INSTITUTO TECNOLÓGICO DE CIUDAD MADERO

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

TESIS

DESARROLLO DE ARQUITECTURA DE SOPORTE A LA TOMA DE DECISIONES PARA AGENTES EN HARDWARE

Que para obtener el Grado de

Maestro en Ciencias de la Computación

Presenta

I.S.C César Alejandro Aguilar Rodríguez

G13071757

Director de tesis

Dr. Nelson Rangel Valdez

Co-director de tesis

Dra. Claudia Guadalupe Gómez Santillán

Cd. Madero, Tamaulipas Mayo, 2021

Autorización de Impresión de Tesis

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

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

Cd. Madero, Tam. 18 de mayo de 2021

OFICIO No. : U.015/21
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN DE TESIS

C. CÉSAR ALEJANDRO AGUILAR RODRÍGUEZ
No. DE CONTROL G13071757
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:

"DESARROLLO DE ARQUITECTURA DE SOPORTE A LA TOMA DE DECISIONES PARA AGENTES EN HARDWARE"

El Jurado está integrado por los siguientes catedráticos:

PRESIDENTE:	DRA. MARÍA LUCILA MORALES RODRÍGUEZ
SECRETARIO:	DR. HÉCTOR JOAQUÍN FRAIRE HUACUJA
VOCAL:	DR. NELSON RANGEL VALDEZ
SUPLENTE:	DRA. CLAUDIA GUADALUPE GÓMEZ SANTILLÁN
DIRECTOR DE TESIS:	DR. NELSON RANGEL VALDEZ
CO-DIRECTOR DE TESIS:	DRA. CLAUDIA GUADALUPE GÓMEZ SANTILLÁN

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"®

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

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE CIUDAD MADERO
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

c.c.p.- Archivo
MACG 'mdcoa'

Av. 1° de Mayo y Sor Juana I. de la Cruz S/N Col. Los Mangos,
C.P. 89440 Cd. Madero, Tam. Tel. 01 (833) 357 48 20, ext. 3110
e-mail: depi_cdmadero@tecnm.mx
tecnm.mx | cdmadero.tecnm.mx

México 2021 Año de la Independencia

Declaración de Originalidad

Yo, César Alejandro Aguilar Rodríguez en mi calidad de autor declaro y prometo que este documento de tesis es producto de mi trabajo original y que no infringe los derechos de terceros, tales como derecho de publicación, derechos de autor, patentes y similares.

Además, declaro que en las citas textuales que he incluido (las cuales aparecen entre comillas) y en los resúmenes que he realizado de publicaciones ajenas, indico explícitamente los datos de los autores y las publicaciones.

Además, en caso de infracción a los derechos de terceros derivados de este documento de tesis, acepto la responsabilidad de la infracción y relevo de ésta a mi director y co-director de tesis, así como al Instituto Tecnológico de Cd. Madero y sus autoridades.

Cd. Madero, Tamaulipas, Mayo 2021.

I.S.C. César Alejandro Aguilar Rodríguez

Agradecimientos

Quiero agradecer a mis padres que han sido mi fortaleza del día a día, sin ellos no estaría en estos momentos, por siempre darme su cariño y consejos, como este logro que también es de ellos.

Agradezco a mi director de tesis, Dr. Nelson Rangel Valdez por permitirme ser uno de sus tesistas, por brindarme su tiempo y apoyo. Por sus consejos buscando siempre sacar lo mejor de mí, estaré siempre agradecido.

A los doctores que estuvieron en mi estancia de estudiante por sus enseñanzas y amor al aprendizaje, es algo que me llevare de ustedes.

A mis amigos del tecnológico que me apoyaron en estos años, encontrando grandes personas con ganas de salir adelante, como también cada una de las personas que laboraba en el campus 3 del tecnológico de ciudad madero, personas que fueron muy amables.

Y por último agradecer a todas las personas que han estado en mi vida desde antes apoyándome día a día, disfrutando de mis logros como si fueran suyos. Muchas Gracias.

TABLA DE CONTENIDO

Contenido

Índice de figuras

Índice de Tablas

Capítulo 1. Introducción

1.1.Introducción

En la vida real los problemas tienden a ser multiobjetivo. Esto, por lo general, genera una problemática al buscar optimizarlos, es decir, buscar que se logre que todos los objetivos sean simultáneamente factibles, ya que alcanzar dicho estatus en un objetivo usualmente provoca que otros dejen de serlo, por lo que su resolución es compleja, computacionalmente hablando. Aun así, dentro del área de optimización computacional existen diferentes algoritmos que nos ayudan a aproximarnos a un conjunto de soluciones factibles, dentro de ese conjunto se encuentran los algoritmos evolutivos multiobjetivo, algunos de los algoritmos son (Pratap, A., Agarwal, S. and Meyarivan, T., 2002)NSGA-II (Non-dominated Sorting Genetic Algorithm), NOSGA-II (Non-Outranked Sorting Genetic Algorithm), (Rivera, G., 2015) NOACO (Non-Outranked Ant Colony Optimization), (Sanchez, J., 2017) H-MCSGA (Hybrid Multi-Criteria Sorting Genetic Algorithm), (Quingfu, Z. and Hui, L., 2007) MOEA-D (Multiobjective Evolutionary Algorithm Based on Descomposition), etc.

Dentro del conjunto de soluciones que los algoritmos previamente mencionados, usados para resolver problemas de optimización multiobjetivo surge otra problemática a resolver la cual es que toda solución factible es equivalente matemáticamente, por lo que es necesario implementar un tomador de decisiones (Tomador de Decisiones, DM) que generalmente se preocupa por varios criterios que determinan la calidad de las soluciones. El DM es la parte esencial para la toma de decisiones, ya que se encarga de escoger la solución que más se adapta a las preferencias de entre todas las posibles. Para lograr obtener las preferencias del DM e incorporarlas dentro de los algoritmos de optimización se necesita de modelos matemáticos capaces de incorporar las preferencias del DM dentro del proceso algorítmico. Dentro de la literatura existe una familia de métodos llamado ELECTRE (Elimination et Choix Traduisant la Realite) en específico ELECTRE III (Roy, 1978) que están basados en la definición de relaciones de superación.

1.2.Motivaciones

Dentro de la literatura existen algoritmos de optimización dedicados a resolver problemas multiobjetivo, así como algoritmos capaces de incorporar preferencias de un DM y reducir tiempo en el acotamiento de espacio de búsqueda para escoger el mejor compromiso o solución. Actualmente muchas investigaciones promueven la mejora de estos algoritmos tanto en calidad de soluciones como en reducción de tiempo, sin embargo, todos ellos utilizan hardware de gran

capacidad. Es por esto que este trabajo se centra en proveer una arquitectura capaz de implementar un algoritmo de optimización pero que a su vez sea capaz de incorporar preferencias dentro de un hardware limitado, como son las tarjetas microcontroladoras de Arduino, en concreto en la MEGA 2560 que compara la calidad de soluciones que este hardware puede proporcionar como el tiempo.

1.3.Problema de Investigación

Se tiene un robot censando un conjunto de N materiales donde el i-ésimo material está representado por un vector p-dimensional $f(i) = \langle f_1(i), f_2(i), \ldots, f_p(i) \rangle$, donde cada $f_j(i)$ indica la contribución del material i al j-ésimo objetivo. Cada objetivo representa el valor obtenido de a cada sensor (temperatura, humedad, gas, distancia). Cada material tiene un costo asociado expresado por C_i. Por otro lado, una caja x es un subconjunto de materiales modelada como un vector binario $x = \langle x_1, x_2, \ldots, x_N \rangle$, donde N indica el número de materiales.

Este vector, x_i es una variable binara donde $x_i = 1$ representa que el i-ésimo material se escoge y $x_i = 0$ no se escoge. El robot debe escoger los mejores materiales que son convenientes al criterio de un individuo mediante un modelo de preferencias $OM = \{w_i, p_i, q_i, v_i\}$ donde representan (peso, preferencia, indiferencia y veto) respectivamente. Los materiales están sujetos a una restricción que es representado como B y se representa como $\sum_{i=1}^{N} x_i C_i \leq B$ que es la capacidad máxima disponible.

1.4.Justificación

Se ha demostrado que los algoritmos evolutivos que incorporan preferencias han tenido buenos resultados cuando se habla del problema PSP. Se encuentran muchas propuestas de algoritmos que se comparan contra otros algoritmos de la literatura con base en tiempo, calidad de soluciones o cardinalidad, pero son pocos los algoritmos que se han tratado de implementar en un Hardware limitado que pudiera ser relevante la información obtenida ya que se pudieran abrir líneas de investigación sobre las diferentes tarjetas de Hardware que existen en el mercado.

1.5.Objetivos de la investigación

A continuación, se muestran los objetivos, tanto general como específicos de esta investigación:

1.5.1. Objetivo general

Integrar algoritmos de optimización con manejo de preferencias en agentes de hardware con recursos limitados para apoyo a la toma de decisiones.

1.5.2. Objetivos específicos

- Analizar el manejo de preferencias en algoritmos de optimización a través de métodos de superación.
- Desarrollar arquitectura de integración de algoritmos de optimización en Agente Hardware.
- Implementar algoritmos de optimización con manejo de preferencias en Agente hardware.

1.6. Alcances y Limitaciones

I. Como caso de estudio se propone el problema de cartera de proyectos con cuatro objetivos;
II. Para la arquitectura de Hardware se usan cuatro sensores de Arduino;
III. Se utiliza una tarjeta de Arduino con un microcontrolador Arduino MEGA 2560;
IV. Incorporar preferencias en MOEA-D;
V. La capacidad de la tarjeta de Arduino cuenta con una memoria Flash de 256 KB, una memoria dinámica de 8 KB y una velocidad de 16 MHz;
VI. Como modelo de preferencias se utiliza el método ELECTRE III;
VII. Se utiliza como base el algoritmo evolutivo multiobjetivo MOEAD.

1.7. Organización del documento

En este documento se desarrollan los siguientes capítulos que representan como se ha distribuido la información para la finalización de este trabajo. A lo largo del Capítulo 2 se comunican los argumentos teóricos que representan esta investigación, tomando en cuenta la descripción del problema a resolver, así como los módulos que integran esta investigación. En el Capítulo 3 se describen los diferentes algoritmos que utilizan modelos de preferencia mediante ELECTRE, también algoritmos que han trabajado con un problema de optimización como lo es cartera de proyectos y algunas arquitecturas propuestas en el estado del arte de agentes inteligentes; este capítulo presenta el análisis del estado del arte. En el Capítulo 4 se describe la forma en la que se analizó y se propuso la metodología para resolver la problemática de este trabajo y una propuesta de solución, como lo es la descripción del caso de estudio, de la instancia, una propuesta para la incorporación de preferencias en un algoritmo de optimización y cómo interactúa cada elemento

dentro de la arquitectura propuesta para la implementación en Hardware. En el Capítulo 5 se presenta la validación experimental y por último en el Capítulo 6 se dan conclusiones obtenidas de este trabajo como posibles trabajos futuros.

Capítulo 2. Marco Conceptual

2.1. Optimización multiobjetivo

En muchas disciplinas surgen problemas que involucran la optimización de múltiples objetivos, de hecho, no es una idea tan reciente como suele creerse. Es una parte inherente del equilibrio económico y, por lo tanto, puede rastrearse al menos desde el siglo XVIII (Rivera, 2015). Se genera un problema de optimización multiobjetivo cuando no se conoce la manera óptima de combinar los diferentes objetivos o estos sean inadecuados (Meza, 2019).

Según Zhang and Li (2007) un problema de optimización multiobjetivo (MOP) puede ser como el siguiente:

$$\begin{aligned} &\text{maximizar } F(x) = \big(f_1(x), \dots, f_m(x)\big)^T \\ &\text{sujeto a } x \in \Omega \end{aligned} \tag{1}$$

Donde Ω es el espacio de decisión factible (variables), $F: \Omega \to R^m$ es una función que mapea el espacio de decisión factible Ω a el espacio de valores de las funciones objetivo R^m, compuesto por m funciones objetivo. El conjunto de valores de objetivos alcanzables es definido como el conjunto $\{F(x) | x \in \Omega\}$, donde Ω es descrito por:

$$\Omega = \big\{x \in R^n \big| h_j(x) \leq 0, j = 1, \dots, m\big\} \tag{2}$$

Donde $h_j(x)$, para $j = 1, \dots, m$, es el conjunto de restricciones consideradas. Este tipo de problemas a menudo suelen estar en conflicto al momento de optimizarlos porque no se pueden maximizar todos los objetivos simultáneamente.

2.2. Problema de selección de cartera de proyectos

Los problemas de distribución de recursos son tan ubicuos como la actividad humana. Los hombres (como ente aislado) y las organizaciones (de cualquier índole) son incapaces de llevar a cabo todos los proyectos que idean, principalmente por el carácter finito de los recursos, ya sean fondos, capacidad o tiempo. Y es que, prácticamente, siempre los proyectos requieren recursos. La organización encara entonces el problema de cómo distribuirlos entre estas actividades

transformadoras. En este contexto, un proyecto no es más que la búsqueda de una solución inteligente a un problema que intenta resolver una necesidad humana (Rivera, 2015).

Una cartera x es un subconjunto de proyectos modelada generalmente por un vector binario $x = \langle x_1, x_2, \dots, x_n \rangle$ donde n indica el número de proyectos. El vector x_i es una variable binaria donde $x_i = 1$ si el i-esimo proyecto es aceptado y $x_i = 0$ en caso contrario.

Según (Sánchez, 2017), cada proyecto *i* de una cartera *x* está representado por un vector de m-dimensional $f(i) = \langle f_1(i), f_2(i), \dots, f_m(i) \rangle$, donde cada $f_j(i)$ indica la contribución del proyecto i al j-ésimo objetivo. Cada proyecto tiene un costo asociado expresado por c_i. Considerando los valores de las funciones objetivo $F(x) = \langle f_1(x), f_2(x), \dots, f_m(x) \rangle$ y el costo $C(x)$ de una cartera *x* se obtienen a partir de los valores acumulado de objetivos y costo de cada proyecto *i* que forman parte de la cartera, el Problema de Selección de Cartera de Proyectos (PSP) se puede definir formalmente como $\max_x\{F(x)|C(x) \leq Budget\}$, donde *Budget* es el presupuesto máximo disponible para crear la cartera.

Cabe mencionar que, en este trabajo de tesis, los objetivos están asociados a valores de los sensores considerados, que son de temperatura, humedad, gas, ultrasonido.

2.3.Algoritmos evolutivos multiobjetivo

Los algoritmos evolutivos (AE), se consideran como técnicas de aprendizaje no supervisados que, en general, no requieren de ningún tipo de conocimiento ya que el propio algoritmo es capaz de generar por sí mismo el conocimiento que requiere. Se consideran como técnicas meta-heurísticas de búsqueda que usan operadores probabilísticos y que funcionan como cajas negras, pues no requieren conocimiento específico del dominio para actuar, y que son aplicables a un gran número de aplicaciones.

Los AE trabajan con una población de soluciones candidatas que se utilizan como variables para producir un conjunto de soluciones en una sola ejecución y son insensibles a la forma de las funciones objetivo como la discontinuidad, la no uniformidad del espacio de búsqueda, entre otros (Hion, 2018). La Tabla 1 presenta el pseudocódigo básico de un Algoritmo Evolutivo.

Algoritmo 1. Algoritmo Evolutivo tradicional

Input:

P_{tam} =Tamaño de la población

P_{cruce} = Tipo de cruza entre individuos de la población

P_{mut} = Tipo de muta con los individuos de la población

Output:

1. $P \leftarrow$Inicializar_Población (P_{tam});
2. Evaluar(*P*);
3. **Do**

4. P'←Aplicar_Cruce(P,P_{cruce});
5. P''←Aplicar_Mutación(P',P_{mut});
6. Evaluar(P'');
7. P←Seleccionar($P \cup P$'');
8. **While** (no se cumpla la condición de parada)
9. Retornar Mejor Solución(P);
10. **End EA**

Tabla 1. Pseudocódigo básico de un algoritmo evolutivo.

La finalidad de los MOEAs es ayudar al Tomador de Decisiones (DM) a seleccionar la solución final que vaya más acorde a sus preferencias. Sin embargo, los MOEAs proporcionan al DM un amplio número de soluciones, lo cual dificulta la tarea de elegir la solución final. Por el contrario, el DM no está interesado en que se le presente el frente de Pareto completo, sino una solución que corresponda con sus preferencias. Con la finalidad de encontrar esa única solución, nuevos métodos, que en su mayoría son variantes de los MOEAs existentes, utilizan las preferencias del DM para guiar la búsqueda hacia una porción preferida del frente de Pareto, es decir, la región de interés del DM (Sánchez, 2017).

2.3.1. Algoritmo evolutivo multiobjetivo basado en descomposición (MOEA/D)

La descomposición es una estrategia básica tradicional de optimización multiobjetivo. Sin embargo, aún no se ha utilizado ampliamente en optimización evolutiva multiobjetivo. Un algoritmo evolutivo multiobjetivo basado en la descomposición (MOEA-D) descompone un problema de optimización multiobjetivo en una serie de subproblemas de optimización escalar y los optimiza de manera simultánea. Cada subproblema está optimizado sólo utilizando información de sus varios subproblemas vecinos, lo que hace que MOEA-D tenga una menor complejidad computacional en cada generación que MOGLS. La Tabla 2 presenta el pseudocódigo de MOEA/D.

Algoritmo 2. Variante de MOEA/D

Entrada:

N= Número de funciones escalares.

K= Número de objetivos.

*Vector= C*onjunto de vectores de peso distribuidos uniformemente.

T=N/2, Tamaño del vecindario de los vectores de peso.

Salida:

EP = Aproximación a la frontera de Pareto.

0. (x,z,FV,$B(i)$) ← Inicialización()
1. For i = 1 a N hacer
2. (x_k, x_l) ← SelecciónAleatoria($B(i)$,T)
3. y ← CruzaSBX(x_k, x_l)
4. y' ← MutaciónPolinomial(y)
5. y'' ← OperadorRepaciónMejora (y')
6. ActualizarConjuntoZ(z,K,y'')

z: para cada j =1, …K, si $z_j < f_j(y')$ entonces el conjunto $z_j = f_j(y')$.

7. ActualizarVecindario($B(i)$,FV,y'') para cada $j \in B(i)$, si $g^{te}(y' | V_j, z)$ el conjunto $x_j = y'$ y $FV_j = F(y')$.
8. ActualizarEP(EP,y'') Remover de EP todos los conjuntos de vectores dominados por $F(y')$, Añadir $F(y')$ a EP si no existen vectores en EP dominador por $F(y')$.
9. **Criterio de Parada:** Si se alcanza el máximo de evaluaciones de lo contrario pasa a la línea 1.

Tabla 2. Pseudocódigo del algoritmo MOEA/D

2.4.Frente de Pareto

La optimización multiobjetivo, no se restringe a la búsqueda de una única solución, sino de un conjunto de soluciones llamadas *soluciones no-dominadas.* Cada solución de este conjunto se dice que es un *óptimo de Pareto* y, al representarlas en el espacio de los valores de las funciones objetivos, conforman lo que se conoce como *frente de Pareto* (PF). El obtener el frente de Pareto es una de las principales finalidades de los problemas de optimización multiobjetivo (García, 2010).

Según Rivera (2015) la dominancia de Pareto supone un principio básico: Si x domina a y, un DM racional siempre preferiría a x sobre y. Como se muestra en la siguiente figura (Figura 1).

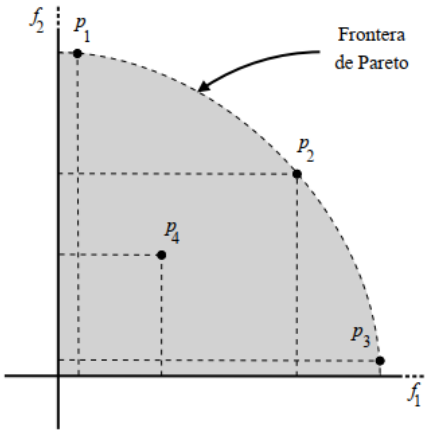


Figura 1. Ejemplo de la FP para un problema de maximización de dos objetivos (Rivera, 2015)

En la Figura 1 se muestra un problema de optimización de dos objetivos donde ambos objetivos son de maximización, por lo cual, existen tres soluciones contenidas en el PF por lo que matemáticamente esas soluciones son buenas para el DM.

2.5.Región de interés

La región de interés (RoI) es un subconjunto de PF con la característica de que estas soluciones están apegadas a las preferencias del DM (Meza, 2019). La Figura 2 muestra una representación gráfica de la RoI en comparación con el PF.

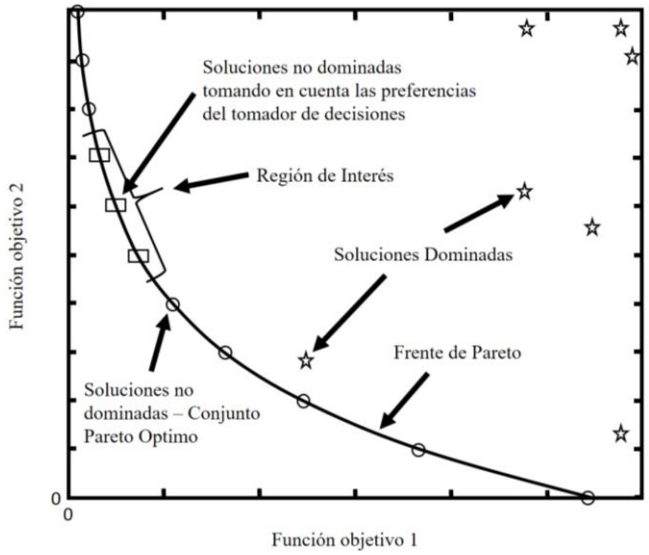


Figura 2. Región de interés (Meza, 2019)

En la Figura 2 se muestra un problema de optimización de dos objetivos, donde un subconjunto de soluciones que son no dominadas en el frente de Pareto son también parte de la región de interés, ya que son las soluciones que mejor definen las preferencias del tomador de decisiones.

2.6.Preferencias en la toma de decisiones

Las preferencias pueden ser expresadas ***a priori***, ***a posteriori***, o ***interactiva*** cuando se usan AE, dentro de las investigaciones que han utilizado preferencias encontramos las siguientes:

A priori: Las preferencias del DM son incorporadas antes de ejecutar el algoritmo. El proceso de optimización se encuentra guiado por la información de preferencias. Este enfoque tiene como incertidumbre que el DM puede no quedar satisfecho cuando se le presenta el conjunto de soluciones, porque los resultados se le muestran hasta el final del proceso de optimización dentro del algoritmo (Balderas, 2018).

A posteriori: Se ignora inicialmente la toma de decisiones, la idea principal es definir un subconjunto del conjunto completo de soluciones no dominadas que se van a presentar al DM con la finalidad de que el DM seleccione la solución compromiso.

Interactivo: Los enfoques interactivos dependen de la definición de las preferencias del DM junto con la exploración del objetivo. La manera de cómo funciona este enfoque es durante el proceso

de optimización, de manera progresiva hacia una región particular del frente. Aquí el DM es crucial, ya que debe de estar dispuesto a participar en el proceso interactivo de identificar las mejores soluciones, con esto el DM no solo aporta las preferencias, sino que aprende acerca del problema y esto ayuda a ajustar los niveles de aspiración (Sánchez, 2017).

2.7.Modelo relacional de preferencias ELECTRE III

El modelo de *elimination et choix traduisant la realité* (ELECTRE) "pertenece a la familia de métodos basados en relaciones de superación y consiste en determinar una solución, que sin ser óptima puede considerarse satisfactoria o bien se determina una jerarquización, alternativas bajo análisis. Este método fue desarrollado por la escuela francófona (Francia, Bélgica, Suiza)" (Meza, 2019).

Las preferencias en los métodos ELECTRE se modelan usando relaciones binarias de superación, S, cuyo significado es "al menos tan bueno como". Considerando acciones a y b, pueden ocurrir cuatro situaciones:

I. aSb y no bSa, es decir, aPb (a es estrictamente preferido a b).
II. bSa y no aSb, es decir, bPa (b es estrictamente preferido a a).
III. aSb y bSa, es decir, aIb (a es indiferente a b).
IV. No aSb y no bSa, es decir, aRb (a es incomparable a b).

Los métodos basados en superación se sustentan en dos grandes conceptos que de alguna manera manejan razones a favor o en contra de una situación de superación, estos conceptos son los siguientes:

I. Concordancia. Para una superación de aSb para ser validada, la mayoría de los criterios deberían estar a favor de esta afirmación.
II. Discordancia. Cuando se cumple la condición de concordancia, ninguno de los criterios de la minoría debe oponerse demasiado a la afirmación aSb.

Según (Figueira et al., 2010) los métodos ELECTRE comprenden dos procedimientos principales, los cuales son:

- Procedimiento de agregación: Construye una o posiblemente varias relaciones de superación, este procedimiento tiene como objetivo comparar de manera integral cada par de acciones.
- Procedimiento de explotación: Deriva recomendaciones a partir de los resultados obtenidos en la primera fase.

Los métodos ELECTRE también hacen uso de los umbrales mostrados en la Tabla 3.

Umbral	Descripción
w_i	Representa el peso del criterio *i*.
p_i	Representa el umbral de preferencia. Es el valor numérico por el cual el DM prefiere una alternativa sobre otra.
q_i	Representa el umbral de indiferencia. Es el valor numérico por el cual el DM es indiferente ante dos alternativas.
v_i	Representa el umbral de veto. Es el valor numérico el cual hace un bloqueo en la relación de superación entre dos alternativas en el criterio *i*, es decir, opone fuertemente a la afirmación de *aSb*.

Tabla 3. Elementos que conforman el método de ELECTRE III

Un ejemplo sobre el cálculo de la concordancia y discordancia como también el cálculo del índice de credibilidad se encuentra en el Anexo A al final de este documento.

2.8. Agentes

El concepto de agente, que deriva del latín “agree” (hacer), describe una abstracción de software, un programa que actúa para un usuario (humano o no), una idea o concepto similar al de los métodos, funciones y objetos, planteada en la programación orientada a objetos. El concepto provee una forma conveniente y poderosa de describir una compleja entidad de software, que es capaz de actuar con algún grado de autonomía, para cumplir tareas en representación o con los mismos objetivos de personas. A diferencia de los objetos (definidos por métodos y atributos), un agente es definido por su propio comportamiento.

Aunque no existe una única definición universalmente aceptada, un agente por defecto debe poder percibir su entorno por medio de “sensores” y realizar modificaciones sobre él, por medio de “actuadores” basado en la evaluación de técnicas de resolución de problemas (Hernández, H., 2016). Cualquier proceso computacional dirigido por un objetivo capaz de interaccionar con su entorno de forma flexible y robusta puede considerarse como un agente, el cual, basado en estos principios, tendrá las siguientes características:

Autonomía: capacidad de actuar sin intervención humana directa o de otros agentes.

Sociabilidad: capacidad de interaccionar con otros agentes, utilizando como medio algún lenguaje de comunicación entre agentes.

Reactividad: un agente está inmerso en un determinado entorno (hábitat) del que percibe estímulos y ante los que debe reaccionar en un tiempo preestablecido.

Iniciativa: un agente no solo debe reaccionar a los cambios que se produzcan en su entorno, sino que incluye un carácter emprendedor y tomar la iniciativa para actuar guiado por los objetivos que debe de satisfacer.

Movilidad: habilidad de trasladarse en una red de comunicación informática.

2.8.1. Arquitectura de Agentes

Arquitectura Reactiva

Se basa en el mapeo directo de la situación a la acción. Es diferente de la arquitectura basada en la lógica donde no hay un modelo simbólico central del mundo y un razonamiento simbólico complejo, este tipo de arquitectura está basada en respuesta a los cambios del entorno, acción-reacción. La Figura 3 muestra un ejemplo de arquitectura reactiva.

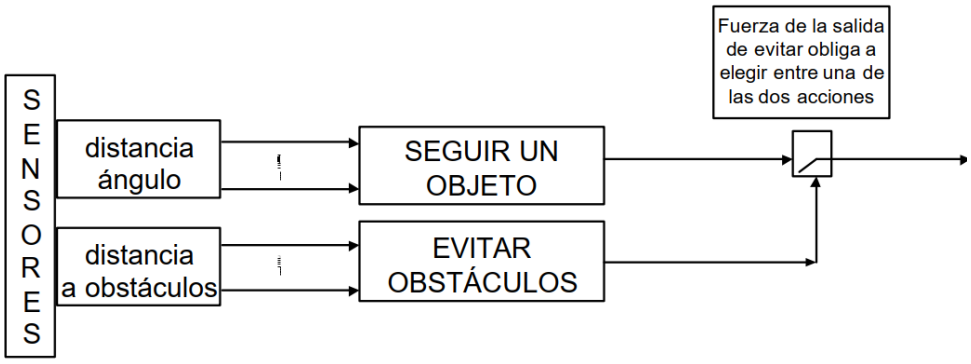


Figura 3. Arquitectura de Subsunción (Brooks, 1986)

Arquitectura Deliberativa

Están conformados por deseos, creencias e intenciones. Las creencias se representan por patrones lingüísticos descritos sintácticamente. Los deseos se entienden como la finalidad del agente para identificar contextos definitorios y las intenciones hacen referencia a una etapa de aprendizaje supervisado por un experto, lo cual conlleva a que el agente pueda identificar de manera automática dichos contextos. Las ***creencias*** representan la información que un agente tiene sobre su entorno. Los ***deseos*** representan las tareas asignadas al agente que corresponden a los objetivos que deben ser realizados por el agente. Las ***intenciones*** representan el compromiso del agente hacia los objetivos.

La Figura 4 muestra un ejemplo de arquitectura deliberativa, la cuál es conocida como ***PRS*** (Sistema de Razonamiento Práctico). Esta arquitectura utiliza *planes* en la representación del conocimiento procesal al describir cómo realizar un conjunto de acciones para alcanzar el objetivo. Los ***planes*** especifican algunos cursos de acción para el agente a fin de lograr sus intenciones.

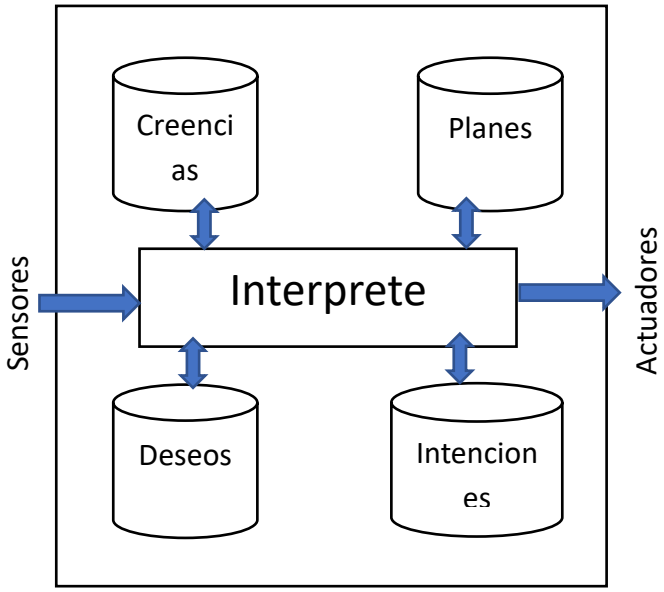


Figura 4. Arquitectura básica del Sistema de razonamiento práctico (PRS).

Arquitectura Híbrida

La arquitectura en capas (Híbrida), por ejemplo, la mostrada en la Figura 5, es una arquitectura de agente, que permite el comportamiento, tanto de un agente reactivo como un agente deliberativo. Tiene la ventaja de la arquitectura reactiva y la lógica, los subsistemas se descomponen en una capa de estructura jerárquica para lidiar con diferentes comportamientos.

Un ejemplo de esta arquitectura es Maquinas de Touring, que es una arquitectura híbrida de capas horizontales. Consta de tres capas productoras de actividad: una capa reactiva *R*, una capa de planificación *P*, y una capa de modelado.

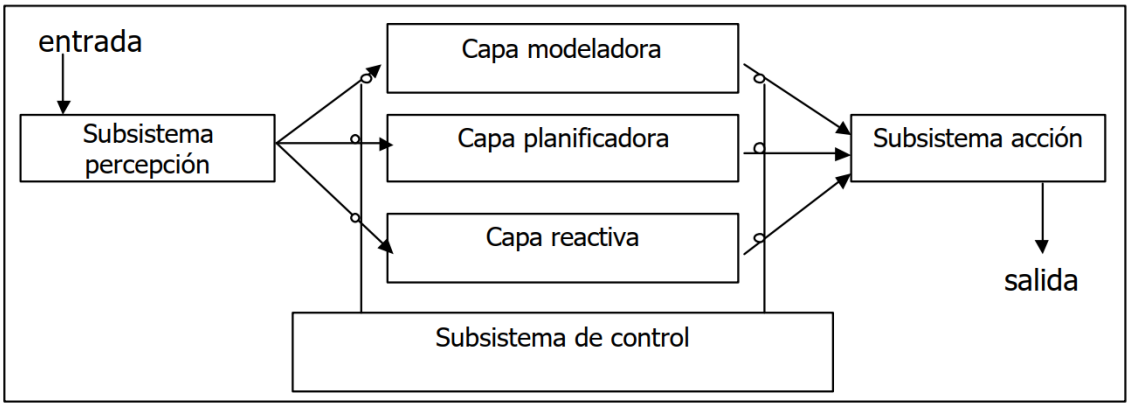


Figura 5. Arquitectura Híbrida (Ferguson, 1992).

Arquitectura Cognitiva

La arquitectura cognitiva se basa en la corriente de la ciencia cognitiva que se centra en la cognición y la psicología humana. Las arquitecturas cognitivas se utilizan para construir un sistema inteligente o un agente que modele el comportamiento humano. Trata de emular al cerebro que guarda recuerdos a corto y largo plazo y de ahí parte para almacenar las creencias, objetivos y conocimientos del agente.

Capítulo 3. Estado del Arte

En este capítulo se hace toda la recopilación de la información encontrada a través de los trabajos encontrados en la literatura, se comienza con las arquitecturas encontradas que cuentan con módulos capaces de incorporar toma de decisiones, así como también el análisis de los diferentes comportamientos que toman las arquitecturas contempladas para la creación de la arquitectura propuesta en este trabajo. Se procede con las diferentes estrategias que existen al momento de incorporar preferencias y por qué se tomó una de ellas. También se hace una comparación de los diferentes algoritmos de optimización que trabajan con problemas de selección de carteras (PSP) contra el algoritmo propuesto en este trabajo. Por último, se da una conclusión del estado del arte.

3.1. Arquitecturas de agentes que incorporan preferencias

Dentro de la literatura se tomaron a consideración diferentes propuestas de agentes que toman un comportamiento deliberativo como se menciona a continuación:

Ferretti et al. (2016) describe que la toma de decisiones y aprendizaje automático en sistemas inteligentes en la web y las temáticas son el uso de argumentación en la toma de decisiones, arquitecturas de razonamiento practico, agentes BDI, así como arquitecturas de agentes inteligentes y su comunicación entre agentes deliberativos.

Budán et al. (2011) describe el contexto de un sistema multi-agente (MAS), la problemática asociada con la especificación e implementación de un agente deliberativo respecto a la capacidad de brindar apoyo a la toma de decisiones, y la realización de un diseño y construcción computacional de una arquitectura para un agente autónomo.

Corchado (2005) describe los diferentes tipos de agentes que existen, así como una arquitectura estándar para el desarrollo de sistemas multiagentes.

Suárez (2009) muestra el análisis, diseño e implementación de un agente deliberativo, con un mecanismo de aprendizaje supervisado, que permite identificar contextos definitorios en textos especializados.

Castro (2018) propone una arquitectura de agentes que utiliza un modelo de preferencia basado en ELECTRE para resolver un problema de optimización PSP (Portafolio Selection Problem).

Chin et al. (2014) describe una arquitectura que utiliza un comportamiento deliberativo donde se tienen los módulos internos: creencias, intenciones, metas y planes.

Autores	Hardware/Arduino
Ferretti et al. (2016)	NO
Budán et al. (2011)	NO
Corchado (2005)	NO
Suárez (2009)	NO
Castro (2018)	NO
Chin et al. (2014)	NO
Este Trabajo	SI

Tabla 4. Arquitecturas de Agentes

En la Tabla 4 se muestran los artículos de donde se recogieron diversos conceptos para la creación de la arquitectura propuesta en este trabajo y cómo identificar los elementos involucrados. Por lo que, para la implementación en Hardware se requiere de una arquitectura que pueda guiar a una implementación del algoritmo en conjunto con la incorporación de preferencias, ya que hay diferencias al implementarlo en software con máquinas de mayores recursos contra aquellos con menores. En este trabajo se implementan, por ejemplo, las tarjetas de Arduino en Hardware con recursos limitado, así que es una arquitectura que intenta demostrar que se puede resolver problemas en esas condiciones analizando el desempeño y las limitantes de las mismas.

3.2.Diferentes enfoques en el manejo de preferencias

Según Coello (2000) existen diferentes enfoques en el manejo de preferencias, menciona que existen cinco enfoques los cuales son:

Logro de metas: La propuesta era extender el MOGA para acomodar la información de metas como un criterio adicional a la no dominancia para asignar rangos a la población.

Funciones de utilidad: La idea era clasificar un conjunto de soluciones del problema de optimización multiobjetivo (por sus siglas en inglés, MOP) en lugar de clasificar explícitamente los atributos del problema. La información de preferencia también se incorpora en los criterios de supervivencia utilizados por la EA.

Relaciones de preferencia: La idea era utilizar relaciones de preferencias binarias que puedan expresarse cualitativamente y luego traducirlas a términos cuantitativos (es decir, pesos) para restringir la búsqueda de una EA.

Superación: La propuesta era utilizar PROMETHEE (método de organización de clasificación de preferencias para el enriquecimiento de evaluaciones), combinado con un EA. Los métodos de

PROMETHEE pertenecen a la familia de los enfoques de superación (como los métodos ELECTRE).

Lógica difusa: La propuesta era utilizar un controlador difuso que regulara automáticamente la presión de selección de una EA mediante el uso de un conjunto de objetivos predefinidos que definen el comportamiento deseable de la población, aunque el enfoque se utilizó solo para mantener la diversidad de la población, podría ampliarse fácilmente para incorporar las preferencias del DM.

Todos estos enfoques abordan el problema de optimización PSP que es uno de los propósitos de este trabajo, resolver dicho problema. Meza (2019), menciona que independientemente del enfoque utilizado para manejar las preferencias en una EA, hay varias cuestiones que deben tomarse en cuenta, tales como: preservar la dominancia, la transitividad, la escalabilidad y el hecho de que pueden existir varios tomadores de decisiones. Se eligió la incorporación de preferencias a través de métodos de superación ya que algoritmos como NOACO, H-MCSGA, NOSGA o NOSGA-II han utilizado los métodos de superación, obteniendo buenos resultados tanto en tiempo de ejecución como calidad de soluciones, por otra parte, la sencillez de incorporar preferencias mediante métodos de superación como su ligera implementación son viables para ser implementadas dentro de un algoritmo como MOEA/D en Hardware.

3.3. Algoritmos de Optimización que Incorporan Preferencias

Como se mencionó en el punto 3.2 existen diferentes estrategias o algoritmos que tratan de resolver el problema de optimización PSP y a su vez plantean una incorporación de preferencias del DM dentro del proceso algorítmico. Estos datos se aclaran en la Tabla 5, que muestra una comparación de los algoritmos.

Autor	Algoritmo	Número de objetivos	Enfoque	Modelo	Arquitectura
Rivera (2015)	NO-ACO	16 objetivos	*A priori*	Relaciones de superación	Software
Sánchez (2017)	H-MCSGA	9 y 16 objetivos	*A priori / Interactiva*	Relaciones de superación	Software
Fernández et al. (2011)	NOSGA-II	9 objetivos	*A priori*	Relaciones de superación	Software
Este trabajo	PMOEA/D	4 objetivos	*A priori*	Relaciones de superación	Software / Hardware

Tabla 5.Comparación entre Algoritmos que incorporan preferencias.

Como se puede ver en la Tabla 5, dentro de la literatura existe un cierto número de algoritmos que han incorporado preferencias a través del modelo de Fernández. El algoritmo que se pretende comparar es NOSGA-II propuesto por el mismo autor (2011), ya que NOSGA toma como base el algoritmo NSGA-II propuesto por Deb et al. (2002), que ha demostrado buenos resultados cuando trabaja el algoritmo con 2 objetivos, pero cuando se ha trabajado con más de tres objetivos ha tenido un poco de deficiencias. Por otro lado, este trabajo de investigación toma como base el algoritmo MOEA/D propuestos por Zhang and Li (2007), que ha demostrado buenos resultados cuando trabaja con instancias de 3 o más objetivos, teniendo mejores resultados que NSGA-II.

Por tal motivo, teniendo en cuenta que MOEA/D genera buenas soluciones diversas a lo largo de la frontera de Pareto, se acoge la incorporación de preferencias para tener una RoI que tenga soluciones diversas y encontrar la mejor solución compromiso para el DM.

3.4.Análisis del Estado del arte

Tras analizar las tres tablas expuestas en este capítulo se puede notar en primera instancia que muchas arquitecturas tomadas de la literatura han sido implementadas con un comportamiento deliberativo basándose en arquitecturas como BDI, PRS etc. Al tener como limitantes un Hardware con pocos recursos, como lo son las tarjetas de Arduino, tomar como base una arquitectura deliberativa con pocos módulos como lo es PRS resulta en una implementación sencilla y ligera. En el subcapítulo 3.2 se pueden notar las diferentes maneras de incorporar preferencias dentro de algoritmos de optimización, algunos manejan las preferencias a través de funciones de penalidad, pero en este trabajo se manejan con el enfoque de superación. Para finalizar, en la Tabla 5 se observa la comparación de algoritmos de optimización que incorporan preferencias resolviendo PSP, denotando que este trabajo propone un algoritmo como MOEA/D por su sencillez de implementación, pero aportando algo nuevo a esos algoritmos que es ser implementado en un Hardware como Arduino. En conclusión, las aportaciones de este trabajo son las siguientes: a) La implementación de una arquitectura basada en PRS; b) Proponer un modelo de preferencias dentro de un algoritmo de optimización como MOEA/D; c) Implementar en Hardware con recursos limitados la arquitectura.

Capítulo 4. Metodología

Se describen a continuación dos puntos los cuales son la metodología de desarrollo y la metodología de investigación, en donde la metodología de desarrollo se encarga del plan para llevar el proceso de desarrollo del proyecto y la metodología de investigación involucra la definición y solución de los elementos necesarios para la solución del problema.

4.1.Metodología de Desarrollo

Dentro de la metodología de desarrollo se encuentran dos fases principales dentro de este proyecto, los cuales son: a) Análisis y Desarrollo de una Arquitectura que permita incorporar un algoritmo de optimización con preferencias; b) Implementación de un algoritmo de optimización como MOEA/D incorporando preferencias mediante relaciones de sobre clasificación.

Las actividades asociadas con las dos fases anteriores se desglosan de la siguiente forma:

1. Analizar qué arquitecturas existen dentro de la literatura que permitan incorporar preferencias.
2. Desarrollar una arquitectura de Hardware con los módulos que se necesitan.
3. Desarrollar un algoritmo de optimización que pueda ser implementado en Hardware con recursos limitados.
4. Desarrollar un modelo que permita incorporar preferencias dentro del algoritmo de optimización propuesto.

A continuación, se describen cada una de las actividades anteriormente mencionadas:

Para el **análisis de arquitecturas** se revisaron en el estado del arte diferentes arquitecturas que lograran incorporar preferencias. Se revisó qué comportamientos toman dichas arquitecturas para lograr identificar los módulos que se necesitan para la incorporación de preferencias, por qué tomar dicha arquitectura y no alguna otra.

Se **desarrolló una arquitectura** basada en el entorno MS-DOSS, pero era muy grande para ser implementada directamente en Hardware como Arduino por las capacidades limitadas, por lo que se optó por crear una versión ligera del entorno adaptando los módulos que requerían la arquitectura que se escogió.

En el **desarrollo del algoritmo** se tomó MOEA/D que es un algoritmo ligero por lo que se optó por implementarlo dentro de la arquitectura como el algoritmo de optimización.

En el **análisis de un modelo de preferencias** se estudiaron los modelos que existen para el manejo de preferencias y ver de una manera sencilla cómo lo es uno de los métodos de ELECTRE III, en concreto el cálculo de credibilidad y un modelo de sobre clasificación.

4.2.Metodología de Investigación

Dentro de la metodología se involucraron lo siguientes elementos para la solución del problema:

Definición del problema: Se resuelve un caso particular del problema de optimización PSP que consta de presupuesto, objetivos, y costo por cada proyecto. Basando en que se tiene una implementación en hardware se basa en un agente inteligente en el cual los proyectos son materiales, cada uno tiene un costo y las carteras se vuelven un conjunto de materiales tratando de minimizar sus objetivos medidos mediante sensores sin sobrepasar el presupuesto. También el agente debe de ser capaz de incorporar las preferencias del DM y es necesario conocer los umbrales de discriminación los cuales son dados por el DM hacia al agente y se describen en la metodología de solución.

Incorporación de Preferencias: Para la incorporación de preferencias se utiliza un método de ELECTRE III, en concreto el cálculo de credibilidad que es una estrategia basada en superación, pero además para tener más discriminación se propone un modelo relacional de sobre clasificación donde en estas estrategias se utilizan los umbrales de discriminación que fueron previamente dados por el DM.

Estrategia de Solución: Para resolver el problema de optimización PSP se necesitó de un algoritmo diseñado para resolverlo, el cual es MOEA/D pero proponiendo una estrategia para incorporar preferencias dentro del algoritmo.

4.3.Metodología de Solución

En la metodología de Solución se desglosa y se detalla paso a paso cómo se construyó la arquitectura y que módulos se involucran dentro de la arquitectura empezando por cómo construir la instancia PSP mediante la lectura de los sensores, la lectura de los umbrales de discriminación dados por el DM y como se implementaron los algoritmos MOEA/D y PMOEA/D y cuál es el procedimiento que se hace internamente dentro del algoritmo para obtener el subconjunto de soluciones que serán los mejores compromisos para el DM. Como objetivo principal de esta sección se tiene demostrar que se logró en la implementación de la arquitectura utilizar el algoritmo de optimización con preferencias para resolver el problema de optimización como lo es PSP en Hardware, en específico en una tarjeta de Arduino con recursos limitados.

Al trabajar con un problema que involucra tomar en cuenta un conjunto de preferencias del tomador de decisiones "DM", se estudió cual era la mejor opción que tomar. Los agentes a destacar fueron tres:

1. Agente Reactivo
2. Agente Deliberativo
3. Agente Híbrido

Los agentes con comportamiento Híbrido involucraban de los dos tipos de agentes (Reactivo y Deliberativo) por lo que conllevaba más módulos, lo que a su vez implicaba más espacio en memoria para la tarjeta de Arduino, por esta razón se descartó, ya que el trabajo intentaba tener un agente más ligero y más sencillo de implementar. Para el agente con comportamiento Reactivo, si era una la mejor opción para implementar viéndolo por el lado de sencillez y tamaño, pero al contener pocos módulos dentro de este agente era complicado proponer un modelo que fuera compatible con colocar preferencias dentro del agente, así que se analizaron los agentes con comportamiento deliberativo que se asemejaban más a este trabajo de incorporación de preferencias. Existen diferentes arquitecturas de agentes con comportamiento Deliberativo, pero también algunas arquitecturas eran más sencillas de implementar, por tal motivo dentro de este análisis se tomó la mejor arquitectura Deliberativa que existe, y fue la arquitectura PRS propuesta por Bratman en 1987 la que más se asemejó a nuestro problema por la parte del agente.

Los módulos principales dentro de la Arquitectura PRS se muestran en la Tabla 6.

Módulo	**Descripción**
Creencias	Representan la información que un agente tiene sobre su entorno
Planes	Especifican algunos cursos de acción para el agente a fin de lograr sus intenciones
Intenciones	Representan el compromiso del agente hacia los objetivos.
Deseos	Representan las tareas asignadas al agente que corresponden a los objetivos que deben ser realizados por el agente

Tabla 6. Módulos de la arquitectura PRS

4.3.1. Arquitectura de Hardware propuesta

Como propuesta de solución se tiene un modelo para el agente de hardware, el cual se muestra en la Figura 6.

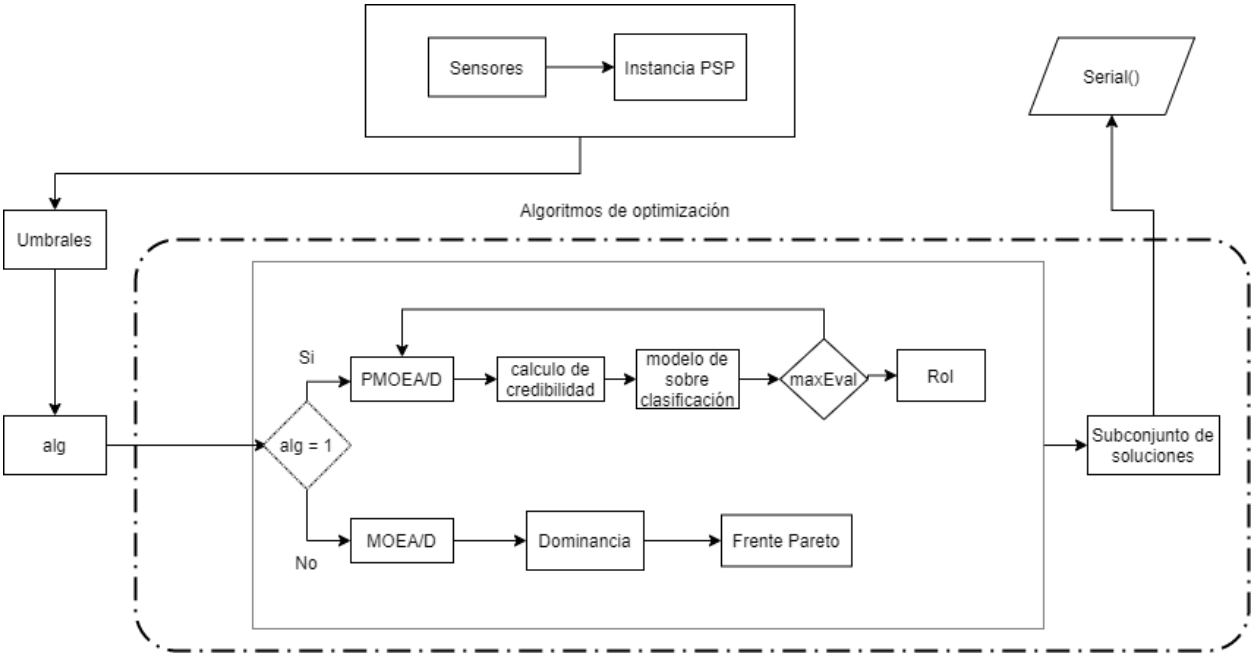


Figura 6. Arquitectura implementada en Arduino

Los módulos internos del Agente de Hardware se agrupan en 3 grandes elementos: 1) Lectura de Sensores, Instancia y Umbrales; 2) Incorporación de Preferencias del DM; y 3) Algoritmos de Optimización; Cada uno de estos módulos es explicado en detalle en el resto del capítulo.

4.3.2 Lectura de Sensores, Instancia y Umbrales

Este trabajo de tesis trata de involucrar la información obtenida externamente a través de los componentes que existen de Hardware para Arduino. Como se explicó en el punto 4.2 sobre la definición del problema a tratar, se busca poder resolver un problema de optimización que ha sido investigado por muchos años como lo es el problema de selección de cartera de proyectos. Al ser un problema de optimización que se quiere optimizar un conjunto de objetivos, la representación que se pretende explicar es que teniendo un conjunto de datos recopilados a través de sensores de Arduino se puede representar un problema de optimización como lo es el de selección de cartera

de proyectos. Tener un conjunto de materiales (proyectos) que contienen atributos como peso y dato censado para poder crear un conjunto de carteras que satisfagan al DM. Como una nota relevante para este trabajo es que al no poder trabajar con archivos de texto simple (para la lectura de la información), es necesario trabajar con un módulo externo de Hardware que permita la lectura de archivos, pero esto conlleva espacio en memoria lo cual se vuelve inviable.

Lectura de Sensores

La lectura de los sensores para este trabajo se llevó a cabo por separado del agente propuesto, ya que de la misma manera el contener sensores que estén trabajando en conjunto con el agente propuesto conlleva la terminación total de memoria de la tarjeta de Arduino (Para MEGA 2560), esto abre líneas de investigación futuras para trabajar con tarjetas de Hardware de mayor capacidad, o bien definir estrategias más eficientes en el manejo de la memoria. Una vez se obtuvo la información de los sensores en el puerto serial de Arduino se copiaron a un documento de texto para su futuro procesamiento dentro de los algoritmos.

Instancias

La Tabla 7 presenta la información que integra una instancia del problema abordado, y cuya solución se implementó a través del agente de Hardware.

Variable	Descripción
`NUM_CONSTRAINTs`	Es el número de restricciones que existe en el problema
`NUM_DMs`	Representa el número de DM que participaran en la toma de decisiones.
`NUM_OBJs`	Representa el número de objetivos.
`NUM_PROYECTs`	Representa el número de proyectos con el cual se cuenta que viene a representar los materiales.
`W_DMs`	Representa los pesos del DM
`Q_DMs`	Representa la indiferencia del DM
`P_DMs`	Representa la preferencia del DM
`V_DMs`	Representa el veto del DM
`CREDIBILITY_DMs`	Es el índice de credibilidad por cada DM (En este trabajo solo existe un DM)
`PROYECTS`	Representa una vector de tamaño m donde m representa sus objetivos por cada m-ésimo proyecto.
`BUDGET`	Representa el presupuesto con el cual se cuenta

Tabla 7. Información de las variables dentro del problema

Umbrales

A continuación, se ejemplifica como se construye un proyecto con base a la información dada por los sensores. Primero se define una matriz de $m \times m + 1$ donde la primera posición $[0,0]$ significa el coste de dicho proyecto, para las siguientes posiciones $[1,4]$ cada una representa cada dato recopilado por los sensores. Por ejemplo, una matriz *projects* de tamaño 4×5 con valores

{50, 67, 28, 111, 47} en la primera fila, representa que el proyecto 0 tiene un costo de 50, y valores censados para humedad, temperatura, gas y distancia de 67, 28, 111, y 47, respectivamente.

Para la representación de los umbrales de preferencias del DM, por cada umbral se define un vector n, por lo que por cada objetivo tendrá un valor asociado del DM. Existen cuatro elementos relevantes que necesita el modelo de preferencias que son (pesos, umbral de preferencia, umbral de indiferencia y veto). Un ejemplo de su definición es el siguiente, *w_DMs* = {0.25, 0.25, 0.25, 0.25}, *q_DMs* = {10, 15, 14, 7}, *p_DMs* = {15, 20, 18, 10}, y *v_DMs* = {20, 30, 25, 28}. Se hace notar que la suma de los pesos deberá ser 1.

4.3.3 Incorporación de preferencias del DM

Un tema importante dentro de este trabajo además de implementar un algoritmo dentro de un Hardware como Arduino, es el proponer un modelo de preferencias que se asemeje a las preferencias de un DM real, con el propósito de obtener las mejores soluciones. El modelo propuesto se basa de un cálculo de credibilidad, en concreto un método de ELECTRE III que afirma que una solución x es al menos tan buena como y, y también de unos umbrales que servirán como las preferencias del DM (véase Anexo A para su definición). Por último, este índice de credibilidad nos servirá para comparar un conjunto de reglas propuesto en este trabajo donde cada regla establece que la solución es satisfactoria para el DM y ser elegida como el mejor compromiso.

A continuación, se detalla el sistema relacional de sobre clasificación propuesto

La definición de una posible relación de preferencia entre un par de soluciones depende de tres parámetros y del grado de credibilidad propuesto por Bernnard Roy (1996), teniendo el cálculo de $\sigma(x, y)$, pero para este trabajo se ocupa otro parámetro para determinar una relación de sobre clasificación que logran proporcionar una mayor capacidad discriminatoria y el parámetro es el siguiente:

Umbral	Descripción
$\boldsymbol{\beta}$	Umbral de credibilidad, asegura que la solución es estrictamente preferida por el DM.

Tabla 8. Umbrales de sobre clasificación

Como ya se describió previamente el modelo de preferencias para que una solución sea el mejor compromiso se debe cumplir al menos una de las siguientes condiciones:

- **Variante 1**.
  - $\boldsymbol{\sigma(x, y) > \sigma(y, x)}$
- **Variante 2**.

- $\sigma(x, y) \geq \beta$
- **Variante 3**.
  - $\sigma(x, y) \geq \beta$
  - $\sigma(y, x) < \beta$
- **Variante 4**.
  - $\sigma(x, y) > \sigma(y, x)$
  - $\sigma(x, y) > 0.5$
- **Variante 5**.
  - $\sigma(x, y) \geq \beta$
  - $\sigma(y, x) < 0.5$

4.3.4 Algoritmos de optimización: MOEA/D y PMOEA/D

El módulo principal del agente es la incorporación de dos algoritmos de optimización como lo son PMOEA/D y MOEA/D, como nota a destacar tenemos que la implementación cuenta con una forma de poder utilizar uno de los dos algoritmos dentro del mismo agente a través de una bandera situada en la incorporación de preferencias. La descripción de MOEA/D se muestra en la sección 2.3.1. El resto de este apartado detalla cómo se diseñó e implementó el algoritmo propuesto PMOEA/D. Cabe mencionar que las soluciones obtenidas son de igual forma mostradas en el Serial de Arduino por lo que también son copiadas a un documento de texto.

Algoritmo de optimización PMOEA/D

Algorithm 3. **PMOEA/D**

Input:
N= número de funciones escalares.
K= número de objetivos, (K=4 en *Problema* (PSP))
Vector= conjunto de vectores distribuidos uniformemente.
T=N/2, tamaño del vecindario de los vectores de peso.
maxEvaluaciones=número máximo de evaluaciones que tendrá el algoritmo
w_i= Representa el umbral de peso.
q_i=Representa el umbral de indiferencia.
p_i=Representa el umbral de preferencia. Donde $i = \{1,2, \dots, K\}$
v_i=Representa el umbral de veto.
OM = Modelo de superación (outranking model)

Output:

1. *EP* = Aproximación de la Frontera de Pareto.
2. (*x*,*z*,*FV*,*B*(*i*)) ← **Inicialización**()
3. do
4. **For** *i* = 1 to *N* do
5. (*xk*, *xl*) ← **SelecciónAleatoria**(*B*(*i*),*T*)
6. *y* ← **CruzaSBX**(*xk*, *xl*)
7. *y*' ← **MutaciónPolinomial**(*y*)
8. *y*'' ← **OperadorReparaciónMejora** (*y*')
9. **ActualizarConjuntoZ**(*z*,*P*,*y*'')

10. **ActualizarVecindario**(*B*(*i*),*FV*,*y*'',*OM*)
11. **ActualizarEP**(*EP*,*y*'')
12. **While** *maxEvaluaciones* no sea alcanzado

Figura 7. Algoritmo PMOEA/D

El algoritmo implementado es una variante de MOEA/D el cual incluye preferencias de manera a priori mediante ELECTRE III como se muestra en la Figura 7, donde la salida será una población elite usada dentro del proceso de optimización de soluciones no dominadas y estrictamente preferidas.

Los parámetros de entrada del algoritmo son el número de funciones escalares N, el número de objetivos K, el conjunto inicial de vectores de peso $Vector = \{V_1, V_2, \dots, V_N\}$, y el tamaño del vecindario de vector de pesos T.

Como primer paso es la fase de inicialización (línea 1 en la Figura 7). El algoritmo inicializa las estructuras de datos necesarios para la construcción del conjunto final de soluciones (EP).

Primero el conjunto EP se convertirá en el conjunto de soluciones que se aproximan a la Frontera de Pareto y a su vez solo estará el subconjunto de soluciones dentro de la RoI, y que inicialmente está vacío.

Segundo los conjuntos de vecindad $B(i)$ de cada $Vector\ V_i$ que contienen los T vectores de peso más cercanos a V_i por distancia euclidiana.

Tercero el conjunto inicial de soluciones $x = \{x_1, x_2, \dots, x_N\}$ donde cada solución $x_j, 1 \leq j \leq n\acute{u}mero\ del\ grupo\ de\ miembros$, el cual se inicializa de manera aleatoria que corresponde a un vector binario $x = \{1,0\}$ que sea aceptable contemplando las restricciones.

Cuarto el conjunto de valores de aptitud $FV = \{FV_1, FV_2, \dots, FV_N\}$, donde cada FV_i contiene los K valores objetivos de cada solución x_i.

Quinto el conjunto $z = \{z_1, z_2, \dots, z_k\}$ formado por valores z_j correspondientes al mejor valor objetivo entre todas las soluciones construidas durante el proceso de inicialización.

Como segundo paso es un proceso de actualización basado en la evolución. En este proceso, para cada solución en la población, se seleccionan aleatoriamente dos soluciones de la población y se utilizan para generar una nueva solución mediante la aplicación de operadores genéticos. Los operadores utilizados son los siguiente:

1. Cruza SBX (Simulated Binary Crossover): Se escogen dos padres elegidos al azar, se inicializa un numero aleatorio r entre $\{0.0,1.0\}$, Si $r \leq 0.5$ se determina β' por cada hijo mediante una ecuación (véase en el punto 5.4.3.2). Si $r > 0.5$ solo se le asigna el valor de y_2 a c_1 y y_1 a c_2 donde y_i es la solución padre. Una vez obteniendo el valor β' se asigna al primer hijo el primer limite como $c_1 = 0.5 * \big((y_1 + y_2) - \beta' * (y_2 - y_1)\big)$ y el segundo

hijo con $c_2 = 0.5 * \big((y_1 + y_2) + \beta' * (y_2 - y_1)\big)$. Si el hijo $c_1 < y_L$ entonces se asigna $c_1 = y_L$, Si $c_2 < y_L$ entonces $c_2 = y_L$ donde y_L es el límite inferior, Si $c_1 > y_U$ entonces $c_1 = y_U$, Si $c_2 > y_U$ entonces $c_2 = y_U$ donde y_U es el límite superior. Y los resultados de esta cruza darán como decendencia dos hijos.

2. Mutación Polinomial: En este proceso genera un valor aleatorio para cada alelo en la cadena de bits que codifica la solución según Liagkouras and Tetaxiotis (2013), se escoge un valor aleatorio $r = [0,1]$. Si $r \leq probabilidad\_de\_muta$ entonces se asigna δ_1 y δ_2 donde $\delta_1 = \frac{x_p - x_l}{x_u - x_l}$ $\delta_2 = \frac{x_u - x_p}{x_u - x_l}$ que será delta_1 y delta_2 respectivamente. Se vuelve a asignar un valor aleatorio $r = [0,1]$. Si $r \leq 0.5$ entonces se hace el cálculo para obtener delta_q que será igual a $d_q = [2r + (1 - 2r)(1 - d_1)^{\eta_m + 1}]^{\frac{1}{\eta_m + 1}}$ sino $d_q = 1 - [2(1 - r) + 2(r - 0.5)(1 - d_2)^{\eta_m + 1}]^{\frac{1}{\eta_m + 1}}$.

Después de aplicar los operadores genéticos, la solución para a la fase de reparación y mejora que es realizada por el Operador de Reparación y Mejora. Este paso se puede proponer una heurística para poder reparar o definitivamente mejorar una solución, es decir, convertir una solución inviable en una solución casi factible, basándose en que una cartera es infactible si sobrepasa el presupuesto, por lo cual al repararlo puede serle factible en presupuesto, pero no factible para el DM.

Luego se hace la actualización del Conjunto Z que contiene los mejores valores para cada objetivo, donde para cada $j = \{1,2, \dots, K\}$ $Si\ z_j < f_j(y')$ entonces $z_j = f_j(y')$.

La inclusión de preferencias se incorpora dentro de la actualización del vecindario, en el algoritmo MOEA/D se tiene que **Para cada** *j* ϵ *B*(*i*), **si** $\big[\, gte\ (y' | Vj, z) \leq gte\ \big(x^j | V^j, z\big)$ && $\text{EvaluarOM}\big(y', x^j, \sigma\ \big)\big]$, entonces el conjunto $x^j = y'$ and $FV^j = F(y')$.

El último paso es verificar la condición de parada, para que el algoritmo se finalice y dé la salida el conjunto EP (conjunto que tendrá las soluciones no dominadas y estrictamente preferidas) debe de alcanzar un máximo de evaluaciones, de lo contrario volverá al segundo paso (línea 3). Durante cada iteración, el algoritmo actualiza B, EP, z.

Los detalles de los métodos y funciones especiales usados en la implementación son los siguientes en detallarse.

Inicialización de los parámetros del Algoritmo

Los parámetros de entrada del algoritmo son el número de funciones escalares $N = 7$, el número de objetivos $K = 4$, el conjunto inicial de vectores de peso $Vector = \{V_1, V_2, \dots, V_N\}$, y el tamaño del vecindario de vector de pesos $T = 3$.

Se inicializa los vectores de peso como se muestra en la Tabla 9.

Vectores de peso				
$\boldsymbol{N}$	V_1	V_2	V_3	V_4
1	0.3	0.3	0.2	0.2
2	0.1	0.25	0.41	0.24
3	0.12	0.32	0.21	0.35
4	0.15	0.21	0.51	0.13
5	0.11	0.35	0.18	0.36
6	0.05	0.12	0.61	0.22
7	0.12	0.03	0.30	0.55

Tabla 9. Vectores de peso

Segundo, los conjuntos de vecindad $B(i)$ de cada $Vector\ V_i$ que contienen los T vectores de peso más cercanos a V_i por distancia euclidiana. La fórmula para el cálculo de la distancia euclidiana es la siguiente, y el resultado para los vectores de peso seleccionados se muestra en la Tabla 10.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

Donde p y q son los vectores de donde se busca obtener la distancia.

$\boldsymbol{B(i)}$	$\boldsymbol{B(1)}$	$\boldsymbol{B(2)}$	$\boldsymbol{B(3)}$	$\boldsymbol{B(4)}$	$\boldsymbol{B(5)}$	$\boldsymbol{B(6)}$	$\boldsymbol{B(7)}$
1	0	0.29698485	0.23537205	0.36276714	0.2541653	0.5132251	0.48764741
2	0.29698485	0	0.23958297	0.16186414	0.27820855	0.24454039	0.39623226
3	0.23537205	0.23958297	0	0.38910153	0.04472136	0.47095647	0.36359318
4	0.36276714	0.16186414	0.38910153	0	0.42778499	0.19026298	0.50378567
5	0.2541653	0.27820855	0.04472136	0.42778499	0	0.51088159	0.39115214
6	0.5132251	0.24454039	0.47095647	0.19026298	0.51088159	0	0.4669047
7	0.48764741	0.39623226	0.36359318	0.50378567	0.39115214	0.4669047	0

Tabla 10. Distancia Euclidiana de cada vector de peso desordenado

Se asocian los $B(i)$ vectores de peso mediante distancia euclidiana como se muestra en la Tabla 10, se muestran de manera desordenada por lo que se continua con el ordenamiento de manera ascendiente como se muestra en la Tabla 11.

$B(i)$	$B(1)$		$B(i)$	$B(2)$		$B(i)$	$B(3)$		$B(i)$	$B(4)$
3	0.23537205		**4**	0.16186414		**5**	0.04472136		**2**	0.16186414
5	0.2541653		**3**	0.23958297		**1**	0.23537205		**6**	0.19026298
2	0.29698485		**6**	0.24454039		**2**	0.23958297		**1**	0.36276714
4	0.36276714		**5**	0.27820855		**7**	0.36359318		**3**	0.38910153
7	0.48764741		**1**	0.29698485		**4**	0.38910153		**5**	0.42778499
6	0.5132251		**7**	0.39623226		**6**	0.47095647		**7**	0.50378567

a)

$B(i)$	$B(5)$		$B(i)$	$B(6)$		$B(i)$	$B(7)$
3	0.04472136		**4**	0.19026298		**3**	0.36359318
1	0.2541653		**2**	0.24454039		**5**	0.39115214
2	0.27820855		**7**	0.4669047		**2**	0.39623226
7	0.39115214		**3**	0.47095647		**6**	0.4669047
4	0.42778499		**5**	0.51088159		**1**	0.48764741
6	0.51088159		**1**	0.5132251		**4**	0.50378567

b)

Tabla 11. Distancia euclidiana para cada vector de peso ordenado

Se ordenan los vectores y se continúa obteniendo solamente los T vectores más cercanos en $B(i)$, donde $T = 3$ para este ejemplo, y el resultado se muestra en la Tabla 12.

$B(i)$	$B(1)$		$B(i)$	$B(2)$		$B(i)$	$B(3)$		$B(i)$	$B(4)$
3	0.23537205		**4**	0.16186414		**5**	0.04472136		**2**	0.16186414
5	0.2541653		**3**	0.23958297		**1**	0.23537205		**6**	0.19026298
2	0.29698485		**6**	0.24454039		**2**	0.23958297		**1**	0.36276714

$B(i)$	$B(5)$		$B(i)$	$B(6)$		$B(i)$	$B(7)$
3	0.04472136		**4**	0.19026298		**3**	0.36359318
1	0.2541653		**2**	0.24454039		**5**	0.39115214
2	0.27820855		**7**	0.4669047		**2**	0.39623226

Tabla 12. Los T vectores peso más cercanos en B(i)

Una vez teniendo los vecindarios con los T vecinos más cercanos se continua con la creación de la población inicial.

Función de inicialización de la población

Para la inicialización de la población se tiene de ejemplo que existen m alternativas (proyectos), los cuales son datos tomados de los sensores $K = \{h, t, g, d\}$, para este ejemplo $m = 7$ como en la siguiente tabla se representa:

Sensores		h	t	g	d	Costo
Valores	Alt 1	67	27.9	110	47	10
	Alt 2	66	25.2	112	15	15
	Alt 3	66	28.0	115	12	20
	Alt 4	70	29.4	90	30	12
	Alt 5	88	35.0	95	18	6
	Alt 6	87	20.0	110	20	30
	Alt 7	63	29.9	417	83	18

Tabla 13. Alternativas por cada sensor dentro del problema PSP

Donde cada alternativa representa un conjunto de sensores y cada valor tomado representa el objetivo tanto a maximizar o minimizar. Como es el problema de selección de carteras, cada cartera representa un conjunto de alternativas donde $x = (x_1, \dots, x_m)^T \in \{0,1\}^m$ donde la probabilidad de $x_i = 1$ es igual a 0.5. Para la construcción de inicialización de la población es la repetir hasta que se obtenga los $\boldsymbol{l}$ individuos de la población, donde $\boldsymbol{l} = 5$ para este problema de ejemplo.

Para cada cartera que representa una fila se tiene un vector donde cada columna representa cada alternativa, como se muestra en la Tabla 14.

	Alternativas	$A1$	$A2$	$A3$	$A4$	$A5$	$A6$	$A7$
Carteras	x_1	1	1	1	0	0	1	0
	x_2	1	1	1	1	0	0	0
	x_3	0	1	1	1	0	0	0
	x_4	1	1	0	1	0	1	0
	x_5	1	1	0	0	1	1	0

Tabla 14. Construcción de la población

Cartera	Índice Alternativas	Obj1	Obj2	Obj3	Obj4	Costo
x_1	1,2,3,6	286	106.6	447	94	75
x_2	1,2,3,4	269	110.5	427	104	57
x_3	2,3,4	202	82.6	317	57	47
x_4	1,2,4,6	290	102.5	422	112	67
x_5	1,2,5,6,7	371	138	844	183	79

Tabla 15. Descripción de las carteras y el conjunto de alternativas

En la Tabla 15, se describe el conjunto de 5 carteras que son las soluciones iniciales con las que se cuenta al inicio del algoritmo, por cada objetivo se hace la sumatoria de cada alternativa que compone una cartera.

Inicializar los vectores Z_{min} y Z_{max}

Para el proceso de inicializar el vector Z mínimo, es tomar por cada solución $\{x_1, \dots, x_n\}$ el objetivo K-esimo de dicha solución y compararlo si es el mínimo en esa posición, como se muestra en la Tabla 16.

	z_1	z_2	z_3	z_4
Z_{min}	202	82.6	317	94

Tabla 16. Vector Z con los valores mínimo de cada solución

Para el vector Z máximo se compara la solución en el K-esimo objetivo si es el máximo en esa posición, como se muestra en la Tabla 17.

	z_1	z_2	z_3	z_4
Z_{max}	371	138	844	183

Tabla 17. Vector Z con los valores máximos de cada solución

Operadores Genético 1: Método de Selección Aleatoria

Para ejercer la selección, se toman las soluciones creadas antes y se eligen dos padres de manera aleatoria considerando de los vecindarios $B(i)$ y los T vecinos más cercanos, por ejemplo, lo mostrado en la Tabla 18.

$B(i)$	$B(1)$
3	0.23537205
5	0.2541653
2	0.29698485

Tabla 18. Vecindario B(i) con los T vecinos más cercanos

Tomando el primer vecindario se escoge aleatoriamente la posición 0 a T soluciones donde $k\ y\ l$ serán las soluciones padres, para este ejercicio $k = 0$ y $l = 2$ por lo tanto el resultado queda como en la Tabla 19.

	Solución	Cromosoma						
x_k	3	0	1	1	1	0	0	0
x_l	2	1	1	1	1	0	0	0

Tabla 19. Cromosomas de dos soluciones aleatorias

Operador Genético 2: Método de Cruza SBX

Para la cruza se establecen dos valores, los cuales son la probabilidad de cruza y el índice de distribución, para este ejercicio $crossoverProbability = 1.0$ y *indexDistribution* = 15.

Se toma un valor aleatorio entre 0.0 y 1.0, si el valor es menor o igual a $crossoverProbability$ se toma el primer valor de las variables de los dos padres, ya que la probabilidad es de 1.0 y siempre entrará la cruza.

	Cromosoma						
$\boldsymbol{padre_1}$	0	1	1	1	0	0	0
$\boldsymbol{padre_2}$	1	1	1	1	0	0	0

Después se toma un nuevo valor aleatorio entre 0.0 y 1.0, si el valor es mayor a 0.5 el primer hijo toma el valor del $padre\_2$ y el segundo hijo toma el valor del $padre\_1$.

	Cromosoma						
$\boldsymbol{y_1}$	1						
$\boldsymbol{y_2}$	0						

Y se continúa con el siguiente índice de la variable.

	Cromosoma						
$\boldsymbol{padre_1}$	0	1	1	1	0	0	0
$\boldsymbol{padre_2}$	1	1	1	1	0	0	0

Ahora se toman los siguientes dos índices con las siguientes variables de los dos padres y se vuelve a tomar un valor aleatorio entre 0.0 y 1.0, si el valor no es mayor a 0.5 sino es que es menor o igual a 0.5 entra al proceso SBX, primero se obtienen los límites inferior y superior. Para el límite inferior $Lower = 0$ y para el límite superior $Upper = 1$. Se escoge un valor mínimo que se toma al restar $\min = padre_1 - Lower$ o $\min = Upper - padre_2$.

Se calcula el valor de $\beta = 1 + \left(\frac{2*min}{padre_2 - padre_1}\right)$ y el valor de $\alpha = 2 - \beta^{-1*(15+1)}$.

Para $\beta = 1$ y $\alpha = 1$, se obtiene un nuevo valor aleatorio u entre 0.0 y 1.0, si es menor o igual a $\frac{1.0}{\alpha}$, para este ejemplo $u = 0.3$, si se cumple $\beta_q = u * \alpha^{\frac{1.0}{15+1}}$, sino se cumple entonces si $u * \alpha \neq 2.0$ entonces $\beta_q = \frac{1.0}{2 - u*\alpha}^{\frac{1.0}{15+1}}$, para este punto se cumple esta última formula entonces $B_q = 0.9043038$. Obteniendo β_q se procede a dos últimas formulas, donde $c_1 = 0.5 * \left((padre_1 + padre_2) - \beta_q(padre_2 - padre_1)\right)$ y $c_2 = \left((padre_1 + padre_2) + \beta_q(padre_2 - padre_1)\right)$. Por lo tanto $c_1 = 1$ y $c_2 = 1$

Si $c_1 < Lower$ c_1 toma el valor de $Lower$ sino $c_1 > Upper$, c_1 toma el valor de $Upper$ y lo mismo para c_2. Por ejemplo $1 < 0$ por lo que $c_1 = 1$ y $c_2 = 1$. Por último, se procede a tomar un valor aleatorio rnd para la asignación del valor de la variable en el bit de cada hijo. Si el valor rnd es menor o igual a 0.5 el y_1 toma el valor del c_2 y el y_2 toma el valor del c_1, en caso contrario el y_1 toma el valor del c_1 y el y_2 toma el valor del c_2. El valor $rnd = 0.2$ por lo que así quedan los hijos en el segundo bit:

	Cromosoma						
y_1	1	1					
y_2	0	1					

Y así sucesivamente se procede por todo el tamaño del cromosoma y utilizando el método SBX para cruzar cada bit de los dos padres y crear dos nuevos hijos. Terminando el proceso de cruza quedan estos dos nuevos hijos:

	Cromosoma						
y_1	1	1	1	1	0	0	0
y_2	0	1	1	1	0	0	0

Operador Genético 3: Método de Mutación Polinomial

Pasando el proceso de cruza esas dos nuevas soluciones hijas pasan al proceso de mutación. Primero se toma el y_1 donde la probabilidad que se haga mutación en el bit actual por cada hijo es de $mutatioProbability = \frac{1.0}{\text{NUM\_PROYECTs}}$, por lo que $mutationProbability = 0.14$. Se toma un valor aleatorio entre 0.0 1.0. En este caso $rnd = 0.3$. Si $rnd \leq mutationProbability$ entra al proceso de mutación, ya que $0.3 \leq 0.14$ entonces para el y_1 en el primer bit no cambia.

	Cromosoma						
y_1	1	1	1	1	0	0	0

Se procede al siguiente bit del cromosoma del y_1 volviendo a obtener otro valor aleatorio, donde $rnd = 0.08$ y se comprueba de nuevo $0.08 \leq 0.14$, como es verdadero entra al proceso de mutación polinomial. Se toma el límite inferior y superior, donde $Lower = 0$ y $Upper = 1$. Primero se comprueba si $Lower = Upper$, si fuera verdadero y_1 en el segundo bit tomaría el valor de $Lower$, en este caso 0, si $Lower \neq Upper$ entonces se obtiene el valor $min = y_1 - Lower$, por lo que $min = 1 - 0$, $min = 1$ y se vuelve a validar si $Upper - y_1 < min$, si es verdadero $min = Upper - y_1$ sino min continua siendo 1, en este caso $min = 0$. Una vez obteniendo el mínimo se procede a obtener δ y δ_q. Donde $\delta = \frac{min}{Upper - Lower}$, entonces $\delta = 0$ y para obtener δ_q

se obtiene un valor aleatorio entre 0.0 y 1.0, asignada a la variable rnd . Si $rnd \leq 0.5$ $\delta_q = (2 * rnd + (1 - 2 * rnd) * (1 - \delta)^{20+1})^{\frac{1.0}{(20+1)}-1}$, sino entonces se utiliza esta otra fórmula para el cálculo de $\delta_q = (2 * (1 - rnd) + 2 * (rnd - 0.5) * (1 - \delta)^{20+1})^{1-\frac{1.0}{(20+1)}}$. Por lo que el valor de $rnd = 0.4$, por lo que se aplica la primera formula, entonces $\delta_q = 1$. Una vez obteniendo δ_q se procede a hacer un último cálculo donde $c = \delta_q * (Upper - Lower)$, entonces $c = 1 * (1 - 0)$, entonces $c = 1$, por lo que $y_1 = c$, por lo que:

	Cromosoma						
y_1	1	1	1	1	0	0	0

El y_1 en el segundo bit queda con el mismo valor de 1 y así se continúa para los restantes bits del cromosoma, tanto del y_1 como del y_2 y quedarían los cromosomas como los siguientes:

	Cromosoma						
y_1	1	1	1	1	1	0	0
y_2	0	1	1	1	0	0	1

El proceso de mutación involucra un índice de distribución con valor 20.

Método de Reparación o Mejora

Para el método de Reparación o Mejora dentro del algoritmo, se propone normalmente una heurística que pueda reparar o mejorar las soluciones hijas previamente creadas, pero tomando en cuenta la memoria dinámica con la que cuentan las tarjetas de Arduino, se vuelve una tarea más pesada para el microcontrolador, por lo que se optó el no incluir una heurística de reparación o mejora dentro de este algoritmo.

Actualizar los conjuntos Z_{min} y Z_{max}

Para la actualización del conjunto Z_{min} es un vector que contiene los mejores valores (mínimo) por cada objetivo, por lo que el conjunto Z_{min} es:

	z_1	z_2	z_3	z_4
Z_{min}	202	82.6	317	94

Para la actualización del conjunto Z_{max} es un vector que contiene los mejores valores (máximos) por cada objetivo, por lo que el conjunto Z_{max} es:

	z_1	z_2	z_3	z_4
Z_{max}	371	138	844	183

Por cada solución hija creada (2) se va a comparar desde $z_1 \ldots, z_n$ y se compara la primera solución y_1:

	Obj_1	Obj_2	Obj_3	Obj_4
y_1	357	145.5	522	122

	z_1	z_2	z_3	z_4
Z_{min}	202	82.6	317	94

Se hace una comparación por cada objetivo si $obj_n < z_n$ entonces $z_n = Obj_n$ y así sucesivamente por cada objetivo. Se puede ver que ningún objetivo cumple la condición por lo cual el vector Z_{min} continúa con los mismos valores y se continúa con la comparación de Z_{max}:

	Obj_1	Obj_2	Obj_3	Obj_4
y_1	357	145.5	522	122

	z_1	z_2	z_3	z_4
Z_{max}	371	138	844	183

Se comprueba que en el obj_2 del y_1 es mayor a Z_{max} en posición z_2 por lo que el vector quedaría

	Obj_1	Obj_2	Obj_3	Obj_4
Z_{max}	371	145.5	844	183

Se prosigue con el segundo hijo y se vuelve a comparar contra los vectores Z_{min} y Z_{max}

	Obj_1	Obj_2	Obj_3	Obj_4
y_2	265	112.5	734	140

	z_1	z_2	z_3	z_4
Z_{min}	202	82.6	317	94

No existe un valor menor al vector Z_{min} y se procede a compararlo contra el vector Z_{max}

	Obj_1	Obj_2	Obj_3	Obj_4
y_2	265	112.5	734	140

	Obj_1	Obj_2	Obj_3	Obj_4
Z_{max}	371	145.5	844	183

No existe tampoco un cambio en el vector Z_{max} por lo que se continúa con el proceso del algoritmo.

Actualizar el Vecindario

Para el cálculo de actualizar el vecindario, es un proceso donde se trata de comparar las soluciones hijas contra las soluciones padres del vecindario y comprobar si existe una mejora en base a tres conceptos importantes dentro de este trabajo, los cuales son el cálculo de Tchebycheff, el cálculo de credibilidad mediante ELECTRE III y un conjunto de reglas basadas en la sobre clasificación que se explicaran a continuación.

Cálculo de Tchebycheff

Según Zhang and Li (2007) el cálculo de Tchebycheff por cada índice $j \in B(i)$ si $g^{te}(y'|\lambda^j, z) \leq g^{te}(x^j|\lambda^j, z)$ entonces el conjunto $x^j = y'$. Por lo que desglosando el cálculo sería de la siguiente manera:

$$\text{minimize } g^{te}(x|\lambda, z^*) = \max_{1 \leq i \leq m} \{\lambda_i | f_i(x) - z_i *|\}$$

En pocas palabras donde el i-esimo objetivo de la j-ésima solución sea peor (o sea, más alejado del punto óptimo de referencia z) dicha solución será remplazada por la nueva solución.

Tomando en cuenta el proceso anterior se tienen las dos soluciones hijas (y_1 e y_2), se procede a hacer el cálculo como se muestra a continuación:

	Obj_1	Obj_2	Obj_3	Obj_4
y_1	357	145.5	522	122

	Obj_1	Obj_2	Obj_3	Obj_4
y_2	265	112.5	734	140

Recordemos que el vecindario contiene las siguientes soluciones:

$B(i)$	$B(1)$
3	0.23537205
5	0.2541653
2	0.29698485

Se hace la comparación de las dos soluciones hijas por cada una del vecindario, por lo que se procede a hacer el cálculo como se muestra a continuación:

	Obj_1	Obj_2	Obj_3	Obj_4
y_1	357	145.5	522	122

	Obj_1	Obj_2	Obj_3	Obj_4
x_3	202	83	317	57

Se procede a tomar los vectores Z_{min} y Z_{max}

	z_1	z_2	z_3	z_4
Z_{min}	202	82.6	317	94

	Obj_1	Obj_2	Obj_3	Obj_4
Z_{max}	371	145.5	844	183

Y se procede a calcular g^{te} por cada solución del vecindario contra las dos soluciones hijas. De manera explícita se va a colocar la tabla con los valores obtenidos de g^{te} por cada vecino y por las dos soluciones hijas.

Solución	g^{te}
x_3	0.0978513
x_5	0.2541653
x_2	0.13199327
y_1	0.23537205
y_2	0.18624316

Se procede a hacer la comparación de las dos nuevas soluciones hijas (y_1 y y_2) contra las soluciones del vecindario (x_3, x_5, x_2), se tendrá una variable de tipo booleana que guardará verdadero o falso, donde solo será verdadero si las nuevas soluciones superan a una de las soluciones del vecindario como se muestra en la siguiente tabla:

Solución	y_1
x_3	Falso
x_5	Verdadero
x_2	Falso

Teniendo en cuanta que la solución x_5 es peor que la solución y_1 se acepta que esta solución tiene este criterio de aceptación mediante Tchebycheff, pero cabe recalcar que este trabajo además de este cálculo, las nuevas soluciones deben de ser aceptadas del modelo de preferencias propuesto como se muestra en el siguiente punto. Recordar que al ser un proceso iterativo primero debe hacerse el cálculo con la primera solución hija, si esta no es aceptada mediante Tchebycheff, el modelo de preferencias se descarta y se continua con la siguiente solución hija.

Cálculo de credibilidad mediante ELECTRE III

Primero se utiliza un método para el cálculo de credibilidad, en concreto un método de ELECTRE III (véase en el Anexo A) que nos dice que una solución tal que x es al menos tan buena como y mediante unos umbrales que determinan las preferencias de un DM respecto al mejor compromiso.

Tomaremos los vectores con los umbrales dados por el DM para este ejemplo, como se muestra a continuación:

Umbrales	Objetivo 1	Objetivo 2	Objetivo 3	Objetivo 4
w	0.25	0.25	0.25	0.25
q	10	15	14	7
p	15	20	18	10
v	20	30	25	28

Tabla 20. Umbrales de preferencia del DM.

Tomamos la solución y_1 ya que fue aceptada mediante Tchebycheff previamente contra la solución del vecindario x_5. Para el cálculo de credibilidad se utiliza la siguiente fórmula:

$$\sigma(x, y) = c(x, y)^* \, d(x, y)$$

Donde el índice de credibilidad es igual a la multiplicación de un índice de concordancia por un índice de discordancia. Por lo tanto, comenzaremos desglosando dicha fórmula para saber cómo calcular ambos índices comenzando con el cálculo de concordancia:

El índice de concordancia puede ser calculado como:

$$c(x,y) = \sum_{k=1}^{n} c_k(x,y)$$

Donde

$$c_k = \begin{cases} w_k & si\ xP_k y \vee xI_k y, \\ 0 & en\ otro\ caso \end{cases}$$

Donde $xP_k y$ y $xI_k y$ son las funciones lógicas de preferencia e indiferencia en él k-ésimo objetivo. La preferencia puede ser definida como:

$$xP_k y = f_k(x) > f_k(y) \wedge \neg xI_k y$$

Donde f_k es la función de evaluación para el k-ésimo objetivo. La indiferencia es definida como:

$$xI_k y = |f_k(x) - f_k(y)| \leq q_k$$

Tomando como ejemplo una de las soluciones hijas que se han trabajado a lo largo del proceso algorítmico, se propondrá una solución de ejemplo a la cual comparar simulando que ésta ya se encuentra dentro de las soluciones aceptadas por el DM.

Se tendrá la solución y_1 como la alternativa a comparar y se propone la solución x_1 ficticia (véase en el anexo A un ejercicio más completo).

	$\boldsymbol{Obj_1}$	$\boldsymbol{Obj_2}$	$\boldsymbol{Obj_3}$	$\boldsymbol{Obj_4}$
$\boldsymbol{y_1}$	357	146	522	122

	$\boldsymbol{Obj_1}$	$\boldsymbol{Obj_2}$	$\boldsymbol{Obj_3}$	$\boldsymbol{Obj_4}$
$\boldsymbol{x_1}$	202	150	317	160

A continuación, se procede a calcular el índice de concordancia $c(y_1, x_1)$ quedando de la siguiente manera:

$$c(y_1,x_1) = c_1(y_1,x_1) + c_2(y_1,x_1) + c_3(y_1,x_1) + c_4(y_1,x_1) = 0.25 + 0 + 0.25 + 0 = 0.5$$

Y similarmente puede calcularse $c(x_1, y_1)$ y quedaría de la siguiente manera:

$$c(x_1,y_1) = c_1(x_1,y_1) + c_2(x_1,y_1) + c_3(x_1,y_1) + c_4(x_1,y_1) = 0 + 0 + 0 + 0.25 = 0.25$$

Ahora se puede arreglar en forma de matriz quedando de la siguiente forma:

$$c(x,y) = \begin{bmatrix} 1.0 & 0.5 \\ 0.25 & 1.0 \end{bmatrix}$$

Una vez que se tiene los índices de concordancia se procede a calcularse el índice de discordancia y puede definirse de la siguiente manera:

$$d(x,y) = \min_{k\in\{1,2,3,\ldots,n\}}\{1 - d_k(x,y)\}$$

Donde

$$d_k(x,y) = \begin{cases} 0 & \text{Si } \nabla_k(x,y) < u_k, \\ \dfrac{\nabla_k(x,y) - q_k}{v_k - q_k} & \text{Si } u_k \le \nabla_k(x,y) < v_k \\ 1 & \text{Si } \nabla_k(x,y) \ge v_k, \end{cases}$$

Donde $\nabla_k(x,y) = f_k(y) - f_k(x)$

Se procede a calcular $d(y_1, x_1)$:

$$d(y_1,x_1) = \min\{1 - d_1(y_1,x_1), 1 - d_2(y_1,x_1), 1 - d_3(y_1,x_1), 1 - d_4(y_1,x_1)\}$$
$$= \{1 - 0{,}1 - 0{,}1 - 0{,}1 - 1\} = \{1{,}1{,}1{,}0\} = 0$$

Donde $\nabla_1(y_1,x_1) = -155, \nabla_2(y_1,x_1) = 4, \nabla_3(y_1,x_1) = -205, \nabla_4(y_1,x_1) = 38$

De igual manera se procede a calcular $d(x_1, y_1)$:

$$d(x_1,y_1) = \min\{1 - d_1(x_1,y_1), 1 - d_2(x_1,y_1), 1 - d_3(x_1,y_1), 1 - d_4(x_1,y_1)\}$$
$$= \{1 - 1{,}1 - 0{,}1 - 1{,}1 - 0\} = \{0{,}1{,}0{,}1\} = 0$$

Donde $\nabla_1(x_1,y_1) = 155, \nabla_2(x_1,y_1) = -4, \nabla_3(x_1,y_1) = 205, \nabla_4(x_1,y_1) = -38$

Ahora se puede arreglar en forma de matriz quedando de la siguiente forma:

$$d(x,y) = \begin{bmatrix} 1.0 & 0 \\ 0 & 1.0 \end{bmatrix}$$

Solo falta realizarse el cálculo de credibilidad $\sigma(x,y) = c(x,y) * d(x,y)$ quedando de la siguiente manera:

$$\sigma(x,y) = \begin{bmatrix} 1.0 & 0 \\ 0 & 1.0 \end{bmatrix}$$

De esta manera el índice de credibilidad no solo da un valor binario de verdad o falso respecto a la afirmación de "x es al menos tan buena que y" sino, que da un valor difuso que representa que

tan bueno es. Para este ejemplo con el índice, nos dice que no hay una mejora de la solución y_1 con respecto a x_1, pero una vez teniendo este índice se procede a aplicarse el modelo propuesto de sobre clasificación para discriminar aún más las soluciones mediante las preferencias del DM.

Modelo de sobre clasificación y Actualizar el conjunto EP

Como nota a tomar en cuenta, ya que el índice de credibilidad de la solución ficticia propuesta "x contra y" y "y contra x", ambos índices dieron 0 se procederá a colocar un índice de credibilidad ficticio propuesto para que se puede ejemplificar mejor el proceso de aplicar el modelo de clasificación, por lo que los nuevos índices serán los siguientes:

$$\sigma(x, y) = \begin{bmatrix} 1.0 & 0.347 \\ 0.289 & 1.0 \end{bmatrix}$$

Una vez teniendo el índice de credibilidad se procede a aplicar el modelo de sobre clasificación propuesto con $\beta = 0.67$ para este trabajo.

Dentro de este trabajo se consideran 5 variantes (Véase el punto 5.3.2) considerando un modelo de sobre clasificación para este trabajo, por cada regla se procederá a simular cómo se hace la comparación de los índices de credibilidad con cada una de las reglas como se muestra a continuación:

La primera variante afirma que el índice de credibilidad de x sobre y debe de ser mayor al índice de credibilidad de y sobre x para cumplirse:

$$\sigma(x, y) > \sigma(y, x) = 0.347 > 0.289$$

Se comprueba que es VERDADERO la primera variante así que se procede con la siguiente regla. Nota: se continúan de todas maneras las comparaciones contra las siguientes reglas restantes.

La segunda variante afirma que el índice de credibilidad de x sobre y debe de ser mayor o igual a β para poder cumplirse:

$$\sigma(x, y) \geq \beta = 0.347 \geq 0.67$$

Se comprueba que es FALSO la segunda variante por lo que esta solución no cumple esta regla y se continua con la siguiente comparación.

La tercera variante afirma que debe de cumplirse que el índice de credibilidad de x sobre y debe de ser mayor o igual a β y que el índice de credibilidad de y sobre x sea menor a β:

$$\sigma(x, y) \geq \beta = 0.347 \geq 0.67$$

$$\sigma(y,x) < \beta = 0.289 < 0.67$$

Se comprueba que es FALSO la segunda variante por lo que esta solución no cumple esta regla, ya que esta regla es una conjunción mediante un AND que debe de cumplirse ambas reglas para poder ser verdadero. Se continúa con la siguiente comparación.

La cuarta variante afirma que el índice de credibilidad de x sobre y debe de ser mayor al índice de credibilidad de y sobre x y el índice de credibilidad de x sobre y debe de ser mayor a 0.5 para cumplirse:

$$\sigma(x,y) > \sigma(y,x) = 0.347 > 0.289$$

$$\sigma(x,y) > 0.5 = 0.347 > 0.5$$

Se comprueba que es FALSO la segunda variante por lo que esta solución no cumple esta regla, ya que esta regla, como la tercera variante es una conjunción mediante un AND que deben de cumplirse ambas reglas para poder ser verdadero. Se continúa con la siguiente comparación.

La quinta variante afirma que el índice de credibilidad de x sobre y debe ser mayor o igual a β y que el índice de credibilidad de y sobre $x$debe ser menor a 0.5 para cumplirse:

$$\sigma(x,y) \geq \beta = 0.347 \geq 0.67$$

$$\sigma(y,x) < 0.5 = 0.289 < 0.5$$

Se comprueba que es FALSO la segunda variante por lo que esta solución no cumple esta regla, al igual que la tercera y cuarta variante esta regla es una conjunción mediante un AND donde se deben de cumplir ambas condiciones. Se continúa con la siguiente comparación.

Una vez aplicado el modelo de sobre clasificación se procede a una comprobación mediante un AND de si la nueva solución es apta para ser agregada al conjunto EP. Debe de cumplir que haya sido aceptada mediante el cálculo de Tchebycheff y haber cumplido una de las reglas que anteriormente se vieron del modelo de sobre clasificación.

De manera de ejemplo se tomará la solución que se ha tratado en este trabajo, recordando que el cálculo de Tchebycheff tiene una variable booleana donde almacena sí la solución era apta y una bandera booleana si la solución fue aceptada mediante el modelo de sobre clasificación. La siguiente comparación se hace de la siguiente manera:

Algoritmo PMOEA/D: Proceso de actualización de EP

1 **sí** $Tchebycheff$ = true && OM=true **entonces**
2 actualizar(EP, y)
  $N \leftarrow N + 1$
3 **en otro caso**
4 $N \leftarrow N + 1$
5 **fin**

Figura 8. Proceso de actualización EP dentro de PMOEA/D.

Si es aceptada la solución, esta entra al conjunto de soluciones EP, continuando la siguiente iteración donde N es el número de funciones escalares de MOEA/D. Si no se cumple la condición solamente continúa con la siguiente iteración y la solución no es almacenada dentro de EP.

Capítulo 5. Validación Experimental

5.1.Descripción del diseño experimental

Dentro de la experimentación se propone un conjunto de siete instancias con la finalidad de ver el comportamiento que toman los algoritmos MOEA/D y PMOEA/D, ver el tiempo que tarda el algoritmo en encontrar el subconjunto de soluciones, ver la calidad de las soluciones, pero también el explicar por qué se tomaron instancias pequeñas para la experimentación. Se trata de comparar cuál es la diferencia de un algoritmo sin preferencias contra un algoritmo con preferencias y finalmente hacer una comparación entre ambos algoritmos.

Se dividirá en 6 subtemas: Descripción de las instancias, Generador de instancias, Características del entorno, Obtención de los mejores compromisos de MOEA/D y PMOEA/D y la comparación de los resultados de ambos algoritmos tanto en tiempo como calidad de las soluciones.

5.2 Descripción de los parámetros de la instancia dentro de los algoritmos

La Tabla 21 muestra los parámetros que se utilizaron dentro las instancias para su ejecución.

Valores	Descripción
4	Número de Objetivos
1	Restricción
1. 100 2. 120 3. 170 4. 250 5. 270 6. 300 7. 450	Peso máximo
1. 3 2. 5 3. 8 4. 10 5. 12 6. 14 7. 17	Número de materiales
1	Número de DMs
obj1 0.5442549904416053 obj2 0.011736011736373404 obj3 0.022483801750960406 obj4 0.42152519607106087	Pesos del DM
obj1 32 obj2 22 obj3 10 obj4 16	Umbral de preferencia
obj1 10	Umbral de indiferencia

obj2 2 obj3 8 obj4 7	
obj1 55 obj2 40 obj3 20 obj4 55	Umbral de veto
0.67	Índice de credibilidad del DM

Tabla 21. Conjunto de Datos de la Instancia

Los datos con los que se trabajan son los siguientes: número de Objetivos igual a 4, se tiene 1 restricción (para este caso es el peso máximo), con la cual se dispone que varía por cada instancia que se propondrá entre 100 y 500, como un peso máximo por cada conjunto de soluciones, en este caso un conjunto de materiales que se escojan. El número de materiales al igual que el peso máximo ira variando por cada instancia de 3 a 17 materiales (variables de decisión), recordar que esta instancia tiene como base el problema de cartera de proyectos, donde el presupuesto equivale al peso máximo para este trabajo y el número de materiales equivale al número de proyectos que existen por escoger y obtener una cartera. El número de DMs para este trabajo es 1 ya que solo se tiene un tomador de decisiones. Por cada objetivo se tendrá un conjunto de valores de peso por cada objetivo, un umbral de preferencia que tiene el DM sobre cada objetivo, un umbral de indiferencia por cada objetivo, un umbral de veto y un índice de credibilidad que representa un parámetro de que una solución satisface como buena al DM.

5.3 Generador de las Instancias

Para generar las instancias dentro de este trabajo se crearon 7 instancias derivadas de la lectura de los 4 sensores propuestos en este trabajo (temperatura, humedad, gas, distancia) que serán los objetivos a optimizar, constará de un conjunto de soluciones iniciales que serán los materiales que tendrán un peso individual, y las cuatro medidas de los sensores por cada material. Como soluciones finales se quiere tener un conjunto de materiales cuyos objetivos sean minimizados y a su vez no sobrepasen el peso máximo final. Tabla 22 resumen las características de las siete instancias con las que se van a trabajar, cada instancia se va a representar con la nomenclatura I01-ARDUINO, I02-ARDUINO, …, I07-ARDUINO.

Instancia	Peso Máximo	Num. Objetivos	Num. Materiales	Num. Evaluaciones	T	Población	Materiales	
							Costo	Objetivos
I01	100	4	3	1000	2	15	[50] [30] [80]	[20, 30, 20, 47] [10, 15, 15, 43] [20, 25, 20, 41]
I02	120	4	5	1000	2	15	[50] [30] [80] [100] [10]	[20,30,20,47] [10,15,15,43] [20,25,20,41] [20,10,29,47] [10,15,20,39]
I03	170	4	8	1000	2	15	[50] [30] [80] [100] [10] [55] [120] [38]	[20, 30, 20, 47] [10, 15, 15, 43] [20, 25, 20, 41] [20,10,29,47] [10,15,20,39] [15,23,25,40] [10, 10, 15, 45] [10, 20, 20, 50]
I04	250	4	10	1000	5	15	[50] [30] [80]	[20, 30, 20, 47] [10, 15, 15, 43] [20, 25, 20, 41]

							[100] [10] [55] [120] [38] [150] [15]	[20,10,29,47] [10,15,20,39] [15,23,25,40] [10, 10, 15, 45] [10, 20, 20, 50] [30,5,30,35] [20,25,10,30]
I05	270	4	12	1000	5	15	[50] [30] [80] [100] [10] [55] [120] [38] [150] [15] [50] [30] [30] [10]	[20, 30, 20, 47] [10, 15, 15, 43] [20, 25, 20, 41] [20,10,29,47] [10,15,20,39] [15,23,25,40] [10, 10, 15, 45] [10, 20, 20, 50] [30,5,30,35] [20,25,10,30] [28,29,20,47] [28,24,11,43] [21,35,18,50] [36,20,25,55]
I06	300	4	14	1000	5	15	[50] [30] [80] [100] [10] [55] [120] [38] [150] [15] [50] [30] [30] [10]	[20, 30, 20, 47] [10, 15, 15, 43] [20, 25, 20, 41] [20,10,29,47] [10,15,20,39] [15,23,25,40] [10, 10, 15, 45] [10, 20, 20, 50] [30,5,30,35] [20,25,10,30] [28,29,20,47] [28,24,11,43] [21,35,18,50] [36,20,25,55]
I07	450	4	17	1000	7	15	[50] [30] [80] [100] [10] [55] [120] [38] [150] [15] [50] [30] [30] [10] [50] [30] [100]	[20, 30, 20, 47] [10, 15, 15, 43] [20, 25, 20, 41] [20,10,29,47] [10,15,20,39] [15,23,25,40] [10, 10, 15, 45] [10, 20, 20, 50] [30,5,30,35] [20,25,10,30] [28,29,20,47] [28,24,11,43] [21,35,18,50] [36,20,25,55] [20,30,12,30] [21,20,27,60] [30,18,20,32]

Tabla 22. Descripción de las Instancias utilizadas en la experimentación.

5.4 Características del Entorno

Se describen las características del entorno con el que se trabajó para la programación y la implementación de este trabajo. Como mención importante el código fue hecho en C++, usando como entorno de desarrollo el IDE de Arduino de su página oficial. Para la programación se utilizó un equipo de cómputo con las características mostradas en la Tabla 23.

Modelo	**Lenovo ThinkPad**
Microcontrolador	Intel Core i7-6820HQ
Memoria RAM	16 GB
Velocidad	2.71 GHZ
Sistema Operativo	Windows 10

Tabla 23. Características del Equipo de Computo

Para la implementación se utilizó una tarjeta microcontroladora de Arduino, en concreto la tarjeta MEGA 2560 que tiene las características mostradas en la Tabla 24.

Modelo	MEGA 2560
Microcontrolador	ATMega2560
Voltaje	5V/7-12V
Velocidad	16 MHz
Memoria Flash	256 KB
Memoria Dinámica	8 KB

Tabla 24. Características de la Tarjeta de Hardware.

5.5 Resultados

Las Tablas 25 y 26 muestran el resumen de los resultados obtenidos por los algoritmos P/MOEAD y MOEA/D respectivamente. Se colocará como una parte importante el peso del sketch de Arduino y los bytes utilizados dentro de la memoria dinámica para el uso de las variables disponibles dentro de la tarjeta. Las tablas muestran para cada instancia una tabla con 6 columnas que son *Vars*, *Objetivos*, *Res*, Ocv, *Nvc*, *ResV*. La columna *Vars* son las variables del problema donde un cromosoma es de tamaño $n = numero\ de\ materiales$ donde 1 representa que se escogió dicho material y 0 lo contrario, la segunda columna *Objetivos* es un conjunto de $m = objetivos$, la tercer columna *Res* es el costo de dicha solución, en la cuarta columna *Ocv* si dicha solución superó la restricción del presupuesto se representará por cuanto se pasó, la quinta columna *Nvc* representa el número de restricciones violadas y la sexta columna *ResV* representa en que posición se encuentran los materiales que violan la restricción.

Vars	**Objetivos**				**Res**	**Ocv**	**Nvc**	**RestV**
0 0 0	0.00	0.00	0.00	0.00	80.00	0.00	0	0
0 0 0	0.00	0.00	0.00	0.00	0.00	0.00	0	0
0 1 0	10.00	15.00	15.00	43.00	30.00	0.00	0	0
0 1 0	10.00	15.00	15.00	43.00	30.00	0.00	0	0
0 1 0	10.00	15.00	15.00	43.00	30.00	0.00	0	0

a) I01

Vars	**Objetivos**				**Res**	**Ocv**	**Nvc**	**RestV**
1 1 0 0 0	30.00	45.00	35.00	90.00	80.00	0.00	0	0
1 1 0 0 0	30.00	45.00	35.00	90.00	80.00	0.00	0	0
1 0 0 0 1	30.00	45.00	40.00	86.00	60.00	0.00	0	0
0 0 0 0 1	10.00	15.00	20.00	39.00	10.00	0.00	0	0
0 0 0 0 1	10.00	15.00	20.00	39.00	10.00	0.00	0	0

b) I02

Vars	Objetivos				Res	Ocv	Nvc	RestV
0 0 0 1 0 0 0 0	20.00	10.00	29.00	47.00	100.00	0.00	0	0
0 0 0 1 0 0 0 0	20.00	10.00	29.00	47.00	100.00	0.00	0	0
0 0 0 1 0 0 0 0	20.00	10.00	29.00	47.00	100.00	0.00	0	0
0 0 0 1 0 0 0 0	20.00	10.00	29.00	47.00	100.00	0.00	0	0
0 0 0 1 0 0 0 0	20.00	10.00	29.00	47.00	100.00	0.00	0	0

c) I03

Vars	Objetivos				Res	Ocv	Nvc	RestV
0 0 0 0 1 0 0 0 0 1	30.00	40.00	30.00	69.00	25.00	0.00	0	0
0 0 0 0 1 0 0 0 0 1	30.00	40.00	30.00	69.00	25.00	0.00	0	0
0 0 0 0 1 0 0 0 0 1	30.00	40.00	30.00	69.00	25.00	0.00	0	0
0 0 0 0 1 0 0 0 0 1	30.00	40.00	30.00	69.00	25.00	0.00	0	0
0 0 0 0 1 0 0 0 0 1	30.00	40.00	30.00	69.00	25.00	0.00	0	0

d) I04

Vars	Objetivos				Res	Ocv	Nvc	RestV
0 0 1 0 0 0 0 1 0 0 0 0	30.00	45.00	40.00	91.00	118.00	0.00	0	0
0 0 1 0 0 0 0 1 0 0 0 0	30.00	45.00	40.00	91.00	118.00	0.00	0	0
0 0 1 0 0 0 0 1 0 0 0 0	30.00	45.00	40.00	91.00	118.00	0.00	0	0
0 0 1 0 0 0 0 1 0 0 0 0	30.00	45.00	40.00	91.00	118.00	0.00	0	0
0 0 1 0 0 0 0 1 0 0 0 0	30.00	45.00	40.00	91.00	118.00	0.00	0	0

e) I05

Vars	Objetivos				Res	Ocv	Nvc	RestV
1 0 0 0 1 1 0 0 1 1 0 0 0 0	95.00	98.00	105.00	191.00	280.00	0.00	0	0
1 0 0 0 1 1 0 0 1 1 0 0 0 0	95.00	98.00	105.00	191.00	280.00	0.00	0	0
1 0 0 0 1 1 0 0 1 1 0 0 0 0	95.00	98.00	105.00	191.00	280.00	0.00	0	0
1 0 0 0 1 1 0 0 1 1 0 0 0 0	95.00	98.00	105.00	191.00	280.00	0.00	0	0
1 0 0 0 1 1 0 0 1 1 0 0 0 0	95.00	98.00	105.00	191.00	280.00	0.00	0	0

f) I06

Vars	Objetivos				Res	Ocv	Nvc	RestV
0 0 0 0 0 0 1 1 0 1 0 0 1 0 1 1 0	102.00	140.00	102.00	265.00	283.00	0.00	0	0
0 1 0 1 1 0 0 0 1 0 0 0 0 0 0 1 1	121.00	83.00	141.00	256.00	420.00	0.00	0	0
0 1 0 1 1 0 0 0 1 0 0 0 0 0 0 1 1	121.00	83.00	141.00	256.00	420.00	0.00	0	0
0 1 0 1 1 0 0 0 1 0 0 0 0 0 0 1 1	121.00	83.00	141.00	256.00	420.00	0.00	0	0
0 1 0 1 1 0 0 0 1 0 0 0 0 0 0 1 1	121.00	83.00	141.00	256.00	420.00	0.00	0	0

g) I07

Tabla 25. Resultados de PMOEA/D.

Vars	Objetivos				Res	Ocv	Nvc	RestV
0 0 0	0.00	0.00	0.00	0.00	80.00	0.00	0	0

0 0 0	0.00	0.00	0.00	0.00	0.00	0.00	0	0
0 1 0	10.00	15.00	15.00	43.00	30.00	0.00	0	0
0 1 0	10.00	15.00	15.00	43.00	30.00	0.00	0	0
0 1 0	10.00	15.00	15.00	43.00	30.00	0.00	0	0

a) I01

Vars	**Objetivos**				**Res**	**Ocv**	**Nvc**	**RestV**
1 1 0 0 0	30.00	45.00	35.00	90.00	80.00	0.00	0	0
1 1 0 0 0	30.00	45.00	35.00	90.00	80.00	0.00	0	0
1 0 0 0 1	30.00	45.00	40.00	86.00	60.00	0.00	0	0
0 0 0 0 1	10.00	15.00	20.00	39.00	10.00	0.00	0	0
0 0 0 0 1	10.00	15.00	20.00	39.00	10.00	0.00	0	0

b) I02

Vars	**Objetivos**				**Res**	**Ocv**	**Nvc**	**RestV**
0 0 0 1 0 0 0 0	20.00	10.00	29.00	47.00	100.00	0.00	0	0
0 0 0 1 0 0 0 0	20.00	10.00	29.00	47.00	100.00	0.00	0	0
0 0 0 1 0 0 0 0	20.00	10.00	29.00	47.00	100.00	0.00	0	0
0 0 0 1 0 0 0 0	20.00	10.00	29.00	47.00	100.00	0.00	0	0
0 0 0 1 0 0 0 0	20.00	10.00	29.00	47.00	100.00	0.00	0	0

c) I03

Vars	**Objetivos**				**Res**	**Ocv**	**Nvc**	**RestV**
0 0 0 0 1 0 0 0 0 1	30.00	40.00	30.00	69.00	25.00	0.00	0	0
0 0 0 0 1 0 0 0 0 1	30.00	40.00	30.00	69.00	25.00	0.00	0	0
0 0 0 0 1 0 0 0 0 1	30.00	40.00	30.00	69.00	25.00	0.00	0	0
0 0 0 0 1 0 0 0 0 1	30.00	40.00	30.00	69.00	25.00	0.00	0	0
0 0 0 0 1 0 0 0 0 1	30.00	40.00	30.00	69.00	25.00	0.00	0	0

d) I04

Vars	**Objetivos**				**Res**	**Ocv**	**Nvc**	**RestV**
0 0 1 0 0 0 0 1 0 0 0 0	30.00	45.00	40.00	91.00	118.00	0.00	0	0
0 0 1 0 0 0 0 1 0 0 0 0	30.00	45.00	40.00	91.00	118.00	0.00	0	0
0 0 1 0 0 0 0 1 0 0 0 0	30.00	45.00	40.00	91.00	118.00	0.00	0	0
0 0 1 0 0 0 0 1 0 0 0 0	30.00	45.00	40.00	91.00	118.00	0.00	0	0
0 0 1 0 0 0 0 1 0 0 0 0	30.00	45.00	40.00	91.00	118.00	0.00	0	0

e) I05

Vars	**Objetivos**				**Res**	**Ocv**	**Nvc**	**RestV**
1 0 1 0 0 0 0 0 1 1 0 0 0 0	90.00	85.00	80.00	153.00	295.00	0.00	0	0
1 0 1 0 0 0 0 0 1 1 0 0 0 0	90.00	85.00	80.00	153.00	295.00	0.00	0	0
1 0 1 0 0 0 0 0 1 1 0 0 0 0	90.00	85.00	80.00	153.00	295.00	0.00	0	0
1 0 1 0 0 0 0 0 1 1 0 0 0 0	90.00	85.00	80.00	153.00	295.00	0.00	0	0
1 0 1 0 0 0 0 0 1 1 0 0 0 0	90.00	85.00	80.00	153.00	295.00	0.00	0	0

f) I06

Vars	Objetivos				Res	Ocv	Nvc	RestV
0 1 0 1 0 0 0 0 0 1 1 0 0 0 0 1 0	99.00	99.00	101.00	227.00	225.00	0.00	0	0
0 1 0 1 0 0 0 0 0 1 1 0 0 0 0 1 0	99.00	99.00	101.00	227.00	225.00	0.00	0	0
0 1 0 1 0 0 0 0 0 1 1 0 0 0 0 1 0	99.00	99.00	101.00	227.00	225.00	0.00	0	0
0 1 0 1 0 0 0 0 0 1 1 0 0 0 0 1 0	99.00	99.00	101.00	227.00	225.00	0.00	0	0
0 1 0 1 0 0 0 0 0 1 1 0 0 0 0 1 0	99.00	99.00	101.00	227.00	225.00	0.00	0	0

g) I07

Tabla 26. Resultados de MOEA/D.

5.6 Análisis de Resultados

PMOEA/D

En la instancia I01, el Sketch de Arduino usa 21088 bytes que representa un 8% del espacio total, las variables globales usadas usan 600 bytes que representa un 7% de la memoria dinámica. Con un tiempo de 8.032 segundos de ejecución.

En la instancia I02, el Sketch de Arduino usa 21316 bytes que representa un 8% del espacio total, las variables globales usadas usan 640 bytes que representa un 7% de la memoria dinámica. Con un tiempo de 8.453 segundos de ejecución.

En la instancia I03, el Sketch de Arduino usa 21538 bytes que representa un 8% del espacio total, las variables globales usadas usan 700 bytes que representa un 8% de la memoria dinámica. Con un tiempo de 10.723 segundos de ejecución.

En la instancia I04, el Sketch de Arduino usa 21708 bytes que representa un 8% del espacio total, las variables globales usadas usan 740 bytes que representa un 9% de la memoria dinámica. Con un tiempo de 15.311 segundos de ejecución.

En la instancia I05, el Sketch de Arduino usa 21974 bytes que representa un 8% del espacio total, las variables globales usadas usan 780 bytes que representa un 9% de la memoria dinámica. Con un tiempo de 15.528 segundos de ejecución.

En la instancia I06, el Sketch de Arduino usa 22214 bytes que representa un 8% del espacio total, las variables globales usadas usan 820 bytes que representa un 10% de la memoria dinámica. Con un tiempo de 17.364 segundos de ejecución.

En la instancia I07, el Sketch de Arduino usa 22520 bytes que representa un 9% del espacio total, las variables globales usadas usan 880 bytes que representa un 10% de la memoria dinámica. Con un tiempo de 22.773 segundos de ejecución.

MOEA/D

En la instancia I01, el Sketch de Arduino usa 21088 bytes que representa un 8% del espacio total, las variables globales usadas usan 600 bytes que representa un 7% de la memoria dinámica. Con un tiempo de 5.400 segundos de ejecución.

En la instancia I02, el Sketch de Arduino usa 21316 bytes que representa un 8% del espacio total, las variables globales usadas usan 640 bytes que representa un 7% de la memoria dinámica. Con un tiempo de 5.854 segundos de ejecución.

En la instancia I03, el Sketch de Arduino usa 21538 bytes que representa un 8% del espacio total, las variables globales usadas usan 700 bytes que representa un 8% de la memoria dinámica. Con un tiempo de 6.815 segundos de ejecución.

En la instancia I04, el Sketch de Arduino usa 21708 bytes que representa un 8% del espacio total, las variables globales usadas usan 740 bytes que representa un 9% de la memoria dinámica. Con un tiempo de 8.795 segundos de ejecución.

En la instancia I05, el Sketch de Arduino usa 21974 bytes que representa un 8% del espacio total, las variables globales usadas usan 780 bytes que representa un 9% de la memoria dinámica. Con un tiempo de 8.966 segundos de ejecución.

En la instancia I06, el Sketch de Arduino usa 22214 bytes que representa un 8% del espacio total, las variables globales usadas usan 820 bytes que representa un 10% de la memoria dinámica. Con un tiempo de 9.926 segundos de ejecución.

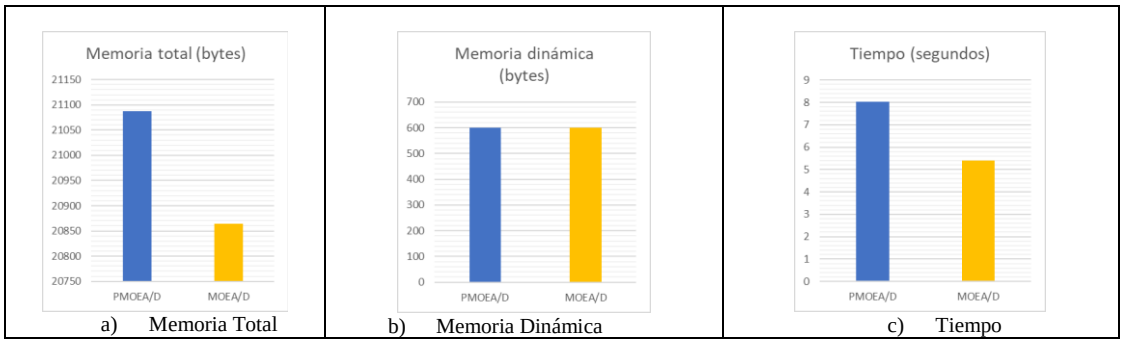
En la instancia I07, el Sketch de Arduino usa 22520 bytes que representa un 8% del espacio total, las variables globales usadas usan 880 bytes que representa un 10% de la memoria dinámica. Con un tiempo de 12.121 segundos de ejecución

5.7 Comparación de Resultados

Para la comparación de resultados se utilizan gráficas para mostrar las diferencias que se obtuvieron al ejecutar ambos algoritmos en la tarjeta de Arduino basados en memoria total (que es lo que peso el sketch al momento de la compilación), memoria dinámica (que son las variables internas que se usaron durante la compilación) y el tiempo que tardo cada ejecución después de su finalización. Nota que por ahora en las soluciones no se verá una comparación ya que en ambos algoritmos no hubo grandes cambios en calidad de soluciones por lo que se está revisando cuál

puede ser el problema, aunque se tiene una posible hipótesis que apunta a que las instancias sean muy pequeñas para encontrar alguna diversidad de soluciones con y sin preferencias, ya que se encontraron pequeños cambios a manera que iba subiendo la cantidad de variables.

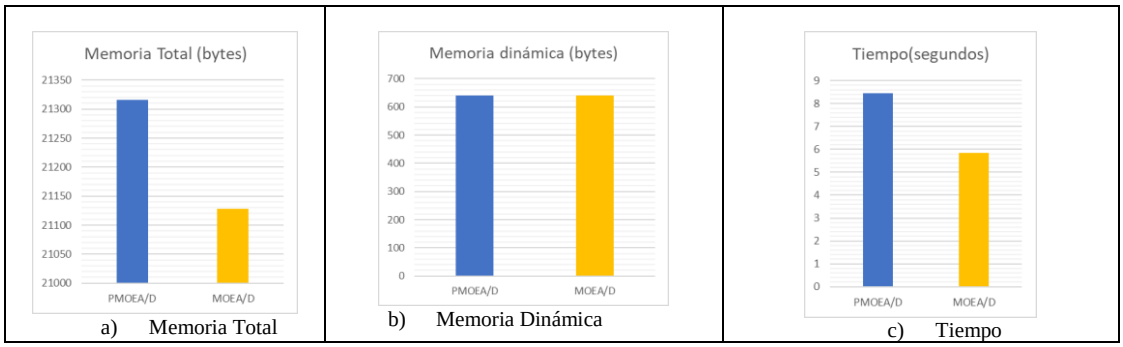
La primera comparación de la instancia I01 de PMOEA/D y MOEA/D se puede ver en la Figura 9. En esta comparación se muestra la diferencia en memoria total, dinámica y tiempo requeridos en Arduino. Se puede observar en la Figura 9a que PMOEA/D ocupa un poco más de espacio que MOEA/D por la incorporación del modelo de preferencias, pero en la Figura 9b se puede ver que se obtuvo el mismo tamaño de memoria dinámica en ambos algoritmos. Por último, en la Figura 9c se puede ver que en promedio PMOEA/D tarda cerca de tres segundos en obtener el conjunto de soluciones.



a) Memoria Total b) Memoria Dinámica c) Tiempo

Figura 9. Comparación de memoria usada en Arduino y tiempo consumido en I01.

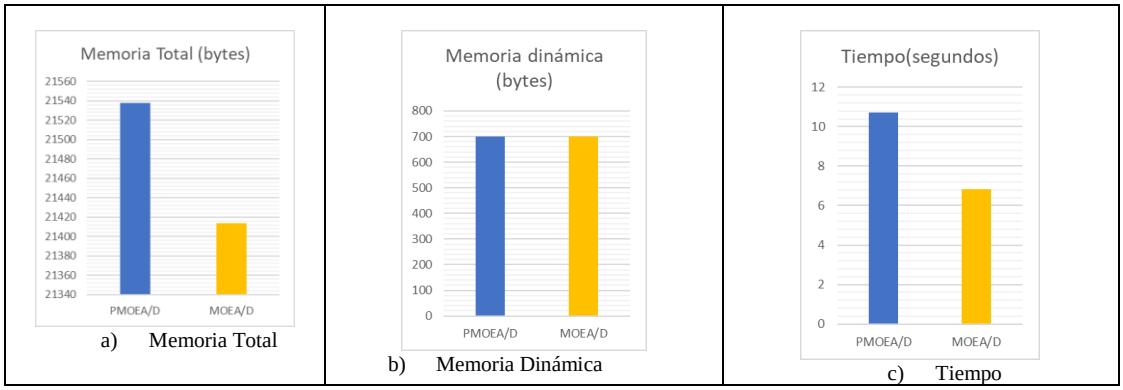
Como se vuelve a notar en esta segunda comparación, la instancia I02 de PMOEA/D y MOEA/D se puede ver en la Figura 10a también que PMOEA/D ocupa un poco más de espacio que MOEA/D como en la anterior comparación de igual manera por la incorporación del modelo de preferencias, pero en la Figura 10b se puede ver que se obtuvo el mismo tamaño de memoria dinámica en ambos algoritmos. Por último, en la Figura 10c se puede ver que en promedio PMOEA/D tarda cerca de tres segundos en obtener el conjunto de soluciones como en la anterior comparación.



a) Memoria Total b) Memoria Dinámica c) Tiempo

Figura 10. Comparación de memoria usada en Arduino y tiempo consumido en I02.

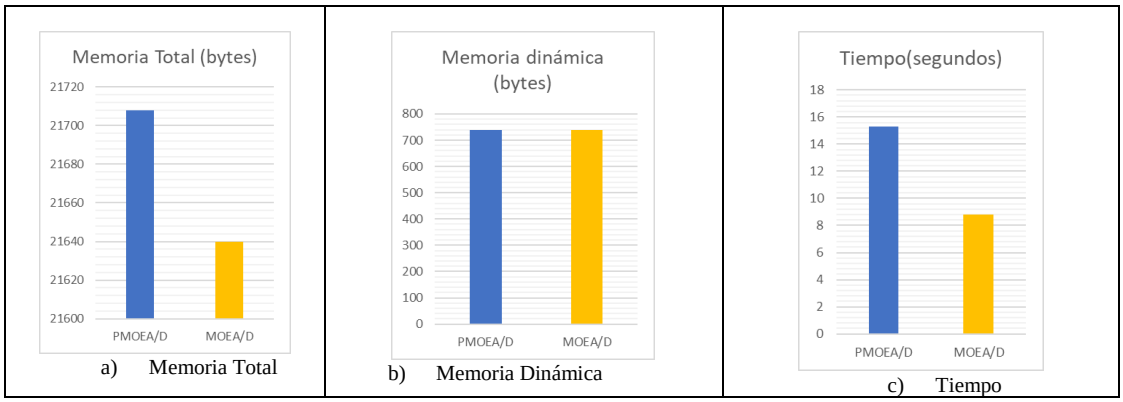
Como se ha visto en las anteriores comparaciones de nuevo en esta instancia I03 de PMOEA/D y MOEA/D se puede ver en la Figura 11a también que PMOEA/D ocupa un poco más de espacio que MOEA/D, pero sigue siendo una constante el tamaño de la memoria dinámica en ambos algoritmos como se ve en la Figura 11b. Por último, en la Figura 11c el algoritmo PMOEA/D se acerca por décimas de segundo a 4 segundos de diferencia en comparación del algoritmo MOEA/D, aun así, no es un nivel de significancia importante la diferencia de tiempo.



a) Memoria Total b) Memoria Dinámica c) Tiempo

Figura 11. Comparación de memoria usada en Arduino y tiempo consumido en I03.

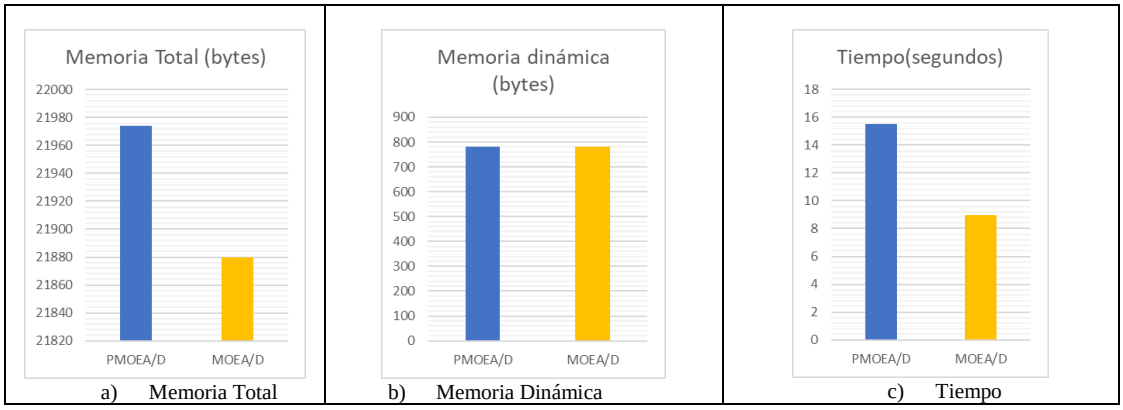
Como se ha visto en las anteriores comparaciones de nuevo en esta instancia I04 de PMOEA/D y MOEA/D se puede ver en la Figura 12a también que PMOEA/D ocupa un poco más de espacio que MOEA/D, pero sigue siendo una constante el tamaño de la memoria dinámica en ambos algoritmos como se ve en la Figura 12b. Por último, en la Figura 12c el algoritmo PMOEA/D es 6.5 segundos más tardado al momento de su terminación para obtener el conjunto de soluciones, pero sigue siendo un nivel bajo de significancia con respecto a MOEA/D.



a) Memoria Total b) Memoria Dinámica c) Tiempo

Figura 12. Comparación de memoria usada en Arduino y tiempo consumido en I04.

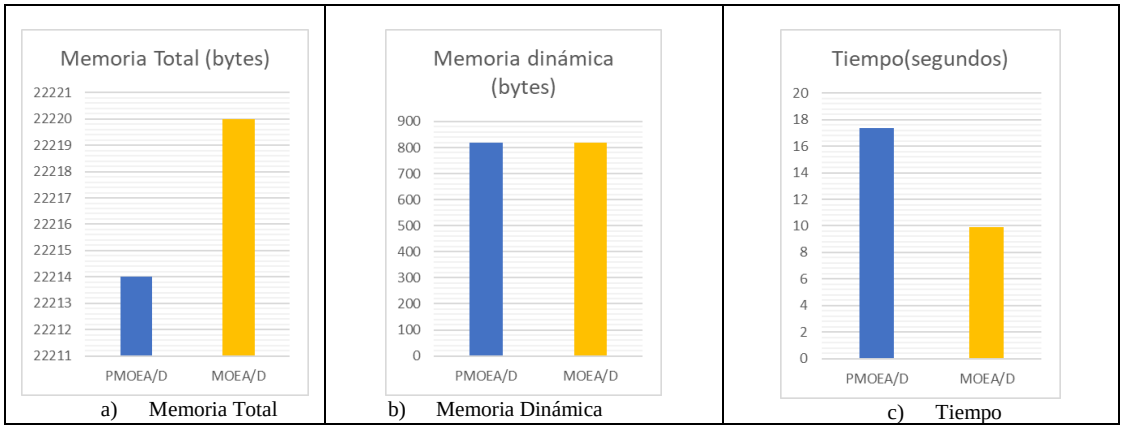
Como se ha visto en las anteriores comparaciones de nuevo en esta instancia I05 de PMOEA/D y MOEA/D se puede ver en la Figura 13a también que PMOEA/D ocupa un poco más de espacio que MOEA/D, pero sigue siendo una constante el tamaño de la memoria dinámica en ambos algoritmos como se ve en la Figura 13b. Por último, en la Figura 13c el algoritmo PMOEA/D es igualmente como la comparación anterior 6.5 segundos más tardado al momento de su terminación para obtener el conjunto de soluciones, pero sigue siendo un nivel bajo de significancia con respecto a MOEA/D.



a) Memoria Total b) Memoria Dinámica c) Tiempo

Figura 13. Comparación de memoria usada en Arduino y tiempo consumido en I05.

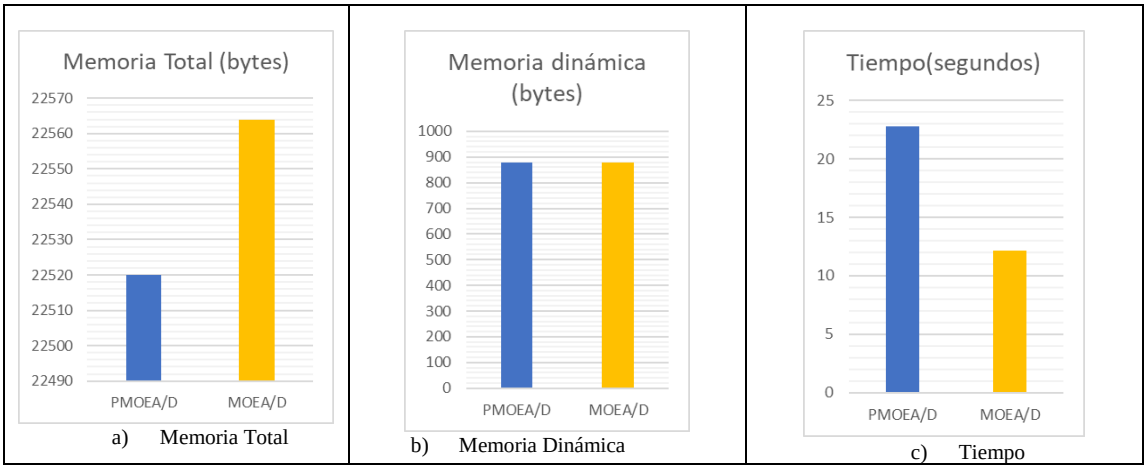
En esta instancia I06 de PMOEA/D y MOEA/D se puede ver en la Figura 14a que ahora el algoritmo MOEA/D es ligeramente más pesado que PMOEA/D, esto puede deberse a que estas instancias trabajan con un número mayor a los T vecinos más cercanos del vecindario, de igual manera el tamaño de la memoria dinámica en ambos algoritmos sigue siendo constante como se ve en la Figura 14b. Por último, en la Figura 14c el algoritmo PMOEA/D es 7.5 segundos más tardado al momento de su terminación para obtener el conjunto de soluciones, esto aumenta en un segundo a la comparación anterior, pero sigue siendo un nivel bajo de significancia con respecto a MOEA/D.



a) Memoria Total b) Memoria Dinámica c) Tiempo

Figura 14. Comparación de memoria usada en Arduino y tiempo consumido en I06.

En esta instancia I07 de PMOEA/D y MOEA/D se puede ver en la Figura 15a que de nueva cuenta el algoritmo MOEA/D es ligeramente más pesado que PMOEA/D, y como se mencionó anteriormente esto puede deberse a que estas instancias trabajan con un número mayor a los T vecinos más cercanos del vecindario, de igual manera el tamaño de la memoria dinámica en ambos algoritmos sigue siendo constante como se ve en la Figura 15b. Por último, en la Figura 15c el algoritmo PMOEA/D ahora es 10 segundos más tardado al momento de su terminación para obtener el conjunto de soluciones, esto puede ser un poco significativo porque se toma el patrón que a manera que aumenta el número de variables el algoritmo PMOEA/D puede ser más tardado que MOEA/D debido al modelo de preferencias que incorpora en el proceso algorítmico.



a) Memoria Total b) Memoria Dinámica c) Tiempo

Figura 15. Comparación de memoria usada en Arduino y tiempo consumido en I07.

Capítulo 6. Conclusiones y Trabajo futuro

En este Capítulo Final se presentan a manera de conclusiones, las contribuciones que se obtuvieron derivadas de este trabajo de investigación, posteriormente se especifican posibles líneas de investigación o trabajos futuros que pueden dar continuidad a este trabajo de investigación.

6.1. Conclusiones

Este documento presentó una propuesta que espera implementar un algoritmo de optimización dentro de Hardware como Arduino, dentro del proceso algorítmico se propuso un modelo de incorporación de preferencias con relaciones de superación basándose en un método de ELECTRE III. Se propuso como tal una arquitectura que uniera la tarjeta de Arduino, el algoritmo de optimización y el modelo de preferencias. A través del diseño experimental propuesto, y los resultados observados, se llegaron a las siguientes conclusiones:

- Fue posible desarrollar con éxito de la Arquitectura de Agente de Hardware que contara con una estrategia de integración de preferencias en un algoritmo de optimización;
- Fue posible implementar un algoritmo de optimización evolutivo para resolver un caso particular del problema de selección de cartera de proyectos en un agente de hardware;
- Fue posible integrar manejo de preferencias en el algoritmo de optimización en hardware a través del uso de métodos de superación;
- Fue implementado con éxito la estrategia de optimización en un hardware con requerimientos mínimos equivalente a los que posee una tarjeta Arduino Mega;
- Existe diferencia entre las soluciones obtenidas por los algoritmos de optimización con y sin preferencias;
- El espacio requerido para implementar PMOEA/D requiere más memoria;
- Las soluciones construidas son muy similares entre sí, sin embargo, se observó que esta similitud esta correlacionada con los tamaños de las instancias incluidos y el número de evaluaciones utilizado.

Finalmente, este trabajo también aporta una manera de obtener los datos y modelarlos para leerlos como instancias para resolver un problema multiobjetivo como lo es PSP caracterizándolo con componentes de Arduino como por ejemplo los sensores.

6.2. Trabajo futuro

A través de este trabajo se encontraron posibles líneas de investigación que pueden dar continuidad con este trabajo de investigación, y son las siguientes:

a) Desarrollar e implementar alguna otra estrategia de optimización que incorpore preferencias dentro de Arduino y compararse contra este trabajo;
b) Trabajar con otras tarjetas de Arduino que tengan mayor capacidad de Hardware como lo puede ser DUE para ver la calidad de soluciones que se obtienen contra MEGA;
c) Optimizar aún más esta arquitectura a nivel de variables para trabajar a nivel de bits en lugar de enteros, y permitir superar las limitaciones actuales en las dimensiones del problema utilizado.

Referencias bibliográficas

Balderas, F. (2018). Modelando la Imprecisión del Problema de Cartera de Proyectos con Filosofía Gris, p 20-21. Instituto Tecnológico de Tijuana, Baja California, México.

Baños R. (2006). Meta-heurísticas Hibridas para Optimización Mono-objetivo y Multiobjetivo Paralelización y Aplicaciones, pp 26-27, Universidad de Almería, Almería.

Budán, M.C., Simari, G.R., & Costaguta, R. (2011). Especificación e implementación de agentes inteligentes para el soporte a la toma de decisiones. Mayo 10, 2020, de *Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET).*

C. A. C. Coello, "Handling preferences in evolutionary multiobjective optimization: a survey," Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512), La Jolla, CA, USA, 2000, pp. 30-37 vol.1, doi: 10.1109/CEC.2000.870272.

Corchado, J. (2005). Modelos y Arquitecturas de Agente.

Chin, K.O., Gan, K.S., Alfred, R., Anthony, P., & Lukose, D. (2015). Agent Architecture : An Overview.

Fernández, E., López, E., López, F. & Coello, C. (2010). Increasing selective pressure towards the best compromise in evolutionary multiobjective optimization: The extended NOSGA method. *Journal ElSevier*, 1, 13.

Ferretti, E., Sosa, C., Aguirre, G., Loyola, J., Cagnina, L. & Errecalde, M. (mayo 11, 2016). Toma de decisiones y aprendizaje en agentes artificiales inteligentes. Mayo 11, 2020, de *Red de Universidades con Carreras en Informática (RedUNCI)* Sitio web: http://sedici.unlp.edu.ar/handle/10915/52730

Figueira, J. R., Greco, S., Roy, B., & Słowiński, R. (2010). ELECTRE Methods: Main Features and Recent Developments. *Handbook of Multicriteria Analysis*, *103*, 51–89. https://doi.org/10.1007/978-3-540-92828-7_3

G. Z. Rivera, “Enfoque Metaheurístico Hibrido para el Manejo de Muchos Objetivos en Optimización de Cartera de Proyectos Interdependientes con Decisiones de Apoyo Parcial”, Tijuana, Tecnológico Nacional de México, 2015.

García, R. (2010). Hiper-heuristico para Resolver el Problema de Cartera de Proyectos Sociales, p. 23. Instituto Tecnológico de Ciudad Madero, México.

González, R. and González, S. (2007). Agentes inteligentes para controlar la dieta de personas con problemas de salud, Universidad Pontifica de Salamanca Campus Madrid, España.

Hernández, H. (2016). Implementación de un agente inteligente BDI sobre hardware reconfigurable, pp 7, Departamento de Ingeniería Electrónica, Bogotá.

Herrero, I. (2015). Arquitectura de Comportamientos Reactivos para Agentes Robóticos basada en CBR, Tesis Doctoral, Escuela Técnica Superior de Ingeniería de Telecomunicación, Málaga.

Hion, C. (2018). Implementación de un algoritmo metaheurístico multiobjetivo para el problema de cartera de proyectos, p, 9-10. Universidad Autónoma de Ciudad Juárez, México.

Iglesias, C. (1997). Fundamentos de los Agentes Inteligentes. Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, España.

K. Liagkouras and K. Metaxiotis, "An Elitist Polynomial Mutation Operator for Improved Performance of MOEAs in Computer Networks," 2013 22nd International Conference on Computer Communication and Networks (ICCCN), Nassau, 2013, pp. 1-5, doi: 10.1109/ICCCN.2013.6614105.

Molinet, J. (2014). Optimización Evolutiva Multiobjetivo basada en el Algoritmo de Huhn-Munkres, Tesis Doctoral, Departamento de Computación Unidad Zacatenco, México.

Mora, J. and Pérez, S. (2006). Método Hibrido basado en la Estructura de Agentes para Localización de fallas en Sistemas de Distribución de energía Eléctrica, Universidad Industrial de Santander, Colombia.

Morales, Ma., 2000). Agentes y Arquitecturas de Control para Robots Autónomos Móviles. En: Memorias del 1er Simposium Inter tecnológico de Computación e Informática SICI'00. Cd. Madero, México.

Q. Zhang and H. Li, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition," in *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712-731, Dec. 2007, doi: 10.1109/TEVC.2007.892759.

Rivera, G. (2015). Enfoque Metaheurístico Híbrido para el Manejo de Muchos Objetivos en Optimización de Cartera de Proyectos Interdependientes con Decisiones de Apoyo Parcial, p. 9. Instituto Tecnológico de Baja California, México.

Rojas R. (2017). Estudio del impacto de patrones de diseño en la implementación de un framework de optimización para apoyo a la toma de decisiones. Tesis de Maestria, Instituto Tecnológico de Ciudad Madero.

Saborido Rubén, Belén Alan & Luque Mariano (2013). Un nuevo Algoritmo Evolutivo de Optimización Multiobjetivo basado en Preferencias: WASF-GA. Departamento de Economía Aplicada (Matemáticas). Facultad de Ciencias Económicas y Empresariales. Universidad de Málaga, 2-3.

Sánchez, J. (2017). Incorporación de preferencias en Metaheurísticas evolutivas a través de clasificación multicriterio, p. 29. Instituto Tecnológico de Tijuana, México.

Schmidt, M. (2015). Arduino: A Quick-Start Guide,

Suárez, M. (2009, diciembre 2). Análisis, diseño e implementación de un agente deliberativo para extraer contextos definitorios en textos especializados. *Revista Interamericana de Bibliotecología*, 32, 84.

Wooldridge M. and Jennings N. R. (1994) 'Agent Theories, Architectures and Languages: A Survey', Intelligent Agents (ECAI-94 Workshop Proceedings on Agent Theories, Architectures and Languages, Amsterdam, Aug, 1994), pp 1-22, Wooldridge Michael J, and Jennings, Nicholas, (Eds), Springer Verlag, Berlin.

Yevseyeva, I. & Guerreiro, Andreia & Emmerich, Michael & Fonseca, Carlos. (2014). A Portfolio Optimization Approach to Selection in Multiobjective Evolutionary Algorithms. 672-681. 10.1007/978-3-319-10762-2_66.

Anexo A

Como se mencionó a lo largo de este trabajo, donde las preferencias son una parte importante del proceso algorítmico para escoger la mejor solución de acuerdo a un DM, se necesita de un cálculo para la credibilidad de decir que una cartera x es mejor que otra cartera y por lo que a manera de ejemplo se procederá a describir una instancia ficticia sobre cartera de proyectos el cual solo considera dos objetivos a maximizar, un presupuesto a repartir que serán de $500 en diez proyectos como se muestran a continuación:

Proyecto	Objetivo 1	Objetivo 2	Costo del proyecto
1	10	45	120
2	10	50	38
3	30	35	150
4	20	30	70
5	28	47	50
6	28	43	30
7	21	50	100
8	36	55	80
9	20	30	50
10	21	60	95

Para este ejercicio se proponen tres carteras ficticias en la siguiente tabla:

Cartera	Índices de los proyectos	Objetivo 1	Objetivo 2	Costo de la cartera
$\boldsymbol{x_1}$	1,3,5,7,9	109	207	$470
$\boldsymbol{x_2}$	3,4,5,7,10	120	222	$465
$\boldsymbol{x_3}$	1,6,7,8,10	116	253	$425

Para obtener el calculo de credibilidad se necesitan establecer un conjunto de parámetros que serán el modelo de preferencias del DM que son los siguientes:

Vector de pesos para los objetivos (w_i): Representa el umbral de peso, el cual es un valor numérico que indica la importancia de un objetivo con respecto a otro en el objetivo i. Cada elemento w_i del vector cumple que $w_i > 0$.

Umbral de indiferencia (q_i): Representa el umbral de indiferencia, el cual es un valor numérico que indica la indiferencia del TD con respecto a dos alternativas en el objetivo i.

Umbral de preferencia (p_i): Representa el umbral de preferencia, el cual es un valor numérico que indica la preferencia del TD de una alternativa sobre otra en el objetivo i. Cada elemento p del vector debe de cumplir $p_i \geq q_i$.

Umbral de veto (v_i): Representa el umbral de veto, el cual es un valor numérico que sirve como un bloqueo de la relación de superación entre dos alternativas en el objetivo i. Cada elemento v del vector debe cumplir $v_i \geq p_i$.

Para el cálculo del índice de concordancia se calcula de la siguiente forma:

$$c(x,y) = \sum_{k=1}^{n} c_k(x,y) \tag{1}$$

Donde

$$c_k = \begin{cases} w_k & si\ xP_ky \vee xI_ky, \\ 0 & en\ otro\ caso \end{cases} \tag{2}$$

Donde xP_ky y xI_ky son las funciones lógicas de preferencia e indiferencia en él k-ésimo objetivo. La preferencia puede ser definida como:

$$xP_ky = f_k(x) > f_k(y) \wedge \neg xI_ky \tag{3}$$

Donde f_k es la función de evaluación para el k-ésimo objetivo. La indiferencia es definida como:

$$xI_ky = |f_k(x) - f_k(y)| \leq q_k \tag{4}$$

Donde q_k representa el umbral de indiferencia para el j-ésimo objetivo.

Por otra parte, el índice de discordancia se define como:

$$d(x,y) = \min_{k\in\{1,2,3,\ldots,n\}} \{1 - d_k(x,y)\} \tag{5}$$

Donde

$$d_k(x,y) = \begin{cases} 0 & \text{Si } \nabla_k(x,y) < p_k, \\ \dfrac{\nabla_k(x,y) - q_k}{v_k - q_k} & \text{Si } p_k \leq \nabla_k(x,y) < v_k \\ 1 & \text{Si } \nabla_k(x,y) \geq v_k, \end{cases} \tag{6}$$

Donde $\nabla_k(x,y) = f_k(y) - f_k(x)$

Y por último se hace el cálculo de $\sigma(x, y)$, que se hace la siguiente manera:

$$\sigma(x, y) = c(x, y)^* \, d(x, y) \quad (7)$$

Para ejemplificar los pesos que se tendrán los siguientes valores: $w = (0.3, 0.7)$, $q = (10{,}12)$, $p = (15{,}18)$, $v = (32{,}23)$, que van a ser las preferencias del TD para este problema.

A continuación, se procede a calcular el índice de concordancia $c(x_1, x_2)$ quedando de la siguiente manera:

$$c(x_1, x_2) = c_1(x_1, x_2) + c_2(x_1, x_2) = 0 + 0 = 0$$

Se puede mostrar la información en forma de matriz y se puede calcular las siguientes carteras quedando de la siguiente manera:

$$c(x, y) = \begin{bmatrix} 1.0 & 0 & 0.3 \\ 1.0 & 1.0 & 0.3 \\ 1.0 & 1.0 & 1.0 \end{bmatrix}$$

Para el cálculo de discordancia se ejemplificará el cálculo de la cartera $d(x_1, x_2)$

$$d(x_1, x_2) = \min\{1 - d_1(x_1, x_2), 1 - d_2(x_1, x_2)\} = \{1 - 0{,}1 - 0\} = \{1{,}1\} = 1$$

Donde $\nabla_1(x_1, x_2) = 11, \nabla_2(x_1, x_2) = 15$

$$d(x, y) = \begin{bmatrix} 1.0 & 1.0 & 0 \\ 1.0 & 1.0 & 0 \\ 1.0 & 1.0 & 1.0 \end{bmatrix}$$

Una vez teniendo el índice de concordancia y discordancia se procede a realizarse el cálculo del índice de credibilidad $\sigma(x, y) = c(x, y)^* \, d(x, y)$ quedando de la siguiente forma

$$\sigma(x, y) = \begin{bmatrix} 1.0 & 0 & 0 \\ 1.0 & 1.0 & 0 \\ 1.0 & 1.0 & 1.0 \end{bmatrix}$$