PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

BÚSQUEDA DE PARÁMETROS POR OPTIMIZACIÓN DE ENJAMBRE DE PARTÍCULAS PARA UN SOLVER DE PROBLEMAS DE SATISFACCIÓN DE RESTRICCIONES.

CRISTOFFER EDUARDO MORALES LÓPEZ

TESIS DE GRADO

MAGÍSTER EN INGENIERÍA INFORMÁTICA

DICIEMBRE 2010

Pontificia Universidad Católica de Valparaíso
Facultad de Ingeniería
Escuela de Ingeniería Informática

BÚSQUEDA DE PARÁMETROS POR OPTIMIZACIÓN DE ENJAMBRE DE PARTÍCULAS PARA UN SOLVER DE PROBLEMAS DE SATISFACCIÓN DE RESTRICCIONES.

CRISTOFFER EDUARDO MORALES LÓPEZ

Profesor Guía: **Broderick Crawford Labrín.**

Programa: **Magíster en Ingeniería Informática**

DICIEMBRE 2010

Resumen.

Los problemas de satisfacción de restricciones utilizan técnicas de resolución como las Estrategias de Enumeración las cuales pertenecen a la Programación de Restricciones, las cuales para distintos problemas o después de cierto tiempo en el proceso de búsqueda, no son lo suficientemente óptimas para encontrar una solución, por tal motivo nace la idea de la utilización de un mecanismo de adaptación de parámetros de configuración automática para los problemas de satisfacción restricciones. El algoritmo de enjambre de partículas es un proceso en donde los individuos (partículas) se comunican entre si buscando una mejor solución, con este algoritmo de optimización se realizará la búsqueda de los mejores parámetros para la resolución de problemas.

Palabras claves: enjambre, inteligencia de enjambre, constraint programing, adaptación, parámetros.

Abstract.

The constraint satisfaction problems use techniques such as Enumeration Strategies system which belong to a Restriction Programming. Because of different problems or after a period of time in the searching process, they are not good enough to find a solution. For that reason, the idea of a parameter adaptation mechanism with an automatic configuration was created in order to solve the satisfaction restriction problems. The particle swarm algorithm is a process in which the individuals (particles) communicate with each other in order to look for a better solution. This optimization algorithm will search the best parameters to find the resolution to the problems.

Key words: Swarm, Swarm Intelligence, Constraint Programing, adaptation, parameters.

Índice de Contenido

Índice de Figuras

Índice de Tablas

CAPÍTULO 1

1. Introducción

1.1. Introducción.

La capacidad en que los seres vivos pueden resolver ciertos problemas en su vida cotidiana con el objetivo de sobrevivir o realizar labores simples, ha despertado el interés de poder imitar o emular el comportamiento evolutivo y adaptativo en que estos grupos son capaces de resolver estos problemas de manera cotidiana. Es aquí donde las ciencias de la computación junto con otras aéreas, han buscado por muchos años, técnicas y métodos computacionales que puedan comportase con un cierto nivel de inteligencia. La programación con restricciones ha sido la fuente de investigación para resolver problemas abstraídos del mundo real, siendo una de las bases para la investigación de la Inteligencia Artificial. Junto con lo anterior, el estudio del comportamiento de grupos de especies de animales, ha formado la denominada Inteligencia de Enjambre, donde por medio de mensajes, los integrantes del grupo van interactuando para adaptarse y así encontrar la solución al problema que tienen como objetivo común.

Durante el proceso de la solución en un problema de restricciones, la selección de los parámetros que están involucrados, será el determinante si el proceso tendrá una respuesta favorable o no. En el caso de los problemas de satisfacción de restricciones la forma en que las variables y los valores son seleccionados determinará si el proceso de búsqueda tendrá una solución y si esta será la mejor para el problema. Aún así, si se seleccionó una estrategia con los valores y las variables adecuadas, puede que en un determinado tiempo durante la búsqueda de la solución, esta estrategia de selección, no sea la más conveniente de utilizar, siendo necesario seleccionar una nueva estrategia que permita avanzar de mejor manera en el proceso de búsqueda de la solución. Tomando esto como base es que el objetivo de esta investigación es lograr un mecanismo de adaptación que realice la elección de estrategias de solución de forma automática.

Este documento intenta introducir en las metodologías que permitirán realizar este mecanismo de adaptación de parámetros en un problema de satisfacción de restricciones. Partiendo por explicaciones teóricas, para posteriormente, la implementación de ellas. Es importante recalcar que este trabajo ha sido realizado en colaboración con el grupo de investigación de Optimización e Inteligencia Computacional de la Escuela de Ingeniería Informática. Este enfoque colaborativo se ha utilizado con éxito en diferentes tesis, trabajos de título y proyectos de investigación donde han participado estudiantes y profesores, los cuales también son parte referencial de este documento.

1.2. Objetivos de la investigación

1.2.1. Objetivo General

Determinación de parámetros, bajo el enfoque de Programación con Restricciones, de una función de elección de estrategias de enumeración en la resolución de Problemas de Satisfacción de Restricciones a través de Optimización por Enjambre de Partículas.

1.2.2. Objetivos Específicos

- Elaborar un modelo multinivel de búsqueda de ajuste de parámetros a través de Optimización por Enjambre de Partículas.
- Implementación a través de lenguaje de programación JAVA para el acople con el proceso de búsqueda de una solución de un Problema de Satisfacción de Restricciones.
- Selección de estrategias e indicadores de forma autónoma para el proceso de búsqueda de una posible solución.

1.3. Metodología.

El presente trabajo, responde a metodologías de estudio Exploratorio, Descriptivo-Cualitativo y Explicativo, por medio de las cuales se busca recopilar la documentación necesaria para lograr comprender los aspectos relevantes que contextualicen la investigación, y por consiguiente llevar a cabo y sustentar el tema en cuestión, abordando la problemática de mejor manera.

Esta investigación se fragmenta en tres etapas, la primera corresponde a la Descriptiva, en donde se desarrolla principalmente la recopilación y análisis informaciones bibliográficas abstraídas de Libros y diversos Paper expertos en el tema en cuestión, y además documentos de Tesis que aborden el tema en su contexto investigativo.

La segunda etapa Descriptiva-Cualitativa, despliega a partir del procesamiento de la información obtenida en la primera etapa, donde luego de lograr generar un espectro preliminar del tema se procede a identificar los puntos relevantes de las materias concernientes al tema de tesis trabajado.

La tercera etapa Explicativa, desarrollada en base a información concreta obtenida en las dos etapas previas, consiste en la implementación de instrumentos experimentales que expliquen y den respuesta a los fenómenos de la investigación, por medio de análisis comparativos de sus resultados.

1.4. Organización del Texto

En los siguientes capítulos se contextualizará con mayor detalle sobre los diferentes tópicos que se utilizará para la investigación. Se comenzará explicando loa conceptos básicos de un problema de satisfacción de restricciones. Luego se realizará una breve descripción sobre las nociones de las búsquedas adaptativas. Más adelante nos enfrentaremos en la descripción de optimización de enjambres de partículas, conociendo su comportamiento, formas de búsqueda, entre otros. Al finalizar el documento se encontrará el capítulo donde se realizará una descripción del objetivo de la investigación describiendo la interacción de los distintos componentes y la implementación además de los experimentos realizados, concluyendo con el respectivo análisis y conclusiones de los resultados obtenidos.

CAPÍTULO 2

2. Problema de Satisfacción de Restricciones

2.1. Introducción.

Los problemas de satisfacción de restricciones o más conocidos por sus siglas en ingles CSP (Constraint Satisfaction Problem) son una de las ramas en la búsqueda de soluciones a problemas, en la programación de restricciones.

La programación de restricciones se podría definir como el estudio de sistemas computacionales basados en restricciones, que tiene como objetivo resolver problemas mediante restricciones, con tal que estas restricciones sean satisfechas y que optimicen criterios determinados [1]. Tiene su origen en la Inteligencia Artificial y puede ser dividida en dos grupos:

- Satisfacción de Restricciones: está orientada a problemas con dominios finitos y está compuesta por un conjunto de variables finitas, un dominio finito de posibles valores y un conjunto finito de restricciones. El objetivo es lograr asignar a cada variable un valor de su dominio que cumplan todas las restricciones. La tarea es encontrar una solución o las soluciones a todos.

- Resolución de Restricciones. está orientada a problemas con dominios infinitos. Utiliza métodos algebraicos y numéricos en lugar de combinaciones y de búsqueda.

Para efectos de esta investigación se enfocará a Satisfacción de Restricciones.

Como se mencionó anteriormente, un problema de satisfacción de restricciones es la búsqueda de una solución, tal que todas las restricciones de un problema no sean violadas, de forma tal que todas las asignaciones de variables sean realizadas. Las típicas aplicaciones de CSP son problema de vendedor viajero, problema de las 8 reinas, reconocimiento de patrones, movimientos de robots, entre otras.

2.2. Definición Formal.

CSP se define como una tupla (X, D, C)s donde [2]:

- X es un conjunto de variables $\{x_1, \dots, x_n\}$
- D es un vector de dominios $< D_1, \dots, D_n >$ donde a su vez, cada posición i-esima del dominio, son los posibles valores que puede tomar la variable $x_{i-esima}$.
- C es un conjunto finito de restricciones, el cual restringe los valores simultáneos que pueden tomar las variables.

2.3. Búsqueda de la Solución.

Existen varias técnicas en la búsqueda de una solución para un problema de satisfacción de restricciones. A continuación se describirán algunas de estas técnicas.

2.3.1. Algoritmos de Búsqueda Sistemática.

Estos algoritmos de búsqueda sistemática tienen la desventaja de la búsqueda en un árbol, lo que implica si existe un incumplimiento con las restricciones del modelo, se debe retroceder [2] [3].

2.3.1.1. Generar y Probar.

Conocido por sus siglas en inglés como GT (Generate and Test), es una técnica de fuerza bruta la cual asigna la totalidad de valores a las variables y luego comprueba una por una si las asignaciones cumplen con todas las restricciones. Este método prueba todas las combinaciones entre variables y valores. Si una combinación cumple todas las restricciones esa es la solución [3] [4]. Como se puede observar la desventaja de esta técnica es que puede producir un elevado número de errores durante el proceso de asignación, ya que no comprueba la validez de las variables contra las restricciones antes de realizar las asignaciones, lo que produce una pérdida de trabajo en asignaciones inútiles.

2.3.1.2. Backtracking Cronológico.

Este algoritmo en su nombre en inglés Chronological Backtracking (BT o CBT), busca instanciando las variables una por una hasta que encuentra una solución o se queda sin posibles valores de instancias. Cuando se produce una asignación de un valor a una variable se comprueban todas las restricciones hasta el momento. Si se cumplen con

todas las restricciones entonces se pasa a la siguiente variable. En cambio cuando una instancia no cumple con alguna restricción, un valor alternativo es tomado que cumpla las restricciones. Cuando todas las variables han sido instanciadas, el problema se ha encontrado una solución. Si una variable no puede tomar ningún valor sin violar alguna restricción se produce un backtracking, lo que significa que se vuelve a la variable previamente asignada al tratar de asignar un valor [4].

2.3.2. Árbol de Búsqueda.

El espacio de búsqueda de una solución para CSP, puede ser mirado como un árbol, el cual está compuesto por nodos, arcos y hojas que corresponde a las variables instanciadas. La búsqueda de una solución se basa en Backtracking (BT) que se describió en la sección anterior. Los componentes de un árbol de búsqueda T son [5]:

- Nodos, arcos y arcos salientes ordenados de cada nodo.
- Un nodo representan una instanciación de un par ordenado $I(u) = \{x_{i_1} = v_{i_1}, \dots, x_{i_k} = v_{i_k}\}$
- Nodo raíz u_0 con $I(u_0) = \emptyset$
- Existe un arco desde un nodo u a un nodo u_1 siempre y cuando $I(u_1) = I(u) \uplus (x = v)$, x y v es una variables y uno de sus valores, respectivamente ($\uplus$ es una concatenación). u_1 será llamado hijo de u y u padre de u_1.
- Para todo nodo u, T(u) es un subárbol de T con la raíz de u. El conjunto $ch(u, T)$ de hijos de u en T es totalmente ordenado por $< c$
- El árbol de búsqueda T_P^A de un algoritmo de búsqueda backtrack A resolviendo un particular problema P es el árbol que hay un mapeo de uno a uno entre los nodos en el árbol y la instanciaciones visitadas por A hasta haber alcanzado una solución o probando la inconsistencia de P.
- Siendo u_1 y u_2 hijos del nodo $u \in T_P^A$, $u_1 <_c u_2$ si solo si $I(u_1)$ a sido visitado por A antes $I(u_2)$.
- El árbol de búsqueda $\overline{T_P^A}$ de A sobre P es el árbol de búsqueda desarrollado por A si A no fue detenido antes de encontrar una solución, sino continuado hasta el fin de la enumeración.

Como se dijo anteriormente el árbol de búsqueda utiliza el backtracking cronológico como base, el cual es un algoritmo muy sencillo y a la vez ineficiente en la búsqueda de las soluciones, entre una de las falencias es que no tiene memoria que le ayude a recordar acciones anteriores que le ayuden a evitar realizar asignaciones innecesarias. Según esto el árbol de búsqueda también utiliza algoritmos de búsqueda más robustos los cuales se definen en las siguientes secciones.

2.3.3. Algoritmos Look-Back

Los algoritmos de tipo look-back son estrategias inteligentes para enfrenarse a los problemas o situaciones sin salida, explotando la información desde las variables actuales hacia las pasadas, del mismo modo que el Backtracking cronológico, donde se llevan a cabo, en primer lugar, la comprobación de la consistencia, hacia atrás, como un modo más eficiente de comportarse frente a este tipo de situación [1].

2.3.3.1. Backjumping (BJ)

Los algoritmos Backjumping (BJ) son similares a al Backtracking cronológico, la diferencia radica cuando se produce una inconsistencia, en vez de retroceder a la variable anteriormente instanciada, BJ salta a la variable más cercana a la actual, comportándose así de una manera más inteligente cuando se encuentra en situaciones sin salida [1].

2.3.3.2. Conflict-Directed Backjumping(CBJ)

Es una mejora a BJ, donde cada variable X_i mantiene un conjunto denominado conflicto, el cual contiene las variables pasadas en donde estas estén en conflicto con X_i En el momento en el que la comprobación de la consistencia entre la variable actual X_i y una variable pasada X_j falla, la variable X_j se inserta en el conjunto de conflicto de X_i. En una situación sin salida, CBJ salta a la variable más profunda en su conjunto conflicto, por ejemplo X_k, donde $k < i$. Al mismo tiempo se incluye el conjunto conflicto de X_i al de X_k, por lo que no se pierde ninguna información sobre conflictos [6].

2.3.3.3. Learning

Learning es un método que guarda todas las restricciones implícitas que son abstraídas en el momento de la búsqueda, usándolas para reducir el espacio de indagación. La idea es mantener almacenar aquellas asignaciones de variables las cuales nos produjeron realizar un backtrack.

2.3.4. Algoritmos Look-Ahead: For-ward Checking

Como anteriormente se mencionó, los algoritmos Look-Back realizan una comprobación de la consistencia pero hacia atrás, ignorando futuras variables, a diferencia de lo que ocurre con los algoritmos Look-Ahead, puestos que éstos realizan una comprobación hacia adelante en cada iteración de la búsqueda, vale decir, éstos algoritmos pueden

descubrir valores inconscientes y rebajar las variables futuras, ya que se efectúan las comprobaciones para lograr las inconsistencias de las variables futuras involucradas, además en las variables actual y pasadas [1].

2.3.4.1. Forward Checking

Dentro de los algoritmos look-ahead los algoritmos Forward Checking (FC) son los más comunes. FC comprueba hacia adelante la asignación actual con todos los valores de las futuras variables que están restringidas con la variable actual, donde los valores de las futuras variables que son inconsistentes con la asignación actual son temporalmente eliminados de sus dominios; por otro lado, si el dominio de una variable futura se queda vacío, la instanciación de la variable actual se deshace y se prueba con un nuevo valor, del cual si ningún valor es consistente, entonces se lleva a cabo el backtracking cronológico. FC garantiza en cada iteración una solución parcial actual, debido a que es consistente con cada valor de cada variable futura, asimismo cuando se asigna un valor a una variable, solamente se comprueba hacia adelante con las futuras variables con las que están involucradas; así FC puede identificar antes las situaciones sin salida, por medio de la comprobación hacia adelante, y reducir el espacio de búsqueda en cada lapso de la búsqueda.

El proceso de forward checking se puede ver cómo aplicar un simple paso de arco-consistencia sobre cada restricción que involucra la variable actual con una variable futura después de cada asignación de variable.

El pseudocódigo de FC seria como sigue [1]:

a) Se selecciona una variable x_i.
b) Se le instancia un valor a_i. desde su dominio D_i. $x_i \leftarrow a_i \in D_i$.
c) Se elimina de los dominios de las variables, aún no instanciadas con un valor, aquellos valores inconsistentes con respecto a la instanciación. $x_i \leftarrow a_i$, de acuerdo al conjunto de restricciones.
d) Si quedan valores posibles en los dominios de todas las variables por instanciar, entonces:
    i. Si $i < n$ incrementar valor de i e ir a)
    ii. Si $i = n$ retornar resultado
e) Si existe una variable por instanciar, sin valores posibles en su dominio, entonces retractar los efectos de la asignación. $x_i \leftarrow a_i$,
    i. Si quedan valores por asignar en D_i ir a b)
    ii. Si no quedan valores por asignar en D_i, decrementar el valor de i si $i > 1$ e ir b), si $i = 1$ no se retorna solución.

2.3.4.2. Minimal forward checking (MFC)

Mínimal forward checking (MFC) es una versión de FC que retrasa la realización que comprueba la consistencia de FC hasta que es completamente necesario. Lo que hace MFC es comprobar si la asignación actual causa una limpieza de dominios, para lo cual es suficiente comprobar la asignación actual con los valores de cada variable futura hasta que se encuentra una que es consistente; luego, si el algoritmo ha retrocedido, vuelve atrás y lleva a cabo las comprobaciones 'perdidas'. MFC siempre lleva a cabo a la suma el mismo número de comprobaciones que FC.

2.3.5. Heurísticas.

2.3.5.1. Ordenación de Variables.

El orden en que las variables son seleccionadas para ser asignadas durante el proceso de búsqueda de la solución influirá en el resultado de este mismo. Por lo general la tendencia para seleccionar qué variable se debe considerar primero, es aquella que restringen mayormente a las demás variables [1]. El objetivo es tratar de asignar las variables que son más restrictivas para poder evitar asignaciones innecesarias y disminuir la cantidad de backstrack. Se puede dividir en dos tipos: Heurísticas de ordenación de variables estáticas y Heurísticas de ordenación de variables dinámicas.

2.3.5.1.1. Ordenación de Variables Estáticas.

Se obtiene una información preliminar de las restricciones sobre las variables, obtenido del grafo de restricciones inicial. Algunos tipos de este tipo de selección de variables son:

- Minimun Width(MW): se basa en que la ordenación de variables, corresponde a la ordenación lineal de la red que dé lugar a su menor anchura (ancho de la red). Mediante este orden se persigue que cuando se asigne una variable, sus padres estén ya asignados, de forma que las situaciones sin salida puedan identificarse antes y se reduzca el número de backtrack [7] [8].

- Maximun Degree (MD): intenta seguir el orden de variables de anchura mínima. Esta heurística selecciona primero las variables de mayor grado, es decir, con el número de variables con las que está conectada [1].

2.3.5.1.2. Ordenación de Variables Dinámicas.

Pueden cambiar el orden de selección de las variables de forma dinámica durante el proceso de búsqueda, cada vez que requiere la instanciación de una variable. Generalmente se utilizan en algoritmos look-backward. La ordenación se basa en las instanciaciones ya realizadas y en el estado de la red en cada momento de la búsqueda.

- Minimum Remaining Values (MRV): Selecciona la variable con el dominio de instanciación más pequeño [1].

- Maximun Cardinality (MC): selecciona la primera variable arbitrariamente y después en cada paso, selecciona la variable que está relacionada con el mayor número de variables ya instanciadas.

2.3.5.2. Ordenación de Valores.

La idea básica es seleccionar el valor de la variable actual que más probabilidad tenga de llevarnos a una solución, es decir, identificar la rama del árbol de búsqueda que sea más probable que obtenga una solución.

- Min Conflicts: ordena los valores de acuerdo a los conflictos que generan con las variables aún no instanciadas. Esta heurística asocia a cada valor a de la variable actual, el número total de valores en los dominios de las futuras variables adyacentes con la actual que son incompatibles con a [1].

- Max domain-size: selecciona el valor de la variable actual que deja el máximo dominio en las variables futuras.

- Weighted max domain size: se basa en el número de futuras variables que tienen el mayor tamaño de dominio.

- Point Domain Size: asigna un peso a cada valor de la variable actual dependiendo del número de variables futuras que se quedan con ciertas tallas de dominios

- Max promise: se selecciona el valor que tiene el mayor número de valores que hay compatibles en cada variable adyacente, de los valores. Este resultado representa una cota superior del número de soluciones diferentes que pueden existir después de que el valor se asigne a la variable.

.

CAPÍTULO 3

3. Procesos de Adaptación.

En este capítulo se mostrarán los distintos enfoques que permiten realizar una adaptación de los problemas de satisfacción de restricciones. Estos enfoques ayudarán a resolver de forma dinámica el problema dependiendo del estado del proceso de búsqueda de una solución.

3.1. Mecanismo de Adaptación de Restricciones.

El mecanismo de Adaptación de Restricciones o ACE de sus siglas en inglés para Adaptive Constraint Engine es un proceso de aprendizaje para obtener mejor solución a los problemas de restricciones (CSP). Este proceso cuenta con los denominados Asesores, los cuales son un conjunto de métodos primitivos que caracterizan el comportamiento de la solución [9].

ACE se basa en una arquitectura subyacente Asesor basado llamada FORR (FOr the Right Reasons), el cual es una arquitectura para el desarrollo de conocimiento de múltiples heurísticas. Para tomar una decisión FORR combina la salida de un conjunto de procedimientos llamados consejeros, los cuales representan el principio general para apoyar en el conocimiento, esto se basa en que las personas integran una variedad de estrategias para lograr una solución a un problema. En cada intento de solución se va adquiriendo experiencia [9].

ACE ha demostrado que no sólo puede redescubrir heurística clásica, sino también hacer nuevos descubrimientos de igual o mayor valor. En general, ACE proporciona un banco de pruebas de gran alcance para explorar la naturaleza del proceso de búsqueda con mucha más facilidad y sutileza que ha sido hasta ahora disponibles.

3.2. Búsqueda Autónoma

Una búsqueda autónoma se define como la habilidad de modificar ventajosamente los componentes internos cuando las fuerzas y oportunidades externas cambian. Corresponde a un especial caso de adaptación en donde se busca mejorar el rendimiento de la solución adaptando su estrategia de búsqueda [10]. Los componentes internos y las fuerzas externas son:

- Componentes internos: son los algoritmos que intervienen en el proceso de búsqueda heurística, los mecanismos de inferencia.
- Fuerzas externas: informaciones recogidas durante la evolución de este proceso de búsqueda

Las soluciones autónomas nos referimos al control que contienen en su proceso de búsqueda. También incluye un proceso en donde, como se dijo anteriormente, algunos de sus componentes cambia y el comportamiento después de algunas función de soluciones. Existen dos tipos de adaptaciones:

- Auto adaptativa: Comúnmente el proceso auto adaptativa está estrechamente ligado con el proceso de búsqueda de una solución y usualmente existe un costo bajo, es decir el algoritmo observa su propio comportamiento actualizando sus componentes.

- Supervisada: Trabaja sobre un alto nivel. Lo que quiere decir que a diferencia del auto adaptativo, no tiene relación directa con el proceso de búsqueda. Puede ser visto como un monitor que observa la búsqueda y la analiza modificando los componentes con el fin de adaptarlo.

3.3. Metaheurísticas

El termino Metaheurística es introducido en [11]. El método de búsqueda Metaheurística puede ser definida como un nivel superior de metodología generales que pueden ser usadas como guía en el diseño de las estrategias de las heurísticas subyacentes para resolver problemas específicos de optimización. La palabra *heurística* procede de un término griego que significa hallar o inventar, en tanto la palabra meta también procede de un término griego que significa “metodología de nivel superior”.

Las Metaheurísticas proveen una solución aceptable en relación al tiempo para solucionar un problema difícil y complejo en la ciencia y la ingeniería. Esto ha producido el notable crecimiento de las investigaciones realizadas sobre este ámbito. A diferencia de los algoritmos de optimización exacta, las metaheurísticas no garantizan la optimización de las soluciones obtenidas [12].

Las principales aplicaciones donde están presente las metaheurísticas cae en un gran número de áreas, algunas de ellas son [12]:

- Diseño de ingeniería, optimización de topologías y estructuras electrónicas, aerodinámica, fluidos dinámicos, telecomunicaciones, automotriz, y robótica.
- Mecanismos de aprendizajes y minería de datos en bioinformática y biología computacional y finanzas.

- Modelo de sistemas, simulaciones e identificación en química, física y biología; procesamiento de control, señal y de imagen.
- Problemas de planeación de enrutamientos, planificación de robots, problemas de programación y producción, logística y transporte, manejo de cadenas de suministros, entorno, etc.

El principal criterio de clasificación que pueden ser usado en Metaheurísticas es [12] [13]:

- Inspiradas en la naturaleza versus las no inspiradas en la naturaleza: las principales metaheurísticas están inspiradas en el proceso natural.
- El uso de memoria versus con los métodos sin memoria: Algunas Metaheurísticas no utilizan memoria, esto es, que no contienen información durante la búsqueda. Mientras que otras sí utilizan cierta información durante el proceso de búsqueda, ya sea esta de corto plazo o de largo plazo.
- Determinísticas versus estocásticas. En el caso de métodos Determinísticos, usando la misma solución inicial se obtendrá la misma solución final, mientras que en métodos estocásticos se pueden obtener diferentes soluciones finales con una misma solución inicial.
- Basados en poblaciones versus los basados búsqueda de una sola solución: los basados algoritmos de una sola solución manipulan y transforman una solución una única solución en la búsqueda, mientras que en los algoritmos basados en la población (por ejemplo, enjambre de partículas, algoritmos evolutivos) toda una población de soluciones se desarrolla durante el proceso de búsqueda.
- Iterativo versus codiciosos: En los algoritmos iterativos, la solución va siendo transformada por cada iteración según algún operador, mientras que los métodos codiciosos, se inicia desde una solución vacía hasta generar una solución completa.

3.3.1. Intensificación y Diversificación.

Cada enfoque metaheurístico debe estar diseñado con el objetivo de realizar una exploración en un espacio de búsqueda de manera eficaz y eficiente. La búsqueda realizada por una metaheurística debe ser un enfoque inteligente tanto para explorar intensamente áreas del espacio de búsqueda de soluciones de alta calidad, y para trasladarse a zonas inexploradas del espacio de búsqueda, cuando sea necesario. Los conceptos para alcanzar estos objetivos son hoy en día llama la intensificación y diversificación.

La principal diferencia entre la intensificación y la diversificación es que la intensificación su búsqueda se centra en la indagación de los vecinos que contengan las mejores soluciones, mientras que la diversificación en cambio alienta el proceso de búsqueda para examinar las regiones no visitadas y para generar soluciones que difieren en varios aspectos importantes de las que había visto antes [13].

3.4. Hiperheurísticas

El término "heurística" se utiliza a veces para referirse a un algoritmo de búsqueda general y se utiliza a veces para referirse a un proceso de decisión particular, asentada en una estructura de control repetitiva. En [14] hacen mención a que no se puede determinar que los algoritmos heurísticos, en un espacio de búsqueda finito, tienen el mismo rendimiento medio, es decir, que un tipo de algoritmo heurístico podría ser útil (optimo) para un problema que otro algoritmo. Esto llevándolo a un modo más genérico podríamos decir que una heurística tiene fuerzas y debilidades que dependerán del tipo de problema que se desea solucionar. La Hiperheurísticas intenta combinar distintas heurísticas para compensar las debilidades de otra.

Una manera sencilla de demostrar lo anterior como se muestra a continuación.

```
Si el problema es del tipo A entonces
        Ejecutar heurística A
Sino el problema es del tipo B entonces
        Ejecutar heurística B
Sino el problema es….
```

El ejemplo anterior es un caso extremo en donde por cada tipo de problema conocido es asignada la heurísticas conocidas que más se acomodan. La idea principal en la Hiperheurística es utilizar los miembros de un conjunto de heurísticas conocidas y comprendidas razonablemente para transformar el estado de un problema tratando de asociar las heurísticas con las condiciones problema, aplicando una heurística diferente en diferentes partes o fases del proceso de solución. Muchos de estos procesos son ayudados por los denominados algoritmos evolutivos.

3.5. Algoritmos Evolutivos.

Los algoritmos evolutivos son métodos sistemáticos para la resolución de problemas de búsqueda y optimización que aplican a estos los mismos métodos de la evolución biológica. Las características principales de los algoritmos evolutivos son [15]:

- Algoritmos Evolutivos utilizan el proceso de aprendizaje colectivo de una población de individuos. Por lo general, cada individuo representa un punto de búsqueda en el espacio de posibles soluciones a un problema

dado. Además, los individuos también pueden incorporar información adicional, por ejemplo, los parámetros de la estrategia del algoritmo evolutivo.

- Los descendientes de los individuos son generados por procesos aleatorios destinados al modelo de mutación y recombinación. La mutación corresponde con una auto-replicación errónea de los individuos (por lo general, pequeñas modificaciones son más propensas que las más grandes), mientras que la recombinación intercambia información entre dos o más individuos existentes.

- Se pueden evaluar los individuos en su entorno, dándoles una medida de calidad o idoneidad. Esto es a través de la comparación entre los mismos individuos del grupo.

3.5.1. Configuración de Parámetros.

El rendimiento de los algoritmos evolutivos está determinado por la configuración de parámetros tales como el tamaño de la población y número de operaciones. Es común que estos parámetros sean adaptados a través de pruebas error, lo que requiere muchas repeticiones, haciendo difícil encontrar una mejor solución.

Según lo anterior existe una taxonomía de configuración de parámetros, donde los parámetros se dividen en tres ramas, según el grado de autonomía de las estrategias. Por un lado existe el ajuste de parámetros, que son los antes de la ejecución, y por el otro lado están los de control de parámetros, que son durante la ejecución. En esta última existen tres subniveles los cuales son [10]:

- Determinísticos: los parámetros son cambiados a través de reglas determinísticas, en otras palabras cambian por un esquema ya determinado.
- Adaptativas: los parámetros son cambiados según el estado de la búsqueda.
- Auto-Adaptativas: los parámetros se codifican en los individuos con el fin de desarrollar conjuntamente con las otras variables del problema.

La siguiente figura muestra un esquema de la taxonomía de configuración de parámetros.

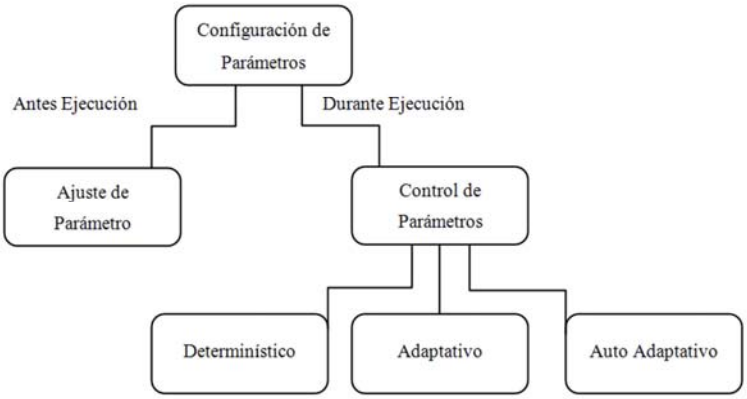


Ilustración 1: Taxonomía de control de parámetros

CAPÍTULO 4

4. Optimización por Enjambre de Partículas

Optimización por Enjambre de Partículas (PSO por su nombre en inglés Particle Swarm Optimization) es una técnica basada en el paradigma de inteligencias de poblaciones o enjambres (Swarm Intelligence). Fue introducido por Kennedy y Eberhart en 1995, quienes desarrollaron métodos simples que permitían optimizar eficientemente funciones matemáticas no lineales [16]. En PSO las soluciones candidatas de una población son las llamadas partículas que coexisten y se desarrollan de forma simultánea sobre la base de intercambio de conocimientos con el resto de las partículas del enjambre [17].

La optimización por enjambre de partículas (PSO) pertenece a las técnicas estocásticas de cálculo evolutivo, considerada unas de las representantes de la rama de la inteligencia de enjambre (Swarm Intelligence), al igual que la técnica de optimización por colonia de hormigas (ACO), las cuales son los métodos más populares y utilizados en el área de la inteligencia computacional. Ambos tratan de imitar los comportamientos sociales de una comunidad a partir de la interacción de los individuos entre sí y con su entorno.

En [18] proponen una teoría que se basa en el modelo de la cultura adaptativa y enjambre de partículas, donde el proceso de adaptación cultural cuenta con un componente de alto nivel, que aparece en la formación de patrones a través de las personas y la capacidad de resolver problemas, y un componente de bajo nivel que tienen que ver con el comportamientos reales y probablemente universal de las personas que se resumen en tres principios:

- Evaluar: son los estímulos a que se consideran positivas o negativas. El aprendizaje no puede ocurrir a menos que los organismos puedan evaluar, donde se distinguen las características del ambiente que se atraen o repelen (reconocer el bien y mal), es decir, la capacidad de que se le da un organismos de mejorar la evaluación promedio de su ambiente.

- Comparación: las comparaciones sirve como una especie de motivación para aprender y cambiar. En un modelo de cultura de adaptación y de enjambre de partículas los individuos, se comparan con sus vecinos midiendo el criterio e imitar sólo aquellos vecinos que son superiores a él mismo.

- Imitar: es la forma efectiva de aprender a hacer las cosas, que no todos los seres tienen la capacidad de realizar. Esta capacidad está formada por tomar la perspectiva de otro individuo, no sólo imitando su comportamiento, sino que darse cuenta de su finalidad, la ejecución de la conducta cuando es apropiado.

Estos tres principios pueden combinarse incluso en los programas de series sociales computaciones, lo que permite adaptarse a complejos problemas. Estos conocimientos son la base de lo que denominamos como Optimización por enjambres de partículas.

Una de las ventajas del algoritmo de PSO es su rápida convergencia, en comparación con otros algoritmos de optimización global como los algoritmos genéticos (GA), colonia de hormigas, enfriamiento simulado, y otros algoritmos de optimización global [19].

4.1. Comportamiento de la Población

Las ideas iniciales sobre enjambres de partículas fueron de Kennedy (psicólogo social) y Eberhart (ingeniero eléctrico) los cuales realizaron un fundamento a la producción de inteligencia computacional mediante una exploración simple de la analogía del comportamiento social y en menor grado de las capacidades cognitivas individuales, como por ejemplo las bandadas de pájaros en busca de alimento [18].

El enjambre trabaja según los movimientos de las partículas, las cuales se posicionan en un espacio de búsqueda de un problema o función, y cada partícula evalúa la función objetivo en la posición actual. Las partículas determinan el movimiento en su espacio de búsqueda, mediante la combinación de algunos aspectos de su actual historia y los mejores lugares de uno o más miembros del enjambre con ciertas perturbaciones aleatorias, dicho de otra forma, una partícula modifica su velocidad para encontrar una solución mejor (posición) mediante la aplicación de su propia experiencia de vuelo (la memoria, es decir, teniendo mejor posición en los vuelos anteriores) y la experiencia de las partículas vecinas (es decir, la mejor solución encontrada de la población).

Cada individuo en el enjambre de partículas se compone de tres vectores de largo D, donde D es la dimensión del espacio de búsqueda. La forma en que las partículas actualizan sus posiciones y velocidades se muestra a continuación:

$$V_{id}^{t+1} = WV_{id}^{t} + \underbrace{C_1 R_1 \left(P_{id} - X_{id}^{t} \right)}_{\text{Componente Cognitiva}} + \underbrace{C_2 R_2 (P_{gd} - X_{id}^{t})}_{\text{Componente Social}} \quad (1.1)$$

$$X_{id}^{t+1} = X_{id}^{t} + V_{id}^{t+1} \quad (1.2)$$

Donde:

- V_{id}^{t} es la velocidad de la partícula i y esta representa la distancia viajada desde la actual posición.
- V_{id}^{t+1} es la velocidad futura o bien actualizada.
- W es una constante de amplitud de inercia, en otras palabras se puede representar como la importancia de la velocidad previa.
- X_{id}^{t} representa la posición de la partícula. También puede ser considerado como un conjunto de coordenadas describiendo un punto en el espacio. Por cada iteración del algoritmo, la actual posición es evaluada una solución al problema. Si la posición es mejor que cualquier otra hasta ese momento es almacenado en el vector P_{id}.
- X_{id}^{t+1} es la posición futura o bien la posición actualizada.
- P_{id} es la mejor solución local, llamado común mente “pbest”. Esta Variable es utilizada como comparación durante el proceso de búsqueda de una solución.
- P_{gd} es la mejor solución global, también llamado “gbest”.
- C_1 y C_2 representan las contantes de aceleración de impulso de las partículas a la mejor posición.
- R_1 y R_2 son números aleatorios que van entre el rango [0,1] y que intentan emular el comportamiento impredecible de un enjambre y así tener un valor estocástico.

Un problema que suele suceder, es el aumento o disminución constante de las velocidades de las partículas, con tal motivo es común que se utilicen valores máximos y mínimos [20]. Este rango de velocidad va a estar determinado por $[-V_{max,}V_{max}]$. Si el valor de V_{max} es demasiado grande las partículas pueden sobrepasar e ignorar continuamente la zona con la solución global. En el extremo opuesto, si $-V_{max,}$ toma valores extremadamente pequeños las partículas explorarán el espacio de soluciones muy lentamente y podrán quedar atrapadas alrededor de soluciones locales, incapaces de librarse de la base de atracción.

4.1.1. Actualización Asíncrona y Síncrono.

4.1.1.1. Asíncrono

Este tipo de modelo, todas las partículas comparten la información acerca de la mejor solución global., en otras palabras, todas las partículas se mueven en paralelo, donde en cada iteración se evalúan y se actualiza su memoria local P_{id} y el conocimiento global P_{gd}, actualizando su nueva posición, el enjambre completo tomando como referencia esta información.

4.1.1.2. Síncrono.

En el modelo síncrono en cada iteración k, la partícula i-ésima se desplaza hacia un nuevo punto $X_i(k+1)$ utilizando la información de $P_{id}(k)$ y P_{gd}, que son mejor posición local y global respectivamente actualizada previamente por la partícula previa. Luego la partícula actual, evalúa su calidad y si es el caso actualiza las variables $P_{id}(k+1)$ y P_{gd}.

4.2. Algoritmo PSO

Como se logra observar, el objetivo principal de PSO es actualizar las posiciones de cada partícula, para así encontrar una partícula que sea mejor que las demás. Con este esquema el algoritmo de PSO seria básicamente lo siguiente [20]:

a) Inicializar el arreglo de la población de las partículas con posiciones aleatorias y velocidades, con un tamaño de largo D.
b) Para cada Partícula, evaluar la función fitness deseada para las D variables
c) Comparar la función fitness con *pbest*. Si el valor actual es mejor que *pbest* entonces asignar a *pbest* el actual valor.
d) P_{id} asignarle la actual posición en el espacio de tamaño D.
e) Identificar la partícula vecina con el mejor existo hasta ahora y asignar su índice a la variable P_{gd}
f) Cambiar la velocidad y la posición de la actual partícula dada la ecuación de velocidad mostrada anteriormente.
g) Verifica el o los criterios de la solución.
h) Repetir hasta el máximo de iteraciones desde el paso b)

4.3. Peso de Inercia.

El concepto de Peso de inercia se introdujo con la finalidad de controlar los valores posibles de la variable $Vmax$, la cual, como se mencionó anteriormente, sirve como una restricción de control en la habilidad de exploración de una enjambre de partícula, para la velocidad máxima. Si $Vmax$ mantiene un valor elevado facilita una exploración global, mientras que un valor pequeño lo hace para una explotación local, es decir que el peso de inercia sirve para tener un mejor control en la exploración y explotación [20] [21]. Un valor adecuado para el peso de inercia, por lo general, proporciona un equilibrio entre las capacidades de exploración global y local y, en consecuencia resulta en una reducción del número de iteraciones necesarias para encontrar la solución óptima [19].

En la ecuación, el peso de inercia está ilustrado por la variable w. En esta ecuación se puede observar que al utilizar un peso de inercia se elimina de cierto modo la problemática de pensar en un rango fijo para la variable $Vmax$, ya que el peso de inercia establece un rango dinámico, es decir, el peso de la inercia (w) controla el movimiento de la partícula mediante la ponderación de la contribución de la velocidad anterior [22].

4.4. Topologías.

La topología de un enjambre va estar dada por la cercanía en el espacio de búsqueda y la forma en que las partículas se comunican.

4.4.1. Topología Global.

La topología global se puede ver como todos las partículas tienen relaciones entre si, obteniendo de esta manera la información de sus vecinos. Si bien esta topología puede ser visualizado como un grafo donde las partículas están completamente conectadas, en la práctica solo se necesita tener el mejor resultado y el índice de la partícula que lo encontró. Es decir, una partícula, está influenciada por el mejor vecino de la población entera [20].

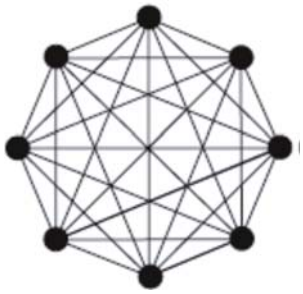


Ilustración 2: Topología Global.

4.4.2. Topología Local.

En la topología local [23] cada partícula queda influenciada por un número determinado de vecinos adyacentes N haciendo que quede apartada en cierto sentido de las partículas más alejadas y seleccionado los N vecinos más próximos. Esto hace que el enjambre sea particionado en pequeños grupos que exploran diferentes regiones del espacio de búsqueda. En este caso la variable utilizada como vector de soluciones globales P_{gd} será remplazada por el vector P_{ld} que indicará el vector de las N particulas vecinas. De este vector la partícula elegirá aquella variable con mejor fitness para actualizar sus vectores de velocidad y posición. Si se selecciona un N demasiado elevado el comportamiento será parecido a la topología global.

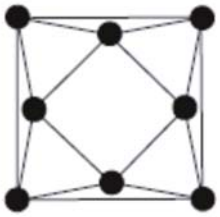


Ilustración 3: Topología Local con N=4

4.5. Enjambre Cuántico.

PSO ha sufrido una gran cantidad de cambios desde su creación. Una de las variantes realizadas a PSO es la aplicación de las leyes de la mecánica cuántica al comportamiento del enjambre. Esto es denominado Quantum PSO (QPSO). Algunas variantes de QPSO incluyen mutación basadas en PSO [24], donde la mutación se aplica mejor a la mbest (mejor media) y gbest (el mejor global) de las posiciones de la partícula.

En el modelo cuántico de PSO, el estado de una partícula se representa por una función de onda (wavefunction), en vez de una posición y velocidad. El comportamiento dinámico de la partícula es ampliamente divergente de una partícula en los sistemas de PSO tradicional en que los valores exactos de *x* y *v* (posición y velocidad) no se pueden determinar de forma simultánea, esto de acuerdo con el principio de incertidumbre y en donde la trayectoria no tiene sentido.

El movimiento de las partículas va estar determinado por las siguientes ecuaciones interactivas [25]:

$$X_{id}^{t+1} = p + \beta * \left| mbest - X_{id}^{t} \right| * \ln\left(\frac{1}{u}\right), if \quad \kappa \geq 0.5 \tag{1.3}$$

$$X_{id}^{t+1} = p - \beta * \left| mbest - X_{id}^{t} \right| * \ln\left(\frac{1}{u}\right), if \quad \kappa < 0.5 \tag{1.4}$$

Donde los parámetros propios de QPSO son:

$$mbest = \frac{1}{N}\sum_{i=1}^{N} P_i = \left(\frac{1}{N}\sum_{i=1}^{N} P_{i1}, \frac{1}{N}\sum_{i=1}^{N} P_{i2}, ..., \frac{1}{N}\sum_{i=1}^{N} P_{id} \right) \tag{1.5}$$

$$p = \left(c_1 * p_{id} + c_2 * p_{gd} \right) \Big/ (c_1 + c_2) \tag{1.6}$$

$mbest$ es el valor medio de las mejores posiciones de todas las partículas de la población (Mean Best Position), u, k, c_1 y c_2 son valores aleatorios de forma distribuida uniformemente entre [0,1]. β es denominado coeficiente de Contracción y Expansión el cual que se puede ajustar para controlar la velocidad de convergencia del enjambre.

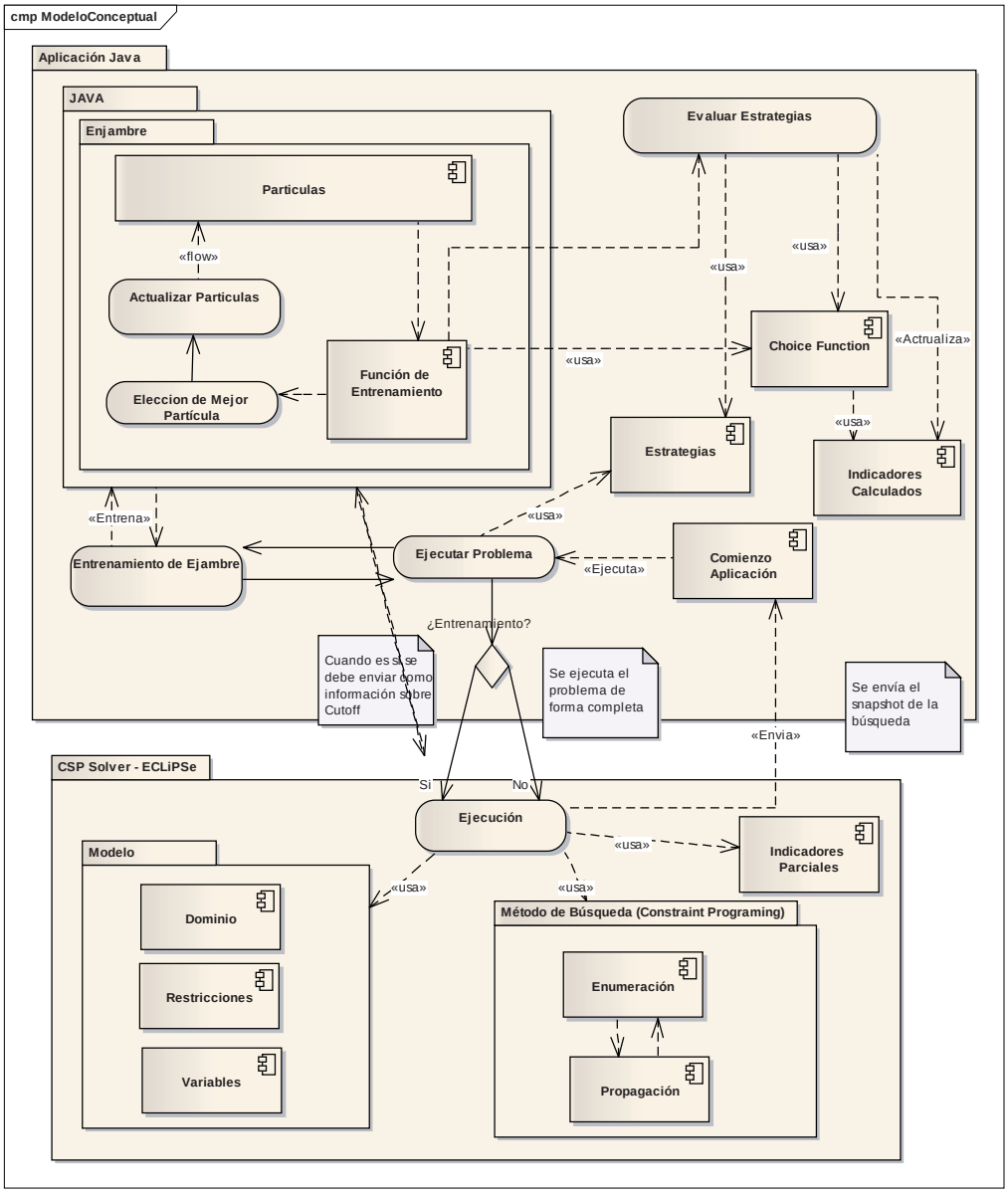
4.5.1. Algoritmo PSO Cuántico

a) Inicializar el Enjambre.
b) Realizar hasta el valor de término.
    i. Calcular $mbest$ según la ecuación (1.5)
    ii. Actualizar posición de las partículas usando las ecuaciones *(1.3)* y (1.4) según corresponda
    iii. Actualizar *P*
    iv. Actualizar P_g

CAPÍTULO 5

5. Propuesta.

Como se explicó en las secciones anteriores, los enfoques heurísticos y metaheurísticas están desarrollados para un problema en particular, pero estos problemas pueden no ser de aplicación general a los dominios de otro problema. Además, los enfoques heurísticos y metaheurísticas tienden a necesitar una experiencia considerable tanto en el dominio del problema y las técnicas adecuadas de heurística [26]. Es por esto que se propone una estructura multinivel (Hiperheurística) para la resolución de problemas. Una manera de ilustrar el objetivo de la investigación se puede observar en la siguiente figura.



[27]

Ilustración 4: Esquema general de propuesta investigativa

En la figura anterior se ve que la idea principal es realizar un mecanismo multinivel de elección de estrategias. En donde el proceso de búsqueda de solución al modelo de satisfacción de restricciones será implementado mediante ECLiPSe [28]. ECLiPSe es un sistema de programación de código abierto para el desarrollo de aplicaciones con programación con restricciones. En ECLiPSe se implementarán los modelos de CSP para los distintos problemas, junto con las estrategias de búsqueda de solución a estos problemas.

ECLiPSe se comunicará con el componente desarrollado en el lenguaje de programación JAVA, enviando los resultados del proceso de búsqueda y también recibiendo desde JAVA las decisiones tomadas respecto a las estrategias que se deben utilizar. Este envío de información hacia el solver (ECLiPSe) corresponderá a lo denominado como Estrategia de Enumeración y que gracias a observaciones continuas sobre el estado de la resolución del problemas, proporcionarán la información para determinar el comportamiento de la búsqueda sobre la estrategia de Enumeración que se está ejecutando. Con dicha información se podrá tomar la decisión si mantener la Estrategia o bien seleccionar otra que posiblemente sea más eficiente en la búsqueda de una solución.

Esta información u observaciones sobre el proceso de búsqueda corresponden a lo denominado como Indicadores, Snapshots o bien monitores que evidencian el comportamiento de la resolución del problema. En la siguiente figura demuestra este proceso de actualización de indicadores.

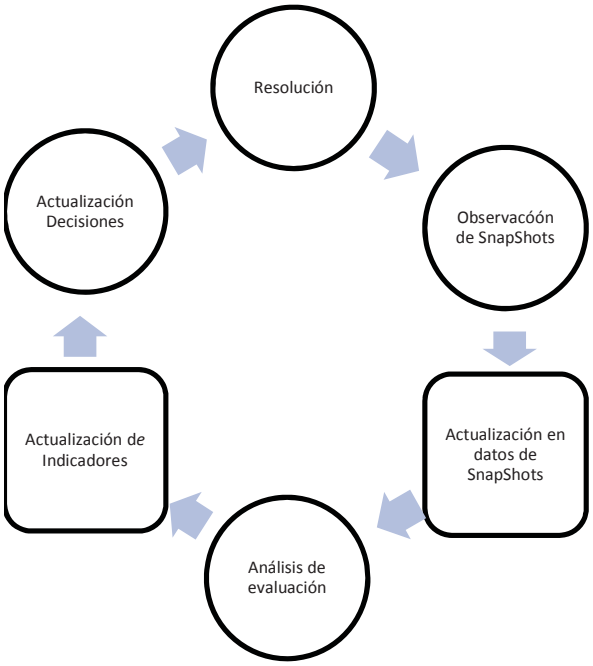


Ilustración 5: Estrategia de Enumeración

Teniendo los Indicadores correspondientes del estado resolutivo del problema, la forma en que esta información decidirá si una estrategia es mejor que otra, será a través de una función de elección o Choice Function, que será explicada más adelante.

En tanto la componente que corresponde al enjambre de partículas, será entrenada con una función fitness que será uno de los mismos indicadores y considerando un criterio de corte en la resolución del problema de satisfacciones en ECLiPSe. Este criterio de corte lo denominaremos cutoff, el cual representará cual será el tiempo o la condición

de término en la búsqueda de la solución, un ejemplo del criterio de corte podría ser, el porcentaje de niveles alcanzado en el árbol de búsqueda, porcentaje de variables instanciadas, porcentaje de pasos alcanzados, entre otras. Se podrá notar que el valor de cutoff intenta ser un valor representativo, independiente del problema que se desea ejecutar, esto evitará realizar cortes prematuros o demasiados extensos para un determinado problema. El objetivo del entrenamiento del enjambre es buscar la mejor partícula que indicará cuales son los mejores parámetros para la función de elección de estrategias, la cual elegirá que estrategia se adecua mejor al problema.

5.1. Estructura Multinivel.

En [29] se propone una estructura multinivel, con el objetivo de que un algoritmo genético (GA), sea capaz de simular la evolución de algoritmo de bajo nivel en la solución de un problema. En nuestro caso la idea sería que el algoritmo del nivel superior PSO, sea capaz de encontrar cuales son los mejores parámetros que se ajusten a un problema, realizando las combinaciones necesarias entre estos mismos. La forma en que se evaluará cuáles son los parámetros con un mejor rendimiento, va estar determinado por una función de elección, que se describirá más adelante.

Esta función de elección nos permitirá realizar un ranqueo de aquellas estrategias de búsqueda de solución que tienen un mejor comportamiento al momento de resolver un problema y sean las más adecuadas en un determinado momento. Esto, como se mencionó anteriormente se realizará con la ayuda de los Indicadores, que nos permitirán obtener evidencias del comportamiento de una determinada estrategia que está resolviendo el problema, indicando si es necesario realizar un cambio de estrategia o bien mantenerla.

5.1.1. Estrategias de Enumeración.

Como se mencionó anteriormente, al seleccionar una estrategia repercutirá instantáneamente en la forma y en el desempeño que la búsqueda de una solución tendrá. Las estrategias que a continuación se muestran están basadas en la Heurísticas vistas en la sección Heurísticas del Capítulo 2 de este documento. En donde tendremos heurísticas de selección de variables y heurísticas de selección de valor. La combinación de ellas nos dará el número de estrategias que se utilizarán que son [30]:

Nº Estrategia	Heurística de selección de Variables	Heurísticas de Selección de Valor
1	Fir - Escoge la primera variable de la lista	Ido : Comienza con el elemento más pequeño del dominio
2	AMRV : Escoge la variable con el dominio más grande	Ido : Comienza con el elemento más pequeño del dominio
3	MRV : Escoge la variable con el dominio más pequeño	Ido : Comienza con el elemento más pequeño del dominio
4	Occurence : Escoge la variable con el mayor número de restricciones	Ido : Comienza con el elemento más pequeño del dominio
5	Fir - Escoge la primera variable de la lista	IDM : Comienza con el elemento mayor del dominio
6	AMRV : Escoge la variable con el dominio más grande	IDM : Comienza con el elemento mayor del dominio
7	MRV : Escoge la variable con el dominio más pequeño	IDM : Comienza con el elemento mayor del dominio
	Occurence : Escoge la variable con el mayor número de restricciones	IDM : Comienza con el elemento mayor del dominio

Tabla 1 - Estrategias Utilizadas

5.1.2. Indicadores.

Como su mismo nombre lo indica, los indicadores nos proveen información sobre cómo el proceso de búsqueda de la solución se encuentra. Algunos de los principales indicadores son [9]:

- Degree: Número de vecinos en el gráfico de restricción.
- Forward Degree: Numero de vecinos invalorables.
- Backward Degree : Número de vecinos de valoración.
- Domainn/Degree : Relación entre el tamaño del dominio y el proceso.
- Domainn/Forward Degree: Relación entre el tamaño de dominio con el número de vecinos no evaluados.
- Common Value: Número de variables ya asignadas a este valor.
- Options Value: Número de limitaciones de la variable seleccionada que incluyen este valor.
- Backtracks: Muestra la cantidad de malas decisiones hechas durante la búsqueda de la solución, en otras palabras, los cálculos o las decisiones implementadas nos conduzca a una solución.

5.1.2.1. Cálculo de indicadores a utilizar

En las siguientes tablas se mostrará el cálculo y las descripciones de los indicadores que se utilizarán para realizar los experimentos.

- Indicadores Bases: Estos indicadores ayudarán a realizar el cálculo de otros indicadores derivados. Estos serán calculados en su mayoría por ECLiPSe.

Código	Descripción	Cálculo	Referencia
STEP	Cada vez que una variable es instanciada por enumeración	$\sum STEP$	
TV	Número de variables del problema	Lenght(Allvars,N)	
VU	Número de variables que no son instanciadas	Lenght(Rest,N)	
SB	Shallow Bracktrack: cuando se trata de instanciar un valor a una variable sin éxito, entonces este recorre al próximo valor.		[31] [32]
SS	Espacio actual de búsqueda	$\prod(Dom_i)_t$	[5]
SS_pr	Espacio actual de búsqueda previo	$\prod(Dom_i)_{t-1}$	[5]
B	Bracktracks, Si la actual variable lleva a una ruta sin salida, entonces el algoritmo lleva a la variable previa		[33] [34]
N	Número de nodos visitados		[33]

Tabla 2 – Indicadores Base

- Indicadores Procesados: Estos indicadores serán calculados a partir de los indicadores bases antes descritas:

Código	Descripción	Cálculo	Referencia
VF	Número de variables instanciadas por enumeración y propagación.	(TV - VU)	[34] [35]
VFES	Número de variables instanciada por numeración en cada paso	1	[5] [35]
TAVFES	Número total acumulado de variables instanciadas por enumeración en cada paso	$\sum VFE$	[33] [35]
VFPS	Variables instanciada por propagación en cada paso	$VFP_{t-1} - VFP_t - 1$	[33]

TVFP	Número total Acumulado de variables instanciadas por propagación en cada paso	$\sum VFPS$	[33]
TSB	Número total de Shallow Backtrack	$\sum SB$	[31]
D_pr	Profundidad previa en el árbol de búsqueda	$(TV - VU)_{t-1}$	[33]
D	Profundidad actual en el árbol de búsqueda	$(TV - VU)_t$	[33] [35]
DMAX_pr	Profundidad máxima previa en el árbol de búsqueda	$Maximo(DMAX)_{t-1}$	[33]
LN1	Es la profundidad máxima actual menos la profundidad máxima previa. Variación de la profundidad previa y actual	$(DMAX - DMAX\_or)$	[33]
LN2	Es la profundidad actual menos profundidad previa. Cuando resulta positivo significa que el nodo actual está más profundo que el anterior	$(D - D_{pr})$	[33]
LN3	Reducción del espacio de búsqueda. Si es positivo, el espacio de búsqueda es menor que el de la captura.	$(SS_{pr} - SS/SS_{pr})$*100	[33]
PVFES	Porcentaje de Variables instanciadas por enumeración en cada paso	$\frac{VFES}{TV} * 100$	[35]
TAVFESP	Porcentaje del total acumulado de variables instanciadas por enumeración en cada paso sobre el total de variables.	$\frac{TVFES}{TV} * 100$	[35]
PVFPS	Porcentajes de variables instanciada por propagación en casa paso.	$\frac{VFPS}{TV} * 100$	[35]
TAVFPSP	Porcentaje del total acumulado de variables instanciadas por propagación en cada paso sobre el total de variables	$\frac{TAVFPS}{TV} * 100$	[35]
TRASH	Thrashing	$(D_{pr} - VPF_{pr})$	[35]
DB	Si la actual variable lleva a una ruta sin salida, entonces el algoritmo lleva a la variable previa, y cuenta el retroceso que existió	Si D – d_pr = 0 entonces se incrementa en 1, Si D<d_pr, entonces se incrementa en d_pr-D +1 Si no se mantiene constante.	

Tabla 3 – Indicadores Procesados

5.1.3. Función de Elección.

Como se mencionó anteriormente, la Función de Elección o Choice Function, nos permite ranquear las estrategias seleccionadas para la solución del problema con la ayuda de los indicadores [27].

En [36] definen la función de elección como un enfoque Hiperheurístico basado en ranking estadístico de heurísticas de bajo nivel. En este método, la información histórica sobre la evolución reciente de la heurística de nivel bajo es acumulado en una función de elección. Para objetivos de este documento esta función de elección está basada en los valores de los indicadores antes descritos, considerando, además un parámetro de importancia para cada indicador.

La función de elección para este trabajo tendrá la siguiente forma.

$$CF = \sum_{i=1}^{n} \left(\alpha_i I_i + \beta_i H_i \right) \tag{1.7}$$

Donde:

- n: Es la cantidad de indicadores que se desean observar.
- α_i: Es el parámetro de elección. En otras palabras es el parámetro de control sobre la importancia del indicador.
- I_i: Indicador observado desde el proceso de búsqueda de solución.
- β_i: Factor de relevancia del valor histórico del indicador I_i
- H_i: Valor histórico del indicador I_i

Como se mencionó anteriormente la finalidad del proceso multinivel será encontrar los mejores parámetros α_i, que determinarán la importancia del indicador I_i . Con esto dependerá tanto de la relevancia de los valores para cada indicador para indicar que estrategia utilizar, a través de un ranking para mantener que estrategia mantiene el mejor valor.

La idea del factor β_i, es indicar la relevancia del valor de un indicador durante el transcurso del tiempo, es decir qué importancia se le da a los valores acumulados (histórico) respecto del valor presente. Esto basado en [37] donde se menciona que este factor ayudará a diversificación de los elementos (estrategias), favoreciendo a aquellas estrategias que aún no han sido utilizadas recientemente.

Manteniendo el mismo enfoque que la función de elección anterior, también se utilizará una función que mantendrá un suavizado exponencial. Está dada por la siguiente ecuación:

$$CF = \sum_{i=1}^{n} \left(\beta_i (\alpha_i I_i) + (1 - \beta_i) H_i \right) \tag{1.8}$$

La principal diferencia sobre la primera función de elección es que en vez de castigar a medida que el factor de β_i este mas cerca de 0 este será favorecido de forma mayor. Es decir se considerará más importante el valor acumulado (histórico) que el valor actual del indicador I_i.

5.2. Enjambre.

Según lo estudiado anteriormente sobre PSO, la idea para esta investigación es crear un cúmulo de partículas, en el cual cada una de ellas representen los valores para los parámetros α_i. Donde la dimensión de la partícula será de tamaño n, siendo n la cantidad de indicadores considerados para la función de elección.

Por tanto las partículas del enjambre tendrán la siguiente forma:

α_1	α_2	$\cdots$	α_{n-1}	α_n

Ilustración 6: Partículas para proceso de búsqueda propuesto

Los elementos de la partícula determinarán el grado o peso de importancia de indicador *i*, según la función de elección vista anteriormente.

Los valores iniciales de las posiciones de cada partícula serán de forma distribuida uniformemente entre la posición máxima y mínima que serán instanciados al momento inicial de la ejecución del problema.

El proceso de entrenamiento del enjambre se realizará mediante la resolución de una pequeña porción del problema. Cuando el proceso de entrenamiento de la partícula concluya, se ejecutará el proceso de resolución del problema de forma completa, seleccionando aquella partícula que tenga el mejor fitness.

5.2.1. Función Fitness del Enjambre.

En la sección anterior dedicada al algoritmo de PSO, se denotó que las partículas necesitan una función fitness que evaluará a cada una de las partículas durante el proceso de entrenamiento. Esta función fitness nos determinará los

valores de los parámetros sobre la Función de Elección antes descrita y va a estar determinada por los Indicadores que fueron indicados como evaluadores, será el reconocimiento de que partícula tiene un mejor comportamiento que otra.

La función fitness va a estar dada por la siguiente ecuación:

$$fitness = f(I_i) \tag{1.9}$$

Donde:

- $f(I_i)$: Será la ecuación de cálculo para el indicador I_i, $\forall i$

A simple vista queda de manifiesto que la función fitness estará estrechamente relacionada con las funciones de cálculo de cada indicador.

5.3. Flujo de Implementación.

La siguiente imagen muestra el flujo general de la implementación para la configuración de los parámetros de la función de elección según un enjambre de partículas.

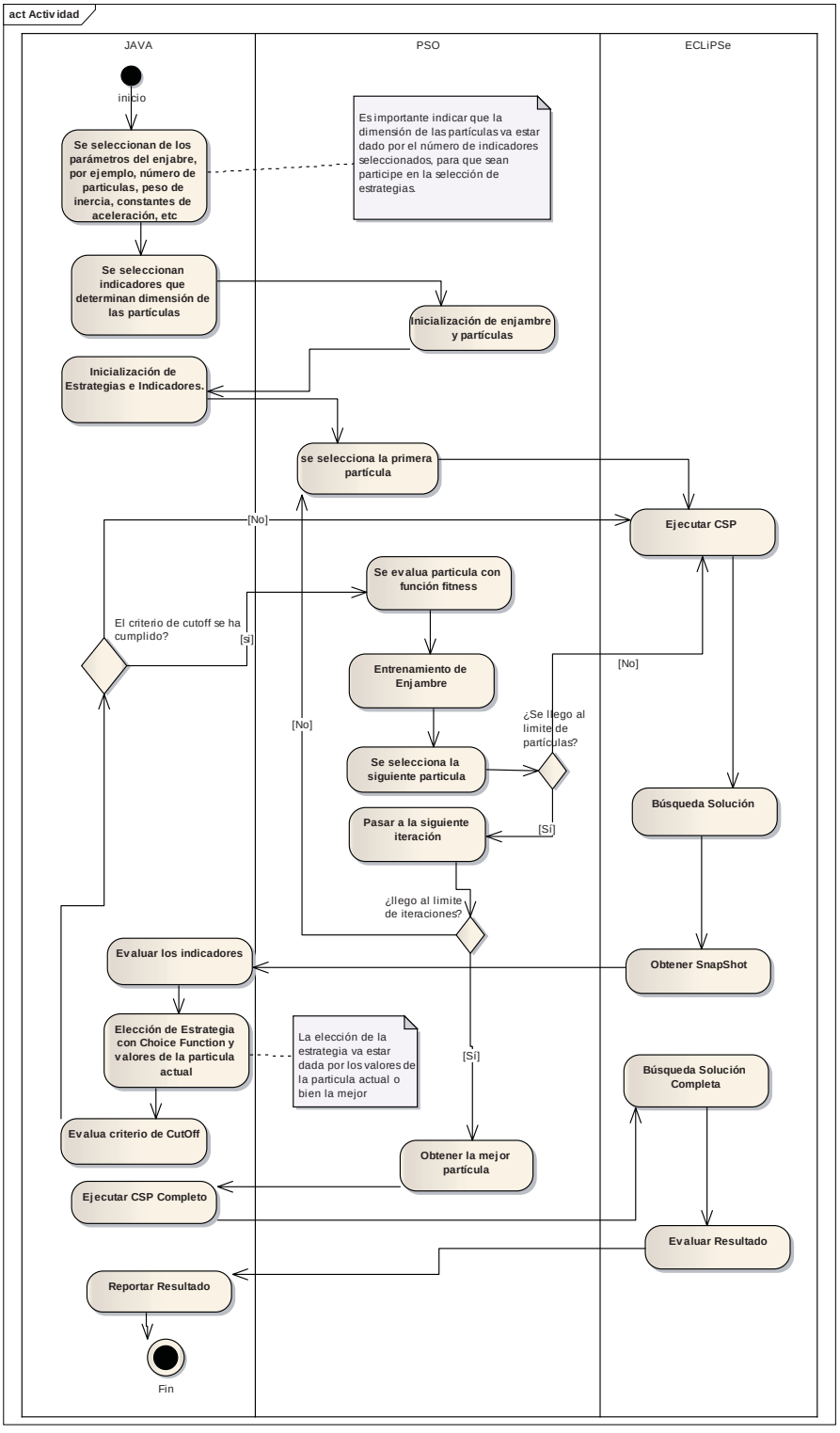


Ilustración 7 – Flujo de implementación para la búsqueda de parámetros a través de PSO

Como se ha mencionado anteriormente, por cada partícula existente en el enjambre se realizará una ejecución del problema seleccionado, de tal manera que durante éste proceso los valores de los Indicadores sean analizados para tener una visualización del estado de la búsqueda. Cada partícula evaluará su mejor posición según la función fitness, permitiendo entregar los parámetros necesarios para el cálculo de la Función de Elección de estrategias. Una vez que se ejecute la Función de Elección de Estrategia se volverá a ejecutar el problema, esta vez con la siguiente partícula del enjambre, esto hasta que la totalidad de las partículas sean evaluadas y entrenadas. Cuando se complete la totalidad de las iteraciones para el entrenamiento del enjambre se elegirá la partícula que tenga la mejor posición global del enjambre, asignando los valores de su posición a los parámetros α_i de cada indicador de la Función de Elección de estrategia. Finalmente se ejecutará el proceso completo de búsqueda para obtener el resultado general del problema.

5.4. Experimentos y Resultados.

Para tener una base comparativa con el resto de los experimentos que se realizarán, es necesario como primer objetivo desarrollar la ejecución de un problema determinado, considerando para la búsqueda de una solución con la ejecución de una estrategia en particular. De esta manera se tendrá una línea de comparación con el resto de los experimentos, luego de esto, es necesario realizar los experimentos que nos indiquen que configuración es la más indicada para la Metaheurística de PSO. Finalizado lo anterior es que podremos realizar la ejecución de los experimentos con función de elección utilizando múltiples estrategias.

5.4.1. Mediciones

De manera de indicar que resultado es mejor que otro, es que se hace necesario elegir una medición que para todos los casos sea la más representativa posible para cada ejecución del problema. Por tal motivo es que se basará en los resultados obtenidos por los indicadores de Bracktrack y Step (Cantidad de Pasos). El motivo de elegir estos indicadores como medición es su simplicidad de cálculo y por sobre todo no tienen factores externos que influyan en su resultado, como por ejemplo tiempo, capacidad de recursos del Computador, etc.

5.4.2. Problemas a Desarrollar.

El problema que se ejecutará será el denominado N-Reinas el cual consiste en, colocar N-Reinas en un tablero NxN, sin que ninguna de ellas se ataque entre sí. Esto quiere decir que al colocar las reinas se debe fijar que se cumpla las restricciones de que ninguna otra reina se encuentre en la misma fila, columna y además en la diagonal. El tamaño

del problema puede tomar valor con un N igual a 1 y también con un N igual o mayor que 4. En la siguiente figura se muestra una solución posible para un N-Reina de dimensión 4:

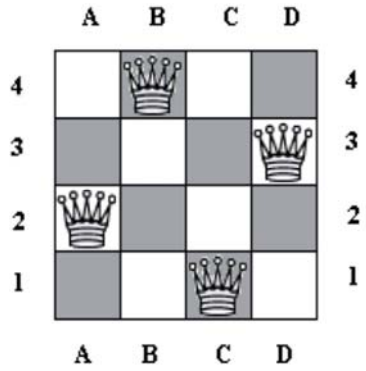


Ilustración 8 – Solución N-Reina Dimensión 4

5.4.3. Resultado Estrategias Unitarias

En la siguiente tabla se muestra los resultados obtenidos de la ejecución del problema de N-Reinas con distintas dimensiones y los resultados obtenidos con la ejecución en particular de cada Estrategia de Enumeración.

	N reina 8		N Reinas 16		N Reinas 20		N Reinas 25		Promedio	
Estrategias	TB	STEP	TB	STEP	TB	STEP	TB	STEP	TB	STEP
Estrategia 1: First + Indomain	10	18	542	1123	10026	22220	2014	4507	3148,00	6967,00
Estrategia 2: AMRV + Indomain	11	17	517	885	2539	4322	-	-	1022,33	1741,33
Estrategia 3: MRV + Indomain	10	18	3	15	11	39	21	67	11,25	34,75
Estrategia 4: Occurrence + Indomain	10	18	542	1123	10026	22220	2014	4507	3148,00	6967,00
Estrategia 5: First + Indomain Max	10	18	542	1123	37320	22220	2014	4507	9971,50	6967,00
Estrategia 6: AMRV + Indomain Max	11	17	517	885	2539	4322		-	1022,33	1741,33
Estrategia 7: MRV + Indomain Max	10	18	3	15	11	39	21	67	11,25	34,75
Estrategia 8: Occurrence + Indomain Max	10	18	542	1123	10026	22220	2014	4507	3148,00	6967,00

Tabla 4 - Resultados Estrategias Unitarias

En la tabla anterior se visualiza que las estrategias que mantienen un mejor comportamiento durante la búsqueda de la solución son las estrategias 3 y 7, y en caso contrario están las estrategias 5, 2 y 6, estas dos últimas estrategias en el problema de N-Reinas con dimensión 25 no fueron capaces de encontrar una solución.

5.4.4. Configuración de PSO

En la siguiente tabla de experimentos se mostrará qué indicador tiene el mejor Fitness para PSO, basándonos en el número de Backtrack y número de pasos que se obtuvieron al resolver el problema. El criterio de cutoff que se utilizará será D (Deep) que es la profundidad del árbol de búsqueda, en sentidos prácticos este indicador nos indicará además de la profundidad en el árbol de búsqueda, cuantas variables han sido instanciadas hasta ese momento, es decir mantiene la equivalencia del indicador VF (Numero de Variables Instanciadas). El valor de criterio de corte tendrá que ser lo suficientemente generoso para obtener la mayor de cantidad posible de información de cómo se está resolviendo el problema, pero a la vez debe ser un criterio que no realice un porcentaje tan alto del problema.

Nuestro problema a resolver será el N-Reinas de 15. Si analizamos la cantidad de variables que existirán en este problemas es fácil determinar que serán 15 (15 reinas que serán posicionadas en una posición del tablero), por lo tanto y según dicho anteriormente el criterio de corte (CutOff) será alrededor del 65%, es decir, una profundidad de 9 o 10 niveles alcanzados.

Fitness (Indicador)	Maximizar/ Minimizar	BT	STEP
IN2 - Relación de Profundidad actual – profundidad previa	Max	186,6	342,2
IN3 - Reducción del espacio de búsqueda	Max	203,8	384,9
IN1 - Variación de la profundidad previa y actual	Max	204,0	386,5
TSB - Número total de Shallow Backtrack	Min	210,6	395,2
VFPS - Numero de variables instanciadas por propagación en cada paso	Max	224,6	525,3
STEP - Paso	Min	231,6	434,1
TAVFES - Número total acumulado de variables instanciada por enumeración	Max	241,6	459,2
PVFPS - Porcentajes de variables instanciadas por propagación en cada paso	Max	249,7	462,2
PTAVFES - Porcentaje del total acumulado de variables instanciadas por enumeración en cada paso sobre el total de variables.	Max	250,9	458,9
DMAX - Profundidad máxima en el árbol de búsqueda	Max	251,4	467,3
TAVFPS - Número total acumulado de variables instanciada por propagación en cada paso	Max	255,0	485,2

DB - Depth Back	Min	257,9	473,5
VFES - Número de variables instanciadas por enumeración en cada paso.	MAx	259,6	474,1
D - Profundidad actual del árbol de búsqueda.	Max	263,3	484,2
TB - Número total de backtrack	Min	276,8	510,8
PVFES - Porcentaje de Variables instanciadas por enumeración en cada paso	Max	278,1	510,5
SS - Espacio de búsqueda	Min	289,1	532,6
NODE - Nodos	Min	295,9	538,7
VF - Número de variables instanciadas	Max	312,7	420,8
PTAVFPS - Porcentaje del total acumulado de variables instanciadas por propagación en cada paso sobre el total de variables	Max	312,9	576,8
VU - Número de variables no instanciadas	Min	332,5	604,2
SB - Shallow Backtrack	Min	356,6	651,3
B - Backtracks	Min	382,0	703,3

Tabla 5 - Resultados búsqueda de Indicador Fitness (30 ejecuciones por caso)

Los datos de la tabla anterior corresponden a 30 ejecuciones por cada indicador utilizado como fitness en el entrenamiento de PSO. Se puede observar que existe una clara diferencia entre los indicadores que aportan de mejor manera a la solución final al ser utilizado como fitness de PSO, es decir, que tienen un mayor grado de implicancia en el entrenamiento de las partículas. Cabe señalar que a pesar que todos los indicadores en más de una ocasión alcanzo a desarrollar el problema de manera no siempre lo hacían con la misma frecuencia que aquellos que obtuvieron un mejor resultado.

Teniendo un conjunto de indicadores que nos ayuden de mejor manera a que el enjambre sea entrenado, es necesario entonces encontrar la mejor configuración para PSO. Los siguientes parámetros de PSO se mantendrán fijos ya que según referencias estos mantienen los mejores parámetros observados.

- Vmax y Vmin = 1,5.
- Constante de aceleración Local: 0,9 [20].
- Constante de aceleración Global: 1,5 [20].

En tanto a que Fitness se utilizara será el indicador IN3 (Reducción del espacio de búsqueda) a pesar de que no es el primero de la lista de los mejores fitness, es el que mantiene un comportamiento más regular.

Número Iteraciones	Número Partículas	Peso Inercia	BT	STEP
100	30	0,9	219,9	408,8
75	30	0,95	220,5	409,3
100	35	0,9	225,2	418,4
150	35	0,9	226,8	427,9
75	25	0,8	232,9	434,0
100	25	0,95	242,6	452,8
150	35	0,95	250,2	464,0
100	30	0,95	250,4	465,1
75	25	0,9	260,9	486,6
75	25	0,95	266,8	577,9
150	25	0,95	268,1	498,3
75	35	0,9	269,8	491,0
75	30	0,8	275,8	506,7
75	35	0,8	281,2	518,1
100	30	0,8	283,9	518,7
100	25	0,9	284,8	518,5
150	35	0,8	290,6	536,7
150	25	0,8	293,7	546,4
75	35	0,95	301,8	542,7
150	30	0,9	303,1	557,3
100	25	0,8	310,0	575,1
75	30	0,9	310,1	570,9
100	35	0,8	314,1	577,4
150	25	0,9	314,3	575,5
150	30	0,8	316,8	583,7
150	30	0,95	324,5	604,7
100	35	0,95	327,6	597,7

Tabla 6 - Resultados configuración PSO - (30 ejecuciones por caso)

De la tabla anterior se desprende que la mejor configuración es con 100 iteraciones 30 partículas y un peso de inercia de 0.9.

5.4.5. Experimentos Función de Elección.

Es de interés también conocer la dependencia que podrían tener algunos de los indicadores entre sí, esto con la finalidad de poder omitir ciertos indicadores al momento de elegir entre estos cuales serán parte de la Función de

elección. Como se mencionó desde un principio esta es una investigación colaborativa, en donde se determinó según un estudio existe una alta relación entre algunos indicadores. En estos estudios se determinó que si se selecciona uno de los indicadores B (Backtracks), NODE (Nodos), TAVFES (Número total acumulado de variables instanciada por enumeración), TSB (Número total de Shallow Backtrack), PTAVFES (Porcentaje del total acumulado de variables instanciadas por enumeración en cada paso sobre el total de variables) o DB (Depth Back), no es necesario utilizar el resto en la función de elección, lo mismo sucede con los indicadores VF (Numero de variables instanciadas) y D (Profundidad actual del árbol de búsqueda), VFPS (Numero de variables instanciadas por propagación en cada paso) y PVFPS (Porcentajes de variables instanciadas por propagación en cada paso), D_PR (Profundidad previa en el árbol de búsqueda) y THRASH (Thrashing). Esta correlación tiene sus fundamentos en que los cálculos de estos tienen una misma base, en otras palabras, para calcular un indicador es necesaria la utilización de otro indicador que en definitiva mantendrá su correlación.

A continuación se presentan los experimentos que determinarán que conjunto de indicadores obtienen un mejor resultado. La base de estos experimentos es la misma que los anteriores, resolviendo el problema de N-Reinas y con la mejor configuración de PSO encontrada.

	N - Reinas 8		N - Reinas 16		N - Reinas 20		Promedio	
Indicadores	**BT**	**STEP**	**BT**	**STEP**	**BT**	**STEP**	**BT**	**STEP**
TODOS	8,97	16,37	132,07	241,73	671,23	1243,53	270,76	500,54
IN1 - IN2 - THRASH - DB	8,70	16,13	147,17	267,40	559,57	1028,03	238,48	437,19
VU - D - IN3	8,97	16,87	143,57	257,30	851,17	1414,97	334,57	563,04
B - STEP - PTAVFES	9,10	17,10	143,67	267,00	479,77	892,73	210,84	392,28
SB - IN1 - IN2	8,57	16,07	118,40	218,57	379,17	711,33	168,71	315,32
IN1 - IN2 - B - STEP	8,63	16,10	155,03	279,90	494,80	925,00	219,49	407,00
VU - B - IN3	8,90	16,87	145,10	264,33	760,50	1401,73	304,83	560,98
VF - DMAX - DB	8,73	16,40	115,57	210,07	371,03	702,33	165,11	309,60

Tabla 7 - Resultados de Indicadores en Función de Elección - (30 ejecuciones por caso)

De la tabla anterior podemos determinar que los mejores conjuntos de indicadores para que serán utilizados en la función de elección son: SB - IN1 - IN2, VF - DMAX – DB , B - STEP – PTAVFES y IN1 - IN2 - B – STEP.

Hasta el momento sólo se ha utilizado la función de elección 1 de la ecuación (1.7). En la siguiente tabla se comparará los resultados obtenidas por estas dos funciones junto con cada conjunto de indicadores antes seleccionados como mejores.

	N - Reina 16				N - Reina 20			
	Función de Elección 1		Función de Elección 2		Función de Elección 1		Función de Elección 2	
Indicadores para Función de Elección	TB	STEP	TB	STEP	TB	STEP	TB	STEP
SB - IN1 - IN2	118,40	218,57	88,87	162,73	379,17	711,33	430,23	817,40
VF - DMAX – DB	115,57	210,07	102,67	186,83	371,03	702,33	337,93	635,67
B - STEP – PTAVFES	143,67	267,00	136,63	254,40	479,77	892,73	402,00	751,00
IN1 - IN2 - B – STEP	147,17	267,40	126,27	234,03	559,57	1028,03	573,23	1048,53

Tabla 8 - Resultados con PSO a problema N Reina - (30 ejecuciones por caso)

En la tabla anterior se tienen los resultados de ejecuciones realizadas con PSO utilizando las 2 funciones de elecciones. Se demuestra que la función de elección 2 (ecuación 1.8) tiene un mejor comportamiento. Es necesario indicar que lo parámetros betas (β_i), tienen un valor de 1 para todos los indicadores que participan de la función de elección.

De la misma manera que se hizo para PSO, la siguiente tabla muestra los resultados de las ejecuciones con QPSO.

	N - Reina 16				N - Reina 20			
	Función de Elección 1		Función de Elección 2		Función de Elección 1		Función de Elección 2	
Indicadores para Función de Elección	TB	STEP	TB	STEP	TB	STEP	TB	STEP
SB - IN1 - IN2	74,13	137,60	78,80	144,70	302,23	584,70	232,30	469,53
VF - DMAX – DB	83,73	153,23	124,77	225,60	306,03	595,30	497,60	964,87
B - STEP – PTAVFES	45,20	90,60	108,50	204,00	349,87	656,07	349,87	656,07
IN1 - IN2 - B – STEP	131,60	237,73	164,83	300,47	507,33	934,00	386,50	730,70

Tabla 9 - Resultados con QPSO a Problema N Reina - (30 ejecuciones por caso)

En los resultados utilizando QPSO, se obtuvieron resultados notoriamente mejores que con PSO en todas las combinaciones de estrategias.

En la siguiente tabla se mostrará el mejor resultado obtenido por cada uno de las funciones de elecciones combinadas con las dos variantes de enjambre de partículas (PSO y QPSO), junto con los resultados que se obtiene al seleccionar una única estrategia. Para el caso de PSO se ocuparán el conjunto de indicadores que obtuvieron un mejor resultados,

al ser incluidos en la función de elección, según tabla 8, los cuales serán VF - DMAX – DB. En tanto para QPSO se seleccionaran según la tabla 9 los indicadores SB - IN1 - IN2.

	N Reinas 20		Magic square 4		Latinsquare 6	
Estrategias	**TB**	**STEP**	**TB**	**STEP**	**x**	**STEP**
Estrategia 1: First + Indomain	10026	22220	12	23	4	19
Estrategia 2: AMRV + Indomain	2539	4322	1191	1816	163	287
Estrategia 3: MRV + Indomain	11	39	3	19	0	18
Estrategia 4: Occurrence + Indomain	10026	22220	10	21	0	19
Estrategia 5: First + Indomain Max	37320	22220	51	84	0	19
Estrategia 6: AMRV + Indomain Max	2539	4322	42	58	163	287
Estrategia 7: MRV + Indomain Max	11	39	97	155	0	18
Estrategia 8: Occurrence + Indomain Max	10026	22220	29	44	0	19
PSO - Función Elección 1	11	39	89	109	0	18
PSO - Función Elección 2	11	39	29	43	0	18
QPSO - Función Elección 1	11	39	23	40	0	18
QPSO - Función Elección 2	15	45	31	50	0	18

Tabla 10 - Mejores resultados de PSO y QPSO

En los resultados de la tabla anterior se muestra que para estos distintos problemas, la configuración de parámetros para la función de elección de estrategia, es útil ya que mantiene un óptimo igual o cercano a la mejor estrategia que se conoce. Aunque se debe indicar que existe una gran fluctuación de resultados entre el mejor y el resto, por ejemplo, si el óptimo para N Reinas 20 es de 11 BackTrack puede existir un salto considerable al segundo mejor óptimo que puede llegar fácilmente a los 200 BackTrack, sin embargo, la cantidad de resultados óptimos prevalece sobre aquellos que no lo son.

CAPÍTULO 6

6. Conclusiones y Comentarios Finales

Durante el proceso de investigación, se dió a conocer las distintas áreas de la computación que podríamos resumir en un mismo objetivo, que es de resolver problemas complejos de forma sencilla, a través de abstracciones de la vida real. Estas abstracciones hacen que el área investigativa se enfrente con nuevos y más complejos problemas. A pesar de lo anterior queda demostrado que existen técnicas que cada día van evolucionando para poder adaptarse a estos cambios.

Por el lado de CSP quedó de manifiesto que las técnicas de resolución no sirven para una gran diversidad de problemas, siendo necesario un proceso superior que realice cambios en estos procesos de búsqueda de la solución. Es por eso que se planteó la utilización de un algoritmo evolutivo el cual tiene la bondad de evolucionar a medida que se busca una solución a los problemas. En este punto es cuando nos introdujimos en el mundo de Swarm Intelligence, en donde el algoritmo de optimización por enjambre de partículas se encargará de encontrar los mejores parámetros que determinará cuál será la mejor estrategia a utilizar para dar solución a un problema modelado de CSP.

Sobre los experimentos se destacó que entre un enjambre PSO sin modificación y el enjambre quántico QPSO existe una mejora sustancial de los resultados, esto puede deberse a que las velocidades utilizadas en PSO se escapen demasiado pronto, encontrándose en los bordes impuestos por Vmax, esto causando que se explore o se explote recursivamente sin poder realizar un modo de indagación en el espacio de búsqueda más depurado.

La calidad del resultado utilizando la Metaheurística poblacional, específicamente de enjambre, tiene una fuerte relación con la función fitness que se le esté asociando. Es decir, una buena elección de qué indicador será seleccionado como fitness del enjambre, será crucial para obtener buenos o malos resultados, esto se demostró al momento de comprar con una serie de indicadores, los cuáles de estos tienen un mejor comportamiento y rendimiento en el entrenamiento de las partículas. Sobre este mismo punto al seleccionar los indicadores adecuados para que sean parte de la función de elección es que se tienen mejores o peores resultados, esto basándonos en que hay ciertos indicadores que están estrechamente relacionados y que al mantenerlos juntos en la función de elección, producirán una redundancia que al fin y al cabo se entorpecerán entre si.

Es importante destacar que los valores de los parámetros α_i para los indicadores que participan en la función de elección, pueden no ser siempre los mismos, pero aun así se pueden obtener los mismos resultados, esto lleva a

pensar que estos valores pueden compensarse unos con otros, de manera tal que el factor de importancia de un indicador sea considerado de distinta manera a otro indicador.

Tareas pendientes que al momento de término de este documento que no se pudieron completar, es la interfaz gráfica en donde es posible realizar la configuración de la ejecución del problema de manera intuitiva, sin embargo la implementación de ésta, no demandará un trabajo mayor, ya que existe una estructura de diseño, la cual fue utilizada para la ejecución de los experimentos, que permitirá el fácil acoplamiento de la interfaz a lo ya desarrollado.

Los posibles trabajos futuros sobre esta investigación, serán encontrar los mejores parámetros betas para las funciones de elección que de alguna manera influirán en los resultados obtenidos durante esta investigación.

7. Referencias

[1] F. Barber and M. Salido, "Introducción a la Programación de Restricciones," *Inteligencia artificial: Revista Iberoamericana de Inteligencia Artificial, ISSN*, vol. 7, no. 20, pp. 13-30, 2003.

[2] E. Tsang, *Fundations of Constraint Satisfaction*. London, UK: Academic Press Limited, 1996.

[3] Guide to Constraint Programming. [Online]. http://kti.mff.cuni.cz/~bartak/constraints/backtrack.html

[4] Z. Liu, *Algorithms for Constraint Satisfaction Problems (CSPs)*. Waterloo_ Ontario, Canada: University of Waterloo, 1998.

[5] C. Bessière, B. Zanuttini, and C. Fernández, "Measuring Search Trees," *Proceedings ECAI'04 Workshop on Modelling and Solving Problems with Constraints*, pp. 31--40, 2004.

[6] F. Manya and C. Gomez, "Técnicas de Resolución de Problemas de Satisfacción de Restricciones," *revista iberoamericana de inteligencia artificial*, vol. 7, no. 19, 2003.

[7] E. Freuder, "A Sufficient Condition for Backtrack-Free Search," *Journal of the ACM (JACM)*, vol. 19, pp. 24-32, 1982.

[8] H. Guesgen and A. Philpott, *Heuristics for Fuzzy Constraint Satisfaction*. New Zealand, 1995.

[9] S. Epstein, E. Freuder, R. Wallace, A. Morozov, and B. Samuels, "The Adaptive Constraint Engine," *Lecture Notes In Computer Science*, vol. 2470, p. 525–542, 2002.

[10] Y. Hamadi, E. Monfroy, and F. Saubion, *What is Autonomous Search?*. 2008.

[11] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Comput. Oper. Res*, vol. 13, pp. 533-549, May 1986.

[12] E.-G. Talbi, *Metaheuristics: From Design to Implementation*. New Jersey, Hoboken: John Wiley & Sons, Inc, 2009.

[13] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv*, vol. 35, pp. 268--308, Sep. 2003.

[14] E. Burke, et al., "Sonia ," *Handbook of Metaheuristics (F. Glover and G. Kochenberger, eds.)*, p. 457–474, 2003.

[15] T. Bäck, *Evolutionary Algorithms and Their Standard Instances*. Publishing Ltd and Oxford University Press, Handbook of Evolutionary Computation, 1997.

[16] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization," *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942-1948, Dec. 1995.

[17] P. S. Shelokar and P. Siarry, *Particle swarm and ant colony algorithms hybridized for improved continuous*

optimization. Universidad de Paris, 2006.

[18] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.

[19] A. Abraham, H. Guo, and H. Liu, "Swarm Intelligence: Foundations, Perspectives and Applications," *Studies in Computational Intelligence (SCI)*, vol. 26, pp. 3-25, 2006.

[20] R. C. Eberhart and Y. Shi, "Particle Swarm Optimization: Develoments, Applications and Resources," *Evolutionary Computation*, vol. 1, pp. 81-86, 2001.

[21] Y. Shi and R. Eberhart, "Parameter Selection in Particle Swarm Optimization," *Lecture Notes In Computer Science*, vol. 1447, 1998.

[22] R. Firsandaya Malik, T. Abdul Rahman, S. Zaiton Mohd. Hashim, and R. Ngah, "New Particle Swarm Optimizer with Sigmoid Increasing Inertia Weight," *International Journal of Computer Science and Security (IJCSS)*, vol. 1, no. 2, pp. 1-65, 2007 .

[23] R. Eberhart and J. Kennedy, *A new optimizer using particle swarm theory*, Proceedings of the sixth international symposium on micro machine and human science ed. Nagoya, Japan, 1995.

[24] J. Liu, J. Sun, and W. Xu, "Quantum-Behaved Particle Swarm Optimization with Adaptive Mutation Operator," *Lecture Notes in Computer Science*, vol. 4221, pp. 959-967, Sep. 2006.

[25] M. Pant, R. Thangaraj, and A. Abraham, "A New Quantum Behaved Particle Swarm Optimization," *Genetic And Evolutionary Computation Conference, Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pp. 87-94, Jul. 2008.

[26] P. Cowling, G. Kendall, and E. Soubeiga, "A Hyperheuristic Approach to Scheduling a Sales Summit," *Selected papers from the Third International Conference on Practice and Theory of Automated Timetabling III*, vol. 2079, pp. 176-190, 2001.

[27] B. C. Labrín, "Dynamic Selection of Enumeration Strategies for Solving Constraint Satisfaction Problems," Universidad Federico Santa María Tesis doctoral, 2011.

[28] The ECLiPSe Constraint Programming System. [Online]. http://www.eclipse-clp.org/

[29] W. Souffriau, P. Vansteenwegen, G. Vanden Berghe, and D. an Oudheusden, *Automated Parameterisation of a Metaheuristic for the Orienteering Problem*, Studies in Computational Intelligence ed. Berlin/Heidelberg, 2008.

[30] B. Crawford, et al., "Adaptive Enumeration Strategies for Constraint Solving," Mar. 2012.

[31] U. Montanari, "Networks of Constraints: Fundamental Properties and Applications to Picture Processing," *Information Sciences*, vol. 7, pp. 95-132, Jan. 1974.

[32] E. Monfroy, C. Castro, and B. Crawford, "Adaptive Enumeration Strategies and Metabacktracks for Constraint Solving," *Advances in Information Systems, Lecture Notes in Computer Science*, vol. 4243, pp. 354-363, 2006.

[33] R. J. Bayardo and D. P. Miranker, "An optimal backtrack algorithm for tree-structured constraint satisfaction problems," *Artificial Intelligence*, vol. 71, no. 1, pp. 159-181, Nov. 1994.

[34] A. K. Mackworth and E. C. Freuder, "The complexity of some polynomial network consistency algorithms for constraint satisfaction problems," *Artificial Intelligence*, vol. 25, no. 1, pp. 65-74, Jan. 1985.

[35] E. Soubeiga:, "Development and application of hyperheuristics to personnel schudeling. PhD thesis," The University of Nottingham thesis, 2003.

[36] K. Chakhlevitch and P. I. Cowling, "Hyperheuristics: Recent developments," *Adaptive and Multilevel Metaheuristics, Studies in Computational Intelligence*, vol. 136, pp. 3-29.

[37] E. Soubeiga, P. I. Cowling, and G. Kendall, "Hyperheuristic Approach to Scheduling a Sales," *Selected papers from the Third International Conference on Practice and Theory of Automated Timetabling III*, pp. 176--190, 2001.

[38] R. Poli, J. Kennedy, and T. Blackwell, *Particle Swarm Optimization an Overview*. Swarm Intell, 2007.