Flujo de trabajo para la experimentación colaborativa en Ingeniería del Software guiada por búsqueda

Isabel María del Águila, José del Sagrado y Alfonso Bosch

Departamento de Informática, Universidad of Almería,
Ctra. de la Playa s/n, 04120 Almería, Spain
{imaguila,jsagrado,abosch}@ual.es

Resumen La Ingeniería del Software Guiada por Búsqueda persigue reformular problemas de Ingeniería del Software que a menudo comprenden objetivos en conflicto, como problemas de optimización. Así, las técnicas que se aplican en esta disciplina buscan una o un conjunto de soluciones casi-óptimas en un espacio de soluciones candidatas con la ayuda de una función de aptitud que les permita distinguir las mejores soluciones. La naturaleza estocástica de los algoritmos de optimización requiere de la repetición de las búsquedas para mitigar los efectos de la aleatoriedad. A la hora de comparar algoritmos, el investigador comparará los resultados con mejor calidad (mejores valores en la función de aptitud, en indicadores de calidad y rendimiento) devueltos en las búsquedas, lo que conlleva un trabajo adicional por parte del investigador. La sobrecarga que implica esta actividad puede aminorarse si la experimentación se enfoca de manera colaborativa. Este artículo propone un flujo de trabajo para la experimentación colaborativa basado en resultados e indicadores de calidad y rendimiento.

Keywords: metaheurísticas, experimentación cooperativa, indicadores de calidad

1. Introducción

La Ingeniería del Software basada o guiada por búsqueda une los campos de algoritmos de Inteligencia Artificial e Ingeniería de Software. En la literatura encontramos numerosos trabajos que ponen en valor esta colaboración y que según Harman et al. [15] se pueden clasificar en tres áreas principales: 'Clasificación, aprendizaje y predicción en Ingeniería del Software', 'Ingeniería del Software probabilística' e 'Ingeniería del Software guiada por búsqueda'. En clasificación, aprendizaje y predicción algunos autores proponen modelos de predicción del riesgo [18], o bien estudian los defectos [17] o predicen el esfuerzo necesario para la ejecución del proyecto [31]. Técnicas probabilísticas basadas en redes Bayesianas modelan cuestiones de Ingeniería del Software para el control de calidad [19], la predicción de defectos [29] o el estudio de los requisitos [4] La Ingeniería

del Software guiada por búsqueda es un campo de investigación cuyo objetivo es reformular problemas de Ingeniería del Software para automatizar la construcción de soluciones a los problemas que abarcan desde el análisis de requisitos hasta la refactorización y mantenimiento [15].

Una vez que un problema de Ingeniería del Software se define como un problema de búsqueda pueden aplicársele diferentes aproximaciones metaheurísticas que nos darán soluciones de distinta calidad, lo que unido a la naturaleza estocástica de los algoritmos de optimización, hace que, en la mayoría de los casos, sea necesario recurrir a la experimentación, en lugar de al análisis puramente teórico, a la hora de evaluar las propuestas de investigación en este campo de investigación.

Un experimento consiste en la manipulación y observación de la realidad para el estudio detallado de un fenómeno. En Ingeniería del Software es difícil describir los experimentos con el detalle suficiente como para que otros puedan reproducirlos dentro de sus propios resultados de trabajo [16][11]. En el campo de las metaheurísticas existen entornos de trabajo que permiten manipular los experimentos y los resultados de estos [22], incluso definiendo lenguajes y entornos para describir con precisión experimentos centrados en optimización con metaheurísticas, (e.g. https://examplar.us.es) Pero, en general, cuando buscamos la utilización del método científico experimental para la generación de nuevo conocimiento sobre el proceso de desarrollo de software, estas aproximaciones deben ser recubiertas con un nivel superior de abstracción, donde se defina un marco unificado de los trabajos partiendo de la Ingeniería del Software, no sólo centrado en los algoritmos y sus soluciones.

En este trabajo definimos un proceso común que ayude a los investigadores a mantener un patrón estándar y que además, permita a los participantes de los ensayos de Ingeniería del Software guiada por búsqueda colaborar para la comparación de los algoritmos y la reproducibilidad de los experimentos, haciendo así verificables los nuevos avances propuestos en cada caso.

El resto del trabajo se organiza como sigue. En la sección 2 se enmarca el trabajo dentro de la experimentación colaborativa en disciplinas científicas. A continuación, la sección 3 describe el flujo de trabajo genérico para los ensayos de Ingeniería del Software guiada por busqueda. Una vez descrito, se instancia para el caso específico del problema de selección de los requisitos de la siguiente versión del software en la sección 4. Para finalizar, en la sección 5 se presentan las conclusiones y se esbozan posibles líneas de trabajo futuro.

2. Experimentación colaborativa

Unificar las características específicas de experimentación en Ingeniería de Software y metaheurísticas no es una tarea trivial, se requieren mecanismos e instrumentos que permitan la comunicación entre investigadores. Por una parte, es necesario mejorar los procedimientos para la investigación en metaheurísticas incrementando la trasparencia de las implementaciones y la compresión de cada una de las partes que las componen [28]. Y por otra, puesto que en ciencia e

ingeniería no tiene sentido hablar de experimentos aislados, el resultado de un solo experimento no puede ser representativo. En la mayoría de los casos la experimentación necesita poder ser validada por replicación o al menos debe existir la posibilidad de comparar, de forma efectiva, los resultados de diversos ensayos utilizando medidas o indicadores de calidad a nivel del algoritmo en sí y del problema al que se busca solución.

De cara a ser capaces de hacer útiles y efectivos los ensayos en el campo de la Ingeniería del Software guiada por búsqueda necesitamos unificar el proceso y así hacer que los resultados sean comparables. La responsabilidad no se debe centrar en un sólo investigador, sino más bien en la comunidad [10], por lo que este proceso unificado debe abordarse desde una perspectiva colaborativa. Lograr la comunicación entre los investigadores es una tarea compleja, que supone inicialmente sobrepasar barreras para la mera transferencia de conocimiento. Es necesario acordar cuáles, cómo y con qué se ejecutarán las distintas etapas de los ensayos experimentales que permitirán estudiar la validez de la aplicación de las técnicas de búsqueda a los problemas específicos de la Ingeniería del Software. Además, cada tipo de técnica presenta sus propias particularidades: no es lo mismo aplicar el recocido simulado que un algoritmo genético o un resolutor basado en programación lineal. Del mismo modo, no es igual tratar con el problema de selección de requisitos que con el de selección de los casos de prueba o de la arquitectura de software. Cada uno presenta dimensiones propias en la toma de decisiones y para definir la mejor solución. Es decir, además de los indicadores de calidad a nivel puramente de la técnica de búsqueda utilizada, se necesita utilizar indicadores de calidad específicos de cada tipo de problema [2]. La comparabilidad por tanto se definirá en torno a indicadores que representen la abstracción y encapsulación del ensayo realizado.

Una nueva aportación en esta disciplina se debe articular en torno a actividades genéricas, con una clara definición de cómo éstas se comunican entre sí. Las tareas individuales pueden desarrollarse por el mismo investigador que propone el trabajo o bien pueden ejecutarse, o haber sido ejecutadas, por otros investigadores. Por ejemplo, si queremos proponer una nueva técnica metaheuristica no sería necesario volver a realizar la implementación de otras técnicas contra las que realizar la comparativa, siempre y cuando el ensayo respete la estructura definida en las tareas genéricas y se pueda trabajar a nivel de indicadores. También será posible colaborar compartiendo las rutinas implementadas con tal de que exista y se cumplan los protocolos de interfaz.

3. Actividades genéricas de colaboración

De forma general, los ensayos experimentales en Ingeniería del Software guiada por búsqueda son costosos de organizar. Si extrapolamos las ventajas generales de la estandarización de metaheurísticas [20], tener una referencia estándar que pueda adaptarse o instanciarse en cada caso sería beneficioso en diversos aspectos: *menor dependencia de los investigadores* puesto que se facilitará la automatización, *mejorar la transparencia* al registrar los procesos y, por tanto,

saber dónde se ha de *tomar cada decisión* sobre el ensayo, *fomentar la reproducibilidad* al explicitar las condiciones experimentales, *facilitar las tareas de análisis de los resultados* definiendo los procesos y medidas estadísticas, así como los indicadores que abstraen los datos en bruto y ,finalmente, *permitir compartir y generar conocimiento*, puesto que tener un marco de referencia permitirá trabajar cooperativamente distribuyendo responsabilidades y sacar partido de la experiencia individual.

Este escenario colaborativo se tiene que abordar desde distintos puntos de vista o niveles. Para eso hemos representado con piscinas o carriles los diversos participantes en el proceso, (figura 1). Por una parte está el responsable del ensayo que representa al equipo de investigación que plantea la definición de una nueva técnica a un problema dado. Por otra está el marco colaborativo, que describe las tres grandes responsabilidades que aparecen cuando se realiza un ensayo. Estas tareas no tienen porqué ser realizadas por los investigadores responsables del ensayo, sino que pueden delegar en trabajos previos o paralelos desarrollados por otros equipos, siempre y cuando se acojan a esta marco de referencia.

Los aspectos básicos de un ensayo de Ingeniería del Software guiada por búsqueda se articulan en tres niveles de responsabilidades:

- *El nivel del conjunto de datos sobre el que realizar los ensayos.* Debido a las políticas de confidencialidad de la mayoría de las empresas de desarrollo de software comercial, disponer de datos sobre los que aplicar los ensayos no es nada fácil. Una alternativa es utilizar datos publicados en los trabajos de investigación [7,33]. Otra es extraer los datos de los repositorios públicos de proyectos de código abierto [32,12]. Pero no existen formatos unificados para la especificación de estos datos de acuerdo con el problema para el que van a ser utilizados. Además, en ocasiones se deben trasformar los datos para adaptarlos al problema, como el caso presentado en Sagrado et al, 2015 [25] donde se muestra como trasformar un problema con dependencias entre requisitos a un conjunto de datos sin ellas.
  Con lo cual, junto con la selección y formulación del problema, se tendrá que construir el conjunto de datos sobre los que ejecutar los experimentos. Se puede recuperar una formulación previa o definir una nueva propuesta lo que implica, por ejemplo, seleccionar entre estrategias monobjetivo o multiobjetivo. El escenario de prueba puede ser cualquiera de los ya unificados para ese problema o bien necesitar la trasformación o extracción de los datos para generar un nuevo escenario, que además generará un nuevo caso disponible para el resto de la comunidad.
- *El nivel de la técnica inteligente a aplicar.* Las metaheurísticas han sido las preferidas para afrontar problemas de Ingeniería del Software guiada por búsqueda, pero se han aplicado con éxito otras técnicas [14,3]. La estandarización y cooperación en la experimentación con metaheuríticas tiene la ventaja de mejorar la comunicación y reproducibilidad, la interoperatividad, el ensamblado automático de las metaheurísticas, la generación de conocimiento y la eficiencia [28].

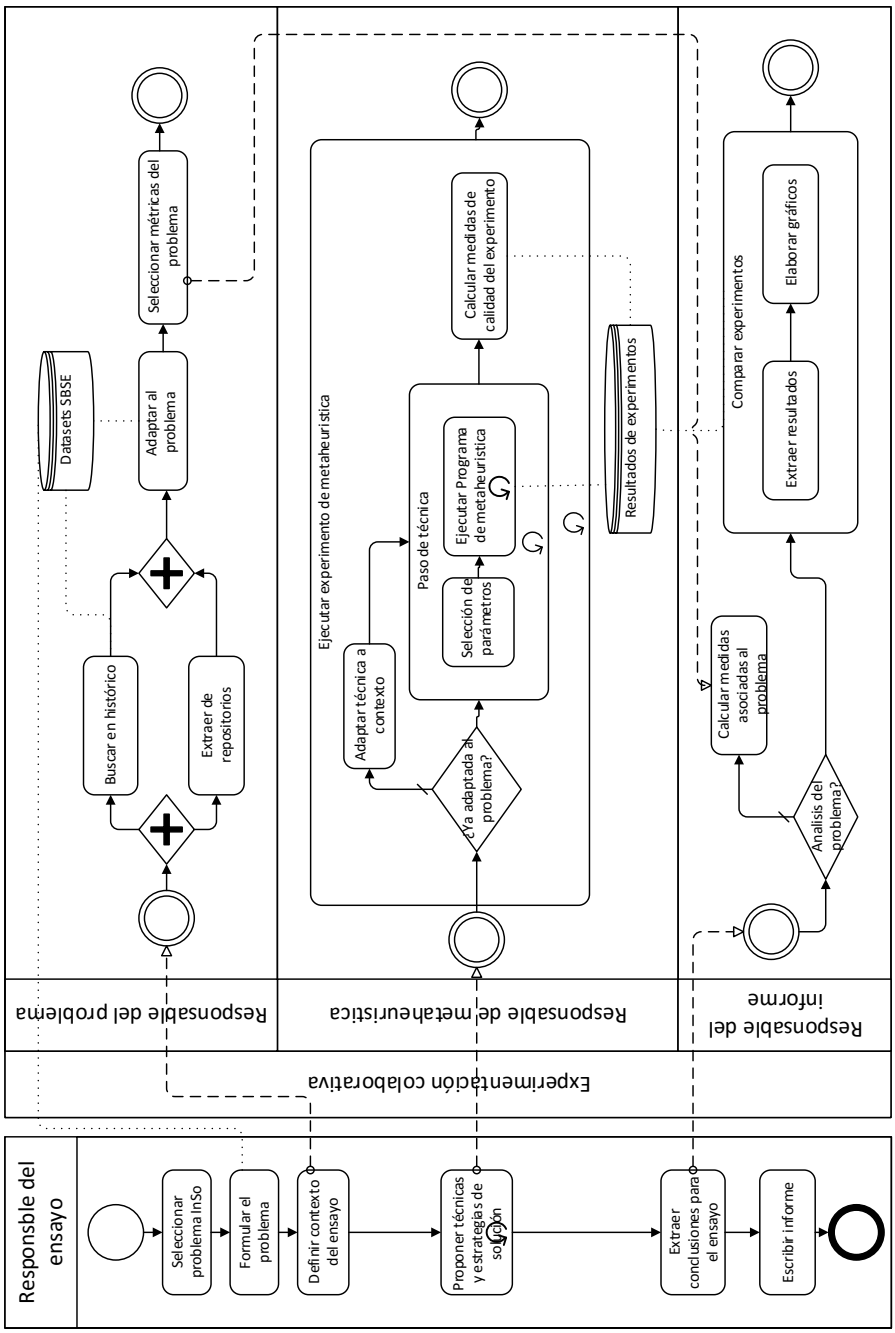


Figura 1: Flujo de trabajo para un ensayo en Ingeniería del software guiada por búsqueda

Cuando queremos aplicar una nueva metaheuristica y establecer su efectividad, necesitamos compararla bien contra un caso base, o bien contra los resultados obtenidos por otro conjunto de técnicas aplicadas en anteriores trabajos de investigación. Pero esta cuestión no está totalmente resuelta en la literatura. En primer lugar, algunas metaheurísticas tienen muchos parámetros que requieren un ajuste manual (ej. probabilidad de mutación o cruce, grado de voracidad), y en segundo lugar, porque son de naturaleza estocástica y necesitamos recurrir a profundos estudios estadísticos y de indicadores de calidad para poder hacer una comparación efectiva.
Si la metaheuristica no ha sido antes adaptada a ese problema, entonces habrá de serlo. Antes de poder compararla con los resultados de otras, se debe buscar cuál es la configuración que mejor resultado presenta. Esta selección de parámetros y la posterior comparativa se debe realizar en base a indicadores de calidad [35,21] y que son estrechamente dependientes de la estrategia de resolución (e.g. monobjetivo, multiobjetivo).

- *El nivel de generación de informes y extracción de conclusiones.* El análisis efectivo de los datos obtenidos durante el ensayo no es una tarea fácil. Debe abordarse, tanto desde el punto de vista de la metaheurística, como desde el punto de vista del problema y, en muchos casos, ampliar el análisis al contexto concreto del problema o conjunto de datos [3]. Además, en la mayoría de las ocasiones se deben abstraer los datos utilizando distintos tipos de gráficos y test estadísticos que permitan a los investigadores definir conclusiones que puedan compartirse en un formato unificado a toda la comunidad.

4. Caso del problema de la siguiente versión

En esta sección describimos como se instancia el conjunto de actividades genéricas mostradas en la figura 1 a un problema concreto de Ingeniería del Software (el problema de la siguiente versión) aplicando técnicas y algoritmos guiados por búsqueda.

Seleccionar el problema Nos centraremos en un problema ampliamente conocido en esta disciplina, el problema de la siguiente versión o NRP [6], para el que existen numerosos trabajos que ofrecen buenas soluciones utilizando distintas aproximaciones [23,1]. La limitación de recursos en los proyectos de desarrollo de software, obliga a dejar fuera de la siguiente versión de una aplicación ya construida algunas de las propuestas hechas por los clientes. Es decir, el equipo de desarrollo tiene que decidir cuál es el subconjunto de requisitos que tiene que contemplarse. Se busca combinar los métodos computacionales con la experiencia de los expertos humanos para obtener mejores conjuntos de requisitos de los que se producirían sólo mediante el juicio experto. Para ello este problema se formula como sigue.

Formular el problema Las selección de requisitos presenta variantes en su formulación, cada una destaca una parte del problema. Se definen requisitos y

clientes y se obliga a satisfacer por completo a los clientes seleccionando todas sus propuestas [6]; o se definen en la importancia de cada requisito individualmente [5], permitiendo la satisfacción parcial de los clientes. En cualquier caso, los requisitos presentan dependencias o interacciones entre ellos, imponiendo un orden de desarrollo determinado, pero la versión extensa que incorpora todos los tipos de dependencias entre requisitos no se formuló utilizando una representación en forma de grafo y matricial hasta el trabajo de del Sagrado et al. [25].

De esta manera para formular el problema, el responsable del ensayo podrá reutilizar una formulación ya propuesta o modificar o redefinir alguna de las existentes, generando un nuevo tipo de problema, por ejemplo, incluyendo el riesgo asociado a cada requisito como una dimensión más.

En nuestro caso, la formalización parte del conjunto de requsitos que aún no han sido desarrollados $\mathbf{R} = \{r_1, r_2, \ldots, r_n\}$ y que han sido propuestos por un conjunto de m clientes. Cada cliente i tiene una importancia para la empresa dentro del proyecto, $w_i \in \mathbb{R}$. Cada requisito $r_j \in \mathbf{R}$ tiene un coste c_j que recoge el esfuerzo de desarrollo. El valor del requisito r_j para el cliente i se representa con $v_{ij} \in \mathbb{R}$. La satisfacción, s_j, o valor añadido por la inclusión de r_j en la siguiente versión del software, se puede calcular como, $s_j = \sum_{i=1}^{m} w_i * v_{ij}$.

Si llamamos $X \subseteq R$ al conjunto de requisitos seleccionados, el coste y el valor de X vienen dados por las funciones:

$$esfuerzo(X) = \sum_{j,r_j \in X} c_j \quad \text{y} \quad satisfaccion(X) = \sum_{j,r_j \in X} s_j$$

respectivamente. En relación con la **estrategia**, consideraremos una versión multiobjetivo del problema que minimice el coste y maximice el valor del conjunto de requisitos seleccionados (i.e. buscaremos soluciones en un espacio de búsqueda de dos dimensiones).

Definir el contexto del ensayo Este proceso implica fijar el conjunto de datos sobre el que ejecutar las pruebas y, por lo tanto, entra dentro de las tareas del agente responsable del problema. En nuestra instancia se reutiliza el caso descrito por primera vez en [13] con el límite de esfuerzo fijado en 60. No obstante, puede que el conjunto de datos necesitase un pretratamiento, que ejecuta este agente, para trasformar el problema eliminando dependencias [25] o definir un nuevo caso extrayendo los datos de repositorios públicos de software para asignar errores como peticiones de nuevos requisitos [32].

Proponer técnicas y estrategias de solución Actualmente, se han propuesto numerosas técnica inteligentes para la solución del problema NRP con éxito. Bagnall et al. [6] mostró como aplicar ascenso de colinas, algoritmos voraces y recocido simulado teniendo en cuenta solo un tipo de dependencia. Otras soluciones alternativas se han basado fundamentalmente en algoritmos genéticos, [13] Del Sagrado et al. [24] adaptaron un sistema de colonia de hormigas para requisitos independientes y compararon los resultados con recocido simulado y un algoritmo genético. Estos algoritmos de colonia de hormigas también se han

utilizado contemplando sólo las dependencias de implicación [27]. Todas estas soluciones plantean una estrategia con un sólo objetivo, maximizar la satisfacción considerando el límite de esfuerzo como restricción. La consideración de todas las dependencias funcionales dentro del problema de búsqueda ha sido resuelta modelando las dependencias como un grafo dirigido [26,1].

Otros trabajos han aplicado una estrategia multiobjetivo con distintas técnicas metaheurísticas de naturaleza genética [34,9,8] y soluciones basadas en colonias de hormigas [25]. La visión multiobjetivo se ha usado para aplicar métodos exactos [30,14,3] o que reducen el espacio de búsqueda de las soluciones [32].

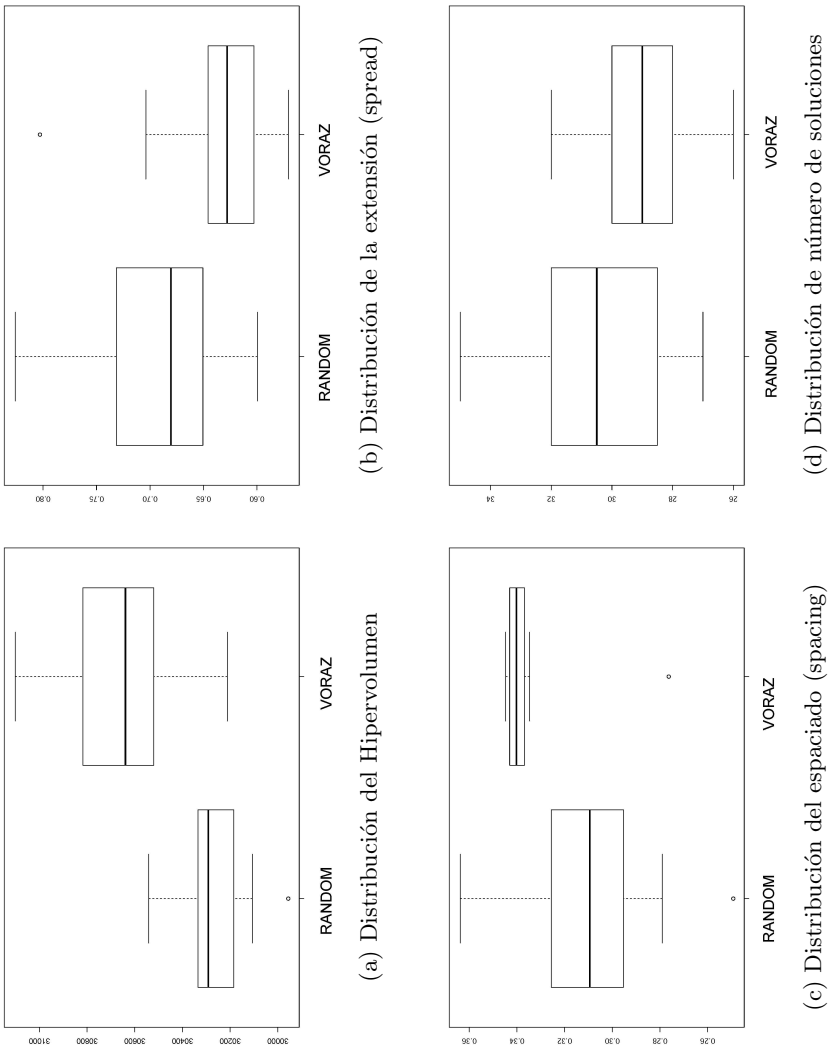
Para el ensayo que se está desarrollando en este caso vamos a estudiar la aplicabilidad de un algoritmo voraz [6] considerando todas las dependencias funcionales. Sus resultados se comparan con los obtenidos mediante una búsqueda aleatoria y con el frente de Pareto exacto, tras haber seleccionado un frente tipo de cada técnica aplicada.

Al tratarse de una nueva metaheurística, deberemos evaluar su rendimiento, en lo que se ha llamado *paso de técnica* en la figura 1. Aquí entraría en juego el ajuste de parámetros y la naturaleza estocástica de la técnica. Recurriremos al análisis de indicadores de calidad [35,21] de los resultados obtenidos en una serie de ejecuciones independientes de la técnica. Cada ejecución de la técnica nos proporcionará un frente de Pareto sobre el que calcularemos los indicadores de calidad: hipervolumen, extensión (*spread*), espaciado (*spacing*) y el número de soluciones. La distribución de los datos recopilados sobre estos indicadores servirá para darnos una idea del comportamiento individual de la técnica. Podemos ampliar el ámbito de comparación a otras técnicas comparando la distribución de indicadores calculados sobre el mismo número de ejecuciones independientes. Estos indicadores serán los que proporcionen el nivel de abstracción suficiente para permitir la comparabilidad de las técnicas.

En nuestro caso de estudio, vamos a comparar las soluciones obtenidas utilizando una técnica voraz, basada en la productividad $prod_j = s_j/c_j$ de cada requisito r_j, con las calculadas al realizar una búsqueda aleatoria. La tabla 1 recoge el resumen estadístico relativo a los indicadores de rendimiento utilizados para el caso de una técnica bi-objetivo, considerando 20 ejecuciones independientes.

Tabla 1: Indicadores para 20 ejecuciones (20 frentes)

	Algoritmo aleatorio				Algoritmo voraz			
	Hipervol	Spread	Spacing	Sols.	Hipervol	Spread	Spacing	Sols.
Min.	29956	0.60	0.25	27.00	30212	0.57	0.28	26.00
1st Qu.	30186	0.65	0.30	28.75	30529	0.61	0.34	28.00
Median	30292	0.68	0.31	30.50	30639	0.62	0.34	29.00
Mean	30274	0.69	0.31	30.65	30644	0.63	0.34	28.95
3rd Qu.	30333	0.72	0.33	32.00	30799	0.64	0.34	30.00
Max.	30541	0.83	0.36	35.00	31101	0.80	0.34	32.00



(a) Distribución del Hipervolumen (b) Distribución de la extensión (spread)

(c) Distribución del espaciado (spacing) (d) Distribución de número de soluciones

Figura 2: Comparativa gráfica de las distribuciones de los indicadores.

El comportamiento tipo de las técnicas se determina en base al valor de referencia que proporciona la mediana. Así, podemos guiarnos por este valor en los indicadores para seleccionar, de entre las ejecuciones realizadas, un frente de Pareto tipo para cada una de las técnicas aplicadas. La tabla 2 recoge los indicadores del frente de Pareto exacto y de los frentes tipo obtenidos tanto aleatoriamente, como aplicando la técnica voraz y cuya comparativa gráfica vemos en la figura 2.

Tabla 2: Indicadores asociados al frente exacto y a los frentes tipo

	Hipervol	Spread	Spacing	Sols.
Exacto	31165	0.587	0.341	31
Aleatorio	30339	0.692	0.315	31
Voraz	30637	0.630	0.345	29

La figura 3 muestra el frente exacto y los frentes tipo elegidos. Nos resta comprobar la similitud entre ellos. Esta tarea se aborda con la ayuda de test-no paramétricos sobre la distribución de la productividad de las soluciones en los frentes de Pareto. La hipótesis nula H_0 es que los datos de productividad de dos frentes de Pareto corresponden a poblaciones idénticas. Para comprobar la hipótesis aplicamos el test de Mann-Whitney-Wilcoxon para comparar las muestras independientes. En todos los casos, como el p-valor obtenido es mayor que el nivel de significancia de 0,05, aceptamos la hipótesis nula. Los datos de productividad asociados a los frentes de Pareto obtenidos con las distintas técnicas corresponden a poblaciones idénticas (ver tabla 3).

Tabla 3: Resultados del test del Wilcoxon

	Exacto-Aleatorio	Exacto-Voraz	Aleatorio-Voraz
p-value	0.2425	0.7899	0.1492

Después de obtener toda esta información sobre la técnica, si un investigador idease una nueva técnica, no sería necesario que volviese a realizar la implementación de las otras técnicas. En vez de eso, se centraría en ejecutar un número determinado de veces su técnica sobre el problema integrado en el flujo colaborativo, en obtener los indicadores y resultados, para, finalmente, compararlos con los disponibles. El flujo colaborativo agiliza la comparativa y descarga de trabajo al investigador.

Extracción de conclusiones y generación de informes Además de manejar y dar formato a los resultados generados directamente de las ejecuciones del software que implementa las técnicas, similares a las mostradas en la figuras 2 y 3 y las tablas, se deben acercar los resultados de los algoritmos de búsqueda al problema de Ingeniería del Software al que dan solución. Para ello, es necesario reinterpretar los resultados propuestos utilizando los términos y medias del dominio del problema, y así permitir que el ingeniero del software decida de entre las soluciones propuestas cuál es la más apropiada.

Esta trasferencia de los resultados a los proyectos software en desarrollo supone también la explotación de las técnicas de búsqueda en proyectos reales y se arbitra en base a indicadores de calidad dependientes de la formulación del problema. Para el caso que nos ocupa la generación de informes necesita cons-

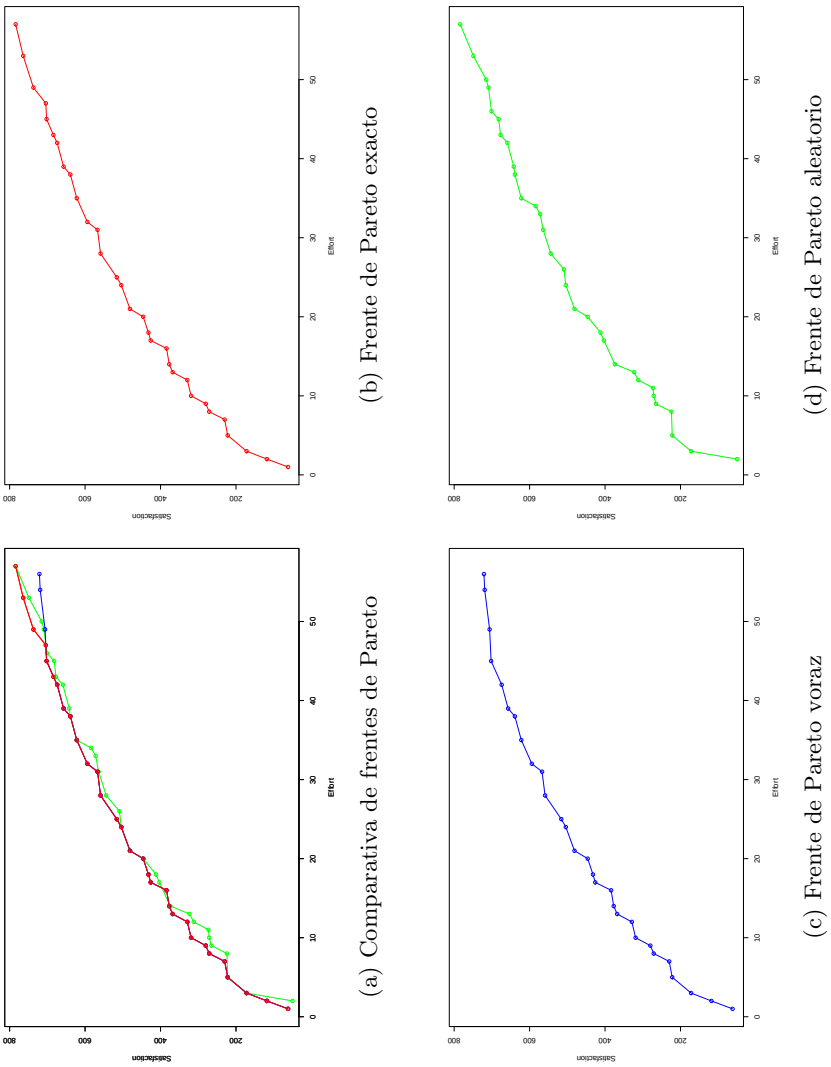


Figura 3: Frentes de Pareto.

truir indicadores de productividad, contribución y cobertura tanto de requisitos, como de clientes y soluciones [2].

Con todos estos datos disponibles el equipo de investigación responsable del ensayo está en disposición de generar un trabajo de carácter empírico para su difusión a la comunidad SBSE o a la comunidad científica en general.

5. Conclusiones

Hemos definido un flujo de trabajo unificado para la experimentación colaborativa basado en resultados e indicadores de calidad y rendimiento. Con ello proporcionamos un proceso común que ayuda a los investigadores a mantener un patrón estándar y que permite a los participantes de los ensayos de Ingeniería del Software guiada por búsqueda, colaborar en la comparación de los algoritmos y la reproducibilidad de los experimentos.

Para mostrar el uso del flujo de trabajo, se ha instanciado el conjunto de actividades genéricas al problema de la siguiente versión aplicando técnicas y algoritmos guiados por búsqueda simples.

Desde el punto de vista del equipo de investigación, una clara ventaja es la agilización de comparativas y la consiguiente descarga de trabajo. Así, un investigador que plantea una nueva técnica, no tiene que implementar otras técnicas como referencia y puede concentrarse en la configuración y ejecución de su propuesta sobre el problema generando solo indicadores y resultados que podrá comparar con el resto de resultados disponibles. No obstante, una necesidad inherente a este enfoque es la existencia de repositorios en los que los grupos de investigación compartan los resultados de experimentación, siguiendo los criterios comunes fijados en el flujo de trabajo unificado.

Como trabajo futuro planteamos la extensión el flujo de trabajo para cubrir también las tareas de explotación de los algoritmos de búsqueda en los proyectos software y la posiblidad de incorporarlas dentro de herramientas software que den soporte a la gestión del proyecto de desarrollo.

Agradecimientos. Este trabajo ha sido financiado parcialmente por la Universidad de Almería, la Red de Excelencia SEBASENet (TIN2015-71841-REDT) y por el Ministerio de Economía y Competitividad a través del proyecto TIN2013-46638-C3-1-P.

Referencias

1. del Águila, I., del Sagrado, Orellana, F.J.: Metaheurísticas como soporte a la selección de requisitos del software. In: Actas XVII Jornadas de Ingeniería del Software y Bases de Datos. SISTEDES, Almería, España (2012)
2. del Águila, I., del Sagrado, J., Bosch, A.: Análisis de las soluciones guiadas por búsqueda para el problema de selección de requisitos. In: XX Jornadas de Ingeniería del Software y Bases de Datos. SISTEDES (2015)
3. del Águila, I., del Sagrado, J., Chicano, F., Alba, E.: Resolviendo un problema multi-objetivo de selección de requisitos mediante resolutores del problema sat. In: XX Jornadas de Ingeniería del Software y Bases de Datos. SISTEDES (2015)
4. del Águila, I.M., del Sagrado, J.: Bayesian networks for enhancement of requirements engineering: a literature review. Requirements Engineering (may 2015)
5. van den Akker, J.M., Brinkkemper, S., Diepen, G., Versendaal, J.: Determination of the next release of a software product: an approach using integer linear programming. In: Proceeding of the 11th International Workshop on Requirements Engineering: Foundation for Software Quality REFSQ'05. pp. 247–262 (2005)

6. Bagnall, A.J., Rayward-Smith, V.J., Whittley, I.: The next release problem. Information & Software Technology 43(14), 883–890 (2001)
7. Boetticher, G.and Menzies, T., Ostrand, T.: Promise repository of empirical software engineering data. west virginia university, department of computer science. West Virginia University, Department of Computer Science (2007)
8. Durillo, J.J., Zhang, Y., Alba, E., Harman, M., Nebro, A.J.: A study of the bi-objective next release problem. Empirical Software Engineering 16(1), 29–60 (2011)
9. Finkelstein, A., Harman, M., Mansouri, S.A., Ren, J., Zhang, Y.: A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. Requirement Engineering 14(4), 231–245 (2009)
10. Fomel, S.: Reproducible research as a community effort: Lessons from the Madagascar project. Computing in Science and Engineering 17(1), 20–26 (jan 2015)
11. Gómez, O.S., Juristo, N., Vegas, S.: Understanding replication of experiments in software engineering: A classification. Information and Software Technology 56(8), 1033–1048 (2014)
12. González-Barahona, J.M., Robles, G.: On the reproducibility of empirical software engineering studies based on data retrieved from development repositories. Empirical Software Engineering 17(1-2), 75–89 (2012)
13. Greer, D., Ruhe, G.: Software release planning: an evolutionary and iterative approach. Information and Software Technology 46(4), 243–253 (Mar 2004)
14. Harman, M., Krinke, J., Medina-Bulo, I., Palomo-Lozano, F., Ren, J., Yoo, S.: Exact scalable sensitivity analysis for the next release problem. ACM Transactions on Software Engineering and Methodology (TOSEM) 23(2), 19 (2014)
15. Harman, M., Mansouri, S.A., Zhang, Y.: Search-based software engineering: Trends, techniques and applications. ACM Comput. Surv. 45(1), 11:1–11:61 (Dec 2012)
16. Juristo, N., Vegas, S.: The role of non-exact replications in software engineering experiments. Empirical Software Engineering 16(3), 295–324 (jun 2011)
17. Kastro, Y., Bener, A.B.: A defect prediction method for software versioning. Software Quality Journal 16(4), 543–562 (2008)
18. Menzies, T., Shepperd, M.: Special issue on repeatable results in software engineering prediction. Empirical Software Engineering 17(1-2), 1–17 (2012)
19. Misirli, A.T., Bener, A.B.: Bayesian networks for evidence-based decision-making in software engineering. IEEE Transactions on Software Engineering 40(6), 533–554 (2014)
20. Neumann, G., Swan, J., Harman, M., Clark, J.a.: The executable experimental template pattern for the systematic comparison of metaheuristics. Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion - GECCO Comp '14 pp. 1427–1430 (2014)
21. Okabe, T., Jin, Y., Sendhoff, B.: A critical survey of performance indices for multi-objective optimisation. In: Evolutionary Computation, 2003. CEC'03. The 2003 Congress on. vol. 2, pp. 878–885. IEEE (2003)
22. Parejo, J.A., Ruiz-Cortés, A., Lozano, S., Fernandez, P.: Metaheuristic optimization frameworks: A survey and benchmarking. Soft Computing 16(3), 527–561 (2012)
23. Pitangueira, A.M., Maciel, R.S.P., de Oliveira Barros, M.: Software requirements selection and prioritization using sbse approaches: A systematic review and mapping of the literature. Journal of Systems and Software 103, 267–280 (May 2015)
24. del Sagrado, J., del Águila, I., Orellana, F.: Ant colony optimization for the next release problem: A comparative study. In: Proceeding of Second International Sym-

posium on Search Based Software Engineering (SSBSE 2010), Benevento, Italy. pp. 67 –76 (sept 2010)
25. del Sagrado, J., del Águila, I.M., Orellana, F.J.: Multi-objective ant colony optimization for requirements selection. Empirical Software Engineering 20(3), 577–610 (2015)
26. del Sagrado, J., del Águila, I.M., Orellana, F.J.: Requirements interaction in the next release problem. In: Proceedings of 13th Annual Genetic and Evolutionary Computation Conference (GECCO 2011), Dublin, Ireland. pp. 241–242 (2011)
27. de Souza, J.T., Maia, C.L.B., do Nascimento Ferreira, T., do Carmo, R.A.F., Brasil, M.M.A.: An ant colony optimization approach to the software release planning with dependent requirements. In: Search Based Software Engineering, pp. 142–157. Springer (2011)
28. Swan, J., Adriaensen, S., Bishr, M., Burke, E.K., Clark, J.A., Causmaecker, P.D., Durillo, J., Hammond, K., Hart, E., Johnson, C.G., Kocsis, Z.A., Kovitz, B., Krawiec, K., Martin, S., Merelo, J.J., Minku, L.L., Ozcan, E., Pappa, G.L., Pesch, E., Garc, P., Schaerf, A., Sim, K., Smith, J.E., St, T., Voß, S., Wagner, S., Yao, X.: A Research Agenda for Metaheuristic Standardization. Mic pp. 1–3 (2015)
29. Tosun, A., Bener, A.B., Akbarinasaji, S.: A systematic literature review on the applications of Bayesian networks to predict software quality. Software Quality Journal pp. 1–33 (2015)
30. Veerapen, N., Ochoa, G., Harman, M., Burke, E.K.: An integer linear programming approach to the single and bi-objective next release problem. Information and Software Technology 65(0), 1 – 13 (2015)
31. Wen, J., Li, S., Lin, Z., Hu, Y., Huang, C.: Systematic literature review of machine learning based software development effort estimation models. Information and Software Technology 54(1), 41–59 (2012)
32. Xuan, J., Jiang, H., Ren, Z., Luo, Z.: Solving the large scale next release problem with a backbone-based multilevel algorithm. Software Engineering, IEEE Transactions on 38(5), 1195–1212 (2012)
33. Zhang, Y., M., H., Mansouri, A.: The sbse repository: A repository and analysis of authors and research articles on search based software engineering. *http* : $//crestweb.cs.ucl.ac.uk/resources/sbse_repository/$. (2012)
34. Zhang, Y., Harman, M., Mansouri, S.A.: The multi-objective next release problem. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2007), London, England, UK. pp. 1129–1137 (2007)
35. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Da Fonseca, V.G.: Performance assessment of multiobjective optimizers: an analysis and review. Evolutionary Computation, IEEE Transactions on 7(2), 117–132 (2003)