Informe de Entrega Final

Universidad Nacional General Sarmiento

Modelado y Optimización

Profesor:

Marcelo Mydlarz

Integrantes:

Juan Manuel Losada

Facundo Ruiz

Matías Morales

Vanesa Vera

Índice

0. Introducción

En este informe se presentan los resultados obtenidos tras la resolución de los problemas planteados en el Trabajo Práctico de Modelado y Optimización. El análisis se centra en las partes 4 a 7, que abordan la resolución integral del problema denominado *El Desafío* mediante diferentes enfoques de optimización.

El problema central consiste en la optimización de procesos de picking en centros de distribución, donde se debe seleccionar un conjunto de pasillos y órdenes que maximice la productividad del proceso de recolección, respetando restricciones de capacidad y límites operacionales.

Descripción general del problema

El *Desafío* aborda la selección óptima de pasillos y órdenes en un almacén para maximizar la eficiencia del proceso de picking. Los elementos clave del problema incluyen:

- **Órdenes (O):** Conjunto de pedidos que requieren elementos específicos en cantidades determinadas.
- **Elementos (I):** Productos almacenados que pueden ser requeridos por las órdenes.
- **Pasillos (A):** Ubicaciones físicas donde están disponibles los elementos con capacidades limitadas.
- **Límites operacionales:** Restricciones de límite inferior (LB) y superior (UB) para la cantidad total de unidades a recolectar.

Enfoques implementados

- **Parte 4:** Implementación de un solucionador básico con exploración sistemática de diferentes cantidades de pasillos.
- **Parte 5:** Extensión mediante generación de columnas para manejar instancias de mayor escala.
- **Parte 6:** Evaluación de mejoras específicas al modelo de generación de columnas.
- **Parte 7:** Implementación de enfoques híbridos y heurísticos para instancias complejas.

1. Parte 4

(a) **Modelos matemáticos utilizados:**

I. **Descripción del problema que resuelve cada modelo.**

En esta etapa se implementa la clase `Basic` que resuelve el problema *Desafío* mediante un enfoque de fuerza bruta inteligente. El modelo explora sistemáticamente diferentes cantidades de pasillos k y utiliza reutilización de modelos para mejorar la eficiencia computacional.

La clase implementa tres métodos principales:

- `Opt_cantidadPasillosFija(k, umbral)`: Resuelve el problema para exactamente k pasillos dentro del tiempo límite especificado.
- `Opt_PasillosFijos(umbral)`: Optimiza la selección de órdenes para un conjunto fijo de pasillos previamente determinado.
- `Opt_ExplorarCantidadPasillos(umbral)`: Estrategia integral que explora múltiples valores de k basándose en un ranking heurístico.

El enfoque utiliza un modelo maestro reutilizable que se adapta para diferentes valores de k modificando únicamente la restricción de cardinalidad, evitando la recreación completa del modelo en cada iteración.

II. **Definición de variables utilizadas.**

- $x_a \in \{0, 1\}$: variable binaria que indica si el pasillo a es seleccionado.
- $y_o \in \{0, 1\}$: variable binaria que indica si la orden o es seleccionada.
- $W_{o,i}$: cantidad del elemento i requerida por la orden o.
- $S_{a,i}$: capacidad del elemento i disponible en el pasillo a.
- k: número fijo de pasillos a seleccionar.
- LB, UB: límites inferior y superior de unidades totales a recolectar.

III. **Modelo matemático completo.**

El modelo básico para un valor fijo de k se formula como:

$$
\begin{aligned}
\text{máx} \quad & \sum_{o \in O} \sum_{i \in I} W_{o,i} \cdot y_o \\
\text{s.t.} \quad & LB \leq \sum_{o \in O} \sum_{i \in I} W_{o,i} \cdot y_o \leq UB \\
& \sum_{o \in O} W_{o,i} \cdot y_o \leq \sum_{a \in A} S_{a,i} \cdot x_a \quad \forall i \in I \\
& \sum_{a \in A} x_a = k \\
& x_a, y_o \in \{0, 1\} \quad \forall a \in A, o \in O
\end{aligned}
$$

IV. **Explicación de la función objetivo y las restricciones.**

- **Función objetivo:** Maximiza la cantidad total de elementos recolectados de todas las órdenes seleccionadas.
- **Restricciones:**
  - **Límites operacionales:** La cantidad total recolectada debe estar dentro del rango $[LB, UB]$.

- **Capacidad:** Para cada elemento i, la demanda total no puede exceder la capacidad disponible en los pasillos seleccionados.
- **Cardinalidad:** Se seleccionan exactamente k pasillos.
- **Binaridad:** Las variables representan decisiones discretas de selección.

(b) **Pseudocódigo a alto nivel del algoritmo:**

Algorithm 1: Algoritmo de exploración básica (Parte 4)

Datos: umbral_total (tiempo límite total)
Resultado: Mejor solución encontrada

```
Inicializar tiempo_inicio ← tiempo actual;
Inicializar mejor_valor ← −∞;
Inicializar mejor_solucion ← ∅;
Crear ranking_k ← Rankear() ;              // Lista ordenada de valores k prometedores
foreach k en ranking_k do
    tiempo_restante ← umbral_total − (tiempo_actual − tiempo_inicio);
    if tiempo_restante ≤ 0 then
        break ;                                                  // Tiempo agotado
    modelo_k ← modelo_para_k(k) ;          // Reutilizar modelo maestro
    solucion_k ← resolver(modelo_k, tiempo_restante);
    if solucion_k es válida y solucion_k.valor > mejor_valor then
        mejor_valor ← solucion_k.valor;
        mejor_solucion ← solucion_k;
// Etapa de refinamiento
tiempo_restante ← umbral_total − (tiempo_actual − tiempo_inicio);
if tiempo_restante > 0 y mejor_solucion ≠ ∅ then
    pasillos_fijos ← mejor_solucion.pasillos;
    solucion_refinada ← Opt_PasillosFijos(tiempo_restante);
    if solucion_refinada.valor > mejor_valor then
        mejor_solucion ← solucion_refinada;
return mejor_solucion;
```

(c) **Detalles de implementación relevantes:**

- **Reutilización de modelos:** Se construye un modelo maestro una vez y se modifica dinámicamente la restricción de cardinalidad para diferentes valores de k.
- **Función Rankear:** Implementa una heurística simple que ordena los valores de k de 1 a $n\_pasillos$.
- **Gestión de tiempo:** Control estricto del tiempo asignado con distribución proporcional entre diferentes valores de k.
- **Solver utilizado:** SCIP via PySCIPOpt con configuración optimizada para problemas de programación entera mixta.

2. Parte 5

(a) **Modelos matemáticos utilizados:**

I. **Descripción del problema que resuelve cada modelo.**

En esta etapa se implementa la clase `Columns` que resuelve el problema *Desafío* mediante generación de columnas. Este enfoque es especialmente eficaz para instancias grandes donde el número de patrones factibles (combinaciones pasillo-órdenes) es exponencial.

El algoritmo se basa en dos modelos complementarios:

- **Modelo Maestro Restringido (RMP):** Optimiza la selección de columnas desde un conjunto limitado de patrones conocidos.
- **Subproblema de Pricing:** Genera nuevas columnas (patrones) con costo reducido positivo basándose en los valores duales del RMP.

Correcciones implementadas según feedback del profesor:

- Eliminación de la restricción de çobertura por ítem"del modelo maestro.
- Adición de restricciones para evitar seleccionar múltiples columnas del mismo pasillo.
- Eliminación de restricciones artificiales en el subproblema (restricciones 4 y 5).

II. **Definición de variables utilizadas.**

Modelo Maestro:

- $x_j \in \{0, 1\}$: variable binaria que indica si la columna j es seleccionada.
- $ordenes_{j,o} \in \{0, 1\}$: parámetro que indica si la orden o está incluida en la columna j.
- $pasillo_j$: pasillo asociado a la columna j.
- $unidades_j$: total de unidades que aporta la columna j.

Subproblema:

- $z_o \in \{0, 1\}$: variable binaria que indica si la orden o se incluye en la nueva columna.
- $y_a \in \{0, 1\}$: variable binaria que indica si el pasillo a se selecciona para la nueva columna.
- $\pi_k, \alpha_o, \lambda, \beta_a$: valores duales del modelo maestro.

III. **Modelo matemático completo.**

Modelo Maestro Restringido:

$$
\begin{aligned}
\text{máx} \quad & \sum_{j \in J} x_j \cdot unidades_j \\
\text{s.t.} \quad & \sum_{j \in J} x_j = k \quad \text{(Cardinalidad)} \\
& \sum_{j \in J} ordenes_{j,o} \cdot x_j \leq 1 \quad \forall o \in O \quad \text{(Cobertura única)} \\
& \sum_{j \in J} unidades_j \cdot x_j \leq UB \quad \text{(Límite superior)} \\
& \sum_{j \in J : pasillo_j = a} x_j \leq 1 \quad \forall a \in A \quad \text{(Pasillos únicos)} \\
& x_j \in \{0,1\} \quad \forall j \in J
\end{aligned}
$$

Subproblema de Pricing:

$$
\begin{aligned}
\text{máx} \quad & \sum_{o \in O} \left(u_o - \sum_{i \in I} \pi_i W_{o,i} - \lambda u_o - \alpha_o \right) z_o - \gamma_k \sum_{a \in A} y_a \\
\text{s.t.} \quad & \sum_{a \in A} y_a = 1 \quad \text{(Un pasillo)} \\
& \sum_{o \in O} W_{o,i} z_o \leq \sum_{a \in A} S_{a,i} y_a \quad \forall i \in I \quad \text{(Capacidad)} \\
& z_o, y_a \in \{0,1\} \quad \forall o \in O, a \in A
\end{aligned}
$$

IV. **Explicación de la función objetivo y las restricciones.**
Modelo Maestro:

- **Función objetivo:** Maximiza el total de unidades recolectadas por las columnas seleccionadas.
- **Restricciones:**
  - Seleccionar exactamente k columnas/pasillos.
  - Cada orden cubierta por máximo una columna.
  - No exceder el límite superior de unidades.
  - No seleccionar múltiples columnas del mismo pasillo (NUEVA).

Subproblema:

- **Función objetivo:** Maximiza el costo reducido de la nueva columna considerando los valores duales.
- **Restricciones:**
  - Seleccionar exactamente un pasillo para la nueva columna.
  - Respetar la capacidad disponible de cada elemento en el pasillo seleccionado.
  - Variables binarias para decisiones discretas.

(b) **Pseudocódigo a alto nivel del algoritmo:**

Algorithm 2: Algoritmo de generación de columnas (Parte 5)

```
Datos: k (número de pasillos), umbral (tiempo límite)
Resultado: Solución óptima para k pasillos
Inicializar columnas iniciales: una por pasillo;
while tiempo_restante >0 do
    // Resolver modelo maestro
    modelo_maestro ← construir_RMP(columnas_actuales, k);
    solucion_maestro ← resolver(modelo_maestro);
    if modelo_maestro no es óptimo then
        break;
    // Extraer valores duales
    duales ← extraer_duales(modelo_maestro);
    // Resolver subproblema de pricing
    nueva_columna ← resolver_subproblema(duales);
    if costo_reducido ≤ 0 then
        // Convergencia alcanzada
        break;
    // Agregar nueva columna al modelo maestro
    agregar_columna(nueva_columna);
    // Verificar si es columna repetida (indicaría error)
    if nueva_columna es repetida then
        imprimir("[ADVERTENCIA] Columna repetida generada");
        break;
return solucion_final;
```

(c) **Detalles de implementación relevantes:**

- **Columnas iniciales:** Se genera una columna por pasillo, cada una conteniendo las órdenes factibles para ese pasillo específico.
- **Detección de convergencia:** El algoritmo termina cuando el costo reducido del subproblema es ≤ 0 o se agota el tiempo.
- **Gestión de duales:** Extracción automática de valores duales: cardinalidad, órdenes, límite UB y pasillos únicos.
- **Restricciones eliminadas:** Ya no se incluyen restricciones de límite UB ni .[a]l menos una orden.[en] el subproblema, según corrección del profesor.

3. Parte 6

(a) **Modelos matemáticos utilizados:**

I. **Descripción del problema que resuelve cada modelo.**

En esta etapa se evalúan mejoras específicas al modelo de generación de columnas de la Parte 5. Se mantiene la estructura fundamental del algoritmo pero se implementan tres variantes independientes para optimizar diferentes aspectos:

1. **Columnas iniciales eficientes:** Mejora la inicialización seleccionando pasillos con mayor capacidad total en lugar de usar todos los pasillos.
2. **Función Rankear mejorada:** Utiliza clustering (KMeans) para agrupar pasillos por capacidad y priorizar valores de k más prometedores.
3. **Eliminación de columnas inactivas:** Remueve periódicamente columnas no utilizadas para reducir el tamaño del modelo maestro.

Cada variante se evalúa por separado contra el modelo base para determinar su impacto individual en el rendimiento.

II. **Definición de variables utilizadas.**

Las variables del modelo son idénticas a la Parte 5. Las mejoras se centran en la gestión del conjunto de columnas J y la estrategia de exploración:

- $x_j \in \{0, 1\}$: selección de columna j.
- $J_{activo} \subseteq J$: subconjunto de columnas activas (para variante 3).
- $capacidad\_total_a$: métrica de capacidad para el pasillo a (para variantes 1 y 2).
- $k_{prometedores}$: valores de k priorizados por clustering (para variante 2).

III. **Modelo matemático completo.**

El modelo matemático base es idéntico al de la Parte 5. Las mejoras modifican:

Variante 1 - Columnas iniciales eficientes:

$$J_{inicial} = \{j : pasillo_j \in top\_pasillos\_por\_capacidad\}$$

Variante 2 - Rankear mejorado:

$$k_{ranking} = clustering\_kmeans(capacidades\_pasillos, n\_clusters)$$

Variante 3 - Eliminación de columnas:

$$J_{t+1} = J_t \setminus \{j : x_j = 0 \text{ en últimas } n \text{ iteraciones}\}$$

IV. **Explicación de la función objetivo y las restricciones.**

La función objetivo y restricciones son idénticas a la Parte 5. Las mejoras impactan en:

- **Eficiencia computacional:** Reducción del tamaño del modelo maestro.
- **Calidad de inicialización:** Mejor punto de partida para el algoritmo.
- **Estrategia de exploración:** Priorización inteligente de valores de k.

(b) **Pseudocódigo a alto nivel del algoritmo:**

Algorithm 3: Framework de evaluación de mejoras (Parte 6)

Datos: Configuración de mejoras activadas
Resultado: Comparación de métricas entre variantes

```
1  foreach instancia en conjunto_pruebas do
2      foreach variante en [Base, Mejora1, Mejora2, Mejora3] do
3          if variante == Mejora1 then
4              columnas_iniciales ← seleccionar_mejores_pasillos();
5          else if variante == Mejora2 then
6              ranking_k ← clustering_capacidades();
7          else if variante == Mejora3 then
8              activar_eliminacion_columnas();
9          resultado ← ejecutar_generacion_columnas();
10         metricas[instancia][variante] ← extraer_metricas(resultado);
11 generar_tabla_comparativa(metricas);
```

(c) **Detalles de implementación relevantes:**

- **Selección de pasillos:** Se ordenan por capacidad total y se seleccionan los top-k para inicialización.
- **Clustering KMeans:** Agrupa pasillos en clusters por capacidad para guiar la exploración de valores k.
- **Eliminación adaptativa:** Remueve columnas no utilizadas cada 5 iteraciones para mantener el modelo compacto.
- **Métricas comparativas:** Se evalúan: tiempo de ejecución, número de columnas generadas, valor objetivo final, y convergencia.

4. Parte 7

(a) **Modelos matemáticos utilizados:**

I. **Descripción del problema que resuelve cada modelo.**

En esta etapa se implementa un enfoque híbrido que combina el mejor modelo de la Parte 6 con un solucionador heurístico (`FastSolver`) para abordar instancias de diferentes escalas:

- **Instancias pequeñas/medianas:** Se utiliza el modelo exacto de generación de columnas con las mejores optimizaciones de la Parte 6.
- **Instancias grandes:** Se aplica `FastSolver`, un algoritmo heurístico greedy con búsqueda local para obtener soluciones de buena calidad en tiempo limitado.

FastSolver implementa una heurística constructiva seguida de mejoramiento local:

1) **Fase constructiva:** Selección greedy de pasillos y órdenes basada en ratios beneficio/capacidad.
2) **Fase de mejora:** Búsqueda local mediante intercambio de órdenes para mejorar la solución inicial.

II. **Definición de variables utilizadas.**

Modelo exacto (Generación de columnas): Variables idénticas a las Partes 5-6.

FastSolver (Heurístico):

- $pasillos\_seleccionados$: lista de índices de pasillos elegidos.
- $ordenes\_seleccionadas$: lista de índices de órdenes elegidas.
- $capacidad\_disponible[i]$: capacidad restante del elemento i tras selecciones.
- $valor\_objetivo$: suma total de unidades recolectadas.
- $ratio\_beneficio[o]$: métrica de atractivo de la orden o.

III. **Modelo matemático completo.**

Para el modelo exacto: Idéntico al modelo corregido de la Parte 5.

Para FastSolver: El algoritmo heurístico no utiliza formulación matemática explícita, sino que implementa las siguientes reglas de decisión:

Selección de pasillos:

$$pasillo^* = \arg\max_{a \in A} \left\{ \sum_{i \in I} S_{a,i} \right\}$$

Selección de órdenes:

$$orden^* = \arg\max_{o \in O_{factible}} \left\{ \frac{\sum_{i \in I} W_{o,i}}{max\_capacidad\_requerida(o)} \right\}$$

Criterio de factibilidad:

$$O_{factible} = \{o \in O : W_{o,i} \leq capacidad\_disponible[i] \quad \forall i \in I\}$$

IV. **Explicación de la función objetivo y las restricciones.**

Modelo exacto: Idéntico a la Parte 5 con las correcciones implementadas.

FastSolver:

- **Función objetivo implícita:** Maximizar el total de unidades recolectadas mediante decisiones greedy localmente óptimas.
- **Restricciones manejadas heurísticamente:**
  - **Capacidad:** Se verifica factibilidad antes de cada selección.
  - **Límites LB/UB:** Se monitorean durante la construcción de la solución.
  - **Cardinalidad:** Se controla el número de pasillos seleccionados.

(b) **Pseudocódigo a alto nivel del algoritmo:**

Algorithm 4: Algoritmo híbrido (Parte 7)

```
Datos: Instancia del problema, umbral de tiempo
Resultado: Mejor solución encontrada
// Clasificación de instancia
tamaño ← clasificar_instancia(num_ordenes, num_pasillos);
if tamaño == "pequeña." tamaño == "mediana" then
    // Usar modelo exacto mejorado
    solucion ← generacion_columnas_mejorada(umbral);
else
    // Usar FastSolver heurístico
    solucion ← FastSolver(umbral);
return solucion;
Procedure FastSolver(umbral):
    // Fase 1: Construcción greedy
    pasillos ← seleccionar_mejores_pasillos_greedy();
    ordenes ← [];
    capacidad_restante ← calcular_capacidad_total(pasillos);
    while existen_ordenes_factibles() do
        orden_mejor ← seleccionar_orden_mayor_ratio();
        agregar_orden(orden_mejor);
        actualizar_capacidad(orden_mejor);
    // Fase 2: Búsqueda local
    repeat
        orden_out, orden_in ← encontrar_mejor_intercambio();
        if intercambio_mejora_solucion() then
            realizar_intercambio(orden_out, orden_in);
    until no_hay_mejora o tiempo_agotado;
    return construir_solucion_final();
```

(c) **Detalles de implementación relevantes:**

- **Clasificación automática:** Las instancias se clasifican por tamaño basándose en número de órdenes, elementos y pasillos.
- **FastSolver optimizado:** Implementa estructuras de datos eficientes para verificación rápida de factibilidad.
- **Búsqueda local inteligente:** La fase de mejora evalúa intercambios que respeten todas las restricciones.

- **Gestión de memoria:** Para instancias grandes, se evita la construcción de matrices completas cuando es posible.
- **Criterios de parada:** El algoritmo termina por convergencia, tiempo agotado, o imposibilidad de mejora.

5. Resultados en tablas

En esta sección se presentan los resultados experimentales obtenidos en las evaluaciones de las Partes 6 y 7.

5.1. Parte 6 - Evaluación de mejoras

En la siguiente tabla se muestran los resultados de evaluar tres estrategias de mejora del modelo de generación de columnas de la Parte 5. Las métricas incluyen número de restricciones, variables iniciales y finales, cota dual, y mejor valor objetivo encontrado.

Abreviaturas utilizadas:

- **Col. iniciales:** Mejora en la selección de columnas iniciales
- **Rankear:** Mejora en la función de ranking usando clustering
- **Elim col.:** Eliminación de columnas no utilizadas
- **# var:** Número de variables
- **últ. maestro:** Variables en la última iteración del modelo maestro

Análisis de resultados Parte 6:

- La estrategia de eliminación de columnas inactivas mostró mejora en la instancia input_0003 (objetivo 75 vs 68).
- Las instancias pequeñas (input_0001, input_0002) no mostraron diferencias significativas entre estrategias.
- La instancia input_0004 no pudo resolverse con ninguna estrategia en el tiempo límite.

Instancia	Métrica	Parte 5	Col. iniciales	Rankear	Elim col.
input_0001	Restricciones	157	157	157	157
	# var iniciales	17	17	17	17
	# var últ. maestro	18	17	18	17
	Cota dual	51.0	51.0	51.0	51.0
	Mejor objetivo	68	68	68	68
input_0002	Restricciones	9	9	9	9
	# var iniciales	6	6	6	6
	# var últ. maestro	6	6	6	6
	Cota dual	8.0	8.0	8.0	8.0
	Mejor objetivo	2	2	2	2
input_0003	Restricciones	248	248	248	248
	# var iniciales	14	14	14	14
	# var últ. maestro	15	15	15	14
	Cota dual	50.0	50.0	50.0	50.0
	Mejor objetivo	68	68	68	75
input_0004	Restricciones	0	0	0	0
	# var iniciales	0	0	0	0
	# var últ. maestro	0	0	0	0
	Cota dual	0.0	0.0	0.0	0.0
	Mejor objetivo	0	0	0	0

Cuadro 1: Comparación de mejoras en generación de columnas - Parte 6

5.2. Parte 7 - Comparación con enfoque híbrido

La siguiente tabla compara el mejor modelo de la Parte 6 con el enfoque híbrido de la Parte 7, evaluado sobre instancias de diferentes conjuntos de datos.

Instancia	**Métrica**	**Mejor Parte 6**	**Parte 7**
input_0001	Restricciones	157	157
	# variables iniciales	17	17
	# variables últ. maestro	17	17
	Cota dual	51.0	51.0
	Mejor objetivo	68	68
input_0002	Restricciones	9	9
	# variables iniciales	6	6
	# variables últ. maestro	6	6
	Cota dual	8.0	8.0
	Mejor objetivo	2	2
input_0003	Restricciones	248	248
	# variables iniciales	14	14
	# variables últ. maestro	14	14
	Cota dual	50.0	50.0
	Mejor objetivo	75	75
input_0005	Restricciones	0	2625
	# variables iniciales	0	161
	# variables últ. maestro	0	161
	Cota dual	0.0	1250.0
	Mejor objetivo	0	1250

Cuadro 2: Comparación entre mejor Parte 6 y enfoque híbrido Parte 7

Análisis de resultados Parte 7:

- Para instancias pequeñas y medianas, ambos enfoques obtienen resultados idénticos.
- Para la instancia grande (input_0005), solo el enfoque híbrido logró encontrar una solución factible.
- El FastSolver permitió abordar instancias que el modelo exacto no podía resolver en tiempo limitado.
- La clasificación automática de instancias permitió seleccionar el método más apropiado para cada caso.

6. Conclusiones

Dificultades encontradas

1. **Corrección del modelo matemático**

   El principal desafío fue implementar correctamente las correcciones sugeridas por el profesor. Inicialmente, el modelo maestro incluía una restricción de çobertura por ítem"que era redundante, y el subproblema contenía restricciones artificiales que impedían la convergencia natural del algoritmo.

   Solución implementada: Se eliminó la restricción redundante del modelo maestro y se agregó la restricción de pasillos únicos. En el subproblema se removieron las restricciones de límite UB y .ªl menos una orden", permitiendo que el algoritmo converja teóricamente cuando no existen más columnas mejoradoras.

2. **Generación de columnas repetidas**

   Durante la implementación observamos que el subproblema generaba columnas idénticas para diferentes valores de k, indicando que los valores duales de la restricción de cardinalidad no influían suficientemente en la función objetivo.

   Solución implementada: Se implementó detección automática de columnas repetidas como indicador de problemas en la implementación, y se mejoró la gestión del pool de columnas para reutilización entre diferentes valores de k.

3. **Escalabilidad para instancias grandes**

   El modelo exacto con generación de columnas presentaba tiempos de cómputo prohibitivos para instancias con miles de órdenes y cientos de pasillos. El proceso quedaba detenido indefinidamente sin devolver resultados útiles.

   Solución implementada: Se desarrolló FastSolver, un algoritmo heurístico que combina construcción greedy con búsqueda local. Este enfoque permite obtener soluciones de buena calidad para instancias grandes en tiempo razonable, complementando el modelo exacto para instancias pequeñas y medianas.

4. **Función objetivo y valores duales**

   Existía confusión sobre si la función objetivo debía maximizar elementos totales o elementos por pasillo. La implementación inicial mezclaba ambos enfoques inconsistentemente.

   Solución implementada: Se clarificó que el modelo maximiza elementos totales durante la optimización, mientras que el cálculo elementos/pasillos se realiza en el análisis de resultados. Esto es consistente con los requisitos del TP donde los beneficios son unitarios.

Principales aprendizajes obtenidos

- **Generación de columnas:** Se adquirió comprensión profunda de la técnica de generación de columnas, incluyendo la importancia de la correcta formulación del subproblema de pricing y la interpretación de valores duales. La experiencia práctica mostró cómo restricciones aparentemente lógicas pueden inhibir la convergencia del algoritmo.

- **SCIP y PySCIPOpt:** Se desarrolló experticia en el uso del solver SCIP a través de la interfaz Python PySCIPOpt, incluyendo manejo de parámetros, extracción de valores duales, reutilización de modelos, y optimización de rendimiento para problemas de programación entera mixta.
- **Diseño de algoritmos híbridos:** La experiencia demostró la importancia de combinar métodos exactos y heurísticos según las características de cada instancia. La clasificación automática de problemas permite aprovechar las fortalezas de cada enfoque.
- **Importancia del feedback académico:** Las correcciones del profesor fueron fundamentales para entender los aspectos teóricos sutiles de la generación de columnas que no son evidentes en implementaciones iniciales. La iteración entre implementación y corrección teórica fue clave para el éxito del proyecto.
- **Optimización práctica:** Se aprendió a balancear precisión matemática con eficiencia computacional, implementando técnicas como reutilización de modelos, eliminación de columnas inactivas, y estrategias de inicialización inteligente.
- **Validación experimental:** La importancia de diseñar experimentos controlados para evaluar mejoras, incluyendo la definición de métricas apropiadas, selección de instancias representativas, y análisis estadístico de resultados.

Impacto de las correcciones implementadas

Las correcciones sugeridas por el profesor no solo mejoraron la corrección teórica del modelo, sino que también impactaron positivamente en:

- **Convergencia:** El algoritmo ahora converge naturalmente sin necesidad de restricciones artificiales.
- **Calidad de soluciones:** La eliminación de restricciones redundantes permite explorar un espacio de soluciones más amplio.
- **Eficiencia computacional:** Modelos más simples con menos restricciones se resuelven más rápidamente.
- **Robustez:** El código es más estable y predecible en su comportamiento.

Este proyecto demostró que la implementación exitosa de técnicas avanzadas de optimización requiere no solo conocimiento técnico, sino también comprensión profunda de los fundamentos teóricos subyacentes.