Sistema de gestión de flujos de trabajo para la definición visual de aplicaciones basadas en algoritmos evolutivos

Rubén Salado-Cid[1], Gabriel Luque[2] y José Raúl Romero[1]

[1] Dpto. de Informática y Análisis Numérico
Universidad de Córdoba, Campus de Rabanales, 14071 Córdoba
{rsalado,jrromero}@uco.es
[2] Andalucía Tech
Universidad de Málaga, Campus Teatinos, 29071 Málaga
gabriel@lcc.uma.es

Resumen Los algoritmos evolutivos son métodos capaces de encontrar soluciones a problemas de optimización y búsqueda que necesitan una gran potencia computacional. La aplicación de este tipo de algoritmos requiere de amplios conocimientos en ciencias de la computación que limita su utilización en un mayor número de dominios, ya que las personas expertas en esos ámbitos no son expertas en computación. Este trabajo propone integrar las principales características de la programación visual, la reutilización de componentes y la gestión de la ejecución, todas ellas aplicadas al dominio de los algoritmos evolutivos, en un sistema de gestión de flujos de trabajo. Así, cualquier usuario puede modelar visualmente las soluciones a sus problemas sin la necesidad de conocer los detalles de bajo nivel de implementación. Además, se favorece el uso de aspectos más avanzados como la creación de diseños experimentales replicables y la automatización del análisis estadístico.

Keywords: algoritmos evolutivos, sistema de gestión de flujos de trabajo, experimentación

1. Introducción

Los sistemas de gestión de flujos de trabajo [1] (*workflow management systems*, WfMS) son aplicaciones *software* que permiten la definición, creación y gestión de la ejecución de un conjunto de tareas que trabajan coordinadamente para alcanzar un objetivo común (denominado como flujo de trabajo) [1]. Para ello, utilizan uno o más motores de ejecución que permiten gestionar los recursos disponibles e invocar las tareas especificadas. La representación visual es llevada a cabo por una interfaz gráfica a través de diferentes nodos o componentes, que son interconectados para definir el orden de ejecución según la lógica de negocio.

Los WfMS han sido utilizados tradicionalmente para la automatización de los procesos de negocio [14]. Sin embargo, recientemente han surgido como un paradigma para la representación y gestión de tareas complejas de computación

científica [6]. Gracias a estos sistemas, es posible integrar distintos tipos de recursos disponibles (bases de datos, servidores o servicios) para facilitar el intercambio de conocimiento entre áreas tan variadas como la biología molecular, investigación clínica, ciencia computacional, física, química o estadística. Los investigadores pueden acceder a diversas herramientas e incorporar todo tipo de datos que se encuentran distribuidos en distintos entornos, para implementar sus propios procedimientos para el análisis científico. También existen otras herramientas independientes del dominio [10], que permiten definir y configurar los elementos funcionales propios de cada área, independientemente del WfMS donde se van a utilizar, para así obtener herramientas ya adaptadas a uno o varios dominios. De esta manera, el usuario final ahorra tiempo y esfuerzo al disponer de los elementos ya configurados y listos para su reutilización, aunque inicialmente fueran preconcebidos para otros ámbitos de trabajo.

Gracias al uso de la programación visual y al nivel de abstracción que ofrecen, estas herramientas son muy útiles a la hora de facilitar el acceso a determinados tipos de tecnologías que poseen una gran curva de aprendizaje. Un ejemplo de ello son los algoritmos evolutivos [15]. La utilización de este tipo de técnicas queda relegada mayoritariamente al ámbito de la informática, debido a la dificultad que supone tanto su implementación como su ejecución sobre paradigmas tradicionales de programación. Por este motivo, no son habitualmente adoptados para la resolución de problemas en un amplio rango de dominios.

Los WfMS favorecen el uso de paradigmas basados en componentes intercambiables, gracias a la reutilización implícita que existe en el diseño y configuración de los flujos de trabajo. Su integración con algoritmos evolutivos (AEs) aportaría un nuevo enfoque, tal y como se indica en [11], al considerar este tipo de técnicas no como una unidad indivisible, sino como un conjunto de componentes que interactúan entre sí. Este nuevo planteamiento se basa en que algunos investigadores están poniendo en duda el rigor científico y la necesidad de las nuevas metáforas empleadas en la investigación de metaheurísticas [11].

En este artículo se presenta la integración de algoritmos evolutivos en un WfMS para facilitar la implementación de soluciones basadas en este tipo de técnicas. Los usuarios de esta herramienta se ven beneficiados al poder obviar detalles de bajo nivel de implementación, por la facilidad de uso que proporciona la programación visual y por el nuevo enfoque basado en componentes intercambiables, que permite analizar la influencia de cada elemento del flujo de trabajo sobre el resultado final.

En el resto del trabajo se presenta el sistema de gestión de flujos de trabajo. En la Sección 2 se muestran algunas herramientas relacionadas y se hace una exposición general de la propuesta, aportando una visión a alto nivel de la arquitectura del sistema. En la Sección 3 se describe su estructura e implementación. En la Sección 4 se muestran tres casos de estudio, mostrando los flujos de trabajo creados para dos problemas específicos y uno para experimentación científica. Finalmente, en la Sección 5 se exponen las conclusiones obtenidas, destacando algunas líneas de trabajo futuro.

2. Visión general de la propuesta

Dentro del ámbito científico, ya se encuentran disponibles algunos WfMS que permiten trabajar sobre dominios específicos, como minería de datos [3,5] o bioinformática [9], gracias a que proporcionan elementos ya adaptados a esos dominios. Debido a que cada vez es más habitual trabajar en ambientes multidisciplinares, han surgido WfMS multidominio, como Taverna [8], Kepler [7] o Triana [4], que permiten al usuario su adaptación para trabajar en un amplio rango de ámbitos. A pesar de las ventajas que presentan estos últimos, suponen un aumento del tiempo de desarrollo y en la complejidad de uso debido a que deben ser adecuados a un dominio específico. También existen otro tipo de sistemas especializados, como HeuristicLab [13], que incluyen una interfaz visual para la creación de algoritmos evolutivos. Sin embargo, estos quedan limitados al uso de sus propios componentes y los recursos computacionales utilizados no son fácilmente escalables. Por tanto, es destacable mencionar que no se ha encontrado ningún WfMS para AEs como el que se presenta en este trabajo.

Para paliar la falta de este tipo de aplicaciones, se propone el diseño e implementación de un WfMS, cuya adaptación al dominio de los AEs se ha llevado a cabo utilizando la herramienta propuesta en [10]. El sistema resultante proporciona al usuario final una serie de elementos funcionales ya configurados y reutilizables sobre distintos problemas, e intercambiables entre sí. Estos elementos, que han sido desarrollados utilizando JCLEC [12], son incluidos dentro de los flujos de trabajo como nodos de ejecución, los cuales se encargan de aspectos específicos del algoritmo evolutivo para el problema a tratar.

2.1. Arquitectura del sistema

El sistema está conformado por la integración de diversos componentes *software* orientados a la composición y gestión de la ejecución de flujos de trabajo. Éstos son representados mediante un grafo dirigido, donde sus vértices representan las tareas a llevar a cabo, y sus aristas indican las dependencias entre cada una de ellas. En la Figura 1 se muestra la arquitectura general del WfMS propuesto, donde la definición de los flujos de trabajo es realizada a través del área de trabajo, y su ejecución es llevada a cabo por el motor de ejecución. Internamente, se utiliza un lenguaje específico de dominio para la especialización de flujos de trabajo, o metalenguaje, para la definición de los recursos que forman parte de la lógica computacional a ejecutar.

El área de trabajo contiene una serie de herramientas, cuyo componente principal es un editor gráfico que permite la composición visual a través de diversos elementos gráficos, junto con una herramienta de validación que verifica que la configuración de cada elemento es correcta, dando soporte al usuario durante la fase de diseño. Para almacenar, intercambiar y recuperar los flujos de trabajo diseñados a través del editor, el área de trabajo proporciona un repositorio que contiene todos los ya creados por el usuario.

El motor de ejecución es el encargado de gestionar la ejecución de los flujos de trabajo definidos en el área de trabajo, los cuales contienen una definición

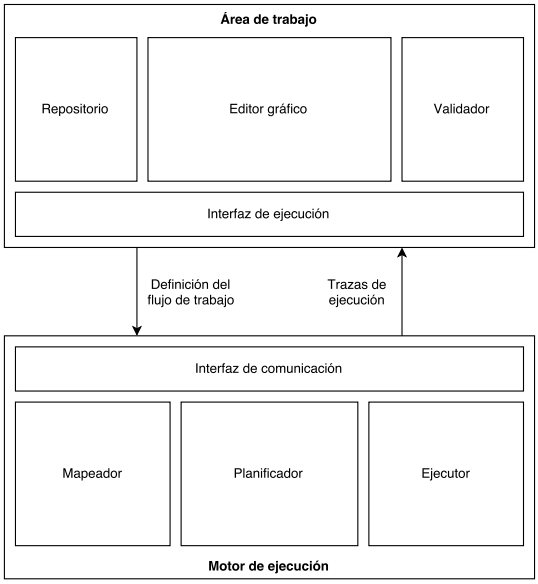

Figura 1: Arquitectura general del sistema de gestión de flujos de trabajo

a alto nivel de los recursos y tareas que son necesarias para su ejecución. Por tanto, se realiza una transformación para obtener una versión ejecutable del mismo, llevada a cabo por el componente "mapeador". A partir de esta versión ejecutable, cada elemento es planificado para su ejecución según los recursos computacionales disponibles, aislando al usuario de los detalles de bajo nivel.

2.2. Flujo de trabajo específico del dominio

Para tratar con problemas resolubles usando algoritmos evolutivos, se han establecido tres capas que proporcionan los distintos elementos necesarios para el uso de AEs mediante la definición de flujos de trabajo (Figura 2). Cada capa aporta un conjunto de elementos funcionales a la capa superior.

Capa de evolutivos. Esta capa proporciona los elementos funcionales necesarios para la definición visual de flujos de trabajo que permiten al usuario resolver los problemas de su dominio utilizando algoritmos evolutivos. Estos elementos son configurables y están agrupados en las siguientes categorías:

- Operadores de mutación: *Modal continuous*, *Modal discrete*, *Mühlenbein*, *Non uniform*, *One locus*, *Random range*, *Several loci* y *Uniform*.
- Operadores de cruce: *Arithmetic*, *Discrete*, *Flat*, *HUX*, *Linear*, *One point*, *Two points*, *Uniform* y *Wright*.
- Operadores de selección: *Betters*, *Boltzmann*, *Disruptive*, *Random*, *Tournament* y *Worses*.
- Algoritmos clásicos: *SG* y *SGE*.
- Codificación de individuos: *Real*, *Int* y *Binary*.

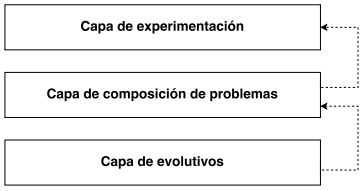

Figura 2: Estructura de capas que proporciona los distintos elementos funcionales

Capa de composición de problemas. Esta capa utiliza los elementos proporcionados por la capa de evolutivos en la composición de los flujos de trabajo. Un usuario sin conocimientos en informática puede hacer un uso básico del sistema combinando los operadores, algoritmos evolutivos y elementos disponibles. Además, los usuarios más avanzados pueden desarrollar sus propios algoritmos e incorporarlos a través de un elemento genérico que permite extender la funcionalidad de la herramienta para adaptarla a sus problemas específicos.

Capa de experimentación. Establece un marco de experimentación para validar los resultados obtenidos por los flujos de trabajo compuestos en la capa anterior. Inicialmente, se proporcionan los siguientes tests estadísticos: test t de Student, test ANOVA y prueba de Kolmogorov-Smirnov. Sin embargo, el conjunto de tests de validación proporcionados es fácilmente escalable.

3. Descripción del sistema resultante

El sistema de gestión de flujos de trabajo ha sido desarrollado como una aplicación de escritorio compatible con múltiples plataformas gracias a la utilización del lenguaje de programación Java. En la Figura 3 se muestra la interfaz gráfica que presenta el área de trabajo de la herramienta.

El área de trabajo provee las funciones para interactuar con el motor de ejecución que permiten al usuario ejecutar todos los elementos del flujo de trabajo de una vez o de uno en uno, pausar la ejecución o detenerla por completo, así como reiniciar el estado de cada elemento. El motor de ejecución permite tanto la ejecución de procesos locales, como puede ser un programa implementado en Java o un flujo de trabajo previamente diseñado, como la ejecución de procesos remotos, invocando servicios SOAP y servicios REST.

El editor gráfico cuenta con una paleta de herramientas, que contiene todos los elementos que pueden formar parte de un flujo de trabajo, un área de diseño para la composición visual de los flujos de trabajo, una hoja de propiedades para la visualización rápida de la configuración de los elementos y una vista en miniatura que proporciona una visión general del flujo de trabajo.

La paleta de herramientas está organizada en base a una serie de categorías que agrupan los elementos que tienen una funcionalidad similar:

- *Connections*: Permite seleccionar el tipo de conexión para interconectar dos elementos del flujo de trabajo. Cada tipo permite: indicar el orden de ejecución, definir el intercambio de información o asociar información textual.

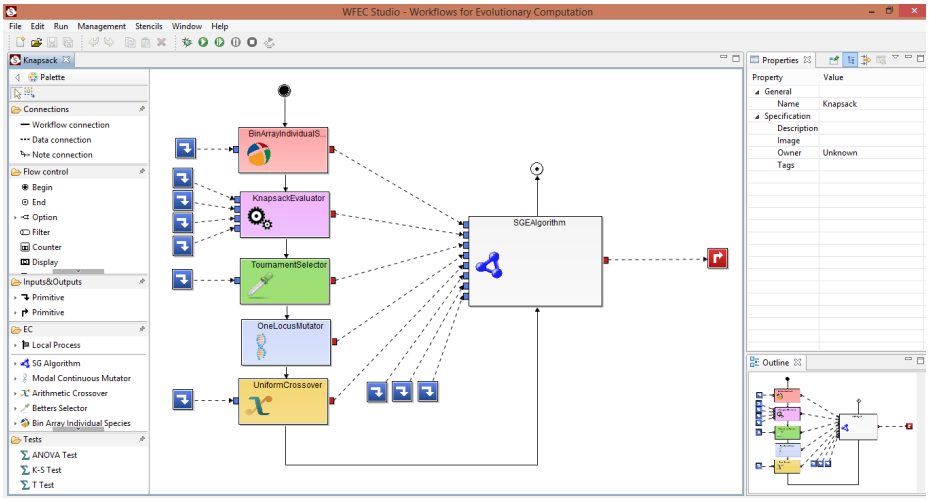

Figura 3: Interfaz gráfica del área de trabajo de la herramienta

- *Flow control*: Proporciona los constructores básicos del lenguaje de programación visual para controlar el flujo de ejecución. Estos constructores son: inicio y fin del flujo de trabajo, selección condicional de caminos de ejecución alternativos, unificador de múltiples caminos de ejecución, contador para la realización explícita de bucles, visualizador de datos, notas textuales y filtros para la adecuación de los datos que son intercambiados entre los procesos.
- *Inputs&Outputs*: Contiene los distintos tipos de entradas y salidas de información que serán utilizados por los procesos del flujo de trabajo. La herramienta es flexible respecto a los tipos de datos soportados, permitiendo que el usuario pueda añadir nuevos tipos que sean propios de su dominio.
- *EC*: Proporciona los elementos de la capa de evolutivos indicados en la Sección 2.2. Estos elementos están agrupados según su funcionalidad. Así, los operadores de cruce, de mutación o de selección se encuentran agrupados en sus respectivas categorías. Los procesos de cada categoría son intercambiables entre sí, de modo que se facilita la creación de distintas configuraciones. También se provee un proceso genérico que permite extender la funcionalidad ofrecida por los elementos de la paleta al permitir incorporar cualquier algoritmo, tanto escrito en Java como un flujo de trabajo modelado con el propio sistema, para poder utilizarlo dentro de los flujos de trabajo definidos.
- *Tests*: Agrupa los distintos tipos de tests estadísticos de la capa de experimentación (Sección 2.2) que permiten al usuario verificar los resultados obtenidos tras la ejecución de un flujo de trabajo.

El área de diseño permite utilizar los elementos de la paleta de herramientas para componer y configurar los elementos que forman parte del flujo de trabajo. Para validar que los elementos utilizados son configurados e interconectados adecuadamente, se muestran avisos visuales para informar sobre la existencia de un problema que impedirá su ejecución. Además, se permite la configuración del

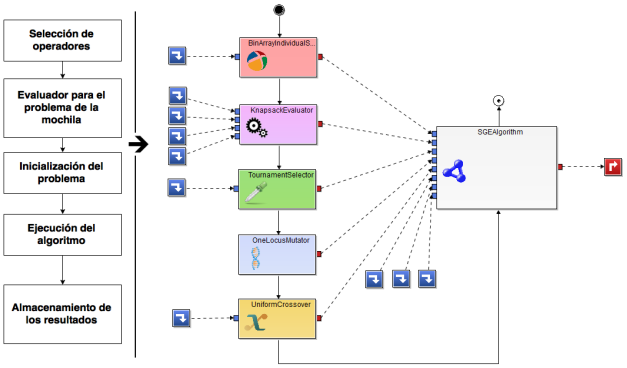

Figura 4: Diseño conceptual y flujo de trabajo para el problema de la mochila

aspecto visual de todos los procesos que forman parte del mismo, permitiendo modificar distintos parámetros, como el color, el tipo línea o el icono.

Durante la fase de ejecución, se dispone de una consola que muestra las trazas generadas por el motor de ejecución en tiempo real. Estas trazas muestran el valor de los resultados que se obtienen tras la ejecución de cada elemento, además de otro tipo de información útil para el usuario, como el tiempo empleado o la configuración de los valores de entrada utilizados.

4. Casos de estudio

Se han realizado una serie de casos de estudio para dos tipos de problemas en el dominio de la optimización continua y combinatoria. En ambos, se ha diseñado y configurado el flujo de trabajo para su ejecución, utilizando los elementos proporcionados por el editor gráfico. También se ha modelado un experimento con dos configuraciones distintas de uno de los problemas previamente diseñados para mostrar el uso de los elementos de la capa de experimentación.

4.1. Problema de la mochila

La Figura 4 muestra el flujo de trabajo diseñado para ejecutar una instancia del problema de la mochila. Cada proceso es una configuración específica del algoritmo evolutivo. El más importante es el tipo de algoritmo a utilizar, cuyas dependencias son los componentes proporcionados por la capa de evolutivos (Sección 2.2). El resto de procesos se corresponden con los elementos específicos que conformarán la solución a este problema (operadores de mutación y cruce, tipo de codificación, número de generaciones, etc). A continuación, a cada proceso se le asigna los valores concretos de entrada que serán empleados para su ejecución, utilizando la conexión de datos. Una vez configurados, la conexión de control especifica el orden de ejecución. Finalmente, se incluyen los elementos de inicio y fin de la ejecución, y se incluye el elemento para mostrar los resultados.

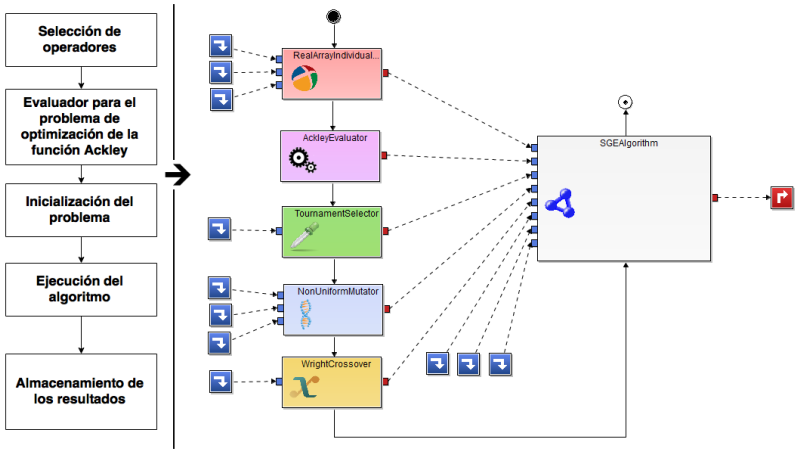

Figura 5: Diseño conceptual y flujo de trabajo para la función Ackley

4.2. Problema de optimización de la función Ackley

En la Figura 5 se muestra el flujo de trabajo modelado para la ejecución de una instancia del problema de optimización de la función Ackley [2]. Debido a que los algoritmos evolutivos comparten una estructura común, los flujos de trabajo son muy similares a pesar de resolver problemas muy diferentes. Esto facilita la labor del usuario ya que simplemente necesita intercambiar los elementos de una misma categoría. En este caso, se han escogido los componentes correspondientes de la capa de evolutivos (Sección 2.2) para adaptar la solución diseñada a las necesidades del problema actual.

4.3. Modelado de la experimentación

Para extraer conclusiones acerca de la ejecución de distintas instancias de un mismo problema, se realiza la verificación de los resultados obtenidos para comprobar qué configuración es más adecuada en cada caso. En la Figura 6 se muestra el flujo de trabajo definido para la experimentación con dos instancias del problema de la mochila descrito en la Sección 4.1.

En primer lugar, se realizan una serie de ejecuciones de los algoritmos modelados en la capa de composición de problemas (denominados “Knapsack1” y “Knapsack2”) para obtener un número de resultados que sea suficiente para extraer conclusiones. Para ello, se utilizan los elementos adecuados para realizar iteraciones durante la ejecución del flujo de trabajo. Utilizando el elemento genérico, se incluye un nuevo proceso para almacenar los resultados obtenidos tras cada ejecución de los algoritmos (los procesos “Memento1” y “Memento2”). Una vez realizadas todas las ejecuciones de ambos algoritmos para una misma instancia del problema, se procede a la ejecución de los tests estadísticos sobre la lista de resultados obtenidos tras cada ejecución. Para comprobar si es posible ejecutar un análisis paramétrico sobre los resultados, se verifica la normalidad de

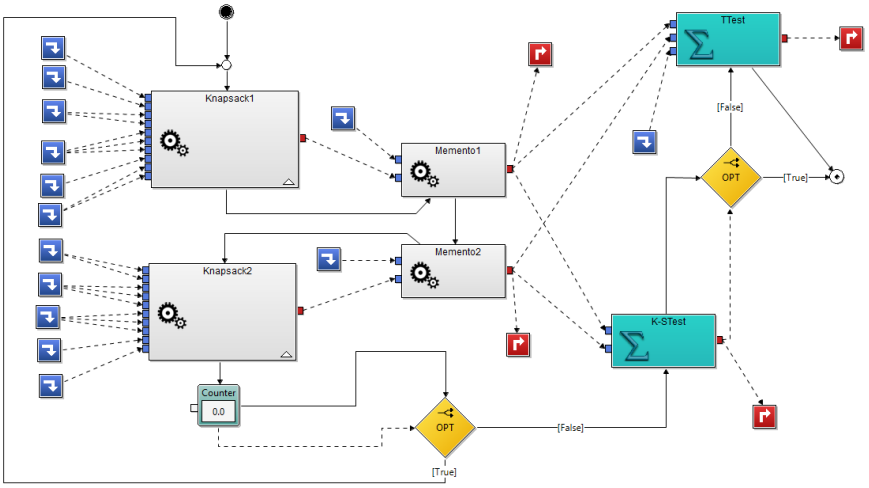

Figura 6: Flujo de trabajo para el modelado de un experimento

los datos ejecutando la prueba de Kolgomorov-Smirnov. Si se obtiene un resultado positivo, se ejecuta el test paramétrico t de Student y se muestra el resultado, si se obtiene un resultado negativo, finaliza la ejecución del flujo de trabajo.

5. Conclusiones y trabajo futuro

En este trabajo, se ha presentado el sistema de gestión de flujos de trabajo para la creación de aplicaciones basadas en algoritmos evolutivos. El sistema cuenta con un editor gráfico para la composición visual de flujos de trabajo. Los elementos proporcionados en la paleta de herramientas son intercambiables entre sí, permitiendo al usuario experimentar con distintas configuraciones y analizar cuál de ellas de adapta mejor a su problema particular. El motor de ejecución hace uso los recursos computaciones disponibles para realizar la ejecución de cada elemento del flujo de trabajo.

Los elementos adaptados al dominio se organizan en capas. La capa de evolutivos proporciona los elementos básicos para la configuración de problemas en la capa de composición de problema. Ésta, a su vez, proporciona los elementos necesarios para que en la capa de experimentación se puedan validar los resultados obtenidos, gracias a la utilización de una serie de tests estadísticos.

En el futuro se aumentará el número de elementos proporcionados por el sistema tanto en la capa de evolutivos como en la capa de experimentación para conformar los flujos de trabajo. De esta manera, se aumenta la variedad de configuraciones y, por tanto, aumenta la variedad de soluciones que se pueden adoptar para cada problema particular. Finalmente, se realizarán pruebas con usuarios reales expertos y no expertos para recoger sus impresiones.

Agradecimientos

El primer y tercer autor agradecen la financiación por parte del proyecto MINECO TIN2014-55252-P. El segundo autor agradece la financiación por parte del proyecto 8.06/5.47.4142 (en cooperación con VSB-Technical University de Ostrava), del programa de fortalecimiento de Universidad de Málaga UMA/FEDER FC14-TIC36 y del proyecto MINECO TIN2014-57341-R (`http://moveon.lcc.uma.es`).

Referencias

1. Terminology & Glossary. Technical Report WFMC-TC-1011, Workflow Management Coalition (1999)
2. Bäck, T.: Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford University Press, USA (1996)
3. Berthold, M.R., Cebron, N., Dill, F., Gabriel, T.R., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., Wiswedel, B.: KNIME: The Konstanz Information Miner. In: Data Analysis, Machine Learning and Applications, pp. 319 – 326. Springer (2008)
4. Churches, D., Gombás, G., Harrison, A., Maassen, J., Robinson, C., Shields, M.S., Taylor, I.J., Wang, I.: Programming scientific and distributed workflow with Triana services. Concurrency and Computation: Practice and Experience 18(10), 1021–1037 (2006)
5. Demšar, J., Zupan, B., Leban, G., Curk, T.: Orange: From experimental machine learning to interactive data mining. In: Knowledge Discovery in Databases: PKDD 2004, Lecture Notes in Computer Science, vol. 3202, pp. 537 – 539. Springer (2004)
6. Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., Myers, J.: Examining the challenges of scientific workflows. Computer 40(12), 24–32 (2007)
7. Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J., Zhao, Y.: Scientific workflow management and the Kepler system. Concurrency and Computation: Practice and Experience 18(10), 1039–1065 (2006)
8. Oinn, T., Greenwood, M., Addis, M., Ferris, J., Glover, K., Goble, C., Hull, D., Marvin, D., Li, P., Lord, P.: Taverna: Lessons in creating a workflow environment for the life sciences. Concurrency and Computation: Practice and Experience 18(10), 1067–1100 (2006)
9. Rampp, M., Soddemann, T., Lederer, H.: The MIGenAS integrated bioinformatics toolkit for web-based sequence analysis. Nucleic Acids Res 34 (2006)
10. Salado-Cid, R., Romero, J.R., Ventura, S.: Metaherramienta para la generación de aplicaciones científicas basadas en workflows. X Jornadas de Ciencia e Ingeniería de Servicios (JCIS) pp. 96–105 (2014)
11. Sörensen, K.: Metaheuristics—the metaphor exposed. International Transactions in Operational Research 22(1), 3–18 (2015)
12. Ventura, S., Romero, C., Zafra, A., Delgado, J., Hervás, C.: JCLEC: a Java framework for evolutionary computation. Soft Computing 12(4), 381–392 (2008)
13. Wagner, S., Affenzeller, M.: Heuristiclab: A generic and extensible optimization environment. Adaptive and Natural Computing Algorithms pp. 538–541 (2005)
14. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer Berlin Heidelberg (2007)
15. Whitley, D.: An overview of evolutionary algorithms: practical issues and common pitfalls. Information and Software Technology 43(14), 817–831 (2001)