Universidad Diego Portales
Facultad de Ingeniería y Ciencias
Algoritmos Exactos y Metaheurísticas

udp Escuela de Informática y Telecomunicaciones

Tarea 1

Project Manager Pro

Profesor: Victor Reyes.
Ayudante: Johan Fuentes
Integrantes: Jonas Oviedo.
Natalia Romero.
Sección: 01.

Índice

1. Introducción

El presente informe tiene como objetivo analizar la implementación de un problema de optimización de ganancias para la empresa BTFC a través del algoritmo Minimal Forward Checking (MFC).

2. Problema modelado

El problema planteado nos indica la necesidad de maximizar las ganancias en la realización de los proyectos, donde por cada proyecto se tienen tareas asociadas que si se realizan se le considera un costo. A continuación, se presenta el modelamiento del problema como un COP:

2.1. Índices

- i : Proyectos, $i \in 1, 2, 3, .., n$
- k : Tareas, $k \in 1, 2, 3, .., m$

2.2. Parámetros

- B : Budget máximo.
- g_i : Ganancia de cada proyecto i.
- c_k : Costo asociado a cada tarea k.
- A_{ik} : Tareas k asociadas a cada proyecto i.

2.3. Variables de Decisión

- x_i : Variable binaria que vale 1 si realiza proyecto i y 0 en caso contrario.
- y_k : Variable binaria que vale 1 si realiza tarea k y 0 en caso contrario.

2.4. Función Objetivo

$$max \sum_{i=1}^{n} x_i \cdot g_i \tag{2.1}$$

2.5. Restricciones

- Restricción de presupuesto

$$\sum_{i=1}^{m} c_i \cdot x_i \leq B \tag{2.2}$$

- Naturaleza de las variables

$$\begin{aligned} x_i &\in \{1, 2, 3, .., n\} \text{ para todo } i \\ y_k &\in \{1, 2, 3, .., m\} \text{ para todo } k \end{aligned} \tag{2.3}$$

- Asociación de proyectos con tareas

$$x_i \leq y_k \cup A_{ik} = 1, \forall i, k \tag{2.4}$$

3. Minimal Forward Checking

3.1. Algoritmo

Minimal Forward Checking (MFC) es una técnica utilizada en la resolución de problemas de satisfacción de restricciones para encontrar soluciones a problemas complejos que involucran múltiples variables y restricciones. Funciona de la siguiente manera:

1. Selecciona una variable y asigna un valor.
2. Verifica las restricciones de las variables futuras.
3. Aplica reducción de dominio si para una variable no se cumple la restricción.
4. Se genera retroceso cuando la asignación de valores futuros se vuelve inconsistente.
5. Este proceso se repite, moviéndose de una variable a otra, hasta que se encuentre una solución completa que satisfaga todas las condiciones o se determine que no existe tal solución.

Con lo anterior en consideración, se desarrolló nuestra solución. Además, investigamos más de este algoritmo leyendo papers donde encontramos uno titulado “On The Forward Checking Algorithm” [1]. En este trabajo exploran distintas implementaciones de Forward Checking, entre ellas MFC donde se proporciona un pseudocódigo del cual basamos nuestro trabajo.

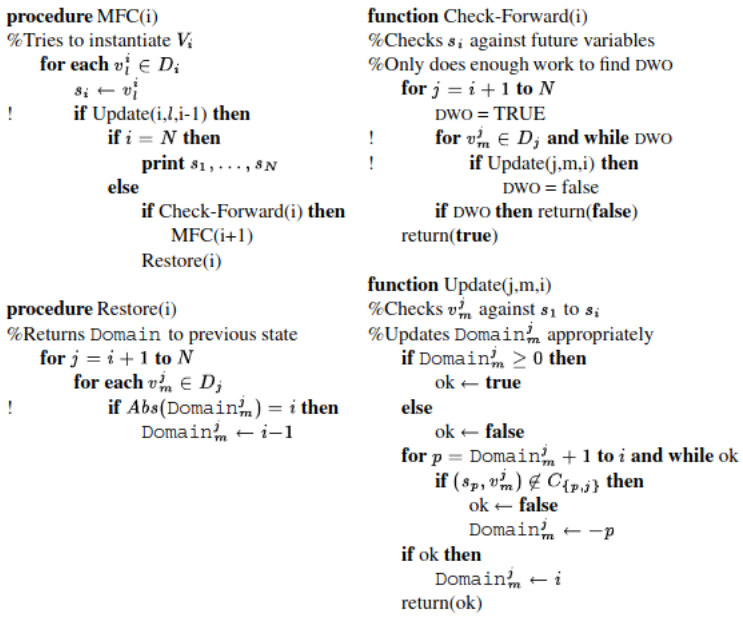

Figura 3.1: Pseudocódigo

A continuación, se presenta una explicación a cada una de las funciones implementadas en nuestro código propuesto (por temas de espacio no se pondrá todo el código aquí):

```
def extract_data(filepath):
```

Esta función permite extraer los datos de los archivos txt, ordenándolos con una variable asignada para cada dato.

```
def plot(time_data, profit_data):
```

Esta función permite graficar las ganancias en función del tiempo.

```
def minimal_forward_checking(i, selected_projects, remaining_budget, profits,
costs, tasks, covered_tasks, times, profit_changes, start_time)
```

Esta es una de las funciones más importantes, ya que tenemos el MFC aplicado. En primer lugar, se verifica que se cumpla el tiempo definido como 1800 s (30 minutos). Además, va guardando el máximo según las iteraciones que se van haciendo. Verifica cada variable de desición asociada al proyecto en sus dos casos 0 y 1, y revisa la factibilidad de realizarse o no.

```
def check_forward(i, selected_projects, costs, tasks, remaining_budget, covered_tasks):
```

Esta función, permite verificar las variables posteriores a una variable determinada por la función de MFC y revisa sus restricciones para su factibilidad.

```
def restore(i, selected_projects, remaining_budget, costs, tasks, covered_tasks):
```

Esta función permite borrar el dominio seleccionado que no sea factible o consistente.

```
def update(i, decision, costs, tasks, remaining_budget, covered_tasks):
```

Esta función permite actualizar el dominio siempre y cuando sea factible según las restricciones.

```
def main()
```

Finalmente, esta función es la principal, donde se ejecutan los 3 archivos y se muestra el valor mejor por cada uno. Cabe destacar que se inicia el MFC con el proyecto 0.

3.2. Análisis

Los resultados obtenidos por cada archivo son los siguientes:

- 1-2024:

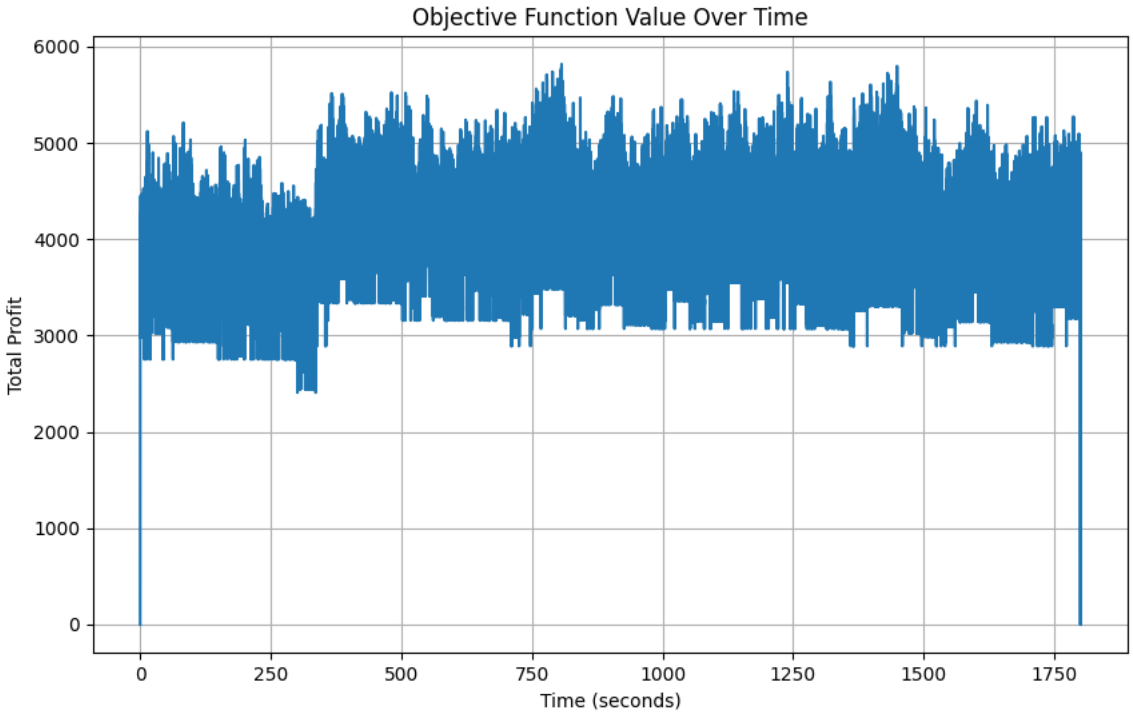

Figura 3.2: Función Objetivo (Ganancia) vs Tiempo 1

Max Ganancia: $5819
Mejor set de proyectos que maximiza la ganancia: [0, 1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 22, 28, 117, 140, 164, 173, 176, 179, 192] Se puede observar que el máximo valor alcanzado para esta ejecución se obtuvo casi a la mitad de la ejecución, en unos 4 minutos aproximadamente. Sin embargo, no se alcanzó el óptimo global, aunque con más capacidad de CPU y tiempo, probablemente si se hubiese alcanzado.

- 2-2024:

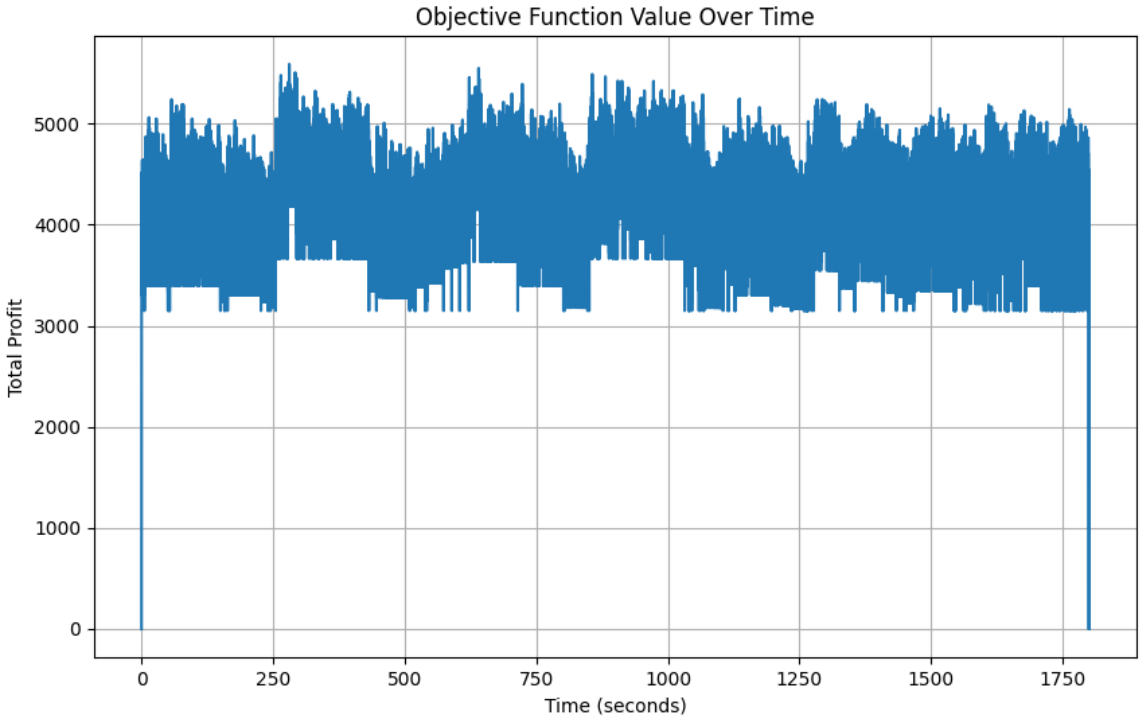

Figura 3.3: Función Objetivo (Ganancia) vs Tiempo 2

Max Ganancia: $5588
Mejor set de proyectos que maximiza la ganancia: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 24, 40, 75, 86, 222, 243]
Se puede observar que el máximo valor alcanzado para esta ejecución se obtuvo al principio la ejecución, en unos 5 minutos aproximadamente. Sin embargo, no se alcanzó el óptimo global, aunque con más capacidad de CPU y tiempo, probablemente si se hubiese alcanzado.

- 3-2024:

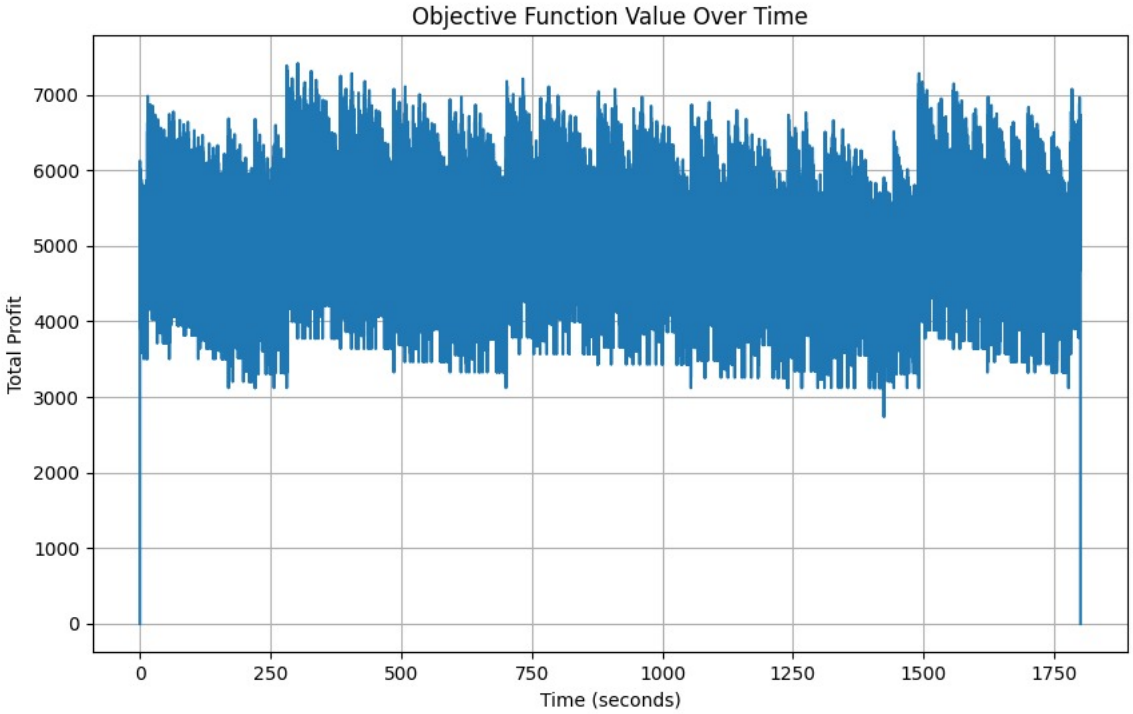

Figura 3.4: Función Objetivo (Ganancia) vs Tiempo 3

Max Ganancia: $7415
Mejor set de proyectos que maximiza la ganancia: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 14, 15, 20, 25, 35, 44, 50, 54, 72, 74, 78, 79, 86, 95, 97]
Se puede observar que el máximo valor alcanzado para esta ejecución se obtuvo al principio la ejecución, en unos X minutos aproximadamente. Sin embargo, no se alcanzó el óptimo global, aunque con más capacidad de CPU y tiempo, probablemente si se hubiese alcanzado.

Finalmente, se puede concluir, que el archivo 3 obtuvo mejor ganancia con sus proyectos realizados.

4. K-Means

4.1. Algoritmo

Para incorporar K-Means en nuestro código, primero importamos la librería 'KMeans' de sklearn. Luego, implementamos una función llamada 'k_means_cluster', que utiliza los datos de ganancias de proyectos y aplica el algoritmo K-Means para agrupar los proyectos en 4 grupos distintos.

En el proceso principal (‘main‘), utilizamos un bucle para realizar cálculos para los proyectos y seleccionar los clústeres que aún no han sido seleccionados de forma secuencial.

Posteriormente, dentro de cada clúster seleccionado, aplicamos el algoritmo MFC (Minimal Forward Checking) a los proyectos correspondientes. Todo esto se realiza mediante un bucle dentro del ‘main‘, que recorre cada clúster y aplica el MFC a los proyectos contenidos en él.

Este enfoque nos permite utilizar la clusterización para organizar los proyectos de manera más efectiva y luego aplicar el algoritmo MFC de forma secuencial en cada clúster, lo que potencialmente mejora la eficiencia y calidad de las soluciones encontradas.

4.2. Análisis

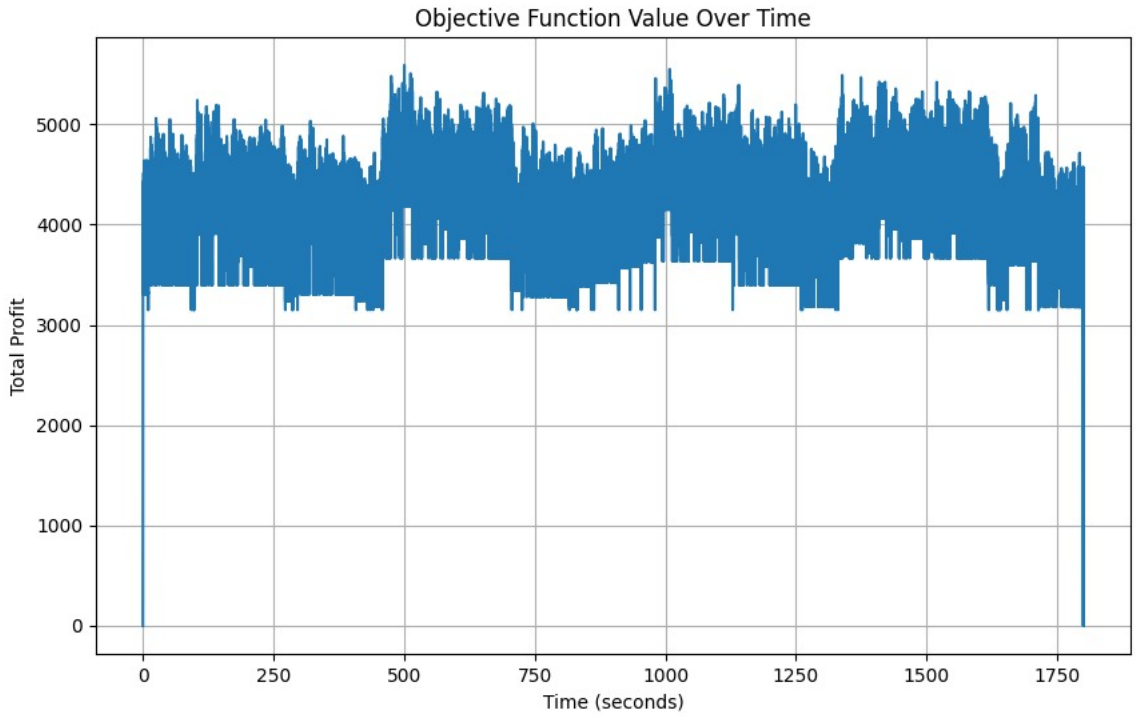

Figura 4.1: Función Objetivo (Ganancia) vs Tiempo K-Means

Max Ganancia: $5588
Mejor set de proyectos que maximiza la ganancia: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 24, 40, 75, 86, 222, 243]
Observando la gráfica, se puede notar cierto parecido a la gráfica obtenida en el mismo archivo sin K-Means. Sin embargo, en general obtiene resultados un poco más altos en el tiempo que sin este algoritmo. También cabe destacar que se encontró el mismo máximo, esto puede

deberse a que quizá si se hubiese ejecutado más tiempo, se hubiesen encontrado resultados distintos, pero durante el tiempo K-Means sí demostró mejores resultados.

5. Conclusión

En esta tarea, se investigó la aplicación del algoritmo Minimal Forward Checking (MFC) para maximizar las ganancias en un entorno empresarial. Se exploró la combinación de MFC con el algoritmo K-Means para mejorar la selección de proyectos.
Los datos obtenidos de los códigos revelaron que la aplicación conjunta de K-Means y el algoritmo MFC produjo resultados superiores en comparación con el uso exclusivo de MFC. Esta mejora podría atribuirse a una selección más efectiva de proyectos al agruparlos en clusters con características similares. Esto facilita una asignación más estratégica de recursos y una exploración más eficiente del espacio de soluciones.

6. Referencias

[1] Bacchus, F., & Grove, A. (1debe September). On the forward checking algorithm. In *International Conference on Principles and Practice of Constraint Programming* (pp. 292-309). Berlin, Heidelberg: Springer Berlin Heidelberg.

7. Anexos

1. Repositorio con el código