Chapter 6
Knowledge Representation and Modelling: Structures and Trade-offs

Leandro L. Minku, Lukas Esterle, Georg Nebehay, and Renzhi Chen

Abstract As explained in chapter 5, self-aware and self-expressive systems can be designed based on a number of patterns and primitives. In this chapter, we discuss issues to be considered when developing such systems, specially when going through phases 3 (selecting the best pattern) and 5 (determining primitives and alternatives), and possibly also phase 7 (score alternative primitives) of the methodology for designing and implementing self-aware and self-expressive systems described in section 5.4. Specifically, we explain several features which may be present in self-aware and self-expressive systems, namely adaptivity, robustness, multi-objectivity and decentralisation. We discuss their implications in terms of knowledge representation and modelling choices, including potential trade-offs among different choices. Knowledge representation is interpreted loosely, referring to any structure used to store knowledge, whereas knowledge modelling is considered to be the process used to create and update such knowledge structures. The discussion raises awareness of general issues to be considered and carefully reflected upon when developing self-aware and self-expressive systems.

Leandro L. Minku and Renzhi Chen
School of Computer Science, University of Birmingham, UK, e-mail: {l.l.minku,rxc332}@cs.bham.ac.uk

Lukas Esterle
Institute of Networked and Embedded Systems, Alpen-Adria-University Klagenfurt, Austria, e-mail: lukas.esterle@aau.at

Georg Nebehay
Safety and Security Department, Austrian Institute of Technology, Austria, e-mail: georg.nebehay@ait.ac.at

6.1 Introduction

As explained in Chapter 5, self-aware and self-expressive systems can be designed based on a number of patterns and primitives. In this chapter, we discuss issues to be considered when developing such systems. Specifically, we explain different features which may be present in self-aware and self-expressive systems, and their implications in terms of knowledge representation and modelling choices. The following features are considered: adaptivity (Section 6.2), robustness (Section 6.3), multi-objectivity (Section 6.4), and decentralisation (Section 6.5).

These features and their implications should be considered specially when going through phases 3 (selecting the best pattern) and 5 (determining primitives and alternatives) of the methodology for designing and implementing self-aware and self-expressive systems described in Section 5.4, and possibly also phase 7 (score alternative primitives).

In the context of self-aware and self-expressive systems, the terms knowledge representation and modelling are interpreted in a loose way. Knowledge representation is not restricted to formalisms such as semantic nets, frames, rules or ontologies. Instead, it refers to any structure used to store knowledge. Knowledge modelling can then be seen as the process used to create and update such knowledge structures / models.

The discussion provided in this chapter highlights general choices and their trade-offs in the specific context of self-aware and self-expressive systems, and is not intended to be exhaustive and comprehensive or empirically demonstrated through experiments. In particular, different applications may be affected by different choices in different ways, and certain choices may be more relevant for some applications than for others. It would thus be infeasible to list all possible choices and to empirically analyse how these choices affect all possible applications of self-aware and self-expressive systems. Rather than that, the discussions provided in this chapter aim at raising awareness of general points to be considered and carefully reflected upon when developing self-aware and self-expressive systems.

The rest of this chapter is organised as follows: Sections 6.2 to 6.5 explain the features of self-aware and self-expressive systems and discuss their implications to knowledge representation and modelling, and Section 6.6 provides a summary of the chapter.

6.2 Adaptivity

In order to perform well in a given environment, systems usually need to be adapted not only to their goals, but also to the environments where they are embedded. In standard computing tasks, the environment where a system operates typically exhibits a static behaviour. When the type of expected input is well-known, it is relatively easy to devise a program to map the input to the desired output. As computing tasks get more and more complex, this static property can no longer be relied upon,

because the environments where these tasks must be performed are frequently dynamic and present uncertainties. Therefore, self-aware and self-expressive systems usually need the ability not only to self-adapt to a given environment condition, but also to self-adapt to (possibly unexpected) changes in the environment. In some applications, the goals of the system may also change with time, implicating on the need for self-adaptation to new goals.

The need for adaptivity in self-aware and self-expressive systems leads to several general knowledge representation and modelling choices, such as whether and how to model time and what structures to use for dealing with changes. Each individual application of self-aware and self-expressive systems also involves choices related to whether and how to model the environment; whether and how to represent the suitability of a solution to the environment; and what components of the system to adapt. Section 6.2.1 explains adaptivity in self-aware and self-expressive systems and gives some examples of applications involving adaptivity. Section 6.2.2 explains some general implications of adaptivity to the development of such systems.

6.2.1 Definition and Examples

Two definitions of the verb *adapt* given by the Oxford dictionary are as follows [305]:

- (With object) Make (something) suitable for a new use or purpose; modify.
- (No object) Become adjusted to new conditions.

In biology, adaptation can be seen as the process whereby an organism becomes better suited to its habitat(s) [103][1]. Similarly, in the context of self-aware and self-expressive systems, *adaptation* can be seen as the process whereby a system becomes better suited to its environment, given its purpose. If its environment or purpose changes, the system must become adjusted to the new conditions. The term *adaptivity* refers to the ability of a system to perform such adaptation automatically, i.e., the ability of a system to self-adapt. It differs from the term *adaptability*, which refers to systems that can be substantially customised by users through tailoring activities by themselves [414].

Adaptivity can be advantageous for many self-aware and self-expressive systems. For example, let's consider a credit card approval machine learning system whose aim is to accurately predict whether a customer should be given credit or not [408, 269]. Learning consists in creating a model of the relationship between customers' features (e.g., age, salary, gender, etc.) and a dependent variable describing whether or not they have paid all their bills in, say, the last twelve months. This model represents the system's knowledge of its environment, and is created based

[1] Please note that we do not intend to provide a comprehensive set of possible definitions of adaptivity, but focus on a couple of definitions that can help understanding the concept of adaptivity in self-aware and self-expressive systems.

on examples of existing customers and their payments (training examples). Predictions on whether a new customer should be given credit or not are performed based on this model. The environment where credit card approval systems operate may change due to several factors. For example, customers who usually paid all their bills may become less likely to pay their bills due to some economic crisis. A credit card approval system should thus be able to adapt to such changes. This can be achieved by monitoring how well existing models of the environment perform on new incoming training examples, and updating such models or creating new models to reflect new environment conditions [408, 22, 269, 139, 224, 326].

Another example of self-aware and self-expressive system where adaptivity can be advantageous is that of distributed smart camera networks [124] (Section 7.4.1). Such networks can be used, for example, to track objects. In order to accomplish this goal, each smart camera must implement a handover mechanism, which refers to finding the next camera to see the target object once it leaves the field of vision of the current camera [120]. This mechanism could be based on a static known vision graph whose neighbourhood structure is encoded *a priori* in the cameras. However, this would require extensive manual design-time work to determine this graph. Moreover, if a change such as the failure of a given camera or a camera being moved to a different position happens, the cameras would need to be manually updated to reflect the new situation. A system where cameras can automatically learn their vision graph and adapt their handover mechanism would be desirable to avoid heavy manual (re-)tuning of the system. This can be achieved, for example, by using a market-based mechanism where each camera "bids" for objects with the goal of maximising its own utility [124]. Each camera's utility is updated with time, and increases based on the objects it tries to track, their visibility in the camera's field of vision, the performance of the camera in tracking these objects, and the payments it receives from other cameras. The utility reduces based on the amount it pays to sell objects to other cameras. Each camera's relative utility is then used to automatically determine which camera the object should be handed over to. A model of the environment (vision graph) can be automatically created and updated based on the cameras that trade objects with each other.

An additional example of self-aware and self-expressive system where adaptivity is used is the decentralised system for synchronisation of music agents described by Nymoen et al. [297] (Section 7.3.1). This system is designed for use in collaborative active music mobile apps, where a group of non-musicians use their mobile phones (agents) to interact with music together. This interaction could either have the purpose of creating music or playing back pre-recorded music. The goal of the synchronisation system is to synchronise the timing of the several agents who are participating in the musical interaction. In order to achieve this goal, each agent must be able to adapt its timing to the timing of other agents. For that, each agent has knowledge of its own timing, which is represented by a state that oscillates in cycles with a certain phase and frequency. Whenever the cycle of an agent is complete, it emits a short sound that can be heard by other agents. This sound is used by agents to update their states so that their timing becomes more similar to the timing of the agent who emitted the sound.

In general, the advantage of adaptivity is to enable automated reactive modification of behaviour at run-time in order to suit (1) a given environment condition and (2) a set of goals. Adaptivity not only avoids the need for pre-defining the system's behaviour for a particular environment, but also allows it to operate in possibly unforeseen situations. Such advantage does not come without risks. For instance, in some systems, it may be difficult to ensure that adaptation to changing conditions is successfully and timely achieved. Depending on the application, the result of unsuccessful self-adaptation could range from slight under-performance to overall system failure.

Adaptivity is achieved based on the capabilities possessed by an agent and/or collective. In order to be able to adapt, a system must be able to monitor its own state and/or the state of its external environment. For example, the credit card approval system explained above is able to monitor its own state through a performance measure which is updated whenever new training examples are made available from the environment, besides being able to maintain and update a model of its external environment by learning such incoming training examples. In the distributed smart camera example, cameras can monitor their own utility, besides being able to detect currently visible objects given their position and acquire knowledge of their relationship to neighbouring cameras in the network. In the music synchronisation system, each agent has knowledge of its own timing and can listen to sounds emitted by other agents in the environment. As explained in Chapter 2, a self-aware system could be defined as a system able to obtain knowledge about its internal state and/or knowledge about its environment. Such knowledge should be obtained by the system on an ongoing basis throughout the system's lifetime, rather than being programmed by a domain expert. Therefore, we could see self-awareness as an enabler for adaptivity.

6.2.2 Implications

This section describes the implications of adaptivity on knowledge representation and modelling. Adaptivity is usually associated to a cost; a limited amount of adaptivity may have lower cost than a more thorough level of adaptivity [389]. Therefore, separating different types of adaptivity into different levels of self-awareness can help to avoid the cost associated to unnecessary adaptation. This section first discusses implications of adaptivity in terms of choices of levels of self-awareness (Section 6.2.2.1). Then, it discusses implications in terms of modelling states and environments (Section 6.2.2.2), modelling time (Section 6.2.2.3) and dealing with changing environments (Section 6.2.2.4).

6.2.2.1 Choice of Levels of Self-Awareness

Stimulus-awareness: this type of self-awareness allows a node to adapt to *static* environmental conditions. When the environment is dynamic, stimulus-awareness is also capable of adapting to changes, even though this might be rather limited. The efficiency and effectiveness of the adaptation would highly depend on the type and severity of the change in the environment, given that this level of self-awareness alone would not be able to distinguish between past, present and future stimuli.

Interaction-awareness: when a system is composed of several agents, a given agent will frequently need to adapt not only to its external environment, but also to other agents in this environment. Interaction-awareness allows multi-agent self-aware systems to achieve that. For example, the distributed smart camera system described in Section 7.4.1 [124] is interaction-aware, because each smart camera can interact with other smart cameras in the environment in order to hand objects over to them. It is interesting to note that, when a system is composed of several agents, adaptivity of the collective system as a whole may be achieved through the adaptivity of each of its agents.

Time-awareness: this type of self-awareness can be very helpful for achieving adaptivity to dynamic environments, which are environments that may suffer changes with time. Time-awareness allows a system to distinguish between past and current events, helping it to acquire knowledge about the current environment condition and to adapt to any changes suffered by the environment. For example, a credit card system able to adapt to dynamic environments such as the one mentioned in Section 6.2.1 [408, 269] must be able to distinguish between past and current examples while monitoring the performance of existing models, so that the monitored performance reflects the current situation of the environment rather than being outdated.

Goal-awareness: given that adaptation is the process whereby a system becomes better suited to its environment *given its purpose*, the concept of adaptation is closely linked to the goal(s) of the system. Therefore, many self-aware systems will use explicit knowledge of their goal(s) in order to achieve adaptivity. For example, a credit card system such as the one explained in Section 6.2.1 could explicitly try to minimise the error of its predictions by using a certain error measure. Some systems are composed of several agents who are aware of their own goals, even though each agent may not be aware of the goals of the system as a whole. For example, each smart camera in the system explained in Section 7.4.1 [124] is unaware of the goals of the system as a whole. Instead, each smart camera tries to maximise its own goal, which is represented by its utility value. This allows each camera to self-adapt to its environment and to other cameras, leading to the achievement of the goals of the system without explicitly considering the goals of the system as a whole. A system able to adapt to changing goals would also normally be expected to be goal-aware. However, a system does not necessarily need to be aware of its goal(s) in order to achieve adaptivity to changes in the environment, as long as its adjustments, which

can possibly be made via implicit goals, make it more suitable to the environment given its purpose.

Meta-self-awareness: self-adaptive systems do not need to be meta-self-aware. However, meta-self-aware systems allow a more thorough level of adaptivity, where systems are able to adapt the way in which the levels of self-awareness are realised (e.g., by changing algorithms), or adapt the levels of self-awareness themselves (e.g., by activating / deactivating certain levels). For instance, the active music system described in Chapter 14 can automatically decide which strategy to adopt for mapping between a gestural controller and sound engine in order to adapt the music to the current user intention. Another example of meta-self-awareness would be to to use hyper-heuristic optimisation algorithms [49] to automatically decide which meta-heuristic optimisation algorithms to use in goal-aware systems.

While the different levels of self-awareness may help a system to achieve better adaptivity, they also require additional computation time. This may not be relevant to standard computing systems, but is crucial in applications requiring near-real-time performance or in embedded devices where resources are very limited. Especially with meta-self-awareness, the system may need to rely on multiple learning techniques simultaneously. A trade-off between the level of self-awareness that a system possesses and the amount of learning a system that it is able to perform arises.

6.2.2.2 Modelling States and Environments

In order to achieve adaptivity, self-aware and self-expressive systems will frequently rely on learning models of the state of their nodes, or models of the environment where they are embedded. However, the nature of the data available for learning must be considered when deciding what type of model to build and which type of learning algorithm to use. With respect to the nature of the data, models can be built based on the following types of learning algorithms:

Supervised learning algorithms: data come in the form of examples with inputs and desired outputs (labelled data). Desired outputs frequently need to be provided by a "teacher" from outside of the computing system. While knowledge of desired outputs may facilitate learning, providing desired outputs can be a time-consuming and difficult task. An example of supervised learning system is a credit card system fed with data describing the features of customers and explicit information on whether these customers have paid their bills or not [408, 269].

Unsupervised learning algorithms: data come in the form of input attributes with no desired outputs provided (unlabelled data). This avoids the need for a possibly costly external "teacher" to provide desired outputs. However, in self-aware and self-expressive systems, this frequently means that nodes have to learn in a completely autonomous fashion, employing only the information present within the

node itself or within nodes in the immediate environment. An example of unsupervised learning algorithm would be a clustering algorithm to identify the characteristics of different groups of customers, but without pointing out whether these customers are good or bad payers.

Semi-supervised learning algorithms: these algorithms can be used to achieve a compromise between the two extremes represented by supervised and unsupervised learning. In semi-supervised learning, part of the data come labelled and part come unlabelled. Semi-supervised algorithms can be used to find hidden structures in the data by using unlabelled data and infer outputs based on the labelled data.

Reinforcement learning algorithms: data come in the form of feedback from the environment representing how close the system is to achieving its goal(s), or how well it is performing its task(s). This type of learning algorithm is adequate when it is not possible to obtain desired outputs, but feedback from the environment is available. For instance, a robot learning how to walk may receive feedback from the environment in the form of how far it moved forward. An example of reinforcement learning algorithm is the algorithm for scheduling tasks in wireless sensor networks presented by Khan and Rinner [216], where each node in the network uses a reward function to decide which task to perform next.

6.2.2.3 Modelling Time

Stimuli from the environment will frequently come in the form of data and be used to learn and update a model representing the knowledge that the system has of the environment. Therefore, the choice of how to model time can be seen as the choice of whether and how to distinguish the time stamps in which different data points were produced or presented to the system. There are different ways to process data with respect to time, and the relevancy of their advantages and disadvantages can vary considerably depending on the intended application. In general, the following options are available:

Offline data processing algorithms: these algorithms create models with pre-existing data sets describing the environment, and there is no distinction between the time stamp of different data points within pre-existing data sets. Therefore, models of the environment are built before the system is put into operation. The advantage of that is that the designer of the system can frequently have a good idea of how well adapted the model is to the environment before it starts to be used. The disadvantage is that it is not possible to update the model with additional data incoming with time. Therefore, the model cannot be adapted to any changes in the environment. Traditional machine learning algorithms typically process data in offline mode, e.g., Backpropagation using several epochs for learning [36].

Online data processing algorithms: these algorithms are aware of the chronological order of each data point. Each data point is processed separately and then discarded. The advantage of that is that models can be improved over time with incoming data. As old data points are not re-processed, these algorithms can also be memory and time efficient, being suitable for applications with strict memory and time constraints or for applications where large amounts of data need to be processed. These algorithms can also be combined with strategies to adapt models to changes in the environment (see Section 6.2.2.4). The disadvantage is that it may be difficult to ensure beforehand how well the model will behave over time. An example of algorithm that processes data online is the multi-objective ensemble method for class imbalance learning [408], which has been applied to fault detection, credit card approval and network intrusion detection. Another example is the learning algorithm proposed by Fern and Givan [130], which has been applied to prediction of conditional branch outcomes in microprocessors in order to cope with the strict memory and time constraints of this application.

Chunk-based data processing algorithms: these algorithms process data in chunks. There is no distinction between the time stamp of different data points within a chunk, but the relative chronological order of the chunks themselves is known. Each data chunk is processed and then discarded, but data points within a chunk can be re-processed several times before the chunk is discarded. The advantage of chunk-based algorithms is that models can be improved over time with incoming data. Chunk-based algorithms can also be combined to strategies to adapt models to changes in the environment (see Section 6.2.2.4). As data points belonging to a given chunk can be re-processed several times before a chunk is discarded, chunk-based algorithms may be able to achieve higher accuracy on each chunk than online algorithms. The disadvantage is that this is less memory and time efficient than online learning. In addition, chunk-based algorithms need to wait for a whole chunk of data to arrive before the model can be updated, being unable to update the model with continuously incoming examples. Choosing the best chunk size may also be difficult. A too large chunk would lead to slow adaptation, whereas a too small chunk may cause the system to perform poorly depending on the algorithm being used. An example of algorithm that processes data in chunks is the the music synchronisation algorithm [297] explained in Section 6.2.1. Rather than immediately updating the frequency of an agent once it hears a short sound from another agent, each agent collects the short sounds received from other agents within a whole oscillation cycle before updating its frequency. In this application, this can prevent premature convergence of the system to an undesirable state [297], i.e., waiting for chunks can help the system to adapt better.

6.2.2.4 Dealing with Changing Environments

There are different possible strategies to deal with changing environments, and each strategy incurs different decisions in terms of knowledge representation and mod-

elling. For instance, a system may or may not use knowledge models able to detect changes, i.e., change detection methods:

Change detection method: in this case, explicit mechanisms are used to monitor the environment or the suitability of the system to the current environment condition, in order to detect changes. When a change is detected, a mechanism to adapt the system to the change is triggered. For example, in the credit card system explained in Section 6.2.1, the accuracy of the predictive model being used could be monitored and updated based on new incoming training examples. When the accuracy suffers a significant drop, a change is detected [139]. The predictive model could then be deleted and a new model could be created to start learning the new situation of the environment. More advanced strategies could also incorporate mechanisms to accelerate the learning of the new environment condition rather than having to learn it from scratch [269]. Another example of approach using a change detection method is the multi-objective ensemble for online class imbalance learning [408]. Class imbalance learning refers to learning algorithms able to deal with classification problems where at least one class is under-represented in comparison to other classes (see Section 7.2.3.2). The approach presented by Wang et al. [408] uses an explicit method to detect whether a certain learning problem is a class imbalance learning problem and what classes should be currently considered as minority and majority classes. The advantage of using explicit change detection methods is that a system can be designed to swiftly react to a change once it is detected (e.g., by deleting old models as in the example above). The disadvantage of using change detection methods is that they may trigger false positive drift detections, which may hinder the system's performance if not carefully catered for. For more details on algorithms based on change detection, please refer to Section 7.2.3.3.

No change detection method: in this case, mechanisms to adapt the system to changes are continuously active, without the need for being triggered by explicit change detection methods. For example, an ensemble of learning algorithms can be used to create a credit card system such as the one explained in Section 6.2.1. This ensemble could maintain different models and assign a weight to each model based on its accuracy (i.e., a weight can be used to monitor the suitability of each model). This weight could be updated with new training examples by using a decay function that would give higher emphasis to newer examples [224]. When the system performs wrong predictions, new models could be created and added to the ensemble. Therefore, if some change has happened and is the cause for the wrong predictions, then the new models may be able to learn the new situation from scratch. When the weight of a given model is below a certain threshold, this model could be deleted for not representing the current environment well [224].

Another example of mechanism that does not use an explicit change detection is the mechanism used by the smart camera system in Esterle et al.' work [124]. Cameras exchanging an object to be tracked should not be communicating with other cameras that are too far away, in order to reduce the communication overhead of the system. By keeping track of the exchanged objects, each individual camera is

able to learn about its neighbouring cameras. Inspired by the ant foraging mechanism, the cameras use artificial pheromones to depict a graph of their local neighbourhood. With every exchanged object, additional pheromones are deposit on the respective link. However, if there are no exchanges of objects between two cameras the pheromone evaporates, allowing the system to overcome changes in topology or cameras' field of view over time.

The advantage of not using an explicit drift detection method is that it is not necessary to decide when exactly a change occurred. Deciding when exactly a change detection should be triggered may be particularly difficult when changes are slow and take some time to complete. The disadvantage is that the lack of a change detection method may in some cases lead to a difficult trade-off between stability and plasticity. If the system is set to forget old knowledge quickly, it may be too unstable to perform well. If the system is set to forget old knowledge slowly, it may take too long to adapt to changes. For more details on algorithms without change detection, please refer to Section 7.2.3.3.

6.3 Robustness

As described in Section 6.2, adaptivity is a property necessary for a computing system to be able to *react* to changes. In contrast, robustness is necessary in order to address changes in a *proactive* way, so that mechanisms can be adopted beforehand to reduce negative effects that future events could incur to the system. The development of robust systems involves making suitable knowledge representation and modelling choices such as how to model anticipation mechanisms and how to maintain different solutions so that they can be quickly recovered if necessary. Section 6.3.1 further explains and exemplifies robustness, and Section 6.3.2 explains the knowledge representation and modelling choices related to robustness.

6.3.1 Definitions and Examples

We define robustness as every kind of proactive behaviour with the aim of avoiding or diminishing the detrimental effects caused by changes, dynamic events or certain choices. This aspect of self-aware and self-expressive systems is important in many different systems, for instance in the work of Nymoen et al. [297], where multiple nodes strive towards the common goal of being in synchrony in order to engage in the creation and / or playback of collaborative music as briefly described in Section 6.2.1. In order to achieve this goal, the phase and the frequency of the oscillators of the participating nodes need to be adjusted accordingly. Adaptivity is achieved by each node reacting to short sounds emitted by other nodes, reflecting their timing / state. It is however not clear how this adaptation should be performed in an optimal way. If the adaptation is too slow, it will not reach the goal of synchrony in time.

If the adaptation is too fast, multiple nodes will try to achieve synchrony at the same time and will not find a stable state. In order to make the system more robust, each node can vary its degree of self-adaptation based on its anticipated value (self-confidence) of being in synchrony.

A second example is the work of Nebehay et al. [283], which was performed in the context of visual object tracking. We present this work more in-depth here, because it implements robustness in different ways, anticipating not only certain changes, but also the effect of certain choices. The goal of visual object tracking is to follow one or more objects of interest in a video stream as long as possible. However, several different types of change can (and frequently do) occur, making tracking difficult. For example, there may be changes in illumination, object or camera pose, as well as substantial changes to the object of interest itself. Achieving robustness to such changes is thus arguably one of the most desirable properties of visual object tracking systems.

Nebehay et al. [283] propose the keypoint-based method Consensus-based Matching and Tracking (CMT) for long-term object tracking, where the idea of robustness takes a central role. Here, a model of the object being tracked consists of multiple keypoints on the object of interest, which are nothing more than individual nodes working towards the common goal of tracking an object. In each new frame of the video sequence, each keypoint in the object model aims at finding its correct location on the object in the image, as defined by two measures of visual similarity [236, 252] that correspond to both matching and tracking of keypoints.

In fact one single keypoint would be sufficient to localize the object of interest. As visual information is highly ambiguous, it is however rather unlikely that each keypoint will position itself correctly on the object of interest. This is caused for instance by similar objects appearing in the background or by changes on the object that disallow a correct re-identification. Inevitably, some keypoints will end up on wrong parts of the object or even on different objects. In CMT, it is anticipated that these kinds of errors will occur and *redundancy* is introduced to address them as a form of robustness. Instead of creating a single hypothesis, each keypoint provides its own hypothesis for locating the object of interest by voting for its center, as shown in Figure 6.1. These votes are combined robustly by means of clustering them directly in the image space. A basic assumption here is that the relative majority of the keypoints is able to correctly identify the object center. In Figure 6.1, the majority cluster is shown in blue, while all minor clusters are shown in red, representing keypoints that failed to establish the correct position on the object of interest. By removing all keypoints in minority clusters, a much better object tracking result can be obtained, for instance by averaging all votes in the majority cluster as an estimate for the object center.

An interesting aspect about CMT is that the object model is never updated, which is in stark contrast to the prevalent paradigm in object tracking, where the object model is updated continuously. The problem of updating the object model continuously is that it always bears the danger of introducing new errors. For instance, when the localization is not exact, an update might lead to the incorporation of background information. From the experimental evaluation presented in [283], it appears

that the robustness of the overall system in CMT is strong enough to make adaptation of individual nodes unnecessary. While beneficial to the tracking performance, one drawback of the proposed approach of achieving robustness lies in its relatively high computational cost, where the communication between the individual nodes contributes the largest share.

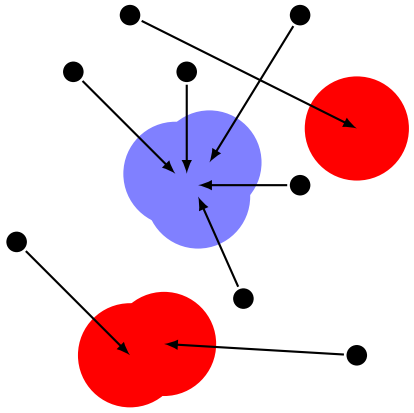

Fig. 6.1 Finding consensus in voting behaviour. Cast votes are clustered based on their proximity. The consensus cluster (shown in blue) is identified based on the highest number of votes. The minority clusters (shown in red) are not used for estimating the actual location of the object.

Another aspect in CMT that was born out of considerations for improving robustness is the question how voting is performed. As shown in Figure 6.2 on the left side, the voting vectors are initialised in the first frame, pointing to the object center. In later frames, these votes have to undergo a certain transformation in order to reflect changes in the location, pose and appearance of the object of interest. If this transformation is modelled with a degree of freedom as high as possible, this introduces a series of problems. First, it is very difficult to estimate the correct parameters for a complex transformation as the amount of training examples is highly limited. Second, the chances that multiple votes agree for one location get lower for more complex transformations. Third, overly complex transformations might further increase the computational load. In CMT, it is anticipated that the changes in the object of interest will be composed of translation, scaling and rotation, and the transformation of votes are restricted to these transformations, as shown on the right side of Figure 6.2. This class of transformations nicely balances the trade-off between robustness and expressiveness that is present in this kind of problem. In summary, robustness is achieved by a *restriction* of the output space. Obviously, one drawback of this kind of robustness is a reduction of the class of problems that can be handled, as already for instance perspective transformations are out of scope.

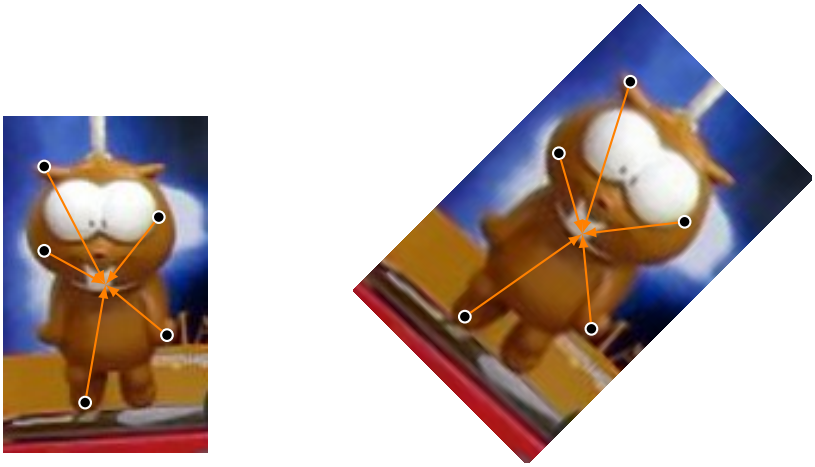

Fig. 6.2 Left: In CMT, voting vectors of keypoints are initialised in the first frame. Right: In frame t, each keypoint casts a vote for the object center. The object transformation is anticipated to be composed of translation, scaling and rotation.

6.3.2 Implications

6.3.2.1 Choice of Levels of Self-Awareness

Stimulus-awareness: stimulus-awareness can be used to obtain data to learn how to behave pro-actively. However, stimulus-awareness on its own may not be enough to learn how to anticipate changes, because it would not enable the system to distinguish between past, present and future stimuli. It is worth noting that stimulus-aware systems do not necessarily present robustness, but could certainly benefit from mechanisms to achieve certain types of robustness. For instance, the functional correctness of a computing system is highly dependent on its input. Therefore, it is important to ensure the correct functioning of its direct connection to the environment. For example, it may be desirable to achieve robustness to failure in the sensors of the individual nodes that allow them to learn about and interact with their immediate environment. Section 6.3.2.2 discusses some strategies that could be used to achieve that.

Interaction-awareness: as shown in the object tracking examples explained in Section 6.3.1, interaction-awareness can be used to provide a great degree of robustness in cases where the correctness of the output of the individual nodes is not ensured. Another example is the multi-camera object tracking application that will be presented in Chapter 13, where the use of multiple nodes makes the system robust to failures of individual nodes, for instance when a camera loses connection to the network.

Time-awareness: time-awareness is an enabling component for designing robust self-aware computing systems. It allows the system to learn how to anticipate changes by making predictions about possible changes in the future based on how changes occurred in past events. For instance, if errors made in previous actions or decisions are recognised, then behaviour in the future can be adapted accordingly. An exemplar application in this respect can be found in Chapter 14, where time-awareness enables extracting a rhythmic pattern from users shaking input devices in the SoloJam application, a process that has to be robust to changing input patterns.

Goal-awareness: In general, similar to adaptivity, robustness will frequently be achieved with respect to certain goals, and being aware of such goals could thus be helpful. Moreover, a goal-aware system could assess the importance of different goals as an interesting way of establishing robustness to changes. For instance, a self-aware node monitoring its battery status might decide that now is the time to pause striving towards its current goal of high priority and rather find a way of recharging itself in order to avoid running out of battery in the future. Note that this way of reasoning is different from adaptivity – in this case, anticipation about one's own state and the consequences of (not performing) actions is of utmost importance. Goal-aware systems could also establish certain objectives to evaluate how well a system is likely to perform in the future and in the event of changes, rather than only evaluating its current suitability to the present environment [136]. It is clear that, even though achieving robustness through this level of self-awareness can be beneficial, it requires a substantial amount of knowledge about oneself and its immediate environment, which might be difficult to obtain for certain systems. In SoloJam (Chapter 14), goal-awareness is also present. The concrete goal of a node in this application is represented by the hamming distance between the leader's rhythmic patterns and its own pattern. A change in the activity of a user then directly has an influence on its goal.

Meta-self-awareness: achieving robustness on a meta-self-awareness level can be considered the holy grail of achieving robustness in a computing system. A self-aware computing node might reason based on the outcome of its self-awareness components to decide what kind of robustness is actually important, and then take action to establish it. On a positive side, the computing system might discover aspects of its task during run-time that have not been under consideration at design-time and that lack a sufficient degree of robustness. On the negative side, this kind of robustness is very difficult to achieve as it requires a very abstract understanding of the environment consequences of actions, goals and oneself. In the multi-camera object tracking application (see Chapter 13), one form of meta-self-awareness is achieved by dynamically deciding on the strategy for handing over objects from one camera to another camera. This increases the robustness of the system compared to a static strategy selection mechanism.

Similar to what has been discussed in Section 6.2.2.1, while the different levels of self-awareness may help a system to achieve better robustness, they also require

additional computation time, which can be particularly concerning in applications requiring near-real-time performance or in embedded devices where resources are very limited.

6.3.2.2 Robustness through Redundancy of Components

Redundancy of components can be a very useful way to achieve robustness in self-aware and self-expressive systems. In general, redundancy could be employed in two different ways:

Deployment of multiple identical components: this could be used, e.g., to achieve robustness to failure of a given type of component. For example, it might be reasonable to employ multiple *identical* sensors to account for malfunctioning hardware components. This way, correctness of the input is ensured even during changes to the proper functioning of individual system components. Another example are the multiple keypoints used by the object tracking system explained in Section 6.3.1 in order to achieve robustness to wrong votes given by single keypoints. In [410] a memetic algorithm is presented that finds a solution to the Redundancy Allocation Problem (RAP), aiming at optimally allocating redundancy to components under some resource constraints.

Deployment of multiple different components: In [357] a system is presented that improves the fault-tolerance of electronic circuits. Here, robustness is achieved by creating multiple circuits exhibiting different error patterns that are then combined into a single strong circuit. However, failure of a component to function well may be caused not only by an error in the component, but also by changes in the environment which may cause a given component to be unsuitable. For example, one type of sensor might work well during daytime (such as standard cameras), while other sensors work well only at night time (for instance infra-red cameras). It therefore makes sense to combine multiple types of sensors to account for changes in the environment. Another example is the use of a set of different models representing different environment conditions in anticipation of possible changes in the environment [269, 181]. Even though a certain model may not be appropriate for the current environment condition, it may be appropriate for future conditions.

When employing multiple components, there are also different choices in terms of how to use the outputs produced by these components. For example, one could use one or more of the following:

Combining nodes' outputs: the output of multiple nodes is combined into a single output. In this case, there is the question on how to combine the outputs. If one node has to be designated as a coordinating node, this introduces a potential single point of failure. This is related to the issue of decentralisation, further explained in Section 6.5. Alternatively, one may wish to have no central node. In this case, nodes can

either all contribute equally to the combined output, or they could be given different emphases, depending on how suitable they are to the environment.

Selecting nodes: erroneous nodes might be detected and actions might be taken to prevent these nodes from participating in future actions. Care has to be taken about the quality of the failure: a hardware error might be easy to recognize (for instance by measuring the component temperature), while it is difficult to assess the correctness of the high-level self-expression of individual nodes. For instance, in the multi-camera object tracking application (Chapter 13), a node is given full control when its confidence about having localized the object correctly is high enough, as it can be assumed that it is well adapted to its environment.

Node communication: multiple nodes might be used in parallel and communicate with each other to account for potential defects in hardware or software.

The positive side of redundancy of components is that it could enable robustness to certain events or changes. However, it is clear that employing multiple nodes can also lead to unwanted side-effects, such as increased power-consumption, increased processing time, and conflicting nodes data.

6.3.2.3 Learning How to Anticipate Events or Changes

Anticipation can either be incorporated in the system at design time (e.g., the object tracking example given in Section 6.3.1), or learnt during the system's lifetime. Learning models able to anticipate events or changes can provide a much higher level of autonomy to a self-aware and self-expressive system. However, the nature and availability of the data that can be used for learning must be considered. This issue is also relevant to adaptivity of models of the environment, and has been discussed in Sections 6.2.2.2 and 6.2.2.3.

6.4 Multi-Objectivity

Self-aware and self-expressive systems are built with one or more goals/objectives to be accomplished. These goals/objectives can be either explicit or implicitly implemented in the system. The process of finding a best possible solution given a set of objectives and a set of limitations is referred to as optimisation process. The traditional optimisation problem aims to minimise/maximise a specific objective (usually in the form of a function). This type of problem is called Single-objective Optimisation Problem (SOP). However, only optimizing one objective cannot satisfy actual demands in several cases, because most of the real-world optimisation problems need to achieve a balance among multiple objectives, which are frequently in conflict with each other. Hence, optimizing one objective with respect to an SOP often

results in unacceptable results in the other objectives. For example, when buying a new laptop, we want it as cheap as possible but with the best quality. Or, when scheduling tasks on a chip [61], we want to improve the performance while keeping the chip cool. Obviously these objectives conflict their counterparts. We may not be able to buy a laptop which is both cheap and powerful. Similarly, we may not be able to obtain very high performance when scheduling tasks at the same time as keeping the chip cool. SOPs cannot handle this type of balance well. In this case, we extend the SOPs to Multi-objective Optimisation Problems (MOPs). Instead of minimizing/maximizing one objective, MOPs aim to achieve a balance among several objectives. As mentioned above, it is difficult to achieve a perfect multi-objective solution to minimise/maximise every objective. Therefore, we need to use some other method to find a reasonable solution which satisfies the objectives at an acceptable level.

Besides being relevant for optimisation tasks, multi-objectivity can also be relevant for machine learning tasks. For example, when learning how to detect faults in some machinery, one may wish to minimise both false positive and false negative fault detections. The issue of multi-objectivity has been investigated more in depth in the optimisation than in the machine learning literature. This section will thus concentrate mainly on multi-objectivity in optimisation. However, works on multi-objectivity in machine learning can also exist [408, 270, 58].

Section 6.4.1 further explains multi-objectivity, whereas Section 6.4.2 explains the implications of multi-objectivity to knowledge representation and modelling choices.

6.4.1 Definition and Examples

The general multi-objective optimisation problem can be defined as:

$$\begin{array}{rc} \textit{minimise } \mathbf{F}(\mathbf{x}) = (F_1(\mathbf{x}), F_2(\mathbf{x}), ..., F_k(\mathbf{x}))^T \\ s.t. \qquad g_i(\mathbf{x}) = 0, i = 1, 2, ..., a \\ h_j(\mathbf{x}) < 0, j = 1, 2, ..., b \end{array} \tag{6.1}$$

where k is the number of objectives, a is the number of equality constraints, b is the number of inequality constraints, $\mathbf{x} \in X$ is a vector of decision variables, X is the search space (set of all candidate solutions), and $\mathbf{F(x)}$ is the vector of objective functions $F_i(\mathbf{x})$, $\mathbf{F(x)} : X \rightarrow R^k$. Objective functions can also be called goal functions, payoff functions, cost functions or fitness functions. Please note that functions to be maximised can be easily converted to functions to be minimised. Therefore, without loss of generality, the MOP can be defined as a minimisation problem as above.

In the example of buying new laptops, we have two conflicting objectives: lower price and higher performance. Consider that there are n alternative laptops $X = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}\}$ available for purchase, and that their costs are $c_1, c_2, ..., c_n$ and performances are $p_1, p_2, ..., p_n$. A solution $\mathbf{x} \in X$ to this problem is a vector of size one representing a single laptop to be purchased. The objectives of this MOP can

be described as: minimise c_i for lower prices and minimise $1/(p_k+\varepsilon)$ for higher performance, where ε is a small constant to avoid division by zero.

In the chip scheduling example, X represents all the possible schedules of tasks on the chip. If we have n cores and m tasks, then $|X| = C_m^n$. We can use a vector to stand for a given scheduling of tasks to cores: $\mathbf{x} = (u_1, u_2, ..., u_n)$, where $u_i \in (1,m) \cup \{-1\}$. If $u_i \in (1,m)$, then that means that core i runs task u_i; otherwise it means that nothing is running on core i. It is worth noting that other structures could be used to represent a solution for the same problem. For example, having a vector with $n+m$ entries instead of n could be used to allow certain tasks to be assigned to no core, while still enforcing all tasks to be mentioned at least once in the vector. This could be desirable, for example, to allow tasks that are unscheduled in two solutions to be scheduled in a new solution created by crossing-over these two solutions in evolutionary algorithms [61]. The objectives can be described as:

- Maximise the overall performance of chip:

$$F_1 = \min(-\sum_{j=1}^{m} \overline{p_j}(\mathbf{x}))$$

- The average temperature of chip:

$$F_2 = \min(\sum_{j=1}^{m} \overline{\theta_j}(\mathbf{x}))/m$$

where $\overline{p_j}(x)$ and $\overline{\theta_j}(x)$ are the performance and temperature of core j under scheduling $\mathbf{x}$, respectively.

As the objectives in a MOP are frequently conflicting, it is impossible to find a single solution that optimises all objectives simultaneously. Instead, we are usually interested in obtaining a set containing the best trade-off solutions, which are called Pareto optimal solutions. Pareto optimal solutions are solutions non-dominated by any other solution. The definition of a solution $\mathbf{x}$ being dominated by another solution $\mathbf{y}$ ($\mathbf{x} \prec \mathbf{y}$) is as follows:

$$\mathbf{x} \prec \mathbf{y} \iff \forall i, F_i(\mathbf{x}) \leq F_i(\mathbf{y}) \wedge \exists j, F_j(\mathbf{x}) < F_j(\mathbf{y}) \tag{6.2}$$

Given the set of Pareto optimal solutions (Pareto set) for a given MOP, the decision-maker would be able to select his/her desired solution with his/her preferred trade-off among all optimal solutions in the Pareto set.

In order to find the Pareto set, we need optimisation algorithms specifically designed for MOPs. Alternatively, it is not uncommon to convert MOPs to SOPs and then use existing SOP algorithms to solve them. For instance, most current research on temperature control of multi-cores converts this MOP to a SOP by using linear weight methods. Nevertheless, SOP algorithms aim at finding a single optimal solution. Converting MOPs to SOPs leads to some additional problems to be tackled, e.g.:

1. How to set the relative importance of different objectives

Given that SOPs will attempt to find a single optimal solution, approaches to convert MOP to SOP usually require some method to set the relative importance of the different objectives beforehand. The algorithm will then look for a single best solution considering this particular relative importance. However, it may not be straightforward to chose values to represent relative importance, and a wrong choice will lead the algorithm to miss useful trade-offs among the objectives. Moreover, in certain problems, the relative importance of different objectives should ideally not be constant, making it difficult to choose ideal values manually. This is the case, for example, of the chip task scheduling problem. When the current temperature of the chip is already too high, we should give higher importance for temperature in order to cool down the chip more quickly. When the current temperature is low, we do not need to pay too much notice on increasing temperature. So, the relative importance of these objectives can change with time.

2. How to normalize the objective functions
   The objective functions F_i, $1 \leq i \leq k$, are the values which will stand for the quality of the solution. However, these values can be incomparable. For instance, it is hard to say which unit is larger in the chip scheduling problem – one unit of performance or one unit of temperature. We need methods to normalize these values into comparable ones. For instance, we can normalize them by:

$$f_i(\mathbf{x}) = \frac{\max_{\mathbf{x_1} \in \mathbf{X}} \mathbf{F_i}(\mathbf{x_1}) - \mathbf{F_i}(\mathbf{x})}{\max_{\mathbf{x_1} \in \mathbf{X}} \mathbf{F_i}(\mathbf{x_1}) - \min_{\mathbf{x_2} \in \mathbf{X}} \mathbf{F_i}(\mathbf{x_2})}, \mathbf{1 \leq i \leq k}$$

   where k is the number of objective functions. Or we can normalize them by:

$$f_i(\mathbf{x}) = \frac{\mathbf{F_i}(\mathbf{x})}{\max_{\mathbf{x_1} \in \mathbf{X}} \mathbf{F_i}(\mathbf{x_1})}, \mathbf{1 \leq i \leq k}$$

   The two methods above both can normalize the objective values but will lead to different results, and it is unclear which of them to adopt.

Given the problems of converting MOPs to SOPs, methods specifically designed for solving MOPs are desirable. For more details on MOPs, we recommend Deb's book [90].

6.4.2 Implications

6.4.2.1 Choice of Levels of Self-Awareness

The issue of multi-objectivity can be relevant to systems with any level of self-awareness. This is because any self-aware system will be designed with a purpose in mind. This purpose can only be achieved if there is some "force" pushing the system towards eventually achieving this goal. In systems that are not goal-aware,

this "force" can be viewed as one or more implicit goals. If the system is goal-aware, then the goal(s) will be explicit in such a way that the system can reason about it(them). Therefore, multi-objectivity could be tackled by any level of self-awareness.

Stimulus-awareness: the ability to perceive stimuli is essential to evaluate how well a system performs in a given environment, or to decide which actions to take so as to perform well in a given environment. Stimuli can be used directly or indirectly to compute objective functions, no matter if these functions are explicit or implicit.

Interaction-awareness: systems to deal with multiple objectives do not necessarily need to be interaction-aware. However, interaction among nodes could be beneficial if different nodes are responsible for different objectives. Interaction-awareness could also be used by a node to acquire information for evaluating its objectives. For instance, each node in the smart camera system [124] explained in Section 7.4.1 uses payments made to and received from other cameras in order to compute its utility function, as mentioned in Section 6.2.1.

Time-awareness: given that this type of self-awareness allows a system to distinguish between past, current and future events, it can help to compute different objectives for reflecting the suitability of the system to situations encountered in the past, to the current situation, and to possible future situations. The multi-objective ensemble approach presented by Wang et al. [408] makes use of time-awareness by computing each objective based on a time-decayed function, reflecting the suitability of the system to the current situation of the environment.

Goal-awareness: this type of self-awareness is the most relevant for multi-objectivity. Goal-awareness would allow a system to explicitly consider its objectives and reason about them. This could facilitate dealing with multiple objectives by making it easier to consider different trade-offs among objectives, and by allowing objectives to be added/removed over time. Therefore, goal-awareness could make it easier to design a system to cope with multiple conflicting objectives. An example of system that benefits from goal-awareness to deal with multiple objectives is the multi-objective ensemble approach for class imbalance learning presented by Wang et al. [408]. The system is designed to perform classification tasks and its learning procedure explicitly considers classification performance in terms of recall on each existing class separately.

Meta-self-awareness: meta-self-awareness is not essential for a system to be able to deal with multi-objectivity. However, multiple objectives could be designed in order to represent how beneficial different levels of self-awareness are in order to develop a meta-self-aware system.

Similar to what has been discussed in Section 6.2.2.1, while the different levels of self-awareness may help a system to better handle multi-objectivity, they also

require additional computation time, which can be particularly concerning in applications requiring near-real-time performance or in embedded devices where resources are very limited. These issues must be considered when deciding what level of self-awareness to adopt.

6.4.2.2 Searching for Solutions with a Particular Trade-Off Among Objectives

Weighted methods: Classical methods for solving MOPs usually convert MOPs into SOPs, and then use a single objective optimisation algorithm to find a best solution. The simplest method to do that is to multiply objectives by a manually set weight and then sum them up [428, 225, 17, 255]. Therefore, the MOP is redefined as a SOP as follows:

$$\begin{aligned} &minimise\ F(x) = \textstyle\sum_{i=1}^{k} w_i F_i(\mathbf{x}) \\ &s.t. \quad g_i(\mathbf{x}) = 0, i = 1,2,...,a \\ &\qquad h_j(\mathbf{x}) < 0, j = 1,2,...,b \end{aligned} \tag{6.3}$$

Another method is to use the weighted distance between the value of each objective for a given solution $\mathbf{x}$ to the best achievable value $\mathbf{z}^*$:

$$\begin{aligned} &minimise\ F(x) = (\textstyle\sum_{i=1}^{k} w_i(|F_i(\mathbf{x}) - z_i^*|^p))^{1/p} \\ &s.t. \quad g_i(\mathbf{x}) = 0, i = 1,2,...,a \\ &\qquad h_j(\mathbf{x}) < 0, j = 1,2,...,b \end{aligned} \tag{6.4}$$

where p can be any value from 1 to ∞. This method will be the same as the previous one when $p = 1$.

A similar method [111] uses the objective values $\mathbf{z}^0$ of a feasible non-Pareto-optimal solution rather than the ideal $\mathbf{z}^*$:

$$\begin{aligned} &maximise\ F(x) = (\textstyle\sum_{i=1}^{k} w_i(|F_i(\mathbf{x}) - z_i^0|^p))^{1/p} \\ &s.t. \quad z_i^0 < F_i(\mathbf{x}), i = 1,2,...,k \\ &\qquad g_j(\mathbf{x}) = 0, j = 1,2,...,a \\ &\qquad h_l(\mathbf{x}) < 0, l = 1,2,...,b \end{aligned} \tag{6.5}$$

Likewise, we can also multiply all the objectives instead of adding them up, as follows:

$$\begin{aligned} &minimise\ F(x) = \textstyle\prod_{i=1}^{k} (F_i(\mathbf{x}))^{w_i} \\ &s.t. \quad g_i(\mathbf{x}) = 0, i = 1,2,...,a \\ &\qquad h_j(\mathbf{x}) < 0, j = 1,2,...,b \end{aligned} \tag{6.6}$$

The advantage for this type of method is that, once the weights are set, SOP algorithms can be used to find a solution with the trade-off represented by these weights. However, finding a suitable weight vector may not be easy, especially when the number of objectives is larger than three. The weight vector will strongly affect the results obtained by the SOP algorithms.

Interactive methods: The main idea of these methods is to guide the search direction according to the information provided by the decision-maker during the runtime of the optimisation algorithm, rather than having all information on preferences set beforehand. The main process is as follows:

1. Initialise the algorithm and generate a starting point for the search.
2. Ask the decision-maker for preference information, e.g., desirable objective function values, or number of new solutions to be generated.
3. Generate new solutions according to the preference and show them to the decision-maker.
4. Ask the decision-maker to select the best solution so far and adjust the preference according to the selection.
5. Stop if decision-maker wishes to; otherwise return to step 2.

Popular methods include Interactive Tchebycheff Metric approach[374], NIMBUS[267] and Guess Methods[46]. The advantage of interactive methods is that they require less prior information than the weighted methods. However, they may cause user fatigue, as the decision-maker may need to provide inputs several times throughout the optimisation process.

6.4.2.3 Searching for a Set of Solutions with Different Trade-Offs

There are different frameworks for optimisation algorithms specifically designed to search for the Pareto optimal set of solutions for MOPs [433]. One of the main choices in terms of knowledge representation and modelling in these frameworks is how to model the relative quality of the solutions. This section will briefly explain some different strategies that can be used to compare different solutions in MOP algorithms.

Pareto dominance-based comparisons: most optimisation algorithms for solving MOPs are evolutionary algorithms based directly on the concept of Pareto dominance explained in Section 6.4.1 [371, 179, 437, 436, 91]. These algorithms consider that a solution that dominates another solution is better than this other solution. They also frequently use some strategy to maintain the diversity of the search, given that concentrating on dominance on its own could lead to lack of diversity and thus early convergence to local optima. For instance, mechanisms such as fitness sharing and crowding distance have been proposed [179, 436, 91] to prioritise solutions that will lead to higher diversity. A drawback of these algorithms is that they may struggle to cope with large numbers of objective functions (e.g., larger than three).

Indicator-based comparisons: another way to compare solutions is to use a scalar quality indicator/metric to compare solutions [110, 27, 45]. The indicator can be used to compare pairs of solutions [110]. However, a very popular approach is to use scalar quality indicators to evaluate sets of solutions to guide the search. For instance, several algorithms based on the hypervolume have been proposed

[33, 45, 21]. These algorithms can perform better than algorithms such as the ones based on Pareto dominance especially when the number of objectives is large [21, 33]. Depending on how the indicator is defined, it is also possible to avoid the need for separate mechanisms to preserve diversity. However, care must be taken when choosing an indicator or designing an algorithm to use indicators, as some indicators can be expensive to compute [33]. Section 6.4.2.4 further explains indicators/metrics.

MOP decomposition: the strategy adopted by the multi-objective evolutionary algorithms based on decomposition (MOEA/D) [433] is to decompose the MOP into a number of SOPs, where each SOP is a weighted aggregation of the individual objectives. A neighbourhood between SOPs is modelled based on the distances between their weight vectors [433] or some other neighbourhood structure [246]. Each SOP is then optimised simultaneously by using information from its neighbours. The advantage of MOP decomposition is that solutions for a given SOP can be compared against each other based on the scalar corresponding to their single objective, eliminating the need to decide how to compare solutions based on multiple objectives. However, care must be taken to design effective neighbourhood structures.

Algorithms for MOPs usually work efficiently when dealing with two and three-objectives. When the number of objectives increases, i.e., when we have many-objective problems, Pareto dominance-based algorithms' performance deteriorates [217]. One of the possible reasons is that nearly all solutions are non-dominated when the number of objectives increases. For instance, Ishibuchi et al. [196] showed that 200 random solutions can be all non-dominated when the number of objectives is over 16. Indicator-based algorithms take indicator function as the only objective, avoiding this problem. However, calculating indicators such as hypervolume is usually a significantly time-consuming process. For methods based on MOP decomposition, it is difficult to decide on a good balance among all the objectives when dealing with many-objective problems. Currently, there is plenty of works trying to solve the problems caused by many-objectives. For example, Narukawa et al. [280] incorporated the preference of a decision maker with a dominance-based algorithm, Ishibuchi et al. [195] proposed an iterative indicator-based approach which attempts to decrease the cost of computing the indicator, and Wang et al. [244] proposed an improved version of the Two-Archive Algorithm to try to incorporate a ranking mechanism for updating the convergence of the archive.

It is worth noting that it is possible to hybridise algorithms in order to combine their advantages. For instance, the memetic multi-objective evolutionary algorithms proposed by Ishibuchi and Murata [194] randomly draw a scalar function to evaluate solutions in the evolutionary search process and then use a local search method to further improve solutions.

6.4.2.4 Metrics for Comparing Sets of Solutions

Algorithms for SOPs generally aim at finding a single best solution to the problem. However, as explained in Section 6.4.1, we are frequently interested in finding the Pareto set. The Pareto set is frequently too large and it could be even infeasible to find the whole set. As a result, many algorithms will find a set of non-dominated solutions expected to be a good approximation of the Pareto set. Therefore, we may need to compare sets of non-dominated solutions, specially when comparing the effectiveness of different algorithms. There are mainly two types of quality metrics:

Unary quality metrics: unary quality metrics give a quality value to each set of solutions. The advantage of such measures is that they provide a scalar indication of the quality of a set of solutions with respect to a reference point, where the reference point should ideally be the quality of the best or worst possible solution. However, the disadvantage of this type of metric is that it is not always possible in real world applications to determine the ideal reference point. Moreover, these measures need to consider not only the quality of the solutions in terms of convergence with respect to the reference point, but also how diverse the solutions in the set are. This is because it is desirable to have a diverse set so that the decision-maker can choose his/her desired trade-off among objectives. However, it has been argued that both aspects of convergence and diversity cannot be properly measured by a single unary metric [438]. Examples of unary metrics are [284]:

- Generational distance – this measure represents how far a set of solutions is from the Pareto set on the objective space:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{6.7}$$

  where n is is the number of solutions in the set of non-dominated solutions found and d_i is the Euclidean distance (measured in objective space) between each of these solutions and the nearest member of the Pareto set. It is clear that GD = 0 indicates that all the generated solutions are in the Pareto set. In order to get reliable results, objective values are normalized before calculating this metric.
- Hypervolume – this measure calculates the volume (in the objective space) covered by the solutions in the non-dominated set M being evaluated with respect to a reference point x_{ref} representing the worst possible point. This is done by calculating the volume Λ of the union of the hypercubes a_i defined by each non-dominated solution $m_i \in M$ and the reference point x_{ref}:

$$S(M) = \Lambda(\{\cup_i a_i | m_i \in M\}) = \Lambda(\{x | m \prec x \prec x_{ref}\}) \tag{6.8}$$

  Larger hypervolume values represent better values. As this measure is also dependent on the scale of the objectives, objectives should be normalised before computing it.

Binary quality metrics: binary quality measures assign a quality value not to a single set of solutions, but to the relation between two sets. It can overcome the drawbacks of unary quality measures mentioned above, as they do not consider convergence and do not need an ideal reference point to be set. However, its disadvantage is obvious: for k solutions, we have $k(k-1)$ values for every pair of solutions, while in unary quality measures, there are only k values. This makes it harder to analyse the results. Examples of metrics are:

- Coverage of two sets [435] – given sets of solutions A and B, this measure is defined as:

$$CS(A,B) = \frac{|\{\mathbf{b} \in B | \exists \mathbf{a} \in A, \mathbf{a} \succeq \mathbf{b}\}|}{|B|} \tag{6.9}$$

  The value $C(A,B) = 1$ means that all solutions in B are dominated by or equal to solutions in A. The value $C(A,B) = 0$ means that none of the solutions in B are covered by the set A. Both $C(A,B)$ and $C(B,A)$ should be considered, as $C(B,A)$ is not necessarily the same as $1 - C(A,B)$.
- I_ε metric [438] – in order to define this metric, we first need to define a new relation called ε dominance ($\prec_\varepsilon$). Given two solutions $\mathbf{a}$ and $\mathbf{b}$, this relation is defined as follows:

$$\mathbf{a} \prec_\varepsilon \mathbf{b} \iff F_i(\mathbf{a}) \leq \varepsilon F_i(\mathbf{b}), i \in 1, ..., k \tag{6.10}$$

  The metric I_ε for two sets of solutions A and B is then defined as:

$$I_\varepsilon(A,B) = \inf_{\varepsilon \in R} \{\forall \mathbf{b} \in B \; \exists \mathbf{a} \in A : \mathbf{a} \prec_\varepsilon \mathbf{b}\} \tag{6.11}$$

  Therefore, $I_\varepsilon(A,B)$ equals the minimum factor ε such that any objective vector in B is ε-dominated by at least one objective vector in A. In the single-objective case, $I_\varepsilon(A,B)$ is simply the ratio between the two objective values represented by A and B.

6.5 Decentralisation

Self-aware and self-expressive applications are frequently composed of multiple agents without a central unit to control them. While this might not be true for all applications, decentralised approaches generally have various advantages over centralised applications. For instance, distributed agents may interact with each other to acquire knowledge and perform certain tasks. As each agent performs its task independently, tasks can be processed concurrently. Due to this autonomy of each agent, the robustness of the entire system is increased. While acquired knowledge can be exchanged among the independent agents, it is not mandatory. This reduces the communication overhead in comparison to centralised systems. Section 6.5.1 explains what is meant by decentralisation and provides examples of applications using decentralisation, and Section 6.5.2 explains the implications of decentralisa-

tion in terms of the capability primitives introduced in Section 5.3.2, knowledge representation and modelling choices.

6.5.1 Definitions and Examples

In a centralised computing system, all information is collected at and all resources are coordinated by a central node. While central / networked nodes may be able to make their own computations, only the central node will be able to make the decisions for the entire network. Furthermore, besides being responsible for making decisions for the entire network, this central node is usually also responsible for coordinating tasks among all participating nodes. A typical example of centralised systems are Automated Teller Machines (ATMs) where the ATM receives information about the clients' current balance from a central component. The key fact here is that the relevant information about the current balance is not defined by the individual teller machine, but provided by the central component. There are various advantages and disadvantages of a centralised system.

- **Advantages:**
  - \+ *Coordination:* all information is gathered at a single entity. This single computing node only submits processed information to the requesting components. A coordination of which node does what at what time is not necessary.
  - \+ *Maintenance:* as there is only a single entity responsible for performing the different tasks, there is only one single entity to maintained. Also, localisation of problems is much easier in a centralised system.
- **Disadvantages:**
  - – *Bottlenecks:* in a large system, the centralised node might receive a lot of different requests simultaneously. This creates bottlenecks on two different fronts: the processor and the network. Having the different entities sending requests for processing their raw data to the server might lead to congestion when it comes to the server receiving this data. Furthermore, the server might be overwhelmed by the amount of work. Finally, storing the raw data on site for further processing might become problematic.
  - – *Single point of failures:* even though a single centralised node responsible for processing all data can have advantages when it comes to maintenance and localisation of errors, it is highly disadvantageous when it comes to robustness, reliability and availability. If the central node fails, the other nodes in the system are paralysed and cannot perform any further operations.

In comparison to centralised computing systems, decentralised or distributed computing systems compile a set of autonomous computers. In the absence of a central node, the computers in the network have to coordinate their required tasks autonomously. This means that each node in a decentralized computing system has

to be able to make its own decisions. These decisions can be based on its very own information, on interactions with or on information from other nodes. While in such a distributed approach, the autonomous computing nodes share their information about their own state, desired goals and required tasks among each other, a complete exchange of information might be infeasible. Especially larger systems containing nodes which change their state frequently will inevitably congest their own network by only transmitting such updates. Hence, a single machine has virtually no chance of having complete information about the state of all nodes in the network. This also means that having a single machine failing does not bring the entire system to a halt. In a centralised system, if the central server crashes the other nodes of the system will not be able to perform any further tasks. An example in the area of decentralised systems is the telephone network or the well known Peer-to-Peer networks, where upon connection of two or more participants no central component is required for exchanging information between the participants. The advantages and disadvantages of decentralised systems are as follows:

- **Advantages:**
  - \+ *Concurrency:* a single node is not responsible for processing all tasks. This allows the system to distribute the workload and process multiple tasks concurrently on different nodes.
  - \+ *Reliability:* having multiple entities capable of performing the same tasks makes failing of single entities have less impact than in centralised systems. While failing of single entities might slow down the system or require it to repeat tasks, it does not paralyse it completely. This is related to robustness by redundancy of components explained in Section 6.3.2.2.
  - \+ *Scalability:* in a decentralised system, each node is independent. New nodes can be added to the system without need for them to be registered at a central component. This allows new nodes to join at any time without prior notice to prepare the system. Additionally, new nodes will be integrated autonomously and they can take over tasks immediately.
  - \+ *Shared resources:* resources such as processing power or memory can be shared among the different nodes in the network. This allows nodes with less hardware capabilities to transfer unneeded data and workload to other nodes with sufficient resources available.
- **Disadvantages:**
  - – *Coordination:* in order to ensure tasks are not unnecessarily performed twice, the system has to coordinate tasks among the nodes. This coordination requires additional effort from the system.
  - – *Security:* having multiple nodes interacting with each other opens up various security risks. In sensitive applications, transmissions between the different nodes have to be protected, requiring a secure network communication. Furthermore, malicious entities have to be detected and neutralised in order to protect the system's original purpose.

- *Error localisation:* in case of failures in the system, the error has to be localised among all participating nodes. This can be very hard especially when interdependent tasks are distributed among multiple nodes.
- *Replicated data:* to allow for reliability but also robustness of a decentralised system, often data has to be replicated multiple times on different nodes. This replication consumes unnecessary resources from a system-level point of view, but allows for concurrent processing of information, increased reliability and robustness.

In the following, two examples of self-aware and self-expressive systems that can benefit from decentralisation will be explained. The first one is a multi camera system, where a set of deployed cameras pursue a common goal. This goal could be tracking of objects, as in the example given in Section 6.2.1, or monitoring a specific area or detecting and identifying certain behaviour. Using standard cameras requires a centralized approach, where all cameras send their information to a central server for further analysis. Not only will the large amount of video data congest the network, but also video analysis is a resource intensive task. A central server will require a lot of computational power in order to deal with the video streams of multiple cameras consecutively and in near real-time. Alternatively, standard cameras could be replaced with the so-called smart cameras [343, 338]. Smart cameras combine the image sensor with a processing unit and a network interface. Here, the data can be preprocessed and only relevant or even aggregated data is transmitted to a central control. This reduces the workload for the central server tremendously and distributes the work among the nodes of the network. However, in such a scenario, the server still collects and coordinates information and assigns tasks to the cameras as necessary. While this approach is in general much more scalable than a completely centralized approach, in the case of a server failure, the system is still halted. Only if the decision making capability is given to the individual cameras a central coordination can be omitted and a completely decentralised system developed. In this case, each camera still may have to collect various information from other cameras. However, the whole system becomes more robust against camera failures, i.e., if a camera fails, the entire system will not be paralysed [123].

The second example of self-aware and self-expressive system that can benefit from decentralisation is the hypermusic system. Hypermusic allows non-musicians to participate in a music oriented activity using hand-held devices (e.g. a mobile phone) for interaction. A synchronisation mechanism that can be used in this system has been briefly discussed in Sections 6.2.1 and 6.3.1, but the system itself will be briefly discussed in this section from the perspective of decentralisation. The self-aware and self-expressive Hypermusic system recognises the interactions of the person with a device and interprets it as music. In a group, multiple persons can join together in a bigger ensemble to create music. A central server would be able to collect all information from the different nodes and synthesise music, but might face limitations on the network as well as with respect to its own processing power. Simple coordination based on data aggregated by a server poses a risk to robustness in case the server fails. Only when decisions are being made by the individual nodes

and music generated by the collective behaviour of the nodes, workload can be distributed accordingly among the network and failure of individual nodes does not stop the system from producing music.

6.5.2 Implications

When designing a decentralised or distributed system, a system designer has to make various choices. In this section we focus on important choices to be made for self-aware and self-expressive systems, focusing on choices of levels of self-awareness and structures to be used.

6.5.2.1 Choice of Levels of Self-Awareness

Stimulus-awareness: the individual nodes of a stimulus-aware, decentralised system are only able to interact with each other by reacting to stimuli and triggering new external stimuli in return. An excellent example in nature is the frequency synchronisation of fireflies. Only by perceiving the stimulus of the blink of a light, unaware of its origin or exact time of blink, the firefly reacts by blinking itself. An example in the context of self-aware self-expressive systems is the synchronisation mechanism [297] that can be used in systems such as Hypermusic. It uses a stimulus-aware approach in order to synchronise musical patches, as explained in Section 6.2.1. Section 7.3.1.3 gives a more detailed explanation of a computational self-aware approach for frequency synchronisation.

Interaction-awareness: typical decentralised systems are expected to implement inter-action-aware behaviour. Here the individual nodes can not only distinguish between the different stimuli but are also able to differentiate between the same stimuli from different nodes. Furthermore, the individual nodes can select a specific subset of nodes in the system for interaction. This allows them to not only improve the performance of the entire system, but also make collective decisions and exploit local behaviour. For example, the smart camera system [124] uses an economy-inspired interaction-aware approach in order to coordinate tracking responsibilities.

Time-awareness: time-awareness allows a system not only to adapt to and anticipate changes in the environment as discussed in Sections 6.2 and 6.3, but also in the behaviour of other nodes. Additionally, the system is enabled to 'forget' previously learnt information which has become obsolete. An example for such time-awareness is the pheromone-based foraging process of ants. Pheromones are strengthened while the food source lasts, but evaporates over time, when the food source is depleted and ants do not deploy pheromones anymore. Artificial pheromones can be seen as a way for nodes in decentralised systems to interact with each other in a time-aware fashion. Two different computational time-aware approaches employ-

ing artificial pheromones are used in the smart camera system and the Hypermusic system. They are discussed in Section 7.4.1.1 and Section 14.3.1

Goal-awareness: in the absence of a central component, goals are defined and represented by the individual nodes and are based on their individual capabilities. This abstraction enables the system to deal with heterogeneous nodes and scale better. Additionally, each node is focused on its own capabilities and goals, allowing the system to distribute load more equally among all nodes and execute tasks concurrently. In the smart camera system the individual cameras are aware about their own goals without having a central control steering them.

Meta-self-awareness: while meta-self-awareness is not essential to a decentralised system, it allows the system to improve its own performance based on the environment it has been deployed in. Additionally, each individual node can reason about its own behaviour in the dynamic environment and the actions and reactions of its neighbouring nodes. An example of computational meta-self-awareness in the smart camera system is given in Section 7.4.1.3 and by Lewis et. al [242], where each node learns about the performance of a set of actions and over time selects the best for its given situation.

Similar to what has been discussed in Section 6.2.2.1, while the different levels of self-awareness may help a system to better handle decentralisation, they also require additional computation time, which can be particularly concerning in applications requiring near-real-time performance or in embedded devices where resources are very limited.

6.5.2.2 Choice of Neighbourhood

In a distributed system, each node is able to build up its social neighbourhood, i.e., those nodes it is interacting with during its lifetime. Essentially, a designer of a self-aware and self-expressive system has to make a choice regarding the neighbourhood to be limited or unbound in terms of the number of neighbours.

Limited neighbourhood: in a limited neighbourhood, each node is only able to interact with a limited number of other nodes from the network. This limit can either be fixed or relative to the system size, but is defined by a system designer before deployment. A limit to the neighbourhood size allows the system designer to control the resource consumption to some extent. On the other hand, it also limits the possible capabilities of the system when it comes to requesting new resources or processing capabilities from other nodes.

Unbound neighbourhood: in contrast to a limited neighbourhood, in an unbound neighbourhood the node itself decides the number of neighbouring nodes it wants to interact with. An unbound neighbourhood allows each individual node to adapt

its own neighbourhood during runtime by adding new nodes or removing unneeded ones. This allows the individual node to access more resources through other nodes or reduce its communication overhead by removing nodes from its network. Nevertheless, even unbound neighbourhoods are limited by the communication and sensing capabilities of the respective node. For example, a node will not be able to communicate with all nodes in a wide-area network with single-hop communication over a wireless network interface (e.g. in the hypermusic system). In the smart camera system, the field of view of the individual cameras naturally limits the size of the neighbourhood.

6.5.2.3 Choice of Information Accessibility

In a distributed system, resources as well as information is shared among all nodes in the network. This means a node can access resources and information locally or remotely. Local information is based on the experiences a node makes by itself. In contrast, with remote information nodes explicitly exchange knowledge about their own experiences (e.g. sensor data or available resources). While accessing remote resources allows to distribute workload, information from remote nodes may have benefits as well drawbacks for a node in comparison to using only local information. We will briefly discuss the benefits and drawbacks of limiting the access of a node.

Local information: a node only relies on its very own experiments made. Only local information is used in order to make decisions and continue towards a global, system-wide goal. While this may limit the capabilities of the entire network in terms of possible performance, nodes may not rely on possibly incorrect information from other nodes. Additionally, it avoids nodes exchanging irrelevant information among each other, which would otherwise increase the network traffic unnecessarily. For example, the smart camera networks, where each node only values objects it is able to 'see', are based on local information. This means a camera ignores objects in neighbouring fields of view, even if they might enter its own field of view in the near future.

Remote information: in contrast to limiting a node to only local information, remote information enables each entity to draw from experiences other nodes have already made. This allows faster transition of knowledge but requires clear structures on how knowledge is represented. If facts in a given situation are omitted, a seemingly similar situation for another node might result in worse performance when drawing from another node's experiences. Taking the smart camera system as an example, when one camera omits its available resources when representing their knowledge and experience, another node might use a similar behaviour but may perform worse if it has less resources at its disposal.

Mixed information: while information from remote nodes might be insufficient to be applied at a given node, they can also be very helpful. In a mixed approach each

node could individually learn how much remote information it wants in addition to local information. Furthermore, nodes could even learn which remote node provided what kind of information to what degree of usefulness to them. This would allow them to distinguish between information from different nodes, filter noise from useful information and learn faster in their environment. Various learning methods will be discussed in the following Chapter 7.

6.6 Summary

This chapter discussed different features that may be present in self-aware and self-expressive system from the perspective of knowledge representation and modelling. The features discussed were adaptivity, robustness, multi-objectivity and decentralisation. A common theme in the discussion of these features was that different levels of self-awareness can be helpful for successfully implementing these features. However, trade-offs in terms of the benefit that such levels of self-awareness can provide and computation time must be carefully considered, especially in systems that must operate in near real-time or in embedded systems. Besides the choice of the level of self-awareness, several other knowledge representation and modelling choices must also be made to decide how exactly each of these levels of self-awareness should be implemented. These choices include, but are not limited to, how and whether to model states and environments, model time, deal with changing environments, use redundancy of components, anticipate changes, formulate multiple objectives, compare solutions with different objectives, implement nodes' neighbourhoods, and exchange information among nodes. Different options must be carefully considered when developing a self-aware and self-expressive systems. The best choice will depend on the application to be developed and the type of environment where it will be embedded.

References

1. Aberdeen, D., Baxter, J.: Emmerald: a fast matrix-matrix multiply using intel's SSE instructions. Concurrency and Computation: Practice and Experience **13**(2), 103–119 (2001)
2. Abramowitz, M., Stegun, I.: Handbook of Mathematical Functions. Dover Publications (1965)
3. Agarwal, A., Harrod, B.: Organic computing. Tech. Rep. White paper, MIT and DARPA (2006)
4. Agarwal, A., Miller, J., Eastep, J., Wentziaff, D., Kasture, H.: Self-aware computing. Tech. Rep. AFRL-RI-RS-TR-2009-161, MIT (2009)
5. Agne, A., Platzner, M., Lübbers, E.: Memory virtualization for multithreaded reconfigurable hardware. In: Proc. Int. Conf. on Field Programmable Logic and Applications (FPL), pp. 185–188. IEEE Computer Society (2011). DOI 10.1109/FPL.2011.42
6. Ahuja, S., Carriero, N., Gelernter, D.: Linda and friends. IEEE Computer **19**(8), 26–34 (1986). DOI 10.1109/MC.1986.1663305
7. Al-Naeem, T., Gorton, I., Babar, M.A., Rabhi, F., Benatallah, B.: A quality-driven systematic approach for architecting distributed software applications. In: Proceedings of the 27th International Conference on Software Engineering, ICSE '05, pp. 244–253. ACM, New York, NY, USA (2005). DOI 10.1145/1062455.1062508. URL `http://doi.acm.org/10.1145/1062455.1062508`
8. Ali, H.A., Desouky, A.I.E., Saleh, A.I.: Studying and Analysis of a Vertical Web Page Classifier Based on Continuous Learning Naive Bayes (CLNB) Algorithm, pp. 210–254. Information Science (2009)
9. Alippi, C., Boracchi, G., Roveri, M.: Just-in-time classifiers for recurrent concepts. IEEE Transactions on Neural Networks and Learning Systems **24**(4), 620–634 (2013)
10. Amir, E., Anderson, M.L., Chaudhri, V.K.: Report on DARPA workshop on self-aware computer systems. Tech. Rep. UIUCDCS-R-2007-2810, UIUC Comp. Sci. (2007)
11. ANA: Autonomic Network Architecture. URL `www.ana-project.org`. Website: `www.ana-project.org`, (accessed January 2015)
12. Angelov, P.: Nature-inspired methods for knowledge generation from data in real-time (2006). URL `http://www.nisis.risk-technologies.com/popup/Mallorca2006\_Papers/A333\_13774\_Nature-inspiredmethodsforKnowledgeGeneration\_Angelov.pdf`
13. Apache: Hadoop. `http://hadoop.apache.org/docs/r1.2.1/mapred\_tutorial.html`. Online; accessed 2-April-2015
14. Araya-Polo, M., Cabezas, J., Hanzich, M., Pericàs, M., Rubio, F., Gelado, I., Shafiq, M., Morancho, E., Navarro, N., Ayguadé, E., Cela, J.M., Valero, M.: Assessing accelerator-based HPC reverse time migration. IEEE Trans. Parallel Distrib. Syst. **22**(1), 147–162 (2011)

15. Asanovic, K., Bodik, R., Catanzaro, B.C., Gebis, J.J., Husbands, P., Keutzer, K., Patterson, D.A., Plishker, W.L., Shalf, J., Williams, S.W., Yelick, K.A.: The landscape of parallel computing research: A view from Berkeley. Tech. Rep. UCB/EECS-2006-183, EECS Department, University of California, Berkeley (2006)
16. Asendorpf, J.B., Warkentin, V., Baudonnière, P.M.: Self-awareness and other-awareness. ii: Mirror self-recognition, social contingency awareness, and synchronic imitation. Developmental Psychology **32**(2), 313 (1996)
17. Athan, T.W., Papalambros, P.Y.: A note on weighted criteria methods for compromise solutions in multi-objective optimization. Engineering Optimization **27**(2), 155–176 (1996)
18. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning **47**(2–3), 235–256 (2002)
19. Babaoglu, O., Binci, T., Jelasity, M., Montresor, A.: Firefly-inspired heartbeat synchronization in overlay networks. In: First International Conference on Self-Adaptive and Self-Organizing Systems (SASO), pp. 77–86 (2007)
20. Babenko, B., Yang, M.H., Belongie, S.: Robust object tracking with online multiple instance learning. Pattern Analysis and Machine Intelligence **33**(8), 1619–1632 (2011)
21. Bader, J., Zitzler, E.: HypE: an algorithm for fast hypervolume-based many-objective optimization. Tech. Rep. TIK 286, Computer Engineering and Networks Laboratory, ETH Zurich, Zurich (2008)
22. Baena-García, M., Campo-Ávila, J.D., Fidalgo, R., Bifet, A.: Early drift detection method. In: Procedings of the 4th ECML PKDD International Workshop on Knowledge Discovery From Data Streams (IWKDDS), pp. 77–86. Berlin, Germany (2006)
23. Baker, S.: The identification of the self. Psyc. Rev. **4**(3), 272–284 (1897)
24. Banks, A., Gupta, R.: Mqtt version 3.1.1. `http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html` (2014)
25. Bartolini, D.B., Sironi, F., Maggio, M., Cattaneo, R., Sciuto, D., Santambrogio, M.D.: A Framework for Thermal and Performance Management. In: Workshop on Managing Systems Automatically and Dynamically (MAD) (2012)
26. Basheer, I.A., Hajmeer, M.: Artifcial neural networks: fundamentals, computing, design, and application. Journal of Microbiological Methods **43**(1), 3–31 (2000)
27. Basseur, M., Zitzler, E.: Handling uncertainty in indicator-based multiobjective optimization. International Journal of Computational Intelligence Research **2**(3), 255–272 (2006)
28. Basudhar, A., et al.: Constrained efficient global optimization with support vector machines. Struct. Multidiscip. Optim. **46**(2), 201—-221 (2012)
29. Baumann, A., Boltz, M., Ebling, J., Koenig, M., Loos, H.S., Marcel Merkel, W.N., Warzelhan, J.K., Yu, J.: A review and comparison of measures for automatic video surveillance systems. EURASIP Journal on Image and Video Processing **2008**(4) (2008). DOI 10.1155/2008/824726
30. Becker, T., Agne, A., Lewis, P.R., Bahsoon, R., Faniyi, F., Esterle, L., Keller, A., Chandra, A., Jensenius, A.R., Stilkerich, S.C.: EPiCS: Engineering proprioception in computing systems. In: Proc. Int. Conf. on Computational Science and Engineering (CSE), pp. 353–360. IEEE Computer Society (2012)
31. Ben-Hur, A., Weston, J.: A user's guide to support vector machines. Data Mining Techniques for the Life Sciences **609**, 223—-239 (2010)
32. Betts, A., Chong, N., Donaldson, A.F., Qadeer, S., Thompson, P.: GPUVerify: a verifier for GPU kernels. In: OOPSLA'12 (2012)
33. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. European Journal on Operational Research **181**(3), 1653–1669 (2007)
34. Bevilacqua, F., Zamborlin, B., Sypniewski, A., Schnell, N., Guédy, F., Rasamimanana, N.: Continuous realtime gesture following and recognition. In: Gesture in embodied communication and human-computer interaction, pp. 73–84. Springer (2010)
35. Biehl, J.T., Adamczyk, P.D., Bailey, B.P.: Djogger: A mobile dynamic music device. In: CHI '06 Extended Abstracts on Human Factors in Computing Systems, pp. 556–561. ACM (2006)

36. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, United Kingdom (2005)
37. Bojic, I., Lipic, T., Podobnik, V.: Bio-inspired clustering and data diffusion in machine social networks. In: Computational Social Networks, pp. 51–79. Springer (2012)
38. Bongard, J., Lipson, H.: Evolved machines shed light on robustness and resilience. Proceedings of the IEEE **102**(5), 899–914 (2014)
39. Bongard, J., Zykov, V., Lipson, H.: Resilient machines through continuous self-modeling. Science **314**(5802), 1118–1121 (2006)
40. Borkar, S.: Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation. IEEE MICRO pp. 10–16 (2005)
41. Bouabene, G., Jelger, C., Tschudin, C., Schmid, S., Keller, A., May, M.: The Autonomic Network Architecture (ANA). Selected Areas in Communications, IEEE Journal on **28**(1), 4–14 (2010). DOI 10.1109/JSAC.2010.100102
42. Brdiczka, O., Crowley, J.L., Reignier, P.: Learning situation models in a smart home. IEEE Trans. Sys. Man Cyber. Part B **39**, 56–63 (2009)
43. Breiman, L.: Bagging predictors. Machine Learning **24**(2), 123–140 (1996)
44. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)
45. Brockhoff, D., Zitzler, E.: Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods. In: Proceedings of the 2007 IEEE Congress on Evolutionary Computation, pp. 2086–2093 (2007)
46. Buchanan, J.T.: A naive approach for solving mcdm problems: The guess method. Journal of the Operational Research Society **48**(2), 202–206 (1997)
47. Buck, J.: Synchronous rhythmic flashing of fireflies. The Quarterly Review of Biology **13**(3), 301–314 (1938)
48. Buck, J.: Synchronous rhythmic flashing of fireflies II. Quarterly review of biology **63**(3), 265–289 (1988)
49. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Qu, R.: Hyper-heuristics: A survey of the state of the art. Journal of the Operational Research Society **206**(1), 241–264 (2013)
50. Buschmann, F., Henney, K., Douglas, S.C.: Pattern-oriented software architecture: On patterns and pattern languages. John Wiley and Sons (2007)
51. Buss, A.H.: Self-consciousness and social anxiety. W. H. Freeman, San Fransisco, CA, USA (1980)
52. Calinescu, R., Ghezzi, C., Kwiatkowska, M., Mirandola, R.: Self-adaptive software needs quantitative verication at runtime. Communications of the ACM **55**(9), 69–77 (2012)
53. Caramiaux, B., Wanderley, M.M., Bevilacqua, F.: Segmenting and parsing instrumentalists' gestures. Journal of New Music Research **41**(1), 13–29 (2012)
54. Carver, C.S., Scheier, M.: Attention and Self-Regulation: A Control-Theory Approach to Human Behavior. Springer (1981)
55. Castro, L.N.d.: Fundamentals of natural computing: basic concepts, algorithms, and applications. Chapman & Hall/Crc Computer and Information Sciences (2006)
56. Chandra, A.: A methodical framework for engineering co-evolution for simulating socio-economic game playing agents. Ph.D. thesis, The University of Birmingham (2011)
57. Chandra, A., Nymoen, K., Volsund, A., Jensenius, A.R., Glette, K., Torresen, J.: Enabling participants to play rhythmic solos within a group via auctions. In: Proc. Int. Symp. on Computer Music Modeling and Retrieval (CMMR), pp. 674–689 (2012)
58. Chandra, A., Yao, X.: Ensemble learning using multi-objective evolutionary algorithms. Journal of Mathematical Modelling and Algorithms **5**(4), 417–445 (2006)
59. Chang, C., et al.: BEE2: a high-end reconfigurable computing system. IEEE Trans. Designs & Test of Computer (DT) **22**(2), 114–125 (2005)
60. Chen, J., John, L.K.: Efficient program scheduling for heterogeneous multi-core processors. In: Proc. Design Automation Conference (DAC). ACM (2009)
61. Chen, R., Lewis, P.R., Yao, X.: Temperature management for heterogeneous multi-core FPGAs using adaptive evolutionary multi-objective approaches. In: Proceedings of the International Conference on Evolvable Systems (ICES), pp. 101–108. IEEE (2014)

62. Chen, S., Langner, C.A., Mendoza-Denton, R.: When dispositional and role power fit: implications for self-expression and self-other congruence. Journal of Personality and Social Psychology **96**(3), 710–27 (2009)
63. Chen, T., Bahsoon, R.: Self-adaptive and sensitivity-aware qos modeling for the cloud. In: Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '13, pp. 43–52. IEEE Press, Piscataway, NJ, USA (2013). URL http://dl.acm.org/citation.cfm?id=2487336.2487346
64. Chen, T., Bahsoon, R.: Symbiotic and sensitivity-aware architecture for globally-optimal benefit in self-adaptive cloud. In: Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2014, pp. 85–94. ACM, New York, NY, USA (2014). DOI 10.1145/2593929.2593931. URL http://doi.acm.org/10.1145/2593929.2593931
65. Chen, T., Bahsoon, R., Yao, X.: Online qos modeling in the cloud: A hybrid and adaptive multi-learners approach. In: 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC), pp. 327–336. IEEE (2014)
66. Chen, T., Faniyi, F., Bahsoon, R., Lewis, P.R., Yao, X., Minku, L.L., Esterle, L.: The handbook of engineering self-aware and self-expressive systems. Tech. rep., EPiCS EU FP7 project consortium (2014). URL http://arxiv.org/abs/1409.1793. Available via EPiCS website and arXiv
67. Chen, X., Li, X., Wu, H., Qiu, T.: Real-time Object Tracking via CamShift-based Robust Framework. In: Int. Conf. on Information Science and Technology (ICIST). IEEE (2012)
68. Chow, G.C.T., Grigoras, P., Burovskiy, P., Luk, W.: An efficient sparse conjugate gradient solver using a beneš permutation network. In: 24th International Conference on Field Programmable Logic and Applications, FPL 2014, Munich, Germany, 2-4 September, 2014, pp. 1–7 (2014)
69. Chow, G.C.T., Tse, A.H.T., Jin, Q., Luk, W., Leong, P.H.W., Thomas, D.B.: A mixed precision monte carlo methodology for reconfigurable accelerator systems. In: Proceedings of the ACM/SIGDA 20th International Symposium on Field Programmable Gate Arrays, FPGA 2012, Monterey, California, USA, February 22-24, 2012, pp. 57–66 (2012)
70. Christensen, A.L., O'Grady, R., Dorigo, M.: From fireflies to fault-tolerant swarms of robots. IEEE Transactions on Evolutionary Computation **13**(4), 754–766 (2009)
71. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Services Description Language (WSDL) 1.1. World Wide Web Consortium (2001)
72. Chu, F., Zaniolo, C.: Fast and light boosting for adaptive mining of data streams. In: Proceedings of the Eight Pacific-Asia Knowledge Discovery and Data Mining Conference (PAKDD), pp. 282–292. Sydney (2004)
73. Cichowski, A., Madden, C., Detmold, H., Dick, A., Van den Hengel, A., Hill, R.: Tracking Hand-off in Large Surveillance Networks. In: Proceedings of the International Conference Image and Vision Computing New Zealand, pp. 276–281. IEEE Computer Society Press (2009). DOI 10.1109/IVCNZ.2009.5378396
74. Claus, C., Boutilier, C.: The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems. In: Proceedings of the Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence, pp. 746–752. American Association for Artificial Intelligence, Menlo Park, CA, USA (1998)
75. Collins, N.: The analysis of generative music programs. Organised Sound **13**, 237–248 (2008)
76. Collins, R.T., Liu, Y., Leordeanu, M.: Online selection of discriminative tracking features. Pattern Analysis and Machine Intelligence **27**(10), 1631–1643 (2005). DOI 10.1109/tpami.2005.205
77. Colorni, A., Dorigo, M., Maniezzo, V., et al.: Distributed optimization by ant colonies. In: Proceedings of the first European conference on artificial life, vol. 142, pp. 134–142. Elsevier (1991)
78. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based object tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence **25**(5) (2003). DOI 10.1109/tpami.2003.1195991

79. Connors, K.: Chemical kinetics: the study of reaction rates in solution. VCH Publishers (1990)
80. Cox, M.: Metacognition in computation: A selected research review. Artificial Intelligence **169**(2), 104–141 (2005)
81. C.Pilato et al: Speeding-up expensive evaluations in highlevel synthesis using solution modeling and fitness inheritance, vol. 2, pp. 701—-723 (2010)
82. Cramer, T., Schmidl, D., Klemm, M., an Mey, D.: Openmp programming on intel xeon phi coprocessors: An early performance comparison. In: Many-core Applications Research Community (MARC) Symposium at RWTH Aachen University, November 29th-30th 2012, Aachen, Germany, pp. 38–44 (2012)
83. Crockford, D.: The application/json Media Type for JavaScript Object Notation (JSON). RFC 7159, RFC Editor (2014). URL `http://tools.ietf.org/pdf/rfc7159.pdf`
84. Czajkowski, T.S., Aydonat, U., Denisenko, D., Freeman, J., Kinsner, M., Neto, D., Wong, J., Yiannacouras, P., Singh, D.P.: From opencl to high-performance hardware on FPGAS. In: 22nd International Conference on Field Programmable Logic and Applications (FPL), Oslo, Norway, August 29-31, 2012, pp. 531–534 (2012)
85. Datta, K., et al.: Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures. In: SC, p. 4. IEEE (2008)
86. Davidson, A.A., Owens, J.D.: Toward techniques for auto-tuning GPU algorithms. In: Applied Parallel and Scientific Computing - 10th International Conference, PARA 2010, Reykjavík, Iceland, June 6-9, 2010, Revised Selected Papers, Part II, pp. 110–119, Owner = Xinyu, Timestamp = 2015.05.07 (2010)
87. Day, J.: Patterns in Network Architecture: A Return to Fundamentals. Prentice Hall International (2008)
88. Day, J., Matta, I., Mattar, K.: Networking is IPC: A Guiding Principle to a Better Internet. In: Proceedings of the 2008 ACM CoNEXT Conference, CoNEXT '08, pp. 67:1–67:6. ACM, New York, NY, USA (2008). DOI 10.1145/1544012.1544079. URL `http://doi.acm.org/10.1145/1544012.1544079`
89. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. In: 6th Symposium on Operating System Design and Implementation (OSDI 2004), San Francisco, California, USA, December 6-8, 2004, pp. 137–150 (2004)
90. Deb, K.: Multi-objective optimization using evolutionary algorithms, vol. 16. John Wiley & Sons, England (2001)
91. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
92. Denholm, S., Inoue, H., Takenaka, T., Luk, W.: Application-specific customisation of market data feed arbitration. In: Proc. Int. Conf. on Field Programmable Technology (ICFPT), pp. 322–325. IEEE (2013)
93. Denholm, S., Inouey, H., Takenakay, T., Becker, T., Luk, W.: Low latency FPGA acceleration of market data feed arbitration. In: Proc. Int. Conf. on Application-Specific Systems, Architectures, and Processors (ASAP), pp. 36–40. IEEE (2014). DOI 10.1109/ASAP.2014.6868628
94. Dennett, D.C.: Consciousness Explained. Penguin Science (1993)
95. Dennis, J.B., Misunas, D.: A preliminary architecture for a basic data flow processor. In: Proceedings of the 2nd Annual Symposium on Computer Architecture, December 1974, pp. 126–132 (1974)
96. Dieber, B., Simonjan, J., Esterle, L., Rinner, B., Nebehay, G., Pflugfelder, R., Fernandez, G.J.: Ella: Middleware for multi-camera surveillance in heterogeneous visual sensor networks. In: Proc. Int. Conf. on Distributed Smart Cameras (ICDSC) (2013). DOI 10.1109/ICDSC.2013.6778223
97. Dieber, B., Simonjan, J., Esterle, L., Rinner, B., Nebehay, G., Pflugfelder, R., Fernandez, G.J.: Ella: Middleware for multi-camera surveillance in heterogeneous visual sensor networks. In: Proceedings of the Seventh International Conference on Distributed Smart Cameras (ICDSC), 2013, pp. 1–6. IEEE (2013)

98. Dietterich, T.G.: Ensemble methods in machine learning. In: Proceedings of the First International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science, pp. 1–15. Springer-Verlag (2000)
99. Diguet, J.P., Eustache, Y., Gogniat, G.: Closed-loop–based Self-adaptive Hardware/Software-Embedded Systems: Design Methodology and Smart Cam Case Study. ACM Transactions on Embedded Computing Systems **10**(3), 1–28 (2011)
100. Dinh, M.N., Abramson, D., J. Chao, D.K., Gontarek, A., Moench, B., DeRose, L.: Debugging scientific applications with statistical assertions. Procedia Computer Science **9**(0), 1940––1949 (2012)
101. Dobson, S., Denazis, S., Fernández, A., Gaïti, D., Gelenbe, E., Massacci, F., Nixon, P., Saffre, F., Schmidt, N., Zambonelli, F.: A survey of autonomic communications. ACM Transactions on Autonomous and Adaptive Systems **1**(2), 223–259 (2006)
102. Dobson, S., Sterritt, R., Nixon, P., Hinchey, M.: Fulfilling the vision of autonomic computing. IEEE Computer **43**(1), 35 –41 (2010)
103. Dobzhansky, T., Hecht, M., Steere, W.: On some fundamental concepts of evolutionary biology. Evolutionary Biology **2**, 1–34 (1968)
104. Dorigo, M.: Optimization, learning and natural algorithms. Ph. D. Thesis, Politecnico di Milano, Italy (1992)
105. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization. Computational Intelligence Magazine, IEEE **1**(4), 28–39 (2006)
106. Dorigo, M., Blum, C.: Ant colony optimization theory: A survey. Theoretical computer science **344**(2), 243–278 (2005)
107. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **26**(1), 29–41 (1996)
108. Dutta, R., Rouskas, G., Baldine, I., Bragg, A., Stevenson, D.: The SILO Architecture for Services Integration, controL, and Optimization for the Future Internet. In: IEEE International Conference onCommunications (ICC), pp. 1899–1904 (2007). DOI 10.1109/ICC.2007.316
109. Duval, S., Wicklund, R.A.: A theory of objective self awareness. Academic Press (1972)
110. Eckart Zitzler, S.K.: Indicator-based selection in multiobjective search. In: Proceedings of Parallel Problem Solving from Nature (PPSN), vol. 3242, pp. 832–842 (2004)
111. Ehrgott, M.: Other methods for pareto optimality. In: Multicriteria Optimization, Lecture Notes in Economics and Mathematical Systems, vol. 491, pp. 77–102. Springer (2000)
112. Eiben, A.E., Smith, J.E.: Introduction to evolutionary computing. Springer (2003)
113. Eigenfeldt, A., Pasquier, P.: Considering vertical and horizontal context in corpus-based generative electronic dance music. In: Proceedings of the Fourth International Conference on Computational Creativity, p. 72 (2013)
114. Eigenfeldt, A., Pasquier, P.: Evolving structures for electronic dance music. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, GECCO '13, pp. 319–326. ACM, New York, NY, USA (2013)
115. Elkhodary, A., Esfahani, N., Malek, S.: Fusion: a framework for engineering self-tuning self-adaptive software systems. In: Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering, FSE '10, pp. 7–16. ACM, New York, NY, USA (2010). DOI 10.1145/1882291.1882296. URL `http://doi.acm.org/10.1145/1882291.1882296`
116. Elliott, G.T., Tomlinson, B.: Personalsoundtrack: context-aware playlists that adapt to user pace. In: CHI'06 extended abstracts on Human factors in computing systems, pp. 736–741. ACM (2006)
117. Ellis, T., Makris, D., Black, J.: Learning a Multi-camera Topology. In: Proceedings of the Joint International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, pp. 165–171. IEEE Computer Society Press (2003)
118. Elwell, R., Polikar, R.: Incremental learning of concept drift in nonstationary environments. IEEE Transactions on Neural Networks **22**, 1517–1531 (2011)

119. Endo, T., Matsuoka, S.: Massive supercomputing coping with heterogeneity of modern accelerators. In: 22nd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008, Miami, Florida USA, April 14-18, 2008, pp. 1–10 (2008)
120. Erdem, U.M., Sclaroff, S.: Look there! predicting where to look for motion in an active camera network. In: Proceedings of the IEEE Conference on Vision and Signal-based Surveillance, pp. 105–110. Como, Italy (2005)
121. Esterle, L., Lewis, P.R., Bogdanski, M., Rinner, B., Yao, X.: A socio-economic approach to online vision graph generation and handover in distributed smart camera networks. In: Proc. Int. Conf. on Distributed Smart Cameras (ICDSC), pp. 1–6. IEEE (2011). DOI 10.1109/ICDSC.2011.6042902
122. Esterle, L., Lewis, P.R., Caine, H., Yao, X., Rinner, B.: CamSim: A distributed smart camera network simulator. In: Proceedings of the International Conference on Self-Adaptive and Self-Organizing Systems Workshops, pp. 19–20. IEEE Computer Society Press (2013). DOI 10.1109/SASOW.2013.11
123. Esterle, L., Lewis, P.R., Rinner, B., Yao, X.: Improved adaptivity and robustness in decentralised multi-camera networks. In: Proceedings of the International Conference on Distributed Smart Cameras, pp. 1–6. ACM (2012)
124. Esterle, L., Lewis, P.R., Yao, X., Rinner, B.: Socio-economic vision graph generation and handover in distributed smart camera networks. ACM Transactions on Sensor Networks **10**(2), 20:1–20:24 (2014). DOI 10.1145/2530001
125. Faniyi, F., Lewis, P.R., Bahsoon, R., Xao, X.: Architecting self-aware software systems. In: Proc. IEEE/IFIP Conf. on Software Architecture (WICSA), pp. 91–94. IEEE (2014)
126. Farrell, R., Davis, L.S.: Decentralized discovery of camera network topology. In: Proceedings of the International Conference on Distributed Smart Cameras, pp. 1–10. IEEE Computer Society Press (2008). DOI 10.1109/ICDSC.2008.4635696
127. Fels, S., Hinton, G.: Glove-talk: A neural network interface between a data-glove and a speech synthesizer. IEEE Trans. Neural Networks **4**(1), 2–8 (1993)
128. Feng, W.: Making a case for efficient supercomputing. ACM Queue **1, Owner = Xinyu, Timestamp = 2015.05.07**(7), 54–64 (2003)
129. Fenigstein, A., Scheier, M.F., Buss, A.H.: Public and private self-consciousness: Assessment and theory. Journal of Consulting and Clinical Psychology **43**(4), 522–527 (1975)
130. Fern, A., Givan, R.: Online ensemble learning: An empirical study. Machine Learning **53**(1–2), 71–109 (2003)
131. Fette, B.: Cognitive radio technology. Academic Press (2009)
132. Fiebrink, R., Trueman, D., Cook, P.R.: A meta-instrument for interactive, on-the-fly machine learning. In: Proceedings of the International Conference on New Interfaces for Musical Expression. Pittsburgh (2009)
133. Fielding, R.T., Taylor, R.N.: Principled design of the modern web architecture. ACM Trans. Internet Technol. **2**(2), 115–150 (2002). DOI 10.1145/514183.514185. URL `http://doi.acm.org/10.1145/514183.514185`
134. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Proceedings of the 13th International Conference on Machine Learning, pp. 148–156 (1996)
135. Froming, W.J., Walker, G.R., Lopyan, K.J.: Public and private self-awareness: When personal attitudes conflict with societal expectations. Journal of Experimental Social Psychology **18**(5), 476 – 487 (1982). DOI DOI:10.1016/0022-1031(82)90067-1
136. Fu, H., Sendhoff, B., Tang, K., Yao, X.: Finding robust solutions to dynamic optimization problems. In: Proceedings of the 16th European conference on Applications of Evolutionary Computation (EvoApplications), pp. 616–625 (2013)
137. Funie, A., Salmon, M., Luk, W.: A hybrid genetic-programming swarm-optimisation approach for examining the nature and stability of high frequency trading strategies. In: 13th International Conference on Machine Learning and Applications ICMLA, pp. 29–34. Detroit,USA (2014). DOI 10.1109/ICMLA.2014.11. URL `http://dx.doi.org/10.1109/ICMLA.2014.11`
138. Gallup, G.G.: Chimpanzees: self-recognition. Science (1970)

139. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. In: Proceedings of the 7th Brazilian Symposium on Artificial Intelligence (SBIA) - Lecture Notes in Computer Science, vol. 3171, pp. 286–295. Springer, São Luiz do Maranhão, Brazil (2004)
140. Gao, J., Fan, W., Han, J.: On appropriate assumptions to mine data streams: Analysis and practice. In: Seventh IEEE International Conference on Data Mining (ICDM), pp. 143–152 (2007)
141. Garlan, D., Cheng, S.W., Huang, A.C., Schmerl, B., Steenkiste, P.: Rainbow: architecture-based self-adaptation with reusable infrastructure. Computer **37**(10), 46–54 (2004)
142. Gelenbe, E., Loukas, G.: A self-aware approach to denial of service defence. Computer Networks **51**, 1299—1314 (2007)
143. Goto, M.: Active music listening interfaces based on signal processing. In: IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 4, pp. 1441—1444 (2007)
144. Gouin-Vallerand, C., Abdulrazak, B., Giroux, S., Mokhtari, M.: Toward autonomic pervasive computing. In: Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services, iiWAS '08, pp. 673–676. ACM, New York, NY, USA (2008)
145. Goukens, C., Dewitte, S., Warlop, L.: Me, myself, and my choices: The influence of private self-awareness on preference-behavior consistency. Tech. rep., Katholieke Universiteit Leuven (2007)
146. Grabner, H., Bischof, H.: On-line boosting and vision. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 260–267 (2006)
147. Grabner, H., Leistner, C., Bischof, H.: Semi-supervised On-Line boosting for robust tracking. In: European Conference on Computer Vision, Lecture Notes in Computer Science, vol. 5302, pp. 234–247 (2008)
148. Group, K.: The opencl specification, version: 1.1. `http://www.khronos.org/registry/cl/specs/opencl-1.1.pdf`. [Online; accessed 2-April-2015]
149. Gudger, E.W.: A historical note on the synchronous flashing of fireflies. Science **50**(1286), 188–190 (1919)
150. Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.J., Nielsen, H.F., Karmarkar, A., Lafon, Y.: SOAP Version 1.2. World Wide Web Consortium (2007)
151. Guo, C., Luk, W.: Accelerating maximum likelihood estimation for hawkes point processes. In: Proc. Int. Conf. on Field Programmable Logic and Applications (FPL), pp. 1–6. IEEE (2013)
152. Guo, C., Luk, W.: Accelerating parameter estimation for multivariate self-exciting point processes. In: Proc. Int. Symp. on Field-Programmable Gate Arrays (FPGA), pp. 181–184. ACM (2014). DOI 10.1145/2554688.2554765
153. Haikonen, P.O.: Reflections of consciousness: The mirror test. In: Proceedings of the 2007 AAAI Fall Symposium on Consciousness and Artificial Intelligence, pp. 67–71 (2007)
154. Hamid, R., Maddi, S., Johnson, A., Bobick, A., Essa, I., Isbell, C.: A novel sequence representation for unsupervised analysis of human activities. Artificial Intelligence **173**(14), 1221 – 1244 (2009). DOI DOI:10.1016/j.artint.2009.05.002
155. Hansen, N.: The CMA evolution strategy: A comparing review. In: J. Lozano, P. Larrañaga, I. Inza, E. Bengoetxea (eds.) Towards a New Evolutionary Computation, *Studies in Fuzziness and Soft Computing*, vol. 192, pp. 75–102. Springer Berlin Heidelberg (2006)
156. Hansen, N.: The CMA evolution strategy: A comparing review. In: J. Lozano, P. Larrañaga, I. Inza, E. Bengoetxea (eds.) Towards a New Evolutionary Computation, *Studies in Fuzziness and Soft Computing*, vol. 192, pp. 75–102. Springer Berlin Heidelberg (2006)
157. Happe, M., Agne, A., Plessl, C.: Measuring and predicting temperature distributions on FPGAs at run-time. In: Proc. Int. Conf. on ReConFigurable Computing and FPGAs (ReConFig), pp. 55–60. IEEE Computer Society, Los Alamitos, CA, USA (2011). DOI 10.1109/ReConFig.2011.59
158. Happe, M., Hangmann, H., Agne, A., Plessl, C.: Eight ways to put your FPGA on fire – a systematic study of heat generators. In: Proc. Int. Conf. on ReConFigurable Computing and FPGAs (ReConFig). IEEE Computer Society (2012). Received Best Paper Award

159. Happe, M., Huang, Y., Keller, A.: Dynamic protocol stacks in smart camera networks. In: Proc. Int. Conf. on ReConFigurable Computing and FPGAs (ReConFig). IEEE (2014). To appear
160. Happe, M., Traber, A., Keller, A.: Preemptive Hardware Multitasking in ReconOS. In: International Symposium on Applied Reconfigurable Computing (ARC), Springer (2015)
161. Hart, J.W., Scassellati, B.: Robotic self-modeling. In: J. Pitt (ed.) The Computer After Me, pp. 207–218. Imperial College Press / World Scientific Book (2014)
162. Heath, D., Jarrow, R., Morton, A.: Bond pricing and the term structure of interest rates: A new methodology for contingent claims valuation. Econometrica **60**(77-105), 60–65 (1992)
163. Hernandez, H., Blum, C.: Distributed graph coloring in wireless ad hoc networks: A lightweight algorithm based on japanese tree frogs' calling behaviour. In: Proceedings of the 4th Joint IFIP Wireless and Mobile Networking Conference (WMNC), pp. 1–7 (2011)
164. Herzen, B.V.: Signal processing at 250 mhz using high-performance FPGA's. In: Proceedings of the 1997 ACM Fifth International Symposium on Field-programmable Gate Arrays, pp. 62–68 (1997)
165. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(8), 832–844 (1998)
166. Ho, T.K., Hull, J.J., Srihari, S.N.: Decision combination in multiple classifier systems. IEEE Transactions on Pattern Analysis and Machine Intelligence **16**(1), 66–75 (1994)
167. Ho, T.S.Y., b. Lee, S.: Term structure movements and pricing interest rate contingent claims. Journal of Finance **41**(5), 1986 (1011–1029)
168. Hoare, C.A.R.: An axiomatic basis for computer programming. Commun. ACM **12**(10), 576–580 (1969)
169. Hockman, J.A., Wanderley, M.M., Fujinaga, I.: Real-time phase vocoder manipulation by runner's pace. In: Proceedings of the International Conference on New Interfaces for Musical Expression (2009)
170. Hoffmann, H., Eastep, J., Santambrogio, M., Miller, J., Agarwal, A.: Application heartbeats for software performance and health. In: ACM SIGPLAN Notices, vol. 45, pp. 347–348. ACM (2010)
171. Hoffmann, H., Eastep, J., Santambrogio, M.D., Miller, J.E., Agarwal, A.: Application Heartbeats: A Generic Interface for Specifying Program Performance and Goals in Autonomous Computing Environments. In: International Conference on Autonomic Computing (ICAC) (2010)
172. Hoffmann, H., Holt, J., Kurian, G., Lau, E., Maggio, M., Miller, J.E., Neuman, S.M., Sinangil, M., Sinangil, Y., Agarwal, A., Chandrakasan, A.P., Devadas, S.: Self-aware computing in the angstrom processor. In: Proceedings of the 49th Annual Design Automation Conference, DAC '12, pp. 259–264. ACM, New York, NY, USA (2012)
173. Hoffmann, H., Maggio, M., Santambrogio, M.D., Leva, A., Agarwal, A.: SEEC: A general and extensible framework for self-aware computing. Tech. Rep. MIT-CSAIL-TR-2011-046, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology (2011)
174. Holland, B., et al.: An analytical model for multilevel performance prediction of Multi-FPGA systems. ACM Trans. Reconfigurable Technol. Syst **4**(3), 27–28 (2011)
175. Holland, O., Goodman, R.B.: Robots with internal models: A route to machine consciousness? Journal of Consciousness Studies **10**(4), 77–109 (2003)
176. Holopainen, R.: Self-organised sound with autonomous instruments: Aesthetics and experiments. Ph.D. thesis, University of Oslo (2012)
177. Hölzl, M., Wirsing, M.: Towards a system model for ensembles. In: Formal Modeling: Actors, Open Systems, Biological Systems, pp. 241–261. Springer (2011)
178. Hölzl, M., Wirsing, M.: Issues in engineering self-aware and self-expressive ensembles. In: J. Pitt (ed.) The Computer After Me, pp. 37–54. Imperial College Press / World Scientific Book (2014)
179. Horn, J., Nafpliotis, N., Goldberg, D.E.: A niched pareto genetic algorithm for multiobjective optimization. In: Proceedings of the 1st IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, pp. 82–87. IEEE (1994)

180. Horn, P.: Autonomic computing: IBM's perspective on the state of information technology. Armonk, NY, USA. International Business Machines Corporation. (2001)
181. Hosseini, M.J., Ahmadi, Z., Beigy, H.: Using a classifier pool in accuracy based tracking of recurring concepts in data stream classification. Evolving Systems **4**(1), 43–60 (2013)
182. hsing Hsu, C., chun Feng, W.: Reducing overheating-induced failures via performance-aware cpu power management. In: In The 6th International Conference on Linux Clusters: The HPC Revolution (2005)
183. Hu, F., Evans, J.J.: Power and environment aware control of beowulf clusters. Cluster Computing **12**, 299–308 (2009)
184. Hu, W., Tan, T., Wang, L., Maybank, S.: A Survey on Visual Surveillance of Object Motion and Behaviors. IEEE Transactions on Systems, Man and Cybernetics, Part C **34**(3), 334–352 (2004)
185. Huang, T., Russell, S.: Object Identification in a Bayesian Context. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 1276–1283 (1997)
186. Huebscher, M., McCann, J.: Simulation model for self-adaptive applications in pervasive computing. In: Proceedings of the 15th International Workshop on Database and Expert Systems Applications,, pp. 694–698. IEEE Computer Society (2004)
187. Hume, D.: A Treatise of Human Nature. Gutenberg eBook (1739). Digital edition, 2010, available at http://www.gutenberg.org/ebooks/4705, retrieved 21st January 2013
188. Hunkeler, U., Truong, H.L., Stanford-Clark, A.: Mqtt-s—a publish/subscribe protocol for wireless sensor networks. In: In Proceedings of the Third International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE), 2008, pp. 791–798. IEEE (2008)
189. Hunt, A., Wanderley, M.M., Paradis, M.: The importance of parameter mapping in electronic instrument design. In: Proceedings of the International Conference on New Interfaces for Musical Expression, pp. 1–6. National University of Singapore, Singapore, Singapore (2002)
190. IBM: An architectural blueprint for autonomic computing (2003)
191. Iglesia, D.: Mobmuplat (iOS application). Iglesia Intermedia (2013)
192. Inc., I.: Sophisticated library for vector parallelism. `http://software.intel.com/en-us/articles/intel-array-building-blocks/`. [Online; accessed 2-April-2015]
193. Investigating RINA as an Alternative to TCP/IP. URL `irati.eu`. Website: `irati.eu`, (accessed January 2015)
194. Ishibuchi, H., Murata, T.: A multiobjective genetic local search algorithm and its application to flowshop scheduling. IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews **28**(3), 392–403 (1998)
195. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Iterative approach to indicator-based multiobjective optimization. In: Proceedings of the 2007 IEEE Congress on Evolutionary Computation, pp. 3967–3974. IEEE (2007)
196. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization. In: Proceedings of the 3rd International Workshop on Genetic and Evolving Systems (GEFS), pp. 47–52. IEEE (2008)
197. J., C., G., S., D., G.A.: High-level synthesis of in-circuit assertions for verification, debugging, and timing analysis. International Journal of Reconfigurable Computing **2011** (2011)
198. James, W.: The principles of psychology. Henry Holt & Co. (1890)
199. Janusevskis, J., et al.: Expected improvements for the asynchronous parallel global optimization of expensive functions: Potentials and challenges. In: Learning and Intelligent Optimization, pp. 413—-418. Springer (2012)
200. Javed, O., Khan, S., Rasheed, Z., Shah, M.: Camera Handoff: Tracking in Multiple Uncalibrated Stationary Cameras. In: Proceedings of the Workshop on Human Motion, pp. 113–118. IEEE Computer Society Press (2000). DOI 10.1109/HUMO.2000.897380
201. Javed, O., Rasheed, Z., Shafique, K., Shah, M.: Tracking across Multiple Cameras Disjoint Views. In: Proceedings of IEEE International Conference on Computer Vision, p. 952–957 (2003)

202. Jia, J., Veeravalli, B., Ghose, D.: Adaptive load distribution strategies for divisible load processing on resource unaware multilevel tree networks. IEEE Transactions on Computers **56**(7), 999–1005 (2007)
203. Jin, Q., et al.: Optimising explicit finite difference option pricing for dynamic constant reconfiguration. In: Proc. FPL, pp. 165—-172 (2012)
204. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. of Global Optimization **13**(4), 455—-492 (1998)
205. Jones, P., Cho, Y., Lockwood, J.: Dynamically optimizing FPGA applications by monitoring temperature and workloads. In: Proc. Int. Conf. on VLSI Design (VLSID). IEEE (2007)
206. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-Learning-detection. Pattern Analysis and Machine Intelligence **34**(7), 1409–1422 (2012)
207. Kalman, R.E.: A New Approach to Linear Filtering and Prediction Problems. Journal of Fluids Engineering **82**(1), 35–45 (1960)
208. Kamil, S., Chan, C., Oliker, L., Shalf, J., Williams, S.: An auto-tuning framework for parallel multicore stencil computations. In: Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on, pp. 1–12. IEEE Computer Society Press (2010)
209. Kang, J., Cohen, I., Medioni, G.: Continuous Tracking within and across Cameras Streams. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 267–272 (2003)
210. Kant, I.: The critique of pure reason. Gutenberg eBook (1781). Digital edition, 2003, available at http://www.gutenberg.org/ebooks/4280, retrieved 21st January 2013
211. Kela, J., Korpipää, P., Mäntyjärvi, J., Kallio, S., Savino, G., Jozzo, L., Marca, D.: Accelerometer-based gesture control for a design environment. Personal and Ubiquitous Computing **10**(5), 285–299 (2006)
212. Keller, A., Borkmann, D., Neuhaus, S., Happe, M.: Self-awareness in computer networks. Int. Journal of Reconfigurable Computing (IJRC) (2014). DOI 10.1155/2014/692076
213. Keller, A., Plattner, B., Lübbers, E., Platzner, M., Plessl, C.: Reconfigurable nodes for future networks. In: Proc. IEEE Globecom Workshop on Network of the Future (FutureNet), pp. 372–376. IEEE (2010). DOI 10.1109/GLOCOMW.2010.5700341
214. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. IEEE Computer **36**(1), 41–50 (2003)
215. Kettnaker, V., Zabith, R.: Bayesian Multi- camera Surveillance. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 117–123 (1999)
216. Khan, M.I., Rinner, B.: Energy-aware task scheduling in wireless sensor networks based on cooperative reinforcement learning. In: Proceedings of the International Conference on Communications Workshops (ICCW). IEEE (2014). DOI 10.1109/ICCW.2014.6881310
217. Khare, V., Yao, X., Deb, K.: Performance scaling of multi-objective evolutionary algorithms. In: Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science, vol. 2632, pp. 376–390. Springer (2003)
218. Kim, H.S., Sherman, D.K.: Express yourself: Culture and the effect of self-expression on choice. Journal of Personality and Social Psychology **92**(1), 1–11 (2007). DOI DOI:10.1037/0022-3514.92.1.1
219. Kim, J., Seo, S., Lee, J., Nah, J., Jo, G., Lee, J.: Opencl as a unified programming model for heterogeneous CPU/GPU clusters. In: Proceedings of the 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPOPP 2012, New Orleans, LA, USA, February 25-29, 2012, pp. 299–300 (2012)
220. Kittler, J., Hatef, M., Duin, R.P., Matas, J.: On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(3), 226–239 (1998)
221. Klinglmayr, J., Bettstetter, C.: Self-organizing synchronization with inhibitory-coupled oscillators: Convergence and robustness. ACM Transactions on Autonomous and Adaptive Systems (TAAS) **7**(3), 30:1–30:22 (2012)
222. Klinglmayr, J., Kirst, C., Bettstetter, C., Timme, M.: Guaranteeing global synchronization in networks with stochastic interactions. New Journal of Physics **14**(7), 073,031:1–073,031:13 (2012)

223. Knutzen, H., Nymoen, K., Torresen, J.: PheroMusic [iOS application]. URL \url{http://itunes.apple.com/app/pheromusic/id910100415}
224. Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: An ensemble method for drifting concepts. Journal of Machine Learning Research **8**, 2755–2790 (2007)
225. Koski, J., Silvennoinen, R.: Norm methods and partial weighting in multicriterion optimization of structures. International Journal for Numerical Methods in Engineering **24**(6), 1101–1121 (1987)
226. Kramer, J., Magee, J.: Self-managed systems: an architectural challenge. In: Future of Software Engineering, 2007. FOSE'07, pp. 259–268. IEEE (2007)
227. Krishnamoorthy, S., et al.: Effective automatic parallelization of stencil computations. In: ACM Sigplan Notices, vol. 42, pp. 235–244. ACM (2007)
228. Kuhn, H.W., Yaw, B.: The Hungarian Method for the Assignment Problem. Naval Research Logistics Quarterly pp. 83–97 (1955)
229. Kuncheva, L.I.: A theoretical study on six classifier fusion strategies. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**(2), 281–286 (2002)
230. Kurek, M., Becker, T., Chau, T.C., Luk, W.: Automating optimization of reconfigurable designs. In: Proc. Int. Symp. on Field-Programmable Custom Computing Machines (FCCM), pp. 210–213. IEEE (2014). DOI 10.1109/FCCM.2014.65
231. Kurek, M., Becker, T., Luk, W.: "parametric optimization of reconfigurable designs using machine learning. In: ARC'13, pp. 134—-145 (2013)
232. Legrain, L., Cleeremans, A., Destrebecqz, A.: Distinguishing three levels in explicit self-awareness. Consciousness and Cognition **20**, 578–585 (2011)
233. Legrand, D.: Pre-reflective self-as-subject from experiential and empirical perspectives. Consciousness and Cognition **16**(3), 583–599 (2007)
234. Leidenfrost, R., Elmenreich, W.: Firefly clock synchronization in an 802.15. 4 wireless network. EURASIP Journal on Embedded Systems **2009**, 7:1–7:17 (2009)
235. Leland, W., Taqqu, M., Willinger, W., Wilson, D.: On the self-similar nature of ethernet traffic (extended version). Networking, IEEE/ACM Transactions on **2**(1), 1–15 (1994). DOI 10.1109/90.282603
236. Leutenegger, S., Chli, M., Siegwart, R.Y.: BRISK: Binary robust invariant scalable keypoints. In: Proceedings of the International Conference on Computer Vision, pp. 2548–2555. IEEE (2011). DOI 10.1109/iccv.2011.6126542
237. Lewis, P., Platzner, M., Yao, X.: An outlook for self-awareness in computing systems. Awareness Magazine (2012). DOI 10.2417/3201203.004093
238. Lewis, P.R., Chandra, A., Faniyi, F., Glette, K., Chen, T., Bahsoon, R., Torresen, J., Yao, X.: Architectural aspects of self-aware and self-expressive computing systems. Computer **48**(8) (2015)
239. Lewis, P.R., Chandra, A., Parsons, S., Robinson, E., Glette, K., Bahsoon, R., Torresen, J., Yao, X.: A Survey of Self-Awareness and Its Application in Computing Systems. In: Proceedings of the International Conference on Self-Adaptive and Self-Organizing Systems Workshops (2011)
240. Lewis, P.R., Chandra, A., Parsons, S., Robinson, E., Glette, K., Bahsoon, R., Torresen, J., Yao, X.: A survey of self-awareness and its application in computing systems. In: Proceedings of the International Conference on Self-Adaptive and Self-Organizing Systems Workshops, pp. 102–107. IEEE Computer Society, Ann Arbor, MI, USA (2011)
241. Lewis, P.R., Esterle, L., Chandra, A., Rinner, B., Torresen, J., Yao, X.: Static, dynamic, and adaptive heterogeneity in distributed smart camera networks. ACM Trans. Auton. Adapt. Syst. **10**(2), 8:1–8:30 (2015). DOI 10.1145/2764460
242. Lewis, P.R., Esterle, L., Chandra, A., Rinner, B., Yao, X.: Learning to be different: Heterogeneity and efficiency in distributed smart camera networks. In: Proceedings of the International Conference on Self-Adaptive and Self-Organizing Systems (SASO), pp. 209–218. IEEE Computer Society Press (2013). DOI 10.1109/SASO.2013.20
243. Lewis, P.R., Marrow, P., Yao, X.: Resource allocation in decentralised computational systems: An evolutionary market based approach. Autonomous Agents and Multi-Agent Systems **21**(2), 143–171 (2010)

244. Li, B., Li, J., Tang, K., Yao, X.: An improved two archive algorithm for many-objective optimization. In: Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 2869–2876. IEEE (2014)
245. Li, G., Gopalakrishnan, G.: Scaleable SMT-based verification of GPU kernel functions. In: FSE-18 (2010)
246. Li, H., Zhang, Q.: Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. IEEE Transactions on Evolutionary Computation **13**(2), 284–302 (2009)
247. Li, Y., Bhanu, B.: Utility-based Camera Assignment in a Video Network: A Game Theoretic Framework. Sensors Journal **11**(3), 676–687 (2011)
248. Liang, C.J.M., Liu, J., Luo, L., Terzis, A., Zhao, F.: Racnet: A high-fidelity data center sensing network. Proc. SenSys (2009, Owner = Xinyu, Timestamp = 2015.05.07)
249. Liu, J., Zhong, L., Wickramasuriya, J., Vasudevan, V.: uwave: Accelerometer-based personalized gesture recognition and its applications. Pervasive and Mobile Computing **5**(6), 657–675 (2009)
250. Liu, Y., Yao, X.: Ensemble learning via negative correlation. Neural Networks **12**(10), 1399–1404 (1999)
251. Lübbers, E., Platzner, M.: ReconOS: Multithreaded programming for reconfigurable computers. ACM Transactions on Embedded Computing Systems (TECS) **9** (2009)
252. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp. 674–679 (1981)
253. Lübbers, E., Platzner, M.: Cooperative multithreading in dynamically reconfigurable systems. In: Proc. Int. Conf. on Field Programmable Logic and Applications (FPL), pp. 1–4. IEEE (2009)
254. Makris, D., Ellis, T., Black, J.: Bridging the Gaps between Cameras. In: Proceedings of Conference on Computer Vision and Pattern Recognition, vol. 2 (2004)
255. Marler, R.T., Arora, J.S.: Function-transformation methods for multi-objective optimization. Engineering Optimization **37**(6), 551–570 (2005)
256. Marrow, P.: Nature-inspired computing technology and applications. BT Technology Journal **18**(4), 13–23 (2000)
257. Marsaglia, G., Bray, T.A.: A convenient method for generating normal variables. SIAM Review **6**(3), 260–264 (1964)
258. Masahiro, N., Takaesu, H., Demachi, H., Oono, M., Saito, H.: Development of an automatic music selection system based on runner's step frequency. In: Proc. of 2008 International Conf. on Music Information Retrieval, pp. 193–8 (2008)
259. Massie, M.L., b, B.N.C., Culler, D.E.: The Ganglia distributed monitoring system: design, implementation, and experience. Parallel Computing **30**, 817–840 (2004)
260. Mathar, R., Mattfeldt, J.: Pulse-coupled decentral synchronization. SIAM Journal on Applied Mathematics **56**(4), 1094–1106 (1996)
261. Max [computer software]. URL `\url{http://cycling74.com}`
262. Mckay, M.D., et al.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics pp. 55—-61 (2000)
263. Mehta, N.R., Medvidovic, N.: Composing architectural styles from architectural primitives. In: ESEC / SIGSOFT FSE, pp. 347–350. ACM (2003). URL `http://dblp.uni-trier.de/db/conf/sigsoft/fse2003.html\#MehtaM03`
264. Menasce, D.A., Sousa, J.a.P., Malek, S., Gomaa, H.: Qos architectural patterns for self-architecting software systems. In: Proceedings of the 7th International Conference on Autonomic Computing, ICAC '10, pp. 195–204. ACM, New York, NY, USA (2010). DOI 10.1145/1809049.1809084
265. Metcalfe, J., Shimamura, A.P. (eds.): Metacognition: Knowing about knowing. MIT Press, Cambridge, MA, USA (1994)
266. Michalski, R.S.: A Theory and Methodology of Inductive Learning. In: Machine Learning, Symbolic Computation, pp. 83–134. Springer Berlin Heidelberg (1983)

267. Miettinen, K., Mäkelä, M.M.: Interactive bundle-based method for nondifferentiable multiobjeective optimization: nimbus §. Optimization Journal **34**(3), 231–246 (1995)
268. Minku, L.L.: Online ensemble learning in the presence of concept drift. Ph.D. thesis, School of Computer Science, University of Birmingham, Birmingham, UK (2010)
269. Minku, L.L., Yao, X.: DDD: A new ensemble approach for dealing with concept drift. IEEE Transactions on Knowledge and Data Engineering **24**(4), 619–633 (2012)
270. Minku, L.L., Yao, X.: Software effort estimation as a multi-objective learning problem. ACM Transactions on Software Engineering and Methodology **22**(4), 35:1–32 (2013)
271. Miranda, E.R., Wanderley, M.: New Digital Musical Instruments: Control And Interaction Beyond the Keyboard. A-R Editions, Inc., Middleton, WI (2006)
272. Mirollo, R.E., Strogatz, S.H.: Synchronization of pulse-coupled biological oscillators. SIAM Journal on Applied Mathematics **50**(6), 1645–1662 (1990)
273. Mitchell, M.: Self-awareness and control in decentralized systems. In: Metacognition in Computation, pp. 80–85 (2005)
274. Modler, P.: Neural networks for mapping hand gestures to sound synthesis parameters, vol. 18. IRCAM — Centre Pompidou (2000)
275. Moens, B., van Noorden, L., Leman, M.: D-jogger: Syncing music with walking. In: Sound and Music Computing Conference, pp. 451–456. Barcelona, Spain (2010)
276. Morin, A.: Levels of consciousness and self-awareness : A comparison and integration of various neurocognitive views. Cons. and Cog. **15**(2), 358–71 (2006)
277. Morin, A., Everett, J.: Conscience de soi et langage interieur: Quelques speculations. [self-awareness and inner speech: Some speculations]. Philosophiques **XVII**(2), 169–188 (1990)
278. Müller-Schloer, C., Schmeck, H., Ungerer, T.: Organic computing: a paradigm shift for complex systems. Springer (2011)
279. Nakashima, H., Aghajan, H., Augusto, J.C.: Handbook of ambient intelligence and smart environments. Springer (2009)
280. Narukawa, K., Tanigaki, Y., Ishibuchi, H.: Evolutionary many-objective optimization using preference on hyperplane. In: Proceedings of the 2014 Conference on Genetic and Evolutionary Computation Companion, pp. 91–92. ACM (2014)
281. Natarajan, P., Atrey, P.K., Kankanhalli, M.: Multi-camera coordination and control in surveillance systems: A survey. ACM Transactions on Multimedia Computing, Communications and Applications **11**(4), 57:1–57:30 (2015). DOI 10.1145/2710128
282. Nebehay, G., Chibamu, W., Lewis, P.R., Chandra, A., Pflugfelder, R., Yao, X.: Can diversity amongst learners improve online object tracking? In: Z.H. Zhou, F. Roli, J. Kittler (eds.) Multiple Classifier Systems, *Lecture Notes in Computer Science*, vol. 7872, pp. 212–223. Springer, Berlin / Heidelberg (2013). DOI 10.1007/978-3-642-38067-9_19
283. Nebehay, G., Pflugfelder, R.: Consensus-based matching and tracking of keypoints for object tracking. In: Proceedings of the Winter Conference on Applications of Computer Vision (WACV). IEEE (2014)
284. Nebro, A.J., Luna, F., Alba, E., Beham, A., Dorronsoro, B.: AbYSS: adapting scatter search for multiobjective optimization. Tech. Rep. ITI-2006-2, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, Malaga (2006)
285. Neisser, U.: The roots of self-knowledge: Perceiving self, it, and thou. Annals of the NY AoS. **818**, 19–33 (1997)
286. netem. URL `http://www.linuxfoundation.org/collaborate/workgroups/networking/netem`. Website: `http://www.linuxfoundation.org/collaborate/workgroups/networking/netem`, (accessed January 2015)
287. Nguyen, A., Satish, N., Chhugani, J., Kim, C., Dubey, P.: 3.5-d blocking optimization for stencil computations on modern CPUs and GPUs. In: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–13. IEEE (2010)
288. Niezen, G., Hancke, G.P.: Evaluating and optimising accelerometer-based gesture recognition techniques for mobile devices. In: AFRICON, 2009. AFRICON'09., pp. 1–6. IEEE (2009)

289. Nishida, K.: Learning and detecting concept drift. Ph.D. thesis, Hokkaido University (2008). URL http://lis2.huie.hokudai.ac.jp/~knishida/paper/nishida2008-dissertation.pdf
290. Nishida, K., Yamauchi, K.: Detecting concept drift using statistical testing. In: Proceedings of the Tenth International Conference on Discovery Science (DS'07) - Lecture Notes in Artificial Intelligence, vol. 3316, pp. 264–269. Sendai, Japan (2007)
291. Niu, X., Chau, T.C.P., Jin, Q., Luk, W., Liu, Q.: Automating elimination of idle functions by run-time reconfiguration. In: 21st IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM 2013, Seattle, WA, USA, April 28-30, 2013, pp. 97–104 (2013)
292. Niu, X., Coutinho, J.G.F., Luk, W.: A scalable design approach for stencil computation on reconfigurable clusters. In: 23rd International Conference on Field programmable Logic and Applications, FPL 2013, Porto, Portugal, September 2-4, 2013, pp. 1–4 (2013)
293. Niu, X., Tsoi, K.H., Luk, W.: Reconfiguring distributed applications in FPGA accelerated cluster with wireless networking. In: International Conference on Field Programmable Logic and Applications, FPL 2011, September 5-7, Chania, Crete, Greece, pp. 545–550 (2011)
294. Niu, X., et al.: Exploiting run-time reconfiguration in stencil computation. In: Proc. FPL, pp. 173–180 (2012)
295. Niu, X., et al.: Exploiting run-time reconfiguration in stencil computation. In: Proc. FPL, pp. 173—-180 (2012)
296. NVIDIA: Cuda zone. http://www.nvidia.com/object/cuda_home_new.html. Online; accessed 2-April-2015
297. Nymoen, K., Chandra, A., Glette, K., Torresen, J.: Decentralized harmonic synchronization in mobile music systems. In: Proceedings of the International Conference on Awareness Science & Technology (iCAST), pp. 1–6 (2014)
298. Nymoen, K., Chandra, A., Glette, K., Torresen, J., Voldsund, A., Jensenius, A.R.: PheroMusic: Navigating a musical space for active music experiences. In: Proc. Int. Computer Music Conference (ICMC) joint with the Sound and Music Computing Conference, pp. 1715–1718 (2014)
299. Nymoen, K., Song, S., Hafting, Y., Torresen, J.: Funky Sole Music: Gait recognition and adaptive mapping. In: Proc. Int. Conf. on New Interfaces for Musical Expression (NIME), pp. 299–302 (2014)
300. Okuma, K., Taleghani, A., de Freitas, N., Little, J., Lowe, D.: A Boosted Particle Filter: Multitarget Detection and Tracking. In: Proceedings of 8th European Conference on Computer Vision, vol. 3021, pp. 28–39 (2004)
301. Olfati-Saber, R.: Distributed kalman filtering for sensor networks. In: Proceedings of the Conference on Decision and Control, pp. 5492–5498 (2007). DOI 10.1109/CDC.2007.4434303
302. Olsson, R.A., Keen, A.W.: Remote procedure call. The JR Programming Language: Concurrent Programming in an Extended Java pp. 91–105 (2004)
303. Ong, Y.S., Nair, P.B., Keane, A.J.: Evolutionary optimization of computationally expensive problems via surrogate modeling. AIAA Journal **41**(4), 689–696 (2003)
304. Ontañón, S., Plaza, E.: Multiagent Inductive Learning: An Argumentation-based Approach. In: J. Fürnkranz, T. Joachims (eds.) Proceedings of the 27th International Conference on Machine Learning (ICML), pp. 839–846. Omnipress, Haifa, Israel (2010)
305. Oxford: Oxford dictionaries – adapt. http://www.oxforddictionaries.com/definition/english/adapt (Accessed in December 2014)
306. Oza, N.C.: Online bagging and boosting. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pp. 2340–2345 (2005)
307. Oza, N.C., Russell, S.: Experimental comparisons of online and batch versions of bagging and boosting. In: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 359–364. ACM (2001)
308. Özuysal, M., Calonder, M., Lepetit, V., Fua, P.: Fast keypoint recognition using random ferns. Pattern Analysis and Machine Intelligence **32**(3), 448–461 (2010). DOI 10.1109/tpami.2009.23

309. Page, I., Luk, W.: Compiling occam into field-programmable gate arrays. In: International Conference on Field programmable Logic and Applications (FPL) (1991)
310. Papakonstantinou, A., Liang, Y., Stratton, J.A., Gururaj, K., Chen, D., Hwu, W.W., Cong, J.: Multilevel granularity parallelism synthesis on fpgas. In: IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM 2011, Salt Lake City, Utah, USA, 1-3 May 2011, pp. 178–185 (2011)
311. Parashar, M., Hariri, S.: Autonomic computing: an overview. In: Proceedings of the 2004 international conference on Unconventional Programming Paradigms, UPP'04, pp. 257–269. Springer-Verlag, Berlin (2005)
312. Parsons, S., Bahsoon, R., Lewis, P.R., Yao, X.: Towards a better understanding of self-awareness and self-expression within software systems. Tech. Rep. CSR-11-03, University of Birmingham, School of Computer Science, UK (2011)
313. Paul, C., Bass, L., Kazman, R.: Software Architecture in Practice. MA: Addison-Wesley (1998)
314. Paul, C., Kazman, R., Klein, M.: Evaluating Software Architectures: Methods and Case Studies. Addison-Wesley (2002)
315. Paulson, L.: DARPA creating self-aware computing. IEEE Computer **36**(3), 24 (2003). DOI 10.1109/MC.2003.1185213
316. Peleg, A., Wilkie, S., Weiser, U.C.: Intel MMX for multimedia pcs. Communications of the ACM **40**(1), 24–38 (1997)
317. Perkowitz, M., Philipose, M., Fishkin, K., Patterson, D.J.: Mining models of human activities from the web. In: Proceedings of the 13th international conference on World Wide Web, pp. 573–582 (2004)
318. Perrone, M., et al.: Reducing data movement costs: Scalable seismic imaging on blue gene. In: IPDPS, pp. 320–329 (2012)
319. Perrone, M.P., Cooper, L.N.: When networks disagree: Ensemble methods for hybrid neural networks. Neural Networks for Speech and Image Processing, Chapman-Hall, New York pp. 126–142 (1993)
320. Peskin, C.S.: Mathematical aspects of heart physiology. Courant Institute of Mathematical Sciences, New York University New York (1975)
321. Pflugfelder, R., Bischof, H.: People Tracking across Two Distant Self-calibrated Cameras. In: Proceedings of International Conference on Advanced Video and Signal based Surveillance. IEEE Computer Society Press (2006)
322. Pflugfelder, R., Bischof, H.: Tracking across Non-overlapping Views Via Geometry. In: Proceedings of the International Conference on Pattern Recognition (2008)
323. Phelps, S., Mcburney, P., Parsons, S.: Evolutionary mechanism design: A review. Autonomous Agents and Multi-Agent Systems **21**(2), 237–264 (2010)
324. Piciarelli, C., Esterle, L., Khan, A., Rinner, B., Foresti, G.: Dynamic reconfiguration in camera networks: a short survey. IEEE Transactions on Circuits and Systems for Video Technology **PP**(99), 1–1 (2015). DOI 10.1109/TCSVT.2015.2426575. To appear
325. Polikar, R.: Ensemble based systems in decision making. IEEE Circuits and Systems Magazine **6**(3), 21–45 (2006)
326. Polikar, R., Udpa, L., Udpa, S., Honavar, V.: Learn++: An incremental learning algorithm for supervised neural networks. IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews **31**(4), 497–508 (2001)
327. P.T.Eugster, P.A.Felber, R.Guerraoui, A.M.Kermarrec: The many faces of publish/subscribe. ACM Computing Surveys **35**, 114–131 (2003)
328. Puckette, M.: Pure Data (Pd) (software). URL `\url{http://puredata.info}`
329. Pylvänäinen, T.: Accelerometer based gesture recognition using continuous hmms. In: Pattern Recognition and Image Analysis, pp. 639–646. Springer (2005)
330. Quaritsch, M., Kreuzthaler, M., Rinner, B., Bischof, H., Strobl, B.: Autonomous Multicamera Tracking on Embedded Smart Cameras. EURASIP Journal on Embedded Systems Volume 2007 **2007**(1), 35–45 (2007)

331. Rajko, S., Qian, G., Ingalls, T., James, J.: Real-time gesture recognition with minimal training requirements and on-line learning. In: Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, pp. 1–8. IEEE (2007)
332. Ramamurthy, S., Bhatnagar, R.: Tracking recurrent concept drift in streaming data using ensemble classifiers. In: Proceedings of the Sixth International Conference on Machine Learning and Applications (ICMLA), pp. 404–409. Cincinnati, Ohio (2007)
333. Rammer, I., Szpuszta, M.: Advanced. NET Remoting. Springer (2005)
334. Ramos, C., Augusto, J.C., Shapiro, D.: Ambient intelligence - the next step for artificial intelligence. Intelligent Systems, IEEE **23**(2), 15–18 (2008). DOI 10.1109/MIS.2008.19
335. Rasmussen, C., Williams, C.: Gaussian Processes for Machine Learning. MIT Press (2006)
336. Reason [computer software]. URL `\url{https://www.propellerheads.se}`
337. ReconOS: A programming model and OS for reconfigurable hardware (2013)
338. Reisslein, M., Rinner, B., Roy-Chowdhury, A.: Smart camera networks. IEEE Computer **47**(5), 26–28 (2014)
339. Reyes, R., de Sande, F.: Automatic code generation for gpus in llc. The Journal of Supercomputing **58**(3), 349–356 (2011)
340. Richter, U., Mnif, M., Branke, J., Müller-Schloer, C., Schmeck, H.: Towards a generic observer/controller architecture for organic computing. In: C. Hochberger, R. Liskowsky (eds.) INFORMATIK 2006 – Informatik für Menschen, *LNI*, vol. P-93, pp. 112–119. Bonner Köllen Verlag (2006)
341. Rietmann, M., Messmer, P., Nissen-Meyer, T., Peter, D., Basini, P., Komatitsch, D., Schenk, O., Tromp, J., Boschi, L., Giardini, D.: Forward and adjoint simulations of seismic wave propagation on emerging large-scale GPU architectures. In: SC Conference on High Performance Computing Networking, Storage and Analysis, SC '12, Salt Lake City, UT, USA - November 11 - 15, 2012, pp. 38, Owner = Xinyu, Timestamp = 2015.05.07 (2012)
342. Rinner, B., Esterle, L., Simonjan, J., Nebehay, G., Pflugfelder, R., Fernandez, G., Lewis, P.R.: Self-Aware and Self-Expressive Camera Networks. Computer **48**(7), 33–40 (2015)
343. Rinner, B., Winkler, T., Schriebl, W., Quaritsch, M., Wolf, W.: The evolution from single to pervasive smart cameras. In: Proceedings of the Second ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC), pp. 1–10. IEEE Computer Society Press (2008). DOI 10.1109/ICDSC.2008.4635674
344. Rinner, B., Wolf, W.: Introduction to Distributed Smart Cameras. Proceedings of the IEEE **96**(10), 1565–1575 (2008). DOI 10.1109/JPROC.2008.928742
345. RNA: Recursive Network Architecture. URL `www.isi.edu/rna`. Website: `www.isi.edu/rna`, (accessed January 2015)
346. Rochat, P.: Five levels of self-awareness as they unfold in early life. Consciousness and Cognition **12**, 717–731 (2003)
347. Russell, S.J., Norvig, P.: Artificial Intelligence - A Modern Approach, 3 edn. Pearson Education (2010)
348. Saaty, T.L.: The Analytical Hierarchical Process. McGraw-Hill (1980)
349. Sakellari, G.: The cognitive packet network: A survey. The Computer Journal **53** (2010)
350. SanMiguel, J.C., Shoop, K., Cavallaro, A., Micheloni, C., Foresti, G.L.: Self-Reconfigurable Smart Camera Networks. IEEE Computer **47**(5), 67–73 (2014)
351. Santambrogio, M., Hoffmann, H., Eastep, J., Agarwal, A.: Enabling technologies for self-aware adaptive systems. In: 2010 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), pp. 149–156. IEEE (2010)
352. Santner, J., Leistner, C., Saffari, A., Pock, T., Bischof, H.: PROST: Parallel robust online simple tracking. In: Computer Vision and Pattern Recognition, pp. 723–730 (2010)
353. Schaumeier, J., Jeremy Pitt, J., Cabri, G.: A tripartite analytic framework for characterising awareness and self-awareness in autonomic systems research. In: Self-Adaptive and Self-Organizing Systems Workshops (SASOW), 2012 Sixth IEEE Conference on, pp. 157–162 (2012)
354. Schlömer, T., Poppinga, B., Henze, N., Boll, S.: Gesture recognition with a wii controller. In: Proceedings of the 2nd international conference on Tangible and embedded interaction, pp. 11–14. ACM (2008)

355. Schmeck, H.: Organic computing - a new vision for distributed embedded systems. In: Object-Oriented Real-Time Distributed Computing (ISORC), Eighth IEEE International Symposium on, pp. 201–203. IEEE (2005)
356. Schmickl, T., Thenius, R., Moslinger, C., Timmis, J., Tyrrell, A., Read, M., Hilder, J., Halloy, J., Campo, A., Stefanini, C., et al.: Cocoro–the self-aware underwater swarm. In: Proc. Int. Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW), pp. 120–126. IEEE Computer Society, Ann Arbor, MI, USA (2011)
357. Schnier, T., Yao, X.: Using negative correlation to evolve fault-tolerant circuits. In: Proceedings of the 5th International Conference on Evolvable Systems: From Biology to Hardware (ICES'2003) – Lecture Notes in Computer Science, vol. 2606, pp. 35–46. Springer-Verlag (2003)
358. Scholz, M., Klinkenberg, R.: Boosting classifiers for drifting concepts. Intelligent Data Analysis **11**(1), 3–28 (2007)
359. Sharan, K.: Java remote method invocation. In: Beginning Java 8 APIs, Extensions and Libraries, pp. 525–548. Springer (2014)
360. Shaw, M.J., Sikora, R.: A distributed problem-solving approach to inductive learning. Tech. Rep. CMU-RI-TR-90-262, School of Computer Science, Carnegie Mellon University (1990)
361. Shipp, C.A., Kuncheva, L.I.: Relationships between combination methods and measures of diversity in combining classifiers. Information Fusion **3**(2), 135–148 (2002)
362. Showerman, M., et al.: QP: A heterogeneous multi-acceleator cluster. In: Proc. ICHPCC (2009)
363. Shukla, S.K., Yang, Y., Bhuyan, L.N., Brisk, P.: Shared memory heterogeneous computation on pcie-supported platforms. In: 23rd International Conference on Field programmable Logic and Applications, FPL 2013, Porto, Portugal, September 2-4, 2013, pp. 1–4 (2013)
364. Simonjan, J., Esterle, L., Rinner, B., Nebehay, G., Dominguez, G.F.: Demonstrating autonomous handover in heterogeneous multi-camera systems. In: Proceedings of the International Conference on Distributed Smart Cameras, pp. 43:1–43:3 (2014). DOI 10.1145/2659021.2669474
365. Sironi, F., Bartolini, D.B., Campanoni, S., Cancare, F., Hoffmann, H., Sciuto, D., Santambrogio, M.D.: Metronome: Operating System Level Performance Management via Self-adaptive Computing. In: Proc. Design Automation Conference (DAC). ACM (2012)
366. Sironi, F., Cuoccio, A., Hoffmann, H., Maggio, M., Santambrogio, M.: Evolvable Systems on Reconfigurable Architecture via Self-aware Adaptive Applications. In: NASA/ESA Conference on Adaptive Hardware and Systems (AHS) (2011). DOI 10.1109/AHS.2011.5963933
367. Sironi, F., Triverio, M., Hoffmann, H., Maggio, M., Santambrogio, M.: Self-aware Adaptation in FPGA-based Systems. In: Int. Conf. on Field Programmable Logic and Applications. IEEE (2010)
368. Smallwood, J., McSpadden, M., Schooler, J.: The lights are on but no one's home: meta-awareness and the decoupling of attention when the mind wanders. Psychonomic Bulletin and Review **14**(3), 527–533 (2007)
369. Song, S., Chandra, A., Torresen, J.: An ant learning algorithm for gesture recognition with one-instance training. In: Proc. Int. Congr. on Evolutionary Computation (CEC), pp. 2956–2963. IEEE (2013)
370. SRC Computers, L.: Src-7 mapstation. Tech. rep., SRC Computers, LLC (2009)
371. Srinivas, N., Deb, K.: Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Computation **2**(3), 221–248 (1994)
372. Stanley, K.O.: Learning concept drift with a committee of decision trees. Tech. Rep. UT-AI-TR-03-302, Department of Computer Sciences, University of Texas at Austin (2003)
373. Sterritt, R., Parashar, M., Tianfield, H., Unland, R.: A concise introduction to autonomic computing. Advanced Engineering Informatics **19**(3), 181–187 (2005)
374. Steuer, R.E., Choo, E.U.: An interactive weighted tchebycheff procedure for multiple objective programming. Mathematical Programming **26**(3), 326–344 (1983)
375. Stone, P.: Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer. MIT Press (2000)

376. Strassen, V.: Gaussian elimination is not optimal. Numerische Mathematik pp. 13:354–356 (1969)
377. Street, W., Kim, Y.: A streaming ensemble algorithm (SEA) for large-scale classification. In: Proceedings of the Seventh ACM International Conference on Knowledge Discovery and Data Mining (KDD), pp. 377–382. ACM Press, New York (2001)
378. Strenski, D.: The cray xd1 computer and its reconfigurable architecture. Tech. rep., Cray Inc. (2005)
379. Strey, A., Bange, M.: Performance analysis of intel's MMX and SSE: A case study. In: Proceedings of 7th International Euro-Par Conference on Parallel Processing (Euro-Par), pp. 142–147. Manchester, UK (2001)
380. Susanto, K.W., Todman, T., Coutinho, J.G.F., Luk, W.: Design validation by symbolic simulation and equivalence checking: A case study in memory optimization for image manipulation. In: SOFSEM, *LNCS*, vol. 5404, p. 509–520. Springer (2009)
381. Sutter, H.: The free lunch is over: A fundamental turn toward concurrency in software. Dr. Dobb's Journal (2005)
382. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)
383. Taj, M., Cavallaro, A.: Distributed and decentralized multi-camera tracking. Signal Processing Magazine **28**(3), 46–58 (2011)
384. Tawney, G.A.: Feeling and self-awareness. Psyc. Rev. **9**(6), 570 – 596 (1902)
385. Tesauro, G.: Reinforcement learning in autonomic computing: A manifesto and case studies. IEEE Internet Computing **11**(1), 22–30 (2007)
386. Thomas, D., Luk, W.: Non-uniform random number generation through piecewise linear approximations. In: Proc. FPL, pp. 1—-6 (2006)
387. Thomas, D.B., Luk, W.: Credit risk modelling using hardware accelerated monte-carlo simulation. In: 16th IEEE International Symposium on Field-Programmable Custom Computing Machines, FCCM 2008, 14-15 April 2008, Stanford, Palo Alto, California, USA, pp. 229–238 (2008)
388. Todman, T., Boehm, P., Luk, W.: Verification of streaming hardware and software codesigns. In: Proc. Int. Conf. on Field Programmable Technology (ICFPT), pp. 147–150. IEEE (2012)
389. Todman, T., Stilkerich, S.C., Luk, W.: Using statistical assertions to guide self-adaptive systems. International Journal of Reconfigurable Computing **2014**, 724,585.1–8 (2014). DOI 10.1155/2014/724585
390. Tong, X., Ngai, E.: A ubiquitous publish/subscribe platform for wireless sensor networks with mobile mules. In: Proceedings of the IEEE Eighth International Conference on Distributed Computing in Sensor Systems (DCOSS), 2012, pp. 99–108. IEEE (2012)
391. Torresen, J., Hafting, Y., Nymoen, K.: A new Wi-Fi based platform for wireless sensor data collection. In: Proceedings of the International Conference on New Interfaces for Musical Expression, pp. 337–340 (2013)
392. Touch, J., Pingali, V.: The RNA Metaprotocol. In: Proceedings of the International Conference on Computer Communications and Networks, pp. 1–6 (2008). DOI 10.1109/ICCCN.2008.ECP.46
393. Trucco, E., Plakas, K.: Video Tracking: A Concise Survey. Journal of Oceanic Engineering **31**(2), 520–529 (2006)
394. Tse, A.H.T., Chow, G.C.T., Jin, Q., Thomas, D.B., Luk, W.: Optimising performance of quadrature methods with reduced precision. In: Proc. Int. Conf. on Reconfigurable Computing: Architectures, Tools and Applications (ARC), *Lecture Notes in Computer Science*, vol. 7199, pp. 251–263. Springer, Berlin / Heidelberg (2012). DOI 10.1007/978-3-642-28365-9_21
395. Tse, A.H.T., Thomas, D.B., Tsoi, K.H., Luk, W.: Dynamic scheduling monte-carlo framework for multi-accelerator heterogeneous clusters. In: Proceedings of the International Conference on Field-Programmable Technology, FPT 2010, 8-10 December 2010, Tsinghua University, Beijing, China, pp. 233–240 (2010)
396. Tsoi, K.H., Luk, W.: Axel: A heterogeneous cluster with fpgas and gpus. In: Proc. FPGA, pp. 115–124 (2010)

397. Tsymbal, A., Pechenizkiy, M., Cunningham, P., Puuronen, S.: Dynamic integration of classifiers for handling concept drift. Information Fusion **9**(1), 56–68 (2008)
398. Vassev, E., Hinchey, M.: Knowledge representation and awareness in autonomic service-component ensembles – state of the art. In: 14th IEEE International Symposium on Object/Component/Service-oriented Real-time Distributed Computing, pp. 110–119. IEEE Computer Society (2011)
399. Vasudevan, S.: What is assertion-based verification? SIGDA E-News **42**(12) (2012)
400. Vermorel, J., Mohri, M.: Multi-Armed Bandit Algorithms and Empirical Evaluation. In: Proceedings of the European Conference on Machine Learning, pp. 437–448. Springer (2005)
401. Vickrey, W.: Counterspeculation, auctions, and competitive sealed tenders. The Journal of Finance **16**(1), 8–37 (1961)
402. Vinoski, S.: Corba: Integrating diverse applications within distributed heterogeneous environments. IEEE Communications Magazine **35**(2), 46–55 (1997)
403. Volker, L., Martin, D., El Khayaut, I., Werle, C., Zitterbart, M.: A Node Architecture for 1000 Future Networks. In: IEEE International Conference onCommunications (ICC), pp. 1–5 (2009). DOI 10.1109/ICCW.2009.5207996
404. Volker, L., Martin, D., Werle, C., Zitterbart, M., El-Khayat, I.: Selecting Concurrent Network Architectures at Runtime. In: IEEE International Conference on Communications (ICC), pp. 1–5 (2009). DOI 10.1109/ICC.2009.5199445
405. Wang, J., Brady, D., Baclawski, K., Kokar, M., Lechowicz, L.: The use of ontologies for the self-awareness of the communication nodes. In: Proceedings of the Software Defined Radio Technical Conference SDR, vol. 3 (2003)
406. Wang, S., Minku, L.L., Yao, X.: A learning framework for online class imbalance learning. In: Proceedings of the IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL), pp. 36–45 (2013)
407. Wang, S., Minku, L.L., Yao, X.: Online class imbalance learning and its applications in fault detection. International Journal of Computational Intelligence and Applications **12**(4), 1340,001(1–19) (2013)
408. Wang, S., Minku, L.L., Yao, X.: A multi-objective ensemble method for online class imbalance learning. In: Proceedings of the 2014 International Joint Conference on Neural Networks (IJCNN), pp. 3311–3318. IEEE (2014). DOI 10.1109/IJCNN.2014.6889545
409. Wang, S., Minku, L.L., Yao, X.: Resampling-based ensemble methods for online class imbalance learning. In: IEEE Transactions on Knowledge and Data Engineering (TKDE), vol. 27, pp. 1356–1368. IEEE (2015). DOI 10.1109/TKDE.2014.2345380
410. Wang, Z., Tang, K., Yao, X.: A memetic algorithm for multi-level redundancy allocation. IEEE Transactions on Reliability **59**(4), 754–765 (2010)
411. Watson, R.: The Delta-t Transport Protocol: Features and Experience. In: Local Computer Networks, 1989., Proceedings 14th Conference on, pp. 399–407 (1989). DOI 10.1109/LCN.1989.65288
412. Werner-Allen, G., Tewari, G., Patel, A., Welsh, M., Nagpal, R.: Firefly-inspired sensor network synchronicity with realistic radio effects. In: Proceedings of the 3rd international conference on Embedded networked sensor systems, pp. 142–153 (2005)
413. Weyns, D., Schmerl, B., Grassi, V., Malek, S., Mirandola, R., Prehofer, C., Wuttke, J., Andersson, J., Giese, H., Gäschka, K.M.: On patterns for decentralized control in self-adaptive systems. In: R. Lemos, H. Giese, H. Müller, M. Shaw (eds.) Software Engineering for Self-Adaptive Systems II, *Lecture Notes in Computer Science*, vol. 7475, pp. 76–107. Springer Berlin Heidelberg (2013)
414. Wikipedia: Wikipedia– adaptation (computer science). `http://en.wikipedia.org/wiki/Adaptation(CHANGEURL)` (Accessed in December 2014)
415. Winfield, A.: Robots with internal models: a route to self-aware and hence safer robots. In: J. Pitt (ed.) The Computer After Me. Imperial College Press / World Scientific Book (2014)
416. Wirsing, M., Hölzl, M., Koch, N., Mayer, P.: Software Engineering for Collective Autonomic Systems: The ASCENS Approach, *Lecture Notes in Computer Science*, vol. 8998. Springer (2015)

417. Wolf, W., Ozer, B., Lv, T.: Smart Cameras as Embedded Systems. IEEE Computer **35**(9), 48–53 (2002)
418. Wright, M.: Open Sound Control: an enabling technology for musical networking. Organised Sound **10**(3), 193–200 (2005)
419. Xiao, L., Zhu, Y., Ni, L., Xu, Z.: Gridis: An incentive-based grid scheduling. In: Proceedings of the 19th IEEE InternationalParallel and Distributed Processing Symposium, p. 65b (2005). DOI 10.1109/IPDPS.2005.237
420. Xilinx: Sdaccel development environment. `http://www.xilinx.com/products/design-tools/sdx/sdaccel.html`. [Online; accessed 2-April-2015]
421. Y. Jin et al: A framework for evolutionary optimization with approximate fitness functions. IEEE Transactions on Evolutionary Computation **6**(5), 481—-494 (2002)
422. Ye, J., Dobson, S., McKeever, S.: Situation identification techniques in pervasive computing: A review. Pervasive and Mobile Computing **8**(1) (2012)
423. Yiannacouras, P., Steffan, J.G., Rose, J.: VESPA: portable, scalable, and flexible fpga-based vector processors. In: Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, pp. 61–70 (2008)
424. Yilmaz, A., Javed, O., Shah, M.: Object Tracking: A Survey. Computing Surveys **38**(4) (2006)
425. Yin, F., D., M., Velastin, S.: Performance evaluation of object tracking algorithms. In: Proceedings of the International Workshop on Performance Evaluation of Tracking and Surveillance (2007)
426. Yin, L., Dong, M., Duan, Y., Deng, W., Zhao, K., Guo, J.: A high-performance training-free approach for hand gesture recognition with accelerometer. Multimedia Tools and Applications pp. 1–22 (2013)
427. Yu, X., Tang, K., Chen, T., Yao, X.: Empirical analysis of evolutionary algorithms with immigrants schemes for dynamic optimization. Memetic Computing **1**(1), 3–24 (2009)
428. Zadeh, L.: Optimality and non-scalar-valued performance criteria. IEEE Transactions on Automatic Control **8**(1), 59–60 (1963)
429. Zagal, J.C., Lipson, H.: Towards self-reflecting machines: Two-minds in one robot. In: Advances in Artificial Life. Darwin Meets von Neumann, *Lecture Notes in Computer Science*, vol. 5777, pp. 156–164. Springer Berlin Heidelberg (2011)
430. Zambonelli, F., Bicocchi, N., Cabri, G., Leonardi, L., Puviani, M.: On self-adaptation, self-expression, and self-awareness in autonomic service component ensembles. In: Self-Adaptive and Self-Organizing Systems Workshops (SASOW), 2011 Fifth IEEE Conference on, pp. 108 –113 (2011)
431. Zarezadeh, A.A., Bobda, C.: Hardware Middleware for Person Tracking on Embedded Distributed Smart Cameras. Hindawi International Journal of Reconfigurable Computing (2012)
432. Zeppenfeld, J., Bouajila, A., Stechele, W., Bernauer, A., Bringmann, O., Rosenstiel, W., Herkersdorf, A.: Applying ASoC to Multi-core Applications for Workload Management. In: C. Müller-Schloer, H. Schmeck, T. Ungerer (eds.) Organic Computing — A Paradigm Shift for Complex Systems, *Autonomic Systems*, vol. 1, pp. 461–472. Springer Basel (2011)
433. Zhou, A., Qu, B.Y., Li, H., Zhao, S.Z., Suganthan, P.N., Zhang, Q.: Multiobjective evolutionary algorithms: A survey of the state of the art. Swarm and Evolutionary Computation **1**(1), 32–49 (2011)
434. Ziliani, F., Velastin, S., Porikli, F., Marcenaro, L., Kelliher, T., Cavallaro, A., Bruneaut, P.: Performance evaluation of event detection solutions: the CREDS experience. In: Proceedings of the International Conference on Advanced Video and Signal Based Surveillance, pp. 201–206 (2005)
435. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithm: Empirical results. Evolutionary Computation **8**(2), 173–195 (2000)
436. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: improving the strength pareto evolutionary algorithm. Tech. Rep. 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich (2001)

437. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation **3**(4), 257–271 (1999)
438. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on Evolutionary Computation **7**(2), 117–132 (2003)