Inteligencia Artificial

Estado del Arte: **Weapon target assignment problem**

Pedro Donoso Aguilera

20 de julio de 2022

Evaluación

Resumen (5 %):	_____
Introducción (5 %):	_____
Definición del Problema (10 %):	_____
Estado del Arte (35 %):	_____
Modelo Matemático (20 %):	_____
Conclusiones (20 %):	_____
Bibliografía (5 %):	_____
Nota Final (100 %):	_____

Resumen

En este informe se expondrá un problema relacionado a la estrategia militar, más específicamente la asignación de armas a objetivos de guerra. Este problema conocido como *Weapon Target Assignament Problem* (WTA) tiene diversas formas de obtener una solución, una técnica incompleta bastante eficiente [2] es *Branch and Bound* utilizando relajación y también se han encontrado técnicas incompletas que encuentren óptimos locales que se trabajarán más adelante. Lo que se busca es que el lector comprenda la formulación de la problemática y la implementación de una de las técnicas incompletas utilizando heurísticas para mejorar la búsqueda del óptimo local con la finalidad de entender que no existe un solo método para solventar un problema, sino que es posible combinar o crear heurísticas para encontrar óptimos locales que nos entreguen soluciones de calidad.

1. Introducción

1.1. Propósito

Este informe fue desarrollado con el propósito de que lector conozca que existen diferentes forma de abordar un problema de optimización combinatoria, específicamente el problema de asignación de armas a objetivos de guerra, para que así pueda aplicar este conocimiento en problemas similares.

1.2. Descripción del problema

En este informe se dispondrá al lector un problema relacionado con las estrategias de guerra, específicamente el problema de Asignación de objetivos de armas, este problema conocido como *Weapon Target Assignment* (WTA) consiste en la asignación de armas amigas a los objetivos hostiles con el fin de proteger los activos amigos o destruir los objetivos hostiles, es considerado un problema NP-completo, por consiguiente tiene variaciones en su formulación y diversas formas de obtener una solución. Este documento busca expresarle al lector la esencia del problema para luego estudiar las diversas formas de encontrar soluciones incompletas, heurísticas a utilizar y compararlas, comprendiendo así que no sólo existe una forma para encontrar soluciones.

1.3. Estructura del documento

Primero se presentará la Definición del problema, donde se explicará el caso general del problema de una manera verbal para que sea entendido fácilmente y se tenga una idea general de este informe. En segundo lugar se tendrá el Estado del Arte, donde se mostrarán los estudios realizados con antelación, con la finalidad de proveer al lector información relevante que le sirva para ver las diferentes formas de abordar el problema de optimización combinatoria. En tercer lugar estarán los Modelos Matemáticos tanto para el caso general del WTA como para un caso específico de este, es necesario tener presenta las variables, restricciones y funciones objetivo para poder utilizar los algoritmos que se presentarán en el documento. Por último se tendrán las Conclusiones, que comparará los métodos planteados en el informe y se dará un pequeño resumen del más prometedor.

1.4. Motivación

La motivación para crear este informe es aprender como un problema puede ser resuelto de diferentes formas, con el fin de posteriormente intentar crear una solución al WTA recurriendo al conocimiento extraído de este informe, ya sea utilizando una de esas técnicas o combinándolas.

2. Definición del Problema

El problema nacido a mediados del siglo XX, introducido por la necesidad de ventaja militar en la época, llamado Problema de Asignación de Objetivos de Armas, más conocido como *Weapon Target Assignment Problem* (WTA) es una clase de problemas de optimización combinatoria presentes en los campos de optimización e investigación de operaciones. El problema anterior nace como una variación del problema de asignación, conocido como *Assignment Problem* (AP), problema fundamental de los denominados problemas de optimización combinatoria.

Es bastante común encontrar en la literatura sobre el WTA que se centra en la perspectiva defensiva [1] [4] [6], encontrar la asignación óptima de armas a las amenazas de una manera que minimice el daño esperado, pero es inevitable que algunos consideren la perspectiva ofensiva [3], encontrar una asignación óptima de un conjunto de armas de varios tipos a un conjunto de objetivos para maximizar el daño total esperado infligido al oponente. Para cuestiones de este acercamiento al problema consideraremos su perspectiva defensiva.

El problema básico consiste en considerar una cantidad de tipos de armas y objetivos, a cada objetivo se le asigna un valor destructivo, disponemos también de la probabilidad de destrucción de los objetivos para la asignación de armas y objetivos, por otra parte, restringiendo el espacio de búsqueda debemos considerar que la cantidad de armas disponibles y la cantidad mínima de armas requeridas en cada objetivo condicionan la cantidad de armas asignadas a cada objetivo, esto para asegurarnos de que el número total de armas utilizadas no exceda la cantidad de armas disponibles. También debemos tener en cuenta que cada objetivo deba tener al menos 1 arma asignada y que no podemos asignar más de 1 arma por objetivo.

Nuestro objetivo es minimizar el valor esperado de supervivencia, es decir, minimizar el valor de daño para cada objetivo teniendo en cuenta la probabilidad de que cierta arma no destruya cierto objetivo y su valor destructivo.

Antes de explicar las diferentes variaciones de este problema debemos definir lo que es un escenario o etapa, esto es una iteración donde se realiza una evaluación de los daños, y en base a esta evaluación se reasignan las armas disponibles a los objetivos supervivientes de forma iterativa.

En la literatura podemos encontrar diferentes variaciones de este problema, por ejemplo, la multi-objetivo, donde se puede asignar más de un arma a cada objetivo y no se requiere que todos los objetivos tengan armas asignadas. Además, tenemos las variaciones más comunes, el problema WTA estático y dinámico. En el caso estático, las armas se asignan a los objetivos una vez, por el contrario el caso dinámico implica agregar el tiempo como una dimensión adicional, los escenarios posteriores se verán relacionados directamente por el escenario anterior, así irá modificándose a medida que avanza el tiempo.

Para este trabajo se considerará la variación más simple, el problema WTA estático y no se utilizará ningún proceso de defensa iterativo, es decir, se pasará un escenario inicial y se trabajará siempre con ese mismo, no cambiará con el transcurso de la búsqueda.

3. Estado del Arte

En esta sección del trabajo no se discutirán los detalles de los algoritmos estudiados, porque no es parte esencial de este trabajo, solamente se dispondrá información recopilada sobre las diferentes técnicas utilizadas en la literatura para resolver de la forma más óptima este problema combinacional. Los algoritmos estudiados serán Tabu Search, Simulated Annealing, Genetic Algorithm y Variable Neighborhood Search.

En al figura 1 se muestra una comparación realizada en un estudio del problema de los algoritmos antes mencionados, dispuestos como TS, SA, GA y VNS, este gráfico nos muestra el tiempo en unidades de CPU de estos algoritmos a medida que aumentan las iteraciones o escenarios, con un máximo de 19 escenarios.

Se puede notar que los valores de tiempo de CPU se aproximan hasta el escenario 10, después, hay pendientes pronunciadas para GA, TS y especialmente para VNS. EL tiempo de solución de SA tiene una pendiente muy baja y se puede ver fácilmente que SA supera a estos algoritmos cuando la dimensión del problema es mayor.

4. Modelo Matemático

Para expresar matemáticamente esta variación estática del problema WTA, de ahora en adelante denotada como SWTA, extraída del paper guía llamado A Parallel Simulated Annealing Algorithm for Weapon-Target Assignment Problem debemos definir primero que $i = 1, \ldots, n$ corresponde a cada objetivo con n la cantidad total de objetivos y $j = 1, \ldots, m$ para cada objetivo con m la cantidad total de tipos de armas.

4.0.1. Parámetros

Definiremos las siguientes variables.

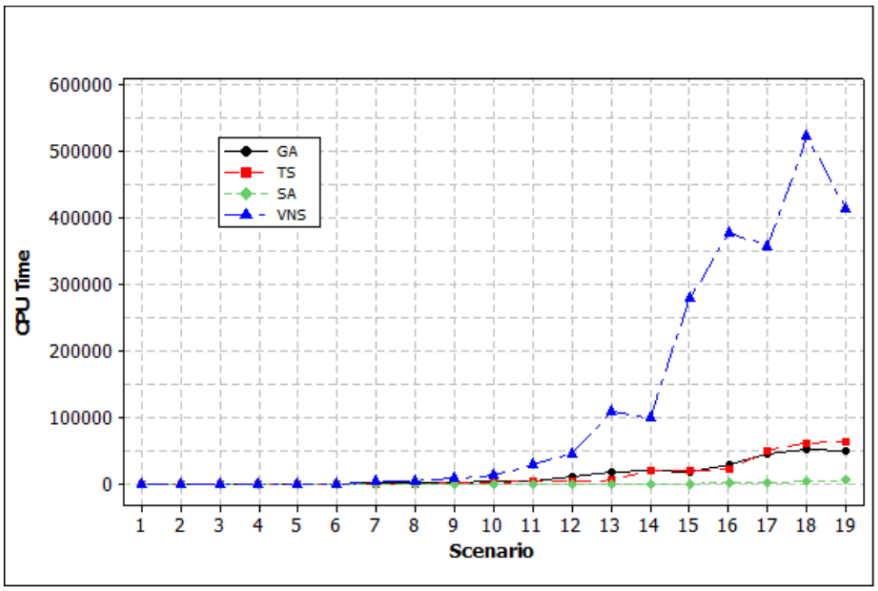

Figura 1: CPU Time Comparison of Algorithms [6, pág 47]

m : número de objetivos a destruir.
n : número de tipos de armas. Con un arma por cada tipo.
p_{ij} : la probabilidad de que el arma j destruya el objetivo i.
V_j : el valor de destructivo del objetivo j

4.0.2. Variable

La variable 1 definida nos ayudará a representar nuestro escenario, para cuestiones matemáticas se representará como una variable binaria para mostrarnos a qué objetivo se le asigna el arma i. Para cuestiones de representación en el algoritmo, será un arreglo que contenga el identificador del arma del tipo j asignada al objetivo i, y cada casilla corresponderá a un objetivo distinto, esto nos permitirá tratar con la restricción mediante la representación.

$$\mathcal{X}_{ij} : \begin{cases} \text{Si el arma } j \text{ es asignado a el objetivo } i \\ 0 \text{ otro caso} \end{cases} \tag{1}$$

4.0.3. Formulación SWTA

Un defensor tiene $w_i = 1, ..., m$ tipos de armas con las que defenderse contra $j = 1, ..., n$ objetivos. Cada arma del tipo i tiene una probabilidad p_{ij} de matar al objetivo j y cada objetivo j tienen un valor destructivo V_j. Con las variables de decisión $\mathcal{X}_{ij}$ indicando el número de armas de tipo i para asignar al objetivo j, se formula el $SWTA$ de la siguiente manera. [3, pág 2]

4.1. Función Objetivo

Los pesos definidos los interpretamos según el autor del paper antes mencionado, como el valor destructivo, entonces buscaremos minimizar la suma ponderada de los pesos $\mathcal{V}_\rangle$, es decir,

minimizar los daños que recibiría el recurso que deseamos proteger.

$$\min \sum_{j=1}^{n} \mathcal{V}_j \prod_{i=1}^{m} (1 - p_{ij})^{\mathcal{X}_{ij}}$$

4.2. Restricciones

En la ecuación 2 está representada la restricción relacionada con que cada objetivo debe tener asignada un arma y un arma debe estar relacionada con un objetivo.

$$\begin{aligned} \text{s.t.} \sum_{j=1}^{n} \mathcal{X}_{ij} = 1\,, \quad & \text{for } i = 1, \ldots, m \\ \mathcal{X}_{ij} \in Z_+\,, \quad & \text{for } i = 1, \ldots, m,\ j = 1, \ldots, n \end{aligned} \tag{2}$$

Más adelante nos daremos cuenta que esta restricción está tratada implícitamente por la representación.

5. Representación

Como entrada al algoritmo se tiene un archivo de texto que contiene en su primera linea la cantidad de objetivos y armas, que se le denominará dimensión de la instancia, para este caso de estudio será la misma cantidad de armas que de objetivos, en sus lineas siguientes especifica los valores de destrucción de cada objetivo y en las restantes, las probabilidades de que el arma de tipo i destruya al objetivo j.

Como estructura de datos para representar la solución, se dispondrá de un arreglo del mismo largo que la cantidad de armas y objetivos, en cada índice se representará los objetivos y los valores de cada casilla representará el arma que se le asignó a cada objetivo, la justificación de ello tiene relación con la sección anterior.

Como se mencionó en la sección anterior, la restricción 2 que acota la cantidad de armas asignadas a una por cada objetivo y viceversa, está tratada implícitamente en esta representación vectorial.

En el modelo matemático se representa X_{ij} como una matriz bidimensional que indica la cantidad de cada arma relacionada con cada objetivo.

		Armas		
		0	**1**	**2**
Objetivos	**0**	2	0	1
	1	1	3	0
	2	8	3	1

Cuadro 1: Representación matricial de la solución.

Como se puede ver en la representación anterior 1, los índices de las columnas, marcadas en negrita, representa cada tipo de arma con índices 0,1 y 2, y en las filas se representan los objetivos con índices 0,1 y 2, entonces podemos decir que al objetivo 0, se le asignan 2 armas del tipo 0, 0 armas del tipo 1 y 1 arma del tipo 2.

Esto ayuda a estandarizar una representación para los problemas del tipo WTA que permite tener más de un arma asignada a un objetivo, pero esto solo se representa así en este documento para la comprensión del lector, cuando se acota el dominio de los valores que se le asignan en cada casilla de la matriz X_{ij} al dominio de este caso particular de problema SWTA con un arma

asignada para cada objetivo 2, entonces se puede generar una matriz diagonal que se aproxime más a lo que se necesita.

		Armas		
		0	**1**	**2**
Objetivos	**0**	1	0	0
	1	0	1	0
	2	0	0	1

Cuadro 2: Representación matricial de la solución.

Con esto se puede evidenciar que no es necesario tener una matriz que representa una posible solución porque las casillas que representan los ceros están demás, consumen memoria y puede llegar a ser difícil de manejar, es por lo anterior que se modificó la representación de la posible solución a una representación vectorial.

0	**1**	**2**
2	0	1

Cuadro 3: Representación vectorial de la solución.

Entonces la nueva representación consiste en mostrar en cada casilla el tipo de arma que se le asigna a cada objetivo y los índices de dicho vector representan a los objetivos, de esta manera se utiliza mucho menos espacio en memoria y se puede manejar implícitamente la restricción 2, haciendo un movimiento de intercambio de posiciones (switch) por ejemplo.

6. Descripción del algoritmo

El algoritmo que se analizará es una implementación del algoritmo *Simulated Annealing* en combinación con la heurística *Best Improvement* o *Mejor mejora*, esto se implementará como una especie de *Hill Climbing*, dado que se generará un vecindario y de dicho vecindario se escogerá el mejor.

Luego de que se escoja el mejor se aplicará la lógica de *SA*, entonces allí se decidirá con la temperatura y probabilidades si se acepta o no la solución.

Se seguirá de esta forma hasta que se logre la cantidad máxima de iteraciones o hasta que se llegue a una temperatura mínima.

Se debe tener en cuenta que *delta* es el delta de la mejor solución escogida, *new_solution* es la solución a la que corresponder ese mejor *delta* escogido y k es coeficiente de decaimiento de la temperatura.

Algorithm 1 *Simulated Annealing* con *Best Improvement*

Input: WTA5.txt
Output: Mejor solución

Definir variables extraídas de parsear el archivo de texto, dimensión, probabilidades de destrucción y pesos.
Definir temperatura inicial, temperatura mínima, su decaimiento y el máximo de iteraciones.
Generar solución inicial
Instanciar mejor solución y solución actual.
while ($numero\_iteraciones < maximo\_iteraciones$) and ($T\_current > T\_min$) **do**
    Generar movimientos de los índices de los vecinos
    Calcular los deltas de la solución actual y las soluciones con el movimiento.
    Escoger la solución que tenga mejor delta.
    if acceptance_function($delta$, $T\_current$) **then**
        accept_solution($problem$, $new\_solution$)
    else
        reject_solution()
    change_temperature($T\_current$, k)
    $numero\_iteraciones + 1$
Imprimir mejor solución y su calidad

Comenzamos definiendo nuestros parámetros iniciales y constantes a utilizar en el algoritmo, el vector de probabilidades de destrucción y el vector que representa los valores destructivos delos objetivos extraídos de la instancia definida al ejecutar el algoritmo, en esta primera parte también definimos las temperaturas inicial y final, el coeficiente de decaimiento de la temperatura y el máximo de iteraciones.

Iniciando con el algoritmo, generamos una solución inicial definida como la correspondencia secuencial de cada arma a su respectivo objetivo, es decir, el arma 2 asignada al objetivo 2, y así sucesivamente, en un vector en dos dimensiones correspondería a la matriz diagonal. Luego le sigue el bucle que ayudará a realizar la búsqueda, con dos sentencias de control de flujo, una que controla la cantidad de iteraciones y otra que controla que se esté trabajando dentro de los rangos acordados de temperatura.

En la búsqueda se genera el vecindario aplicándole un movimiento diferente a la solución actual, se calculan los deltas de cada vecino, se escoge la solución con el mejor delta y con este delta se realiza el proceso de aceptación de la solución escogida. Si se acepta la solución, es guardada como la mejor hasta el momento y si se rechaza, se sigue con la misma mejor solución con la que se inició la iteración de búsqueda, para finalizar la iteración se cambia la temperatura multiplicando la temperatura actual por el coeficiente de decaimiento.

Con respecto a la búsqueda el movimiento que se le realiza a cada vecino se realiza en base a un objetivo pivot escogido aleatoriamente, esto ayuda a realizar los movimientos swap a todos los vecinos, por ejemplo, en una instancia de 5 dimensiones, se escoge un pivot 2, y se cambia cada objetivo por el del pivot, generando 4 vecinos 4.

Solución Inicial	0	1	2	3	4

Vecino 0	2	1	0	3	4
Vecino 1	0	2	1	3	4
Vecino 2	0	1	3	2	4
Vecino 3	0	1	4	3	2

Cuadro 4: Vecindario generado en base a la solución inicial de ejemplo

7. Experimentos

Antes de comenzar cualquier proceso de experimentación se deben explicitar los óptimos locales propuestos por la literatura de referencia [5].

Instancia	**Mejor SA**
5	48.3640
10	96.3123
20	142.1070
30	248.0285
40	305.5016
50	353.0767
60	415.0528
70	498.1049
80	534.4408
90	594.0639
100	699.8357
200	1306.9126

Cuadro 5: Mejores óptimos locales de literatura de referencia.

Durante todos los experimentos realizados se utiliza esta información para comparar implementaciones de algoritmos.

7.1. Metodología

Para poner a prueba este algoritmo y encontrar el mejor óptimo local se realizaron diferentes experimentos de sintonización de parámetros, acordando al menos 5 iteraciones de ejecución del algoritmo como una cantidad razonable para evidenciar el alcance de las pruebas con temperatura y al menos 10 iteraciones para las pruebas con movimientos, lo anterior para lograr realizar los experimentos en menos de una hora, tiempos totales que se documentaron en los archivos excel adjuntos en la carpeta *stadistics* los parámetros a sintonizar en estos experimentos son los siguientes:

- Movimiento.
- Temperatura y coeficiente de decaimiento.

Cabe destacar que para realizar de forma más óptima este proceso de sintonización se modificó el algoritmo para que se pasasen los parámetros por argumentos. Además para realizar el cambio de parámetros se compiló el código y se llamó a la función *run* del Makefile que nos permite entregarle los parámetros con nombre al algoritmo, esto simplemente se hace con fines demostrativos, simplemente se podría pasar los parámetros directamente al ejecutable resultante de la compilación.

7.2. Entorno de experimentación

- Para realizar las pruebas se utilizó un Windows Subsystem for Linux (WSL) con un sistema operativo virtual Ubuntu 20.04.3 LTS de 9 GB de Ram.
- El WSL se instaló en un sistema operativo Windows 10 Pro 64-bits.
- Se utilizó un procesador de 4 núcleos 2.5 GHz y 12 GB de Ram.
- Para ejecutar el programa se utilizó el compilador gcc version 9.4.0.

7.3. Parámetros del algoritmo

Primero explicitaremos los parámetros del algoritmo, estos mismos parámetros serán expuestos en la salida del programa y guardados en un archivo de texto para tener un historial de las ejecuciones.

7.3.1. Descripción de los parámetros

- **INS**, dimensión de la instancia, cantidad de objetivos y armas de la instancia.
- **T_INIT**, temperatura inicial, el valor predeterminado es 1000.
- **T_FINISH**, temperatura final, el valor predeterminado es 1e-50
- **K**, coeficiente de decrecimiento de la temperatura.
- **MAX_ITER**, cantidad máxima de iteraciones de la búsqueda.
- **F**, valor de la función objetivo del óptimo local.
- **MOVE**, número identificador del movimiento.
- **TIME**, tiempo en segundos que dura la ejecución del algoritmo.

7.4. Experimento 1: Temperatura y coeficiente de decaimiento

El objetivo de este experimento es determinar la injerencia de la temperatura y su coeficiente de decaimiento en la cantidad de iteraciones en la búsqueda del algoritmo, se utilizó la cantidad de 5 intentos para cada prueba para lograr un tiempo de ejecución de 1.6 Horas.

7.4.1. Procedimiento

Se realizará una prueba donde se iterará el coeficiente k entre 0,90 y 0,99 a intervalos de centésimas por cada instancia para determinar si la precisión en el k es determinante en la búsqueda más óptima. En esta prueba se utilizó el movimiento 1 que por el experimento siguiente se sabe tiene mejores resultados a la hora de converger en el óptimo local entregado por la literatura [5]. Además se fijó un máximo de iteraciones como 100000 para evitar la parada por cantidad de iteraciones y así permitir que el algoritmo se detenga por la condición de temperatura.

Para lo anterior se utilizó un script 1 en bash que itera entre los coeficientes, 0,90 y 0,99 a intervalos de centésimas por cada instancia y las repite 5 veces.

```
#!/bin/bash

instances=( 5 10 20 30 40 50 60 70 80 90 100 200 )

for i in $(seq 0.90 0.01 0.99) ; do # coeficientes de decrecimiento
    for n in ${instances[@]} ; do
        for j in {1..5}; do
            make run INS=$n K=$i T_INIT=1000 T_FINISH=0.1 MAX_ITER=100000
                MOVE=1
        done
    done
done
```

Listing 1: Script para realizar sintonización de parámetro coeficiente de decrecimiento.

7.4.2. Comparación estado del arte

Se escogió el intervalo a iterar en el parámetro de coeficiente de decrecimiento tomando en consideración la literatura de referencia [5] que utiliza un k=0,9999, como este número es muy cercano a 1 no nos permite converger a la temperatura predeterminada 0.1 en pocas iteraciones para luego realizar otros tipos de pruebas, es más, para la instancia de dimensión 200 el algoritmo converge a la temperatura mínima en 200000, algo que podría dificultar las pruebas con instancias de dimensiones altas.

7.4.3. Criterios de término

El criterio de término es simplemente llegar a la temperatura mínima 0,1, por lo mismo se fija un máximo de iteraciones de 100000, un número inalcanzable para las instancias y los coeficientes de decrecimiento presentados.

7.5. Experimento 2: Movimiento

En esta segunda prueba se busca encontrar el mejor movimiento que nos lleve a un óptimo local cercano al entregado por la literatura de referencia [5] independiente del número de iteraciones o el tiempo. Para ello se buscaron 3 movimientos razonables para comparar. Se utilizó la cantidad de 10 intentos para cada prueba para lograr un tiempo de ejecución de 1.7 Horas.

7.6. Movimientos

Los tres movimientos swap distintos para generar el vecindario son los siguientes.

- Movimiento 0: Movimiento swap completamente aleatorio entre dos objetivos q y r, este movimiento es el mismo propuesto en la literatura de referencia [5], como en esta implementación del algoritmo SA se genera un vecindario, probablemente este movimiento no es el más óptimo debido a que su aleatoriedad permite repetir movimientos en el vecindario.
- Movimiento 1: Movimiento swap semi-aleatorio entre un objetivo aleatorio pivot o fijo y otro iterativo j, siendo j el objetivo que itera todo el escenario, este movimiento se creó pensando en la mantención de parte de la solución principal desde la que se genera el vecindario, así se fija un objetivo-arma para cada vecindario generado y en cada movimiento realizado en el vecindario, se intercambia por cada uno de los objetivos-arma del escenario.
- Movimiento 2: Movimiento swap completamente iterativo entre un objetivo j y otro $j+1$, este movimiento simplemente cambia los objetivos-armas con su objetivo-arma vecino que no tiene mucha relevancia en este problema por su independencia respecto a su posición.

7.6.1. Procedimiento

En esta prueba se iteró el movimiento con cada instancia al menos 10 veces, es decir, primero se fijó el movimiento cero que se ejecutó 10 veces para cada instancia, así mismo para el movimiento uno y finalmente el movimiento dos.

Para ello se creó un pequeño script en bash que realiza esta tarea automáticamente.

Los parámetros que no se iteraron como el número de instancias y el movimiento, se dejaron predeterminados como muestra el script 2, la temperatura inicial en 1000, la temperatura final en 0,1, el coeficiente de decaimiento en 0,98 y el máximo de iteraciones en 100000.

```
#!/bin/bash

instances=( 5 10 20 30 40 50 60 70 80 90 100 200 )
movements=( 0 1 2 )
```

```
for i in ${movements[@]} ; do
    for n in ${instances[@]} ; do
        for j in {1..10}; do
            make run INS=$n K=0.98 T_INIT=1000 T_FINISH=0.1 MAX_ITER
                =100000 MOVE=$i
        done;
    done
done
```

Listing 2: Script para realizar sintonización de parámetro movimiento.

7.6.2. Comparación estado del arte

En la literatura de referencia [5] se explicitan los valores de los parámetros utilizados para realizar las pruebas, k = 0,9999, T_INIT=1000, T_FINISH=0,1, durante las diferentes pruebas, se notó que utilizar un coeficiente de decrecimiento tan cercano a 1 no afectaba mayormente en la disminución de la temperatura y esto podía llevar al algoritmo a ejecutarse en una cantidad de iteraciones en el orden de los 200,000, entonces se acordó utilizar un coeficiente $k = 0{,}98$ que nos permite llegar a la temperatura final en 179 iteraciones 3. Todo lo anterior se puede evidenciar en las pruebas realizadas para sintonizar los parámetros de temperatura y coeficiente de decrecimiento. Si bien esto se pudo evitar definiendo una cantidad lo suficientemente baja en el parámetro MAX_ITER, en este experimento se pretende pasar por todo el intervalo de la temperatura definida para así aprovechar al máximo el algoritmo de *Simulated Annealing*.

7.6.3. Criterios de término

Como se definió anteriormente, el criterio de término utilizado fué solamente alcanzar la temperatura final para aprovechar al cien por ciento el algoritmo SA.

8. Resultados

8.1. Temperatura y coeficiente de decaimiento

8.1.1. Logro de la experimentación

Con este experimento se pudo determinar que la precisión del k es proporcional a la cantidad de iteraciones, y que a medida que la instancia aumenta en su dimensión, las cantidades de iteraciones son cada vez más relevantes a la hora de encontrar un mejor óptimo.

Luego de la experimentación se realizó un gráfico resumen 2 que muestra el promedio de los óptimos locales obtenidos para cada instancia con cada coeficiente y para la última instancia de dimensión 200 se puede extraer, como se puede ver en la figura 4, que no existe mayor diferencia entre el coeficiente 0,98 y 0,99, con el coeficiente 0,98 se logra una diferencia entre el óptimo local encontrado por la prueba y el óptimo de referencia de 48,2614 puntos, con cerca de 455 iteraciones a diferencia de lo que sucede con el coeficiente 0,99 que se logra una diferencia de 26,4274 con casi el doble de iteraciones, siendo casi despreciable en comparación con la diferencia del orden de los 492,4654 que alcanza el coeficiente 0,90.

En cambio para una instancia pequeña como la de dimensión 5, ver figura 3, un coeficiente 0,90 o 0,99 no afecta en la búsqueda, de hecho siempre se obtiene el óptimo local de referencia, la única gran diferencia es que para un coeficiente de 0,90, se ejecutan 87 iteraciones y para un coeficiente de 0,99, se realizan 916 iteraciones.

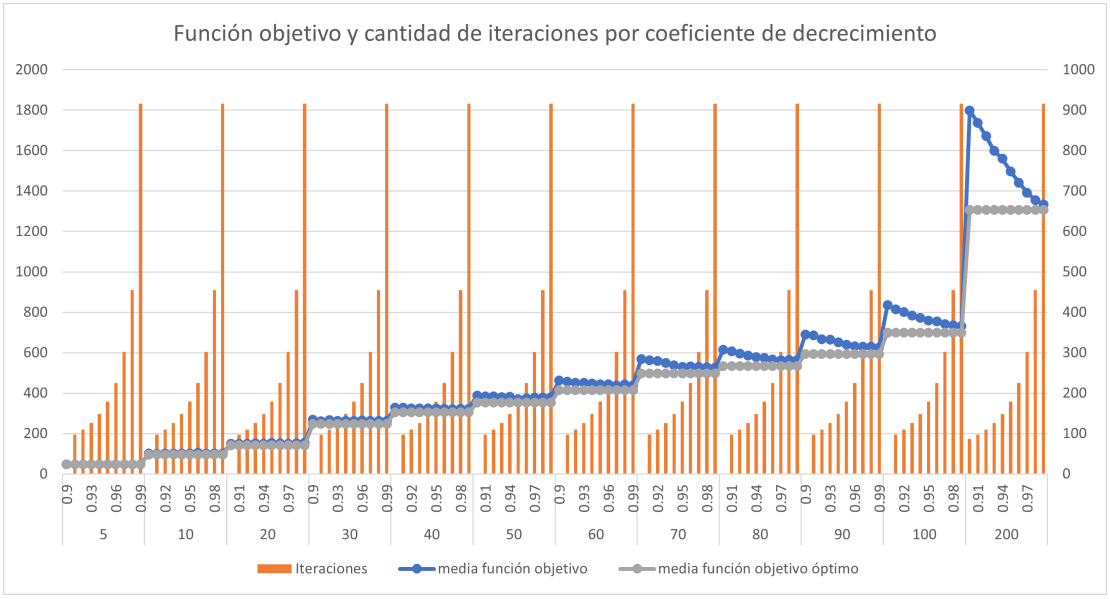

Figura 2: Función objetivo y cantidad de iteraciones por coeficiente de decrecimiento para todas las instancias.

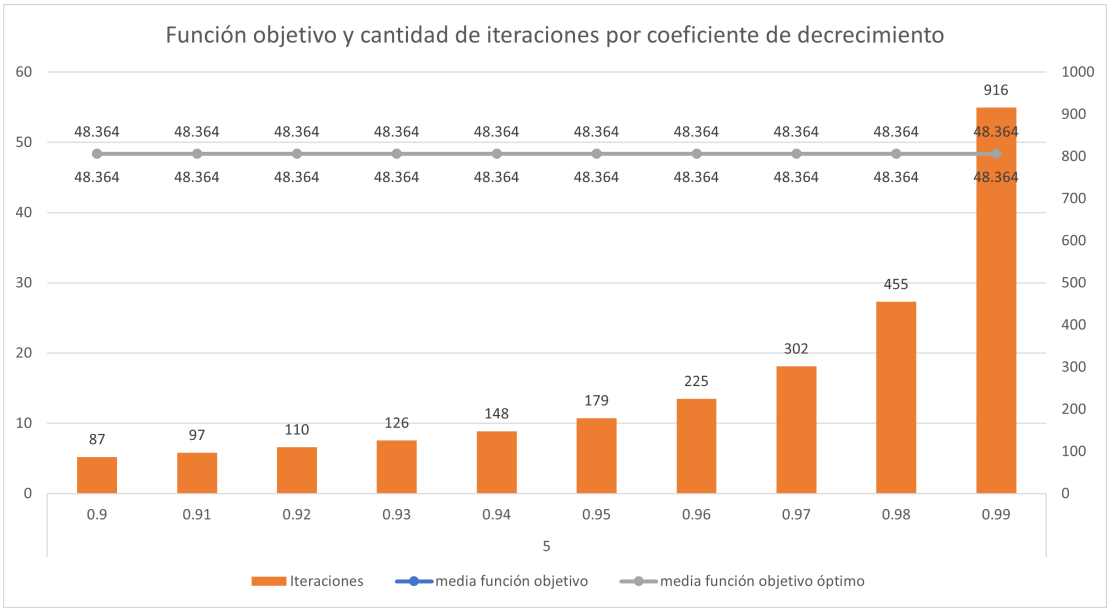

Figura 3: Función objetivo y cantidad de iteraciones por coeficiente de decrecimiento para la instancia de dimensión 5.

8.1.2. Análisis

Se puede entonces ahora determinar que el coeficiente 0,98 es un de los mejores coeficientes para encontrar un óptimo aceptable para realizar pruebas, debido a la insignificante diferencia

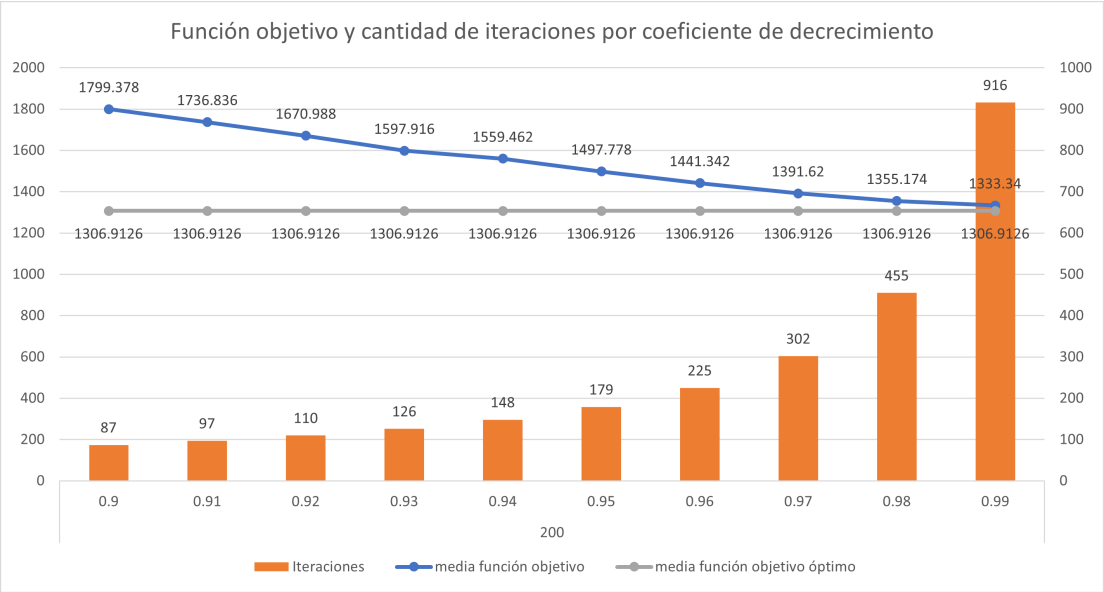

Figura 4: Función objetivo y cantidad de iteraciones por coeficiente de decrecimiento para la instancia de dimensión 200.

entre los óptimos locales que generan los coeficientes 0,99 y 0,98, y la inmensa diferencia entre su cantidad de iteraciones 916 y 455, respectivamente

8.2. Movimiento

8.2.1. Logro de la experimentación

Con la experimentación se logró identificar que el movimiento 2 que corresponde al movimiento completamente iterativo, como se dijo desde un principio no es óptimo para este problema.

Además se identificó una pequeña diferencia entre el movimiento 0 y el 1, como muestran los gráficos 5 y 6, que representan la media de la función objetivo por movimiento para las instancias 5 y 200. Si bien el *movimiento 0* se comporta de mejor manera que el *movimiento 1* en la instancia 200, no es mucho mejor, solamente una diferencia del orden de los 20 puntos destructivos. Y se destaca la dificultad que tiene el *movimiento 0* para lograr alcanzar el óptimo de referencia en la instancia de dimensión 5.

8.2.2. Comparación configuraciones

Para comprar se realizó un gráfico , ver figura 7, que muestra los 3 movimientos, de este gráfico se realizó uno similar que muestra solamente los resultados para las instancias 5 y 200.

8.2.3. Análisis

Para esta experimentación resulta difícil mencionar un movimiento que funcione de la forma más óptima posible para todas las instancias, pero se puede concluir que el movimiento 0 funciona de buena forma para instancias grandes como la 200 pero para instancias pequeñas se aleja un poco del óptimo referencial, el *movimiento 1* funciona de buena manera para las instancias pequeñas y para las más grandes se acerca bastante al óptimo referencial, y el *movimiento 2* no se acerca en absoluto de forma óptima al valor referencial del valor de destrucción.

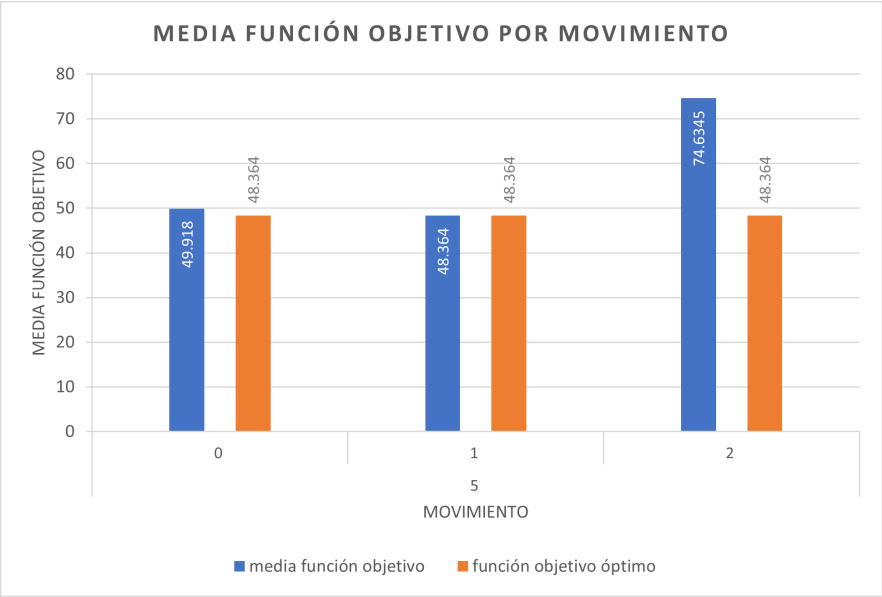

Figura 5: Media de función objetivo por movimiento para la instancia de dimensión 5.

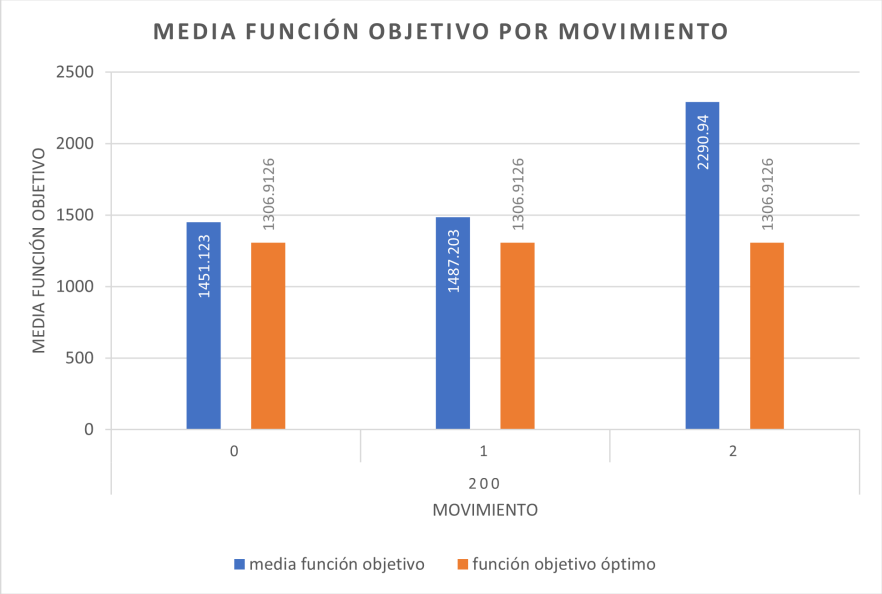

Figura 6: Media de función objetivo por movimiento para la instancia de dimensión 200.

Un punto destacable es el funcionamiento óptimo para una instancia específica del *movimiento 0*, este movimiento solamente funciona mejor que el *movimiento 1* para la instancia 200, esto puede deberse a algún error en la experimentación o alguna injerencia en la dimensión de la instancia.

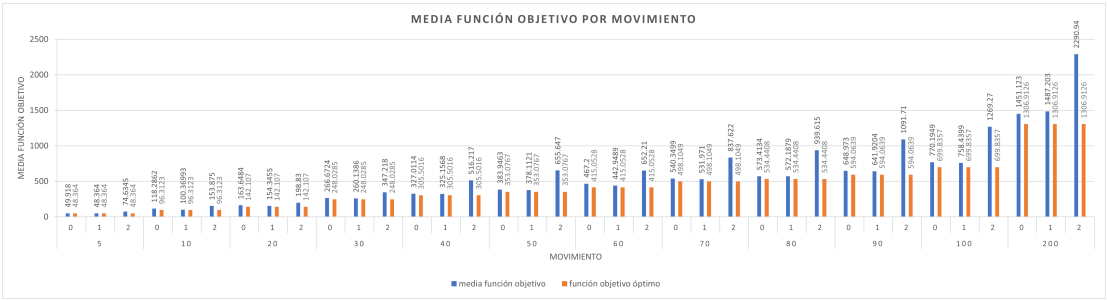

Figura 7: Media de función objetivo por movimiento.

9. Conclusiones

Luego de realizar las experimentaciones expuestas se logró determinar que para este algoritmo de búsqueda local que combina la técnica *Simulated Annealing* con *Mejor mejora*, existe dependencia entre el coeficiente de decrecimiento y la cantidad de iteraciones de la búsqueda, además se logró determinar la injerencia de la precisión del coeficiente de decrecimiento, habiendo acordado el coeficiente 0,98 como el más óptimo para realizar pruebas o ejecuciones de escenarios de dimensión alta para el entorno de experimentación presentado en el informe.

Además se logró determinar que los movimiento 0 y 1, que presentan cierto grado de aleatoriedad, funcionan de forma óptima para las instancias propuestas por la literatura de referencia [5]. Por último se destaca la imposibilidad de lograr llegar al óptimo local de referencia, no se pudo determinar si este problema es por la modificación que se hizo al algoritmo, agregándole la heurística *Mejor Mejora* o algún error en la implementación tanto del algoritmo como la calibración de los parámetros.

Queda propuesto experimentar con instancias de dimensión mayor a 200 para lograr determinar si el *movimiento 0*, totalmente aleatorio, es realmente un buen movimiento para instancias de dimensiones mayores a las expuestas.

Referencias

[1] Ravindra K. Ahuja, Arvind Kumar, Krishna C. Jha, and James B. Orlin. Exact and Heuristic Algorithms for the Weapon-Target Assignment Problem. *Operations Research*, 55(6):1136–1146, 2007.

[2] Alexandre Colaers Andersen, Konstantin Pavlikov, and Túlio A. M. Toffolo. Weapon-target assignment problem: exact and approximate solution algorithms. *Annals of Operations Research*, 312(2):581–606, 2022.

[3] Mohammad Babul Hasan and Yaindrila Barua. Weapon target assignment. In Gary Moynihan, editor, *Concepts, Applications and Emerging Opportunities in Industrial Engineering*, chapter 12. IntechOpen, Rijeka, 2020.

[4] Alexander Kline, Darryl Ahner, and Raymond Hill. The weapon-target assignment problem. *Computers Operations Research*, 105:226–236, 2019.

[5] Emrullah SONUC, Baha SEN, and Safak BAYIR. A Parallel Simulated Annealing Algorithm for Weapon-Target Assignment Problem. *International Journal of Advanced Computer Science and Applications*, 8(4), 2017.

[6] Asim Tokgöz and Serol Bulkan. Weapon target assignment with combinatorial optimization techniques. *International Journal of Advanced Research in Artificial Intelligence*, 2(7), 2013.