Formulación Robusta para el Problema de Selección de Requisitos

Francisco Chicano and Miguel Ángel Domínguez-Rios

Universidad de Málaga, Spain *
chicano@lcc.uma.es, miguel.angel.dominguez.rios@uma.es

Resumen El problema de selección de requisitos consiste en elegir un subconjunto de requisitos que serán desarrollados en la siguiente versión del producto software. Esta elección se debe hacer de tal forma que se maximice la satisfacción de los clientes y se minimice el coste de implementación, cumpliendo, además, con una serie de dependencias entre los requisitos. Tanto el coste de implementación de los requisitos como la satisfacción de los clientes están sujetos a incertidumbre y se pueden modelar mediante variables aleatorias. Como resultado, el coste total y el valor de una solución (subconjunto de requisitos) también son variables aleatorias. Un decisor normalmente prefiere soluciones robustas, es decir, con baja incertidumbre en sus valores objetivos. Esta preferencia se puede modelar minimizando la varianza de las variables aleatorias, a la vez que se optimizan sus valores promedios. En este trabajo presentamos una formulación robusta del problema de selección de requisitos con cuatro objetivos: los promedios del coste y satisfacción, y sus varianzas. Para resolver el modelo empleamos un resolutor de programación entera que se aplica a una suma ponderada de los objetivos, obteniendo soluciones soportadas del frente de Pareto.

Keywords: Selección de requisitos, optimización robusta, optimización multi-objetivo, programación entera

1. Introducción

Partimos de un conjunto de requisitos con dependencias $R = \{r_1, r_2, \ldots, r_n\}$ que aún no han sido desarrollados y que han sido propuestos por un conjunto de m clientes. Cada cliente i tiene un peso asociado $w_i \in \mathbb{R}$ que mide su importancia dentro del proyecto. Cada requisito $r_j \in R$ tiene un coste c_j para la empresa, es decir cada r_j consumirá c_j recursos si se desarrolla. El problema de selección de requisitos (*Next Release Problem*, NRP) bi-objetivo consiste en encontrar un subconjunto de R que maximice la satisfacción de los clientes y minimice el coste. Utilizaremos programación entera para resolver el problema. Para ello, utilizaremos una variable binaria por cada requisito: r_1, r_2, etc.

* Investigación parcialmente financiada por el Ministerio de Economía y Competitividad y FEDER (TIN2017-88213-R), y la Universidad de Málaga (proyecto Exhauro).

En este trabajo seguimos el modelo de satisfacción de Xuan et al. [1]. Un cliente *está satisfecho* sólo cuando todos sus requisitos se implementan y, en este caso, sumamos su peso w_i al *grado de satisfacción* asociado al conjunto de requisitos. Los requisitos presentan dependencias o interacciones entre ellos, imponiendo un orden de desarrollo determinado, lo que limita las alternativas para ser elegidos. Las interacciones entre los requisitos representan restricciones al problema y se agrupan en dos tipos: *dependencias funcionales* o *estructurales* y *dependencias por recursos consumidos* [2]. En este trabajo nos centramos en las dependencias funcionales, que pueden definirse como:

- *Implicación o precedencia.* $r_i \Rightarrow r_j$. El requisito r_i no se puede seleccionar si el requisito r_j no ha sido ya implementado. Esto se expresa con la restricción $r_i \leq r_j$.
- *Combinación o acoplamiento.* $r_i \odot r_j$. Los requisitos r_i y r_j deben ser incluidos de forma conjunta en el software. Esto se expresa con la restricción $r_i = r_j$.
- *Exclusión.* $r_i \oplus r_j$. El requisito r_i no puede incluirse junto al requisito r_j. Esto se expresa con la restricción $r_i + r_j \leq 1$.

El coste y la satisfacción vienen dados por las ecuaciones

$$coste(r_1, \ldots, r_n) = \sum_{j=1}^{n} c_j r_j \quad \text{y} \quad sat(t_1, \ldots, t_m) = \sum_{i=1}^{m} w_i t_i, \tag{1}$$

donde las variables t_i indican si un cliente está o no satisfecho y las variables r_j indican si el requisito está o no seleccionado. La exigencia de que un cliente esté satisfecho sólo cuando todos sus requisitos están implementados implica la incorporación de restricciones de la forma $t_i \leq r_j$ si y sólo si el cliente i quiere que se implemente el requisito r_j. El problema es bi-objetivo [2] y, por tanto no existe una solución única, sino un conjunto de soluciones Pareto óptimas, también llamadas *no dominadas* o *eficientes* [3].

2. Formulación robusta del problema

En la literatura se ha usado simulación de Monte Carlo para tratar la incertidumbre en este problema [4]. La principal desventaja de la simulación es que solo es posible obtener frentes de Pareto aproximados. Otro enfoque usado por Li *et al.* [5] es un marco en el que las instancias son generadas usando una distribución de probabilidad y resueltas mediante un algoritmo exacto, NSGDP, que es un algoritmo de programación dinámica *ad hoc* para este problema basado en el algoritmo de Nemhauser-Ullman para el problema de la mochila. Este enfoque también modela la incertidumbre de forma aproximada (ya que muestrea una distribución de probabilidad) y está limitado en el tipo de restricciones que puede incorporar (no admite las de precedencia de requisitos). También tiene limitaciones computacionales, puesto que la cantidad de subproblemas a resolver puede aumentar exponencialmente con el número de requisitos.

En este trabajo mantendremos la exactitud del modelo para obtener soluciones no dominadas del frente de Pareto. El coste de desarrollo de un requisito y el grado de satisfacción de los clientes son cantidades inciertas. De hecho, es bien conocido en ingeniería del software, que es muy difícil estimar el tiempo requerido para implementar cualquier aspecto funcional del software y se tiende a subestimar el esfuerzo requerido. Por tanto, más que usar un valor fijo y cierto para estas cantidades, es más apropiado modelarlos con variables aleatorias. De esta forma, el coste y la satisfacción totales de una selección particular de requisitos sería, de nuevo, una variable aleatoria.

El interés en optimización robusta es encontrar una solución cuyos valores objetivos no varíen demasiado cuando los valores inciertos se alejan del valor esperado. Esto se modela minimizando la varianza de los objetivos, teniendo en cuenta que cada objetivo es ahora una variable aleatoria. En el caso de NRP los cuatro objetivos a minimizar son:

$$E\left[coste(r_1,\ldots,r_n)\right] = \sum_{j=1}^{n} E[c_j]r_j, \tag{2}$$

$$Var\left[coste(r_1,\ldots,r_n)\right] = \sum_{i=1}^{n} Var[c_i]r_i + 2\sum_{1\leq i<j\leq n} r_i r_j Cov[c_i,c_j], \tag{3}$$

$$-E\left[sat(t_1,\ldots,t_m)\right] = -\sum_{i=1}^{m} E[w_i]t_i, \tag{4}$$

$$Var\left[sat(t_1,\ldots,t_m)\right] = \sum_{i=1}^{m} Var[w_i]t_i + 2\sum_{1\leq i<j\leq m} t_i t_j Cov[w_i,w_j], \tag{5}$$

donde hemos supuesto que los valores c_j y w_i son variables aleatorias y los cuadrados r_i^2 y t_i^2 se han sustituido por r_i y t_i por ser variables binarias.

El modelo anterior se puede linealizar usando $\binom{n}{2}+\binom{m}{2}$ variables adicionales. Pero, como la matriz de covarianzas es semidefinida positiva, es posible resolver el problema cuadrático directamente y obtener una solución más rápida. Si las variables c_j son independientes entre ellas, las covarianzas de variables diferentes es cero y el segundo sumando de (3) desaparece. Lo mismo ocurre en (5) cuando las variables w_i son independientes. Este es el caso que trataremos en los experimentos preliminares, aunque en la práctica la suposición no sea cierta debido al presupuesto limitado de ejecución de los proyectos de desarrollo software.

3. Resultados Preliminares

Hemos utilizado el modelo anterior junto con el resolutor CPLEX para calcular soluciones eficientes soportadas del frente de Pareto de las instancias de Xuan *et al.* [1]. Hemos asignado una varianza en el rango 100 a 300 a los costes de los requisitos y la satisfacción de los clientes, y hemos supuesto que son independientes. Para obtener las soluciones soportadas hemos optimizado una

suma ponderada de los cuatro objetivos del problema, donde los pesos se escogen de manera aleatoria, y hemos resuelto el problema de programación entera resultante. En la Figura 1 mostramos las soluciones soportadas de una parte del frente obtenidas para la formulación robusta en el caso de la instancia `nrp1` de [1]. Se señalan en otro color las soluciones que pertenecen también al frente del problema original bi-objetivo (sin robustez). Los puntos se encuentran en el valor promedio de coste y satisfacción (multiplicada por -1 para que se deba minimizar) y los rectángulos se extienden vertical y horizontalmente una desviación estándar. Se puede apreciar como las soluciones robustas (hacia el interior) poseen un valor promedio de coste y satisfacción más alto.

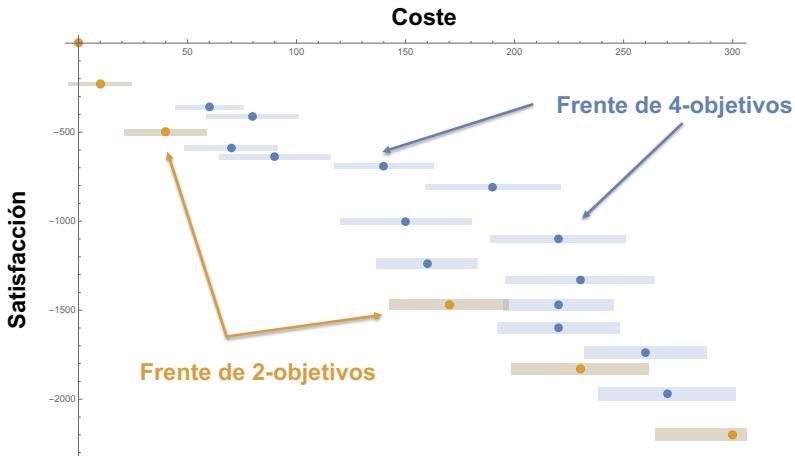

Figura 1. Soluciones eficientes soportadas obtenidas para la instancia `nrp1` de Xuan *et al.* [1]. Se presenta el valor opuesto de la satisfacción para que los dos objetivos sean de minimización. Se destacan en naranja las soluciones que pertenecen al frente del problema no robusto.

Referencias

1. 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)
2. 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)
3. Ehrgott, M.: Multicriteria optimization. Springer (2005)
4. Li, L., Harman, M., Letier, E., Zhang, Y.: Robust next release problem: Handling uncertainty during optimization. In: Proceedings of GECCO. pp. 1247–1254 (2014)
5. Li, L., Harman, M., Wu, F., Zhang, Y.: The value of exact analysis in requirements selection. IEEE Transactions on Software Engineering 43(6), 580–596 (2017)