Código en Internet para resolver problemas de optimización: recursos existentes y buenas prácticas

Fernando Paz M., Angel A. Juan

Resumen

La publicación de código en Internet es una práctica masiva de los miembros de la comunidad que intenta intercambiar conocimiento y promover su prestigio personal. Sus resultados a simple escala son la innovación y generalización de este conocimiento y en acuerdos de mayor volumen son productos de software con impacto decisivo en el mercado, a través de paradigmas como el Software Libre. Este artículo desarrolla un análisis de estas publicaciones desde el punto de vista de comunidad, concentrado en el segmento de las aportaciones de código realizadas para la optimización de los problemas de corte y empaquetamiento. Estructura una búsqueda fundamentada en el comportamiento natural de los internautas para determinar los hallazgos relacionados; posteriormente clasifica los mismos por su tipología [Wascher 2007] y evalúa las características del contexto de su disposición en Internet. El resultado se refleja en tres aspectos: un resumen de las principales herramientas de software identificadas para la solución y disposición de estos problemas de optimización, una agrupación de licenciamientos aptos para estas publicaciones de acuerdo al sentido de colaboración determinado por sus autores, y, un paradigma para la publicación de nuevos trabajos de códigos con carácter científico principalmente, pero adaptable a otros tipos de comunidades.

Keywords: Cutting and Packing Problem, Open Source, Free code, best practices

---

1 Introducción

Los productos y herramientas en general han tenido una evolución histórica a través de la creación del conocimiento de distinta índole desde el punto de vista de pruebas (testing) hasta el punto de vista científico. El conocimiento de carácter científico, a menudo disperso para el enfoque de la solución, no ha tenido una adecuada aplicación y/o evaluación sobre el producto determinado, perdiendo una oportunidad de mejora o confirmación respecto al encaminamiento de la solución adoptada.
Partiendo de esta premisa y la tendencia de la globalización de la información es importante analizar siempre: códigos e implementaciones relacionados al problema a automatizar, en nuestro caso nos enfocamos en un problema recurrente y de múltiples aplicaciones como el Cutting and Packing (C&P) Problem; permitiéndonos valorar este conocimiento en el contexto del desarrollo de software, sus implicaciones, su validez, el esfuerzo real de estudiarlo, y, los beneficios que suponen su sintetización. En este concepto el C&P Problem nos ofrece muchas opciones a la hora de identificar y valorar estos recursos generando conclusiones y recomendaciones sobre su aporte real a la comunidad.

En la actualidad el proceso de desarrollo de software se ve inmerso en una implícita comunidad social [Hippel 2001], en donde la colaboración es la clave de la evolución maximizando las oportunidades de crecimiento y optimizando el producto final; este movimiento ha trastocado el concepto de los derechos de autor (copyright) en donde su aplicación patrimonial presenta una línea delgada entre la expresión del contenido y el contenido mismo de un trabajo dejando espacios para la adopción de conocimiento y su propagación, resultando en mejores soluciones sociales y comerciales [Hernández 2008], y, evolucionando los conceptos de modelos de negocio y paradigmas de desarrollo. Esta colaboración tiene muchas expresiones actuales y ha encontrado en Internet el espacio más adecuado para su crecimiento como foros, bibliotecas virtuales, repositorios de código, repositorios de proyectos, blogs personales y empresariales, y, tendencias más actuales como la Web 2.0 en donde la interacción del contenido cambia de sentido [O'REILLY 2007].

Los problemas de optimización tienen un sólido carácter científico, en donde esta comunidad busca generar mejores alternativas a través del diseño de potentes algoritmos basados en heurísticas simples o combinadas, generando soluciones aplicables en múltiples fines sociales y empresariales que mejoran la calidad de vida individual y colectiva. La sintetización natural de este trabajo científico, es la implementación del código informático que expresa al algoritmo, el mismo que tiende a ser publicado en un medio masivo de colaboración como la Internet. Nuestro interés ha sido estudiar estas publicaciones de un forma estructurada y objetiva desde el punto de vista de comunidad, identificando un conjunto de prácticas que beneficien la disposición de este conocimiento en favor de su evolución.

Identificamos y clasificamos a los hallazgos, de acuerdo a la tipología descrita por Wascher en el 2007 [Wascher 2007]. Para efectos de un análisis correcto del código nos hemos segmentado a aquellas implementaciones escritas con el lenguaje de programación Java y que tengan disponible su código fuente al público.
El apartado de Recursos en Internet presenta una exposición de los hallazgos con su respectivo análisis cuantitativo y cualitativo, remarcando además aspectos relevantes identificados a través del mismo.
Fundamentamos en la tercera parte un conjunto de buenas prácticas para la publicación de este tipo de trabajos en Internet, esquematizado a través de un nuevo paradigma para su disposición.
La última sección describe las conclusiones de esta investigación enmarcadas en los hallazgos y su análisis, así como las tendencias de la comunidad.

2 Recursos en Internet

El estudio ha sido completamente desarrollado sobre la Internet, con una muestra no probabilística segmentada a los problemas de optimización de corte y empaquetamiento (C&P) y el lenguaje Java para su implementación.
La ejecución de las búsquedas asociadas se determinaron en dos aspectos, el primero la selección de las ingenierías de búsqueda (Search Engine) en torno a la región de América/Europa y el segundo aspecto el comportamiento del usuario al seleccionar los resultados de las páginas mostradas por estas ingenierías de búsqueda (SERP) [Enge 2009].
En síntesis la ejecución se realizó sobre tres ingenierías: Google, Yahoo y Bing seleccionando los resultados dentro de las primeras cuatro páginas mostradas por cada una y filtrados por los criterios mencionados.
Esta metodología constituye un aspecto determinante en los hallazgos elevando la certeza del resultado obtenido contrastado con el comportamiento normal de la comunidad, puesto que si bien la materia científica de algoritmos y heurísticas tiene bases en repositorios de conocimiento como IEEE y ScienceDirect, el código como tal se comporta de forma regular a la ubicación de otra tipo de información no científica.
Hay que señalar otro aspecto fundamental para la determinación de estos hallazgos y se concentra en la refinación por categoría de los problemas de optimización de C&P, pues las búsquedas generales de implementaciones sobre este problema presentan escasos resultados, del mismo modo que las búsquedas ejecutadas sobre las categorías denominadas intermedias y refinadas [Wascher 2007]; en este contexto los hallazgos fueron trabajados sobre la categoría denominada básica; y se evidencia una tendencia de la comunidad que genera estas implementaciones, a describir sus trabajos de manera más general.
El resultado final son 24 hallazgos sobre las variantes del problema de optimización de C&P que son implementados en Java, aunque en este camino se identificó muchos más recursos acerca de librerías (frameworks) y repositorios de este tipo de información que del mismo modo serán descritos en los siguientes apartados.

2.1 Análisis

Identificadas las referencias, se exportaron los códigos fuente en un IDE de desarrollo para su estudio, determinando elementos básicos en su estructura y contexto. El análisis no pretende comparar soluciones óptimas por la naturaleza de la variedad de los problemas y factores que cubren cada una de estas implementaciones, y porque en los conceptos de heurística las

soluciones son aproximadas a las ideales; sin embargo, se reconocen ciertas características esenciales comunes en estos trabajos, que desde el punto de vista de comunidad, maximizan las posibilidades de compresión y adopción del conocimiento, posibilitando además procesos de mejoramiento y difusión del mismo (Figura A).

Nombre	**Descripción**	**Escala/Referencia**
Tipología	De acuerdo por lo descrito por Wascher 2007	Clasificación determinada para los tipos de problemas intermedios
Licencia	Mecanismo legal de liberación	• Open Source aprobada por la OSI • Libre • Privada • Privada Implícita • Propia, restringida a uso científico/educativo
Sintaxis	Estructura y escritura del código desde el punto de vista de claridad y comprensión	1(menor)-5(mayor)
Autores	Atribuciones de la obra	Detalle
Descripción	Documentación del trabajo, independientemente a los comentarios en el código, aspecto básico de la asimilación, mejoramiento y reutilización del mismo	1(menor)-5(mayor)
Testing Process	Proceso documentado o codificado para poner en funcionamiento el trabajo	Si = 1 y No = 0
Showcase	Documentación o adición de resultados al poner en funcionamiento la implementación, clarificando las salidas a esperarse de su ejecución	Si = 1 y No = 0
Sharable	Mecanismo de propagación del conocimiento en Internet	Si = 1 y No = 0
Funcionamiento	Determinación si el trabajo corre en circunstancias normales y con la información proveída razonable	Si = 1 y No = 0
Comentarios en Código	Nivel de comentarios en el código	Del 0% al 100%
Concepto Asociados	Información de heurísticas, documentos, algoritmos o teorías utilizadas	Detalle
Notas	Consideraciones adicionales	Detalle

Figura A. Características de las implementaciones

2.1.1 Análisis cuantitativo

Tipología

Basados en las descripciones de las implementaciones y las variables analizadas en su replicación a través del IDE del desarrollo, se clasificó las mismas de acuerdo a la categoría intermedia descrita por Wascher (Figura B).

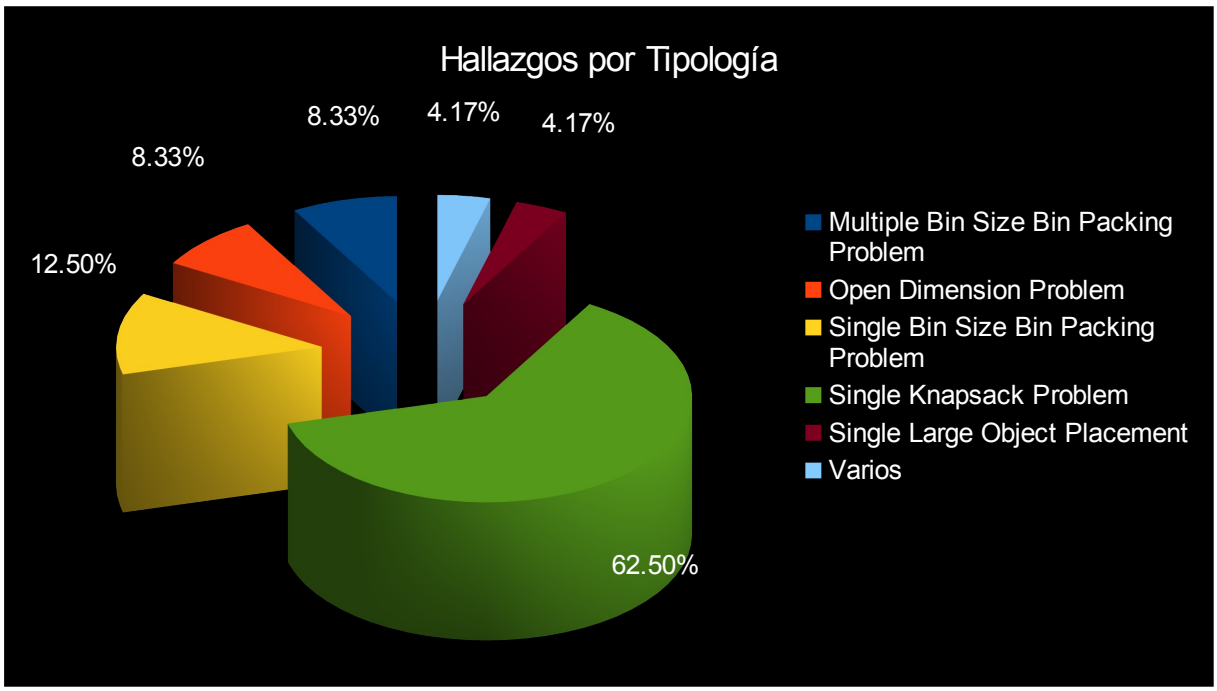

Figura B. Porcentajes de hallazgos por tipologías

Los valores expresados en porcentaje no reflejan la tendencia real por cada tipología puesto que existe una segmentación al lenguaje utilizado; en este contexto sería adecuado evaluar en un estudio los lenguajes influyentes en la implementación de algoritmos matemáticos estableciendo una correlación con los hallazgos para tener cifras más precisas. Sin embargo, si se puede diferenciar dos grupos macro de tipologías desarrolladas: la de "baja complejidad" en donde los objetos grandes e items pequeños presentan características simples y cubren el 80% de los trabajos dispuestos en Internet, y, los de "alta complejidad" donde las características son más complejas y competen un número mayor de variables a considerar constituyendo el 20% de los trabajos dispuestos en Internet.

Licencias

La estructura de las licencias es un tema clave a analizar pues la ausencia de esta información conlleva la restricción absoluta del uso de este trabajo, de acuerdo a la mayoría de legislaciones que tienen acuerdos con la OMC (Organización Mundial del Comercio) y la OMCI (Organización Mundial de Propiedad Intelectual) [Hernández 2008].
Muchos de estos hallazgos no presentan una licencia de uso, pero para efectos de un análisis correcto contactamos a los autores correspondientes para contrastar esta omisión con su deseo definitivo, pues debemos estar conscientes que al publicar nuestro trabajo en Internet formamos parte de una comunidad social que comparte recursos.
Debemos agradecer por su tiempo y gentil respuesta a:

- Michael Andrew Wascher mwash@uw.edu
- Param Sethi param.sethi@gmail.com
- Russell Martin Russell.Martin@liverpool.ac.uk
- Aviv Ron aviv1ron1@gmail.com
- Wenbin Zhu i@zhuwb.com
- Tim Rolfe rolfet@earthlink.net RolfeT@acm.org
- Bob Sedgewick rs@cs.princeton.edu

El resultado principal, es una denominación como "Libre" a un tipo de licencia que nos permite libertad absoluta de uso manteniendo el crédito de los autores y otra denominación como "Propia, restringida a uso científico/educativo" a una licencia más explícita pero que sostiene también el crédito a los autores (Figura C).

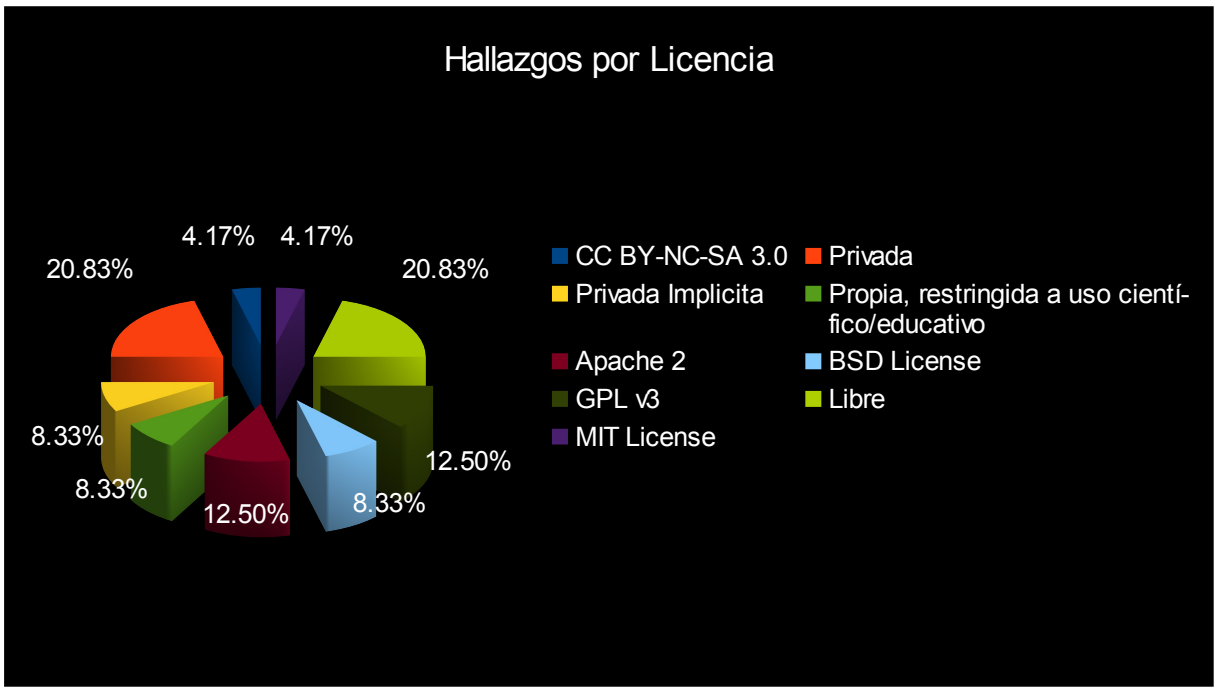

Figura C. Porcentajes de hallazgos por licencia

La licencia denominada "Libre" para efectos de este análisis, tiene dos características importantes: su libertad absoluta de uso que enmarca las cuatro libertades que definen al Software Libre [The GNU Operating System - What is free software?], y, mantener los créditos de autor petición que encaja en la mayoría de licencias definidas nuevamente para el Software Libre. Partiendo de esta premisa se puede agrupar las licencias en dos grupos: las Privadas y las Open Source (Figura D).

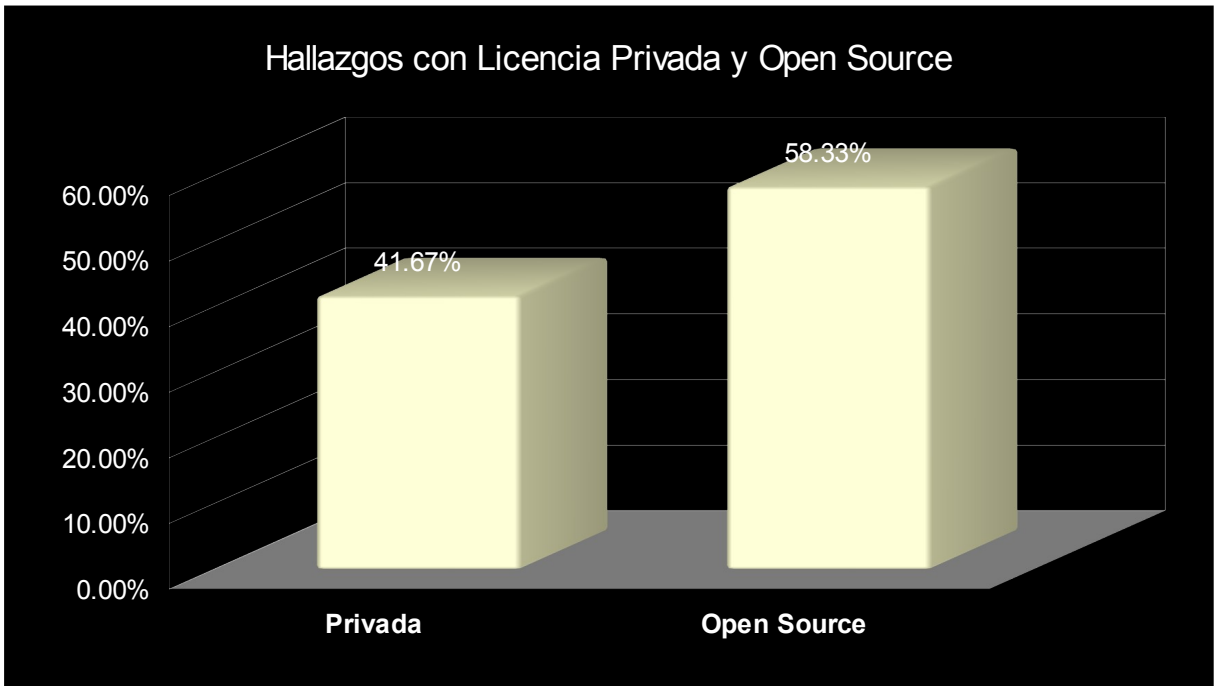

Figura D. Porcentaje de hallazgos con licencia privada y estilo open source

Se puede destacar una tendencia mayor de liberar las implementaciones bajo licencias al estilo Open Source y si sumamos el porcentaje de las licencias con fines sociales en general y no comerciales (CC BY-NC-SA 3.0 y Propia, restringida a uso científico/educativo), podemos enmarcar que el 70% de los autores desean compartir su trabajo con la comunidad libremente.

2.1.2 Análisis cualitativo

Sintaxis

Un aspecto relevante en la asimilación de un código programado es su estructura y escritura, de forma que el usuario pueda asimilar el conocimiento y verificar los resultados en: entornos de pruebas de escritorio, su ejecución real e incluso procesos de depuración; para esto es importante identificar modularidad en las secciones, distinción de procesos, clasificación de fuentes, nombramiento de variables y parámetros, etc. Evidentemente este aspecto dependerá mucho de la tecnología de implementación utilizada por la naturaleza de codificación; en nuestro análisis nos hemos basado en las nombradas Convenciones de Código Java [Sun Microsystems, Inc. - Java Code Conventions] aplicadas al nivel de complejidad de cada implementación dejándonos una media de 3.5 puntos (ver Figura A – característica Sintaxis) que muestra una condición aceptable de los códigos dispuestos en la Intranet.

Descripción/Documentación

La documentación de una implementación es complementaria a la codificación, si bien el código es el ejecutor de la teoría, se puede tratar a un trabajo en Internet como incompleto debido que no es útil para todo el publico en la comunidad. Los temas referentes a la instalación, configuración, entradas y salidas son puntos determinantes en la adopción de una tecnología. En nuestra escala de nivel de documentación, en general tenemos un puntaje de 2.9 (ver Figura A – característica Descripción) lo que quiere decir que es un tema desarrollado a un 60% en promedio.
Remarcamos aquí, que en muchas replicaciones y ejecuciones de las implementaciones este factor ha duplicado o triplicado el esfuerzo necesario para entender la solución y comprender sus resultados, pero motivados por el sentido de la investigación ha sido parte del proceso; sin embargo, dentro de una comunidad esta complicación puede derivar en la perdida de interés, incluso en muy buenos proyectos evaluados en esta investigación.
Puntualizando los problemas de optimización de C&P, se valoriza más aún este aspecto, al tener conceptos y teorías determinantes en la soluciones como las heurísticas o meta-heurísticas, las variables consideradas, el alcance del proyecto, las comparaciones, etc.

Testing Process/Show Case

Este aspecto es parte en realidad de la documentación del proyecto, pero se lo ha separado puesto que se considera básico en la definición de una implementación. Un usuario en la comunidad asimila este conocimiento básicamente al ejecutarlo, pero para esto debe conocer las entradas y poder comparar las salidas con las previstas por el autor, de esta forma al menos puede tener la idea completa de la solución sin necesidad de leer la documentación en caso de tenerla. Nuestro análisis muestra que el 79% de estos trabajos documenta o codifica el proceso de pruebas, pero solo el 46% deja documentada las salidas del proyecto.

2.2 Licencias

El análisis de los recursos encontrados nos lleva a agrupar las licencias en tres grupos: las "Open Source", las de "Fines Sociales" y las "Privadas"; las mismas que se evidencia, son relevantes para los autores de estos trabajos dispuestos en la Intranet.

Open Source

Aquí el autor cuenta con múltiples alternativas, pero la premisa del conocimiento científico es la evolución y colaboración. En este sentido la licencia GPL v3 [The GNU Operating System - GNU General Public License] y AGPL v3 [The GNU Operating System - GNU Affero General Public License] determina principalmente mantener los derechos de autor de la obra y publicar las mejoras realizadas al producto original, en los nuevos productos derivados o basados del mismo que se pretenda distribuir o vender. La versión de la licencia AGPL protege además el uso de productos derivados o basados en el original sobre la red, en los mismos términos que si fuesen distribuidos o vendidos.

Fines sociales

Los autores aquí pretenden ceder los derechos a fines sociales en general para lo cual es factible liberar sus trabajos con la licencia CC BY-NC-SA 3.0 (Attribution-NonCommercial-ShareAlike 3.0 Unported) [Creative Commons - Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0)], que garantiza el uso del software sin fines comerciales y mantiene los derechos del autor en los trabajos derivados o basados en el producto original.

Privadas

Aquí existen ya figuras legales estándar o personalizadas a través de expertos en derecho, sin embargo, por la naturaleza de la protección de la propiedad intelectual y derechos de autor, se puede estimar suficiente el presentar el texto "Nombre Apellido @Copyright Año, Todos los derechos reservados" en las publicaciones.

2.3 Frameworks

Durante la evaluación de estos códigos, observamos el uso de otros proyectos relacionados a la solución de problemas de optimización, que presentan muy avanzadas técnicas de modelamiento de problemas e implementaciones de meta-heurísticas relacionadas a los mismos. Aquí resumimos estas librerías o frameworks como opciones que se pueden tomar en cuenta para futuros trabajos en la solución de los problemas de optimización de C&P.

Choco, es un proyecto Open Source liberado bajo la licencia BSD [Open Source Initiative - The BSD 2-Clause License] que implementa la funcionalidad para resolver problemas de optimización enmarcados en la denominada programación con restricciones (Constraint Programing) [Pisinger 2007] que ha sido aprobada por la la JCP [The Java Community Process - JSR #331 Constraint Programming API Final Approval Ballot] y constituye el estándar para resolver problemas que deben satisfacer restricciones. Presenta una actividad muy intensa y una comunidad sólida de respaldo que contribuye con su evolución [CHOCO - Página Inicial].

GA Playground, es un conjunto de herramientas en donde la comunidad puede definir y ejecutar sus propios problemas de optimización, utiliza los conceptos de algoritmos genéticos para generar los soluciones, posibilitando variaciones al tipo de problema propuesto. Tiene una licencia "Libre" pero no formal, actualmente soporte hasta la versiones 1.4 de JDK y es un proyecto individual. [Artificial Life and Other Experimients - The GA Playground (Genetic Algorithms Toolkit)].

JGAP, se desarrolló como un componente que provee los principales mecanismos genéticos programados que pueden ser fácilmente aplicables al principio de evolución en la solución de problemas. Permite la implementación propia de una función de optimización para enmarcar la mejor solución. Un framework liberado con la licencia GLP v3 que cuenta con un amplia comunidad en su evolución y desarrollo, y es parte de múltiples soluciones desarrolladas en el mercado [Java Genetic Algoriths Package - Página Inicial].

JOpt, es librería que implementa el lenguaje de programación para la optimización (OPL) [Hentenryck 1999], que facilita la formulación y solución de problemas de optimización y presenta soporte a la programación con restricciones. Esta liberado con la licencia CPL (Common Public License Version 1.0) [Open Source Initiative - Common Public License Version 1.0 (CPL)], presenta estructura de soporte a comunidad, es un trabajo bastante elaborado aunque su última versión se publicó en el 2007 [Java OPL Implementation - Página Inicial].

Drools Planner, es el framework desarrollado por la comunidad JBOSS, que tiene una visión completa y más amplia de los problemas considerados como NP hard [Wikipedia - NP-hard], implementa soluciones para varias tipologías de problemas de C&P basados en un amplio rango de heurísticas y meta-heurísticas relacionadas al tema; permite la extensión del concepto con implementaciones propias de nuevos algoritmos de solución y considera una amplia estructura para la definición de restricciones a través de reglas declaradas. Fue creado

como parte del producto Drools que busca automatizar procesos referentes a la logísticas de una organización y esta liberado con la licencia Apache 2 [The Apache Software Fundation - Apache License, Version 2.0], presentando una completa documentación con una comunidad robusta que permite su evolución, es usado por múltiples soluciones en el mercado [JBoss Community - Drools Planner].

2.4 Repositorios

Los hallazgos presentados en este trabajo se pueden clasificar en dos grupos: ”publicaciones con fines culturales” y “publicaciones que buscan generar producto”. Ambas tienen un sentido de contribución a la comunidad y pretenden evolucionar las mismas, al menos de acuerdo al 70% de licenciamientos de estos hallazgos que privilegian varias libertades.

Desde el punto vista científico y educacional la tarea de publicar código en Internet puede resultar compleja puesto que no se espera necesariamente mantener temas relacionados a versionamientos, foros, lista de errores, etc., por consiguiente mostramos tres opciones complementarias para la disposición de estos trabajos de acuerdo a necesidades objetivas:

Algorito, en es una solución realmente muy recomendable para la comunidad científica, puesto que permite publicar el código en forma de archivos comprimidos, además de posibilitar documentar, titular, licenciar (solo con licencias calificadas como Open Source), agregar palabras descriptivas (conocidas como labels o keywords), especificar el lenguaje de implementación, documentar referencias científicas, agregar imágenes e incluso relacionar otras publicaciones dentro del mismo servicio, para cada trabajo realizado. Estas características sin duda cubren el contexto esperado en la publicación de información científica y potencia sus posibilidades y referencias en los trabajos futuros. Sin embargo, también presenta espacio para comentarios/contribuciones de la comunidad en general y para la propagación del contenido a través de los medios sociales [Algorito - Página Inicial]. Se puede ubicar a este servicio como adecuado para el grupo de publicaciones con fines culturales

Google Code, presenta una solución más robusta pero sin llegar a ser compleja que puede contemplar necesidades sencillas de códigos, como proyectos de mayor dimensión. Aquí la publicación no contempla un punto de vista científico pero da espacios para describir al proyecto en general y poder documentarlo suficientemente a través de una Wiki [Wikipedia - Wiki] en la cual los miembros de la comunidad pueden participar abiertamente siempre que sean colaboradores reconocidos por el autor. Se puede titular el trabajo y etiquetarlo con palabras descriptivas, pero la característica más relevante es su control de versiones asociada, con opciones de tecnología como Git, Mercurial y Subversion, con la cual el autor puede evolucionar su código de una forma estructurada con los históricos de sus avances y versionamientos necesarios de acuerdo a su criterio. Posee un sistema de registro de incidencias con el cual la comunidad puede contribuir abiertamente con reportes de problemas o correcciones. Los colaboradores del proyecto los inscribe el autor de acuerdo a su criterio con la posibilidad de asignarles roles específicos de acción en el proyecto [Google Code - Permissions]. Este servicio puede ser utilizado tanto para el grupo de publicaciones con fines culturales como para publicaciones sobre productos, es importante señalar que los licenciamientos disponibles para el uso del servicio son todos aquellos considerados por la OSI [Open Source Initiative - Licenses by Name].

SourceForge, es una alternativa importante para las publicaciones de proyectos; si bien existen numerosas opciones para la publicación de este tipo de código como GitHub o el mencionado Google Code, SourceForge tiene más tiempo en el mercado con una historia pionera en el mundo del software libre [Maguire 2007], esto lo convierte en la actualidad en uno de los sitios con más proyectos de software y más relevantes para la comunidad. Cuenta con las mismas características de Google Code pero adiciona servicios como Blog, Bases de Datos MySQL, sub-proyectos, etc. Presenta un amplia lista posibilidades para el licenciamiento del trabajo e incluso licenciarlo con multiples licencias, todas consideradas como Open Source; la administración de usuarios es más completa permitiendo definir grupos personalizados y asignar los permisos y usuarios a estos. Además esta administración cuenta con opciones para adicionar imágenes, asociar los proyectos con categorías bien definidas, incluir enlaces

externos como el sitio oficial Web y una bitácora de auditoría de los cambios dentro de toda esta infraestructura. Las publicaciones que tenga por objetivo derivar en proyectos pueden escoger este servicio para operar su evolución.

3 Buenas prácticas

El proceso de investigación de los códigos libres dispuestos en Internet nos deja un conocimiento amplio referente a los autores y la adaptación de sus trabajos en la comunidad científica y en general.
Es este apartado se propone un nuevo paradigma para la publicación de códigos en Internet orientado principalmente a la comunidad científica, y puede ser adoptado también por los miembros de la comunidad en general que comparten sus trabajos en busca de innovación y adopción de conocimiento.
Inicialmente identifiquemos seis aspectos que componen un trabajo codificado: título, documentación, código, licencia, lugar de publicación y contactos. Este conjunto de temáticas son explicitas o implícitas cuando un autor decide publicar su trabajo, y de una o de otra forma sintetiza una imagen asociada a su obra.
Enumeramos a continuación los pasos secuenciales recomendados y encajados en el extracto de esta investigación, para los científicos y la comunidad en general:

- Definición del título
- Documentación de la obra
- Escritura del código
- Definición del licenciamiento
- Selección del lugar de publicación
- Difusión de la obra

Definición del título

Nombrar un trabajo de implementación es una tarea tan importante como la de elegir el título de un artículo científico. En la evaluación de las SERP's presentadas por las ingenierías de búsqueda en Internet, se determina el grado de influencia de el concepto general asociado al código expuesto, y de hecho los buscadores priorizan como primer hito de calificación al distintivo claro de titulación para presentar los resultados a los usuarios.
En consecuencia un título muy general como "Implementación para la optimización" o muy específico como "Códigos para el MHLOPP" disminuyen las oportunidades de la transcendencia del conocimiento en la comunidad; y en mayor escala cuando el título de la página presenta algo como "Knapsack.java".
Se recomienda presentar un concepto general seguido de conceptos más especializados, por ejemplo: "Bin Packing Problem: Implementación desarrollada con conceptos evolutivos Genetic Algorithm".
Este nombramiento sin duda maximiza las oportunidades del ranking en las ingenierías de búsqueda y cumple con el objetivo de colaboración con la comunidad que desarrolla conocimiento alrededor de estos conceptos.

Documentación de la obra

El análisis cualitativo de los hallazgos presenta algunos aspectos interesantes identificados en el proceso de documentación de un trabajo, presentamos a continuación una recomendación acerca de los mismos:
Elabore una sinopsis: Aquí se presenta un resumen de la estructura de la implementación desde el punto de vista teórico, detallando el problema que pretende solucionar y que conceptos en general se aplicaron para su solución; las entradas o variables consideradas y las salidas esperadas y explicadas. En los casos de existir una teoría más extensa y compleja es recomendable dejar las referencias a los artículos, conferencias, libros, etc.
Determine las palabras descriptivas: Las ingenierías de búsqueda evalúan el contenido de la página Web utilizada y reconoce esta lista de palabras dispuestas en la publicación, asociando el contenido con las mismas. Por consiguiente es importante mencionar aquí temas como las heurísticas, el lenguaje utilizado, la estrategia de programación (por ejemplo Constraint Programming), el nombre de los conceptos teóricos, las tecnologías, el nombre del problema,

y, todas aquellas descripciones claves que considere el autor válidas dentro de la disciplina que envuelve a la obra.
Explique la codificación: Dependiendo de la profundidad y complejidad del código se puede disponer una sección con la explicación del mismo, con temas como descripciones de funciones, descripciones de variables/objetos, estructura de archivos fuente, dependencias de librerías, proyectos usados y estructura del proyecto en general. Es inevitable sin embargo, en proyectos complejos generar esta documentación y de manera exhaustiva, ya que posibilita a un usuario nuevo, que puede ser un colaborador potencial para su evolución, el uso de este conocimiento y su aplicación real en problemas y ambientes deferentes a los estimados al crear la obra.
Deje un ejemplo: Un usuario nuevo a la implementación debe al menos conocer por donde comenzar, pues si no encuentra una explicación del código entra en un proceso lento y complejo de descubrir como lo debe ejecutar, como debe definir las entradas y como debe interpretar las salidas. Este aspecto a menudo deriva en una evaluación incompleta acerca de todo el alcance de la solución y trunca su aplicación en proyectos en otros ambientes o con otras características. Es mucho más importante aún, si no se documentó la codificación, proveer de esta documentación en la publicación de una obra, buscando complementar la difusión correcta del conocimiento para la comunidad.

Escritura del código

Al codificar la solución se recomienda considerar tres aspectos: la herramienta de desarrollo, las reglas de escritura y la documentación del código.
Herramientas de desarrollo: Hacemos énfasis en esta recomendación y animamos a los autores a no usar simples editores de texto para desarrollar sus trabajos, por dos motivos básicos: el "primero" tiene que ver con las convenciones de codificación presentes en la mayoría de lenguajes de programación que estandarizan la estructura del nombramiento de variables y de funciones o métodos de forma que el código sea más limpio, más legible, más comprensible y se adapte a otros requerimientos que aseguran la compatibilidad con otras tecnologías; aquí, los IDE de desarrollo como Eclipse o Netbeans contemplan en su funcionamiento, implícita o explícitamente a través de opciones de configuración, dichas directrices, evidenciando estas deficiencias en tiempo de codificación a los autores de la obra para tomar las respectivas acciones; esta característica despreocupa a los autores de la temática y le permite concentrar su esfuerzo en el desarrollo de su obra. El "segundo" motivo son las operaciones que se pueden aprovechar con estas herramientas como la generación de documentos completos acerca de las descripciones y comentarios en el código que se conoce en Java como API Docs, además de integraciones con el sistema de control de versiones elegido como SVN facilitando el versionamiento, opciones de empaquetamiento de los fuentes dispuestos a la comunidad, facilidad en los procesos de pruebas y visualización de resultados entre otros.
Reglas de escritura: En general las convenciones de programación de cada lenguaje recomiendan ciertos aspectos en la escritura del código, y que depende de su naturaleza, por ejemplo en Java tenemos el concepto de modificadores que no aplican en el lenguaje C. Si bien aplicar todo el estándar resulta un trabajo extra pesado, mostramos a continuación un conjunto de especificaciones a considerar en la codificación en cualquier lenguaje:

- Estructure los fuentes en directorios, de forma que se agrupen aspectos de la soluciones como fuentes con operaciones utilitarias, fuentes que representan definición de problemas, fuentes que solventan la seguridad del aplicativo, etc. Esto en lenguaje Java se lo hace explícitamente a través de definición de paquetes y en lenguajes como PHP son directorios; su clasificación simplifica el proceso de compresión y mantenimiento, y presenta modularidad.
- Indente (sangrado) el código siempre que defina bloques, sean referentes a bucles, funciones o métodos, o incluso al definir las llamadas clases en la denominada programación orientada a objetos presente en la mayoría de lenguajes de programación. Utilice cuatro caracteres y nunca el tabulador, la razón es que el tabulador es interpretado con un número de espacios muy dependiente al sistema operativo o su configuración, y al comparar dos archivos fuentes a través de software especializados, una línea de instrucción como i = i + 1 puede ser tratada como diferente solo por presentar al inicio un espacio declarado con el tabulador de diferentes

sistemas base. Por otro lado 4 caracteres son suficientes para denotar la estructura del código y son distinguibles a simple vista contribuyendo con la compresión del mismo.

- Nombre descriptivamente sus clases, métodos, funciones, parámetros y variables; utilice sustantivos y no siglas para estas definiciones siempre procurando denotar el rol de cada una en el código. Por ejemplo pensemos en una función que calcule el factor de impacto de un artículo; de acuerdo a las convenciones en Java debemos iniciar con minúsculas su nombramiento, entonces la definimos como calculoFactorImpacto que es muy coherente con lo que hace; por otro lado esta función recibe dos parámetros: el identificador digital del objeto y el número de citaciones en el año corriente, manteniendo el mismo criterio podemos re-definir la función como calculoFactorImpacto(DOI, numCitAnoCor) en donde usamos DOI como siglas ya que son ampliamente conocidas y no representan confusión, y para la segunda variable usamos un nombre corto que resume su expresión evitando el uso del nombramiento completo pero manteniendo su significado central. Es claro por lo expuesto que las malas elecciones son definiciones como calcular(m,n) que expresen la función referida anteriormente, materializando un punto de confusión y poca compresión del código.
- Codifique instrucciones limpias: utilice una línea por cada sentencia, declare una variable/atributo por línea, ubique siempre las llaves en cada bucle así este contenga una única instrucción, procure utilizar espacios en las asignaciones y entre operadores (como por ejemplo al definir i=i+1 que debería ser i = i + 1), use saltos de línea cuando observe que su instrucción es demasiado extensa saliendo del contexto de su pantalla, utilice mayúsculas para definir constantes y use paréntesis en instrucciones compuestas.

Documentación del código: Esta documentación es complementaria a la del proyecto en general y pretende clarificar la estructura y función de los fuentes del código liberado en Internet. Afortunadamente el proceso es muy bien soportado por los IDE de desarrollo y presenta tres secciones principales:

- El encabezado del archivo fuente es el lugar donde se ponen las referencias del autor así como la licencia bajo la cual se libera el mismo, aquí además se puede incluir los datos de contacto de este autor.
- Sobre cada definición de las clases, funciones/métodos y atributos se incluye un bloque de comentarios en donde se explica exactamente su función y su rol dentro de la solución; en el caso de las funciones y métodos, además se incluye la descripción de los parámetros de entrada y la salida esperada después de su ejecución (en Java existen tags especiales a incluir que posteriormente se usan para la auto-generación de un documento llamado API JavaDoc [ORACLE - How to Write Doc Comments for the Javadoc Tool: Tag Conventions]). Las variables se documentan con comentarios sencillos y generalmente ubicados al final de la misma línea de instrucción por ejemplo como:
  String DIO; //digital object identifier
- La última sección, la componen los comentarios simples ubicados entre las líneas de las instrucciones descritas, las cuales toman gran importancia especialmente cuando las variables declaradas en el fuente no han sido nombradas con coherencia, por ejemplo al iterar el "arreglo1".

Definición del licenciamiento

En el apartado de licencias hemos descrito tres posibilidades para la liberación del código por los autores, es necesario aquí analizar cual es el sentido que deseamos expresar a la comunidad para la adopción del conocimiento, y mucho más importante cual es el futuro que esperamos que tenga este conocimiento. Nuestras recomendaciones han sido objetivas al fin derivado del análisis de los trabajos que hoy circulan en Internet; en cualquier caso es necesario y útil para el autor y la comunidad saber las reglas que rigen sobre su obra como una acción intencional en la publicación de la misma y no como consecuencia de una omisión.

Selección del lugar de publicación

Los blogs se han convertido en el lugar masivo de publicación de información además de la páginas personales de cada autor, sin embargo, deja de ser un medio adecuado para un

análisis correcto de la funcionalidad y resultados de códigos libres; por lo que se requiere a menudo poder descargar el código de forma compacta, en archivos comprimidos por ejemplo. Además muchos trabajos son demasiados complejos para disponerlos por estos medios y la retro-alimentación que podría tener su autor se ve limitada a las funcionalidades de estas plataforma tradiciones. En este sentido presentamos en la sección de repositorios unas alternativas que se recomienda considerar de acuerdo al alcance que tenga la solución. En el caso de los problemas puntuales de optimización es evidente que el mejor lugar es el servicio de Algorito, y para proyectos más ambiciosos Google Code representa una selección ideal; sin embargo, si su visión en el futuro es un framework de muchas características las recomendación es usar el servicio de SourceForge.

Difusión de la obra

Esta recomendación puede considerarse opcional, la puntualizamos tomando en cuenta el nuevo paradigma de difusión de información en Internet. Si bien los contenidos científicos tienen lugar en revistas, libros, congresos, etc. especializadas/os, los códigos como tal no tienen un espacio determinado, por ello muchos repositorios ya incluyen la funcionalidad de difusión del conocimiento a través de redes sociales y mecanismos como el correo electrónico; y mucho más allá las sociedades oficiales, institutos de investigación y universidades poseen espacios ya en estas redes y grupos sociales; de tal forma que es muy acertado considerar, después de publicado un trabajo, propagarlo por estos medios y dejar las opciones necesarias para que los usuarios que asimilan este conocimiento puedan difundirlo del mismo modo. Incluso hay redes especializadas en temáticas como la llamada Ohloh [Ohloh - Página Inicial] que constituye un punto de encuentro de los colaboradores del proyectos Open Source en todo el mundo posibilitando estudios más avanzados sobre el comportamiento de esta comunidad.

4 Conclusiones

El análisis de la disposición de los códigos en Internet, evidencia que existe un bajo porcentaje de proyectos robustos y que muestran innovación para la comunidad, muchos de ellos registran una inactividad prolongada y no han evolucionado, otros son opuestos presentando mayor solvencia y más funcionalidad; su diferencia básica es el licenciamiento y su disposición como Open Source permitiendo la producción de grandes productos como Drools Planner o JGAP. En el caso de los problemas de optimización y la comunidad científica, la implementación responde a una teoría y este es el aporte a la comunidad, sin embargo, la única forma de validarla es ponerla en usos reales y aquí la comunidad del software puede colaborar con la ciencia de forma que la teoría pueda crecer con la práctica en un ambiente con múltiples variables y exigencias, e incluso con adaptaciones a conceptos de optimización de código y recursos de cómputo. Los mecanismos del software libre pueden proteger a los científicos y sus obras, y a la vez potenciar su trabajo ganando ambas partes; por un lado el autor se ve beneficiado con nuevos resultados en la aplicación del conocimiento y sus posibles mejoras para tener bases más sólidas y teorías más adaptadas y proyectadas, y, por otro lado la comunidad construye soluciones enmarcadas en estos avances generando comercio y empleo en sociedades con realidades distintas.

El análisis cualitativo nos deja un vacío en la documentación de los proyectos, y si medimos el esfuerzo real de cumplir todas estas expectativas encontramos un trabajo más exigido y que requiere de más tiempo; pero, como los autores pueden evitar este exhausto pero necesario trabajo en sus obras?. Aquí la comunidad del software y en general puede ayudar pues si la licencia y la infraestructura de publicación de la obra es adecuada otras personas gestionarán estos aspectos resultando en un trabajo cada vez más completo y compresible que nuevamente beneficie al autor y a sus usuarios.
Una evidencia interesante de esta necesaria interacción entre la ciencia y la comunidad del software es la llamada Constraint Programing ahora parte de los estándares publicados por la JCP; en este contexto el código de una obra publicada por un científico tiene que adaptarse a esta especificación para ser compatible con el mercado de soluciones informáticas y pueda mostrar su conocimiento en problemas reales. Esta transición puede ser, bien asimilada por el autor o permitir la participación de la comunidad, aislándolo de metodologías y definiciones que lo alejan de sus bases reales de investigación.

5 Referencias:

Harald Dyckhoff, Enero 1990, A typology of cutting and packing problems, I. European Journal of Operational Research, Volume 44, Issue 2, p. 145-159.

Eric Enge, Stephan Specen, Rand Fishkin, Jessie Stricchiola, Octubre 2009, The Art of SEO: Mastering Search Engine Optimization (Theory in Practice), p. 27-79.

Jordi Mas Hernández, David Megías Jiménez, Jesús M. González Barahona, Joaquín Seoane Pascual, Gregorio Robles, Febrero 2008, Software Libre: Introducción al software libre, p. 42-58.

Eric von Hippel, Julio 2001, Innovation by User Communities: Learning From Open-Source Software, MIT Sloan Management Review, Volume 42, Issue 4, p. 82-86.

E Hopper, B.C.H Turton, Enero 2001, An Empirical Investigation of Meta-heuristic and Heuristic Algorithms for a 2D Packing Problem, II. European Journal of Operational Research, Volume 128, Issue 1, p. 34-57.

Tim O'REILLY, Abril 2007, Design Patterns and Business Models for the Next Generation of Software, Communications & Strategies, No. 1, p.17-37.

Pisinger, D (Pisinger, David); Sigurd, M (Sigurd, Mikkel), Invierno 2007, Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem, Informs Journal on Computing Volume:19 Issue: 1, p. 36-51.

Gerhard Wäscher, Heike Haußner, Holger Schumann, Diciembre 2007, An improved typology of cutting and packing problems, III. European Journal of Operational Research, Volume 183, Issue 3, p. 1109-1130.

Pascal Van Hentenryck, Enero 1999, The OPL Optimization Programming Language, The MIT Press.

Algorito, Página Inicial, http://www.algorito.com/ (Junio 2012).

Artificial Life and Other Experimients, The GA Playground (Genetic Algorithms Toolkit), http://www.aridolan.com/ga/gaa/gaa.html (Junio 2012).

CHOCO, Página Inicial, http://www.emn.fr/z-info/choco-solver/ (Junio 2012).

Creative Commons, Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0), http://creativecommons.org/licenses/by-nc-sa/3.0/ (Junio 2012)

Google Code, Marzo 2011, Permissions, http://code.google.com/p/support/wiki/Permissions.

Google Code, Página Inicial, http://code.google.com/ (Junio 2012).

James Maguire, Octubre 2007, The SourceForge Story, http://www.datamation.com/cnews/article.php/3705731.

Java Genetic Algoriths Package, Página Inicial, http://jgap.sourceforge.net/ (Junio 2012).

Java OPL Implementation, Página Inicial, http://jopt.sourceforge.net/index.php (Junio 2012).

JBoss Community, Drools Planner, http://www.jboss.org/drools/drools-planner (Junio 2012).

Ohloh, Página Inicial, http://www.ohloh.net/ (Junio 2012).

Open Source Initiative, Licenses by Name, http://www.opensource.org/licenses/alphabetical (Junio 2012).

Open Source Initiative, Common Public License Version 1.0 (CPL), http://www.opensource.org/licenses/cpl1.0 (Junio 2012).

Open Source Initiative, The BSD 2-Clause License, http://www.opensource.org/licenses/BSD-2-Clause (Junio 2012).

ORACLE, How to Write Doc Comments for the Javadoc Tool: Tag Conventions, http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#tag (Junio 2012).

SourceForge, Página Inicial, http://sourceforge.net/ (Junio 2012).

Sun Microsystems, Inc., Septiemre 1997, Java Code Conventions, VI. http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html.

The Apache Software Fundation, Enero 2004, Apache License, Version 2.0, http://www.apache.org/licenses/LICENSE-2.0.

The Apache Software Fundation, Febrero 2012, Coding Standards, http://commons.apache.org/net/code-standards.html.

The GNU Operating System, Junio 2012, What is free software?, http://www.gnu.org/philosophy/free-sw.html.

The GNU Operating System, Junio 2012, GNU General Public License,

http://www.gnu.org/licenses/gpl.html.
The GNU Operating System, Junio 2012, GNU Affero General Public License, Junio 2012, http://www.gnu.org/licenses/agpl.html.
The Java Community Process, JSR #331 Constraint Programming API Final Approval Ballot, http://jcp.org/en/jsr/results?id=5311 (Junio 2012).
Wikipedia, Febrero 2012, Comparison of free and open source software licenses, V. http://en.wikipedia.org/wiki/Comparison_of_free_and_open_source_software_licenses.
Wikipedia, Mayo 2012, NP-hard, http://es.wikipedia.org/wiki/NP-hard.
Wikipedia, Junio 2012, Wiki, http://es.wikipedia.org/wiki/Wiki.

APENDICE Hallazgos

The JBoss Drools team, Drools Planner, http://www.jboss.org/drools/drools-planner.
Javadream, Continuous knapsack in Java, http://www.algorito.com/algorithm/continuous-knapsack-java.
Javadream, 0-1 Knapsack problem in Java, http://www.algorito.com/algorithm/0-1-knapsack-problem-java.
Dr. Matthew Hyde, Code and Other Research Resources, http://www.cs.nott.ac.uk/~mvh/resources.php.
Chinmay Lokesh, Bin Packing Problem – combinatorial NP-hard problem, http://chinmaylokesh.wordpress.com/2011/02/20/bin-packing-problem-combinatorial-np-hard-problem/.
Robert Sedgewick and Kevin Wayne, Knapsack.java, http://introcs.cs.princeton.edu/java/96optimization/Knapsack.java.html.
Alexander Campbell, Power Set and Knapsack Problem, http://www.rimohead.com/code/powersetKnapsack.html.
Param Sethi, Dynamic Programming - KnapSack Problem Java Implementation, http://www.params.me/2011/11/dynamic-programming-knapsack-problem.html.
Russell Martin, Complexity of Algorithms (COMP202) Home Page (Winter/Spring 2012), http://www.csc.liv.ac.uk/~martin/teaching/comp202/.
Knapsack problem, http://www.codeforge.com/article/176265.
Timothy Rolfe, Floating Point Knapsack Problem — Specimen Data and Code, http://penguin.ewu.edu/cscd320/Topic/Strategies/Knapsack.html.
Tabiul Mahmood, ga-container, http://code.google.com/p/ga-container/.
Babji Prashanth, Chetty, Sample project demo'ing 'Bin Packing's Best Fit Algo', http://code.google.com/p/bin-packing/.
Scott Grissom, The 0-1 Knapsack Problem, http://jhave.org/learner/misc/knapsack/knapsack.shtml.
Y. Jayarathina Madharasan, The 0/1 Knapsack problem, http://www.planet-source-code.com/vb/scripts/ShowCode.asp?txtCodeId=6372&lngWId=2/.
Shmulik London, Knapsack Example (Bruteforce), http://www.cs.huji.ac.il/~popcorn/developer/examples/knapsack/.
Mike Wascher, Knapsack Algorithm in Java, http://coma-coding.com/blog/algorithms/knapsack-algorithm-in-java.
Codeweblog, Greedy method and backtracking to solve, "backpack .0 / 1 knapsack problem" - Java implementation, http://www.codeweblog.com/greedy-method-and-backtracking-to-solve-backpack-0-1-knapsack-problem-java-implementation/.
Codeweblog, Greedy algorithm - Three greedy selection strategy - 0-1 knapsack problem, http://www.codeweblog.com/greedy-algorithm-three-greedy-selection-strategy-0-1-knapsack-problem/.
Codeweblog, 0-1 knapsack problem, http://www.codeweblog.com/0-1-knapsack-problem/.
Sven Kiesewetter, BinPacking, http://sourceforge.net/projects/binpacking/.
Todd Neller, Resources for Teaching Stochastic Local Search, http://cs.gettysburg.edu/~tneller/resources/sls/.
Eduardo Perez Mederos, Miguel Monterrey Varela, Jaime Gonzalez Valdes, Oscar Mateos Lopez, ullbinpacking, http://code.google.com/p/ullbinpacking/.
A. Juan, Quim Castella, Hybrid Algorithms for Applied Optimization, http://dpcs.uoc.edu/joomla/images/stories/software/SS-GNEH_110407.zip.