Una herramienta para aplicar técnicas de Montecarlo al análisis de modelos de características *

José Miguel Horcas[1 [0000−0002−7771−0575]],
A. Germán Márquez[1 [0000−0002−7282−3386]],
José A. Galindo[1 [0000−0001−9293−9784]], and
David Benavides[1 [0000−0002−8449−3273]]

Universidad de Sevilla, España
jhorcas@us.es,antmartru@alum.us.es,jagalindo@us.es,benavides@us.es

Resumen La mayoría de los sistemas configurables describen un amplio espacio de soluciones que hacen intratable su exploración exhaustiva. En la literatura encontramos técnicas de análisis como la resolución SAT o la programación de restricciones. Sin embargo, ninguna de ellas ha explorado los métodos basados en la simulación de la toma de decisiones cuando configuramos un sistema. Nosotros proponemos usar el método de Montecarlo el cual simula la búsqueda en espacios de soluciones colosales de manera aleatoria. Este trabajo presenta la implementación de un marco conceptual que aborda varios de esos análisis utilizando métodos de Montecarlo, los cuales, han demostrado tener éxito en otros dominios con espacios de búsqueda colosales como la teoría de juegos. Concretamente presentamos una implementación en Python del marco de trabajo que muestra la viabilidad de nuestra propuesta con la que prevemos que se pueden abordar diferentes problemas y que nuestro marco pueda utilizarse para avanzar el estado del arte.

Keywords: Análisis de modelos de características · Montecarlo.

1. Introducción

Los sistemas de alta variabilidad y las líneas de producto software son sistemas que deben gestionar la variabilidad para su correcto funcionamiento. Algunos ejemplos de dichos sistemas son el kernel de Linux [11] o Android [5]. En este tipo de sistemas, el análisis de modelos de características (AAFM) se ha convertido en una herramienta indispensable para lidiar con el enorme tamaño del espacio de configuraciones existente.

* Agradecemos a José A. Troyano por inspirarnos en el uso de las técnicas de Montecarlo para el análisis de líneas de producto software. Este trabajo ha sido financiado por el programa FEDER de la Unión Europea, el proyecto OPHELIA (RTI2018-101204-B-C22) del ministerio, el proyecto COPERNICA de la Junta de Andalucía, y por el gobierno de España mediante el programa Juan de la Cierva—Formación 2019.

Concretamente encontramos aplicaciones del AAFM [4] en diversos sistemas que van desde la ingeniería inversa [10] al testing [5]. Muchos de estos análisis tienen que asumir cierta incertidumbre ya que es imposible realizar una exploración exhaustiva de todo el espacio de configuración. Por ejemplo, en algunos casos, es necesario realizar cálculos complejos para llevar a cabo acciones sencillas como decidir cuándo incluir o excluir una característica en una configuración, lo cual repercute en la conveniencia y el análisis de otras selecciones.

Al enfrentarse a grandes espacios de búsqueda, las técnicas existentes presentan varios inconvenientes. Los AAFM se han basado tradicionalmente en lógica proposicional o en la resolución SAT [4]. Sin embargo, existen problemas de escalabilidad para modelos que representan espacios colosales de configuración ya sea usando SAT [13] o usando análisis estadístico [6], debido al requisito de usar técnicas de compilación complejas, como por ejemplo la construcción de diagramas de decisión binarios (BDDs).

En este trabajo, presentamos una implementación de un marco conceptual basado en los llamados métodos de *Montecarlo* [9], que utilizan la aleatoriedad para problemas deterministas que son difíciles o imposibles de resolver con los enfoques tradicionales [9]. Pueden utilizarse con poco o ningún conocimiento del dominio, y han tenido éxito en problemas difíciles donde otras técnicas han fracasado [9]. En particular, adoptamos el método *Monte Carlo Tree Search* (MCTS) [1]. MCTS se ha aplicado con éxito a varios dominios [1], destacando en la teoría de juegos [12] donde se ha demostrado que puede escalar a grandes espacios de búsqueda como los que suelen caracterizar a las líneas de producto software (SPL). Este artículo presenta una herramienta que da soporte a la propuesta enviada a SPLC'21 [7].

2. Funcionamiento de la herramienta

La herramienta nos permite aplicar el método de Montecarlo para las actividades de SPL presentadas a continuación:

Localización de configuraciones erróneas. Un problema común en las pruebas y el mantenimiento de software es identificar las configuraciones que conducen a un determinado defecto o algún otro comportamiento no deseado. Nuestra herramienta, utilizando un enfoque por etapas, nos permite inferir qué característica(s) está(n) causando que la configuración falle.

Búsqueda de configuraciones con el menor número de características. Es frecuente cuando desarrollamos herramientas con sistemas de *plugins* el proporcionar un producto mínimo que contiene el menor número de dependencias posibles. Por lo tanto, estamos interesados en encontrar una configuración válida con el mínimo número de características. Nuestra herramienta ofrece soporte para detectar este tipo de configuraciones y poder proveerla como instalación por defecto de nuestra SPL.

Completitud de configuraciones parciales. El problema anterior de encontrar configuraciones mínimas válidas puede verse como una especialización

del problema de completar configuraciones parciales. Este problema consiste en encontrar el conjunto de características no seleccionadas necesarias para obtener una configuración válida completa. Mientras que en una configuración completa se decide que cada característica esté presente o ausente en la configuración resultante, en las configuraciones parciales, algunas características están indecisas.

Ingeniería inversa de modelos de características Un problema conocido en SPL es sintetizar un modelo de características a partir de un conjunto de configuraciones de forma automática. Dado un conjunto de combinaciones de características presentes en un SPL (es decir, un conjunto de configuraciones), el objetivo es extraer un modelo de características que represente todas las configuraciones. Formalmente, sea C_i el conjunto de configuraciones dadas como entrada. F_i es el conjunto de características presentes en C_i. El problema es construir un modelo de características m con características en F_i para que $C_i \subseteq C_m$ donde C_m es el conjunto de configuraciones válidas del modelo de características m.

3. Implementación y resultados preliminares

Proporcionamos una implementación del marco conceptual MCTS[1] construido sobre el marco de trabajo de Python para AAFMs propuesto en [3]. Se proporcionan los siguientes algoritmos de Montecarlo:

- **Montecarlo básico.** El método clásico con simulaciones aleatorias.
- **Algoritmo UCT.** Una implementación de MCTS que construye un árbol de búsqueda con los resultados de las simulaciones, y usa la estrategia de selección *upper confidence bound for trees* (UCT) [8]. Esta estrategia favorece las configuraciones más prometedoras, pero permite al mismo tiempo explorar aquellas configuraciones que aún no han sido suficientemente exploradas.
- **Greedy MCTS.** Una estrategia *best-first* que favorece la explotación frente a la exploración.

Cada algoritmo de Montecarlo puede configurarse con una condición de parada, como por ejemplo una restricción de tiempo, memoria o número de iteraciones, además de con un criterio de selección para la mejor decisión. Por ejemplo, para seleccionar la característica con la mayor recompensa, la característica más visitada, la característica con el mayor número de visitas y la mayor recompensa, o la característica que maximiza un límite inferior de confianza [2].

4. Conclusiones

Con esta contribución, prevemos que se puedan abordar diferentes problemas y análisis utilizando los métodos de Montecarlo, pasando a formar parte del conjunto de herramientas del investigador de SPL cuando analice modelos de características y sus configuraciones. Este nuevo enfoque puede ser de gran valor para hacer avanzar el estado del arte de los AAFM un paso adelante.

[1] `https://github.com/diverso-lab/fm_montecarlo`

Referencias

1. Browne, C.B., Powley, E., Whitehouse, D., Lucas, S.M., Cowling, P.I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S.: A survey of monte carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in Games **4**(1), 1–43 (2012).
2. Chaslot, G., Winands, M., Herik, H., Uiterwijk, J., Bouzy, B.: Progressive strategies for monte-carlo tree search. New Mathematics and Natural Computation **04**, 343–357 (2008).
3. Galindo, J.A., Benavides, D.: A python framework for the automated analysis of feature models: A first step to integrate community efforts. In: 24th ACM International Systems and Software Product Line Conference (SPLC). vol. B, pp. 52–55. ACM (2020). , https://doi.org/10.1145/3382026.3425773
4. Galindo, J.A., Benavides, D., Trinidad, P., Gutiérrez-Fernández, A.M., Ruiz-Cortés, A.: Automated analysis of feature models: Quo vadis? Computing **101**(5), 387–433 (2019). , https://doi.org/10.1007/s00607-018-0646-1
5. Galindo, J.A., Turner, H.A., Benavides, D., White, J.: Testing variability-intensive systems using automated analysis: an application to android. Softw. Qual. J. **24**(2), 365–405 (2016). , https://doi.org/10.1007/s11219-014-9258-y
6. Heradio, R., Fernández-Amorós, D., Mayr-Dorn, C., Egyed, A.: Supporting the statistical analysis of variability models. In: 41st International Conference on Software Engineering (ICSE). pp. 843–853. IEEE / ACM (2019). , https://doi.org/10.1109/ICSE.2019.00091
7. Horcas, J.M., Galindo, J.A., Heradio, R., Fernandez-Amoros, D., Benavides, D.: Monte carlo tree search for feature model analyses: a general framework for decision-making. In: 25th ACM International Systems and Software Product Lines Conference. Leicester, UK (2021), submitted
8. Kocsis, L., Szepesvári, C.: Bandit based monte-carlo planning. In: Machine Learning: ECML 2006. pp. 282–293. Springer Berlin Heidelberg (2006)
9. Kroese, D.P., Brereton, T., Taimre, T., Botev, Z.I.: Why the monte carlo method is so important today. WIREs Computational Statistics **6**(6), 386–392 (2014). , https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.1314
10. Lopez-Herrejon, R.E., Linsbauer, L., Galindo, J.A., Parejo, J.A., Benavides, D., Segura, S., Egyed, A.: An assessment of search-based techniques for reverse engineering feature models. J. Syst. Softw. **103**, 353–369 (2015). , https://doi.org/10.1016/j.jss.2014.10.037
11. She, S., Lotufo, R., Berger, T., Wasowski, A., Czarnecki, K.: The variability model of the linux kernel. In: 4th International Workshop on Variability Modelling of Software-Intensive Systems (VaMoS). ICB-Research Report, vol. 37, pp. 45–51. Universität Duisburg-Essen, Linz, Austria (2010), http://www.vamos-workshop.net/proceedings/VaMoS_2010_Proceedings.pdf
12. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., Hassabis, D.: Mastering the game of go without human knowledge. Nature **550**(7676), 354–359 (2017). , https://doi.org/10.1038/nature24270
13. Sundermann, C., Thüm, T., Schaefer, I.: Evaluating #sat solvers on industrial feature models. In: 14th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS). pp. 3:1–3:9. ACM, Magdeburg, Germany (2020). , https://doi.org/10.1145/3377024.3377025