DR 3.1: Personalized adaptive action selection Y1 Report

Yiannis Demiris, Antoine Cully, Maxime Petit

Personal Robotics Laboratory, Department of Electrical and Electronic Engineering, Imperial College London, South Kensington Campus London SW7 2AZ, UK
⟨{y.demiris, a.cully, m.petit }@imperial.ac.uk⟩

Project, project Id:	EU H2020 PAL / PHC-643783
Project start date:	March 1 2015 (48 months)
Due date of deliverable:	February 29, 2016
Actual submission date:	February 29, 2016
Lead partner:	Imperial College London
Revision:	final
Dissemination level:	PU

This document reports the work achieved during the first year of the PAL project on the *Personalized adaptive action selection* work package (WP3), and in particular on the first prototype of the action selection computational algorithms.

1 Executive Summary

This document describes the first prototype of the PAL system's action selection mechanisms developed during the first year of the project in work-package 3.

The overall objective of the Work Package 3 is to personalize the behaviour of the PAL system (typically, the robot or its avatar) to each of its users. The adaptation of the system's behaviour to each user has two objectives: 1) increasing the engagement of the user in the PAL system, and 2) allowing the user to reach more effectively its personal goal(s) by adapting to his/her preferences.

Following the workplan, during this first year of work, we have put in place an action selection architecture, which allows us to experiment with and combine different predictive user models (encompassing user characteristics, preferences, level of skills or knowledge) and potential actions, in order to rapidly personalize the behavior of the system to the needs of a specific user.

This document first presents the state of the art of action selection and personalisation approaches in human-robot interaction. In particular, this review highlights the need for considering at the same time personalization (to fit the user's preferences) and adaptation (to follow the changes in the user's preferences over the time). Work package 3 aims to address both of these features in a single action selection framework, which is using as inputs the desired goals and the user cognitive state that are determined by modules in Work package 2 and the potential actions that are suggested by the interaction module, developed in Work package 4. The first prototype of this action selection module is based on the HAMMER architecture (Hierarchical Attentive Multiple Models for Execution and Recognition), developed by Imperial College. This architecture is described in the first section of this report, while subsequent sections detail the features of the implementation we made for this deliverable, and how does it fit in the overall PAL system architecture.

In the two last sections of this document, we describe the first two sets of experiments that have been performed with this first prototype: 1) Validation of the implementation and 2) Stress test of the implementation. Based on the presented results, we demonstrate that the architecture successfully works with several concurrent models and manages to rapidly identify the most accurate one. The results also show that our current implementation of the architecture is able to simultaneously execute several thousands of concurrent simple models per second allowing the system to consider a large number of alternative actions without affecting the system reactiveness, even if the computational complexity of the underlying models increase greatly. These results demonstrate that this action selection architecture has the computational power and efficiency required for use in the PAL Project.

2 Role of personalized adaptive action selection in PAL

Work Package 3 over the four years of the project will focus on the personalization and adaptation over time of PAL actions to fit the child's individual preferences and needs. The core functionality includes an action selection module (*the topic of this deliverable)*, an action provision module, a sentiment mining module and a predictive user model. As the project progresses, the user model will *learn*, based on an analysis of interaction patterns over time, what specific actions are preferred and most effective for a particular child to reach long-term goals set by modules developed in WP2. Learning takes into account interactions with PAL (WP4), usage statistics of mHealth-Apps and PAL (WP5), multimodal behaviours of the individual child (WP4), and usage context (WP2). The user model influences action selection to reflect the child's personalized needs, while the action selection module integrates this information with high-level current and desired state information provided by the cognitive affective simulation of the child from WP2. Based on the current state of the child and context, the action selection module can probe the child and select the behaviour that is most likely to achieve the given goals in the given context. Over time, PAL learns how to adapt based on the analysis of child activities with PAL and the mHealth-Apps. The end result is that the system selects the most appropriate action among the potential actions suggested by the modules in Work Package 4, by taking into account the goals and information regarding the user cognitive state provided by Work Package 2.

3 Tasks, objectives, results

3.1 Planned work

The objectives of this first deliverable in the Work Package 3 were to provide a first prototype of action selection. To reach this objective, our goals have been:

- To review the current state of the art in action selection and personalisation in human robot interaction in order to see whether recent advances in existing works would solve the challenges that we want to face in the PAL Project.
- To evaluate the HAMMER architecture, developed by Imperial College London, would be a suitable tool for the action selection module.
- To adapt the HAMMER architecture to the particular needs of the PAL system, and examine potential ways that it can be used by other modules in the PAL architecture. For example, one potential avenue for integration would see the system taking as inputs the list of potential actions provided by the Natural Multimodal Interaction module (WP4) and producing as output one of these potential actions back to Interaction module as quickly as possible to allow reactive interaction with the users. The action will be selected using both the desired goal provided by WP2 and the user model of the individual child.
- To implement a first prototype of the action selection , based on the HAMMER architecture. This first version of the module aims to be generic and flexible enough to allow the integration of a large number of different user models and to experiment with a large diversity of algorithms. This will allow us to detect the more relevant data needed to create a useful user model as well as elaborating the most efficient methods to extract their effects on the personalization.
- To conduct performance tests on the first prototype of the action selection module.

The following section will detail the work and the results that have been achieved during the first year of the project.

3.2 Relation to the state-of-the-art

The purpose of Work Package 3 in the PAL project is to provide a *personalized* and *adaptive* action selection framework, For the first year, in the form of Deliverable 3.1, we present a first prototype algorithm for action selection. On one side, the *personalized* aspect of the action selection implies the study of inter-user variability in order to generate a user model which will allows to personalize the interaction, based on personal preferences with respect to, for example learning strategy, level of engagement and communication, among others. On the other side, the *adaptive* aspect is more focused about the changes needed to keep the interaction interesting and effective for a particular user during the long-term usage with the framework, taking into account the improvement of the knowledge or the preference changes of the users, for instance.

Among recent works made toward personalization in human-robot interactions, Karami et al [9] have used interaction traces (robot actions, users' feedback among, etc.) in a restaurant ordering scenario to discover the dependencies between user attributes (age, gender, diabetes, vegetarianism, etc.) and contextual attributes (daytime, season, etc.) with the choices from the menu. They have split the intention estimation problem (using a Hidden Markov Model) and the decision making feature (using a Markov Decision Process). However, the results are not personalized to a specific individual but to a specific group of the population, influenced by contextual informations. As the author point out, they cannot consider very personal attributes, their system therefore can achieve a first good guess about the interaction but will need to be tuned during it.

Baraka and Veloso citeBaraka2015 on the other hand tried to tackle the dynamic modelling aspect, and have developed a personalized system for user temporal preferences. Based on the learning of three user preferences over time ("conservative", "consistent but fatigable", "erratic") using human feedback they have studied the dynamic long-term user preferences, in particular the aspect of boredom and appreciation for change or surprise during repetitive interactions. However, these interactions are solely dependent on the initial clustering of the user preference and cannot be changed during the actual interactions.

More related to a tutoring application, Clabaugh et al. [1] used Learning Styles (LS) informations collected previously from parent and teacher questionnaires about the child as method to differentiate between learners. Tested with 31 preschool children in a Socially Assistive Robotics (SAR) framework, using the NAO and a touch tablet, they have shown that LS is an important factor to predict individuals' performances in certain activities, which might be used to select the ideally challenging ones.

However, an important aspect in tutoring application is actually that the user will learn new skills and knowledge during time: such system need

to be taken into account the progression of the learner to adapt the activities accordingly. Leyzberg et al. [10] developed a Bayesian network for skill assessment using robot's observations of the learner in a puzzle solving scenario consisting of 4 successive games. The robot was able to pause the game three times to provide a short lessons about solving strategies, specifically design to help for one of the weakest skill of the user. They have shown a significant improvement in puzzle solving time for participants who received such personalized lessons as opposed to the one without or with random lessons.

As we have shown, some works has been made in either on the personalization or on the adaptation aspect but a coherent system integrating both of them is rare, especially for physical robotic platform. Tvarozek [13] implemented a socially intelligent computer tutor in a simulated learning environment in order to maintain the student motivation for a relatively long period of time, balancing the action selecting between individual, social, cognitive and affective activities. It is an hybrid system, based on a reinforcement learning approach enhanced with a Wizard of Oz guidance, especially during the first encounter which reduce the need for an initial corpus. It allows a quick construction of a user model and insurance coherent action selection from the start, which will then be gradually optimized with reinforcement learning through the interactions. However, because of the initial Wizard of Oz guidance, this approach needs an expert agent in the loop, implying a non-trivial human participation in the long term (*i.e.* each time for a new user) for the project.

Another recent work is the Conscious-Emotional-Learning Tutoring System (CELTS) from Faghihi et al. [7] which is able to learn from past and present events, at different levels (episodic, emotional and causal learning), adapting its behaviour (e.g. give a direct solution or just an hint?) when teaching how to operate a robotic arm. These learning allows in particular to remember a specific mistake made by a precise user or remember positive or negative sequences of interactions from past experience. However, the system is only designed to optimise the learning for a unique task, instead of providing different teaching activities according to the topic of the desired difficulty.

One of the objectives of the action selection module current being developed in the work-package 3 is to allow both personalized and adaptive action selection. This will be achieved by taking inspiration of the existing works, presented previously, but also by investigating new approaches like fusion and hierarchy of models, knowledge transfer across groups of children or online data-efficient learning algorithms.

3.3 Actual work performed

This section details the work that has been done on the development of the first prototype of action selection. More precisely, the next section present the HAMMER architecture, while the next section describe the principal features of the implementation made for this deliverable. The two following sections present the two first experiments that have been performed and discuss the corresponding results.

3.3.1 The HAMMER architecture

The HAMMER architecture is a versatile parallel distributed hierarchical architecture for action selection developed at Imperial College London [5, 2, 3].

The main component of the HAMMER architecture consists in an *Inverse Model* (see Fig. 1) paired with a *Forward Model* (see Fig. 2)[6, 4]. The Inverse Model suggests an action, which is expected to lead the system (e.g., a robot) from its current state to the target state (or at least closer). This suggested action is then fed into the Forward Model, which predicts the next state of the agent after the execution of the action. The predicted state can then be sent back to the Inverse Model to allow it to adjust the parameters of the action and the difference between the predicted state and the target state can also be used as a reinforcement signal by the Inverse model to improve the quality of its suggestions. When the suggested action is actually performed by the system, the difference between the actual reached state and the prediction is used to update the confidence value of the pair of models. The predicted state can then be used as an input state of the Inverse model in order to make long term predictions by calling sequentially the pair of models.

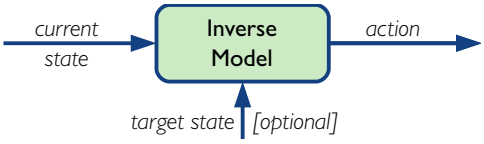

Figure 1: Inverse Model. The role of an inverse model is to suggest an action that will lead the system from its current state to (or closer) the target state.

One of the main properties of HAMMER is to run in parallel several pairs of inverse and forward models (see Fig. 3). Each of these pairs selects an action and predicts the next state and competes against the other pairs in the selection process: only the action suggested by the pair with the highest confidence, as defined above. and which is predicted to lead the system

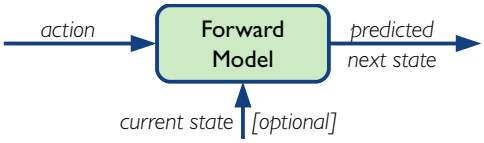

Figure 2: Forward Model. The role of a forward model is to predict the next state of the system based on the current state and the intended action.

closer to the target state is selected by the architecture and executed on by the system. The consequences of this action are measured and used to update the confidence of all the model pairs, for example by considering the distance between the actual state and the states predicted by the model pairs. Based on the updated confidence values and on the new state of the system, the prediction of the model pairs are recomputed and the next action is selected. This process repeats until the system reaches the desired state. In the case the desired state is not reachable, the HAMMER architecture aims to bring the user as close as posible to this state until a more realistic target is defined by the health professionals.

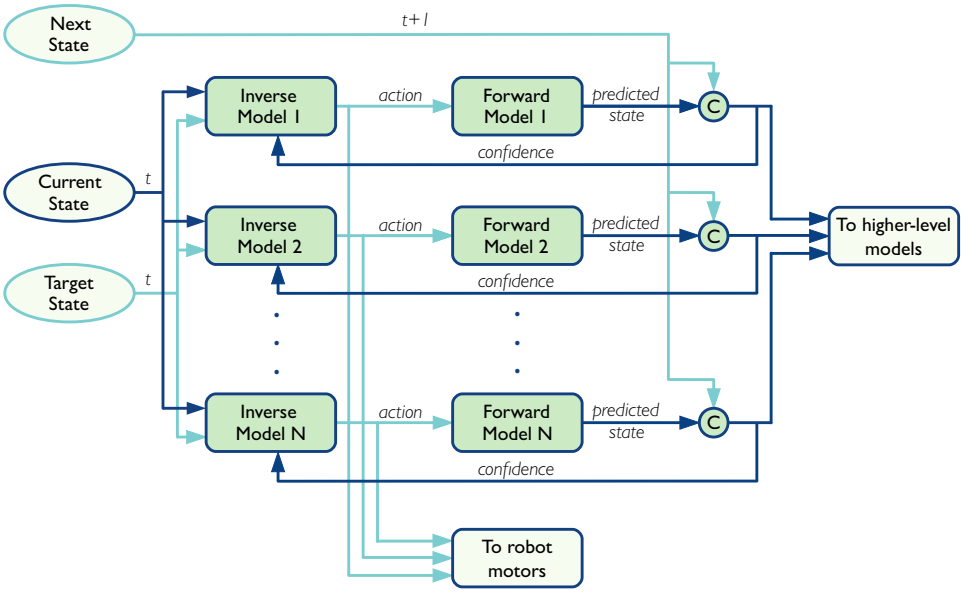

Figure 3: Overview of the HAMMER architecture.

The HAMMER architecture does not impose any constraints on the type of models employed (as long as they follow the same input/output flow) and each pair of models can use different types of models. This property allows the architecture to rely on an heterogeneous set of models that compete

in the action selection phase. This aspect of HAMMER is similar to the *mixture of experts* (or boosting), which is a technique commonly used in supervised learning. It has been proved that the average accuracy of the "council of experts" is better that the average accuracy of its members [8]. This heterogeneous set of models also allows the architecture to personalise the action selection according to the specificities of the system. Indeed, if only a subset of models is relevant with the current situation, the HAMMER architecture will rapidly disregard the other models and focus only on the actions suggested by the pairs of models with a high confidence.

The HAMMER architecture can also observe the decisions made by an agent and can increase the confidence of each model pair accordingly to the similitude between the states predicted by the models and the actual states reached by the agent. Thanks to this behaviour, the architecture can work as an imitation learning algorithm, which is able to reproduce the choices of the observed agent by selecting the model pair with the highest confidence.

This architecture also allows to combine the predictions of the different pairs of models into other pairs of models. This additional set of pairs corresponds to an higher level of hierarchy which are typically involved in long terms predictions or advanced goals. For instance, the pairs of model from the higher level of hierarchy would select the interaction that should be triggered by the agent (the "what", e.g. proposing a quizz for the user to increase its knowledge), while the pairs from the lower levels will determine the parameters (the "how", e.g. the topic of the quizz and the difficulty of the questions). In more complex scenarios, several levels of hierarchy can be considered.

The versatility and flexibility of HAMMER make this architecture particularly suitable for the action selection module that will be used in the PAL system. In particular, this architecture will allow the PAL system to both personalize its behaviour according to the child (inter subject variability) and to adapt to its progress (intra subject changes). Indeed, the heterogeneous set of models will allows the system to rapidly select the most promising models for the current user of the system, while the internal feedbacks within each pair of models will allows them to adapt to the changes in the user's preferences. Conceptually, the HAMMER architecture can be readily adapted to fit in the PAL system. The main modification has been to substitute the inverse models by the list of action proposition that is provided by the Natural Multimodal Interaction module (WP4). Each of the suggested action is forwarded to all the available forward models and the best pair of suggested action and forward model is selected. The state information provided by the child model from GOAL (WP2) is then used to train the models and to progressively personalize the system's behavior. Additional modifications of the HAMMER architecture made for the PAL project are detailed in the next section.

3.3.2 Implementation of the HAMMER architecture

The action selection module that will take place in the PAL system is be based on a new implementation of the HAMMER architecture. This implementation has been developed in JAVA to make easier the integration of the module in the PAL system, as JAVA is one of the main programming languages used in the PAL project. Moreover, the use of JAVA allows the module to be easily installed on any operating system (thanks to the virtual environment). Due to this property, the action selection module will not impose any constraint on the server infrastructure on which the PAL servers will be running during the project.

As mentioned previously, one of the key features of the HAMMER architecture is to execute in parallel several pairs of models. In the designed implementation of the HAMMER architecture, a particular attention has been paid to this aspect in order to take advantage of the multi-core (or multi processors) architecture presents in most of current computers (see Fig. 4). This kind of technology allows a software to execute in parallel several "threads", which each executes a different task. This multi-threading property allows our current implementation of HAMMER to run in parallel several models without severely harming the execution time and the system reactivity. This aspect of our implementation has been evaluated thanks to a stress test that is described in section 3.3.4.

Another feature of the implementation of HAMMER presented in this deliverable is its ability to automatically generating all the possible combinations of inverse and forward models with the different types of model that are implemented in the architecture. While this feature was not technically challenging in its implementation, it allows the users to be agnostic with respect to the different models, and lets HAMMER decide on its own which pairs of models are the most effective.

The action selection module is interfaced with the rest of the PAL system thanks to massage passing library (TECS, designed by PAL partner DFKI). As soon as the action selection module receives a list of potential actions from the Natural Multimodal Interaction module (WP4), the HAMMER architecture is seeded with the different actions and after running the different models, the action selection module sends back a message to the Interaction module with the selected action.

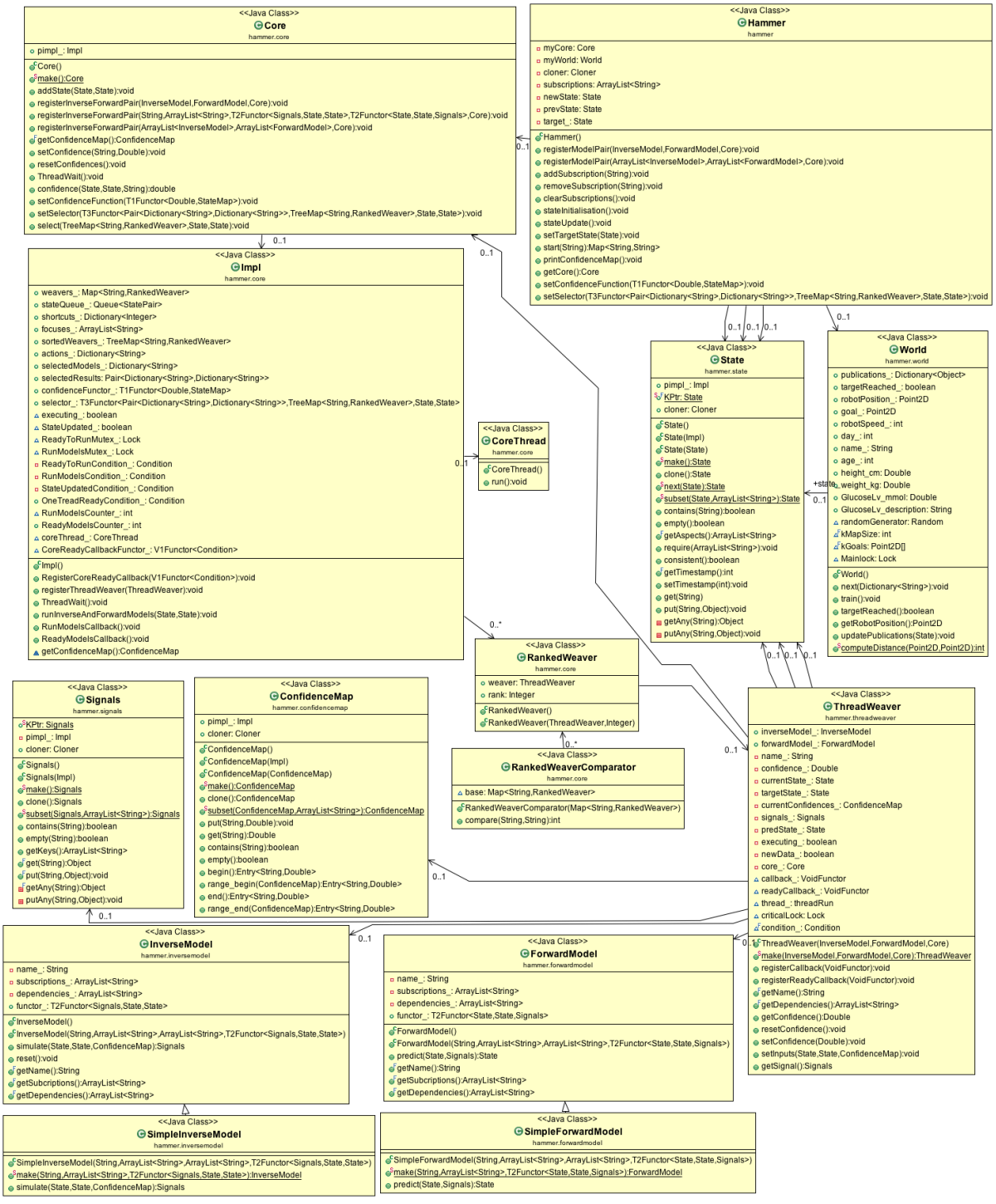

Figure 4: UML diagram of the HAMMER architecture.

3.3.3 Validation of the implementation

In order to validate the core features of our implementation of the HAMMER architecture, we designed an example problem in which the system has to control a numerical value which varies according to the system's choices.At each time step, the system can execute one of the four possible actions: 1) small increase, 2) large increase, 3) small decrease or 4) nothing. The goal in this example problem is to lead the numerical value toward a desired value (in our case: 8), which might represent for instance the insulin level of the child, with the possible action would be to propose a light snack, a proper meal or some exercise.

Three different types of Forward Models are implemented for the purpose of this validation: 1) a random model, 2) a noisy model, and 3) a learnt model. The first type of model always outputs a random prediction, while the noisy model outputs the exact prediction, which is then artificially perturbed by an additive noise. The last model is a regression algorithm which learns to predict the next state of the system. The predictions provided by this model are not always accurate. However, we can expect that the average accuracy of this model is higher than the accuracy of the two other types of models. The implementation of this regression algorithm is based on Gaussian Processes [12]. The model is initialized with 100 random samples and the probabilistic distribution produced by the Gaussian Process is used to predict the next state of the system based on the current state and the proposed action.

In this experiment, only one type of inverse model is used. Nevertheless, this model is couple with the three different forward models in order to constitute 3 different pairs of models. The inverse model is designed to always provide the best action according to the current state of the system and is based on a finite-state machine. We decided to use this kind of model for this validation because the goal of the action selection is mainly to predict the consequences of the suggested actions, and thus to detect the most accurate forward models. Conversely, generating effective Inverse Models is out of the scope of the PAL project, as the action are suggested by the Natural Multimodal Interaction module.

This experiment has been replicated 100 times and the results (see Fig. 5) show that our implementation of HAMMER manages to keep the numerical value close to the desired state. This is not surprising has the inverse model always outputs the best action that should be done. However, this result demonstrates that the architecture works as expected. Moreover, when we consider the selected model at the end of each iteration (see Fig. 6), we can see that our implementation is able to rapidly detect the most accurate pair of models and to select the corresponding suggested action. Indeed, we can observe that after less than 30 iterations, 100% of the different replications are using the regression model.

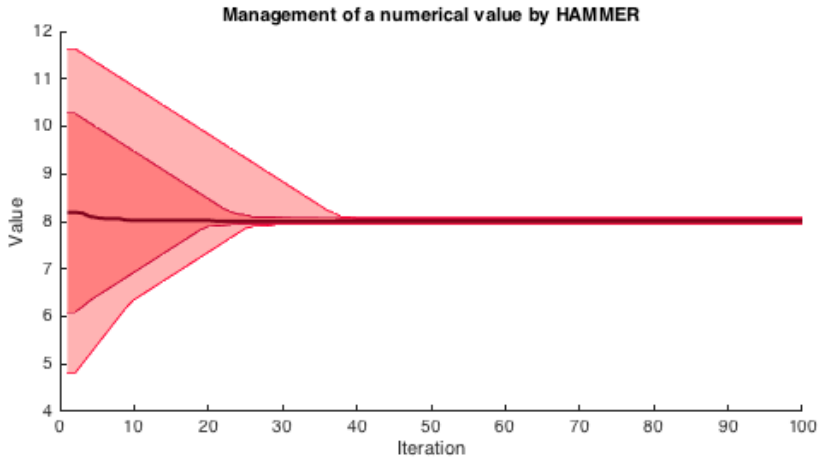

Figure 5: Evolution of the numerical value according to the number of iterations. The target value is 8. The experiment has been replicated 100 times (starting from a random state) to gather statistics. The dark red line represents the median over all the replications. The dark shade area extends to the 25^{th} and 75^{th} percentile, while the bright area extends to the 10^{th} and 90^{th}.

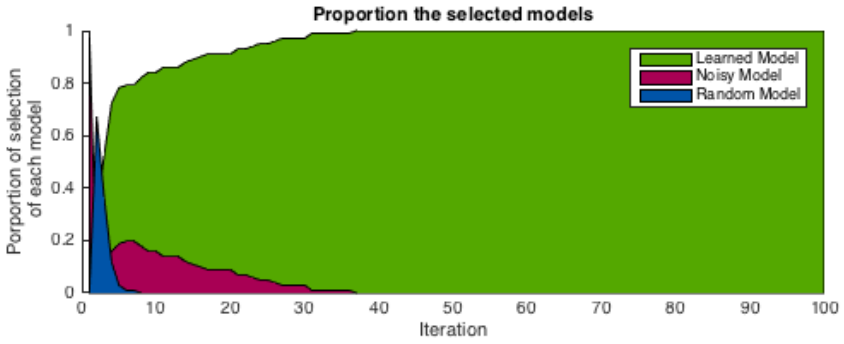

Figure 6: Proportion of the replications selecting each model according to the number of iteration. When the proportion with respect to one model is equal to 1, this means that 100% of the replications selected this model.

3.3.4 Stress test of the implementation

In addition to the validation of the implementation presented in the previous section, the computational capabilities of the architecture have been assessed by running a *stress test*. The goal of this stress test is to evaluate the number of inverse and forward models that can be used simultaneously.

This stress test is based on the same example problem as in the previous

experiment. The only difference is that all the forward models are based on the same type: regression model (GP). This choice has been made to remove the variability in the computational cost among the different models. Therefore, the results of the stress test will not depend on the employed models for the test. However, it is important to note that the global computational cost of HAMMER will be related to the computational cost of the employed models. In other words, using models that are computationally more expensive will slow down the execution of the architecture, while using simpler models will speed up everything. This stress test is of importance because the latency implied by the action selection should be small enough to support real time interaction between the child and the system. This stress test is of importance because the latency implied by the action selection should be small enough to support real time interaction between the child and the system. For instance, while the average human time reaction is 250ms, conversational agents should respond between 100ms, which perceived as an instantaneous reactions, and 1s, which is considered fast enough for users to feel they are interacting freely with the system ([11]).

The performance of the architecture is evaluated by measuring the elapsed time between two action selections (the time required to execute the action can be neglected in this toy problem). During this period, the architecture computes the predictions of each model and selects the one with the highest confidence. In order to see the influence of the number of running model on the computational cost, we executed the architecture with different numbers of running models. The measured execution time is very noisy because of the execution of other processes on the computer. Therefore, we replicated the measures 100 times to gather statistics. The stress test has been executed on a standard desktop computer using an hyper-threaded quad-core processor running at 2.8 GHz (intel core i7) and 8 GB of RAM.

The results of the stress test (see Fig. 7) indicate that at least *1024* simple models can run in parallel without making the computational time intractable (median: 89ms). Indeed, it is very likely that the highest action selection frequency required by the PAL system will be lower than one action selection per second.With the current prototype of action selection, the system can run with more than one thousand simple models and with an action selection frequency one order of magnitude higher. Moreover, the results suggest that the architecture can use up to *64* simple models without any impact on the computational time (median: 30ms).

In order to investigate how our implementation of HAMMER behaves when using computationally more demanding models, we added in the inverse models unnecessary operations in order to artificially increase their computational cost. These operations consist in allocating and generating a large random matrix (500 x 500) and then computing its matrix product with itself and storing the result in a second matrix, which needs to be allocated too. With this modification, each inverse model becomes 4 times

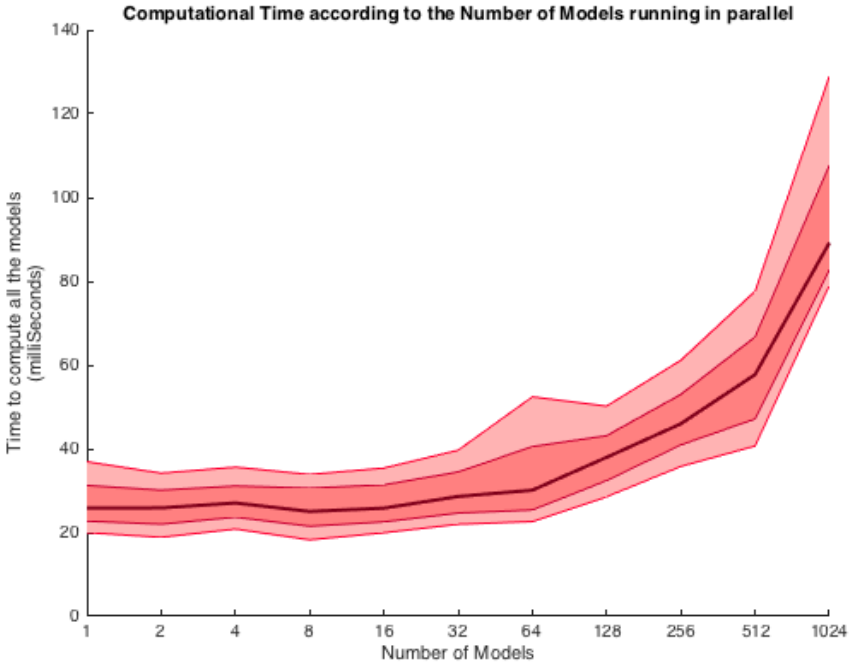

Figure 7: Influence of the number of models running in parallel on the computational time. Each condition has been replicated 100 times to gather statistics and the dark red line represents the median number of milliseconds. The dark shade area extends to the 25^{th} and 75^{th} percentile, while the bright area extends to the 10^{th} and 90^{th}.

computationally more expensive (25ms were required, while 106ms are now required with the modification).

The results of the stress test in this condition (see Fig. 8) show that the parallelization of the model executions works properly, as the influence of the number of running models is negligible when less than 4 models are running in parallel. This result makes sense, as the stress test was running on a quad-code processor. However, we can also observe that running more models increases rapidly the running time. This increase is proportional to the number of models(one can note the logarithmic abscissa) as soon as more than 8 models are running in parallel (slope: 82ms/model).

The conclusion of this stress test is that our current prototype of action selection can work at a decision frequency of one selection per second with either several hundreds of simple models or with a small dozen of computationally demanding models. These results allow us to be relatively optimistic in our future uses of this architecture in the PAL project. Indeed, we can expect that the action selection module will rarely have to deal with more

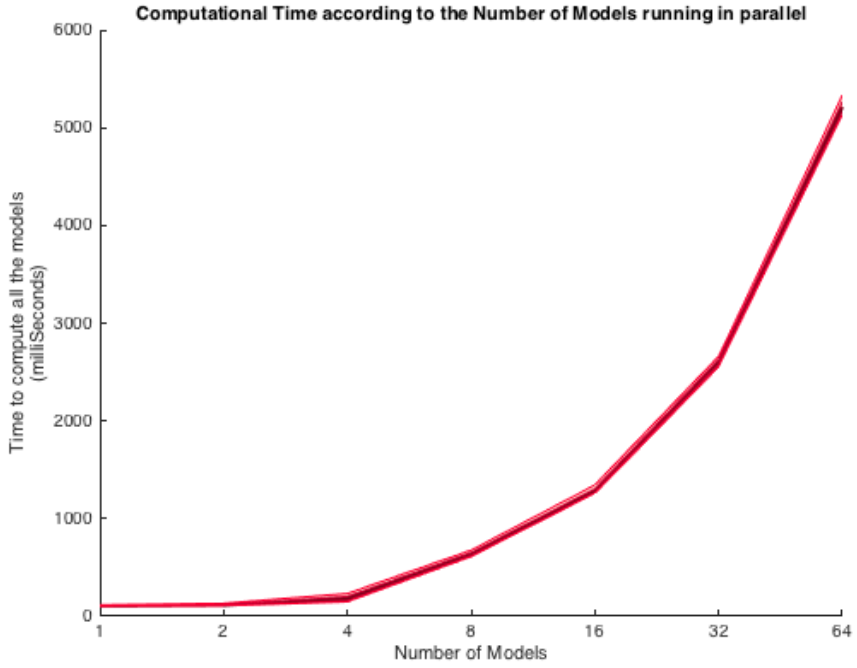

Figure 8: Influence of the number of models running in parallel on the computational time. In contrast with the previous figure, each model has been artificially slow down (with unnecessary operations) in order to simulate the use of computationally expensive models. Each condition has been replicated 100 times to gather statistics and the dark red line represents the median number of milliseconds. The dark shade area extends to the 25^{th} and 75^{th} percentile, while the bright area extends to the 10^{th} and 90^{th}.

than 10 different actions at the same time step (this doesn't restrict the number of actions the PAL system will be able to deal with in total). In such conditions, the action selection module will be able to run several concurrent and heterogeneous models for each action, which is likely to improve its overall accuracy [14, 8].

4 Conclusion

In this report, we described the first prototype of the PAL system's action selection mechanism that has been developed during the first year of the project in the work package 3. The concise review of the state of the art made in this report, highlighted that current methods focus either on the personalization of the action according to the user's preferences or on the adaptation to the changes in the user's preferences over the time. However, only a very limited amount of work has been done to design architectures that cover both of the personalization and adaptation aspects. The action selection module presented in this report is designed to fill this particular gap. In the remaining of the document, we detailed the characteristics of the module that make it suitable for the PAL project. In particular, we detailed how the concurrent execution of heterogeneous pairs of online-adaptable models allows the action selection module to both personalize and adapt the selection with respect to the user and the collected data.This document reports the results of the experimental validations of the module, in which it has been shown the ability of the system to autonomously recognize the most appropriate pairs of models according to the situation and to maintain its reactivity during even when simultaneously considering several hundreds of action hypothesis. These different results demonstrate that the action selection module has the computational power and efficiency required for use in the PAL project.

References

[1] Caitlyn Clabaugh, Gisele Ragusa, Fei Sha, and Maja Mataric. Designing a Socially Assistive Robot for Personalized Number Concepts Learning in Preschool Children. *5th Int Conf on Development and Learning and Epigenetic Robotics*, pages 314–319, 2015.

[2] Yiannis Demiris. Prediction of intent in robotics and multi-agent systems. *Cognitive processing*, 8(3):151–158, 2007.

[3] Yiannis Demiris. Knowing when to assist: Developmental issues in lifelong assistive robotics. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 3357–3360. IEEE, 2009.

[4] Yiannis Demiris* and Matthew Johnson. Distributed, predictive perception of actions: a biologically inspired robotics architecture for imitation and learning. *Connection Science*, 15(4):231–243, 2003.

[5] Yiannis Demiris and Bassam Khadhouri. Hierarchical attentive multiple models for execution and recognition of actions. *Robotics and autonomous systems*, 54(5):361–369, 2006.

[6] Y Derimis and G Hayes. Imitation as a dual-route process featuring predictive and learning components; 4 biologically plausible computational model. *Imitation in animals and artifacts*, page 327, 2002.

[7] Usef Faghihi, Philippe Fournier-viger, and Roger Nkambou. CELTS : A Cognitive Tutoring Agent with Human-Like Learning Capabilities and Emotions. In A.P. Ayala, editor, *Intelligent and Adaptive Educational-Learning Systems: Achievements and Trends*, pages 339–365. Springer, 2013.

[8] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[9] Abir B. Karami, Karim Sehaba, and Benoit Encelle. Learn to adapt based on users' feedback. *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 625–630, 2014.

[10] Daniel Leyzberg, Samuel Spaulding, and Brian Scassellati. Personalizing robot tutors to individuals' learning differences. *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction - HRI '14*, pages 423–430, 2014.

[11] Robert B. Miller. Response time in man-computer conversational transactions. In *Fall Joint Computer Comference*, pages 267–277, 1968.

[12] Carl Edward Rasmussen. *Gaussian processes for machine learning*. Citeseer, 2006.

[13] Jozef Tvarožek. Bootstrapping a Socially Intelligent Tutoring Strategy. *Information Sciences and Technologies Bulletin of the ACM Slovakia*, 3:33–41, 2010.

[14] M Zambelli and Y Demiris. Online ensemble learning of sensorimotor contingencies. 2015.