 

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática en Ingeniería

de Computadores

Trabajo Fin de Grado

Optimización de Requisitos Software mediante un Algoritmo Evolutivo Multiobjetivo basado en Inteligencia de Enjambre

<Nombre y Apellidos del Autor>

<Convocatoria, Año>

 

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática en Ingeniería de Computadores

Trabajo Fin de Grado

Optimización de Requisitos Software mediante un Algoritmo Evolutivo Multiobjetivo basado en Inteligencia de Enjambre

Autor: Jesús Chávez Águedo

Tutor: Miguel Ángel Vega Rodríguez

**ÍNDICE GENERAL DE CONTENIDOS**

**Resumen**

**Introducción**

**Requisitos software(teoría)**

**Técnicas de optimización**

**Optimización multiobjetivo**

**Hardware y Software usados**

**Algoritmo de las ranas**

**Ajustes del algoritmo**

**Resultados obtenidos**

**Comparación de los resultados**

**Conclusión**

**Bibliografía**

**RESUMEN**

Lo que vamos a ver en las siguientes páginas, va a ser un desarrollo sobre cómo se optimizan los recursos softwares mediante intentando cumplir dos objetivos, conseguir la máxima satisfacción con el mínimo esfuerzo posible. Para poder cumplir estos objetivos, utilizaremos algoritmos con inteligencia de enjambre.

Para poder entender todo, debemos explicar cada concepto y cada subapartado detenidamente. Hay muchos conceptos que debemos entender antes de entrar en materia.

Explicaremos términos como la dominancia, sobre cómo un individuo domina a otro y ver qué características han de cumplirse para que ocurra esta dominancia. También veremos la distancia crawding y los diferentes frentes de paretos que existen, esto último se apoya en la dominancia.

Nos hemos apoyado en un software externo que nos ayudaba a calcular el HV (Hipervolumen), es otro término que veremos más adelante.

El software se llama “hyp\_ind.c”, al cual le pasamos uno valores previos para que pueda tener un configuración a la hora de calcularnos el hipervolumen.

El algoritmo evolutivo que usamos es el Algoritmo de las Ranas Saltarinas (Shuffled Frog Leaping Algorithm – SFLA). Es el algoritmo que se propuso desde un principio, además es sencillo de entender y da bastantes buenos resultados en comparación con el resto de algoritmos evolutivos. También decidimos utilizar este algoritmo porque es con el que vamos a hacer la comparación directa.

Los resultado que hemos obtenido son bastante competentes, están muy a la par con los resultados que comparamos, se podrían decir que los resultados son algo mejores que en un principio.

Todo esto lo veremos a lo largo de la documentación de este TFG.

**INTRODUCCIÓN**

Lo que vamos a ver en esta sección va a ser cómo va a estar estructurada nuestra documentación...(extender más tarde)

**Cosas a entender**

**Estructura de la documentación**

**Requisitos Softwares (Teoría)**

Todo lo que vamos a ver en esta sección está orientado a entender todos los conceptos teóricos que vamos a ver sobre los requisitos softwares.

Antes de entrar en en materia y empezar a explicar conceptos, lo que vamos a hacer es dar una breve introducción sobre cómo funcionan los “requisitos softwares”.

Todo proyecto software siempre está basado en plazo de entregas, en saber qué hay que hacer en cada momento y claro, el tiempo del que se dispone es bastante limitado, por consiguiente, debe minimizarse el tiempo empleado en cada tarea y tenerlo todo planificado antes de proceder con el desarrollo de una tarea. También debemos de tener en cuenta de los recursos que disponemos son limitados.

(Imagen sobre la planificación de un proyecto software)

Para que que un proyecto software salga a delante sin problemas la planificación de cada tarea debe estar bien calculada, es decir, que ha de desarrollarse en el momento planificado. Por supuesto y damos a entender que todo no se puede realizar, ya que cada tarea conlleva un tiempo y un esfuerzo realizarse.

Este proyecto se encarga de seleccionar los diferentes requisitos que se van a incluir en la siguiente entrega del proyecto ya como he explicado en el párrafo anterior, es complicado cumplir con los plazos de entrega, de hecho, en diversas ocasiones no se pueden desarrollar todos los requisitos planteados, por eso existe la optimización de requisitos sofwares.

La optimización es esencial en la IS (Ingeniería Software) y sobre todo cuando hablamos de desarrollar y/o planificar un proyecto. Existen diversos tipos de optimización, podremos ver algunas a lo largo de esta documentación. Una técnica que vamos a implementar para desarrollar nuestro producto, está basada en metodologías ágiles, es decir, esta enfocado a incrementar el software en cada entrega. En definitiva, se van generando entregas sobre el proyecto software a corto plazo, por tanto, se disponen de nuevos requisitos en cada entrega.

La ingeniería software si tiene una complicación es que debe seleccionar estos requisitos en cada entrega, pero hay diferentes prioridades para desarrollar dicho requisito, como el tiempo de costo, la influencia que tiene hacia otro requisito o viceversa,… No es fácil hallar una solución, ya que depende de diversos factores.

(incluir referencias a los artículos de Miguel A.)

**NRP (Next Release Problem)**

Como hemos dicho en la introducción de los conceptos teóricos, la optimización de requisitos es una ardua tarea, NRP se encarga de esta tarea. NRP debe seleccionar las características para la siguiente entrega, de modo que debe seleccionar las características con menos costo de desarrollo y que satisfagan en lo máximo posible a los clientes.

Nos percatamos de que el problema no consta en que solo no tenemos una característica a evaluar, si no que disponemos de dos característica y ambos, son igual de importantes a la hora de sopesar la selección de una característica.

(No se si debo poner más teoría sobre los inicios de la búsqueda de requisitos)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Hubo varios métodos de selección de atributos antes del nuestro, por eso vamos a explicarlos brevemente antes de entrar más a fondo en materia.

Uno de los métodos se llama: Proceso Analítico Jerarquico o PAJ, en inglés sería AHP (Analytical Hierarchy Process), donde los requisitos son calificado según un valor-costo. (---->AMPLIAR<----)

Otro de los métodos era el Despliegue de Funciones de Calidad o DFC, que en inglés sería QFD (Quality Function Deployment), en este método se organizan en una escala . (---->AMPLIAR<----)

Pero hay un problema con estos dos métodos y es que ambos no son compatibles con todas las dependencias que existen entre los requisitos. Estos métodos no se percatan de que estos recursos son necesidades reales actualmente y que acarrean muchas comparaciones cuando la escala del proyecto empieza a aumentar.

Cuando se halló la dificultad de la selección de requisitos en la Ingeniería Software, se fundamentó como un problema de mono-objetivo en un campo de Ingeniería de Software Basada en Búsqueda ISBB o SBSE(Serach Based in Software Engenieer).

(---->EXPLICAR CON MIS PALABRAS<----).SBSE es el campo de investigación en el que los algoritmos de optimización basados ​​en búsquedas se proponen para abordar problemas en Ingeniería de Software [6]. El problema original propuesto en [2] se ha resuelto con  
diferentes metaheurísticas en los últimos años. Sin embargo, la mayoría de los enfoques publicados son evolutivos de un solo objetivo algoritmos que combinan los objetivos mediante el uso de una función de agregación [10, 11]. En todos los casos, esos trabajos no consideraron las interacciones producidas entre los requisitos. Además, la formulación de objetivo único tiene el inconveniente de hacer una búsqueda sesgada del espacio de la solución, porque los objetivos tienen que ser artificialmente agregados de alguna manera, por ejemplo, con un suma ponderada de los objetivos.

(Sigue formando parte de los requisitos sofwares)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

dfasdfasdf