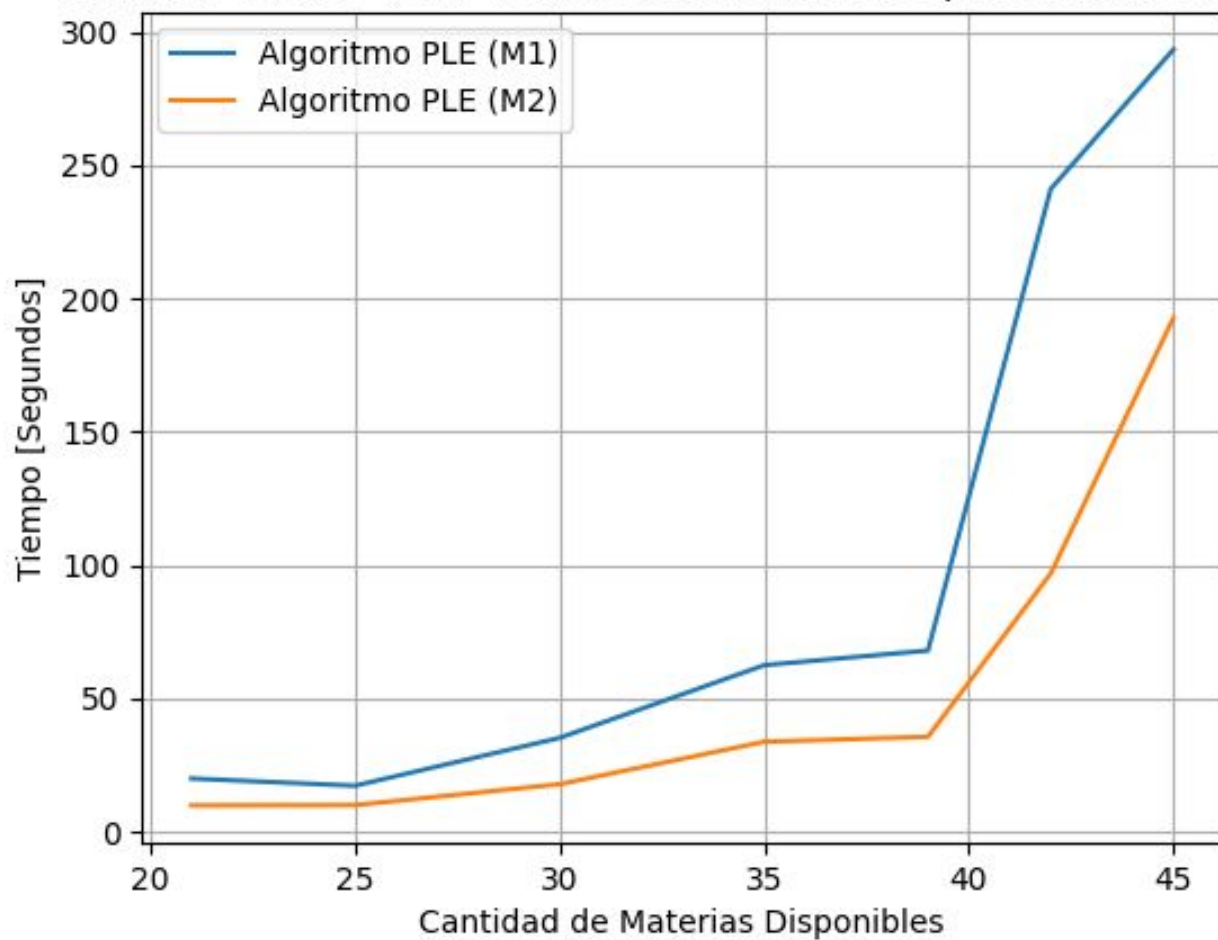



PLE (M1) vs. PLE (M2) - Máx. cant. de materias por cuatrimestre: 4





PLE (M1) vs. PLE (M2) - Máx. cant. de materias por cuatrimestre: 5





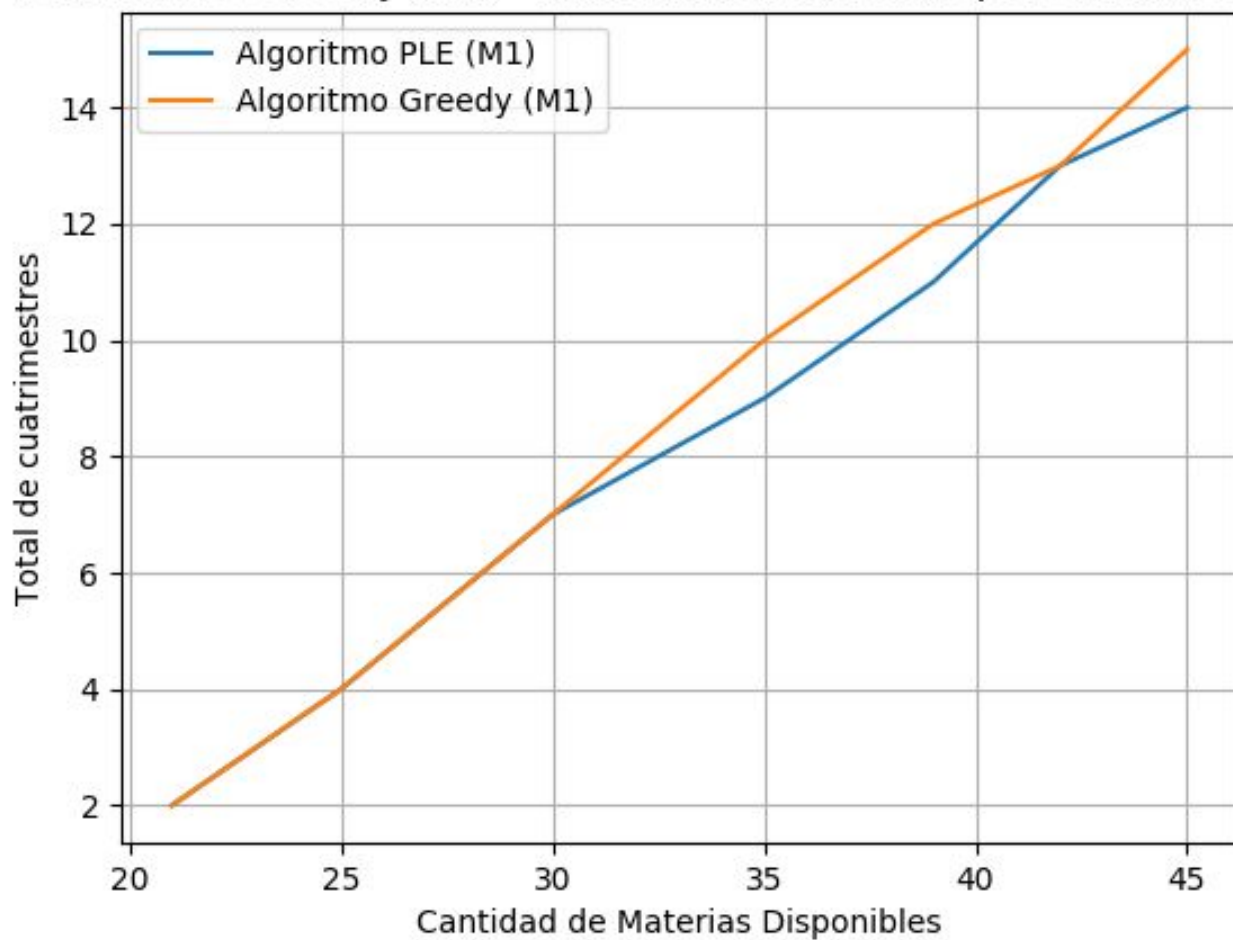
## Algoritmo Greedy

Cantidad de materias Disponibles	42
Max cantidad de materias por cuatrimestre	3
Duración del plan (total de cuatrimestres)	9
Tiempo Máquina M1	58.43 segundos
Tiempo Máquina M2	42.37 segundos

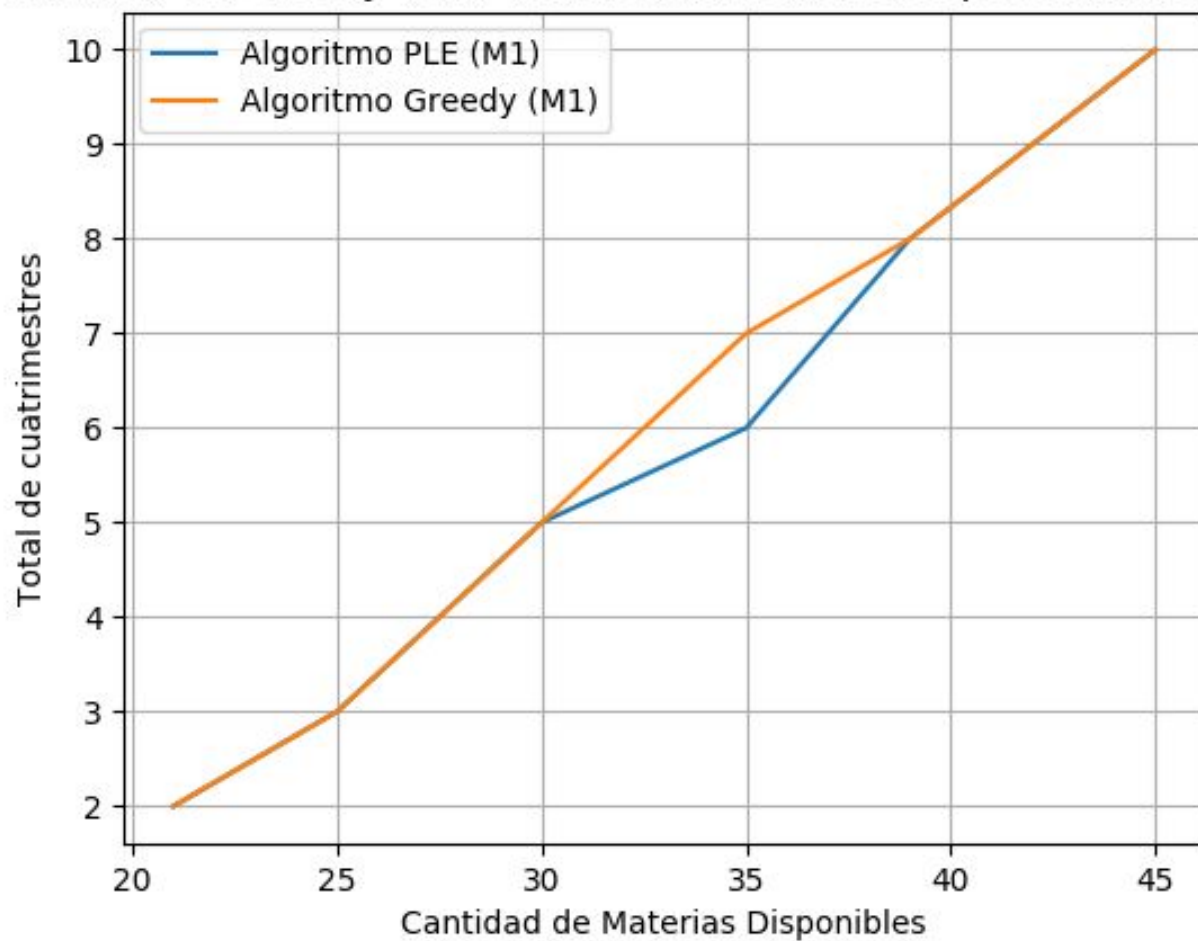
## Algoritmo PLE

Cantidad de materias Disponibles	42
Max cantidad de materias por cuatrimestre	4
Duración del plan (total de cuatrimestres)	7
Tiempo Máquina M1	49 minutos, 17.57 segundos
Tiempo Máquina M2	15 minutos, 19.74 segundos

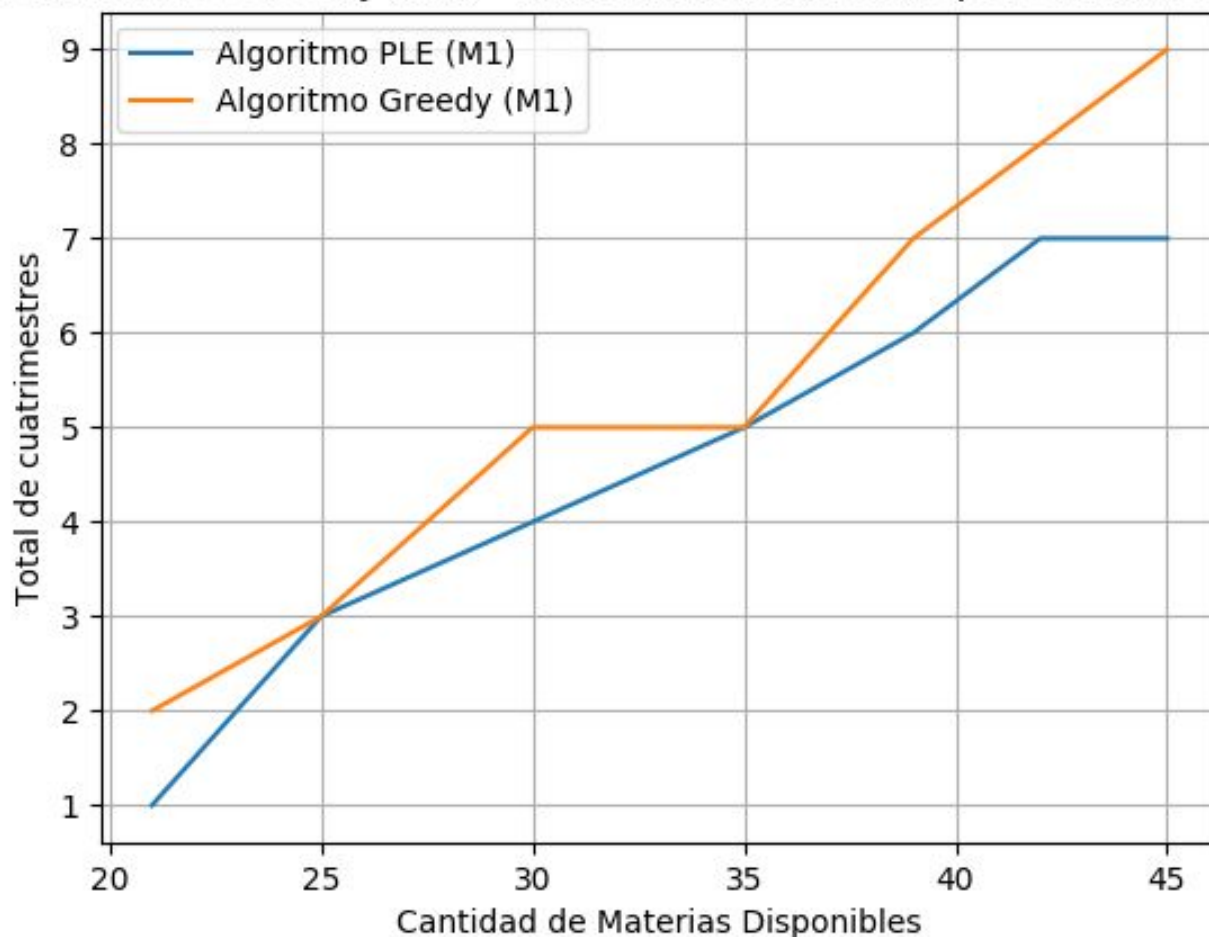
## PLE (M1) vs. Greedy (M1) - Máx. cant. de materias por cuatrimestre: 2



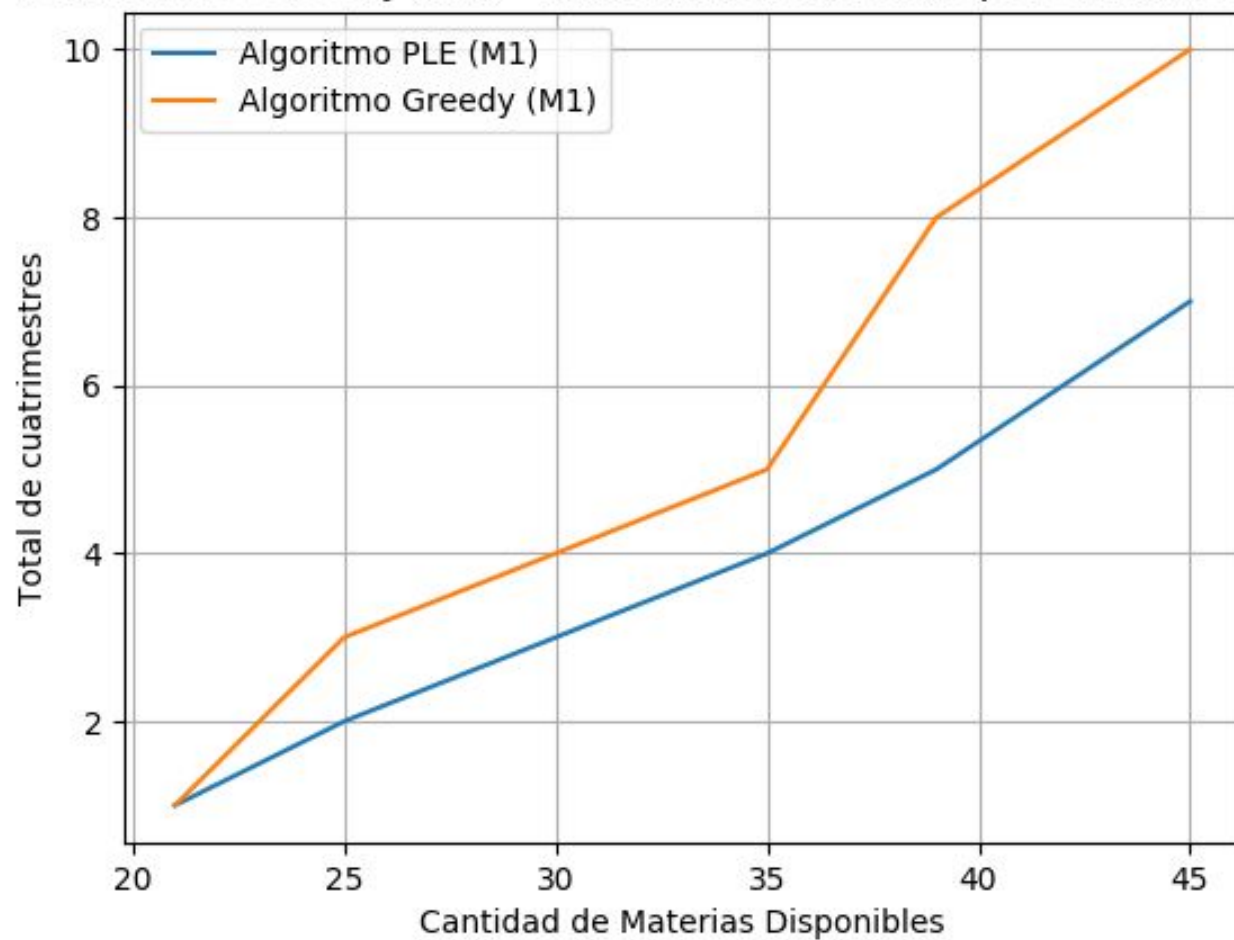
## PLE (M1) vs. Greedy (M1) - Máx. cant. de materias por cuatrimestre: 3



## PLE (M1) vs. Greedy (M1) - Máx. cant. de materias por cuatrimestre: 4



## PLE (M1) vs. Greedy (M1) - Máx. cant. de materias por cuatrimestre: 5





# Pruebas

- Greedy es más rápido que PLE.
- Greedy obtiene muy buenos resultados. Su diferencia más frecuente es de 1 cuatrimestre con un máximo de 2.
- PLE es siempre óptimo pero lento
- Greedy considera cuatrimestre actual y a lo sumo el inmediato siguiente. No todo el futuro.



# Pruebas

- Agregar nuevas restricciones es mucho más sencillo en PLE.
- Como el solver CBC es multithread, el tiempo de ejecución está fuertemente afectado por la cantidad de CPUs.

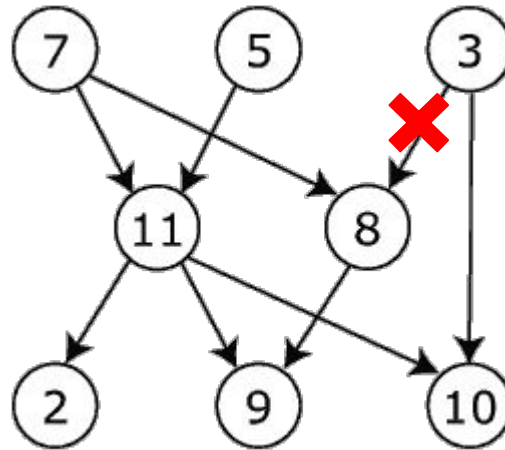





# Mejoras y Nuevas Funcionalidades Futuras



Carreras faltantes



Excepciones  
de  
Correlatividades



# Mejoras y Nuevas Funcionalidades Futuras



Calendario con  
fechas  
importantes



Superposición de  
Horarios



Mensajería

# Mejoras y Nuevas Funcionalidades Futuras



Rol Docente  
Rol Administrador  
Departamento



# Conclusiones

- Uso de tecnologías que si bien eran desconocidas tuvieron una curva de aprendizaje rápida.
- Desconocimiento de cómo se realizaba el procesamiento asincrónico de tareas en un sistema distribuido.
- Modelo de datos con interrelaciones complejas.





# Conclusiones

- Flask-User → Bueno y malo al mismo tiempo
- Diferencia con las estimaciones iniciales
  - ¿Mejor solo o acompañado?



# Conclusiones

- Algoritmo Greedy con buenos resultados pero no óptimos.
- PLE lento (máx. 15 minutos) pero con resultados óptimos.
- Hash de los planes para acotar los tiempos!





# Agradecimientos

- A todos los aquí presentes
- A mi familia de siempre, y a mi familia adquirida. En especial a Lorena, Verónica, Mariel y Nico.
- A mis amigos y compañeros de la facultad. En especial a Nico, Martín, Flor, Eze y Javier.
- A todos mis compañeros de grupo con los que intenté hacer el TP antes...
- A los miembros de la CC de Lic. en Sistemas, de la cual formé parte.



# Agradecimientos

- A los chicos (y no tan chicos) de Casa Informática.
- A mis ex-alumnos de algoritmos.
- A mis tutores, Diego y Rosita.
- A todos los Wachencholdiers!
- A mi pareja, Ariel.



# ¿Preguntas?



[illegible]