Context-Aware Approach for Determining the Threshold Price in Name-Your-Own-Price Channels

Asanga Nimalasena(✉) and Vladimir Getov

Faculty of Science and Technology, University of Westminster,
115 New Cavendish Street, London W1W 6UW, UK
{a.nimalasena, v.s.getov}@westminster.ac.uk

Abstract. Key feature of a context-aware application is the ability to adapt based on the change of context. Two approaches that are widely used in this regard are the context-action pair mapping where developers match an action to execute for a particular context change and the adaptive learning where a context-aware application refines its action over time based on the preceding action's outcome. Both these approaches have limitation which makes them unsuitable in situations where a context-aware application has to deal with unknown context changes. In this paper we propose a framework where adaptation is carried out via concurrent multi-action evaluation of a dynamically created action space. This dynamic creation of the action space eliminates the need for relying on the developers to create context-action pairs and the concurrent multi-action evaluation reduces the adaptation time as opposed to the iterative approach used by adaptive learning techniques. Using our reference implementation of the framework we show how it could be used to dynamically determine the threshold price in an e-commerce system which uses the name-your-own-price (NYOP) strategy.

Keywords: Context-aware systems · Self-adaptation · Multi-action evaluation

1 Introduction

Context-aware systems react to changes in the perceived environment so that computing output is best suited to the current context. Generally, the context-aware systems are associated with mobility and applications related to mobile devices. This is mainly due to the fact that context changes are most likely encountered in mobile devices when these devices navigate through various contexts [1] as opposed to stationary devices where context data is often acquired through sensors.

But this is a narrow view of the context domain as there are many definitions as to what is a context. Context has been defined by location [2], location combined with behavior [3] or encompassing multitude of factors such as the definition given by Dey [4]: "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves". This definition makes no assumption about the mobility of devices and leaves to the context-aware system developers to decide what constitutes a context in their application. The adopted approach allows differentiating the operation environment from context based on potentiality and relevance [5]. Context-aware systems react to a context change by executing an action, while what action to execute is determined by the context inference. A context-aware application does context inference on the basis of the so-called 5W1H (Where, When, What, Who, Why, How) factors [6]. Expanding on this, context-aware applications look at the who's, where's, when's and what's (that is, what the user is doing) of entities and use this information to determine why the situation is occurring [7]. But it is not actually the application that determines why a situation is occurring, but the designer of the application. This means the designer has to capture the domain knowledge and input it to the system. This dependency on application designer to capture the context changes introduces inaccurate contexts and inflexible context definitions [8]. Moreover the context inference would fail if the system encounters a context which the designer did not foresee.

The self-learning and self-adapting methods are employed to overcome the aforementioned limitations. They use an iterative approach to find the best possible action when the system encounters an unknown context. If an action executed as a result of unknown context change is not the optimal then an error-feedback-loop-based correction mechanisms are employed to further refine the action. This process is iterated until the gap between the expected and the actual outcome is reduced or eliminated. However, when there are large numbers of actions to evaluate, the time to find the best action increases resulting in late system reaction to a context change.

This paper proposes a context-aware framework which concurrently executes and evaluates multiple actions from a dynamically created action space when an unknown context is encountered. The proposed framework overcomes the problems in the iterative approach of the self-adapting system and having to rely on application developers to encompass all possible contexts and context changes.

The rest of the paper is organized as follows. Section 2 reviews related work on context and self-adapting context-aware models. Section 3 gives a description of the proposed framework and Sect. 4 describes the implementation of the proposed framework for a NYOP channel. Section 5 presents experimental results of the implementation. Finally, the paper concludes with Sect. 6 which summarizes the findings from the evaluation and outlines directions for future work.

2 Related Work

Based on the association between contexts change and the resulting action(s) the existing context models could be loosely classified as *single context – single action* models and *single context – multiple action* models. The simplest model is the *single context – single action* model most commonly used for smart physical environments [9–11]. In practice, these types of models acquire sensory data from one or more devices (hard sensing) and act on other devices or make state changes that bring optimal result for the current context change. Due to the close association between this model and the physical hardware each context has one and only one precise action. This context-action pairing is built into the context-aware system by the application developers by considering all possible context changes the system is likely to encounter. A generic framework has been proposed [12], which allows system developers to formally define the adaptation to context changes based on system policies. However, this dependency on system developers could result in inaccurate and inflexible context definitions. He et al. [13] provide an example of a smart plant-watering context-aware system. One of the context values considered is the ambient temperature. However, if due to some freaky weather pattern an unusual temperature is encountered by the system which system developers had not foreseen, then the context inference would fail and the system would be unable to act on the perceived context change. A customizable context model which enables customization by the developers in order to recognize more context changes is presented in [14]. Other work makes use of a central repository of context knowledge that is periodically updated [15], but the drawback of having to depend on the system developers is still there.

The self-adapting and self-learning context-aware models are used to overcome these limitations arising from having to depend on the system developers to foresee all context changes. These models could be summarized as single context – multiple actions model. When an unknown context is encountered the system would execute sequence of actions iteratively with feedback loop base learning to self-adapt. A self-adapting algorithm which implements the resource, actors and policy triples (RAP model) is presented in [16] which use a closed feedback loop for adaptation. In [17] a formal method for incremental context awareness is proposed based breadth-monotonic model and depth-monotonic model. A self-adapting context with the use of context edges (a context edge is the border between two contexts) and context spaces is proposed on [18]. The model is based on Q-Learning with a feedback loop which finds the optimal action for each state by the reward it receives from the environment for actions taken in that state. Other self-adapting techniques used by context-aware system includes using case base reasoning to address domain specific problems and incomplete data sets [19] and try to address the lack of domain knowledge through self-adaption. Similarly, the approach described in [20] uses fuzzy sets to allow imperfection in context that is being sensed.

Though not from the context-aware domain, another commonly used autonomic adaptation model is IBM's MAPE-K (Monitor, Analyze, Plan, Execute, and Knowledge) loop reference model [21]. The components of the MAPE-K loop could be superimposed into the three main areas of sensing, inference and action of a context-aware application. However, a MAPE-K loop still depends on the system developers to formulate the event-condition-action (ECA) rules for self-adaptation [22] which makes it unsuitable for situation where unknown context could be encountered. ECA knowledge comes from human experts or other methods such as concept utility [23], Bayesian techniques [24] or reinforcement learning [25] which suffers from poor scalability when large number of ECA state changes exists.

A problem with these feedback-based models is that when the system consists of a large action space the amount of time needed to execute and evaluate each action iteratively keeps increasing and the overall time taken to find the best possible action could become unacceptably long. A context-aware application developed on the basis of soft sensing of social media [26, 27] data provides a different model to that of the feedback-loop-based self-adapting models described earlier. The focus in these models is towards context inference and ontology-based reasoning models are employed to achieve context-aware adaptation in them.

3 Proposed Context-aware Framework

The two primary goals of the context-aware framework that we propose are to reduce the dependency on system developers to capture and input all possible context changes and to eliminate the need for a feedback loop base iterative approach for self-learning/self-adapting. With the proposed framework the system developers are expected to setup few base parameters and input any domain knowledge or past experience they have of the application domain into a knowledge base. But this is not expected to be extensive as the system is expected to expand its knowledge base dynamically. As iterative approach becomes unfeasible when there's a large action space to evaluate, the framework proposes a concurrent multi-action evaluation approach where action space is executed and evaluated in a single pass reducing the time for adaptation.

The proposed framework consists of three systems, namely the context system, the inference system and the actions system. These three systems encompass the main characteristics of a self-adapting context-aware system, which are sensing, actuators (actions) and inference/self-adapting. Figure 1 shows a high-level diagram of the proposed framework and system components.

The primary objective of the context system is context sensing and acquisition. How the context sensing happens is implementation specific and could be either hard sensing or soft sensing. The framework assumes that context space is a heterogeneous where context values are acquired from various sources. As a result of the heterogeneous context space the system could acquire wide variety of context values in different

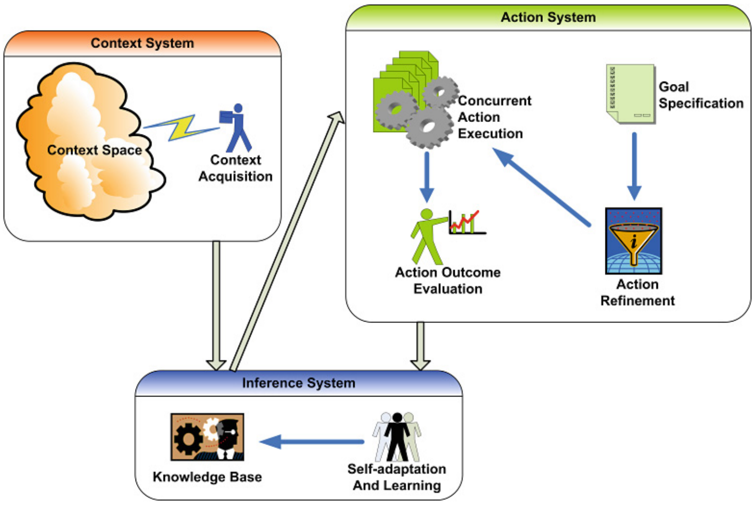

Fig. 1. High level system diagram of the proposed framework

units of measurement. Context acquisition is expected to transform these heterogeneous context value types in different measurement units into single unit of measurement allowing comparison of contexts. This context comparison is used in the action system to find the closest known context to an unknown context.

The inference system consists of a knowledge base and a self-adaption/learning mechanism. When a context change is sensed the context inference is carried out querying the knowledge base to identify if the new context values are known. If the new context is inferred to be unknown then the action system is invoked. The other component in the inference system is the self-adaptation and learning mechanism which updates the knowledge base and adapt the context-aware application based on the outcome from the action system. The knowledge base could be modelled in many different ways such as semantic representation of context [28]. The use of ontology to represent context has the added benefit of leveraging inherent inference capabilities that comes with ontology classifications.

The action system is responsible for concurrent action execution and evaluation when an unknown context is encountered. The goals of the action system are to reduce the number of required actions qualifying for evaluation and to complete the action execution and evaluation in a single pass as opposed to iterative manner. To achieve this first goal the action system uses goal specification and action refinement. The goal specification defines the extremities of the variable parameter used to build the action space. This is different to existing goal driven approaches to self-adaptation [29] which are based on rules created by the system developers. These extremities are denoted as G_{lo} and G_{hi} and are considered elements of the configuration parameter space which are used to differentiate one action from another.

$$(G_{lo},\ G_{hi}) \in \{\text{ configuration parameter space}\}$$

The action refinement limits what action qualifies to be in the action space thus reducing the action space size. Without the limiting effects of the action refinement the context-aware system would have to experiment on every value between G_{lo} and G_{hi} which would be a resource and time intensive endeavor. The action limiting process starts by identifying from the knowledge base, the context that is closest to the unknown context. The closeness is measured by the difference of the context values. If more than one context is found to be the closest then the priority of each context is considered. The configuration parameter setting of this known context is used to device the initial action. This is denoted as A_k and defined as a function of the configuration parameter configuration of the closest known action

$$\text{Initial action} = A_k(\text{configuration}_k)$$

The framework introduces three parameters for the dynamic creation of the action space. They are the lower bound expansion range denoted by p which specifies number of actions to define in the direction of G_{lo}. The upper bound expansion range denoted by q specifies the number of actions to define in the direction of G_{hi} and finally the distance between each configuration parameter denoted by Δ. These three parameters and the goal specification are the only inputs that depend on system developers, effectively eliminating the need to identify all possible context changes. Having defined these, all the actions (action space) that needed to be executed and evaluated to find the best course of action for unknown context could be defined as a union of three action sets.

$$
\begin{aligned}
\text{Action space} = \{\, & A_k\,(\text{configuration}_k) \;\cup \\
& A_p\,(\text{configuration}_p) \;\cup \\
& A_q\,(\text{configuration}_q) \\
& \mid\; p = \{1 \,..\, n\},\, n > 0,\;\; q = \{1 \,..\, m\},\, m > 0, \\
& \text{configuration}_k \;- p\Delta \geq G_{lo,}, \\
& \text{configuration}_k \;+ q\Delta \leq G_{hi}, \\
& \Delta > 0 \\
\}\quad &
\end{aligned}
$$

The defined actions are then executed concurrently in a private workbench. The private workbench ensures that configuration changes in each action under evaluation is opaque to and does not affect the current state of the system. As all actions are executed concurrently the outcome of each action is known at the same time, as opposed to iterative approach where the analysis of results has to be delayed until all actions have finished. This concurrent action evaluation is somewhat similar to the optimizing technique used in particle swarm optimization (PSO) [30] where each particle is a possible solution. However, one key difference between PSO and our action space is that in PSO the particles must update their velocity and position relative to the particle with the global optimal after each iteration. In our proposed framework each action is a candidate to be a global optimal and to evaluate the problem space independent of each other.

The final phase of the action system is the outcome evaluation. The evaluation criteria for choosing the action that results in the highest benefit depends on the domain in which the context-aware system is implemented. Thus, the best action to execute (and its configuration parameter) as a result of the unknown context change could be formally defined as

$$
\begin{aligned}
\text{configuration}_{best} = \{ & \\
& \forall\, \text{configuration}_i \in \{\text{action space configurations}\} \\
& \exists\, A_i\,(\text{configuration}_i)\text{: Maximum (Benfit}(A_i)) \\
\} &
\end{aligned}
$$

Once the best setting for the configuration parameter is known for the unknown context it could be used to update the knowledge base so the context-aware system recognizes this context in the future (learning and adaptation). Figure 2 shows the information flow for known context detection and unknown context detection. $C_1 - C_4$ in Fig. 2 represents context considered relevant to the interaction between a user and an application.

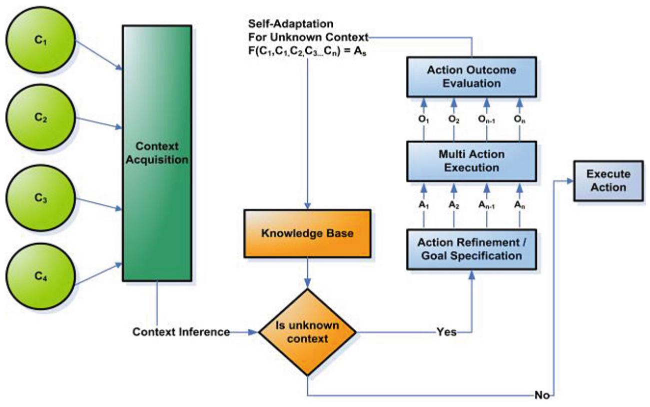

Fig. 2. Information flow for known and unknown context detection

4 Context-aware Framework Implementation for NYOP Channels

The proposed context-aware framework was implemented for a use case where an hotelier sells rooms through a NYOP channel. The NYOP operates by allowing buyers to bid for an item on a perceived value rather than based on the actual value set by the seller. The seller has an internal threshold price hidden from the buyers which he considers to be the minimum value for a bid in order to successfully complete the transaction. For our experiments we do not employ any such NYOP strategies [31]. Instead, each value is considered as an individual bid and not as a subsequent bid part of a bidding transaction. If the hotelier decides to accept or reject a bid solely based on its value, then he will not have the fluidity to react to the demand uncertainty that occurs due to the change in context. A context-aware approach is beneficial in this case, instead of having one threshold price T the context-aware NYOP system could be set up multiple threshold price $T_1 \ldots T_n$. Bids will be evaluated against all threshold prices in real time and results evaluated to find out which threshold price results in highest yield (T_{Max}). Once the highest yielding threshold price is identified, the e-commerce system is adapted to use it to evaluate all bids under current context.

We have developed a scenario where a new event has been planned near the vicinity of the hotel and there is no historical data or knowledge to rely on to set a threshold price which would give a high yield. We define this as an unknown context based on the definition given earlier on [4] as the hotelier is unaware of the threshold price to use in this situation (context) to optimize the interaction between buyer and seller. The context space was modelled with three soft sensed contexts, which are current occupancy (source: internal reservation database), event location, event type (extracted from social media. i.e. Twitter feed). Taking the NYOP threshold price as the configuration parameter, the formal modeling of the proposed context-aware framework was instantiated with the following values. Goal specification (G_{lo}, G_{hi}) = (210, 350). In essence, the goal specification is a sub-range of the entire application value range. For example, if the universe of prices for a hotel room is considered, it could vary between \$0 (100 % discounted) – millions of dollars (based on luxury). But for this particular hotelier such a large value range is irrelevant. His interest lies in a smaller range of values so that accepted bids do not result in loss or high price resulting in low conversions and unsold rooms. Action refinement values (p, q, Δ) = (2, 2, 15). Initial action (closest known context action) = A (250). As stated earlier G_{lo}, G_{hi}, p, q, Δ are the only inputs from the system developer to the system and initial action is retrieved from the knowledge base.

The evaluation criterion was set to threshold price with highest number of successful bids. It is possible that some bids would be successful in more than one threshold. In such cases the bid would be considered successful only in the highest threshold it exceeds. The context-aware application was developed as a Java web application and deployed in Tomcat application container which ran on a server with 12 GB RAM, 2.0 GHz Intel quad core processor and 500 GB SAS disks running on RedHat Linux 6.4. The knowledge base was modelled using Java implementation of Protégé OWL API. We devised two test cases for the evaluation. One test case simulates an unknown context in which the majority of bid values are lower than the threshold value of the closest known context. If the hotelier does not lower the threshold price to capture the bids, he will lose out under the current context. The second test case simulates an unknown context under which the majority of bid values are considerably higher than the threshold price. Under this context the hotelier can increase the threshold price to gain a higher yield. This is a NYOP strategy that encourages higher bidding values. Though we make no assumption about the bidding strategies we include this test case for the completeness of the evaluation, to test the suitability of the framework works for both cases.

5 Experiments and Results

Two sets of bid values were generated for each of the test case (1000 values each) using a normal distribution function where mean values are 237.50 and 268.50 for lower and higher bid value test cases. The control test was defined as using the closest known context threshold price to evaluate the bid values while in the unknown context (non-adaptive system). The bid submissions were emulated using JMeter's http requests. The action space, created dynamically based on the (G_{lo}, G_{hi}, p, q, Δ) resulted in 5 actions to be concurrently executed and evaluated. These are denoted as A(220), A(235), A(250), A(265) and A(280) in the Figs. 3 and 4 below.

In this unknown context, Fig. 3 shows the majority of successful bids which have occurred in the action that had a threshold value of 235. The hotelier could associate the current unknown context with the threshold value of 235, thus effectively evolving the system to recognize the current unknown context in the future. We know that this conclusion is correct as we have generated the bid values using a normal distribution with a mean value of 237.50. For the second test case shown in Fig. 4, the majority of successful bids have occurred in the action that had a threshold value of 265. We know that this is true because the bid values generated under the normal distribution had a mean value of 268.50. In both cases, if the hotelier has decided to stay with the closest

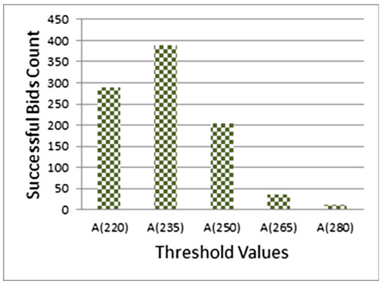

Fig. 3. Context resulting in lower bid values

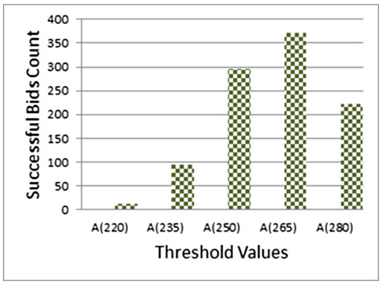

Fig. 4. Context resulting in higher bids values

known context's threshold price, the successful bid count would have been less than the one achieved by the context-aware adaptive approach.

6 Conclusion

In this paper, we proposed a context-aware framework which reduces the dependency on system developers to capture all possible context changes and eliminate the feedback loop base approach to self-adaption. We have listed the generic framework structure and presented the formal model that underpins it. An implementation of the proposed framework was completed for the NYOP scenario. The experimental results from the tests have shown that the framework concurrent multi-action evaluation approach could correctly identifying the best course of action for the unknown context and is able to evolve the system, thus being able to recognizing more contexts over time. Though we implemented the framework for NYOP channel case study, we believe the framework could be easily used in many other domains such as a context-aware approach to experiment-based performance tuning.

References

1. Using Apache Hadoop* for Context-Aware Recommender Systems, Intel Corporation White paper (2014). http://intel.ly/1nRRoUZ
2. Schilit, B.N., Theimer, M.M.: Disseminating active map information to mobile hosts. IEEE Network **8**(5), 22–32 (1994)
3. Brown, P.J., Bovey, J.D., Chen, X.: Context-aware applications: from the laboratory to the marketplace. IEEE Pers. Commun. **4**(5), 58–64 (1997)
4. Dey, A.K.: Understanding and using context. Pers. Ubiquit. Comput. **5**, 4–7 (2001)
5. Roman, G., Julien, C., Payton, J.: Modeling adaptive behaviors in context UNITY. Theoret. Comput. Sci. **376**, 185–204 (2007)

6. Ko, K.-E., Sim, K.-B.: Development of context aware system based on Bayesian network driven context reasoning method and ontology context modeling. In: Proceedings of the International. Conference on Control, Automation and Systems (ICCAS). pp. 2309–2313 (2008)
7. Madhusudanan, J., Selvakumar, A., Sudha, R.: Frame work for context aware applications. In: Proceedings of the International Conference on Computing Communication and Networking Technologies, pp. 1–4 (2010)
8. Lee, H., Lee, S.: Decision supporting approach under uncertainty for feature-oriented adaptive system. In: 2nd User Centered Design and Adaptive Systems (COMPSACW) (2015)
9. Al-Rabiaah, S., Al-Muhtadi, J.: Context-aware security framework for smart spaces. In: Innovative Mobile and Internet Services in Ubiquitous Computing, pp. 580–584 (2012)
10. Wu, C., Weng, M., Lu, C., Fu, C.: Hierarchical generalized context inference or context-aware smart homes. In: Intelligent Robots and Systems (IROS), pp. 5227–5232 (2012)
11. Gupta, A., Pandey, O.J., Shukla, M., Dadhich, A., Ingle, A., Gawande, P.: Towards context-aware smart mechatronics networks: integrating swarm intelligence and ambient intelligence. In: Issues and Challenges in Intelligent Computing Techniques, pp. 64–69 (2014)
12. Alagar, V., Mohammad, M., Kaiyu, W., Hnaide, S.A.: A framework for developing context-aware systems. Trans. Context-aware Syst. Appl. ICST **1**(1), 1–26 (2014). (Springer)
13. He, J., Zhang, Y., Huang, G., Cao, J.: A smart web service based on the context of things. ACM Trans. Internet Technol. **11**(3), 13:1–13:23 (2012)
14. Yu, L., Wang, Z., Huang, Y., Chen, S.: Building customizable context-aware systems. In: 2011 International Joint Conference on Service Sciences (IJCSS), pp. 252–256 (2011)
15. Chang, J., Na, S., Yoon, M.: Intelligent context-aware system architecture in pervasive computing environment. In: Parallel and Distributed Processing with Applications, pp. 745–750 (2008)
16. Cioara, T., Anghel, I., Salomie, I., Dinsoreanu, M., Copil, G., Moldovan, D.: A self-adapting algorithm for context aware systems. In: Proceedings of 9th Roedunet International Conference (RoEduNet), pp. 374–379 (2010)
17. Loke, S.W.: Incremental awareness and compositionality: a design philosophy for context-aware pervasive systems. Pervasive Mobile Comput. **6**(2), 239–253 (2010)
18. O'Connor, N., Cunningham, R., Cahill, V.: Self-adapting context definition. In: Proceedings of the 1st International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2007), pp. 336–339 (2007)
19. Nwiabu, N., Allison, I., Holt, P., Lowit, P., Oyeneyin, B.: Situation awareness in context-aware case-based decision support. In: Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), pp. 9–16 (2011)
20. Anagnostopoulos, C., Hadjiefthymiades, S.: Advanced inference in situation-aware computing. Part A Syst. Hum. **39**(5), 1108–1115 (2009)
21. An architectural blueprint for autonomic computing, IBM Corp. http://ibm.co/1IP7TvG
22. Huebscher, M.C., McCann, J.A.: A survey of autonomic computing—degrees, models and applications. ACM Comput. Surv. (CSUR) **40**(3), 1–28 (2008)
23. Bhola, S., Astley, M., Saccone, R., Ward, M.: Utility-aware resource allocation in an event processing system. In: International Conference on Autonomic Computing, p. 55 (2006)
24. Guo, H.: A Bayesian approach for autonomic algorithm selection. In Proceedings of the IJCAI Workshop on AI and Autonomic Computing: Developing a Research Agenda for Self-Managing Computer Systems (2003)

25. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
26. Derczynski, L.R.A., Yang, B., Jensen, C.S.: Towards context-aware search and analysis on social media data. In: 16th International Conference on Extending Database Technology, pp. 137–142 (2013)
27. Hu, X., Li, X., Ngai, E.C.-H., Leung, V.C.M., Kruchten, P.: Multidimensional context-aware social network architecture for mobile crowd sensing. IEEE Commun. Mag. **52**(6), 78–87 (2014)
28. Ejigu, D., Scuturici, M., Brunie, L.: Semantic approach to context management and reasoning in ubiquitous context-aware systems. In: Proceedings of ICDIM (2007)
29. Salehie, M., Tahvildari, L.: Towards a goal-driven approach to action selection in self-adaptive software. Softw. Pract. Experience **42**(2), 211–233 (2011)
30. Qi, B., Shen, F.: Performance Comparison of Partical Swarm Optimization Variant Models. In: Information Technology: New Generations (ITNG), pp. 575–580 (2014)
31. Hinz, O., Hann, I., Spann, M.: Price discrimination in e- commerce? An examination of dynamic pricing in name-your-own price markets. Mis Q. **35**(1), 81–98 (2011)