Herramienta gráfica para la ejecución de procesos de optimización multiobjetivo mediante algoritmos evolutivos

Angel Zait Hernández López
Escuela Superior de Computo,
Instituto Politécnico Nacional
Ciudad de México 07320, México
ahernandezl1308@alumno.ipn.mx

Luis Fernando Martínez Rodríguez
Escuela Superior de Computo,
Instituto Politécnico Nacional
Ciudad de México 07320, México
lmartinezr1306@alumno.ipn.mx

Eric Uriel Trejo Trejo
Escuela Superior de Computo,
Instituto Politécnico Nacional
Ciudad de México 07320, México
etrejot1300@alumno.ipn.mx

Víctor Adrián Sosa Hernández
Tecnologico de Monterrey, Escuela de Ingeniería y Ciencias, Estado de México 52926, México
vsosa@tec.mx, https://orcid.org/0000-0002-1099-8148

Miriam Pescador-Rojas
Escuela Superior de Computo,
Instituto Politécnico Nacional
Ciudad de México 07320, México
mpescadorr@ipn.mx,
https://orcid.org/0000-0001-7444-6740

***Resumen*— En este artículo se presenta el desarrollo de una herramienta de apoyo para la solución de problemas multiobjetivo mediante algoritmos evolutivos. La herramienta de programación visual denominada GRATMOO (Graphic Tool for Multi Objective Optimization) permite la generación de flujos de trabajo mediante bloques conectados entre sí, el resultado final es un prototipo que permite de forma interactiva y visual solucionar familias representativas de problemas benchmark. El sistema brinda opciones para configurar algoritmos evolutivos representativos tales como: NSGA-II, NSGA-III, SMS-EMOA y MOEA/D mediante la modificación de sus parámetros. Además, muestra un análisis de resultados en forma de gráficas y tablas, generando un reporte para evaluar el desempeño de cada tipo de algoritmo.**

Keywords—programación visual, optimización multiobjetivo, cómputo evolutivo.

I. INTRODUCCIÓN

La programación visual consiste en un entorno de desarrollo con interfaz gráfica de usuario que permite la creación de programas sin escribir líneas de código en un lenguaje de programación. Los ambientes de desarrollo visuales por lo general son herramientas de apoyo educativo. En un mundo globalizado como en el que vivimos ahora, el desarrollo de software es muy demandado y la habilidad de programación es esencial. La programación visual ha logrado motivar a personas que jamás han tenido un acercamiento con principios básicos de programación a crear programas de cómputo. Por otro lado, las herramientas de computación evolutiva son útiles en diversas áreas de conocimiento, tales como física, química e ingeniería para dar solución a problemas de optimización mono y multi-objetivo, sobre todo en este último donde se requiere la comparación de posibles soluciones y que el tomador de decisiones elija la que más convenga al contexto del problema.

La programación visual es un paradigma que cambia el concepto tradicional de la programación convirtiéndola en algo simple, intuitivo y atractivo de aprender. Esta consiste en un entorno de desarrollo con interfaz gráfica de usuario que permite la creación de programas sin escribir líneas de código de un lenguaje determinado como se realiza de manera convencional. La mayoría de estos entornos utilizan bloques predefinidos para ejecutarse posteriormente en algún lenguaje de programación. Por lo general, los ambientes de desarrollo visuales son herramientas de apoyo educativo, que inicialmente predominaron para introducir a los niños al ámbito de la programación. Actualmente ejemplos de estas herramientas son: Scratch que cuenta con cerca de 40 millones de usuarios activos y code.org con alrededor de 50 millones de niños que aprenden a programar [1].

El desarrollo de software es muy demandado y la habilidad de programación es esencial. Sin embargo, cuando las personas observan los lenguajes de programación no creen que sean capaces de lograr con ellos lo que desean, consideran que programar requiere un alto nivel de preparación. Por otro lado, la programación visual ha logrado motivar a personas que jamás han tenido un acercamiento con principios básicos de programación a crear programas de cómputo. Con el auge de este paradigma han surgido sistemas de flujos de trabajo (*workflows systems*) que consisten en aplicar la programación basada en bloques para realizar tareas con mayor grado de complejidad y poder computacional, sin involucrar a un lenguaje de programación en específico. Para lograr esto, se utilizan bloques que funcionan como operadores o procesos predefinidos que pueden ser configurados. Estos procesos por sí solos pueden no realizar grandes tareas, pero al conectarlos en un flujo continuo la salida de uno es la entrada para el siguiente y se logra realizar una tarea de mayor complejidad [1, 2].

Por otro lado, el cómputo evolutivo es una rama de las ciencias de la computación inspirada en la teoría de la evolución natural para poder resolver problemas de optimización y búsqueda. Una clase de estos problemas de optimización requiere resolver de manera simultánea múltiples funciones objetivo en conflicto. En consecuencia, no existe una solución única, sino varias, que representan los mejores resultados posibles entre los objetivos. Por lo tanto,

rara vez se da el caso de que haya un solo punto que optimice simultáneamente todos los objetivos [3].
Los Algoritmos Evolutivos Multiobjetivo (AEMOs) son capaces de resolver este tipo de problemas de manera apropiada [4].

Existen frameworks con la implementación de los AEMOs, desarrollados en distintos lenguajes de programación, cabe mencionar, que ninguno de ellos es de un estilo visual, todo es a través de código.

El software desarrollado tiene como objetivo ser una herramienta de apoyo para la solución de problemas de optimización multiobjetivo mediante algoritmos evolutivos y programación visual por bloques, facilitando que cualquier persona interesada obtenga resultados sin ser expertos en computación evolutiva. La contribución de esta herramienta es facilitar el diseño de los prototipos funcionales para el desarrollo de un sistema. Mediante una implementación de programación web con interfaz interactiva y probar el sistema en la toma de decisiones para problemas de optimización multi-objetivo.

II. CONCEPTOS BÁSICOS

A. Programación visual

La programación visual es una herramienta para la construcción de programas mediante la manipulación directa de representaciones gráficas. En lugar de escribir código de manera textual, el desarrollador utiliza elementos gráficos para construir y estructurar la lógica del programa.

Las características principales de la programación visual incluyen:

- Interfaz gráfica: En lugar de líneas de código, la programación visual utiliza diagramas, bloques, líneas, y otros elementos gráficos para representar lógica y flujo del programa.
- Drag and drop: La mayoría de las herramientas de programación visual permiten al usuario "arrastrar y soltar" bloques de construcción para formar la estructura y lógica del programa.
- Abstracción: Las herramientas de programación visual a menudo abstraen detalles complejos, permitiendo a los desarrolladores concentrarse en la lógica general sin preocuparse por la sintaxis específica.
- Facilidad de uso: Por su naturaleza gráfica y el hecho de que suele abstraer los detalles de programación, estas herramientas son ideales para principiantes, educadores o para aquellos que desean prototipar rápidamente.

B. Problema de optimización multiobjetivo

Un problema de optimización multiobjetivo (POM) se refiere a una clase de problemas de optimización en los que se busca optimizar simultáneamente más de un objetivo. Estos problemas son comunes en situaciones de la vida real donde hay múltiples criterios que deben ser considerados y, a menudo, estos criterios pueden estar en conflicto entre sí.

Matemáticamente, este tipo de problemas se definen de la siguiente manera:

$$minimizar\ \vec{\boldsymbol{f}}(\vec{\boldsymbol{x}}) = (f_1(\vec{\boldsymbol{x}}), f_2(\vec{\boldsymbol{x}}), \dots, f_m(\vec{\boldsymbol{x}}))$$

sujeto a

$$g_i(x) \leq 0; i = 1, \dots, p$$
$$h_j(x) = 0; j = 1, \dots, q$$

Donde $\vec{\boldsymbol{x}} = \{x_1, x_2, \dots, x_n\}$, es el conjunto de variables de decisión del problema, n el número de funciones objetivo a minimizar, p el número de restricciones de desigualdad y q el número de restricciones de igualdad.
Un problema multiobjetivo puede o no considerar funciones de restricción [9]. El conjunto de soluciones que cumplen con las condiciones anteriores es conocido como el conjunto óptimo de Pareto, definido formalmente como sigue:

Un vector $\vec{\boldsymbol{u}} = \{u_1, \dots, u_n\}$ se dice domina a $\vec{\boldsymbol{v}} = \{v_1, \dots, v_n\}$ si y solo si $\vec{\boldsymbol{u}}$ es parcialmente menor que $\vec{\boldsymbol{v}}$, $\forall i\, \epsilon\{1, \dots, n\}$, $u_i \leq v_i \wedge \exists_i\, \epsilon\{1, \dots, n\} : u_i < v_i$. En la figura 2, se muestra un ejemplo donde se considera un problema multiobjetivo que requiere minimizar dos funciones objetivo f_1, f_2 decimos que la solución B domina a las soluciones B y C.

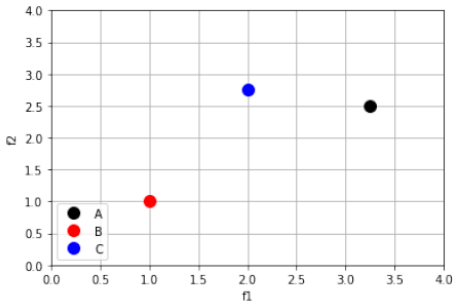

Figura 1 Dominancia de Pareto

C. Algoritmos evolutivos multiobjetivo

Los algoritmos evolutivos multiobjetivo (AEMO) son técnicas diseñadas para resolver problemas de optimización con múltiples objetivos. Estos algoritmos se basan en los principios de la evolución natural para explorar y explotar el espacio de búsqueda de posibles soluciones y encontrar el conjunto óptimo de Pareto.

El funcionamiento básico de un AEMO implica la generación de poblaciones de soluciones, donde cada solución se evalúa en función de múltiples objetivos. A partir de estos resultados, se seleccionan las mejores soluciones basadas en criterios específicos, como la dominancia de Pareto. En cada iteración, se aplican operadores genéticos como la mutación y la cruza para generar nuevas soluciones. Este proceso se repite a lo largo de varias generaciones hasta que se alcanza un determinado criterio de paro. Un AEMO incluyen los siguientes elementos:

- Selección basada en dominancia de Pareto para guiar la selección de soluciones prometedoras.

- Preservación de diversidad. Dado que el objetivo es encontrar un conjunto diverso de soluciones no dominadas que representen el frente de Pareto, los AEMOs incorporan

mecanismos para mantener la diversidad en la población y evitar la convergencia prematura a una región específica del frente. Los AEMOs son herramientas valiosas para resolver problemas complejos de optimización multiobjetivo, especialmente aquellos en los que los objetivos están en conflicto y se necesita encontrar un conjunto de soluciones de compromiso.

III. Trabajos relacionados

La programación visual ha logrado motivar a personas que jamás han tenido un acercamiento con principios básicos de programación a crear programas de cómputo. Con el auge de este paradigma han surgido sistemas deflujos de trabajo (*workows systems*) que consisten en aplicar la programación basada en bloques para realizar tareas con mayor grado de complejidad y poder computacional, sin involucrar a un lenguaje de programación en específico. Estos sistemas cuentan con una interfaz gráfica que proporciona al usuario la capacidad de crear de forma esquemática un flujo de trabajo que será ejecutado para realizar la tarea o tareas definidas.

Para lograr esto, se utilizan bloques que funcionan como operadores o procesos predefinidos que pueden ser configurados. Estos procesos por sí solos pueden no realizar grandes tareas, pero al conectarlos en un flujo continuo la salida de uno es la entrada para el siguiente y se logra realizar una tarea de mayor complejidad [10, 11]. En la tabla 1 se muestran algunas herramientas de software de programación visual basada en bloques [12].

Tabla 1 Principales herramientas de software que utilizan programación visual basada en bloques

Sistema / creador	Descripción
Scratch [13]. MIT, Fundación scratch	Creado por el Massachusetts Institute of Technology (MIT), es una plataforma 2D con recursos gráficos. Este proporciona las herramientas de creación de proyectos, compartir y comentarlos. El lenguaje de programación es similar a C.
App Inventor [14], MIT	Creado por el MIT con apoyo de Google, es una aplicación web para el desarrollo de aplicaciones móviles para los sistemas operativos Android.
Rapid Miner [15], University of Dortmund	Plataforma de software de ciencia de datos diseñada por la universidad de Dortmund en 2001. Esta herramienta proporciona un ambiente integrado para preparación de datos, aprendizaje máquina, aprendizaje profundo, extracción de textos y análisis predictivo todo esto en un ambiente gráfico de interfaz de usuario para diseñar y ejecutar flujos de trabajo analíticos (analytical workflows).

IV. Herramienta Propuesta

En esta sección se describen los componentes usados para la propuesta de herramienta gráfica para la ejecución de procesos de optimización multiobjetivo, denominada GRATMOO (*Graphic Tool for Multi Objective Optimization*), para su implementación se emplearon tecnologías web bajo el patrón de diseño MVVM (*Model View ViewModel*). MVVM se basa en un mecanismo general de enlace de datos que facilita el desarrollo de la capa de separación de vista desde el resto del patrón mediante la eliminación de todo el código subyacente de la capa de la vista. [5, 6]

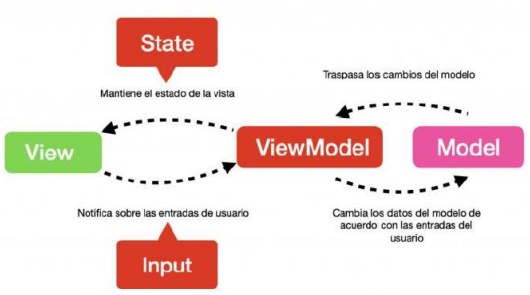

Figura 2. Patro de diseño Modelo Vista - Vista Modelo

De esta manera se puede analizar el sistema en tres partes: el modelo, compuesto por cada una de las clases que proporcionan la lógica resolución de los problemas con un lenguaje de programación; la vista, que es la interfaz gráfica de usuario por medio de tecnologías web; y por último el modelo vista como el servidor web que proporciona los estados o funcionalidades solicitadas desde la vista.

A. Modelo

El Modelo incluye la lógica de negocio y datos, la cual se ocupa de la recuperación y gestión de datos, así como asegurar que las reglas del negocio garanticen coherencia y validez de los datos para maximizar las oportunidades de reutilización [6].

Esta parte del sistema se desarrolló utilizando el lenguaje de programación Python, el cual es un lenguaje frecuentemente utilizado en aplicaciones escolares y proyectos científicos de alto rendimiento debido a que su fácil escritura, legibilidad permiten una rápida codificación y desarrollo de sistemas con un buen desempeño [7].

Entre la gama de frameworks disponibles en este lenguaje de programación se eligió Pymoo[8], debido a que se encuentra completamente escrito en Python, documentación más completa, el desempeño es aceptable y cuanta con familias de problemas benchmark y algoritmos de interés ya implementados.

B. Vista

La Vista hace referencia a la interfaz de usuario y la lógica de la interfaz de usuario. Las vistas definen la interfaz de usuario específica para una parte de la aplicación, son normalmente los controles que se han diseñado para funcionar bien cuando se une a un Modelo de Vista [6].

La vista del sistema esta desarrollada con tecnologías web, tales como la biblioteca React, que con el uso de HTML,

JavaScript y CSS permite la generación de una interfaz web de una forma integral y garantiza la ejecución de todos los recursos en los diferentes navegadores web que existen hoy en día.
La aplicación desarrollada puede verse como un sitio web responsivo, lo que permite que el sistema mantenga una vista compatible con dispositivos móviles como los smartphones o tabletas con una pantalla más pequeña que las computadoras.

La figura 3, muestra los elementos de la herramienta propuesta. Aquí interactúan diferentes módulos para el uso y configuración de algoritmos evolutivos multiobjetivo, así como la definición de escenarios de experimentación para la solución de problemas de optimización multiobjetivo y la visualización de sus resultados.

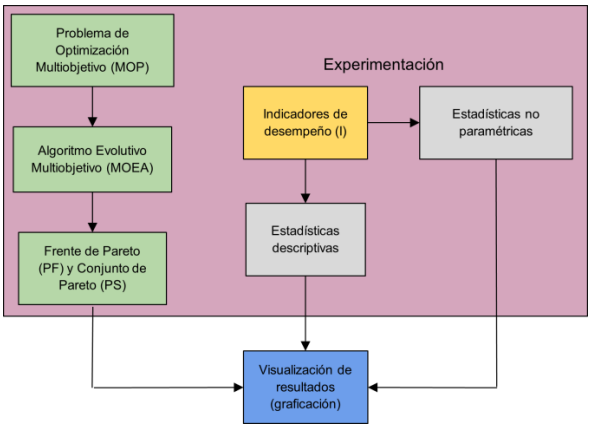

Figura 3 Arquitectura del sistema desarrollado

A continuación, se describen cada uno de los módulos implementados, así como sus alcances.

Módulo 1. Problemas de optimización multiobjetivo (POM)
Este módulo encapsula los objetos relacionados a las familias de problemas benchmark disponibles para resolver, entre estos grupos se encuentran los ZDT, DTLZ, WFG y CEC2009.
De estos problemas se permite variar los parámetros de identificador del problema, número de variables y objetivos de este. De estas cuatro familias algunas se encuentran implementadas por Pymoo, en la tabla 2 se presenta esta relación.

Tabla 2 Origen de implementación de problemas de optimización multiobjetivo

Familia POM	Origen de implementación
ZDT	Pymoo
DTLZ	Pymoo
WFG	Pymoo
CEC2009	Equipo de desarrollo

Módulo 2. Algoritmos de optimización multiobjetivo (AEMO)

Se enfoca en el encapsulamiento de los algoritmos evolutivos y proporciona una estructura escalable que permite el ajuste de parámetros típicos de cada uno de estos. Los algoritmos implementados son cuatro el NSGA-II, NSGA-III, MOEA/D y SMS-EMOA. De igual manera de estos algoritmos es posible modificar parámetros como el tamaño de la población, criterio de paro y número de evaluaciones o generaciones. El origen de la implementación de estos algoritmos se muestra en la tabla 3.

Tabla 3 Origen de implementación de algoritmos evolutivos multiobjetivo

MOEA	Origen de implementación
NSGA-II	Pymoo
NSGA-III	Pymoo
MOEA/D	Pymoo
SMS-EMOA	Equipo de desarrollo

Módulo 3. Dominancia de Pareto
El módulo de dominancia de Pareto, esta ideado para la manipulación de los resultados calculados por el algoritmo, denominado frente y conjunto de Pareto, el objetivo de este módulo es proporcionar una manera estandarizada de utilizar las soluciones de Pareto guías y/o calculadas por un algoritmo para su consumo.

Módulo 4. Indicadores de desempeño

Este módulo encapsula los objetos propios de Pymoo para que a partir del identificador del indicador de desempeño y los parámetros necesarios se pueda calcular el valor adimensional que califica el desempeño de la solución de un MOEA. Los indicadores de desempeño disponibles son el Híper Volumen, Distancia generacional, en sus versiones normal, invertida y *plus*.

Módulo 5. Estadísticas no paramétricas

El módulo NPS (*None Parametric Statistics*), implementa la prueba no paramétrica de Mann-Whitney, la cual es utilizada para determinar si existen o no diferencias significativas entre dos conjuntos de resultados a determinado MOP por diferentes MOEAs.

Módulo 6. Experimentación

El módulo Experimentación, hace uso de los módulos descritos previamente para que, a partir de una configuración compuesta por un conjunto de problemas multiobjetivo, un AEMO y algún indicador de desempeño, se ejecute el modelo definido por un usuario final a un alto nivel.
En conjuntos todos estos módulos desarrollados en el lenguaje de programación Python, componen el modelo del sistema.

C. Modelo de Vista
Este módulo encapsula la lógica de presentación y estado. La lógica de presentación se define como la lógica de la aplicación que se ocupa de casos de uso de la aplicación (o casos de usuario, flujo de trabajo, etc.) y define el comportamiento lógico y la estructura de la aplicación. Para

maximizar las oportunidades de reutilización, el ViewModel no debe tener ninguna referencia a las clases específicas de interfaz de usuario, elementos, controles o comportamiento [6].
La última parte del sistema se desarrolló como un servidor web, para ello se usó el lenguaje de programación Python mediante el micro framework Flask, el cual ofrece la capacidad de escalar aplicaciones complejas mediante el uso de servicios web.

El servidor permite que mediante peticiones bajo el protocolo HTTP se mantenga un canal de comunicación con la vista, es a partir de estas peticiones que la vista proporciona los datos necesarios, mediante un objeto JSON, para que la aplicación del servidor haga uso del módulo integral de experimentación, obtenga el resultado y envíe la respuesta para ser interpretada por la lógica del componente de la vista hacia el usuario final.

Con este último componente se logra la integración completa del sistema siguiendo el patrón de diseño descrito al inicio.
El despliegue del sistema se realizó utilizando la tecnología de contenedores disponible con Docker, para ello cada uno de los componentes se virtualizo dentro de un contenedor que permite ser desplegado en cualquier equipo que tenga instalada la plataforma Docker. Con esta capacidad se garantiza que el sistema es interoperable y puede ser utilizada sin importar el sistema operativo host.
.

La vista de GRATMOO se encuentra dividida en cinco secciones principales: constructor, favoritos, experimentos, análisis y resultados, cada una de ellas se explican en los siguientes párrafos.

1. *El constructor*

Esta sección está dedicada a la creación del flujo de bloques que define el modelo a resolver. El usuario puede crear un modelo a partir de bloques predefinidos de POMs, AEMOs y operadores de cruza y mutación o bien cargarlo desde un archivo previo. La figura 4, muestra un flujo básico.

2. *Favoritos*

El apartado denominado favoritos permite visualizar los modelos guardados en el navegador, en la sección constructor, es aquí donde se puede gestionar si se desea actualizar el modelo, removerlo y/o descargarlo.

3. *Experimentos*

En esta sección el usuario puede elegir el modelo a resolver, el indicador de desempeño que se desea añadir al modelo y al obtener los resultados se pueden descargar un archivo o bien ser guardados en el navegador.

4. *Análisis*

La vista de análisis muestra los resultados obtenidos de la solución de un modelo, es aquí donde mediante una gráfica se observan las soluciones del frente de Pareto, de igual forma se puede visualizar la gráfica de convergencia para observar el desempeño del algoritmo en cada generación o por cada *n* evaluaciones. La figura 6, muestra una gráfica del frente de Pareto de un problema resuelto por la herramienta.

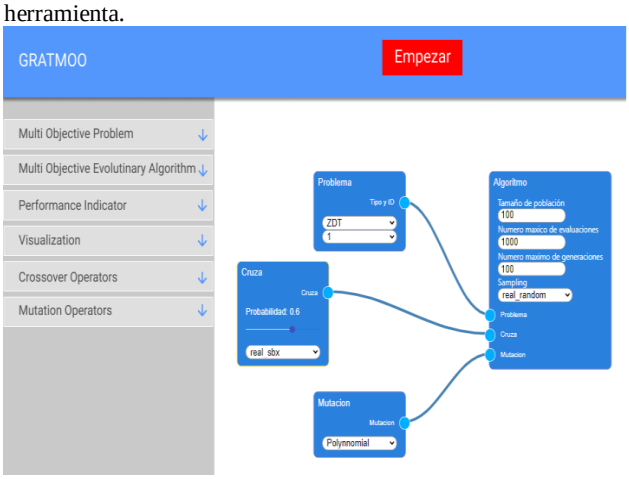

Figura 4. Flujo básico de bloques de un modelo

5. RESULTADOS

La última sección permite al usuario visualizar los resultados que han sido almacenados en el navegador para observarlos en la sección de análisis. De la misma manera permite descargar los resultados en un archivo o bien mediante un archivo valido cargar resultados para realizar un análisis. Aquí también puede ejecutar un proceso para obtener un reporte comparativo sobre el desempeño de dos algoritmos sobre un grupo de POMs con la prueba Mann-Whitney.

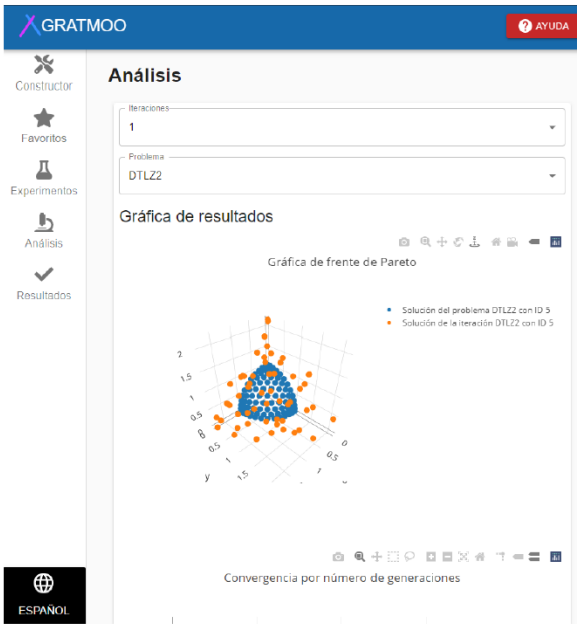

Figura 5. Frente de Pareto para el problema DTLZ-2

Problema	NSGA3		MOEAD		PI Value	Diferencia significativa
	Promedio	Desviación estándar	Promedio	Desviación estándar		
DTLZ1	13547.6342	3514.0675	14222.8294	1989.4367	0.3173	×
DTLZ3	12152.1432	5182.4741	14069.8800	2399.2268	0.3173	×
WFG1	11519.7099	971.4851	10828.0325	427.3678	0.3173	×
WFG2	14545.3763	358.0123	12358.8832	166.0835	0.3173	×
WFG3	12778.8875	234.0261	12549.9449	320.6644	0.3173	×

Figura 6. Resultados estadísticos de una experimentación

Las cinco secciones descritas anteriormente ofrecen al usuario la capacidad de cambiar el idioma a inglés. Estas características permiten la solución problemas multiobjetivo y análisis de sus resultados de una manera visual. Al igual que las vista este contenido audiovisual también está disponible en el idioma español e inglés. La figura 6 muestra la forma en la cual se integran este material a la vista desarrollada.

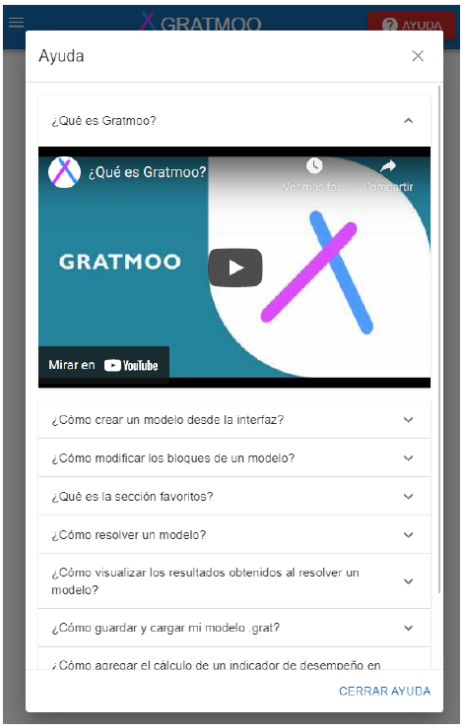

Figura 7. Ejemplo de video tutoriales para el uso de la aplicación

V. RESULTADOS

El sistema fue desplegado en un entorno de nube, esto gracias a la capacidad de interoperabilidad otorgada por los contenedores en Docker. Se realizaron distintas pruebas con diferentes modelos de bloques creados desde la interfaz de usuario, estas pruebas incluyeron la resolución de distintos grupos de problemas desde dos hasta cuatro objetivos, esto limitado por la capacidad del hardware la cual se veía superada al incrementar este valor.

En todos los casos el resultado fue satisfactorio, se concluyeron de forma exitosa y logro visualizar los resultados de cada uno de los problemas propuestos en los modelos.

De igual forma la herramienta demostró ser un complemento de apoyo en la ejecución de pruebas, con ella se pueden modificar parámetros de los problemas multiobjetivo y algoritmos de una forma visual a través los bloques predefinidos, dicha característica agiliza el proceso de prototipado y permite la ejecución del flujo y su análisis sin tener que configurar un entorno dedicado a un framework y adquirir los conocimientos técnicos que esto conlleva.

VI. CONCLUSIONES

Actualmente las ciencias de la computación son un área con diversas ramas de estudio, una de ella es el computo evolutivo, la cual busca resolver problemáticas tomando como inspiración en la teoría de la evolución.

Con esta aplicación se obtuvo como resultado un prototipo de herramienta de apoyo para la resolución de problemas de multiobjetivo con algoritmos evolutivos mediante un flujo de trabajo visual a bloques. La herramienta ha sido desarrollada como una aplicación web para lograr interoperabilidad y flexibilidad entre los distintos sistemas operativos y dispositivos existentes hoy en día. Además, se han utilizado tecnologías escalables y con gran apoyo entre la comunidad de desarrolladores y científica.

Se desarrollo un sistema que permite de forma interactiva y visual la solución de familias representativas de problemas benchmarking con los algoritmos evolutivos NSGA-II, NSGA-III, SMS-EMOA y MOEA/D todo esto con la creación de un flujo a bloques que permite modificar parámetros generales y la posibilidad de analizar los resultados en forma de graficas o generando un reporte para evaluar el desempeño de cada tipo de algoritmo.

VII. RECONOCIMIENTOS

Los autores agradecen a la Escuela Superior de Cómputo del Instituto Politécnico Nacional por el apoyo recibido y las facilidades otorgadas para el desarrollo del trabajo como parte de la opción de titulación para la carrera de Ingeniería en Sistemas Computacionales. Así como el financiamiento del proyecto SIP No. 2023052

VIII. REFERENCIAS

[1] Codejig. Block coding. [En línea]. Disponible en: https://www.codejig.com/en/block-based-coding/. [Último acceso: 14 febrero 2020].

[2] Tiwari, A., and K.T.Sekharb, A.Workflow based framework for life science informatics.Computational Biology and Chemistry 31(Octubre 2007), 305–311.

[3] Hernández Gómez, R. Hiper-Heurísticas Paralelas para Optimización Multi-Objetivo. PhD thesis, Instituto Politécnico Nacional, 2018.

[4] Pescador-Rojas, M., and Coello, C. A. C. Collaborative and adaptive strategies of different scalarizing functions in MOEA/D. In 2018 IEEE Congress on Evolutionary Computation(CEC)(2018), pp. 1–8.

[5] Gamma, E.Design patterns: elements of reusable object-oriented software. Pearson Education India, 1995.

[6] Arcos Medina, G., Menéndez, J., and Vallejo, X. Comparative study of performance and productivity of mvc and mvvm design patterns. KnE Engineering 1(01 2018), 241.

[7] Rao, V. R. Here's why you should use python for scientific research. [En línea]. Disponible en: https://developer.ibm.com/blogs/use-python-for-scientific-research/ [Último acceso: 16 octubre 2020], 2018.

[8] Blank, J.Multi-objective optimization in python.[En línea]. Disponible en: https://pymoo.org/index.html. [Último acceso: 10 octubre 2023], 2020.

[9] Van Veldhuizen, D. A., and B. Lamont, G. Evolutionary Computation and Converence to a Pareto Front. Stanford University Bookstore, Stanford, 1998.

[10] Codejig. Block coding. [En línea]. Disponible en: https://www.codejig.com/en/erp/. [último acceso: 10 Octubre 2023].

[11] Tiwari, A., and K.T.Sekharb, A. Workow based framework for life science informatics.
Computational Biology and Chemistry 31 (Octubre 2007), 305-311.

[12] Glushkova, T., and Academy, I. Application of block programming and game-based learning to enhance interest in computer science. Journal of innovations and sustainability 2 (03 2016), 21-32.

[13] Software Scratch, https://scratch.mit.edu

[14] Software app inventor, https://appinventor.mit.edu

[15] Software Rapid Miner, https://rapidminer.com