# Active Perception at the Architecture Level: A Preliminary Report

Niv Rafaeli
Computer Science Dept.
Bar Ilan University, Israel
rafaeln@cs.biu.ac.il

Gal A. Kaminka
Computer Science Dept. &
Gonda Brain Research Center
Bar Ilan University, Israel
galk@cs.biu.ac.il

## ABSTRACT

Robots rely on a set of beliefs, while looping in an action-perception cycle. These beliefs include results of the perception process which is composed of a chain of actions that use the agent's sensors and computation power in order to receive, fuse, filter and process information about the environment, to acquire new beliefs, and revise existing ones. Current Belief-Desire-Intention (BDI) systems, which provide the architectural framework for this process, implicitly assume that during run time the relevant beliefs over the world are made available. However, in realistic, physical environments this assumption often fails. *Active perception* processes are the solution for handling missing information during run time. However, in most of the cases they are used ad-hoc for specific tasks. This article deals with active perception in the architecture level. We present several algorithms that integrate active perception into the BDI interpreter loop. We show that different methods of implementation creates major differences in the running time of the algorithms. We advocate the DSAP algorithm that takes under consideration the limited available information in order to minimizes the amount of time the agent has to invest in execution of active perception processes.

## Keywords

integration, active perception, architecture

## 1. INTRODUCTION

Intelligent robots rely on a set of beliefs, while looping in an action-perception cycle. The set of beliefs contains, among other things, the agent's beliefs about the world. These are the end result of a *perception process* which uses sensors and computation power in order to receive, fuse, filter and process information about the environment, to acquire new beliefs, and revise existing ones.

One architectural framework for understanding this process is called BDI (Belief, Desire, Intention). BDI architectures/systems have been used to relate how perceptual processes generate and revise beliefs, which lead to reconsideration of goals, which in turn leading to adoption of plans (associated with specific intentions), to pursue of selected goals. BDI-based robots have been used in numerous case studies, investigations, and applications.

In this paper, we focus on the perceptual process of BDI-based agents in general, though the challenge we tackle is often raised in robots. Ketenci et al. [8] and So et al. [15] describe two principled types of perception generating these beliefs. A top-down process, known also as *active perception*, is ideally controlled by the goal-oriented reasoning process, enables the agent to turn its perception—by taking actions—to the most relevant aspects of the environment according to its task. A bottom-up process known also as *passive perception* ideally originates from the sensors, allowing goal-independent, and opportunistic perception.

In physical environments, passive perception is not enough. The agent's resources that are assigned to the perception process are bounded; sensors have limited capabilities (e.g. range, distance and more) and objects may be occluded, out of range, etc.

Unfortunately, current BDI systems and particularly BDI robotic agents use only passive perception. In existing formulations of BDI, the basic assumption is that the agent has all relevant beliefs, but what about the cases where an agent needs to take an action in order to change the state of its beliefs?

Shanahan [14] describe active perception as any form of action that leads to acquiring of new information. Ranging from short duration, low-level actions as applying a filter on a camera sensor to long-tern high-level actions as opening a door in order to look on the other side. Subtle knowledge-producing actions as asking a question or browsing for information through a list [12] are also on this spectrum. Shanahan's definition bounds together under the term active perception various research fields that use to be considered separate, e.g. animate vision, active sensing, active vision and more.

In recent years, designers of intelligent agents have been using active perception capabilities in physical environment. Unfortunately, the majority of the research in the field of active perception deals with specialized algorithms for specific tasks. In particular, there is little or no discussion of active perception capabilities as part of a BDI loop, i.e., at the architecture level. Instead, active perception is folded into the task-specific plans and perceptual models of the agent.

In this article we investigate how active perception is to be integrated in a BDI loop. We will present different possibilities for such extended BDI loops and compare them analytically. We draw conclusions as their relative merits and run-time complexity. One algorithm emerges a clear

To appear in: *Proceedings of the Autonomous Robots and Multirobot Systems (ARMS) workshop, part of the AAMAS conference, 2017.* Non-archival publication.

winner over the others.

## 2. BACKGROUND AND RELATED WORK

Specific instances of active perception—for specific tasks—have often demonstrated the usefulness of this capability [9, 13, 10, 4, 17]. For example, Alomonois et al. [1] prove that vision problems can be solved much efficiently by an active observer than by a passive one. Ballard [2] shows that the visual computation of systems with active gaze control mechanisms is vastly less expensive than passive systems. However, these algorithms provide ad-hoc solutions for a specific problems. Our approach is to enable an agent the use of active perception capability as part of its architecture.

Weyns et al. [18] present a general model for active perception, composed of three functionalities: *Sensing*, which maps the environment to *representation*; *interpreting*, where the agent turns *representation* to *percepts* (in BDI: *beliefs*); and *filtering*, where the agent can give attention to the most relevant information in the context of its current task. They suggest a reusable framework that allows active change of the agent's perception process parameters to improve perception. Our work is different in two ways. First, while Weyns et al. focus their work on low-level actions, we address also high-level behaviors (e.g. move inside room to get a better look). Second, our work focuses on the integration of active perception in the agent's decision making not just in the perception mechanism itself.

So et al. [15] use *situation awareness* (SA) as a meta mechanism that able to switch between top-down goal-driven and bottom-up data-driven models of information processing. It has three layers. The third layer deals with projection to the near future, and here the top-down goal-driven processes takes place: context and/or precondition clauses determine beliefs that need to be refreshed. Our work is different in several ways. First, So et al. suggested that the active perception process will be triggered by the third layer whenever it is needed. However, they left the question of integration of active perception in the agent's decision making mechanism open. Furthermore, our mechanisms do not use future projection of the agent's state but plan's clauses only. And last, we extend So et al's. definition for beliefs that are relevant to active perception plans by taking under consideration beliefs whose value was never known.

## 3. ACTIVE PERCEPTION IN A BDI LOOP

In this section we describe the problem of enhancing BDI control loop with active perception and propose possible solutions. First, (Section 3.1) we review the basic BDI algorithm [11]. Next, we present a formal description of the problem (Section 3.2). Finally, we describe four possible BDI loop algorithms that integrate active perception (Section 3.3).

### 3.1 The Basic BDI loop

We focus on enhancing BDI architectures with active perception capabilities. However, there is no single standard algorithm of a BDI architecture. Wooldridge [19, Chapter 4] presented a detailed BDI architecture for practical reasoning agents that allows the agent to reconsider its commitments during runtime. Tambe [16] and Kaminka et al. [7, 6] describe BDI loops that integrate collaborative processes. De Silva et al. [3] focus on BDI loops that support planning. Others created versions of BDI loops with learning [5], and more.

We therefore turn to the original algorithm presented by Rao et al. [11]. Rao et al's architecture uses three dynamic data structures for the agent's beliefs, desires and intentions together with an input queue of events. The events the system can recognize are both internal external events; they are also assumed to be atomic and to be recognized after they had occurred. The output of the system—the agent's actions—is also assumed to be atomic. Alg. 1 is the basic BDI control-loop described in [11].

**Algorithm 1** BDI Loop.

```
1: initialize-state
2: repeat
     1. $p$:= Get-new-percept()
     2. $b$:= Update-beliefs($b$,$p$)
     3. options:= option-generator(event-queue,$b$)
     4. $P_x$:= deliberate(options)
     5. update-intentions($P_x$)
     6. execute()
     7. get-new-external-events()
     8. drop-unsuccessful-attitudes()
     9. drop-impossible-attitudes()
3: end repeat
```

In line 1 the agent initializes its state, often the initial state includes the agent's initial intentions and beliefs [19]. In line 2 the agent enters into a loop that contains several steps:

1. *option generator()* reads the event queue and supply a list of options. This is a list of plans which specify the means of achieving future world states (the list of future world states often referred to as the agent's desires).

2. *deliberate()* chooses an option to be adopted (intention). Although in their model Rao et al. talk about adopting a set of intentions simultaneously, here, in order to simplify our model we will assume that only a single option is being selected.

3. The selected option (line 2.3) is pushed into the intentions structure (i.e. the agent commits to execute it).

4. If there is an intention to perform an atomic action the agent executes it (*execute()*).

5. The agent updates the event queue with any external event that happed during executions, while internal events are being updated when they occur (line 2.5).

6. The agent then modifies the intention and desires structure by dropping successful desires and satisfied intentions as impossible desires and unrealizable intentions (lines 2.6–2.7).

The information about the means of achieving certain desires represented as plans. Each plan has a body which describes the subgoals or simple action needed to be done in order for the plan to be successful. Each plan has preconditions the specifies the conditions that must hold for the plan to be executed. Often pre-conditions are represented as first order logic formulas over the agent's beliefs.

## 3.2 Architectural Active Perception

Algorithm 1 makes an implicit assumption that an agent has beliefs over the world. However, realistically, some belief values may be *unknown* or *outdated* (suspected to be no longer correct). Therefore the assumption that the decision-making process can use the beliefs values during run time fails.

In the original BDI loop (Alg. 1) there is no explicit reference to such beliefs, or to mechanisms for handling them. Any actions in service of active perception must be carried by the agent executing Algorithm 1 as part of a plan or a behavior for a specific task. In other words, it is up to the *option generation* procedure to generate active perception plans, or plans incorporating and interleaving active perception actions.

In this article we deal with the problem of weaving active perception plans into a BDI loop. In order to do so it is necessary to modify the basic BDI algorithm so it will be able to consider the execution of a goal driven process which its goal is to update a set of unknown or outdated beliefs.

### *3.2.1 Which Beliefs Should Be Updated?*

In order to decide if a plan can be selected for execution the agent compares the plan's preconditions to its beliefs over the world and if they are satisfied, the plan is feasible for execution.

It is clear that not all agent's beliefs are important for the process of decision making all the time. Kaminka [6] introduces the notion of *support* that helps to describe the *relevant beliefs*, who are the beliefs that important for the decision making process. For a given belief $k$, whether it is an atomic datum, or a complex data-structure support$(k)$ is the set of antecedent knowledge that gave $k$ its value, typically through the application of a computational process. support$(k)$ is transitively closed; it includes the antecedents of $k$, their antecedents, and so forth until raw perceptions and axiomatic beliefs are reached.

Using the notation of *support* we can define *Relevant beliefs*:

DEFINITION 1 (RELEVANT BELIEFS). *A belief k is called* relevant to a plan $p$ *if one of the following conditions holds:*

1. $k \in \text{support}(v)$*, where $v$ is a belief directly used in $p$.*
2. $k \in \text{support}(\text{termconds}(p))$*, where* $\text{termconds}(p)$ *is the set of its termination conditions of $p$.*
3. $k \in \text{support}(\text{preconds}(p))$*, where* $\text{preconds}(p)$ *is the set of its preconditions of $p$*

Based on Def. 1 we call the group of beliefs that the agent should update *missing beliefs*, defined as follows:

DEFINITION 2 (MISSING BELIEFS). *A belief is called* missing *if it is a* relevant belief *to $p$ and either* outdated *or* unknown*, where:*

1. *Outdated beliefs are beliefs whose value is known. However, their value is suspected of being wrong (e.g. due to amount of time passed since last update).*
2. *Unknown beliefs are beliefs whose value was never set.*

The definition for missing beliefs, covers all the beliefs that are necessary for an agent selecting and executing a plan $p$ during run time. There are no other beliefs that can be relevant to the agent's operation. When their value is not known, the agent cannot apply the selection and termination mechanisms and also can not guarantee the execution of the selected plan. Therefore *missing beliefs* are the beliefs that it is necessary to apply active perception process in order to acquire their value. In Alg. 1, the only relevant beliefs are those explicitly used to guide option selection, i.e., beliefs supporting preconditions.

### *3.2.2 A Formal Perspective on Active Perception in the BDI Loop*

Theoretically, when an agent knows everything about the world, the set of optional plans $O$ which is considered in line 1 of each cycle is divided to two groups of plans. The first, is a group of *feasible plans* $F = \{p_1, ..., p_n\}$ where the preconditions of plans in $F$ are true, so they can be used by the agent in the context. The second group is a group of *non-feasible plans* $N = \{p_1, ..., p_m\}$ where at least one precondition of each plan in $N$ is false, so plans in $N$ can not be used by the agent in the context.

In line 2 the agent chooses the plan it wants to commit to, the *deliberate()* function chooses a plan $P_{opt} \in F$. The selection is optimal in the sense that *deliberate()* has the full set of options to select from, using whatever knowledge it can bring to bear on the selection process.

However, in reality, because the environment is hidden and there are preconditions whose value is unknown because they are supported by *missing beliefs*, the agent cannot partition the plans in $O$ to subsets $F$ and $N$. Instead, the agent has to classify the plans into one of three sets based on its own beliefs: The first is $F'$, a group of plans that the agent believes belong to $F$. The second is $N'$, a group of plans the agent believes belong to $N$, and $M'$, a group of plans which their preconditions are not fully known, therefore they cannot be part of $F'$ and $N'$:

1. $F' = \{p | p \in O, \forall precond(p) = true\}$
2. $N' = \{p | p \in O, \exists precond(p) = false\}$
3. $M' = \{p | p \in O, \exists precond(p) = Missing, \nexists precond(p) = false\}$

An important observation is that although the agent can execute only options that are in $F'$, there is no guarantee that the ideal selection $P_{opt} \in F'_j$. It might also be in $M'$. In that case, the agent might choose to execute an active perception plan in order to update values of missing preconditions of candidate for $P_{opt}$ and reveal it.

The goal of active perception process is to update the agent's beliefs. The agent can move plan $P_i$ from the missing options set $M'$ to the feasible option set $F'$ by executing an active perception plan $a_k^{Pi}$ that *supports* plan $P_i$, per Def. 3 below:

DEFINITION 3 (SUPPORTING A PLAN). *An active perception plan* $a_k^P$ supports *plan $P$ if it sets the value of a missing belief of $P$.*

By applying a series of active perception plans, one can eliminate all missing beliefs associated with a plan. This process is called *revealing*, as defined below (Def. **??**). Note that in the scope of this paper, all missing beliefs are associated with preconditions, thus the reference to preconditions in the definition.

Definition 4 (Revealing a plan). Revealing *a plan $P_i$ is the process of executing a series of active perception processes that support $P_i$, until one of the following two conditions hold.*

1. preconds$(P_i) \vdash \top$, *i.e., the preconditions of $P_i$ provably hold, and thus $P_i$ will move from $M'$ to $F'$.*

2. preconds$(P_i) \vdash \bot$, *i.e., the preconditions of $P_i$ provably do not hold, and thus $P_i$ will move from $M'$ to $N'$.*

A plan $p_i \in M'$ can move from group $M'$ to $F'$ only if the agent will execute $A_{pi} = \{a_1^{pi}, .., a_k^{pi}\}$, an unordered set of active perception plans. Each $a_j^{pi} \in A_{pi}$ has a goal, to updates values of relevant *missing beliefs* and thus $p_i$'s preconditions.

The set $A' = \{A_{p1} \cup \ldots \cup A_{pm}\}$ is a union of all the active perception plans in all the suggested active perception processes for plans in $M'$, it can be defined also as follow $A' = \{a_j | aj \in A_{pi}, p_i \in M'\}$

The original BDI interpreter (Alg. 1) chooses a plan $P_y$ for execution from $F' \subseteq F$, but it might be that $P_{opt} \notin F'$. The original BDI algorithm does not consider these cases, and therefore there is no guarantee that $P_y = P_{opt}$. In other words, without active perception, Alg. 1 may make sub-optimal selection decisions.

## 3.3 Possible Integrated Active Perception

We present four algorithms that integrate active perception in the BDI loop. Each algorithm is an incremented improvement of its predecessor. The first algorithm (IAP, Section 3.3.1), executes active perception plans for every *missing belief* of the agent. IAP can be used when all *missing beliefs* must be revealed in order to make selection (i.e. when determining $P_{opt}$ necessarily requires deliberating between all plans), however its cost is high. The ITAP algorithm (Section 3.3.2) allows the agent to myopically select between running an active perception or executing a feasible plan instead, thus limiting the number of active perception plans that are executed. However, we show that it may lead to inefficiencies caused by its myopic selection. The SAP algorithm (Section 3.3.3) resolves these inefficiencies, by requiring the agent to commit to a plan to be revealed, before executing all the active perception plans that reveal it. Finally, the DSAP algorithm (Section 3.3.4), makes the agent commit to a plan to be revealed, but allows it to revisit the decision and select the next active perception plan to be executed.

### 3.3.1 Immediate Active Perception (IAP)

Alg. 2 presents the *Immediate Active Perception* method (IAP). The key idea in IAP is that before executing any option from $F'$, IAP executes all the possible active perception plans in $A'$. By doing that, IAP makes sure that $F' = F$. In this case, the selection function gets the whole $F$ to select from, therefore $P_y = P_{opt}$.

The difference from the original BDI loop is that here the interpreter checks if there are any missing options $M'$ (line 2). If there are any (line 3), the agent gets the active perception plans $A'$ that reveals the plans in $M'$ (line 4), then the deliberator decides to which plan from $A'$ to commit (line 5) and updates its intentions (line 6). If there are no missing options the interpreter works as the original BDI Loop.

**Algorithm 2** IAP BDI Loop

```
1: initialize-state
2: repeat
     1. options:= option-generator(event-queue)
     2. $F'$, $M'$:= classify-options(options)
     3. if $M' \neq \emptyset$ then
     4.     $A'$:= get-plans-for-missing-beliefs($M'$)
     5.     $P_x$:= deliberate($A'$)
     6. else
     7.     $P_x$:= deliberate(options)
     8. update-intentions($P_x$)
     9. execute()
    10. get-new-external-events()
    11. drop-unsuccessful-attitudes()
    12. drop-impossible-attitudes()
3: end repeat
```

In IAP, as long as there are plans in $M'$, the agent will not choose any of the plans in $F'$, but only execute active perception plans.

We use the following running example to illustrate. Suppose we have a robot that does not know its target location and the location of its leader, It can consider three *options*: to follow its target, to follow its leader, and to drive home. The first two options have missing beliefs (in $M'$, *missing options* for short) and the third is feasible (in $F'$). The IAP interpreter will execute both active perception processes to solve the *missing options*, it will look for the agent's target and look for the agent's leader and only then will select one of the original *options*.

There are two cases where the use of IAP algorithm is optimal. The first, is when the agent must reveal all the options in $F'$ before determining $P_{opt}$. It might occur when the agent can identify and select $P_{opt}$ only if $F' = F$ (relative selection). In these cases the agent will need to sort all the plans in $M'$ to $N'$ and $F'$ in order to achieve $F = F'$.

The second use case is where there is no cost for executing active perception processes. In this case, by performing all the active perception process in $A'$, the agent achieves $F' = F$ and IAP will guaranty the selection of $P_{opt}$ with no additional cost.

However, IAP will perform poorly when there is cost to active perception, because it performs active perception processes for all the plans in $M'$ even for ones that are not candidates for $P_{opt}$.

### 3.3.2 ITerative Active Perception (ITAP)

ITAP (Alg. 3) solves the problem presented above. The key idea is to allow the agent to choose its next plan for execution whether it is in $F'$ or in $A'$.

In line 2, like IAP, ITAP divides the options into two sets $M'$ and $F'$ (missing and feasible options accordingly). In line 3 it creates a set of active perception plans $A'$ that reveals the plans in $M'$. In line 4 ITAP creates a union of all the plans from $F'$ and $A'$ these are the plans that are feasible for execution. In lines 5 and 6 *deliberate()* selects the plan and ITAP updates the agent's intentions.

Back to the example: for the three original options, ITAP replaces the *missing options* follow the leader and follow the target with the relevant active perception processes—look for the leader and look for the target—and then the selector has to choose between the three options: look for the leader, look for the target or drive home.

**Algorithm 3** ITAP BDI Loop

```
1: initialize-state
2: repeat
     1. options:= option-generator(event-queue)
     2. $F'$, $M'$:= classify-options(options)
     3. $A'$:= get-plans-for-missing-beliefs($M'$)
     4. all-options:= $F' \cup A'$
     5. $P_x$:= deliberate(all-options)
     6. update-intentions($P_x$)
     7. execute()
     8. get-new-external-events()
     9. drop-unsuccessful-attitudes()
    10. drop-impossible-attitudes()
3: end repeat
```

The advantage of the ITAP Algorithm is that it considers execution of plans that do not need any active perception process. In ITAP, compared to IAP, the agent will execute an active perception process only if it has been selected. Therefore, in cases where active perception processes has costs, the agent can choose and execute a candidate to $P_{opt}$ from $F'$ at any iteration and does not must execute first all the plans in $A'$. The ITAP algorithm has a great potential to prevent unnecessary execution of active selection plans.

However, ITAP is myopic. When the agent selects an active perception process from $A'$, it does not consider that $A'$ is a union of active perception plans that relate to various missing options. Executing an active perception process $a_j^{pi} \in A_{pi}$ will move $p_i$ to $F'$ only if all the plans in $A_{pi}$ will be executed. ITAP treats the $A'$ as a whole set and ignores that it is a union of various $A_{pi}$'s.

This can lead to inefficiencies. suppose there are $k$ active perception plans in $A'$ and $P_{opt}$ has two *missing beliefs* (so there will be two active perception plans that need to be executed $A_{popt} = \{a_1^{popt}, a_2^{popt}\}$). Suppose the agent chose on the first iteration to execute $a_1^{popt}$ and found that the first precondition is true. The agent will ignore the relation between $a_1^{popt}$ to $a_2^{popt}$ and may choose to execute the other $k-2$ plans in $A'$ before executing $a_2^{popt}$.

### 3.3.3 Selective Active Perception (SAP)

SAP (Alg. 4) solves the problem of ITAP by allowing the agent to commit to a single plan $P_x$ which is a candidate to be $P_{opt}$ (line 2). If $P_x$ is in $M'$ (lines 3–4), SAP executes the appropriate active perception processes $A_{px}$ in a row (lines 5–10) until the plan is revealed. Then in the next loop iteration, the previous chosen $P_x$ can be chosen again if it is in $F'$. Line 8 executes the active perception plan without exiting the *for* loop and line 9 terminates the *for* loop if $P_x$ is not feasible. In case there are no missing options, SAP executes $P_x$ (lines 12–13).

**Algorithm 4** SAP BDI Loop

```
1: initialize-state
2: repeat
     1. options:= option-generator(event-queue)
     2. $P_x$:= deliberate(options)                  ▷ $P_x \in F' \cup M'$
     3. $F'$, $M'$:= sort-options($\{P_x\}$)
     4. if $P_x \in M'$ then
     5.     $A_{px}$:= get-plans-for-missing-beliefs($\{P_x\}$)
     6.     for $a_i^{px} \in A_{px}$ do
     7.         update-intentions($a_i^{px}$)
     8.         execute()
     9.         if $P_x \in N'$ then
    10.             continue
    11. else
    12.     update-intentions($P_x$)
    13.     execute()
    14. get-new-external-events()
    15. drop-unsuccessful-attitudes()
    16. drop-impossible-attitudes()
3:   end repeat
```

The SAP interpreter takes under consideration the relations between the active perception processes and the plan they reveal. SAP allocates active perception resources only after the agent committed to the selected *option* $P_x$. Once the selection has been done, if it is necessary the agent will execute a series of active perception processes $A_{pi}$. If as a result of the execution of the active perception processes $p_i$ is revealed as part of $F'$, the selector can choose it for execution, otherwise the selector will choose another candidate

In the robot example, first, the robot will choose between following the target, following the leader or driving home. Let us assume that the robot chose following the leader. Next, it will execute the appropriate active perception process that finds the leader's location and then, if it is feasible the robot will consider following the leader.

Choosing $P_x$ a candidate to be $P_{opt}$ is difficult. The methods for selection can vary and depend on the information supplied about the options. Lacking the information about the options in $M'$, $P_x$'s order of selection from $M'$ is a heuristic function that keeps the order between the function's iterations.

SAP focuses on the chosen plan $P_x$ and runs all the relevant active perception process until the selected plan is being revealed. The advantage in this method is that it allows the agent to disqualify methodically the chosen candidates without interfere of other plans as happens in ITAP.

The disadvantage of SAP is that it runs $P_x$'s active perception plans one after the other in a random order. Although the order of execution is not important for revealing a chosen $P_x$, it is useful to allow the agent to choose the order of execution for the active perception plans.

For example, a heuristic approach can use the information that same beliefs $b$ can support preconditions of $n$ plans $\{P_1..P_n\}$. By executing an active perception process for belief $b$ the agent can disqualify all the supported plans together. In our next algorithm we would use SAP's disqualification advantage.

#### 3.3.4 Double Selection Active Perception (DSAP)

We summarize the qualities we would like our algorithm to achieve based on our analysis of the previous versions:

1. As in ITAP the desired algorithm should allow the agent to decide during runtime between execution of active perception process or execution of $P_x \in F'$.
2. As in SAP the desired algorithm should execute active perception $a_i^k$ only when it is part of commitment to reveal $P_k$.
3. The desired algorithm should allow the agent to choose the order of active perception processes it runs.

Alg. 5 (DSAP) allows the agent to choose and commit to revealing a single $P_x$ and then allows the agent to choose again the next active perception plan within the domain of plans who support $P_x$.

As before, the algorithm first chooses one of the options $P_x$ (line 2). As in SAP, the selection function is a heuristic function intended to keep the order of selections, so if $P_x$ has been selected in the previous iteration it is most likely to be selected again. Then DSAP sorts the options to feasible and missing options (line 3). If $P_x$ is a missing option the algorithm starts a process of selection for the next active perception. The algorithm gets $P_x$'s missing beliefs $B_{px}$ (line 5) and all the missing beliefs from the plans in $M'$ (line 6) as a list ($T$). Then DSAP checks (line 7) on the beliefs in $T$ in order to choose the missing belief $b_{max}$ that is a *missing belief* in $P_x$ and also appears in most of the plans in $M'$ (using $T$). In lines 8–9 the algorithm uses the appropriate active perception plan and updates the agents intentions. In line 12 the agent executes the selected plan.

**Algorithm 5** DSAP BDI Loop

```
1: initialize-state
2: repeat
    1. options:= option-generator(event-queue)
    2. $P_x$:= deliberate(options)                    ▷ $P_x \in F' \cup M'$
    3. $F'$, $M'$:= sort-options(options)
    4. if $P_x \in M'$ then
    5.     $B_{px}$:= get-missing-beliefs($P_x$)
    6.     $T$:= get-missing-beliefs($M'$)
    7.     $b_{max}$:=argmax$_{b \in B_{px}}$ count(b in T)
    8.     $a_b^{px}$ :=get-plans-for-missing-belief($b_{max}$)
    9.     update-intentions($a_y^{px}$)
    10. else
    11.     update-intentions($P_x$)
    12. execute()
    13. get-new-external-events()
    14. drop-unsuccessful-attitudes()
    15. drop-impossible-attitudes()
3:   end repeat
```

This heuristic (lines 6-7) chooses the active perception process that supports $P_x$ and also support the highest number of plans in $M'$. In that way, if at the end of the active perception process the precondition found to be false, the maximal number of possible candidates is being disqualified.

Alg. 5 fills the requirements we described above. It allows the agent to choose an option $P_x$ also from $F'$, it executes active perception plan $a_i^k$ only when it is part of commitment to $P_x$, and lastly, it allows the agent to choose heuristically which active perception process will be executed.

## 4. A RUNNING TIME COMPARISON BETWEEN THE FOUR ALGORITHMS

The execution of the algorithms can be divided into two cases. Where there are no *missing beliefs* all four algorithms will operate the same way. But, when there are *missing beliefs* the algorithms will operate differently. We show that DSAP outperforms IAP in time complexity. It outperforms ITAP. And at last, DSAP outperforms SAP when it deliberately chooses the next active perception process.

In the next sections we will assume a case where the agent has $k$ options in $M'$ where each plan has $l$ missing beliefs and therefore $l$ active perception plans need to be executed before moving the selected $P_x$ to $F'$.

### 4.1 DSAP is more efficient than IAP

In most of the cases, the run time of a single plan—such as active perception plan—depends on various parameters such as the plan structure and components, behavior's termination conditions, the state of the world and the agent's perception process and world model. Due to the nature of these parameters, in most of the cases the run time of a single plan is being determined during execution time. Therefore in the next sections we will assume that the running time of all active perception plans is equal and denot it $b$.

When IAP interpreter gets to a decision junction it first executes *active perception* processes for every missing belief that is relevant to any of the given options. Assuming that the number of options in a single selection is $k$ and the number of *missing beliefs* in each option is $l$. The time that is added to the process as a result of the active perception integration is: $O(lkb)$. The time that will be invested in active perception processes does not change with respect to the order of selection of the candidates. Its worst case and best case run time are the same.

In contrast, when the DSAP interpreter gets to a decision junction it first selects an option and than executes a relevant active perception process. In the worst case, the agent will disqualify all $P_x$ and will run all the active perception processes exactly as IAP. In the best case, when the agent finds that the first $P_x \in F'$ the time that is added to the process as a result of the active perception integration is $O(lb)$.

### 4.2 DSAP is more efficient than ITAP

ITAP selects the plans from set $A$ without taking under consideration the relations between the active perception processes to the missing beliefs they reveal. While DSAP first selects $P_x$ and then executes one of the supporting active perception processes. Again assuming that there are $k$ plans where each plan has $l$ missing beliefs with $l$ active perception supporting plans. We prove that DSAP outperforms ITAP.

Theorem 1 and 2 prove that although that the running time of DSAP and ITAP in the best case is the same, the probability for DSAP to achieve the best case scenario is higher.

Theorem 1. *Let $T_{Db}$ and $T_{Ib}$ be the run time of DSAP and ITAP in the best case respectively. Then $T_{Db} = T_{Ib} = O(lb)$.*

Proof.
1. In the best case of ITAP, the agent will select all $l$ active perception processes of $A_{opt}$ one after the other and execute them. The running time is $O(lb)$.
2. In the best case of DSAP, $P_{opt}$ will be selected at first and the running time of $l$ active perception plans will be $O(lb)$.

From 1 and 2 we see that $T_{Db} = T_{Ib} = O(lb)$. □

Theorem 2. *Let $P_{Ib}$ and $P_{Db}$ be the probabilities for ITAP and DSAP to run the best case scenario respectively. Then $P_{Ib} \leq P_{Db}$.*

Proof.
1. ITAP chooses an active perception process to execute regardless of whether it supports $P_{opt}$. Therefore in the best case the probability for ITAP to execute first all $l$ active perception processes of $A_{opt}$ one after the other out of the $l \cdot k$ processes in $A$ is $P_{Ib} = \frac{1}{\binom{l\cdot k}{l}}$.
2. In the best case of DSAP, $P_{opt}$ will be selected as the first out of $k$ candidates. The probability for this case is: $P_{Db} = \frac{1}{k}$.
3. From 1 and 2, when $l > 1$, $P_{Ib} = \frac{1}{\binom{l\cdot k}{l}} < \frac{1}{k} = P_{Db}$.
4. From 1 and 2, when $l = 1$, $P_{Ib} = \frac{1}{k} = \frac{1}{k} = P_{Db}$.

From 3 and 4 $P_{Ib} \leq P_{Db}$. □

The difference between $P_{Ib}$ and $P_{Db}$ is significant. For example, when $l = 5$ and $k = 5$ then $P_{Ib} = 2 \cdot 10^{-5}$ while $P_{Db} = 0.2$.

Theorems 3 and 4 show that in the worst case the run time and the probability of both algorithms is the same.

Theorem 3. *Let $T_{Dw}$ and $T_{Iw}$ be the run time of DSAP and ITAP in the worst case respectively. Then $T_{Dw} = T_{Iw} = O(lkb)$.*

Proof.
1. In the worst case of ITAP the agent selects one of $l$ active perception plans of $A_{opt}$ as the last plan to be executed. The agent will have to execute the other active perception plans in $A$ before, and as $A$ contains $l \cdot k$ plans, the run time is $O(lkb)$.
2. In the worst case of DSAP, $P_{opt}$ will be selected the last. Again the agent will have to execute all the other plans in $A$, before executing the plans of $A_{opt}$. Therefore the run time is $O(lkb)$.

From 1 and 2 we see that $T_{Dw} = T_{Iw} = O(lkb)$. □

Theorem 4. *Let $P_{Iw}$ and $P_{Dw}$ be the probabilities for ITAP and DSAP to run the worst case scenario respectively. Then $P_{Iw} = P_{Dw}$.*

Proof.
1. In the worst case, ITAP chooses one of $l$ active perception plans of $A_{opt}$ as the last out of $l \cdot k$ plans in $A$. The probability of this case is $P_{Iw} = \frac{l}{l \cdot k} = \frac{1}{k}$.
2. In the worst case, the probability of DSAP to choose $P_{opt}$ as the last out of $k$ candidates is $P_{Dw} = \frac{1}{k}$.

From 1 and 2 we see that $P_{Iw} = P_{Dw} = \frac{1}{k}$. □

Figure 1 shows the difference between DSAP and ITAP where $l = 5$ and $k = 5$. The graph shows the values of $P_{ITAP}$ and $P_{DSAP}$ which are the probabilities to reveal $P_{opt}$ with less then x active perception processes using ITAP and DSAP accordingly. The X-axis measures the number of plans $x$ which are revealed, where the last ($x$th) plan revealed is $P_{opt}$. The Y-axis measures the probability of reaching $x$, i.e., the probability that less than $x$ active perception plans are executed before $P_{opt}$ is revealed.

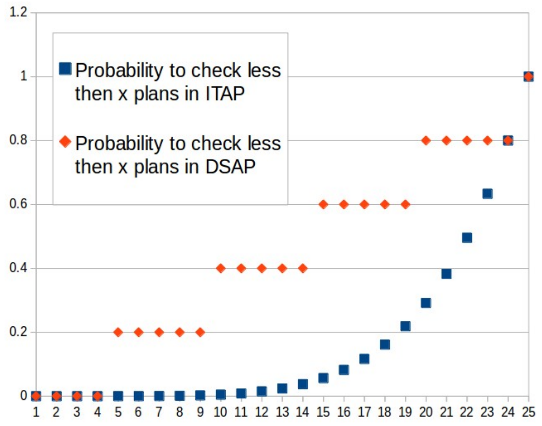

**Figure 1: The probability to check less then x plans in DSAP and ITAP**

The figure leads to two observations. First, for any $x$ between 1 to $l \cdot k$, $P_{itap} \leq P_{dsap}$. Second, the expected number of active perception plans (for the case $l = k = 5$) that has to be executed by ITAP is 22.6, while the expected number of such plans is 15, if DSAP is used. This is an improvement of 144%.

## 4.3 DSAP is more efficient than SAP

The key difference between SAP to DSAP is that after the selection of $P_x$ and in cases that active perception processes are needed, DSAP allows the agent to choose the order of execution for the active perception plans while SAP executes them in a random order.

Notice that each belief can support several options from $M'$. Therefore, when a belief is found to be false, it disqualifies not only $P_x$ but other plans too. We show the use of selection function that choose the order of active perception plans so the mean number of disqualified plans in each iteration will be maximized.

Definition 5 (MNDP). *The mean number of disqualified plans (MNDP) of a plan $P_x$ is the expected number of plans from $M'$ that will be disqualified in the process of revealing $P_x$, by executing the active perception plans supporting $P_x$ in a specific sequence (ordering) $r$. It is denoted $MNDP(P_x, r)$.*

Computing the NMDP is done through the following procedure. Let us assume that an agent choose plan $P_x$ that has

$l$ missing beliefs, the agent has to execute $l$ active perception plans in $A_{px}$ in order to make $P_x$ feasible. Let $R$ be a set of all possible ordering of execution for the plans in $A_{px}$. Let $r \in R$ be an ordering of execution that the agent chose. Let $a_i$ be the $i$-th active perception processes to be executed by $r$. Let $b_{ai}$ be the belief that $a_i$ resolves. Let $p_{bi}$ be the probability that $b_{ai}$ is true. Let $g_{ai}$ be the number of plans in $M'$ that $b_{ai}$ supports. The mean *number of disqualified plans* (MNDP) is calculated as follows:

$$MNDP(P_X, r) = (1 - p_{b1}) \cdot g_{b1} + p_{b1} \cdot (1 - p_{b2}) \cdot g_{b2} + \dots$$
$$\dots + [\prod_{j=1}^{l-1} p_{bj}] \cdot (1 - p_{bl}) \cdot g_{bl}$$

The $y$th plan in a specific ordering of the plans $r$ will disqualify $g_{by}$ plans in $M'$ only if all the beliefs $b_1 \dots b_{y-1}$ (that are related to the first $y-1$ plans in $r$) are true and $b_y$ is false. The probability for this to happen is: $\prod_{j=1}^{y-1} p_{bj} \cdot (1 - p_{by})$.

In order to maximize the number of disqualified plans, the agent should choose some ordering $r_{opt} \in R$ that maximizes the MNDP value, such that $r_{opt} = \text{argmax}_{r \in R}(MNDP(P_x, r))$. However, in order to do so, the agent needs to evaluate $p_{bj}$—the probability of belief $b$ to be true—for each belief. Without prior knowledge about the world $p_{bj}$ can not be evaluated. Therefore, we will assume the case of no information where the probability for any belief $j$ to be true ($p_{bj}$) or false ($1 - p_{bj}$) is 0.5.

When we assume that $p_{bj}$ equals to 0.5 the calculation of MNDP is as follows:

$$MNDP(P_x, r) = 0.5 \cdot g_{b1} + ... + 0.5^l \cdot g_{bl} = \sum_{x=1}^{l} 0.5^x \cdot g_{bx}$$

THEOREM 5. *Let $A_{px}$ be a group of active perception plans needed to be executed to reveal a plan $P_x$. Let $p_{bj}$ equals to 0.5 for all the beliefs that plans in $A_{px}$ resolves. Then, the ordering of the plans in $r$ that solves* $\text{argmax}_{r \in R} MNDP(P_x, r)$ *is set by $g_{bt} \geq g_{b(t+1)}$, for all $t < l$.*

PROOF. 1. Let us assume for contradiction that there are two active perception processes $a_{bx}$ and $a_{by}$ for two beliefs $b_x$ and $b_y$ where $g(bx) < g(by)$. In addition, there is an ordering $r_{fake} = \text{argmax}_{r \in R}(MNDP(P_x, r))$ where $a_{bx}$ is the $i$-th in $r_{fake}$ and $a_{by}$ is in the $i+j$ place. $p_{aj} = 0.5$ for any $j$.

2. $p_{aj} = 0.5$ for any $j$, therefore:

$$MNDP(P_x, r_{fake}) = \sum_{x=1}^{l} 0.5^x \cdot g_{bx}$$

3. $g_{bx}$ contributes $0.5^i \cdot g_{bx}$ to the value of $MNDP(P_x, r_{fake})$, while $g_{by}$ contributes $0.5^{(i+j)} \cdot g_{by}$. Their commune contribution can be written as:

$$0.5^i \cdot (g_{bx} + 0.5^j \cdot g_{by})$$

4. Due to the $0.5^j$ factor that the later program on the order $r_{fake}$ gets and because $g(bx) < g(by)$. It is obvious that if $b_x$ and $b_y$ will switch places in $r_{fake}$ their contribution will be higher. Because

$$g_{by} + 0.5^j \cdot g_{bx} > g_{bx} + 0.5^j \cdot g_{by}, \ if \ g(bx) < g(by)$$

5. Therefore, $r_{fake} \neq \text{argmax}_{r \in R} MNDP(P_x, r)$. Contradiction. $\square$

## 4.4 Comparison summary

In the last two sections we presented four algorithms that integrate BDI architecture with active perception. The main difference between the algorithms is the use of the deliberation mechanism. From our work it seems that there are major differences. While IAP does not allow any deliberation over the suggested active perception and executes all of them, ITAP is losing information that makes the deliberation mechanism inefficient. SAP succeed to over come ITAP's difficulties by using the information about connection between the active perception plans. And finally, DSAP improves SAP's performance by allowing the use of heuristic functions for a second selection.

The following table summarizes the differences between the suggested algorithms:

| | IAP | ITAP | SAP | DSAP |
|---|---|---|---|---|
| Worst case: | $O(lkb)$ | $O(lkb)$ | $O(lkb)$ | $O(lkb)$ |
| Best case: | $O(lkb)$ | $O(lb)$ | $O(lb)$ | $O(lb)$ |
| Best case probability: | 1 | $\frac{1}{\binom{lk}{l}}$ | $\frac{1}{k}$ | $\frac{1}{k}$ |

**Table 1: Time complexity and best case probability for the presented algorithms**

As IAP has no selection regarding the active perception processes, its best case run time is worse then all other algorithms. When considering the probability of achieving the best case run time, we found that because ITAP does not use the information regarding to connection between the relations between the active perception processes, its probability to achieve the same performance as DSAP and SAP is lower (Fig. 1) and also the probability to achieve the best runtime. Furthermore, we showed that using the double selection mechanism DSAP can outperform SAP by maximizing the number of disqualified plans in each iteration.

## 5. CONCLUSION AND FUTURE WORK

A basic building block in BDI is the set of beliefs an agent has over the world. However, in many cases due to the characteristic of the environment, there is no promise that during run time the beliefs' values will be available. active perception processes are the solution for handling missing beliefs during run time. However, in most of the cases they are used as ad-hoc solutions for a specific need, and are therefore assumed to be interleaved with the actions taken by the agent when it executes it domain-dependent plans.

The purpose of this article is to deal with active perception at the architecture level, specifically within the BDI loop. We present four algorithms that integrate active perception into the classic BDI loop. We show that different methods of integration create major differences in the running time of the algorithms. Finally, we suggested the DSAP algorithm that takes under consideration the limited available information in order to minimizes the amount of time the agent has to invest in execution of active perception processes.

## REFERENCES

[1] J. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. *International journal of computer vision*, 1(4):333–356, 1988.

[2] D. H. Ballard. Animate vision. *Artificial intelligence*, 48(1):57–86, 1991.

[3] L. De Silva, S. Sardina, and L. Padgham. First principles planning in BDI systems. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, volume 2, pages 1105–1112. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[4] D. Fox, W. Burgard, and S. Thrun. Active markov localization for mobile robots. *Robotics and Autonomous Systems*, 25(3):195–207, 1998.

[5] A. Guerra-Hernández, A. El Fallah-Seghrouchni, and H. Soldano. Learning in BDI multi-agent systems. In *International Workshop on Computational Logic in Multi-Agent Systems*, pages 218–233. Springer, 2004.

[6] G. A. Kaminka. No robot is an island, no team an archipelago: Plan execution for cooperative multi-robot teams. In *ICAPS 2015 Workshop on Planning and Robotics (PlanRob)*, 2015.

[7] G. A. Kaminka and I. Frenkel. Flexible teamwork in behavior-based robots. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, 2005.

[8] U. G. Ketenci, J.-M. Auberlet, R. Brémond, and E. Grislin-Le Strugeon. Improved road crossing behavior with active perception approach. In *Proc. Transportation Research Board Annual Meeting*, 2012.

[9] D. Kinny, M. Georgeff, and J. Hendler. Experiments in optimal sensing for situated agents. In *Proceedings of the Second Pacific Rim International Conference on Artificial Intelligence*, pages 1176–1182, 1992.

[10] Q. V. Le, A. Saxena, and A. Y. Ng. Active perception: Interactive manipulation for improving object detection. *Standford University Journal*, 2008.

[11] A. S. Rao and M. P. Georgeff. BDI agents: From theory to practice. In *ICMAS*, volume 95, pages 312–319, 1995.

[12] R. B. Scherl and H. J. Levesque. The frame problem and knowledge-producing actions. In *Proceeding of the AAAI conferance on Artificial Inteligance (AAAI 93)*, pages 689–695, 1993.

[13] C. Schlegel, J. Illmann, H. Jaberg, M. Schuster, and R. Wörz. Integrating vision based behaviours with an autonomous robot. In *Computer Vision Systems*, pages 1–20. Springer, 1999.

[14] M. Shanahan. Perception as abduction: Turning sensor data into meaningful representation. *Cognitive science*, 29(1):103–134, 2005.

[15] R. So and L. Sonenberg. The roles of active perception in intelligent agent systems. In *Multi-Agent Systems for Society*, pages 139–152. Springer, 2005.

[16] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[17] A. Unterholzner, M. Himmelsbach, and H.-J. Wuensche. Active perception for autonomous vehicles. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1620–1627. IEEE, 2012.

[18] D. Weyns, E. Steegmans, and T. Holvoet. Towards active perception in situated multi-agent systems. *Applied Artificial Intelligence*, 18(9-10):867–883, 2004.

[19] M. Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, 2009.