Inhaltsverzeichnis

1 Gerneral Notes

- Measurement Bias is hard to control (13)
- PUMA Tool
- Palladio
- PLeTsPerf

2 Automated Feature Detection

2.1 Formeln(13)

Operatoren

$$\begin{aligned}\text{Komposition/Konfiguration} &: a \cdot b \\ \text{Interaction} &: a\#b \\ \text{Feature Interaction} &: a \times b = a\#b \cdot a \cdot b \\ \Pi(a) &= \text{Performance von a, wenn a eine Konfiguration ist}\end{aligned}$$

Performance

$$\begin{aligned}\Pi(a \cdot b) &= \Pi(a) + \Pi(b) \\ \Pi(a\#b) &= \Pi(a \times b) - (\Pi(a) + \Pi(b)) \\ \Rightarrow \Pi(a \times b) &= \Pi(a\#b) + (\Pi(a) + \Pi(b))\end{aligned}$$

Es existieren $\mathcal{O}(n^2)$ viele Konfigurationsmöglichkeiten $\Rightarrow$ zusammenfassen in Konfigurationen C.

$$\begin{aligned}\text{Impact of Feature } a \text{ on } C\text{: } \Delta a_C &= \Pi(a \times C) - \Pi(C) \\ &= \Pi(a\#C) + \Pi(a)\end{aligned}$$

Beachte: C kann Konfiguration aus i Features sein $\Rightarrow \mathcal{O}(i^2)$ Additionen. Schlecht! Deswegen messen versuchen wir eine minimale Konfiguration $min(a)$ zu finden, die a **nicht** enthält aber mit a legitim erweiterbar ist.
Für jedes Feature a gilt:

$$\Delta a_{min} = \Pi(a \times min(a)) - \Pi(min(a))$$

Da das Ergebnis nur für die minimale Konfiguration gilt ist es nicht akkurat. Vorhersagen könne über das Messen eines interaction deltas verbessert werden. Seien a und $b \in C$, dann gilt:

$$\begin{aligned}\Delta(a\#b)_C &= \Pi(a\#b \times C) - \Pi(C) \\ &= \Pi(a\#b\#C) + \Pi(a\#b) + \Pi(c) - \Pi(C) \\ &= \Pi(a\#b\#C) + \Pi(a\#b)\end{aligned}$$

Jetzt können wir halbwegs verlässlich und mit reduziertem Aufwand den Performance Einfluss von einzelnen Features in einer bestimmten Konfiguration bestimmen. Es bleibt aber immer noch das Problem mit der Größe.
Um den Suchraum zu reduzieren stufen wir jedes Feature als interaktiv oder nicht interaktiv ein.

$$a \text{ interacts } \Leftrightarrow \exists C, D | C \neq D \wedge \Delta a_C \neq \Delta a_D$$

Dafür benötigen wir je 4 Messungen: $\Pi(a \times min(a))$, $\Pi(min(a))$, $\Pi(a \times max(a))$, $\Pi(max(a))$. Mit diesen kann man Δa_{min} und Δa_{max} bestimmen. Und für $C = min(a)$ und $D = max(a)$ für $\forall a$ bestimmen ob sie interaktiv sind oder nicht. Allerdings wissen wir nicht, welche Features mit welchen interagieren. $\mathcal{O}(n!)$ Kombinationen $\Rightarrow$ Suche wird durch Heuristiken geleitet:

1. Pair-Wise Interaction ($\mathcal{O}\left(\begin{smallmatrix}2\\n\end{smallmatrix}\right)$)
2. Higher Order Interactions (three way interactions, mostly based on Pair wise founds)
3. Hot-Spot Features additional measurements to find hotspot features.

Diese Heuristiken werden auf Implication chain angewendet um nach interaktiven paaren zu suchen. (s. Dokument)

3 Comparative Analysis of Software Performance Prediction Approaches in Context of Component-based System (2)

Literatur

[1] `https://visualstudio.microsoft.com/de/products/`. Accessed: 10.05.2019.

[2] A. A. Abdelaziz, W. M. N, W. Kadir, and A. Osman. Comparative analysis of software performance prediction approaches in context of component-based system. *IJCA*, pages 15–22, 2011.

[3] J. L. Díaz-herrera, P. Knauber, and G. Succi. Issues and models in software product lines. 2000.

[4] J. Guo, K. Czarnecki, S. Apel, N. Siegmund, and A. Wasowski. Variability-aware performance prediction: A statistical learning approach. In *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on*, pages 301–311. IEEE Press, 2013. ISBN 978-1-4799-0215-6. doi: http://dx.doi.org/10.1109/ASE.2013.6693089.

[5] J. Guo, D. Yang, N. Siegmund, S. Apel, A. Sarkar, P. Valov, K. Czarnecki, A. Wasowski, and H. Yu. Data-efficient performance learning for configurable systems. 2017. URL https://pure.itu.dk/portal/en/publications/dataefficient-performance-learning-for-configurable-systems(e88a1025-3d0c-48fa-9f72-b23.html.

[6] P. Jamshidi, N. Siegmund, M. Velez, C. Kästner, A. Patel, and Y. Agarwal. Transfer learning for performance modeling of configurable systems: An exploratory analysis. In *Proceedings of the 32Nd IEEE/ACM International Conference on Automated Software Engineering*, ASE 2017, pages 497–508, Piscataway, NJ, USA, 2017. IEEE Press. ISBN 978-1-5386-2684-9. URL http://dl.acm.org/citation.cfm?id=3155562.3155625.

[7] P. Jamshidi, M. Velez, C. Kästner, N. Siegmund, and P. Kawthekar. Transfer learning for improving model predictions in highly configurable software. *CoRR*, abs/1704.00234, 2017. URL http://arxiv.org/abs/1704.00234.

[8] M. A. Javidian, P. Jamshidi, and M. Valtorta. Transfer learning for performance modeling of configurable systems: A causal analysis. *CoRR*, abs/1902.10119, 2019. URL http://arxiv.org/abs/1902.10119.

[9] V. Nair, T. Menzies, N. Siegmund, and S. Apel. Faster discovery of faster system configurations with spectral learning. *CoRR*, abs/1701.08106, 2017. URL http://arxiv.org/abs/1701.08106.

[10] K. Pohl, G. Böckle, and F. J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag Berlin Heidelberg, Tiergartenstraße 17, 69121 Heidelberg, 2005.

[11] M. Rosenmüller, N. Siegmund, G. Saake, and S. Apel. Combining static and dynamic feature binding in software product lines. 2009.

[12] A. Sarkar, J. Guo, N. Siegmund, S. Apel, and K. Czarnecki. Cost-efficient sampling for performance prediction of configurable systems (t). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 342–352, Nov 2015. doi: 10.1109/ASE.2015.45.

[13] N. Siegmund, S. S. Kolesnikov, C. Kästner, S. Apel, D. Batory, M. Rosenmüller, and G. Saake. Predicting performance via automated feature-interaction detection. In *In Proc. of ICSE*, pages 167–177. IEEE, 2012.

[14] N. Siegmund, A. Grebhahn, S. Apel, and C. Kästner. Performance-influence models for highly configurable systems. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, pages 284–294, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3675-8. doi: 10.1145/2786805.2786845. URL http://doi.acm.org/10.1145/2786805.2786845.