INSTITUTO TECNOLOGICO DE CIUDAD MADERO

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

TESIS

ARQUITECTURA DE INTERACCIÓN ENTRE DECISORES Y FRAMEWORK DE OPTIMIZACIÓN

Que para obtener el Grado de
Maestro en Ciencias de la Computación

Presenta
Ing. Pablo Eduardo Hernández Vicente
G15071361
1107323

Director de Tesis
Dra. Claudia Guadalupe Gómez Santillán
234490

Co-director de Tesis
Dr. Nelson Rangel Valdez

Cd. Madero, Tamaulipas enero 2023

Instituto Tecnológico de Ciudad Madero
Subdirección Académica
División de Estudios de Posgrado e Investigación

Ciudad Madero, Tamaulipas, 07/diciembre/2022

OFICIO No.: U.166/22
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN DE TESIS

C. PABLO EDUARDO HERNÁNDEZ VICENTE
No. DE CONTROL G15071361
P R E S E N T E

Me es grato comunicarle que después de la revisión realizada por el Jurado designado para su Examen de Grado de Maestría en Ciencias de la Computación, se acordó autorizar la impresión de su tesis titulada:

"ARQUITECTURA DE INTERACCIÓN ENTRE DECISORES Y FRAMEWORK DE OPTIMIZACIÓN"

El Jurado está integrado por los siguientes catedráticos:

PRESIDENTE:	DR.	HÉCTOR JOAQUÍN FRAIRE HUACUJA
SECRETARIA:	DRA.	LAURA CRUZ REYES
VOCAL:	DRA.	CLAUDIA GUADALUPE GÓMEZ SANTILLÁN
SUPLENTE:	DR.	NELSON RANGEL VALDEZ
DIRECTORA DE TESIS:	DRA.	CLAUDIA GUADALUPE GÓMEZ SANTILLÁN
CO-DIRECTOR:	DR.	NELSON RANGEL VALDEZ

Es muy satisfactorio para la División de Estudios de Posgrado e Investigación compartir con usted el logro de esta meta. Espero que continúe con éxito su desarrollo profesional y dedique su experiencia e inteligencia en beneficio de México.

A T E N T A M E N T E
Excelencia en Educación Tecnológica.
"Por mi patria y por mi bien".

INSTITUTO TECNOLÓGICO DE CIUDAD MADERO
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

MARCO ANTONIO CORONEL GARCÍA
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

ccp. Archivo
MACG/MLMR/aecr

Agradecimientos

En general, quiero agradecer a las personas que estuvieron involucradas en la realización de la maestría en ciencias de la computación en la división de estudios de posgrado e investigación del Instituto Tecnológico de Ciudad Madero, principalmente a la Dra. Claudia Gómez y al Dr. Nelson Rangel por su tiempo y atención.

Por su puesto, quiero agradecer a mi familia, a mi padre Fabian Hernández, a mi madre Verónica Vicente y a mi hermana Abigail Hernández por siempre estar ahí conmigo en cada nueva etapa de mi vida, por darme todo su apoyo y los mejores consejos y valores en los momentos más oportunos.

También quiero agradecer a todos los docentes que estuvieron involucrados en el desarrollo de mi perfil profesional, sin duda los conocimientos que me han transmitido son un pilar fundamental en mi vida.

Agradezco a mis amigos Enoc, Alejandro, Marysol, Noe, Fermín, Emmnuel, Gabriel, San Martin y Julio por siempre estar conmigo en los momentos difíciles y por motivarme a seguir adelante y a ser mejor persona, todos ellos a lo largo de los años se han convertido también en mi familia

Resumen

Muchas dificultades del mundo real requieren la resolución de problemas de optimización multiobjetivo. Para resolver estos problemas se necesita identificar un conjunto de soluciones que se aproxime al frente de Pareto, satisfaciendo la condición compromiso de no mejorar un objetivo si se empeora otro. Esta aproximación al frente de Pareto es necesaria para que el tomador de decisiones elija la solución final.

Para resolver problemas multiobjetivo se necesita un algoritmo evolutivo multiobjetivo, el cual emplea técnicas para generar soluciones aleatorias, utilizan procesos de selección para definir las soluciones que serán cruzadas, alteran las nuevas soluciones resultantes y finalmente utilizan un proceso para elegir las soluciones que pasaran a la siguiente generación.

La última población generada por estos algoritmos contiene a las mejores soluciones, sin embargo, esta población puede ser demasiado grande, complicando así el proceso de selección de la solución final que el tomador de decisiones debe realizar. Por lo tanto, se debe integrar una estrategia de incorporación de preferencias que aproxime los intereses del tomador de decisiones para así facilitarle la elección final de la solución.

Para hacer uso las estrategias de incorporación de preferencias anteriormente mencionadas se requieren diferentes parámetros que permitan modelar los intereses del decisor, como, por ejemplo, los pesos de los objetivos. Sin embargo, estos valores generalmente no pueden ser definidos con exactitud por el decisor, por lo que se pueden utilizar rangos o intervalos para cubrir la incertidumbre de estos valores.

Las preferencias del tomador de decisiones se pueden considerar antes de la ejecución del algoritmo evolutivo, al final de la ejecución o de manera interactiva durante la iteración del algoritmo. Este último método no ha sido muy estudiado, debido a que el proceso es más lento en comparación a la incorporación a priori y posteriori a causa de la intervención por parte del tomador de decisiones.

Por lo anterior, se ha propuesto un framework evolutivo interactivo que hace uso del análisis de desagregación de preferencias y una interfaz tipo chat, con el que se permita la incorporación de preferencias del tomador de decisiones de manera eficiente, con el fin de aumentar la cantidad de herramientas que integren este tipo de incorporación de preferencias y demostrar que las soluciones convergen más pronto que otros tipos de articulación de preferencias.

Con esta propuesta, el tomador de decisiones puede observar cómo la búsqueda se va moviendo en el espacio de soluciones gracias a la incorporación de sus preferencias, facilitando así la elección final de la solución.

Summary

Many real-world difficulties require the resolution of multi-objective optimization problems. To solve these problems, it is necessary to identify a set of solutions that approximate the Pareto front, satisfying the compromise condition of not improving one objective if another is worsened. This approximation to the Pareto front is necessary for the decision maker to choose the final solution.

To solve multi-objective problems, a multi-objective evolutionary algorithm is needed, which employs techniques to generate random solutions, uses selection processes to define the solutions that will be crossed, alters the new resulting solutions, and finally uses a process to choose the solutions that will pass to the next generation.

The last population generated by these algorithms contains the best solutions, however, this population may be too large, thus complicating the process of selecting the final solution that the decision maker must perform. Therefore, a preference incorporation strategy must be integrated that approximates the interests of the decision maker to facilitate the final choice of the solution.

To use the previously mentioned preference incorporation strategies, different parameters are required to model the interests of the decision maker, such as, for example, the weights of the objectives. However, these values generally cannot be defined exactly by the decision maker, so ranges or intervals can be used to cover the uncertainty of these values.

The preferences of the decision maker can be considered before the execution of the evolutionary algorithm, at the end of the execution or interactively during the execution of the algorithm. This last method is the least studied because the process is more complex and slower than the priori and posteriori incorporation due to the intervention of the decision maker.

Therefore, an interactive evolutionary framework has been proposed that makes use of the disaggregation analysis of preference s and a chat-type interface, which allows the incorporation of preferences of the decision maker in an efficient manner, to increase the number of tools that integrate this type of preference incorporation and demonstrate that the solutions converge sooner than other types of preference articulation.

With this proposal, the decision maker can see how the search moves in the solution space thanks to the incorporation of their preferences, thus facilitating the final choice of the solution.

Contenido

Capítulo 1 Introducción

En la vida diaria, las personas se enfrentan a diversos problemas, los cuales a su vez tienen diferentes alternativas de solución. Se obtendrán mejores o peores resultados dependiendo de las decisiones que tome el individuo.

Con el fin de obtener mejores resultados durante la resolución de problemas, se han creado diversas estrategias para entenderlos, por ejemplo, Ackoff, en su libro El arte de resolver problemas de 1978 (Ackoff, 1978), menciona que un problema se compone de cinco elementos: los tomadores de decisiones o los que enfrentan el problema, las variables controlables o aspectos del problema que puede controlar el tomador de decisiones, las variables no controlables por el tomador de decisiones que afectan el resultado de la selección, las restricciones de las variables controlables y no controlables, y los posibles resultados de la elección hecha por el tomador de decisiones.

Cuando se está resolviendo un problema se debe tener claro su objetivo, las variables controlables y no controlables. Para Ackoff, un problema se resuelve cuando el que toma las decisiones selecciona los valores de las variables controlables que optimizan el valor del resultado.

Los problemas del mundo real persiguen frecuentemente más de un objetivo, los cuales tienen diferentes grados de importancia o pesos para el o los tomadores de decisiones. Cuando se tiene un conjunto bastante amplio de soluciones factibles, el proceso de selección se vuelve algo complicado.

Existen diversas herramientas de software que incluyen métodos que ayudan en el proceso de toma de decisiones, sin embargo, puede ser complicado para el tomador de decisiones familiarizarse con nuevos entornos, herramientas o frameworks que permitan modelar sus problemas. Es por ello por lo que en este trabajo se propone una arquitectura interactiva que permita a un tomador de decisiones la interacción directa como medio de integración de preferencias en un proceso de optimización para demostrar que puede mejorar otras formas de incorporación de preferencias como a priori o a posteriori, sobre todo en la consecución de la región de interés.

1.1 Antecedentes

En seguida se presenta de forma resumida algunos trabajos que hacen uso del framework de optimización, MS-DOSS, o la herramienta de análisis de desempeño, VistHAA, durante la experimentación. En particular se analiza los problemas de optimización utilizados en el framework MS-DOSS, así como los algoritmos configurados, la versión utilizada MS-DOSS y vistHAA, y verificar si se ha hecho alguna aportación a alguna de las dos herramientas.

En Ponce (2021) se presenta una metodología de comunicación entre la herramienta VistHAA y el framework MS-DOSS, con el fin de robustecer el proceso de análisis de desempeño de algoritmos metaheurísticos. De la experimentación se comprobó que existía una comunicación mediante la metodología que se propuso en fase multiobjetivo.

En Fernández (2021) se explora la incorporación de las preferencias del tomador de decisiones (DM) en MOEA/D usando una incorporación a priori basada en relaciones de intervalos outranking para el manejo de imprecisiones. Se analiza el rendimiento de la propuesta usando problemas estándar contra el algoritmo clásico de MOEA/D que no incorpora preferencias. Los resultados demostraron que existe una mejora en la calidad de las soluciones relacionadas con el DM, cuando aumenta en número de objetivos.

En Castellanos (2022) se introduce una estrategia para enriquecer el algoritmo de optimización de lobos grises multiobjetivo (GWO) y el algoritmo de optimización de colonia de hormigas multiobjetivo basado en indicadores (ACO), a través de la representación basada en intervalos outranking de las preferencias del tomador de decisiones. Durante la experimentación se utilizaron los problemas estándar DTLZ y de los resultados obtenidos se concluyó que la propuesta es ideal para llegar a la región de interés (ROI), cuando se trabaja con varias funciones objetivo.

En la tabla 1 se resumen los trabajos revisados anteriormente.

Tabla 1. Trabajos del estado del arte.

	Problemas de optimización utilizados	Algoritmos Utilizados	Versión del framework MS-DOSS	Versión de la herramienta VistHAA	Aportación al framework MS-DOSS	Aportación la herramienta VistHAA
(Ponce, 2021)	- Knapsack - Selección de cartera de proyectos - DTLZ1-3	- Genético - NSGA-II	Java	Java	Ninguna	- Incorporación de los problemas y algoritmos MS-DOSS para su análisis. - Rediseño del algoritmo genético de VistTHAA
(Fernández, 2021)	- DTLZ1-9	- MOEA/D	Java	No utiliza	- Método de articulación de preferencias a priori	Ninguna
(Castellanos, 2022)	- DTLZ1-9	- GWO-InClass - ACO-InClass	Java	No utiliza	- Clasificador ordinal basado en outranking. - GWO-InClass. - ACO-InClass - Método de articulación de preferencias a priori	Ninguna

1.2 Estado del arte

En seguida se presenta de forma resumida algunos trabajos que integran servicios de Telegram, como la creación y configuración de bots, con otras herramientas de software. En particular se analiza la forma en que estos trabajos tratan la información que un usuario ingresa a Telegram para comunicarse con otros servicios, ya que esto puede ayudar al desarrollo del proyecto de tesis donde también se debe establecer una comunicación entre Telegram y el framework de optimización MS-DOSS y la herramienta de análisis de desempeño de algoritmos VisTHAA.

Cabe señalar que es necesario hacer uso del API de Telegram para la creación de bots, para ello hay que registrarse en https://telegram.me/BotFather. Al final del registro se proporciona un *token* que permite consumir el API de Telegram.

En Rosid (2018) se integra una aplicación web para la administración de quejas de una universidad, llamado *e-Complaint*, con el API de Telegram para la creación de bots. La información que es enviada por el usuario desde Telegram es recibida en formato JSON por una aplicación web donde se encuentra configurado su bot. Si el mensaje recibido es una queja, entonces la información se manda, en formato JSON, al sistema de quejas para ser procesado.

En Idhom (2018) se desarrolla un sistema que permite monitorear de forma remota a un servidor Linux en cualquier momento a través de un bot de Telegram. El administrador del servidor es el que tiene instalado Telegram y envía comandos al servidor a través del bot. Los comandos permiten obtener la siguiente información: el tiempo en línea del servidor, el estado del CPU, el estado de la memoria RAM, el tamaño del disco duro que es utilizado como memoria RAM, los usuarios y el estado de los discos. La comunicación con el servidor Linux es a través de peticiones HTTPS, la información intercambia en formato JSON.

En Daffa (2019) se desarrolla un sistema que utiliza un bot de Telegram y Arduino que ayuda a los padres a monitorear a sus niños en cualquier momento cuando no se encuentran en casa. El sistema puede enviar una foto del lugar en el que se encuentra instalado, detecta cuando el niño cruza la línea de seguridad (un sensor) y automáticamente envía una notificación a los padres para posteriormente cerrar la reja para evitar que el niño salga a algún lugar donde pueda lastimarse. El bot reconoce los siguientes comandos:

- Request monitor: Activa el módulo de la cámara para tomar una foto y después enviarla al chat de Telegram.
- Unlock and lock the door: Permite abrir o cerrar manualmente la puerta del área en la que el niño se encuentra.
- Set relay: Permite activar o desactivar el relé del sistema.

En Ben Akka (2019) se propone un sistema que permite monitorear y controlar un invernadero experimental como la temperatura, la humedad del aire, la presión atmosférica, entre otros, a través de Telegram. Cuando el usuario envía un comando a través de la aplicación de mensajería, su contenido se envía al bot que está conectado al microcontrolador ESP8266. El servidor local conectado a esta tarjeta administra la recepción y el procesamiento de la información para ejecutar una acción o para devolver información consultada.

En la tabla 2 se resumen los trabajos revisados anteriormente.

Tabla 2. Trabajos del estado del arte que integran Telegram con otros servicios.

	Formato de los datos	Intercambio de imágenes	Plataforma con la que se integra el bot de Telegram	Propósito
(Rosid, 2018)	JSON	No	Aplicación web para administración de quejas *e-complaint.*	Registrar quejas de alumnos en una plataforma web.
(Idhom, 2018)	JSON	No	Servidor Linux.	Obtener información del estado actual de un servidor Linux.
(Daffa, 2019)	JSON	Si	Servidor en tarjeta Arduino.	Monitorear niños en cierta área delimitada por una línea segura.
(Akka, 2019)	JSON	No	Servidor en microcontrolador.	Monitorear y controlar un invernadero experimental.

Como se puede observar en la tabla 2 todos los trabajos utilizan el formato JSON para el intercambio de información entre Telegram y el servicio con el que se integra el bot, en parte porque así lo define el API de Telegram, pero sobre todo porque es uno de los formatos preferidos para el envío y solicitud de mensajes ya que presentan los datos en una manera fácil de manejar para otras aplicaciones.

De los cuatro trabajos presentados, solo el desarrollado por Daffa (2019) implementa el intercambio de imágenes entre la cámara del sistema con el chat de Telegram. Gracias a esto se sabe que es posible intercambiar información más allá de texto, lo que es ideal debido a que en el proyecto de tesis se está contemplando la posibilidad de intercambiar gráficas generadas por la herramienta de análisis de desempeño.

Por otro lado, el propósito y la plataforma con la que se integra el bot de Telegram es diferente en cada trabajo, incluso con las intenciones de esta propuesta ya que, para este caso, el bot debe comunicarse con el framework de optimización, MS-DOSS o con la herramienta de análisis de desempeño de algoritmos, VisTHAA con el propósito de que el tomador de decisiones pueda hacer uso de los procesos que implementan dichos sistemas.

Cabe señalar que el proyecto de tesis tiene como uno de sus objetivos el poder ayudar a la toma de decisiones, objetivo que no se comparte con los trabajos presentados anteriormente. En la tabla 3 se resume la comparación de la propuesta de tesis contra los trabajos del estado del arte.

Tabla 3. Comparación de la propuesta con el estado del arte.

	Formato de los datos	Intercambio de imágenes	Plataforma con la que se integra el bot de Telegram	Propósito	Apoya a la toma de decisiones
(Rosid, 2018)	JSON	No	Aplicación web para administración de quejas *e-complaint.*	Registrar quejas de alumnos en una plataforma web.	No
(Idhom, 2018)	JSON	No	Servidor Linux.	Obtener información del estado actual de un servidor Linux.	No
(Daffa, 2019)	JSON	Si	Servidor en tarjeta Arduino.	Monitorear niños en cierta área delimitada por una línea segura.	No
(Akka, 2019)	JSON	No	Servidor en microcontrolador.	Monitorear y controlar un invernadero experimental.	No
Propuesta	JSON	Si	Framework de optimización MS-DOSS o herramienta de análisis de desempeño VisTHAA.	Permitir a un tomador de decisiones hacer uso de los procesos que se implementan en ambos sistemas de manera sencilla.	Si

Debido a que en el proyecto de investigación se utiliza un algoritmo de descomposición con integración de preferencias interactivo, se han revisado algunos trabajos del estado del arte que utilizan MOEA e integran preferencias de manera a priori, posteriori e interactivas que se resumen en la tabla 4:

	Propuesta	Descripción	Modificación al MOEA/D	Problema que utiliza en la experimentación	Resultado

(Fernández, 2021)	MOEA/D/O	En este artículo se explora como incorporar las preferencias del DM al MOEA/D utilizando una incorporación de preferencias a priori basadas en relaciones de intervalos outranking	Fase de actualización: Se complementa la norma Tchebychev con una relación de preferencia	DTLZ1 a DTLZ9	Se obtiene una mejora en la calidad de las soluciones relacionadas a los DMs, en comparación al MOEA/D clásico.
(Li, 2019)	I-MOEA/D-PLVF	En este artículo se utiliza el MOEA/D clásico como base y posteriormente se desarrolla su versión interactiva que es capaz de encontrar soluciones preferidas por el DM de manera progresiva.	Ninguna. Se utiliza como base el MOEA/D clásico. Se utiliza un algoritmo de aprendizaje automático para manejar las preferencias del usuario. Posteriormente se traducen los resultados del algoritmo anterior y se envían al MOEA/D	DTLZ1 a DTLZ4	Se obtienen aproximaciones a las preferencias del DM con marcada superioridad en comparación al MOEA/D clásico.
(Cheng, 2015)	RVPA	En este artículo se propone el uso de vectores de referencia en el espacio objetivo para la articulación de preferencias a posteriori.	Ninguna. Las soluciones aproximadas al óptimo de Pareto obtenidas por un MOEA son enviadas al método RVPA.	DTLZ1 a DTLZ6	El RVPA propuesto es capaz de obtener soluciones en las regiones preferidas especificadas por los vectores de preferencia definidos en el espacio objetivo.

Tabla 4. Trabajos del estado del arte que integran diferentes articulaciones de preferencia.

1.3 Justificación

En el desarrollo de sistemas para dispositivos móviles, la interfaz es clave para el éxito de la aplicación. Bajo estos dispositivos la interacción entre el usuario y el sistema puede ir más allá de entradas de texto y botones, gracias a los diferentes sensores y periféricos propios del terminal.

Debido a lo anterior, la mayoría de los dispositivos móviles ya cuentan con agentes virtuales, con las que el propietario del aparato puede comunicarse con él a través de comandos de voz o texto, ya sea para hacerle preguntas, configurar el dispositivo o ejecutar tareas específicas en las aplicaciones preinstaladas.

Incluso, debido a la practicidad de los agentes virtuales o bots, estos han sido adoptados por sitios y aplicaciones web, generalmente para contestar preguntas frecuentes o dar asesoramiento.

Para poder brindar un mejor servicio por medio de los agentes virtuales, estos necesitan apoyo de otras herramientas que les brinden información para poder tomar decisiones.

Como consecuencia, ha surgido la necesidad de desarrollar un módulo interactivo que permita la conexión entre diferentes herramientas existentes para el análisis de desempeño y frameworks o bibliotecas que permitan definir diferentes problemas de optimización, instancias y algoritmos de solución.

La interacción es uno de los medios para introducir preferencias en un proceso de optimización. El uso de tecnología móvil facilita el involucramiento de parte de los tomadores de decisiones con respecto a estos procesos. Incrementar las herramientas que le permitan retroalimentar el proceso contribuirá a una mejor integración de módulos interactivos en algoritmos de optimización.

1.4 Objetivos

1.4.1 Objetivo general

Desarrollar un módulo interactivo para apoyo a procesos de optimización en el área de toma de decisión en presencia de imprecisión e incertidumbre.

1.4.2 Objetivos específicos

- Analizar métodos para interacción, optimización y análisis de algoritmos.
    - Analizar el framework de optimización MS-DOSS.
    - Analizar la herramienta gráfica VisTHAA.
    - Analizar el Bot API de Telegram.
    - Seleccionar problemas de optimización para la investigación
- Desarrollar el Modelo Interactivo
    - Analizar requerimientos para modelo
    - Diseñar Modelo que permita interacción con un algoritmo de descomposición
    - Implementarlo modelo de interacción
    - Desarrollar Modelo de Retroalimentación del DM

- Desarrollar la Arquitectura de Integración para interacción, optimización y análisis de algoritmos
    - Analizar y delimitar requerimientos para chatbot, framework, y visthaa
    - Diseñar protocolo de comunicación del chat bot.
    - Diseñar interfaz de comunicación.
    - Diseñar módulo de integración de VisTHAA, MS-DOSS, y Telegram.
    - Implementar el módulo de integración.
- Validar el Módulo de Interacción.
    - Implementar Modelo A priori (MOEA/D/O)
    - Implementar Modelo A posteriori (MOEA/D+TOPSIS)
    - Definir indicadores de desempeño
    - Desarrollar diseño experimental
    - Recopilar Resultados
    - Concluir

1.5 Alcances y limitaciones

Durante el desarrollo del proyecto se han identificado los siguientes alcances y limitaciones:

- Se trabajará con un algoritmo de optimización del tipo de descomposición.
- Intercambio de texto e imágenes entre bot y herramientas de la etapa de proceso.
- El intercambio de datos utilizará el formato JSON.
- Se utilizará el lenguaje de programación Java y JavaScript.
- El bot puede ser accedido desde cualquier dispositivo móvil en el que se pueda instalar Telegram, como dispositivos Android e IOS.
- Corpus delimitado al algoritmo de optimización seleccionado.
- Uso del módulo gráfico de VisTHAA.
- El formato de los datos estará delimitado por las herramientas de la etapa de proceso.
- Se utilizará el API DE TELEGRAM para implementar la parte interactiva.
- Se considerará al menos como herramienta de análisis de algoritmos a VisTHAA.
- Se considerará al menos el algoritmo MOEA/D/O por algoritmo de solución
- Se considerará el Framework MS-DOSS como medio de implementación de los procesos de optimización
- Se considerará al menos TOPSIS como estrategia para el manejo de preferencias a posteriori.
- Se considerará al menos una estrategia para el manejo de preferencias a priori.

1.6 Organización del documento

En esta sección se presenta una breve descripción de los capítulos que conforman esta tesis.

El capítulo 2 presenta conceptos teóricos que permitirán un mejor entendimiento sobre el proyecto de esta tesis, de los cuales se destaca la definición del algoritmo de descomposición, información sobre articulación de preferencias y la documentación de las herramientas MS-DOSS, VisTHAA y Telegram, las cuales conforman la parte central de la arquitectura propuesta en esta Tesis.

En el capítulo 3 se explica la metodología utilizada para el cumplimiento de los objetivos específicos mencionados en la sección 1.5.2.

En el capítulo 4 se muestran los resultados obtenidos de las experimentaciones realizadas de…

El capítulo 5 describe las conclusiones obtenidas de la investigación realizada, así como sugerencias para trabajos futuros.

Capítulo 2 Marco Teórico

A continuación, se presentan los términos más importantes que permiten un mejor entendimiento del proyecto de tesis.

2.1 Toma de decisiones

Los problemas de toma de decisiones son procesos complejos en los cuales intervienen múltiples criterios, por lo cual es necesario utilizar herramientas que permitan discernir entre estos para obtener una solución que satisfaga en mejor grado la combinación de alternativas posibles (Osorio, 2008).

Cuando se enfrenta el proceso de toma de decisiones o selección de alternativas, generalmente se tienen múltiples objetivos, que se contraponen entre ellos, haciendo más complejo este proceso y generándose entonces la necesidad de una herramienta o un método que permita comparar esos múltiples criterios frente a la gama de alternativas posibles.

Cualquier actividad involucra de una u otra manera, la evaluación de un conjunto de alternativas en términos de un conjunto de criterios de decisión, donde muy frecuentemente estos criterios están en conflicto unos con otros.

Es vital contar con la información adecuada para tomar la mejor decisión, la cual se determinará dentro de un conjunto de posibles alternativas, las cuales deben ser evaluadas frente a múltiples criterios que se definan para este propósito.

En la toma de decisiones se debe tener en cuenta:

- **Efectos futuros**. Una decisión tiene una influencia a corto o largo plazo.
- **Reversibilidad.** Se refiere a la velocidad con que una decisión puede revertirse.
- **Impacto.** Medida en que otras áreas o actividades se ven afectadas.
- **Calidad.**
- **Periodicidad.** Frecuencia con la que se toma una decisión

2.2 Intervalos

Un intervalo es un rango de valores que recaen entre dos límites. Los intervalos permiten modelar la imprecisión que proviene de medidas imprecisas o por creencias del DM (Fernández, 2021).

Un intervalo se puede representar de la siguiente manera $E = \left[\underline{E}, \overline{E}\right]$, donde $\underline{E}$ y $\overline{E}$ corresponden a los límites inferior y superior.

La suma de dos intervalos *D* y *E* se puede realizar de la siguiente manera:

$$D + E = \left[\underline{D} + \underline{E}, \overline{D} + \overline{E}\right]$$

La resta de dos intervalos se realiza de la siguiente manera (Del Ángel, 2017):

$$D - E = \left[\underline{D} - \overline{E}, \overline{D} - \underline{E}\right]$$

La multiplicación se denota de la siguiente forma (Del Ángel, 2017):

$$D \times E = \left[\min\left(\underline{DE}, \underline{D}\overline{E}, \overline{D}\underline{E}, \overline{DE}\right), \max\left(\underline{DE}, \underline{D}\overline{E}, \overline{D}\underline{E}, \overline{DE}\right)\right]$$

La división se denota (Del Ángel, 2017):

$$D \div C = \left[\underline{D}, \overline{D}\right] \times \left[\frac{1}{\underline{C}}, \quad \frac{1}{\overline{C}}\right]$$

Las relaciones ≥ y > en intervalos se definen usando la función de probabilidad $P(E \geq D)$. Esta función se define como (Fernández, 2021):

$$P(E \geq D) = \begin{cases} 1 & si\ PED > 1, \\ PED & si\ 0 \leq PED \leq 1, \\ 0 & si\ PED \leq 0 \end{cases}$$

Donde

$$PED = \frac{\overline{E} - \underline{D}}{\left(\overline{E} - \underline{E}\right) + \left(\overline{D} - \underline{D}\right)}$$

2.3 Problemas estándar

La siguiente información es extraída de (Li, 2015).

Los problemas de prueba DTLZ fueron construidos por Deb en 2005, y constan de una serie de funciones de prueba comúnmente usadas para evaluar y comparar el rendimiento de los algoritmos evolutivos, lo cual también contribuye a su mejora. Hay un total de nueve funciones en la suite DTLZ y cada una tiene su propia naturaleza de manera que se pueda probar una habilidad especifica de los algoritmos evolutivos.

2.3.1 DTLZ1

Está compuesto por *M* objetivos y es un problema de prueba simple debido a que su límite óptimo es un hiperplano lineal, el cual se puede expresar con la siguiente ecuación:

$$\begin{cases} \min f_1(x) = \frac{1}{2} x_1 x_2 \dots x_{M-1}[1 + g(X_M)] \\ \min f_2(x) = \frac{1}{2} x_1 x_2 \dots (1 - x_{M-1})[1 + g(X_M)] \\ \dots \\ \min f_{M-1}(x) = \frac{1}{2} x_1 (1 - x_2)[1 + g(X_M)] \\ \min f_M(x) = \frac{1}{2} (1 - x_1)[1 + g(X_M)] \end{cases}$$

$$g(X_M) = 100 \left\{ |X_M| + \sum_{x_i \in X_M} (x_i - 0.5)^2 - \cos[20\pi(x_i - 0.5)] \right\}$$

Donde X_M es la última $N - M + 1$ variable de decisión en el vector de decisión x y $|X_M| = N - M + 1$. Para cada $i \{1, 2, \dots, n\}$, se satisface $0 \leq x_i \leq 1$. El límite óptimo objetivo es $\sum_{i=1}^{M} f_i = 0.5$.

2.3.2 DTLZ2

Esta función tiene una forma similar a DTLZ1, su límite óptimo es una hiper esfera que satisface la función $\sum_{i=1}^{M} f_1^2 = 1$ y se puede expresar con la siguiente ecuación:

$$\begin{cases} \min f_1(x) = \cos\left(\frac{\pi}{2}x_1\right) \dots \cos\left(\frac{\pi}{2}x_{M-1}\right)[1 + g(X_M)] \\ \min f_2(x) = \cos\left(\frac{\pi}{2}x_1\right) \dots \sin\left(\frac{\pi}{2}x_{M-1}\right)[1 + g(X_M)] \\ \dots \\ \min f_{M-1}(x) = \cos\left(\frac{\pi}{2}x_1\right)\sin\left(\frac{\pi}{2}x_2\right)[1 + g(X_M)] \\ \min f_M(x) = \cos\left(\frac{\pi}{2}x_1\right)[1 + g(X_M)] \end{cases}$$

$$g(X_M) = \sum_{x_i \in X_M} (x_i - 0.5)^2$$

2.3.3 DTLZ3 y DTLZ4

Ambas funciones están construidas a partir de las funciones de prueba DTLZ2, por lo que tienen el mismo límite óptimo que DTLZ2.

2.3.4 DTLZ6 y DTLZ7

Ambas funciones están principalmente deseñadas para probar la convergencia en una curva especifica. Tiene una forma similar a DTLZ2, pero su límite óptimo es la curva que atraviesa todo el espacio y su función objetivo satisface $\sum_{i=1}^{m} f_1^2 = 1$ si y solo si *xi* en *XM* es igual a 0.5.

Se puede expresar con la siguiente ecuación:

$$\begin{cases} \min f_1(x) = \cos\left(\frac{\pi}{2}\theta_1\right) \dots \cos\left(\frac{\pi}{2}\theta_{M-1}\right)[1 + g(X_M)] \\ \min f_2(x) = \cos\left(\frac{\pi}{2}\theta_1\right) \dots \sin\left(\frac{\pi}{2}\theta_{M-1}\right)[1 + g(X_M)] \\ \dots \\ \min f_{M-1}(x) = \cos\left(\frac{\pi}{2}\theta_1\right)\sin\left(\frac{\pi}{2}\theta_2\right)[1 + g(X_M)] \\ \min f_M(x) = \cos\left(\frac{\pi}{2}\theta_1\right)[1 + g(X_M)] \end{cases}$$

$$o_i = \frac{\pi}{4\left(1 + g(X_M)\right)}[1 + 2g(X_M)x_i], (i = 2, 3, \dots, M - 1)$$

$$g(X_M) = \sum_{x_i \in X_M} (x_i - 0.5)^2$$

2.4 Optimización

Vitoriano en el 2010 (Vitoriano, 2010) escribió un libro de modelos matemáticos de optimización del cual se obtienen todas las siguientes definiciones. La investigación operativa se puede definir como la aplicación de métodos científicos en la mejora de la efectividad en las operaciones, decisiones y gestión. o como la ciencia de aplicar los recursos disponibles para conseguir la satisfacción óptima de un objetivo específico deseado.

La principal característica consiste en construir un modelo científico del sistema del cual se pueden predecir y comparar los resultados de diversas estrategias, decisiones, incorporando medidas del azar y del riesgo El objetivo es ayudar a los responsables a determinar su política y actuaciones en forma científica.

Los profesionales de la investigación operativa colaboran con los decisores en el diseño y mejora de las operaciones y decisiones, resuelven problemas y ayudan en las funciones de gestión, planificación o predicción, aportan conocimiento y ayuda en la toma de decisiones. Aplican las técnicas científicas más adecuadas seleccionadas de la matemática, ingeniería o cualquier ciencia social o de administración de empresas. Su trabajo normalmente consiste en recoger y analizar datos, desarrollar y probar modelos matemáticos, proponer soluciones o recomendaciones, interpretar la información y, en definitiva, ayudar a implantar acciones de mejora. Como resultado desarrollan e implantan aplicaciones informáticas, sistemas, servicios técnicos o productos.

Tipos de optimización:

- Lineal.
- No lineal.
- Entera.
- Estocástica.
- Multiobjetivo.

La optimización es una parte relevante dentro de la investigación operativa. Tuvo un progreso algorítmico inicial muy rápido.

Los problemas de optimización se componen generalmente de estos tres ingredientes:

Función objetivo.

Es la medida cuantitativa del funcionamiento del sistema que se desea optimizar (maximizar o minimizar).

Ejemplos de función objetivo: la minimización de los costes variables de operación de un sistema eléctrico, la maximización de los beneficios netos de venta de ciertos productos, la minimización del cuadrado de las desviaciones con respecto a unos

valores observados, la minimización del material utilizado en la fabricación de un producto, etc.

Variables

Representan las decisiones que se pueden tomar para afectar el valor de la función objetivo. Se pueden clasificar en variables independientes, principales o de control y variables dependientes, aunque matemáticamente son iguales.

Restricciones

Representan el conjunto de relaciones (expresadas mediante ecuaciones e inecuaciones) que ciertas variables están obligadas a satisfacer. Por ejemplo, las potencias máxima y mínima de operación de un grupo de generación, la capacidad de producción de la fábrica para los diferentes productos, las dimensiones del material bruto del producto, etc.

Resolver un problema de optimización consiste en encontrar el valor que deben tomar las variables para hacer óptima la función objetivo satisfaciendo el conjunto de restricciones.

Los métodos de optimización se pueden clasificar en:

- Clásicos.
- Metaheurísticos. Aparecieron ligados a lo que se denominó inteligencia artificial e imitan fenómenos sencillos observados en la naturaleza.

Como menciona (Ramos, 2010) resolver un problema de optimización consiste en encontrar el valor que deben tomar las variables para hacer óptima una función objetivo y satisfacer el conjunto de restricciones. En los cuales existen algunos tipos de problemas de optimización que se alteran medianamente este esquema:

- Sistemas de ecuaciones lineales – no lineales en el cual no existe una función objetivo con tal y únicamente interesa encontrar una solución factible a un problema de un conjunto de restricciones.
- Optimización sin restricciones en el que se trata de encontrar un conjunto de valores que indican el mínimo/máximo de una función.
- Optimización multiobjetivo en el cual existen más de una función objetivo y el problema que se plantea es como tratar varias funciones objetivo al mismo tiempo tomando en cuenta que el valor objetivo de una función no es el mismo para otra.

Los métodos de optimización se pueden clasificar como métodos clásicos los cuales habitualmente se explican en la literatura y los métodos metaheurísticos que están ligados a lo que se denominó inteligencia artificial. Dentro de los métodos clásicos

podemos encontrar la optimización lineal, lineal entera mixta, dinámica, estocástica, no lineal etc. En el segundo se incluyen los algoritmos evolutivos (genéticos entre otros), búsquedas heurísticas como el método tabú, búsqueda aleatoria, etc. De manera general y aproximada se puede decir que los métodos clásicos garantizan encontrar el óptimo local y los metaheurísticos dadas las mecánicas específicas que tienen les permite con frecuencia alcanzar los óptimos globales.

Las restricciones que definen las cotas del problema dividen el espacio de búsqueda en dos las cuales son:

- Soluciones factibles: Son aquellos elementos en el espacio de búsqueda que cumplen con las restricciones.
- Soluciones no factibles: Son aquellas soluciones que no toman parte del dominio de la función porque viola al menos una restricción del problema.

2.4.1 Algoritmo Evolutivo Multiobjetivo basado en Descomposición

Los algoritmos evolutivos para optimización multiobjetivo (MOEA) son capaces de generar como salida un conjunto de soluciones compromiso en una sola ejecución del algoritmo (Sánchez, 2017). Los MOEAs ayudan al DM a tomar la solución final que más se adecúe a sus preferencias, sin embargo, también proporcionan un amplio número de soluciones, lo cual dificulta el proceso de selección (Sánchez, 2017).

Un Algoritmo Multiobjetivo basado en Descomposición (MOEA/D) descompone un problema de optimización multiobjetivo en un número de subproblemas de optimización escalares para después optimizarlos simultáneamente (Zhang, 2007). Las relaciones de vecindad de estos subproblemas entre estos subproblemas se definen en base en las distancias entre sus vectores de coeficientes de aceptación. La solución óptima a dos subproblemas vecinos debe ser muy similares.

Funcionamiento del algoritmo (Zhang, 2007):

Entrada:

- Problema Multiobjetivo.
- Criterio de Parada
- *N*: el número de subproblemas considerados en MOEA/D
- Una dispersión uniforme de *N* vectores de peso $\lambda^1, \dots, \lambda^N$
- *T*: el número de vectores de peso en la vecindad de cada vector de peso.

Salida: Población externa (EP).

Paso 1: Inicialización:

1.1 Definir EP = Ø

1.2 Calcular la distancia euclidiana entre cualquier par de vectores de peso y después utilizar los *T* vectores de peso más cercanos a cada vector de peso. For *i*=1,…,*N, establecer B*(*i*) = {i_1,…,i_T}, donde $\lambda^{i1}, \dots, \lambda^{iT}$ son los *T* vectores de peso cercanos a λ^i.

1.3 Generar una población inicial $x^1, \dots, x^N$ aleatoriamente o por un método específico al problema. Establecer $FV^i = F(x^i)$

1.4 Inicializar $z = (z_1, \dots, z_m)^T$ a través de un método específico al problema.

Paso 2: Actualización:

For i = 1,…,*N*, hacer:

2.1 Reproducción: aleatoriamente seleccionar dos índices *k,l* de *B*(*i*), después generar una nueva solución *y* de x^k y *xl* utilizando operadores genéticos.

2.2 Mejora: Aplicar una heurística de reparación/mejora específica al problema en *y* para producir *y'*

2.3 Actualización de *z*: $For\ j = 1, \dots, m, if\left(z_j\right) < f_i(y'), entonces\ z_j = f_j(y')$

2.4 Actualización de las soluciones vecinas: Para cada índice:

$j \in B(i), \text{if } g^{te}(y'|\lambda^j, z) \leq g^{te}(x^j|\lambda^j, z)$ después $x^j = y'$ y $FV^j = F(y')$.

2.5 Actualización de EP:

Remover de EP todos los vectores dominados por *F*(*y'*).

Añadir *F*(*y'*) a EP si no hay ningún vector en EP dominan *F*(*y'*).

Paso 3: Criterio de paro

Si el criterio de paro se satisface, entonces parar y regresar a EP. En caso contrario ir al paso 2.

2.4.2 Herramienta de análisis de desempeño de algoritmos

2.4.2.1 Análisis de algoritmos

La resolución práctica de un problema exige por una parte un algoritmo o método de resolución y por otra un programa o codificación de aquel en un ordenador real. Ambos componentes tienen su importancia; pero la del algoritmo es absolutamente esencial, mientras que la codificación puede muchas veces pasar a nivel de anécdota.

Un algoritmo es una serie de pasos que nos llevan a resolver efectivamente un problema (Mañas, 2017), mientras que un programa es un algoritmo escrito usando un lenguaje de programación.

Ante un mismo problema, puede haber mejores ideas y peores ideas acerca de cómo afrontar su solución. En lo que sigue sólo nos ocuparemos de algoritmos

correctos, es decir, que nos llevan a una solución válida del problema planteado. Dentro de lo que es correcto, el análisis de algoritmos nos lleva a poder decir si una idea es mejor que otra.

Los criterios para evaluar programas son diversos: eficiencia, portabilidad, eficacia, robustez, entre otros. Entendemos por eficiencia de un algoritmo la cantidad de recursos de cómputo que requiere; es decir, cuál es su tiempo de ejecución y qué cantidad de memoria utiliza (Duch, 2007). A la cantidad de tiempo que requiere la ejecución de un cierto algoritmo se le suele llamar coste en tiempo mientras que a la cantidad de memoria que requiere se le suele llamar coste en espacio.

El análisis de complejidad está relacionado con la eficiencia del programa. La eficiencia mide el uso de los recursos del computador por un algoritmo. Por su parte, el análisis de complejidad mide el tiempo de cálculo para ejecutar las operaciones (complejidad en tiempo) y el espacio de memoria para contener y manipular el programa más los datos (complejidad en espacio). Así, el objetivo del análisis de complejidad es cuantificar las medidas físicas: "tiempo de ejecución y espacio de memoria" y comparar distintos algoritmos que resuelven un mismo problema (Scalise, 2001).

El tiempo de ejecución de un programa depende de factores como:

- Los datos de entrada del programa.
- La calidad del código objeto generado por el compilador.
- La naturaleza y rapidez de las instrucciones de máquina utilizadas.
- La complejidad en tiempo del algoritmo base del programa.

Existen diferentes herramientas utilizadas para el cálculo de complejidad de algoritmos en tiempo y espacio, como (Scalise, 2001):

- Tasas de crecimiento.
- Análisis de complejidad de tiempo.
- Análisis de complejidad en espacio.
- Análisis de complejidad en programas recursivos.

Es evidente que conviene buscar algoritmos correctos que mantengan tan bajo como sea posible el consumo de recursos que hacen del sistema, es decir, que sean lo más eficientes posible. Cabe hacer notar que el concepto de eficiencia de un algoritmo es un concepto relativo, esto quiere decir que ante dos algoritmos correctos que resuelven el mismo problema, uno es m´as eficiente que otro si consume menos recursos. Por tanto, podemos observar que el concepto de eficiencia y en consecuencia el concepto de coste nos permitir´a comparar distintos algoritmos entre ellos (Duch, 2007).

2.4.3 Framework de optimización

2.4.3.1 Framework

La siguiente información se extrajo de (Muente, 2020).

Es un conjunto de archivos y directorios que facilitan la creación de aplicaciones, ya que incorporan funcionalidades ya desarrolladas y probadas, implementadas en un determinado lenguaje de programación.

El objetivo principal de todo framework es facilitar las cosas a la hora de desarrollar una aplicación, haciendo que nos centremos en el verdadero problema y nos olvidemos de implementar funcionalidades que son de uso común como puede ser el registro de un usuario, establecer conexión con la base de datos, manejo de sesiones de usuario o el almacenamiento en base de datos de contenido cacheado.

Los frameworks permiten entregar un proyecto en menos tiempo y con un código más limpio, cuya eficacia ya ha sido comprobada. A partir del framework los programadores pueden complementar y/o modificar la estructura base para entregar el software o la aplicación que cumpla los objetivos requeridos.

Ventajas de utilizar un framework para el desarrollo de software:

- El programador ahorra tiempo ya que dispone ya del esqueleto sobre el que desarrollar una aplicación.
- Facilita los desarrollos colaborativos, al dejar definidos unos estándares de programación.
- Al estar ampliamente extendido, es más fácil encontrar herramientas, módulos e información para utilizarlo.
- Proporciona mayor seguridad, al tener gran parte de las potenciales vulnerabilidades resueltas.
- Normalmente existe una comunidad detrás, un conjunto de desarrolladores que pueden ayudar a responder consultas.

Desventajas de utilizar un framework:

- Se requiere de cierto tiempo de aprendizaje.
- Puede existir exceso de líneas de código.
- Limitación en cuanto a las modificaciones que se pueden hacer sobre el framework.

Tipos de frameworks:

- Para aplicaciones web. Son aquellos frameworks que se utilizan específicamente para la creación de proyectos online.
- Para aplicaciones en general. Permite complementar la estructura de una aplicación para un sistema operativo.

- Para tecnología de JavaScript Asíncrono con XML (AJAX). La tecnología AJAX permite que el usuario haga solicitaciones al servidor sin que sea necesario recargar una página después de cada nueva solicitud.
- De gestión de contenidos. facilita la programación de aplicaciones de un Sistema de Gestión de Contenidos, popularmente conocido como CMS, por ejemplo, WordPress.
- De Multimedia. facilita el trabajo de los programadores que trabajan con video, audio e imagen y colabora con la creación de las aplicaciones multimedia en general, pudiendo servir para proyectos más complejos, como videoconferencias y conversores de medios.

2.4.3.1 Framework MS-DOSS

Es un framework de optimización que permite la declaración de problemas de optimización, así como la definición y configuración de algoritmos solucionadores a un problema dado (Ponce, 2021).

En la tabla 5 se describen las características de los módulos del framework de optimización, el cual es parte fundamental en la construcción del proyecto.

Tabla 5. Módulos MS-DOSS

Módulo	Descripción
Validación	Mediante el uso de pruebas estadísticas como Friedman y Wilcoxon, permite seleccionar las mejores soluciones creadas por los algoritmos, en función de cada problema. Este módulo está compuesto por las clases: *Validate, StatisticsSet, Test_Friedman, Test_Wilcoxon, Operator_Statistics, Mean, Variance y Standard_Desviation*.
Preferencia	Este módulo aborda un sistema relacional de preferencias compuesto de varias relaciones binarias como: Indiferencia, Preferencia estricta, Preferencia débil, Incomparabilidad, K-preferencia y No preferencia. se compone por las clases: *Preference* y *ELECTRE III*
Reporte	Este módulo presentara los resultados obtenidos del módulo “Validación”, está compuesto por la clase: *Report.*
Instancia	Este módulo está diseñado para contener la instancia de cualquier problema, de tal manera al generalizar su información, pueda ser abordado por el módulo “Problema”; sin importar el formato que utilicen los autores

	de cada instancia. Está compuesto por: *Instance, JSSP_Inst y Thompson_JSSP*.
Problema	Este módulo permite generalizar distintos problemas de optimización, enfatizando sus características comunes. Actualmente este módulo está compuesto por las clases: *Problem, JSSP y PPP.*
Algoritmo	Este módulo permite manipular la información generada por módulo “Problema”, según el comportamiento de cada algoritmo. Este permite generalizar distintos algoritmos, enfatizando sus características comunes, facilitando la creación y extensión de sus clases. Este módulo está compuesto por las clases: *Algorithm, Genetic, SimulatedAnnealing y NOSGA.*
Operadores	Este módulo contiene un conjunto de operadores en el área de optimización, como p. e. Cruza, Mutación, Selección, etc. Estos son necesarios para llevar a cabo la ejecución de múltiples algoritmos como p.e. el Genético, Recosido Simulado y NOSGA (todos estos dentro del módulo “Algoritmo”). Este módulo está compuesto por las clases*: Operator, Crossover, Mutation, Selection*.
Solución	Este módulo permite generalizar las soluciones creadas por el módulo “Algoritmo”; que, de otra manera, se tendría que crear tantas clases de soluciones como existan algoritmos en el framework. Este módulo está compuesto por las clases: *Solution* y *SolutionSet*.
Diseño experimental	Este módulo es uno de los más importantes de este framework, ya que permite realizar una experimentación sencilla y cómoda. Además, también permite configurar fácilmente los parámetros pertenecientes a módulos de optimización (p. e. “Problema”, “Algoritmo”, etc.). De esta manera evitemos la configuración directa a esos módulos. Actualmente este módulo está compuesto por las clases: *Parameter, Parameters,* *Exp_Design, Gen_Exp, SA_Exp y NOSGA_Exp e Instance_Exp*.

2.5 Chatbots

La definición clásica de un chatbot es (Khan, 2018): un programa que procesa una entrada de lenguaje natural de un usuario y que genera respuestas inteligentes y en contexto, que son enviadas de vuelta al usuario. El termino *chatterbot* fue usado por primera vez en 1994 y fue originalmente acuñado por Michael Mauldin, el creador del robot verbal Julia.

Actualmente los chatbots están potenciados por motores impulsados por reglas o motores de inteligencia artificial (IA) que interactúan con los usuarios generalmente a través de interfaces basadas en texto. Son programas de computadora independientes que se pueden integrar a cualquiera de las plataformas de mensajería que permitan el uso a los desarrolladores a través de un API, como lo hace Facebook Messenger, Slack, Skype, Microsoft Teams, entre otros.

Con el avance de la tecnología de voz en años recientes, compañías como Google, Apple y Amazon han debutados sus agentes de inteligencia artificial por voz. Apple introdujo a Siri en los iPhone, iPad y macOS. Google lanzó Google Home y Amazon lanzó a Alexa, siendo ambos dispositivos físicos para el hogar u oficina que pueden ayudar en tareas como contratar un taxi, encender o apagar las luces, reproducir música en Spotify, administrar el calendario, entro otros (Khan, 2018).

La gran ventaja de utilizar una interfaz basada en chat es que se le puede permitir al usuario llevar a cabo sus intenciones en lenguaje natural, como si hablara con otra persona. Desde el punto de vista del desarrollador el texto en lenguaje natural es una de las interfaces más difíciles de controlar. Una vez que se recibe la petición de texto en lenguaje natural, el desarrollador debe convertir el texto en bloques entendibles para la aplicación del chatbot para así poder generar una respuesta. Aun así, podría llegar a ser tedioso para el usuario, en algún momento, el tener que estar escribiendo cada petición en lenguaje natural, por ello las plataformas de mensajería introdujeron varios elementos de interfaz de usuario que permitan mostrar fácilmente cierto tipo de información y que a su vez permita al usuario dar respuestas al bot con tan solo tocar un botón.

En general, todas las plataformas de mensajería populares proveen a los desarrolladores con enormes bases de consumo que pueden ser aprovechados para desarrollar múltiples servicios vía chat (Khan, 2018), estas bases de consumo pueden ser utilizadas a través de un API.

2.6 Interfaz de Programación de Aplicaciones

La siguiente información se ha obtenido de (Ofoeda, 2019).

La Interfaz de programación de aplicaciones forma un componente integral del ecosistema de software. Los ecosistemas de software se han convertido en una manera ideal para la construcción de soluciones de software complejas en plataformas de tecnología populares.

Las APIs se desarrollaron principalmente para el intercambio de información entre dos o más programas. El mercado de aplicaciones móviles, que está creciendo rápidamente en áreas de las tecnologías de la información, hace uso de APIs. En general las APIs permiten la reutilización de código y mejoran la productividad del desarrollo de software.

La API es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones. Las API surgieron en los primeros días de la informática, mucho antes que la computadora personal. En esa época, una API normalmente se usaba como biblioteca para los sistemas operativos. Casi siempre estaban habilitadas localmente en los sistemas en los que operaban, aunque a veces pasaban mensajes entre las computadoras centrales. Después de casi 30 años, las API se expandieron más allá de los entornos locales. A principios del año 2000, ya eran una tecnología importante para la integración remota de datos.

Las API permiten que productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados. A su vez otorgan flexibilidad; simplifican el diseño, la administración y el uso de las aplicaciones, y proporcionan oportunidades de innovación, lo cual es ideal al momento de diseñar herramientas y productos nuevos.

Durante la comunicación o consumo del API sí una de las partes envía una solicitud remota con cierta estructura en particular, esa misma estructura determinará cómo responderá el software de la otra parte.

La mayoría de las API están diseñadas de acuerdo con los estándares web, por lo que normalmente usan Protocolos de Transferencia de Hipertexto (HTTP) para solicitar mensajes y proporcionar una definición de la estructura de los mensajes de respuesta. Por lo general, estos mensajes de respuesta toman la forma de un archivo de Lenguaje de Marcado Extensible (XML) o Notación de Objeto de JavaScript (JSON,) que son los formatos preferidos porque presentan los datos en una manera fácil de manejar para otras aplicaciones.

2.6.1 Telegram API

La plataforma de mensajería Telegram proporciona dos tipos de API de las cuales solo Bot API serán de utilidad para el desarrollo del proyecto de tesis, esta API permite crear programas que hacen uso de la interfaz de chat de Telegram.

Los bots de Telegram son cuentas especiales que no requieren un número telefónico adicional para ser creados. Estas cuentas sirven como una interfaz para ejecutar código en el servidor del desarrollador.

No se necesita conocer cómo funciona el protocolo de encriptación MTProto que utiliza Telegram, ya que su servidor intermediario se hará cargo de toda la encriptación y comunicación con la API de Telegram.

Para crear un bot de Telegram solo se tiene que hablar con el BotFather y seguir sus instrucciones. Una vez creado el bot se proporcionará al desarrollador un token de autorización.

Capítulo 3 Propuesta de solución

En esta sección se presenta la arquitectura de integración de los módulos que conforman la propuesta de solución, así como la descripción de cada una de ellas.

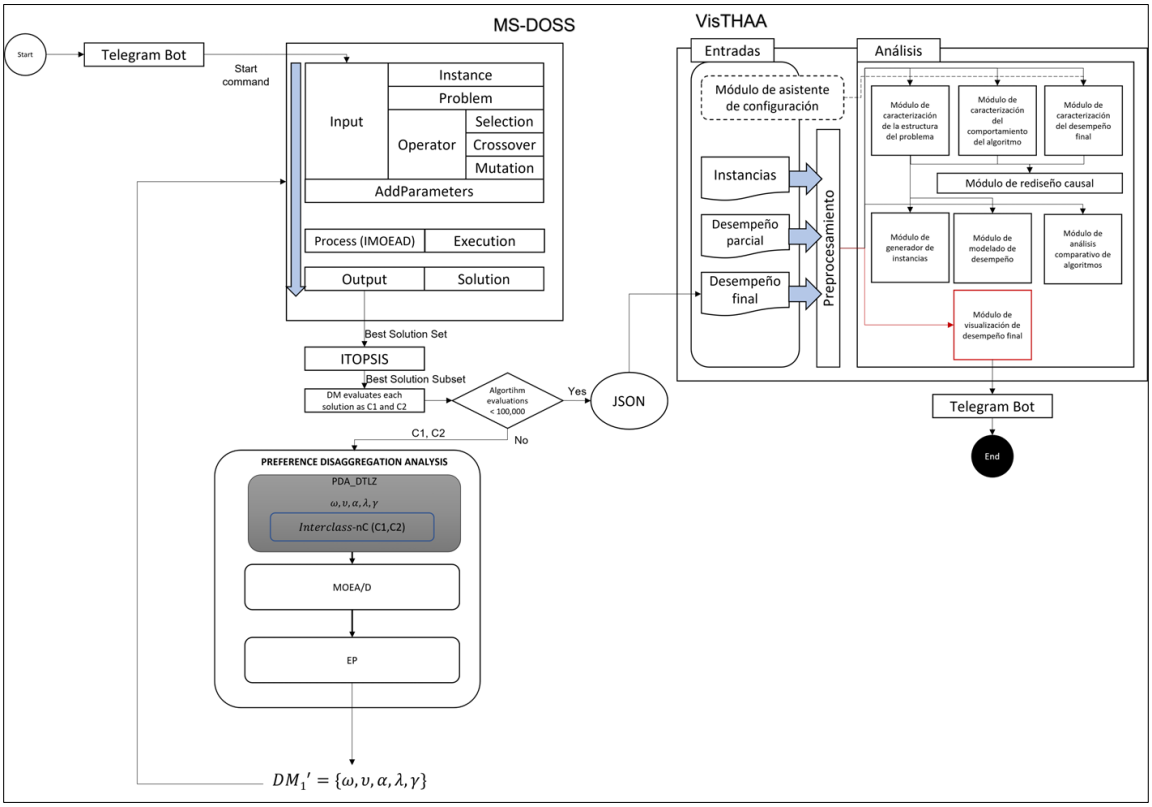

Diagrama 1. Arquitectura de integración.

En el diagrama 1 se puede observar que el DM inicia el proceso interactivo con el bot utilizando el comando principal. Posteriormente, el bot ejecuta un microservicio del framework de optimización MS-DOSS para resolver DTLZ utilizando MOEA/D/O (La primera vez se utilizan valores extremos en los parámetros de configuración del algoritmo). A partir del paso anterior, se obtienen N soluciones, que se ordenarán utilizando el método TOPSIS para elegir las 10 mejores. Luego, cada solución del

paso anterior se presenta al DM para que pueda seleccionar cuáles le gustan (C2) y cuáles no (C1). Si el número de evaluaciones del algoritmo principal es inferior a 100,000, entonces el PDA se resuelve utilizando MOEA/D y el conjunto de referencia evaluado por el DM en el paso anterior, obteniendo así un nuevo conjunto de parámetros que serán utilizados por el algoritmo principal (MOEA/D/O) en la siguiente iteración. Finalmente, se informa gráficamente el estado de las soluciones obtenidas al final del ciclo.

Telegram Bot

A continuación, se muestra la interacción que realiza el DM con el bot desarrollado en Telegram.

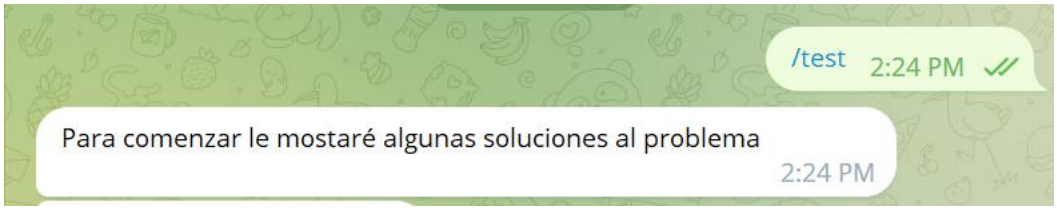

Imagen 1. Inicio del proceso interactivo.

En la imagen 1 se puede observar que el DM inicia el proceso interactivo con el bot al utilizar el comando principal.

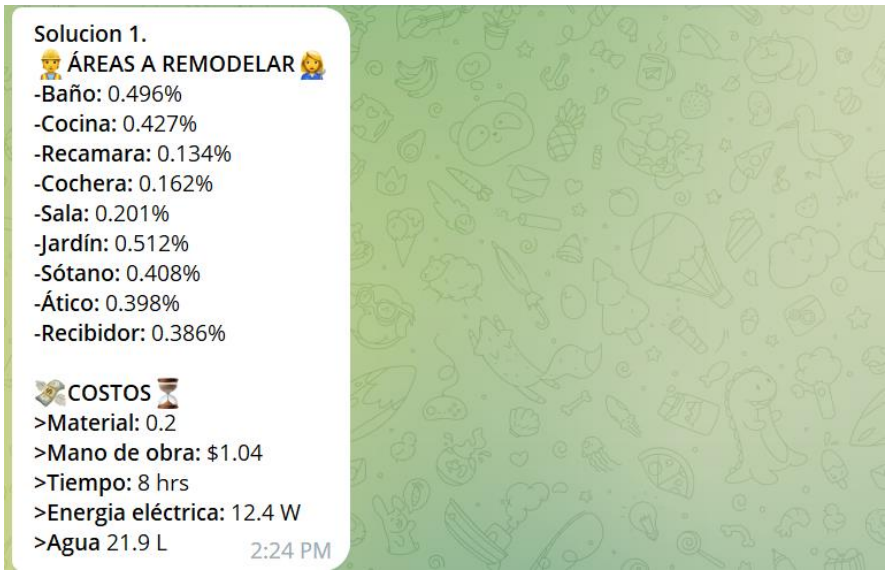

Imagen 2. Presentación de 10 soluciones aleatorias.

Posteriormente el bot resuelve un problema DTLZ con ayuda del framework MS-DOSS y posteriormente le presenta diez soluciones aleatorias al DM (imagen 2)

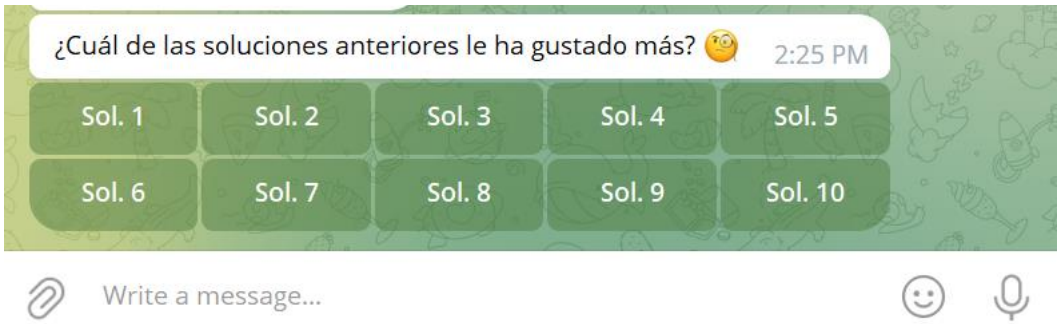

Imagen 3. Obtención de la solución compromiso.

De las diez soluciones aleatorias anteriores, el DM solo podrá elegir una. Este proceso se realiza una sola vez para así obtener la solución compromiso (imagen 3).

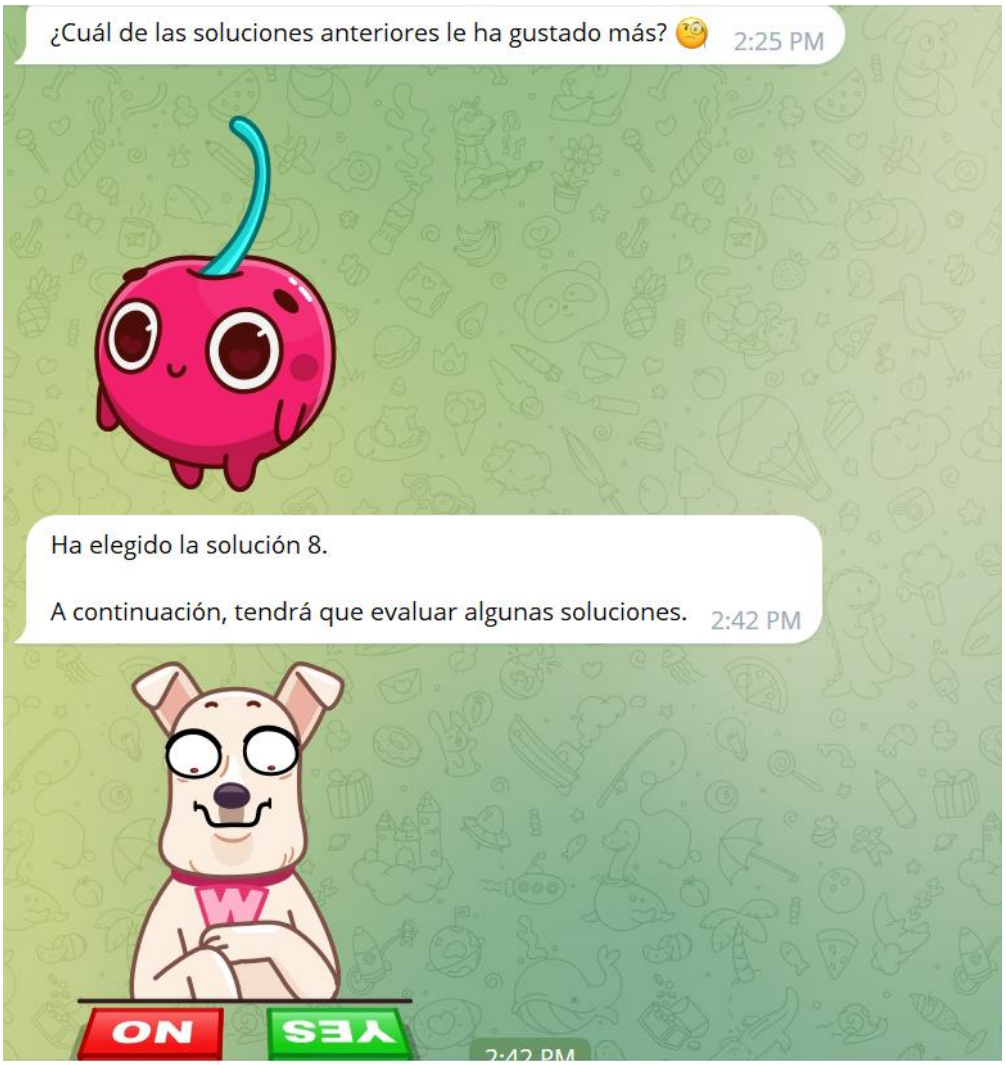

Imagen 4. Inicio del proceso de evaluación.

Una vez que el DM elige la solución compromiso (aquella que más le ha gustado de entre las diez soluciones aleatorias) se da inicio al ciclo principal del proceso interactivo (imagen 4).

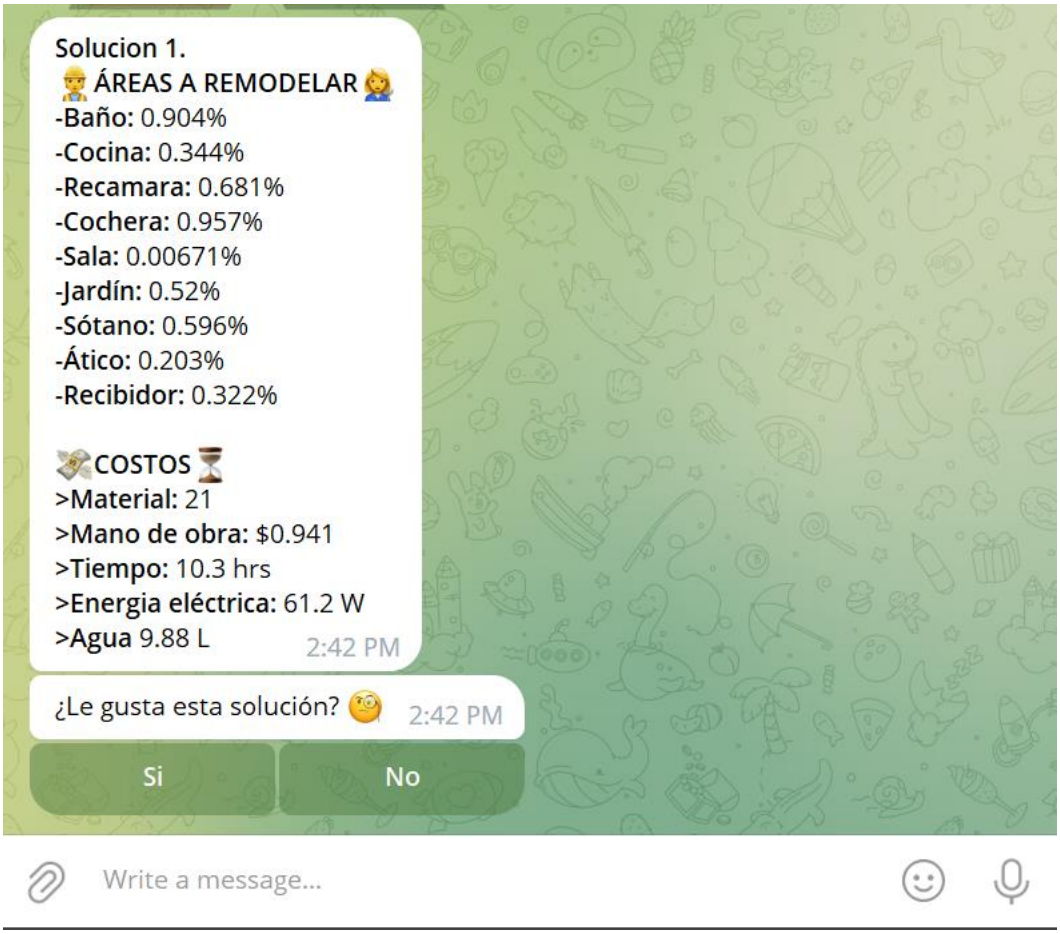

Imagen 5. Evaluación de las soluciones.

El bot le presentará diez soluciones al problema DTLZ, una por una, al DM, el cual solo tendrá que indicar si la solución le ha gustado o no (imagen 5). Posteriormente, el bot resolverá el PDA utilizando el algoritmo MOEA/D y el conjunto de soluciones evaluadas por el DM.

De lo anterior se obtendrá un nuevo conjunto de parámetros preferenciales que serán utilizados en la siguiente iteración del algoritmo MOEA/D/O (aquel que resuelve el problema DTLZ). Este proceso ocurrirá cada 20,000 evaluaciones del algoritmo MOEA/D/O hasta llegar a la condición de paro (alcanzar 100,000 evaluaciones).

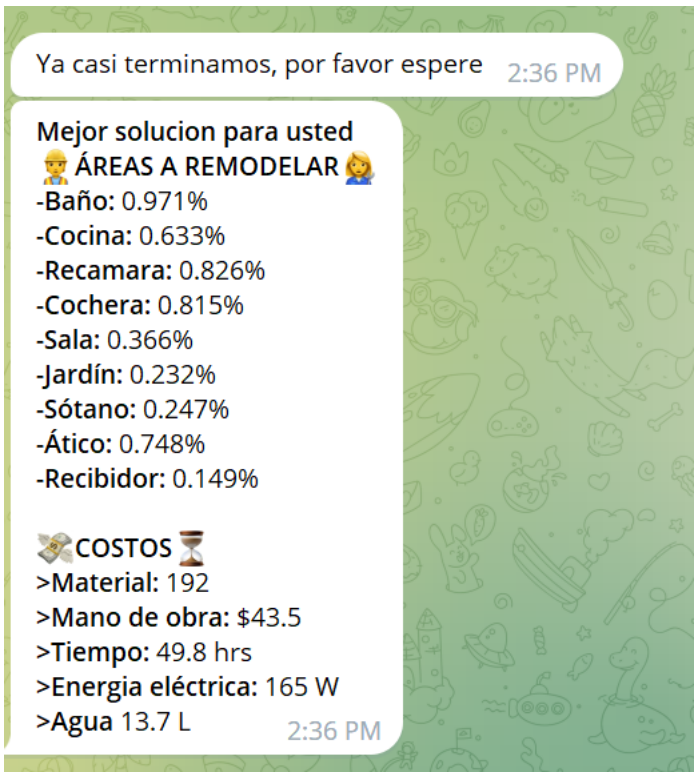

Imagen 6. Presentación de la solución final.

Al final del proceso interactivo se le mostrará la mejor solución al DM, de acuerdo con la estimación de sus preferencias (imagen 6).

MOEA/D/O

Algorithm 1. MOEA/D/O

Input:

m= objective number

N= scalar subproblems number

$\varpi=\{\varpi_1, \varpi_2, \ldots, \varpi_N\}$= set of N weight vectors

$T=N/10$, neigborhood size of each weight vector

DM= (w, v, λ, β), value system of a DM

R_k = A chosen preference relation

Output:

$x=\{x_1, x_2, \ldots, x_N\}$ = last generation of solution associated to the weight vectors.

1. $(x,z,FV,B(i)) \leftarrow$ Initialization(λ,N,m,T)
2. **do**
3. **for** i = 1 to N do
4. $\{y_1, y_2\} \leftarrow$ Reproduction(x,B(i),T)
5. UpdateZ(z, $\{y_1, y_2\}$) // z: for each j =1, …, m, if $z_j < f_j(y')$ then set $z_j = f_j(y')$, y' $\in \{y_1, y_2\}$
6. **UpdateNeighborhood(x,$B(i)$, λ,FV,$\{y_1, y_2\}$, R)** // for each $j \in B(i)$ and $y \in \{y_1, y_2\}$, // if $g^{te}(y \mid \varpi_i, z) \leq g^{te}(x_j \mid \varpi_i, z)$ and $x \, R_k^{DM} \, y$ set $x_j = y$ and $FV_j = F(y)$
7. **end for**
8. **while** Stopping Criterion unsatisfied //when numEvaluations < maxEvaluations

Imagen 7. Seudocódigo algoritmo MOEA/D/O.

Esta variante del algoritmo MOEA/D, combina intervalos y modelos outranking para representar la variabilidad presente en las preferencias del DM y así guiar el proceso de búsqueda (Fernandez et al., 2021).

Esta variante modifica la propuesta Tchebycheff para optimización escalar, con el fin de integrar relaciones outranking durante el proceso evolutivo. También utiliza intervalos en la definición del sistema de evaluación del DM para manejar imprecisión.

En la imagen 7 se puede observar que los nuevos parámetros que recibe el algoritmo son el sistema de evaluación del DM y la relación outranking. También en la línea 6 se observa que en el proceso de actualización de la solución se debe considerar la condición Tchebycheff y la relación de preferencia especificada.

Método ITOPSIS

TOPSIS (Technique of Order Preference Similarity to the Ideal Solution) es un método para el análisis de decisiones multicriterio, desarrollado por Yoon en 1981 (Rodríguez, 2020).

Está basado en la idea de que la solución deseable debería de tener la distancia geométrica más corta a la solución ideal y la distancia geométrica más prolongada a la solución anti-ideal.

La alternativa ideal es aquella que cuenta con una distancia geométrica más cercana al frente de Pareto, mientras que la alternativa anti-ideal tiene una distancia muy lejana al frente de Pareto.

ITOPSIS

Al igual que en el TOPSIS normal los pesos de los criterios se presentan en una matriz de decisión de intervalos. Así que como primer paso es la identificación de dicha matriz.

$$\otimes D = \begin{bmatrix} \otimes x_{11} & \cdots & \otimes x_{1m} \\ \vdots & \ddots & \vdots \\ \otimes x_{n1} & \cdots & \otimes x_{nm} \end{bmatrix}; i = 1, \dots, n; j = 1 \dots, m$$

Donde:

$\otimes x_{ij}$ = Expresa las evaluaciones de intervalos de la alternativa i(cartera) con respecto a j (Objetivos).

n = número de carteras.

m = número de objetivos.

El siguiente paso consiste en la construcción de la matriz de decisión normalizada. Para ello es necesario la utilización de la formula siguiente.

Una vez obtenida la matriz de decisión normalizada se procede a determinar la alternativa ideal A^+ y la anti-ideal A^- de cada objetivo.

Para la obtención de la alternativa A^+ es necesario la aplicación de la fórmula siguiente:

$$A^+ = \{(max_i \bar{r}_{ij} | j \in J), | i \in n\}$$

Y para el cálculo de la alternativa A^- se emplea la siguiente ecuación:

$$A^- = \{(min_i \underline{r}_{ij} | j \in J), | i \in n\}$$

Al realizar el cálculo de las dos alternativas se tiene como resultado dos vectores, una para cada alternativa. Obteniendo las ecuaciones siguientes:

$$A^+ = [r_1^+, r_2^+, \dots, r_m^+]$$
$$A^- = [r_1^-, r_2^-, \dots, r_m^-]$$

Donde:

$$\otimes r_{ij} = \frac{\otimes x_{ij}}{max_i(\bar{x}_{ij})|} = \left(\frac{\underline{x}_{ij}}{max_i(\bar{x}_{ij})}; \frac{\bar{x}_{ij}}{max_i(\bar{x}_{ij})}\right)$$

n = número de carteras.

m = número de objetivos.

Como siguiente procedimiento es el calcular la distancia de separación a la alternativa ideal y anti-ideal, Requiriendo la utilización de la distancia Euclediana.

La separación de cada alternativa i (Cartera) a la alternativa ideal A^+, es obtenida a través de la siguiente ecuación:

$$d_i^+ = \sqrt{\frac{1}{2}\sum_{j=1}^{m} w_j \left[|r_j^+ - \underline{r}_{ij}|^2 + |r_j^+ - \bar{r}_{ij}|^2\right]}$$

Donde:

r = número de carteras.

m = número de objetivos.

w_j= peso del beneficio j.

La separación de cada alternativa i(Cartera) a la alternativa anti-ideal A^+, es obtenida a través de la siguiente ecuación:

Como último proceso del algoritmo es el cálculo de la cercanía relativa a la solución ideal.
Haciendo uso de las distancias y de la siguiente ecuación:

$$C_i^+ = \frac{d_i^-}{d_i^+ + d_i^-}$$

Donde:

$0 \leq C_i^+ \leq 1$. Mientras más cercano a 1 es el valor, mejor será la evaluación de la alternativa.

Análisis de Desagregación de Preferencias

La desagregación de preferencias consiste en el análisis de las preferencias globales del tomador de decisiones (DM) para deducir la importancia relativa del criterio de evaluación y así desarrollar el modelo de preferencia correspondiente a las preferencias globales (Zopounidis & Doumpos, 1999).

El PDA es en sí un problema a resolver, el cual consiste en estimar una función que sea consistente con las preferencias subjetivas conocidas por el DM (Doumpos & Doumpos, 2002).

$$d_i^- = \sqrt{\frac{1}{2}\sum_{j=1}^{m} w_j \left[\left| r_j^- - \underline{r}_{ij} \right|^2 + \left| r_j^- - \overline{r}_{ij} \right|^2 \right]}$$

Para resolver el PDA se necesita un conjunto de soluciones de referencia (RS) previamente evaluadas por el DM como soluciones malas (C1) o soluciones buenas (C2).

Posteriormente MOEA/D Debe resolver el problema PDA(RS) para estimar un nuevo conjunto de parámetros preferencial (pesos, vetos, credibliity, mayoritiy, dominance) del Modelo de Outranking $nC_{inf} = \{w, v, \alpha, \lambda, \gamma\}$.

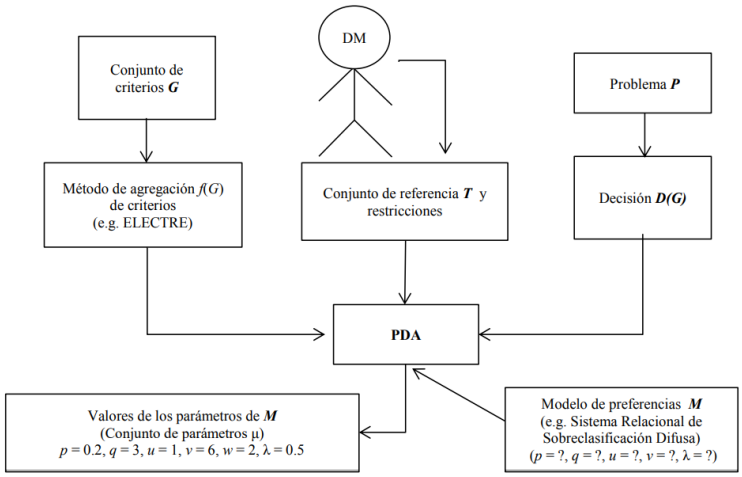

Imagen 8. Elementos involucrados en la resolución del PDA.

Herramienta VisTHAA

Es una herramienta para el análisis de algoritmos en los que se diagnostican las instancias, los algoritmos de solución y su desempeño a través de la calidad de los resultados (Ponce, 2021).

En la tabla 6 se presentan los módulos que conforman a la herramienta VisTHAA:

Tabla 6. Módulos VisTHAA

Módulo	Descripción
Entrada de datos y preprocesamiento	Tiene como función que los investigadores utilicen sus propios formatos para la lectura y procesamiento de archivos.
Caracterización de instancias	Una vez finalizado el proceso de carga de información, el investigador puede aplicar un proceso de caracterización al conjunto de instancias del problema a analizar.
Visualización de las instancias y el comportamiento del algoritmo	El módulo de visualización tiene como objetivo el poder visualizar de manera gráfica las instancias cargadas en la herramienta, graficas de frecuencia de las instancias, graficas del paisaje de aptitud

	del comportamiento del algoritmo y graficas estadísticas.
Rediseño causal	Permite la visualización de los resultados obtenidos en cada una de las fases anteriores.

Capítulo 4 Experimentación y análisis de resultados

Condiciones experimentales

Para la realización de la experimentación se ha utilizado el siguiente equipo:

- Procesador AMD Ryzen 5 4600H with Radeon Graphics 3.00 GHz
- RAM 16.0 GB (15.9 GB usable)
- Tipo de Sistema 64-bit operating system, x64-based processor
- NVIDIA GeForce GTX 1650 Ti

Diseño experimental

La arquitectura propuesta en este proyecto utiliza como interfaz de interacción un chat de Telegram. El propósito del chat es reportar las soluciones que la estrategia de interacción en conjunto con el algoritmo de solución y TOPSIS permiten construir, mientras que la tarea del decisor es retroalimenta al algoritmo con sus intereses.

El problema a resolver en la arquitectura propuesta se ha nombrado **remodelación de áreas de una casa**, el cual es una adaptación de un problema estándar DTLZ. Este problema consiste en presentarle al decisor un conjunto de habitaciones que se quieren remodelar y los costos necesarios para llevar a cabo dicha remodelación (imagen 2), donde los valores asociados a las áreas representan el cambio que sufrirán.

La tarea del decisor es seleccionar si la solución le ha gustado o no en base a cualquier criterio que él decida. El decisor puede centrarse en algún costo(s) que quiera minimizar y/o en alguna habitación(es) que más le interese remodelar.

Para poder probar la arquitectura de interacción propuesta se han diseñado diferentes experimentos.

En el experimento 1 se muestran al decisor, por única vez durante todo el proceso interactivo, diez configuraciones aleatorias en las que se puede llevar a cabo la remodelación, para que a parir de una inspección visual de las mismas el decisor

sea capaz de ofrecer al algoritmo un poco de retroalimentación seleccionando aquella que sea mayormente de su interés, esta solución seleccionada eventualmente se transformará en la mejor solución compromiso.

La solución compromiso es tomada en cuenta a partir de la arquitectura para poder traducirlos en parámetros de decisión del modelo de preferencias, los cuales son utilizados por el algoritmo para encontrar una solución mejor a las propuestas inicialmente.

En la experimentación 2 se guardaron las diez mejores soluciones que se presentaron al DM en cada iteración, con el fin de comprobar si alguna de ellas permanecía en la población en la siguiente iteración.

Para realizar esta primera parte de la experimentación se elaboró un script en Python, en el cual se unieron las soluciones de la población inicial con las soluciones de la siguiente generación para posteriormente hacer una intersección.

Experimentación 1: validación de la calidad de la interactividad

El propósito de esta experimentación fue comparar la solución compromiso (aquella que el tomador de decisiones eligió al comienzo del proceso interactivo) con la solución mostrada al final del proceso interactivo.

Para esta experimentación se resolvió el problema DTLZ1 con 3 objetivos – 7 variables de decisión y 5 objetivos – 9 variables de decisión.

Tabla 8. Distancia euclidiana mínima y promedio de la configuración 1.

	3 Obj 7 Var			
	MIN EUCLIDEAN			
		MOEA/D	MOEA/D/O R1	MOEA/D/O R2
	1	203.5506632	45.87775932	125.5619007
	2	188.7159283	40.62321504	111.442111
	3	202.9390875	35.26545193	117.7805145
	4	197.1189489	47.0603571	119.9640545
	5	196.284303	38.08981622	106.0187064
	6	195.7589589	43.80642904	127.4739362
	7	200.2480185	44.23487425	123.4759912
	8	188.7108804	27.78433085	126.216619
	9	196.2266073	46.63841871	123.8064215
	10	198.0750358	49.00437112	123.02073
	11	187.3661923	24.31830586	124.6974739
	12	199.7921797	33.55681749	124.1268706
	13	204.8695566	23.51029774	135.9691292
	14	193.4703967	41.06740313	120.3893521
	15	189.0636136	34.67347113	115.7936704
	16	198.9890466	50.2232608	116.9493758
	17	203.0838388	33.97484363	114.3254127
	18	195.4942702	40.71051953	114.9070059
	19	196.6531464	28.7546518	128.5318249
	20	193.5960033	36.7828982	129.2922828
AVG		196.5003338	38.29787464	121.4871692
MIN		187.3661923	23.51029774	106.0187064

	3 Obj 7 Var			
	AVG EUCLIDEAN			
		MOEA/D	MOEA/D/O R1	MOEA/D/O R2
	1	273.8680441	169.195676	249.698464
	2	259.8032001	169.8191054	242.6750673
	3	272.9277942	163.1853732	239.2535314
	4	261.4788368	174.4596511	249.8764708
	5	265.3483856	167.4540516	239.6882566
	6	263.8046904	173.9413612	254.3750272
	7	266.0058373	168.5259709	243.9374666
	8	259.9999074	166.1543868	251.8537543
	9	265.7266181	169.2521305	248.7080833
	10	267.6216262	170.2144635	245.6512566
	11	254.0493876	169.8599524	247.8538121
	12	266.8533285	171.4729196	248.0988104
	13	271.2165286	175.9700173	250.8076351
	14	259.2469393	177.4789559	245.2159064
	15	258.5279968	170.6731686	244.0964461
	16	264.8272327	170.0580898	245.4939718
	17	266.6867222	167.4715584	238.6592447
	18	266.0060342	172.2808903	242.3001788
	19	261.9662373	175.1905001	246.9965302
	20	265.8272446	171.6909301	247.9335024
AVG		264.5896296	170.7174576	246.1586708
MIN		254.0493876	163.1853732	238.6592447

Tabla 9. Distancia euclidiana mínima y promedio de la configuración 2.

	5 Obj 9 Var			
	MIN EUCLIDEAN			
		MOEA/D	MOEA/D/O R1	MOEA/D/O R2
	1	194.9703331	101.2452385	127.795827
	2	193.1086495	91.68261517	117.3614562
	3	192.7014421	106.5816101	122.916031
	4	194.0517503	83.86936524	112.9692051
	5	187.0142158	105.9641261	113.0504144
	6	192.3442053	105.790547	109.6943931
	7	185.8656961	90.98789933	109.0712749
	8	192.5280164	105.2105376	119.8350544
	9	193.0106528	89.79148629	122.574913
	10	191.6270247	94.7042092	124.1461754
	11	189.2904226	102.8639109	114.4035598
	12	194.9322021	108.2554245	114.8414059
	13	188.7938956	108.7996367	110.5435629
	14	187.1138277	96.09558936	116.2096141
	15	186.895345	89.77989989	111.3660717
	16	190.2028901	100.245611	113.2822321
	17	202.5673197	106.6413131	109.2656089
	18	195.1777175	85.44526838	112.2181915
	19	191.6067404	103.9232253	117.6504628
	20	187.9927057	85.24939394	108.0507691
AVG		191.5897526	98.15634538	115.3623112
MIN		185.8656961	83.86936524	108.0507691

	5 Obj 9 Var			
	AVG EUCLIDEAN			
		MOEA/D	MOEA/D/O R1	MOEA/D/O R2
	1	224.3025676	191.6218755	216.3472848
	2	225.9995925	194.5639478	210.263876
	3	221.936973	204.3384453	215.1465244
	4	221.6447437	197.1880834	209.420362
	5	215.3442559	192.4620446	212.4767554
	6	221.1342281	195.3421422	209.3617599
	7	214.2829628	197.5701683	208.1962899
	8	219.9877695	199.4164194	212.5672424
	9	223.0044658	194.0986573	213.5772736
	10	226.1315352	200.6926832	212.8539001
	11	218.1618561	197.4532115	212.4340017
	12	223.4040219	196.4440914	209.7953376
	13	220.6712463	197.7878833	210.6670798
	14	216.8434666	197.1286691	212.1723961
	15	216.5703818	191.7043655	210.6167024
	16	219.1234569	196.6991535	206.4516578
	17	230.8635228	200.9476587	209.8838999
	18	224.21624	195.7271799	208.7040425
	19	225.068892	197.021031	213.8793142
	20	218.9843174	193.7101159	206.94406
AVG		221.3838248	196.5958913	211.087988
MIN		214.2829628	191.6218755	206.4516578

Para comparar las soluciones, se calculó la distancia euclidiana entre la solución compromiso y la solución final. Como se puede ver en las tablas 8 y 9, se encontró que ambas soluciones estaban separadas.

También se encontró alguna variación en las soluciones durante cada iteración del proceso interactivo, por lo que es conveniente registrar los parámetros proporcionados por la PDA para determinar si los valores actualizados tienen una diferencia significativa. A partir de esto, se puede definir una estrategia para obtener mejores resultados esperados.

Experimentación 2: validación de la aportación de la interactividad

En esta experimentación se realizaron 15 pruebas y se encontró que ninguna de las soluciones de la población inicial permanecía en la última generación, lo que podría significar que las soluciones están siendo mejoradas y, por lo tanto, reemplazadas. Para comprobar que las soluciones han sido mejoradas a través de las iteraciones se ha decidido ordenar las cinco mejores soluciones de la población inicial junto con las cinco mejores soluciones de la población final utilizando el método TOPSIS. Este ordenamiento permite identificar de manera rápida y sencilla las mejores soluciones de ambos conjuntos.

Tabla 10. Resultados de la prueba 1.

N° sol.		Objetivos				
Mejor	1	8.85	1.1	29.1	28.8	159
	5	5.34	0.00997	51.2	122	131
	3	0.544	21.6	56.7	33.8	156
	2	2.93E+00	10.2	15.7	81.3	189
	6	1.14E-04	0.00021	218	166	127
	9	0.0618	0.109	180	192	135
	10	7.94E-04	0.00146	181	203	127
	4	12.8	53	8.39	22.2	79.1
	7	15.9	53.5	212	53.6	95.3
Peor	8	92.1	3.55	230	35.4	36.7

Tabla 11. Resultados de la prueba 2.

N° Sol.		Objetivos				
Mejor	6	0.0637	252	3.86	252	0.0596
	8	0.616	224	6.16	270	7.86
	7	12.4	215	1.79	272	0.51
	1	9.90E+00	12.3	369	27.2	20.1
	10	6.60E+00	217	6.12	268	10.5
	2	15.4	7.21	405	40	16.2
	9	1.20E+01	215	0.916	264	9.15
	3	16.7	5.49	399	50.6	19.3
	5	47.8	49.6	366	0.0419	0.0294
Peor	4	19.9	46.8	377	0.492	35.1

Tabla 12. Resultados de la prueba 3.

N° Sol.		Objetivos				
Mejor	8	520	2.37	0.0367	0.00746	0.021
	9	520	2.37	0.0367	0.00746	0.021
	10	520	2.37	0.0367	0.00746	0.021
	6	5.24E+02	2.39	0.0168	0.00162	0.0158
	7	5.24E+02	2.39	0.0168	0.00162	0.0158
	2	0.143	0.00879	0.874	99.5	373
	1	2.05E-02	0.000208	17.2	48.9	272
	4	0.0272	0.00000104	0.0838	195	269
	3	0.832	4.54	27.7	7.18	354
Peor	5	2.14	1.43	23.9	191	248

Tabla 13. Resultados de la prueba 4.

N° Sol.		Objetivos				
Mejor	2	4.44	7.58	6.32	134	128
	3	5.6	7.7	0.381	183	172
	1	0.00329	3.26	0.337	99.4	266
	5	5.77E+00	10.2	8.41	179	170
	9	3.32E-01	0.802	156	190	139
	6	1.26	3.05	183	226	69.2
	4	2.81E+01	2.7	0.159	10.1	338
	7	27.3	25.5	172	278	1.53
	8	32	29.8	163	278	1.53
Peor	10	85.4	5.79	296	69	62.3

Tabla 14. Resultados de la prueba 5.

N° Sol.		Objetivos				
Mejor	8	130	51.6	108	123	0.366
	5	76.8	60.6	102	282	0.504
	6	174	180	0.0178	174	0.00403
	10	6.81E+01	226	76.4	131	1.35
	7	2.84E+02	115	61.2	46.4	0.354
	2	52	4.66	99.9	230	111
	9	2.18E+02	221	5.02	18.1	40.3
	3	43.1	3.36	94.6	199	154
	1	18.4	42.5	262	63.3	94.3
Peor	4	26.2	65.2	181	122	126

Tabla 15. Resultados de la prueba 6.

N° Sol.		Objetivos				
Mejor	10	520	0.269	3.03	0.00373	3.98
	6	525	0.303	0.112	0.187	2.05
	7	525	0.303	0.112	0.187	2.05
	8	5.25E+02	0.303	0.112	0.187	2.05
	9	5.28E+02	2.32	0.319	0.00751	0.462
	1	1.09	9.77	69.1	99.2	253
	2	1.09E+00	9.77	69.1	99.2	253
	3	1.09	9.77	69.1	99.2	253
	4	58.6	26.5	218	4.55	211
Peor	5	24.7	122	148	0.602	228

Tabla 16. Resultados de la prueba 7.

N° Sol.		Objetivos				
Mejor	6	521	0	0	0	0.00222
	7	522	0	0	0	0.0026
	8	521	0	0	0	0.00577
	9	5.21E+02	0	0	0	0.00577
	10	5.21E+02	0	0	0	0.00577
	3	28.7	276	0.00022	174	0.00246
	5	4.25E+01	248	0.000537	225	0.00264
	1	27.4	263	0.000537	225	0.0026
	2	27.4	263	0.000537	225	0.0026
Peor	4	37.9	222	0.0339	253	0.00143

Tabla 17. Resultados de la prueba 8.

N° Sol.		Objetivos				
Mejor	1	277	169	1.27	0.747	0.188
	3	300	169	3.99	0.633	0.59
	5	299	161	42.7	18.1	0.0014
	6	1.05E+02	0.0275	266	53.1	39.9
	10	1.32E+02	25.5	297	37	21.3
	8	166	0.0476	253	37.2	58.4
	2	1.78E+02	208	0.00621	0.506	95.9
	7	42.3	0.0873	275	127	24.6
	4	86.8	201	3.17	0.739	189
Peor	9	124	1.56	104	205	59.2

Población inicial Población final

Tabla 18. Resultados de la prueba 9.

N° Sol.		Objetivos				
Mejor	6	50.5	30	320	5.3	72.4
	9	126	27.3	169	2.92	83.3
	2	12.7	14.1	109	68.4	165
	4	4.64E+01	93.8	96	2.08	213
	1	4.74E+01	97.6	30.7	51.1	132
	7	72	46	311	46.3	34.8
	8	7.20E+01	46	311	46.3	34.8
	3	41.7	58.5	7.96	69.7	207
	10	177	51.4	201	10.2	27
Peor	5	102	159	0.713	52.8	68.3

Tabla 19. Resultados de la prueba 10.

N° Sol.		Objetivos				
Mejor	9	50.7	52.1	26.3	294	55.9
	4	149	188	0	184	0.0958
	2	157	224	0	148	0.0524
	1	1.73E+02	223	0	91.7	0.421
	3	1.72E+02	208	0	147	0.0833
	5	42.1	229	5.8	158	75.9
	8	6.74E+01	0.0000123	51.8	295	95.3
	7	118	0.000972	101	185	73.4
	6	4	2.08	162	208	105
Peor	10	55.2	27.1	236	77	74.5

Tabla 20. Resultados de la prueba 11.

N° Sol.		Objetivos				
Mejor	1	168	144	0.927	27.8	8.26
	7	243	112	16.4	135	2.24
	6	193	94.2	28.8	180	0.07
	8	2.31E+02	107	0.182	169	2.24
	9	2.32E+02	107	0.0162	170	2.25
	2	8.03	187	143	37.8	18.9
	10	1.36E+02	66.3	114	180	0.07
	3	0.00867	232	205	1.4	0.0144
	4	10.6	247	158	24.5	25.2
Peor	5	123	162	79.5	2.06	87.5

Tabla 21. Resultados de la prueba 12.

N° Sol.		Objetivos				
Mejor	2	180	99.6	0.511	157	4.02E-07
	5	181	1.01E+02	0.516	180	2.24E-07
	3	164	115	0.511	157	4.02E-07
	4	1.52E+02	128	0.372	157	0.00000358
	6	2.59E+02	0.00219	101	31.9	31.3
	1	82.4	129	3.56	199	0.0000032
	9	2.07E+02	47.9	11.2	200	36.7
	7	302	0.00255	140	39.2	38.4
	10	228	1.21	193	46.3	41.1
Peor	8	164	8.24	98.5	128	124

Tabla 22. Resultados de la prueba 13.

N° Sol.		Objetivos				
Mejor	7	218	0.00157	12.8	267	9.72E-08
	6	271	0.532	6.26	232	0.00441
	9	257	0.506	19.9	232	0.00105
	8	2.53E+02	2.26	19	244	0.0308
	1	2.04E+01	5.23	14.1	282	122
	10	282	2.09	24.3	210	0.00264
	4	6.12E+01	22.9	10	40.4	241
	2	0.178	71.4	18	251	40.1
	3	67	21.1	115	78.1	80
Peor	5	22.3	29.7	76.9	166	162

Tabla 23. Resultados de la prueba 14.

N° Sol.		Objetivos				
Mejor	1	6.93	0.052	3.28	91.7	264
	7	72.3	0.0067	0.421	147	137
	2	5.39	4.13	0.00669	168	277
	4	6.39E+00	5.61	0.031	260	205
	8	1.13E+02	0.0361	0.256	90.5	179
	9	113	0.0361	0.256	90.5	179
	10	1.74E+02	0.071	0.229	0.0674	167
	3	4.47	0.461	23.1	349	105
	5	5.11	0.991	28.4	351	107
Peor	6	51	41.5	14	105	109

Tabla 24. Resultados de la prueba 15.

N° Sol.		Objetivos				
Mejor	9	307	2.72	43.9	71.9	78.2
	8	313	7.13E+00	45.4	61.5	7.85E+01
	6	303	7.71	4.06	31.6	152
	7	2.60E+02	16.6	9.05	33	191
	10	1.79E+02	30.8	6.11	14.7	249
	2	89.2	4.51	47.5	291	12.7
	1	8.91E+01	16	62.9	268	9.03
	3	138	163	35.3	70.9	15.2
	5	95.7	85	115	122	36.2
Peor	4	146	120	154	0.00295	3.54

Población inicial Población final

En las tablas anteriores las soluciones están ordenadas de mejor a peor de acuerdo con el método TOPSIS, las soluciones en color verde (solución 1 a 5) corresponden

a la población inicial, mientras que las soluciones en color naranja (solución 6 a 10) corresponden a la población final.

Se puede observar que en 8 de las 15 pruebas las mejores soluciones se encontraron en la población final (prueba 2, 3, 5, 6, 7, 11, 13 y 15), mientras que en el resto de las pruebas se puede observar que algunas veces la población final no contiene a las mejores soluciones. Este comportamiento puede deberse a que la búsqueda es afectada por las evaluaciones que realiza el tomador de decisiones, el cual decide si una solución le gusta en base a su conocimiento subjetivo. Quizá agregando un conjunto de soluciones elite permita obtener mejores soluciones en la población final.

Capítulo 5 Conclusiones y trabajos futuros

En esta investigación se ha diseñado y validado un framework interactivo bajo condiciones de incertidumbre para optimización multiobjetivo. Para validarlo se construyó una herramienta de software que está compuesta por un conjunto de módulos que se comunican e interactúan entre sí para resolver el problema de encontrar la mejor solución que representa los objetivos de un tomador de decisiones.

Gracias a la arquitectura de integración de los diferentes módulos de optimización con el bot desarrollado en Telegram, ha sido posible incorporar eficientemente las preferencias de un tomador de decisiones en el proceso de resolución de un problema de optimización DTLZ estándar.

Asimismo, con los resultados de la experimentación 1, se observa que existe una diferencia entre la solución de compromiso y la solución obtenida al final del proceso interactivo.

Las principales contribuciones de este proyecto han sido:

- Arquitectura de integración entre un chatbot y una herramienta de optimización.
- MOEA/D/O Interactivo.
- Análisis Comparativo del efecto de la interacción contra estrategias de manejo de preferencias a posteriori.
- Integración del método TOPSIS al framework de optimización MS-DOSS.
- Definición de problema benchmark de remodelación de áreas para algoritmo evolutivo con incorporación de preferencias interactivo.

Como trabajos futuros se pueden considerar los siguientes puntos:

- Incorporación de otros problemas de optimización como el problema de cartera de proyectos.

- Incorporación de un nuevo módulo para la generación de graficas ya sea en VisTHAA u otra plataforma.
- Definir una estrategia para mejorar la búsqueda de la mejor solución, como puede ser la incorporación de una población elite para guiar la búsqueda.
- Contemplar múltiples tomadores de decisiones.

Referencias

BEN AKKA, Y., RAHALI, A., ALAMI, H., & ABDELLAOUI , E. (2019). Control And Command Of Several Greenhouses Via Telegram Messenger. *Association for Computing Machinery*, 1-9.

Bernardin, L., Char, B., & Kaltofen, E. (1999). Symbolic computation in Java: an appraisement. *Symposium on Symbolic and Algebraic Computation*, 237-244.

Bull, J., Smith, L., Westhead, M., Henty, D., & Davey, R. (1999). A methodology for benchmarking Java Grande applications. *ACM*, 81-88.

Castellanos, A. (2022). Hybridisation of Swarm Intelligence Algorithms with Multi-Criteria Ordinal Classification: A Strategy to Address Many-Objective Optimisation. *Mathematics*, 1-23.

Cheng, R., Olhofer, M., & Jin, Y. (2015). Reference Vector Based a posteriori Preference Articulation for Evolutionary Multiobjective Optimization. *IEEE*, 1-8.

Del Angel Franco, E. (2017). *Estudio de la matemática gris y sus aplicaciones.* Cd. Madero.

Doumpos, M., & Zopounidis, C. (2004). Developing sorting models using preference disaggregation analysis: An experimental investigation. *Elsevier*, 1-2.

Fernández, E., Rangel Valdez, N., Cruz Reyes, L., Gomez Santillan, C. G., & Coello Coello, C. A. (2021). PREFERENCE INCORPORATION INTO MOEA/D USING AN OUTRANKING APPROACH WITH IMPRECISE MODEL PARAMETERS. *Swarm and Evolutionary Computation*, 1-24.

Gherardi, L., Brugali, D., & Comotti, D. (2012). A Java vs. C++ Performance Evaluation: A 3D Modeling Benchmark. *Springer-Verlag*, 1-12.

Greyling, C. (22 de Junio de 2020). *medium*. Obtenido de General Chatbot Architecture, Design & Development Overview: https://cobusgreyling.medium.com/general-chatbot-architecture-design-development-overview-58e145398608

Idhom, M., Alit, R., Endah Wahanani, H., & Fauzi, A. (2018). Implementation System Telegram Bot for Monitoring Linux Server. *Atlantis Highlights in Engineering*, 1-5.

Khan, R. (2017). Standardized Architecture for Conversational Agents a.k.a. ChatBots. *International Journal of Computer Trends and Technology* , 114-121.

Khan, R., & Das, A. (2018). *Build Better Chatbots.* Karnataka: Apress.

Li, K. (2019). Progressive Preference Learning: Proof-of-Principle Results in MOEA/D. *Springer*, 1-13.

Li, Y., Liu, H., Xie, K., & Yu, X. (2015). A Method for Distributing Reference Points Uniformly along the Pareto Front of DTLZ Test Functions in Many-Objective Evolutionary Optimization . *International Conference on Information Science and Technology*, 1-3.

Ma, X., Liu, F., Qi, Y., Li, L., Jiao, L., Deng2, X., . . . Wu, J. (2015). MOEA/D with biased weight adjustment inspired by user preference and its application on multi-objective reservoir flood control problem. *Springer*, 1-25.

Mañas, J. (2017). Análisis de Algoritmos – Complejidad . *ADSW*, 1-31.

Muente, G. (08 de Enero de 2020). *rockcontent*. Obtenido de Guía completa del Framework: qué es, cuáles tipos existen y por qué es importante en Internet: https://rockcontent.com/es/blog/framework/

Muhammad Daffa, A., & Roestam, R. (2019). ARDUINO BABY MONITORING WITH TELEGRAM BOT AND WIFI CONNECTION. *IT FOR SOCIETY*, 1-4.

Ofoeda, J., Boateng, R., & Effah, J. (2019). Application Programming Interface (API) Research: A Review of the Past to Inform the Future. *International Journal of Enterprise Information Systems*, 1-20.

Osorio Goméz, J. C., & Orejuela Cabrera, J. P. (2008). EL PROCESO DE ANÁLISIS JERÁRQUICO (AHP) Y LA TOMA DE DECISIONES MULTICRITERIO. EJEMPLO DE APLICACIÓN. *Scientia Et Technica*, 1-7.

Ponce Nájera, J. G. (2021). *Análisis de Desempeño Usando VistHAA Aplicando un Algoritmo Metaheuristico de un Framework de Optimización.* Ciudad Madero.

Ramirez Bracho, F. A. (2007). Análisis de Algoritmos. *ACADEMIA*, 1-8.

Ramos, A. S. (2010). *Modelos matemáticos de optimización.*

Rosid , M. (2018). Integration Telegram Bot on E-Complaint Applications in College. *IOP Conference Series: Materials Science and Engineering*, 1-7.

Roulo, M. (1998). Accelerate your Java apps. *Java World*.

Sánchez Solís, J. P. (2017). *INCORPORACIÓN DE PREFERENCIAS EN METAHEURÍSTICAS EVOLUTIVAS A TRAVÉS DE CLASIFICACIÓN MULTICRITERIO.*

Saxena, N. (29 de Enero de 2020). *medium*. Obtenido de Chatbot Architecture Tutorial: https://medium.com/@nihitextra/chatbot-tutorial-choosing-the-right-chatbot-architecture-5539c8489def

Scalise, E., & Carmona, R. (2015). Análisis de algoritmos y complejidad. *ResearchGate*, 1-22.

Spw, S., Wentworth, S., & Langan, D. (2001). Performance evaluation: Java vs. c++. *39th Annual ACM Southeast Regional Conference*.

Telegram. (s.f.). *Telegram*. Obtenido de Bots: An introduction for developers: https://core.telegram.org/bots

Tsidylo, I., Samborskiy, S., Mazur, S., & Zamoroz, M. (2020). Designing a chatbot for learning a subject in a Telegram messenger . 1-12.

Vitoriano, B., Linares, P., & Ramos, A. (2010). MODELOS MATEMÁTICOS DE OPTIMIZACIÓN. *ACADEMIA*, 1-56.

Vivanco, R. A., & Pizzi, N. J. (2004). Scientific computing with Java and C++: a case study using functional magnetic resonance neuroimages. *SOFTWARE—PRACTICE AND EXPERIENCE*, 1-18.

Zhang, Q. (2007). MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE*, 1-20.