De Frameworks a Ecosistemas: Evolución del Software para Optimización Metaheurística

José Antonio Parejo, Pablo Fernandez, y Antonio Ruiz-Cortés

Dept. de Lenguajes y Sistemas Informáticos

Grupo de Investigación en Ingeniería del Software Aplicada

Universidad de Sevilla

japarejo@us.es

Resumen

La naturaleza de las técnicas metaheurísticas las hace propensas a la creación de herramientas software para su aplicación (típicamente frameworks o librerías). Estas herramientas permiten reducir el esfuerzo de implementación y el riesgo de errores. Sin embargo el desarrollo y mantenimiento de dichos frameworks es una tarea costosa. Además, el proceso general de resolución de problemas de optimización y la investigación con metaheurísticas exige el uso de un conjunto más amplio de herramientas. Los ecosistemas software han demostrado ser claves para conseguir un modelo eficaz y sostenible para el desarrollo, evolución y mantenimiento de herramientas por parte de varias organizaciones distintas. En este artículo se propone la creación de un ecosistema de herramientas software para la resolución de problemas de optimización con metaheurísticas que permita integrar distintos frameworks de optimización así como las aportaciones de los usuarios a los mismos y otras herramientas desarrolladas por distintas organizaciones en este contexto, automatizando los flujos de información entre las mismas.

1. Introducción

Las metaheurísticas proporcionan un esquema conceptual para la resolución aproximada de problemas de optimización. Sin embargo, éstas requieren de una implementación específica para transformar los conceptos generales y aplicables para gran cantidad de problemas, en software de resolución efectiva del problema en cuestión. Los frameworks y librerías para optimización con técnicas metaheurísticas (en adelante MOFs, Metaheuristics Optimization Frameworks) permiten aplicar metaheurísticas para resolver problemas de optimización con un menor esfuerzo y riesgo de errores de implementación. Para ello, los MOFs proporcionan versiones pre-implementadas de los algoritmos generales y mecanismos para su adaptación. Esto permite al usuario implementar únicamente los elementos específicos dependientes del problema a resolver y adaptar el funcionamiento del algoritmo general.

Actualmente existe un gran número de frameworks y librerías para optimización con estas técnicas, con distintas características y un soporte dispar para las distintas metaheurísticas desarrolladas por distintos grupos de investigación. En el cuadro 1 se muestran algunos de estos frameworks y caracterítiscas como: técnicas soportadas, lenguaje de implementación y tamaño en clases y paquetes.

El número de metaheurísticas y variantes con posibles puntos de extensión y personalización de su funcionamiento suponen un importante desafío para el diseño e implementación de los MOFs. Una adecuada respuesta a este desafío exige conseguir un balance adecuado entre: un diseño suficientemente flexible y adaptable, un núcleo de funcionalidad básica disponible para el usuario con el mínimo esfuerzo de implementación, y la necesidad de mantener el tamaño y complejidad del frame-

work en cotas manejables. Esto limita las capacidades de los frameworks y las opciones de los usuarios, debido a que están obligados a renunciar a algunas características al usar un framework de optimización en lugar de otro.

En el modelo de desarrollo y reutilización del software promovido por los frameworks los usuarios reutilizan las clases y componentes proporcionados obteniendo un beneficio técnico, mientras que raramente se incorporan al framework los desarrollos y ampliaciones que los usuarios realizan, y en diversas organizaciones se desarrollan frameworks distintos con el mismo objetivo e incapaces de interoperar.

Por otro lado, el conjunto de herramientas usadas en el contexto de la resolución de problemas de optimización con metaheurísticas es más amplio (especialmente en el contexto de la investigación) incluyendo: herramientas para la gestión de la ejecución de los procesos de optimización (tanto en entornos locales como distribuidos y para optimización paralela), herramientas de análisis estadístico de los resultados, herramientas de diseño de experimentos y herramientas para búsqueda hyperheurística de ayuda a la selección de los parámetros y variantes adecuadas para el problema a resolver. Al igual que con los MOFs, existen alternativas para estas herramientas, con la problemática añadida de que la integración y gestión de los flujos de información entre las mismas no suele estar automatizada. Además el uso de varios frameworks para crear aplicaciones que combinen sus características o integren otras herramientas suele ser muy complejo.

Los autores de este artículo han creado y mantienen un framework de optimización metaheurístico [5, 14]. Esta experiencia junto con la reciente evaluación de distintos frameworks de optimización [15] para establecer una comparativa y estado del arte motiva la creación de la presente propuesta: *crear un ecosistema software para el desarrollo de las herramientas de optimización con metaheurísticas, donde las herramientas desarrolladas por distintas organizaciones puedan integrarse*. El objetivo de este cambio de modelo es promover la reutilización del esfuerzo del desarrollo, pruebas y mantenimiento del software para optimización con metaheurísticas, así como la mejora de la interoperabilidad, automatización de los flujos de información, y utilidad de las herramientas proporcionadas a los usuarios por la comunidad de desarrolladores de herramientas para la optimización con metaheurísticas.

El presente artículo se estructura como sigue: en la siguiente sección definimos qué es un ecosistema software e introducimos el interés que puede suponer un ecosistema para la optimización con metaheurísticas. En la sección 3 se describe la arquitectura del ecosistema, sus elementos y los flujos de información que se establecen entre ellos. En la sección 4 se analizan las ventajas, inconvenientes y desafíos que presenta la creación de un ecosistema. Finalmente se estudia el trabajo relacionado (sec. 5) y se elaboran las conclusiones del artículo.

2. Los Ecosistemas Software

De acuerdo con la visión de Bosch [7],un ecosistema software se constituye a partir de una plataforma software, un grupo de desarrolladores pertenecientes al responsable de dicha plataforma, un grupo de desarrolladores externos y una comunidad de expertos del dominio que componen soluciones relevantes para satisfacer las necesidades de una comunidad de usuarios. El tamaño, alcance y naturaleza de los ecosistemas pueden ser muy diferentes según el caso: pueden crearse alrededor de un producto exitoso de una PYME o de código abierto (como eclipse [3]), alrededor del repositorio de aplicaciones de una gran organización pública (e.g. marcos de desarrollo en la Administración Pública [13]), o privada (plataformas de aplicaciones para móviles tales como Apple AppStore o Android Market).

En un ecosistema software se generan productos que usan los servicios de la plataforma y aplicaciones que extienden estos productos con funcionalidad desarrollada por desarrolladores internos o externos (a las organizaciones desarrolladoras de la plataforma y los productos).

La plataforma base del ecosistema puede ser un sistema operativo (AppStore y el Android

Tabla 1: Frameworks de Optimización Metaheurística

Frameworks	ECJ	ParadisEO	Eva2	FOM	JCLEC	OAT	Opt4j	EasyLocal	HeuristicLab
Recocido Simulado		✔	✔	✔			✔	✔	✔
Búsqueda Tabú		✔		✔				✔	✔
GRASP				∼					
VNS		✔		✔					
Algoritmos Evolutivos	✔	✔	✔	∼	✔	∼	∼		✔
PSO	✔	✔	✔				✔		✔
ACO				✔		✔			
Lenguaje de Implementación	Java	C++	Java	Java	Java	Java	Java	C++	C#
Tamaño en paquetes/módulos	28	10	54	215	63	109	35	80	?
Tamaño en clases/ficheros	226	542	594	510	304	373	417	244	?

Market) o una aplicación extensible (eclipse). Una característica fundamental de los ecosistemas es el uso de un modelo de componentes para habilitar la distribución de los desarrollos sobre la plataforma (en el caso de AppStore y el Android Market son los formatos de instalación en el sistema operativo de soporte y en el del entorno de desarrollo eclipse es OSGI). Otro elemento importante es una infraestructura global de soporte a la distribución de componentes, que ejerce de punto central de intercomunicación habilitando además la relación entre los usuarios del ecosistema y los desarrolladores de componentes, pudiendo incluir mecanismos de interacción social, como valoraciones o comentarios por parte de los usuarios. Esta infraestructura contribuye a dinamizar el ecosistema y potenciar las ventajas de su uso tanto para los desarrolladores como para los usuarios. La plataforma suele incluir un componente de acceso a la infraestructura global que ayuda a la gestión e instalación de los componentes (en eclipse esta infraestructura se basa en repositorios de plugins y los mecanismos para descarga y actualización automatizada incluidos en el entorno de desarrollo).

El interés del ecosistema software para el desarrollo del software de optimización con metaheurísticas y sus usuarios radica en que: (i) permite hacer más sostenible y dinámico el desarrollo de los frameworks y librerías de optimización con metaheurísticas, y (ii) establece una arquitectura con un conjunto de herramientas, sus interfaces y flujos de información que permite automatizar las tareas comunes y dar un soporte global al proceso de resolución de problemas de optimización, desde el modelado e implementanción de los componentes dependientes del problema hasta el análisis estadístico de resultados y la generación de gráficos e informes con los resultados. Esto conformaría un *Ecosistema Software de herramientas para la Optimización con Metaheuristicas (MOSE)*.

3. Arquitectura de un MOSE

El ecosistema software para la optimización con metaheurísticas que proponemos (descrito en la figura 1) se compone de un conjunto de elementos básicos, que forman parte de la plataforma mínima de soporte a la resolución de problemas de optimización en el contexto del ecosistema y un conjunto de elementos opcionales, que proporcionan valor añadido.

Entre los elementos básicos encontramos:

- **Infraestructura de soporte centra-**

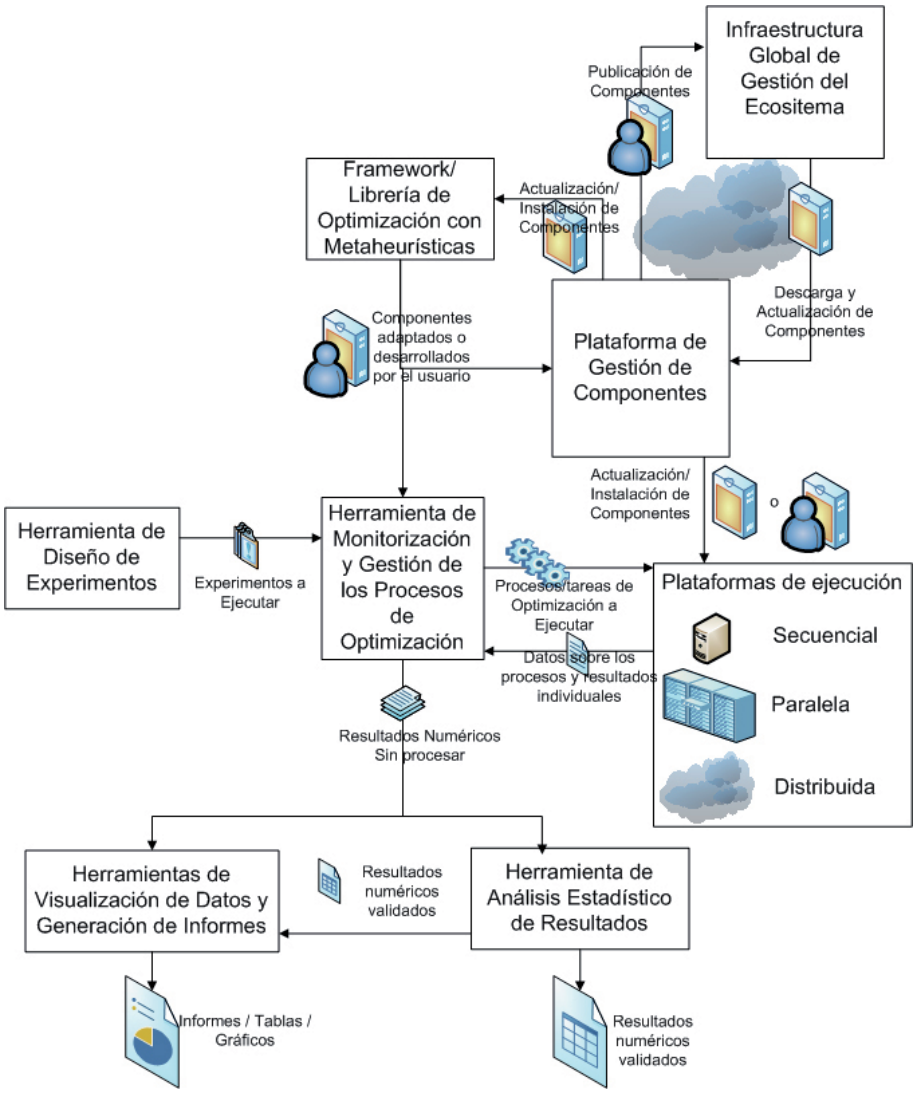

Figura 1: Elementos de un MOSE y flujos de información que se establecen entre los mismos

lizado. Esta infraestructura provee la información de referencia de la plataforma, así como un punto de publicación/sincronización/descarga de los componentes del ecosistema, ejerciendo de nexo de unión entre sus usuarios/desarrolladores. Una característica diferencial de nuestra propuesta es que esta infraestructura no está limitada a un sitio web con la información y versiones de descarga, sino que debe ser compatible con la plataforma del ecosistema y su modelo de componentes, para permitir la actualización y publicación de los elementos del ecosistema de manera automatizada.

- **Plataforma de gestión de componentes/servicios**. Esta plataforma gestiona la instancias del software perteneciente al ecosistema que se usan, y las configura para hacer uso de la infraestructura anterior.

- **Librería/Framework de optimización metaheurística**. Es el elemento clave del ecosistema, su arquitectura hace uso del modelo de componentes adoptado en la plataforma. Permite al usuario especificar el problema a resolver y los algoritmos concretos a usar.

Como elementos opcionales del ecosistema aparecen:

- **Herramienta de diseño de Experimentos y búsqueda hyperheurística**. Un adecuado diseño de experimentos es fundamental para reducir el coste de experimentación y obtener las respuesta adecuadas a las cuestiones y problemas que se desean resolver [16, capítulo 4]. A su vez, la selección de la técnica, sus operadores y valores de parámetros adecuados para la resolución de un problema es una tarea compleja, que puede exigir en algunos casos el uso de estrategias de búsqueda e incluso metaheurísticas [9]. Esta herramienta generará una especificación de los experimentos y procesos de optimización a ejecutar.

- **Herramienta de Monitorización/Gestión de los procesos de optimización**. Esta herramienta se encarga de, a partir de la especificación de experimentos a realizar generada por la herramienta anterior, y en base a los componentes desarrollados por el usuario para especificar el problema y los algoritmos a aplicar, generar un conjunto de tareas concretas de optimización a ejecutar por la plataforma de ejecución, invocar la ejecución de dichas tareas monitorizando su funcionamiento y obtener y agregar los resultados en un conjunto útil para otros elementos del ecosistema.

- **Plataforma de ejecución**. Este componente tiene como misión ejecutar, las tareas gestionadas por la herramienta anterior. La ventaja de incluirlo como un elemento del ecosistema es la posibilidad de exponer con un mismo interfaz al resto del ecosistema plataformas de diversos tipos, como puedan ser plataformas de ejecución clásicas como la máquna virtual de java, de procesamiento paralelo, por ejemplo en cluster o incluso distribuidas, usando por ejemplo Condor. Una particularidad del uso de estos elementos en el ecosistema es que los componentes desarrollados por el usuario para describir el problema o modificar los algoritmos de optimización deben instalarse en estas plataformas para llevar a cabo las tareas de optimización.

- **Herramienta de análisis estadístico de resultados**. El análisis estadístico de los datos es fundamental para la extracción de conclusiones acertadas a partir de los datos de la experimentación. [16, capítulo 3].

- **Herramienta de Visualización de Datos y Generación de Informes**. Esta herramienta permite generar gráficas e informes de manera automática o semiautomatizada, para liberar al usuario y ayudarlo a extraer conclusiones sobre los experimentos. Los datos de entrada a esta herramienta. Ejemplos de estas herramientas pueden ser hojas de cálculo de

paquetes ofimáticos, para las que podrían crearse plugins o scripts que faciliten el acceso a la información y generación de los informes más comunes en el área, o bien módulos ad-hoc integrados en la herramienta de gestión y monitorización de la ejecución.

Los flujos de información entre los elementos del ecosistema y la relación de éstos con el usuario deberían articularse usando interfaces abstractas claramente especificadas y, a ser posible, independientes de la tecnología de implementación, para favorecer las posibilidades de integración de las herramientas como elementos del ecosistema. Por ello proponemos el uso del paradigma orientado a servicios y las tecnologías de servicios web para el establecimiento de dichos interfaces.

Para facilitar la interacción a un nivel de abstracción adecuado y de manera cómoda para el usuario, deben habilitarse lenguajes específicos de dominio (DSLs) para la descripción de los elementos a gestionar por cada herramienta. Esto permite a los usuarios especificar la interacción entre las distintas herramientas y su comportamiento sin necesidad de conocer los detalles específicos de cada elemento del ecosistema y su implementación. Se han realizado algunas propuestas en este sentido para el caso de los frameworks de optimización con algoritmos evolutivos , especificando la sintaxis de un lenguaje [10] y una interfaz de usuario [11] para la descripción de dichos algoritmos. Hasta donde llega nuestro conocimiento, no se han ampliado dichos lenguajes para soportar otro tipo de metaheurísticas ni definido propuestas similares para el resto de elementos del ecosistema propuesto.

4. Ventajas, Inconvenientes y Desafíos para un MOSE

Las consecuencias de esta transformación desde un framework a un ecosistema software para la optimización con metaheurísticas se describen a continuación en términos de ventajas, inconvenientes y desafíos. En cuanto a las ventajas destacamos:

- Incremento de la fidelización de los usuarios, al tener integradas un conjunto de herramientas más completo.
- Aceleración de la innovación través del entorno abierto del ecosistema y la relación entre los usuarios/desarrolladores.
- Colaboración con los socios en el contexto del ecosistema para compartir los costes y riesgos de la innovación y el desarrollo de componentes en el ecosistema.
- Capacidad para incorporar a la plataforma la funcionalidad desarrollada por los usuarios (una vez probada) a través de los distintos componentes.
- Decremento en los costes de mantenimiento y pruebas ya que se hace posible compartirlos entre los socios del ecosistema.
- Compatibilidad con el modelo actual de desarrollo de los frameworks si se estructura adecuadamente la plataforma de soporte al ecosistema. Sería posible mantener dos líneas de desarrollo, para el framework y la plataforma.
- La posibilidad de crear una plataforma de soporte a varios frameworks, creando así un ecosistema más rico. Este escenario tiene la ventaja adicional de que los servicios generales que se creen en el contexto de la plataforma pueden reutilizarse para las distintos herramientas. Un ejemplo de este tipo de servicio serían los servicios de automatización de los flujos de información si se definen adecuadamente las interfaces de los elementos.

Sin embargo el uso de un enfoque basado en ecosistemas también presenta invonvenientes:

- El número y compejidad de las dependencias entre los distintos componentes y sus equipos de desarrollo tiende a hacer que el aseguramiento de la calidad en el contexto de la plataforma requiera un mayor esfuerzo.

- El cambio de la propiedad de los productos que supone el uso y apertura de la plataforma, puede frenar el crecimiento de la comunidad de usuarios del ecosistema, y llevarlo en último término al abandono.
- El uso del ecosistema implica hasta cierto punto una perdida del control sobre el framework y la plataforma, al hacerlos más abiertos a las contribuciones de terceros.

Estos inconvenientes suelen conllevar en el contexto de los ecosistemas software un decremento en la frecuencia de actualización y mejora de la plataforma de soporte y el conjunto de componentes básicos. Además, el contexto concreto de la optimización con metaheurísticas presenta características específicas que suponen desafíos adicionales para la creación de un ecosistema software como el descrito:

- **La enorme variabilidad del dominio de aplicación a abordar**. El número de frameworks, técnicas y variantes a tener en cuenta son muy amplios y complejos.
- **La definición adecuada de las interfaces y flujos de información entre los elementos del ecosistema**, para posibilitar que coexistan alternativas para cada elemento y favorecer la interoperabilidad entre los mismos.

5. Trabajos Relacionados

Existen gran cantidad de frameworks y librerías de optimización de metaheurísticas (en [15] se identifican 33), así como de artículos, e incluso monografías presentando sus características [12]. Sin embargo, solo algunas de estas herramientas han incorporado (y de manera parcial) las características del ecosistema descrito en el presente artículo. La plataforma para la resolución de problemas multiobjetivo PISA [6] propone la creación de manera independiente de módulos que implementan algoritmos y describen los problemas. Esta plataforma provee una interfaz en modo texto para la ejecución de los algoritmos, que además incorpora módulo de monitorización de la ejecución y comparar el rendimiento de los algoritmos, así como una librería de módulos de resolución listos para su uso. En cuanto a la apertura de la plataforma, se acepta la publicación de módulos creados por los usuarios en el sitio web del proyecto, constituyendo el sistema actual más cercano al ecosistema que se propone en este artículo. Sin embargo, PISA carece de una plataforma y modelo para la gestión de componentes más allá de las interfaces básicos que especifican los métodos de los problemas y resolutores, donde además los módulos deben leer y escribir datos de ciertos ficheros de texto plano para establecer la comunicación entre ellos. PISA tampoco incorpora interfaces para establecer la relación con el conjunto adicional de herramientas de nuestra propuesta de ecosistema.

Otros frameworks de optimización metaheurística han incorporado algunas de las características propuestas para el ecosistema, como la adopción de desarrollos de los usuarios (aún cuando sea como ampliaciones al código fuente framework, en lugar del uso de plugins o componentes versionados independientemente e instalables en una plataforma de ejecución), como por ejemplo ECJ [1]. Otro pasos en esta dirección son la incorporación de una plataforma para la ejecución y monitorización de los experimentos en frameworks como HeuristicLab [18], Watchmaker [2], y OAT [8] entre otros, y la integración con otras herramientas, como puede ser el caso de JCLEC [17] en el proyecto KEEL [4].

Sin embargo en la actualidad, no existe ningún ecosistema que incorpore las características propuestas y las interfaces que permitan integrar los flujos de información de las herramientas de ayuda a la optimización con metaheurísticas.

6. Conclusiones y Trabajo Futuro

En el presente artículo se ha propuesto la creación de un ecosistema de herramientas software para la resolución de problemas de optimización con metaheurísticas y las tarea relacionadas con la investigación en este área. Así mismo se han descrito el conjunto de elementos que compondrían el ecosistema y los

flujos de información entre los mismos. La propuesta presentada en este artículo es conceptual por lo que aparecen como trabajos futuros: la creación de la infraestructura de soporte, la transformación de nuestro framework en una plataforma para el ecosistema y la especificación de las interfaces con el resto de elementos para generar la base del ecosistema propuesto. Así mismo se pretende identificar un conjunto de herramientas candidatas para cada elemento del ecosistema.

Agradecimientos

Este artículo ha sido financiado parcialmente por la Comisión Europea (FEDER) y el Gobierno de España a través del proyecto CICYT SETI (TIN2009-07366), y el Gobierno Andaluz a través del proyecto ISABEL (TIC-2533).

Referencias

[1] http://cs.gmu.edu/ eclab/projects/ecj/.

[2] http://watchmaker.uncommons.org/.

[3] http:www.eclipse.org/.

[4] J. Alcalá-Fdez, L. Sánchez, S. García, M. del Jesus, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V. Rivas, J. Fernández, and F. Herrera. Keel: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, 13(3):307–318, 2009.

[5] J. A.Parejo, J. Racero, F. Guerrero, T. Kwok, and K. Smith. Fom: A framework for metaheuristic optimization. *Computational Science, ICCS 2003. LNCS*, 2660:886–895, 2003.

[6] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. Pisa - a platform and programming language independent interface for search algorithms. pages 494–508. Springer, 2003.

[7] Jan Bosch. From software product lines to software ecosystems. In *Proceeding of the 13th Software Product Line Conference*, August 2009.

[8] J. Brownlee. Oat: The optimization algorithm toolkit. Technical report, Complex Intelligent Systems Laboratory, Swinburne University of Technology, 2007.

[9] K. Chakhlevitch and P. Cowling. Hyperheuristics: Recent developments. In *Adaptive and Multilevel Metaheuristics*, pages 3–29. 2008.

[10] P. Collet, E. Lutton, M. Schoenauer, and J. Louchet. Take it easea. In *PPSN VI: Proc. of the 6th Int. Conf. on Parallel Problem Solving from Nature*, pages 891–901, London, UK, 2000. Springer-Verlag.

[11] P. Collet and M. Schoenauer. Guide: Unifying evolutionary engines through a graphical user interface. *Artificial Evolution*, pages 203–215, 2004.

[12] Stefan Voß et al. *Optimization Software Class Libraries*. Kluwer Academic Publishers, 2002.

[13] http://www.juntadeandalucia.es/ repositorio/index.jsf.

[14] J. A. Parejo, P. Fernández, and A. Ruiz-Cortés. Qos-aware services composition using tabu search and hybrid genetic algorithms. In *Talleres de las JISBD. IADIS'08.*, 2008.

[15] J. A. Parejo, S. Lozano, P. Fernández, and A. Ruiz-Cortés. Benchmark of metaheuristic optimization frameworks. Technical report, Grupo de Ing. del Soft. Aplicada (ISA). Universidad de Sevilla, 2010. Disponible en www.isa.us.es.

[16] Ivan Valiela. *Doing Science*. Oxford University Press, 2001.

[17] S. Ventura, C. Romero, A. Zafra, J. Delgado, and C. Hervás. Jclec: a java framework for evolutionary computation. *Soft Computing*, 12(4):381–392, 2008.

[18] S. Wagner and M. Affenzeller. Heuristiclab: A generic and extensible optimization environment, 2005.