Chapter 7

Combining learning and planning

Contents

7.1 Motivation

When it comes to decisional process in robotics, two main schools of though can be distinguished: learning and deterministic processes as planing or states machines. Both ways have their advantages and disadvantages. Learning is usually "cheap" (the decision process is quick) and always proposes a solution to a given problem. However, learning requires either a big amount of data or a long period of learning. Moreover, during the learning period, the robot can produce inconsistent behavior which can be confusing for a potential human collaborator. In another hand, planing can take into account humans through social rules and unsure a whole solution

validity. However, planing does not learn from human behavior, and, when it comes to complex tasks or environments, it can become slow to propose a solution. The idea of this work is to propose a solution where we combine planing and learning in the context of human-robot interaction in order to take advantage of both.

This work has been done in collaboration with ISIR at Paris and more particularly with another PhD student Erwan Renaudo. It has been done in the context of the RoboErgoSum ANR project[1]. This work is based on the work of [Renaudo 2014] and has been the subject of a publication in a workshop at the RoMan conference [Renaudo 2015] as well as a part of a journal article [Khamassi 2016].

7.2 Background

7.2.1 Inspiration from neurosciences

Seminal works on living being behaviors have been done in the late 19th century - beginning of the 20th century - with experiments in mammals. One pioneer work concerning the learning process is the experiment of the cat in a box [Thorndike 1998]. In this experiment, a cat is put in a box each time it is hungry. The cat can see food outside of the box and a system of lever allows it to open the box. Each time the cat is put in the box it takes less time to go out. This experiment allows to show the principle of learning through trial and error.

Latter, studies have highlighted two main kinds of behaviors during decision-making tasks. **Goal-directed behaviors** are governed by estimates of action-outcome contingencies (i.e. decision-making relies on the prior estimation of the outcome expected after an action or an action sequence) and are mainly active at the beginning of the task. Then, when the animal is well trained in the task under stable conditions, a transfer of control to **habitual behaviors** governed by stimulus-response associations occurs [Dickinson 1985]. When rodents, monkeys or humans start a new decision-making task, they appear to initially rely on their goal-directed system. They take time to analyse the structure of the task in order to build an internal model of it, and make slow decisions by planning and inferring the long-term consequences of their possible actions before deciding what to do next. Then as their performance gradually improve, they appear to make quicker and quicker decisions, relying on their habitual system which slowly acquires simple stimulus-response associations to solve the task. Finally, when subjects restart to make errors after a task change, they appear to restart planning within their internal model and thus slow down their decision process before acquiring the new task contingencies [Balleine 2010, Dolan 2013]. The coordination of these two learning systems allows mammals to avoid long and costly computations when the environment is sufficiently stable, while still enabling animals to detect environmental changes requiring to update their internal model and replan.

In computational neuroscience models, these behaviors are modeled using

[1] http://roboergosum.isir.upmc.fr/

the theory of Reinforcement Learning [Sutton 1998]: model-based and model-free algorithms provide a direct analogy with goal-directed and habitual behaviors [Daw 2005]. More recently, different computational criteria have been proposed to decide when to shift between model-based and model-free experts [Pezzulo 2013, Lesaint 2014, Viejo 2015]. Applied to neuroscience tasks, the work from [Daw 2005] proposes that the most certain expert gets control on the agent, while [Keramati 2011] balance speed and accuracy using the cost of planning versus the gain of information. A third approach proposes, in the context of navigation strategies, that a coordination module learns by reinforcement the most efficient behavior (in terms of average obtained reward) in each state [Dollé 2010].

7.2.2 Learning in human-robot interaction

A major part of robotics decision-making algorithms are based on planning processes which take into account a great number of information ([Ingrand 2014]). Such approach to decision-making could be seen as similar to what neuroscientists call the goal-directed system, except that there is most of the time no learning in the system. Such approaches have been extended to HRI by taking into account human-aware costs such as social-rules and humans comfort and preferences [Cirillo 2010, Lallement 2014].

Besides, robots learning abilities are still very limited and require the injection of important prior knowledge by the human in the robot's decision-making system. Early applications of reinforcement learning (RL) algorithms to robotics [Hayes 1994, Morimoto 2001, Smart 2002] - some of which being neuro-inspired - produced limited progresses, due to applications to relatively simple problems (with a small number of states and actions), to slowness in learning and to systematic instability observed throughout the learning process. More recent applications of RL to robotics have permitted to deal with more complex and continuous action spaces, enabling to learn efficient sensorimotor primitives [Kober 2011, Martins 2010, Stulp 2013]. These approaches have been extended in HRI to allow robots to learn to collaborate with humans. In several works, the reward signal is interactively assigned by the human [Kaplan 2002, Knox 2012] while other works use the human to provide demonstrations to the robot [Nicolescu 2003, Thomaz 2006]. A method of cross-training is used and compared to standard reinforcement learning algorithms in the context of human-robot teamwork in [Nikolaidis 2013]. Cross-training is an interactive planning method in which a human and a robot iteratively switch roles to learn a shared plan for a collaborative task. Such approaches to decision-making could be seen as similar to what neuroscientists call habitual behaviors.

Even if we can find more and more interesting works in HRI concerning planing and learning for the robot to collaborate with humans, there is no work to our knowledge concerning how to combine both approaches into a robotics architecture.

7.3 Experts presentation

Inspired from from neuroscience theories and based on the previous work of [Renaudo 2014] the aim of this work is to combine goal-directed and habitual behaviors in the context of human-robot Joint Action. To do so, we use two experts which implement these two kinds of behaviors. The goal-directed behavior is produced here by HATP (Human-Aware Task Planer), a task planner which has proved its efficiency in the field of human-robot interaction. A Qlearning algorithm allows to implement the habitual behavior. We will describe in this section these two experts and their respective strengths and weakness. The next sections will show how we combined those two experts into two different architectures.

7.3.1 HATP

In our work, the goal-directed behavior is provide by HATP, an HTN (Hierarchical Task Network, [Erol 1994]) task planner which has been conceived to work in the context of human-robot collaboration. As a HTN planner, HATP uses known preconditions and effects of actions in order to find the best plan that reaches the given goal. It takes as input a list of all possible actions and their description in terms of preconditions and effects and also a description of the current world state as a set of predicates. Then, it looks for the combination of actions that minimizes the solution cost. This cost is computed based on execution time and human-aware costs (e.g the balance of efforts between agents or the waiting time of the human partner). This plan is meant to be executed step by step until the goal is reached. An example of such a plan can be found Fig. 7.1.

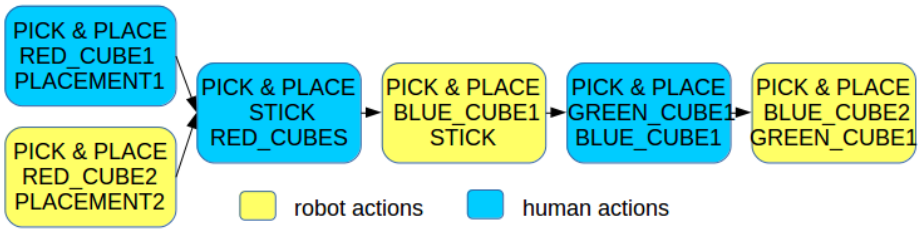


Figure 7.1: An example of a Shared Plan computed by HATP. This plan allows a human and a robot to build a stack of coloured objects together by placing them one on-top of the others.

7.3.2 Qlearning algorithm (MF)

The habitual behavior is provided by a model-free reinforcement learning algorithm (MF) that directly learns the state–action associations by caching in each state the earned rewards in the value of each action. In this implementation, the algorithm is implemented as a neural network (see Fig. 7.2). The network input neurons

represent the different possible states and the output neurons encode the estimation of action values in the current state. The weights are modified to associate each state with the most rewarding action in the current task.

A method similar to [Brafman 2002] is used to compute the value $Qt(St, a_j)$ of an action a_j in a certain state S. $Qt(St, a_j)$ is represented as the scalar product between the input vector and the weights $W_j = (w_{0j}, ..., w_{Nj})$ linking to this action:

$$Qt(St, a_j) = W_j^t \ . \ (S_t, 1)$$

here we set weights at a positive value to provide an initial optimistic estimate of action values ($w_0 = 0.5$). Weights W_t are updated according to the Qlearning algorithm [Watkins 1989]. The Reward Prediction Error δ is spread over the weights of the previously active input and the action a done in the corresponding state:

$$\delta = r_t + \gamma_{Hab} \ . \ \underset{b \in A}{max}(W_b^{t-1}) \ . \ S_t - (W_a^{t-1} \ . \ S_{t-1})$$

$$W_a^t = W_a^{t-1} + \alpha_{Hab} \ . \ \delta / \underset{n}{\sum} s_n$$

with r_t the instant reward received for performing a in S_{t-1} , α_{Hab} the learning rate, γ_{Hab} the decay rate of future rewards. The weights are updated locally: only the state from which the action has been performed is updated. Thus, it requires for the agent to visit every known state of the problem to update values.

7.3.3 Experts comparison

The two different experts have really different ways to decide of the next action to execute. Both methods have their advantages and disadvantages:

- HATP looks for a complete solution to achieve the given goal while the MF only looks for the next action which maximize the probability to get a reward. Consequently, HATP ensures the feasibility of the solution proposed but could find itself in a state where it does not find a valid solution and so where it will not be able to propose an action. In the other hand, the MF does not unsure that its proposed action allows to achieve the goal but will always propose an action to perform.

- As HATP computes a whole plan to achieve the goal, its cost, in the sense of time to take a decision, is far bigger than the one of the MF which only proposes the next action. However, this difference needs to be weighed by the fact that as an HATP plan is composed of several actions, this cost is not needed at each step of the task. Moreover, this cost stay acceptable in a not so complex task.

- HATP is conceived to produce a robot behavior understandable and acceptable by the human. The actions it proposes will produce a consistent behavior of the robot with which one the human can easily collaborate. From its side,

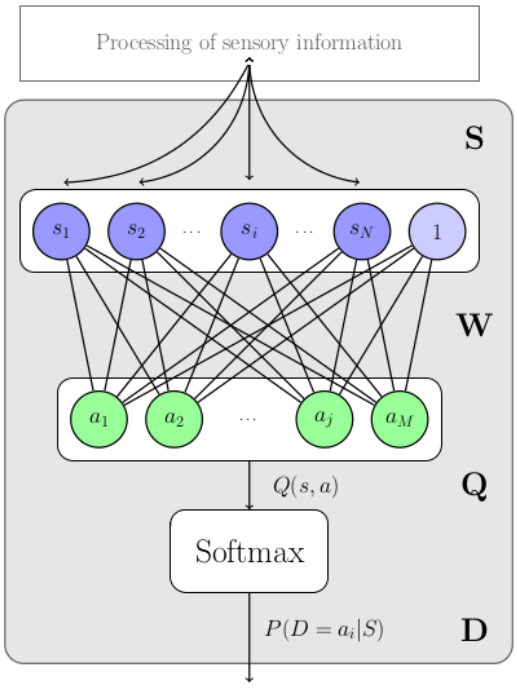


Figure 7.2: Habitual expert, modeled as a Qlearning algorithm implemented as a neural network. The expert receives a state S which is projected onto the input neurons s_i, defining an input activity. This activity is propagated through the network weights W to generate activity of the action layer. This activity corresponds to the values $Q(S, a_j)$, with each neuron coding for a distinct action. This value distribution is converted in probability distribution using a softmax function, which allows the expert to make a decision D on the next action to perform.

the MF have a long period of learning during which one the behavior produced is inconsistent and can be really disturbing for a human collaborating with the robot. Moreover, each time a change happens in the task, a new learning phase is needed. However, the MF is able to learn to adapt its behavior to the human whereas the HATP policy is defined off-line and can not be updated with the behavior of the human during the interaction.

7.4 First architecture: a proof of concept

7.4.1 Control architecture

The first architecture we tried to combine the two experts is the one Fig. 7.3. In this architecture the two experts are placed in parallel. The execution of a task by the architecture follows several steps:

- The Situation Assessment module receives data from perception and maintains the world state representation. This world state is represented with

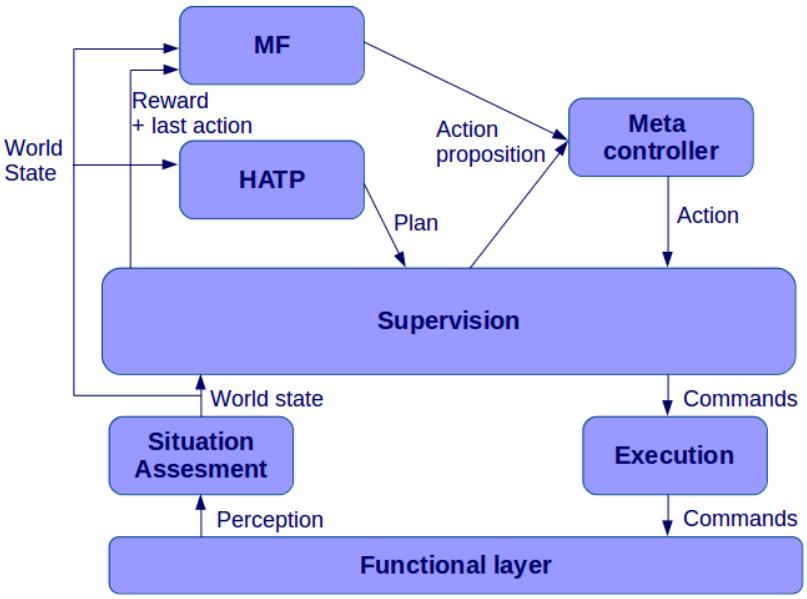


Figure 7.3: First tried architecture to combine the two experts. The Situation Assessment module gets data from perception and maintains the current world state. This world state is used by the supervision to compute the reward and by the experts to take decisions. The propositions of the two experts are sent to the meta controller which decides of the action to execute. The supervisor executes the action with the help of lower execution modules.

predicates (see Sec. 7.4.2).

- The supervisor uses the current world state to compute the reward sent to the MF. This reward is a boolean which is true if the current goal is achieved (see Sec. 7.4.2). The supervisor also sends to the MF the last tried action (which was not necessarily the one proposed by the MF) in order to update the learning.

- The experts decide of the next action to execute based on the current world state. The action proposed by the MF for a given world state is sent directly to the meta controller. Concerning HATP, the supervisor monitors the execution of its plan and sends the next action to execute to the meta controller. A new plan is computed by HATP at the beginning of the task or whenever an unexpected situation happen (an action from the plan fails, the human executes an unexpected action or the robot executes an action proposed by the MF which is not in the current plan).

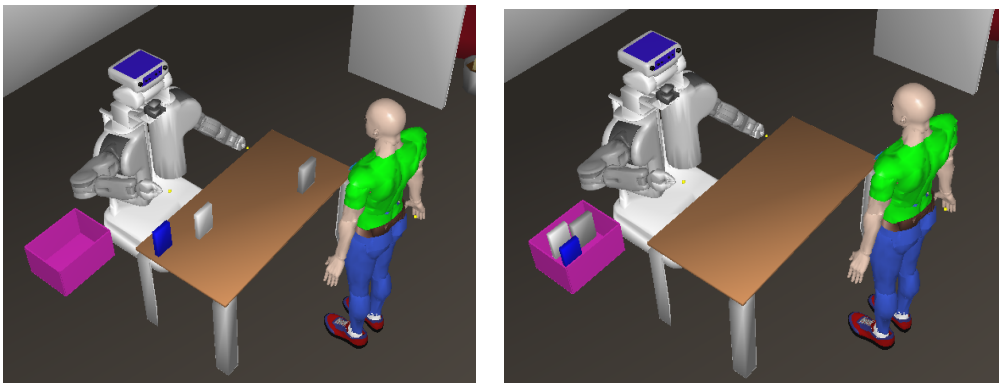
- Once the proposition of action from each expert is received, the meta controller decides which action the robot should execute. In this first implementation the meta controller uses a random arbitration: the action is chosen with an equal probability for each expert.

- The supervisor executes the chosen action with the help of lower execution modules (motion planning, control, ...).

These steps are executed one by one until the goal is achieved.

7.4.2 Task

This first architecture has been tried in a simple task. Moreover, has the learning part of the architecture requires long learning period, the tests have been done in simulation.



(a) Initial situation (b) Final situation

Figure 7.4: Description of the task used with the first architecture. In this task, the human and the robot have to remove all the objects of the table and put them in the pink box. At the beginning of the interaction two objects are accessible only by the robot and another one only by the human. The box is accessible only by the robot.

In the chosen task a human and a robot have to "clean a table" together. To do so, they need to remove all the objects from the table and put them in a box (see Fig. 7.7). At the beginning of the interaction two objects are accessible only by the robot and another one only by the human. The box is accessible only by the robot. To achieve the goal, several actions can be executed by the agents:

- **Pick an object:** both agents can pick an object accessible by them.
- **Throw an object:** the robot can throw an object it has in hand in the box near itself.
- **Give an object:** the robot can give an object to the human.
- **Take an object:** the robot can receive an object from the human
- **Wait:** the robot can wait for the human to execute an action.

All these actions have an impact into the world state. This world state is estimated by the Situation Assessment module and represented with predicates which can be either true or false. For this task, we consider the following predicates:

- **<Object, isReachableBy, Agent>:** these predicates represent for each object if it is reachable by the human or the robot.
- **<Object, isIn, Box>:** these predicates represent the fact that an object has been thrown in the box.
- **<Agent, hasInHand, Object>**: these predicates represent the fact that the human or the robot holds an object.

These predicates allow the experts to take their decisions but also the supervisor to compute the reward needed by the MF. The robot will receive a reward whenever all objects are in the box and it performs the *Wait* action. We chose to impose to the robot to perform a *Wait* action at the end of the task in order for it to learn that the task is over and that no more action is needed.

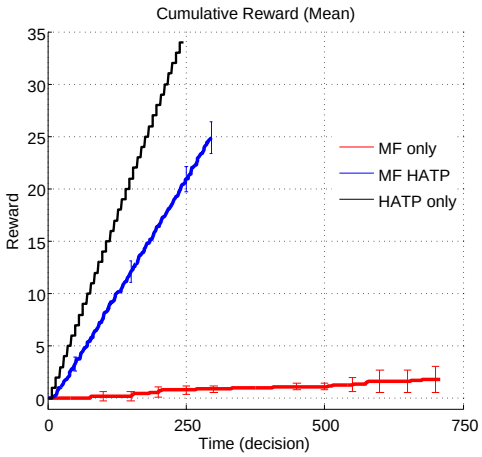
To test our architecture, we compare its performances to the performances of the system running with only the MF and only HATP. We run the experiment in all conditions with a fixed time limit. At the beginning of an experiment the set-up was put at the initial situation (Fig. 7.4(a)). Once the task is achieved and reward is obtained by the robot, the set-up is put back to the initial situation and the task can be performed again.

As we run the task in simulation, the behavior of the human is also simulated. We chose here to have a collaborative human: it performs all actions HATP planned for him and participates to handover whenever the robot requires one.

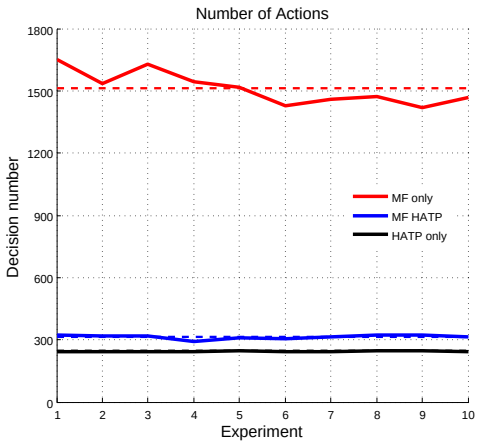
7.4.3 Results

The main criteria used to evaluate our system is the cumulative reward obtained in each run (i.e. the number of time the human and the robot manage to achieve the task in a fixed amount of time). We run 10 times the experiment in each condition (MF only, HATP only and the combination of both) for a duration of approximatively 30 minutes. The number of rewards obtained are presented in Fig. 7.5(a). All experiments last the same fixed time, but the number of decisions taken at the end may vary. We observe a poor performance of the MF alone, which is not able to solve the task more than three times. As the MF has no initial knowledge, it has to discover the right sequence of actions, which is non trivial with the given number of possible states and actions. The random combination HATP-MF is performing much better than the MF alone, solving the task 25 times in average. However, HATP alone performs even better solving the task 34 times in average. Indeed, the task is easy enough to solve for HATP and the time required to find a plan is negligible here. As the simulated human always performs the actions planned by HATP, the plan found by HATP is always optimal and will never change during the task execution. Accordingly, the random combination of HATP and the MF performs worst as it can include actions proposed by the MF that make the plan non optimal.

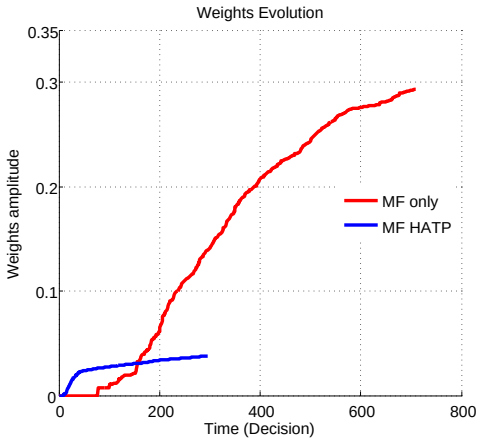
Fig. 7.5(b) shows the number of actions proposed to the supervision system during each experiment. We can see that the MF alone suggests twice to three



(a) Mean cumulative reward on 10 simulations where the robot repeatedly fulfils the task. Errorbars represent the standard deviation from the mean every 100 decisions.



(b) Number of actions tried per experiment. Dashed line is the mean number of action depending on the control method (MF only, HATP only or combination)



(c) Mean MF connection weights evolution for MF alone and MF and HATP combination. The amplitude is defined as the sum of the absolute value of weights. Weights are initialized to zero, thus the higher the amplitude is, the more the MF has learnt which action to do.

Figure 7.5: Performance of the developed system compared to systems with only the MF and only HATP. The results are for 10 runs of approximatively 30 minutes in each condition.

times more actions than HATP or the combination in the same given time. This is mainly due to the way each Expert decides: the MF only needs to compute the values of each action (which is propagating the state activity to action neurons) and to draw an action from the resulting probability distribution. It proposes a lot of infeasible actions and the supervision system will not spend time to execute them as it will stop to the preconditions verifications. HATP checks for action preconditions

when planning and so, for each of the action proposed by HATP, the supervisor spend time to execute it (or try to execute it if the action is not really feasible according to the geometry). The number of actions suggested by the combination of Experts is closer to the one with HATP alone while remaining lightly higher. It can be explained by the fact that a part of the actions proposed by the combination comes from HATP and, for the ones coming from the MF, HATP helps it to learn faster a solution, causing it to propose less infeasible actions.

Finally, we analyse the effect of combination on learning of MF in Fig. 7.5(c). Learning is evaluated by weights amplitude, namely the sum of weights absolute value over actions. The MF starts with weights initialized to zero, each learning step increases or decreases the value of some of the weights, until convergence. The figure shows that learning occurs much earlier for the combination of Experts than when the MF is alone. The combination has a bootstrapping effect and the knowledge about the task from HATP is transferred to the MF. This shows that a human-provided a priori knowledge can be used to guide exploration and learn quicker. Even if not tested in this experiment, this means that a change in task condition for which HATP can find a new plan can be learnt quickly by the MF, so the robot will be able to adapt to the new conditions without taking too much time.

7.4.4 Intermediate conclusion

The first results obtained with this architecture allow to show that the combination of HATP and the MF allows to bootstrap the MF and to learn faster a policy to achieve the goal.

However, this task is too simple for HATP to be in difficulty when deciding alone. The purpose of the second task and architecture presented in the next section is to show the benefits of the combination of the two experts and more particularly how HATP can benefits from the MF. Moreover, we want to test the reaction of our system to changes in the task as well as a more elaborated arbitration criteria for the meta controller.

7.5 Second architecture: the limitations

7.5.1 Control architecture

One of the advantage of the MF against HATP is its computation time. In the previous architecture, both experts where consulted before that the meta controller took a decision. Consequently, even if the MF was chosen, we still lose time to compute plans with HATP. In order to solve this issue, we modified the previous architecture as shown in Fig. 7.6.

In the new architecture, the meta controller is placed upstream from the two experts. Consequently, the order of the previous steps during a task is also slightly modified:

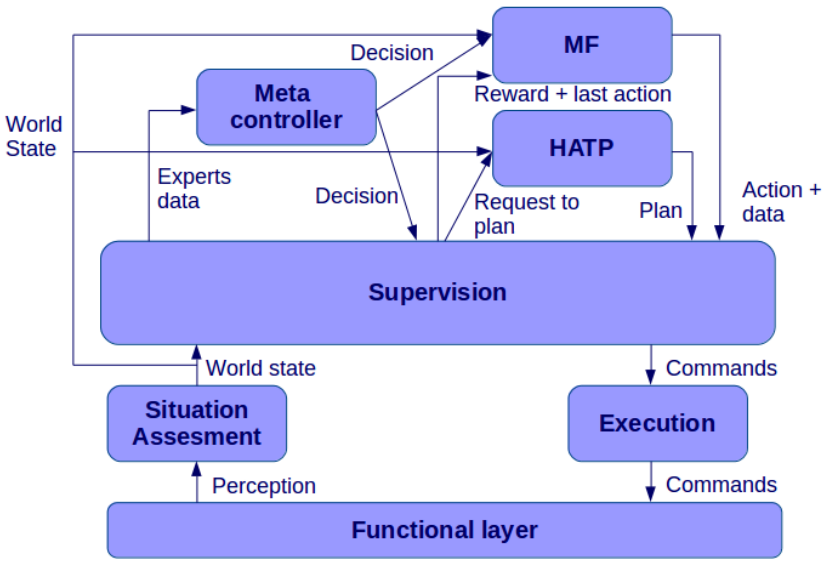


Figure 7.6: Second tried architecture to combine the two experts. The Situation Assessment module gets data from perception and maintains the current world state. This world state is used by the supervision to compute the reward and by the experts to take decisions. Here the meta controller is placed upstream from the two experts. It first decides which expert should propose an action. Then, the supervisor executes the action of the chosen expert with the help of lower execution modules.

- The Situation Assessment module still receives data from perception and maintains the world state representation.
- The supervisor sends the needed data concerning both experts to the meta controller in order for it to take a decision (see bellow).
- Once the meta controller decision taken, we look for the action proposed by the selected expert. If the MF is chosen it directly sends its action to the supervisor as well as data concerning its decision (see bellow). If HATP is chosen, if needed, the supervisor asks for a new plan, else it directly executes the next action of the current plan. A new plan is needed at the beginning of the task or whenever an unexpected situation happen (an action from the plan fails, the human executes an unexpected action or the robot executes an action proposed by the MF which is not in the current plan).
- The supervisor still executes the chosen action with the help of lower execution modules (motion planning, control, ...).

In this architecture, we also introduced a new arbitration criteria for the meta controller. This criteria is based on the cost of each expert (duration to find a solution) and its prediction error. For the MF, the prediction error is the difference

between the probability for the proposed action to lead to a reward and the actual received reward. For HATP, the prediction error is 0 if after the execution of the proposed action the world state corresponds to what HATP predicted (based on the action effects) and 1 if it differs.

$$P_t^E = \alpha \ . \ err_t^E + \beta \ . \ cost_t^E$$

with P_t^E the probability for the expert E to be chosen by the meta controller at a time t, err_t^E the prediction error of the expert E at a time t and $cost_t^E$ the cost of an expert E at a time t. α and β are parameters. The prediction error and the cost of the experts are averaged through time:

$$err_t^E = (1 - \gamma_{err}) \ . \ err_{t-1}^E + \gamma_{err} \ . \ err_t^E$$

$$cost_t^E = (1 - \gamma_{cost}) \ . \ cost_{t-1}^E + \gamma_{cost} \ . \ cost_t^E$$

with γ_{err} and γ_{cost} parameters.

7.5.2 Task

The previous task was too simple to have difficulties with HATP as the only expert. The new task is an upgrade of the previous one with several additions.

More complex task A first way to complex the task for HATP is to increase the combinatory of the task. Indeed, there was not too much ways to solve the previous task, so, HATP didn't need to much time to compute a plan. The goal of the new task is always to "clean a table", however, there is now two different boxes were to put the objects. The blue objects have to go in the blue box and the green objects have to go in the green box. We increased the number of objects in the task: at the beginning of the interaction 6 objects (3 blue and 3 green) are randomly placed on 7 possible placement in the table (see Fig. 7.7(a) and Fig. 7.7(b)).

We also add some new possible actions for the robot:

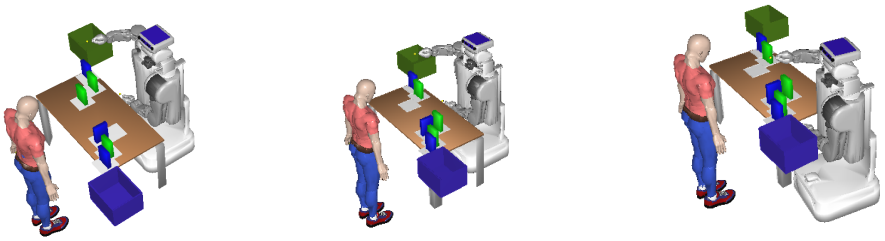
- **Pick an object:** both agents can still pick the objects accessible by them.
- **Throw an object:** the robot can throw an object it has in hand in a box of the same color accessible by itself. The human can throw an object it has in hand in the blue box.
- **Give an object:** the robot can still give an object to the human.
- **Take an object:** the robot can still receive an object from the human
- **Place an object on a placement:** the robot can place an object it has in hand on a placement accessible by itself.
- **Navigate to another position:** the robot can navigate to another position in order to change the objects it can reach. The two possible positions for the robot are the one in Fig. 7.7(a) and Fig. 7.7(b)) and the one in Fig. 7.7(c).

- **Wait:** the robot can still wait for the human to execute an action.

The predicates used to represent the world state also changed. They are now composed of:

- **<Object, isReachableBy, Agent>:** these predicates represent for each objects if they are reachable by the human or the robot.
- **<Placement, isReachableBy, Agent>:** these predicates represent for each placement if they are reachable by the human or the robot.
- **<Box, isReachableBy, Agent>:** these predicates represent for each box if they are reachable by the human or the robot.
- **<Object, isIn, Box>:** these predicates represent the fact that an object has been throw in a box.
- **<Agent, hasInHand, Object>**: these predicates represent the fact that the human or the robot holds an object.
- **<Object, isOn, Placement>**: these predicates represent the fact that an object is on a specific placement.
- **<Robot, isAt, Position>**: these predicates represent the position of the robot (Position are the two possible places it can navigate to).

In this task, a reward is given to the robot whenever all objects are in a box.



(a) One possible initial set-up. In this situation the robot can access four objects (two blue and two green) as well as the green box.

(b) Another possible initial set-up. In this situation the robot thinks it can access the blue object in the middle of the table. However, the green object in front of it blocks its access.

(c) One possible way for the robot to access the blue object it was not able to reach is to move to another position.

Figure 7.7: Description of the task used with the second architecture. In this task, the human and the robot have to remove all the objects of the table and put them in the box of the same color. At the beginning of the interaction several objects are accessible by the robot, others by the human and others by both agents. The green box is accessible by the robot and the blue one by the human. The placements are the white squares on the table.

As there is more objects and more actions to perform for the robot, the number of possible ways to achieve the task highly increases. Indeed, in the initial set-up of the previous task HATP needed around 145ms to find a plan. In this new task it takes it around 22s to find a plan to achieve the goal. Consequently, the difference of cost between the MF and HATP should makes more difference in this experiment.

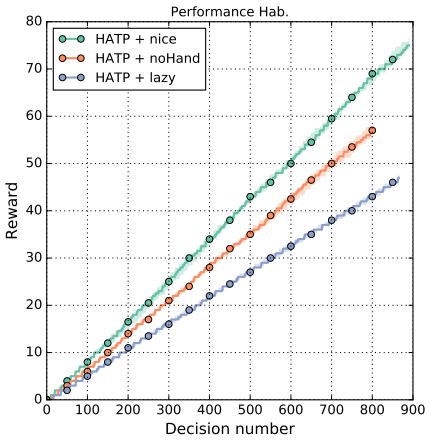
Difference between planning and geometry In order to get closer from possible real life situation, we introduced a geometrical problem in the task. Indeed, sometimes it can happen that the knowledge computed by the robot is not accurate and that, consequently, the computed plan is not valid at execution. In our task, there is two placements in the middle of the table (accessible both by the human and the robot) which are close to each other. Each time on object is on one of these placement, the robot thinks it can reach it. when there is an object in only one of the placement (as in Fig. 7.7(a)) the robot can effectively reach the object. However, when there is an object in both placements, the robot cannot reach the one in the farthest placement as the other one blocks its access (see Fig. 7.7(b)). The Situation Assessment is not able to differentiate the two situations and in each case it will estimate that all objects in these placements are reachable by the robot. The robot will discover that it can not reach an object at motion planning time and so, the initial HATP plan will not take this into account (but when the action to pick the object not reachable failed, the robot will update its knowledge and so the new HATP plan). To access an object not reachable by it the robot can either navigate to another position (as in Fig. 7.7(c)), remove the object which blocks the access or get the object from the human (through handover).

The MF should allow the robot to learn in which case an object is really reachable by the robot and in which case another solution is preferred to get the object.

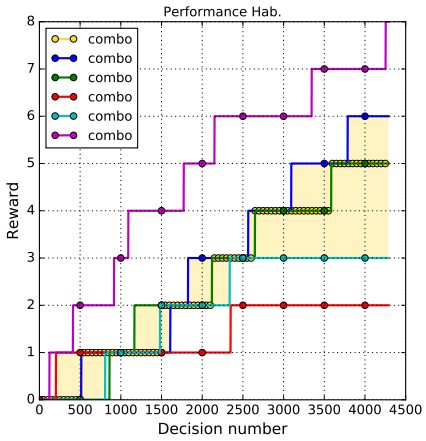
Different human behaviors Finally, in the previous task, one of the reason HATP was performing very well was that the human was always executed the actions planned for him. In real life, even if the human is collaborative, he does not necessarily take the same decisions as the ones HATP took for him. In this task, we introduced three different kinds of human behavior:

- **The collaborative human:** it picks all objects accessible by him (with a priority for blue ones), throws the blue objects in the blue box and participates to all handover engaged by the robot.
- **The anti-handover human:** it picks all the blue objects accessible by him, throws the blue objects in the blue box but does not participate to handover engaged by the robot
- **The lazy human:** it picks the blue objects accessible only by him (and not the ones the robot can access), throws the blue objects in the blue box and does not participate to handover engaged by the robot

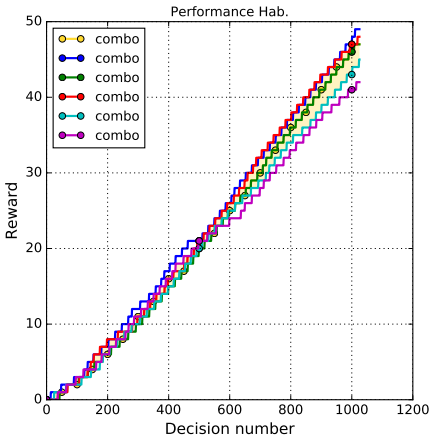
7.5.3 Results



(a) Mean cumulative reward for the system with only HATP. We can see that the system performs better with a collaborative human (nice), then with an human rejecting handover (noHand) and then with a lazy human (lazy).



(b) Mean cumulative reward for the system with only the MF. Different values for the MF parameters have been tested in order to find the best configuration in this task



(c) Mean cumulative reward for the system with the combination of both experts; Different values for the arbitration criteria parameters have been tested in order to find the best configuration in this task.

Figure 7.8: Mean cumulative reward for each conditions tested (HATP only, MF only and combination). The results are for 10 runs of approximatively 40 minutes in each conditions where the robot repeatedly fulfils the task.

We first tried the new architecture and task with HATP as the only expert. We can see in Fig.7.8(a) that, as expected, HATP performs better with a collaborative

human which will have a behavior closer than the one its planned that with humans with less collaborative behaviors.

Then, we tested the system with the MF alone and with different parameters of the learning algorithm in order to get the better possible instantiation for this task. In a first step, we did it with a collaborative human and with only one possible initial set-up without geometrical complications. We can see in Fig. 7.8(b) that, as expected, the MF alone performs poorly compared to HATP.

Then, we tested the combination of both experts. In a first step, we tested it with the collaborative human and with only one possible initial set-up without geometrical complications. We tested several parametrisation of the arbitration criteria in order to get the best implementation for this task. However, we noticed that, even when we put the system in the best possible situation, it performs barely as well as HATP in its worst case (the task was solved around 45 times in each cases, see Fig. 7.8(c)). Indeed, with a more complex task, the bootstrap effect of HATP was not enough for the MF to learn a sufficiently good action policy. We tried with some runs way longer (several hours) but it was still not sufficient for the MF to learn a correct policy.

Moreover, even if we tried to reduce computation time by putting the meta controller upstream from the experts, the effect was not the one expected. Indeed, the meta controller here is probabilist and so, even if the probability to choose the MF becomes higher than HATP, it can still happen for HATP to be chosen. In this case, a whole plan is computed by HATP even if we ask it only one action during the task. Consequently, the planning time remains the same than if HATP follows its plan alone to achieve the task.

7.6 Conclusion

In this chapter, we presented an architecture allowing to combine learning (a model free algorithm) and planing (a human-aware task planer HATP) during the robot decisional process. First results shown that HATP allows to bootstrap the learning and so to quickly learn a consistent and acceptable behavior for the robot.

In a second time, we tried to show the benefits of the learning in the system. Despite the facts that the results was not the one expected, we can still learn some lessons from this work and think to solutions to ameliorate the system. One first possible modification would be to rework on the learning algorithm in order to study if there is methods more adapted to this context. Then, another amelioration would be to look for a new arbitration criteria between the two experts. Maybe a criteria with an hysteresis in order to reduce switches between experts in a task and allow them to have time to develop their own strategy (and not having one expert breaking the strategy the other tried to set-up) would be a good idea. Finally, one interesting idea is to allow HATP to have a feedback on what is learned by the MF. Indeed, the knowledge of HATP concerning the actions is put off-line and is not updated during the interaction. For example, maybe the learning can provide the

real needed time to execute an action or its probability of success given what was learned from previous interactions.

Bibliography

[Balleine 2010] Bernard W Balleine and John P O'doherty. *Human and rodent homologies in action control: corticostriatal determinants of goal-directed and habitual action.* Neuropsychopharmacology, vol. 35, no. 1, pages 48–69, 2010. (Cited in page 2.)

[Brafman 2002] Ronen I Brafman and Moshe Tennenholtz. *R-max-a general polynomial time algorithm for near-optimal reinforcement learning.* Journal of Machine Learning Research, vol. 3, no. Oct, pages 213–231, 2002. (Cited in page 5.)

[Cirillo 2010] Marcello Cirillo, Lars Karlsson and Alessandro Saffiotti. *Human-aware task planning: an application to mobile robots.* IST, 2010. (Cited in page 3.)

[Daw 2005] Nathaniel D Daw, Yael Niv and Peter Dayan. *Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control.* Nature neuroscience, vol. 8, no. 12, pages 1704–1711, 2005. (Cited in page 3.)

[Dickinson 1985] Anthony Dickinson. *Actions and habits: the development of behavioural autonomy.* Philosophical Transactions of the Royal Society of London B: Biological Sciences, vol. 308, no. 1135, pages 67–78, 1985. (Cited in page 2.)

[Dolan 2013] Ray J Dolan and Peter Dayan. *Goals and habits in the brain.* Neuron, vol. 80, no. 2, pages 312–325, 2013. (Cited in page 2.)

[Dollé 2010] Laurent Dollé, Denis Sheynikhovich, Benoît Girard, Ricardo Chavarriaga and Agnès Guillot. *Path planning versus cue responding: a bio-inspired model of switching between navigation strategies.* Biological cybernetics, vol. 103, no. 4, pages 299–317, 2010. (Cited in page 3.)

[Erol 1994] Kutluhan Erol, James Hendler and Dana S Nau. *HTN planning: Complexity and expressivity.* In AAAI, volume 94, pages 1123–1128, 1994. (Cited in page 4.)

[Hayes 1994] Gillian M Hayes and John Demiris. A robot controller using learning by imitation. University of Edinburgh, Department of Artificial Intelligence, 1994. (Cited in page 3.)

[Ingrand 2014] Félix Ingrand and Malik Ghallab. *Deliberation for autonomous robots: A survey.* Artificial Intelligence, 2014. (Cited in page 3.)

[Kaplan 2002] Frederic Kaplan, Pierre-Yves Oudeyer, Enikö Kubinyi and Adám Miklósi. *Robotic clicker training.* Robotics and Autonomous Systems, vol. 38, no. 3, pages 197–206, 2002. (Cited in page 3.)

[Keramati 2011] Mehdi Keramati, Amir Dezfouli and Payam Piray. *Speed/accuracy trade-off between the habitual and the goal-directed processes.* PLoS Comput Biol, vol. 7, no. 5, page e1002055, 2011. (Cited in page 3.)

[Khamassi 2016] Mehdi Khamassi, Benoît Girard, Aurélie Clodic, Sandra Devin, Erwan Renaudo, Elisabeth Pacherie, Rachid Alami and Raja Chatila. *Integration of Action, Joint Action and Learning in Robot Cognitive Architectures.* Intellectica-La revue de l'Association pour la Recherche sur les sciences de la Cognition (ARCo), vol. 2016, no. 65, pages 169–203, 2016. (Cited in page 2.)

[Knox 2012] W Bradley Knox and Peter Stone. *Reinforcement learning from simultaneous human and MDP reward.* In Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1, pages 475–482. International Foundation for Autonomous Agents and Multiagent Systems, 2012. (Cited in page 3.)

[Kober 2011] Jens Kober and Jan Peters. *Learning elementary movements jointly with a higher level task.* In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, pages 338–343. IEEE, 2011. (Cited in page 3.)

[Lallement 2014] Raphaël Lallement, Lavindra de Silva and Rachid Alami. *HATP: An HTN Planner for Robotics.* CoRR, 2014. (Cited in page 3.)

[Lesaint 2014] Florian Lesaint, Olivier Sigaud, Shelly B Flagel, Terry E Robinson and Mehdi Khamassi. *Modelling individual differences in the form of pavlovian conditioned approach responses: a dual learning systems approach with factored representations.* PLoS Comput Biol, vol. 10, no. 2, page e1003466, 2014. (Cited in page 3.)

[Martins 2010] Murilo Fernandes Martins and Yiannis Demiris. *Learning multirobot joint action plans from simultaneous task execution demonstrations.* In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1, pages 931–938. International Foundation for Autonomous Agents and Multiagent Systems, 2010. (Cited in page 3.)

[Morimoto 2001] Jun Morimoto and Kenji Doya. *Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning.* Robotics and Autonomous Systems, vol. 36, no. 1, pages 37–51, 2001. (Cited in page 3.)

[Nicolescu 2003] Monica N Nicolescu and Maja J Mataric. *Natural methods for robot task learning: Instructive demonstrations, generalization and practice.* In Proceedings of the second international joint conference on Autonomous agents and multiagent systems, pages 241–248. ACM, 2003. (Cited in page 3.)

[Nikolaidis 2013] Stefanos Nikolaidis and Julie Shah. *Human-robot cross-training: computational formulation, modeling and evaluation of a human team training strategy.* In Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction, pages 33–40. IEEE Press, 2013. (Cited in page 3.)

[Pezzulo 2013] Giovanni Pezzulo, Francesco Rigoli and Fabian Chersi. *The mixed instrumental controller: using value of information to combine habitual choice and mental simulation.* Frontiers in psychology, vol. 4, page 92, 2013. (Cited in page 3.)

[Renaudo 2014] Erwan Renaudo, Benoît Girard, Raja Chatila and Mehdi Khamassi. *Design of a control architecture for habit learning in robots.* In Conference on Biomimetic and Biohybrid Systems, pages 249–260. Springer, 2014. (Cited in pages 2 and 4.)

[Renaudo 2015] Erwan Renaudo, Sandra Devin, Benoıt Girard, Raja Chatila, Rachid Alami, Mehdi Khamassi and Aurélie Clodic. *Learning to interact with humans using goal-directed and habitual behaviors.* In RoMan 2015, Workshop on Learning for Human-Robot Collaboration, 2015. (Cited in page 2.)

[Smart 2002] William D Smart and L Pack Kaelbling. *Effective reinforcement learning for mobile robots.* In Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on, volume 4, pages 3404–3410. IEEE, 2002. (Cited in page 3.)

[Stulp 2013] Freek Stulp and Olivier Sigaud. *Robot skill learning: From reinforcement learning to evolution strategies.* Paladyn, Journal of Behavioral Robotics, vol. 4, no. 1, pages 49–61, 2013. (Cited in page 3.)

[Sutton 1998] Richard S Sutton and Andrew G Barto. Introduction to reinforcement learning, volume 135. MIT Press Cambridge, 1998. (Cited in page 3.)

[Thomaz 2006] Andrea Lockerd Thomaz, Cynthia Breazeal*et al.* *Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance.* In Aaai, volume 6, pages 1000–1005, 2006. (Cited in page 3.)

[Thorndike 1998] Edward Lee Thorndike. *Animal intelligence: An experimental study of the associate processes in animals.* American Psychologist, vol. 53, no. 10, page 1125, 1998. (Cited in page 2.)

[Viejo 2015] Guillaume Viejo, Mehdi Khamassi, Andrea Brovelli and Benoît Girard. *Modeling choice and reaction time during arbitrary visuomotor learning through the coordination of adaptive working memory and reinforcement learning*. Frontiers in behavioral neuroscience, vol. 9, 2015. (Cited in page 3.)

[Watkins 1989] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989. (Cited in page 5.)