Constructivist Robots Assembling Value-Based Decision Heuristics

Tyler Olson , **Alex Gabriel** and **Carlos Hernandez Corbato**
Delft University of Technology
tyler@tyler-olson.com, {A.Gabriel, C.H.Corbato}@tudelft.nl

Abstract

Modern robots make decisions in many ways, but rely on their designers to choose which strategies to employ and when. AI and robotic architectures commonly use a single decision-making strategy, usually based on either multiple regression, which presumes the principle of total evidence, or optimization under constraints, which risks unbounded calculation of a stopping condition. When agents encounter situations violating these assumptions, their limited ability to adapt results in mission failure. Adopting a perspective of bounded rationality from the cognitive sciences, we propose a constructivist decision-making framework for autonomous agents to formulate their own decision strategies based on their mission-specific values. By semantically annotating different implementations of each modular component, it becomes possible to assemble and execute many different decision strategies. We represent each strategy as a behavior tree and assemble them using an automated theorem prover. For proof of concept, we simulate three example strategies used by a domestic robot performing a search and retrieval task, demonstrating that our method can assemble novel decision strategies and execute them on the order of a few seconds, and that it is applicable recursively to meta decisions.

1 Introduction

Without the deeper intuition employed by human decision makers, decision mechanisms in robotics and artificial intelligence (AI) are prone to spurious failures when they encounter unforeseen circumstances.

Often they implicitly assume that their models of the world are complete and must be confined to what Savage [1972] calls "small worlds", i.e. worlds that can be nearly completely captured by the agent's knowledge.

For robots to break out of the lab into open, uncertain worlds, they must adaptively make decisions in the context of their current situation. In these open environments, robot designers often cannot foresee all the problems their robots will encounter, so they are unable to instill in them a complete understanding. Contrary to *constructionist* architectures that are crafted and finely tuned by engineers, Thórisson *et al.* [2016] argue that self-organizing *constructivist* mechanisms are better suited to open worlds because they can deepen their understanding automatically. Via qualitative self-modeling, such mechanisms can adapt their decision processes in response to a dynamic environment and changing understanding of it Wolter *et al.* [2023].

Taking inspiration from the cognitive sciences, we propose that giving robotic agents an adaptive toolbox of decision strategies can enable them to formulate decisions on their own and consider their choices according to mission values.

The objective of this research is to demonstrate how a robot can make ecologically rational decisions in an open world by assembling decision strategies from an adaptive toolbox of components in a constructivist cognitive architecture.

Explicit components with a precisely designed purpose are superior in robot architectures because they are typically easier to compose, modify, add features to, and model [Scheutz and Andronache, 2004]. Accordingly, we develop an assemblable model of decision making, largely inspired by Kirsch [2019], and formulate the decision-making process using this model as a set of independent modules in the CoreSense cognitive robot architecture [Gabriel *et al.*, 2025]. We use a qualitative reasoning mechanism [Forbus, 2019] to select and compose modules into complete decision-making control structures. This approach encourages code reuse, makes decisions of robotic systems more explainable, and facilitates fine-grained self-adaptation through improved self-understanding.

Section 2 motivates our work with considerations of decision-making theory, ecological rationality, and heuristic decision making. Section 3 contributes a novel, formal, modular model of decision making that enables robots to automatically assemble decision strategies, and an approach to plan and execute them at runtime.

By implementing and testing a prototype in robotics, we demonstrate this method in action in Section 4, analyzing three example decision strategies used by a simulated domestic robot searching for a book in an apartment.

Based on the results, we discuss how the mechanism can be applied to robotics in Section 5, drawing conclusions and addressing future works in Section 6.

2 Background on Decision Making for Robots

The ultimate objective of robot decision making is to choose solution(s) which best align with its operator's combination of potentially conflicting values.

Problem formulation We start by adopting the problem conceptualization method of Haan and Heer [2012], which encodes a decision problem with two distinct, necessary parts: a *gap* and a set of *dilemmas*, as a common framework for decision making. The gap is a precise description of the difference between the current and desired state of the world. Dilemmas arise when solutions of varying quality force an agent to compromise between its values to close the gap. Adopting the terminology of Kirsch [2019] in this work, candidate solutions will be referred to as *alternatives*, while measures used to assess alternatives will be called *cues*.

Decision Process Model Kirsch [2019] proposes a general model of the decision-making process based on the weighted additive rule [Payne *et al.*, 1993] and *QuickEst* heuristic [Hertwig *et al.*, 1999], building on their own on modular decision-making procedures based on the HPS architecture [Kirsch, 2016]. Following Kirsch, we use the term *heuristic* to refer to any strategy fulfilling the decision-making process.

In Kirsch [2019], an initial *choice set* of decision alternatives are each scored by each cue in an initial *working set* of cues to produce an *assessment matrix*[1]. Next, these scores are aggregated together to produce a single utility score for each alternative with respect to the others in the choice set. If the alternative with the highest utility is "good enough", according to some acceptability criterion, it is returned. Otherwise the choice set and the working set of cues are updated and the procedure is called again.

2.1 Decision Principles

Now that we know how decisions can be made methodically, let's shortly cover different principles to base decisions on using the taxonomy of Gigerenzer *et al.* [1999].

Rationality When assuming perfect knowledge about the decision domain, as well as enough resources to exhaustively examine and evaluate all possible alternatives, agents can employ *rational* strategies, i.e. heuristics that deterministically lead to the best solution. Rational agents leveraging *optimization under constraints* assume the existence of a quality-effort tradeoff, stopping computation when the expected computation costs outweigh the expected benefits. Unfortunately, rational agents in open worlds are subject to the *Frame Problem*, as there is too much information available to determine what is relevant for a particular decision in the required amount of time.

Ecological Rationality When resources (e.g. processing power, deliberation time) are constrained or only partial knowledge of the decision domain is available, agents can employ *ecologically rational* strategies, adapting to the information structures and resource tradeoffs in their environment [Bullock and Todd, 1999; Gigerenzer, 2001].

[1]Kirsch [2019] calls this a "decision matrix"; Haan and Heer [2012] refer to this data structure as an "impact table". We have renamed each of these to avoid confusion.

Examples of this can be found in Payne *et al.* [1993] and Minsky [1988] who popularized models of cognition where agents employ various heuristics depending on the topic, effort, and required accuracy. Unfortunately, choosing *the best* heuristic under constraints is a new decision requiring another (best) heuristic, quickly leading to an *infinite regress* [Feeney, 2000] of meta-decisions, a circumstance termed the *Selection Problem* [Shanks and Lagnado, 2000].

Provided choices can be "good enough" without taking all available information into account or trading off performance, at some level of meta-decision a "good enough" heuristic should suffice [Todd and Gigerenzer, 2000; Gigerenzer, 2001].

2.2 Decision Metrics

Let's shortly discuss how operator values can be mapped to alternatives.

Implicit Value to Alternative Mapping When the scope of possible decisions and alternatives is small and well understood, operator values can be captured implicitly in a fixed set of predefined cues. Katsikopoulos *et al.* [2018] discuss how simple heuristics with a fixed cue-ordering are just as effective as complex heuristics in frequent operational decisions such as forecasting and inference problems.

Explicit Value to Alternative Mapping However for one-off strategic decisions frequently encountered in open worlds, it becomes infeasible to predefine this mapping in advance. Haan and Heer [2012] describe a method using what they call "goal trees" for breaking down an agent's abstract values into specific measurable cues for such problems. This process could also be followed in reverse to create more versatile abstract cues. Laird [2012] explore how value functions (cues) can be determined online by abstracting from background information and refined via reinforcement learning.

2.3 Decision Process Creation

Similar to how goal evaluation strategies can be either predefined or assembled at runtime, decision processes themselves can also be handled in two ways.

Predefined Decision Processes In small worlds, the number of decisions arising is small, and so we can predefine all the needed decision processes, leaving the engineer to choose the appropriate one for each situation encountered. Proposed mechanisms humans use to select predefined heuristics include unsupervised reinforcement learning [Rieskamp and Otto, 2006], cost-benefit approaches [Payne *et al.*, 1993], and exemplar-based models [Juslin *et al.*, 2003].

Processes Assembled at Runtime When things are less predictable, decision processes can still be assembled at runtime, provided an appropriate decision framework is available. Gigerenzer [2001] introduce the metaphor of a "backwoods mechanic" with a box of tools which may not be perfect for the problem at hand, but work well enough to get the job done. Dodampegama and Sridharan [2023] demonstrate such a method of constructing fast-and-frugal trees [Martignon *et al.*, 2008] to model heuristics used by allies and attackers in a fort defense scenario. In the next section we describe our proposed decision framework.

3 Methodology

We propose a novel constructivist framework for assembling decision heuristics at runtime based on a modular decision algorithm and composable heuristics implementations. By explicitly modeling the interfaces, requirements, and guarantees of each functional component implementation formally, we can use theorem proving techniques to ensure the generation of novel, valid heuristics at runtime.

Definition 1. Let a *gap* $G = \{A, \mathcal{W}, Q\}$ define a decision problem where A is a set of potential alternatives, $\mathcal{W}$ is a set of potential cues (see Section 3.1), and Q is a set of predicates encoding the problem requirements. We denote the set of all gaps understandable to an agent as $\mathcal{G}$.

Explicitly encoding requirements Q such as solution quality, cardinality, or even relative performance of chosen alternatives, facilitates heuristic assembly by allowing an agent to understand its problem-solving limits and expand them dynamically as it gains knowledge.

Definition 2. Let a *decision* be defined as a subset of the set of potential alternatives $A' \subseteq A$ called a *choice*, paired with a boolean indicating whether A' is acceptable $\top$ or not $\bot$ according to the requirements Q. Let $\mathbb{B} = \{\top, \bot\}$.

Going forward we use boldface math font to denote a power set, e.g. $\mathbf{A}$ is the power set of A.

3.1 Cues

Cues perform the function of assessing the tradeoffs caused by dilemmas. We formally define cues as scoring measures indicating some degree of perceived value or rank of each alternative with respect to the others in the choice set. Through the decision process an agent combines all these measurements into an overall value judgment for each alternative.

Definition 3. CUE : $\mathbf{A} \to \mathbb{R}^n$ Any function whose input is a choice set, $C \subseteq A$ of arbitrary size n, and output is a column vector scoring each alternative by some real-valued function $z : (\mathbf{A}, A) \to \mathbb{R}$ whose inputs are the choice set C and some member $a \in C$. For convenience we assume that a higher value is always encoded with a higher score.

To improve explainability, it is helpful if cues express human readable values, but they could also interface with distributed knowledge such as neural networks.

3.2 Decision Algorithm

Kirsch's algorithm provides a good base for an assemblable model of robot decision making due to its generality and modularity, but it has some limitations impeding automated assembly. Kirsch [2019] identifies several interdependencies between the heuristic components of their model, which must be resolved to ensure component compatibility for heuristic assembly. By modifying their algorithm, we remove some of these interdependencies, the rest of which we resolve using a runtime mechanism.

First, we introduce a new heuristic component TAKE that generalizes Kirsch's FIRST component to allow agents to choose a *set* of winning alternatives instead of just one alternative. Second, we distinguish between aggregating cues and ordering alternatives, removing the interdependency between Kirsch's ACCEPTABLE and AGGREGATEANDORDER components. Separating their responsibilities improves modularity and encourages reuse of functional component implementations between heuristics, reducing the size of an agent's component library.

Algorithm 1: Assemblable decision heuristic

Input: A gap, G, specifying the decision problem
Output: A decision: a choice $A' \subseteq A$ and a boolean acceptability status

```
1  C ← {}                    /* Init choice set */
2  W ← {}                    /* Init working set */
3  A' ← {}                   /* Init choice */
4  do
5      C ← UPDATEALTERNATIVES(G, C, A')
6      W ← UPDATECUES(G, W)
7      if C = {} or W = {} then
8          return ({}, ⊥)
9      end
10     M_W ← ASSESS(C, W)
11     E_IM ← AGGREGATE(M_W)
12     r_C ← ORDER(E_IM)
13     A' ← TAKE(C, r_C)
14 while not ACCEPT(A', E_IM)
15 return (A', ⊤)
```

Definition 4. DECIDE : $\mathcal{G} \to (\mathbf{A}, \mathbb{B})$ A decision heuristic is any function implementing decision making where the input is a gap $G \in \mathcal{G}$ and the output is a *decision*.

Our novel model realizes DECIDE as an iterative process composed of heuristic components following Algorithm 1. Each iteration the agent establishes a *choice set* of potential alternatives, $C \subseteq A$, and commits to a *working set* of cues, $W \subseteq \mathcal{W}$, to assess them by (lines 5 and 6 of Algorithm 1). This is followed by a pipeline of judging[2] each alternative relative to the others in the choice set (lines 10 and 11), then constructing a choice, $A' \subseteq C$, from the most preferred alternatives (lines 12 and 13). Upon failure to "accept" a choice (line 14) in a given iteration, the agent may use that information to update the working set and/or choice set in a subsequent iteration. In the event this function returns a rejected choice, $(A', \bot)$, the decision process has failed (e.g. line 8).

Component 1. UPDATEALTERNATIVES : $(\mathcal{G}, \mathbf{A}, \mathbf{A}) \to \mathbf{A}$ (line 5) Any function which takes as input a gap $G \in \mathcal{G}$, the previous choice set, $C_{-1} \subseteq A$, and the previously taken choice, $A' \subseteq C_{-1}$, and outputs a new choice set $C \subseteq A$.

Component 2. UPDATECUES : $(\mathcal{G}, \mathbf{W}) \to \mathbf{W}$ (line 6) Any function which takes as input a gap $G \in \mathcal{G}$, defining the current decision problem, and the previously used working set of cues, $W_{-1} \subseteq \mathcal{W}$, and outputs a new working set $W \subseteq \mathcal{W}$.

We assume that if either UPDATEALTERNATIVES or UPDATECUES returns the empty set, there is no relevant alternative/cue left and the agent has failed to make a decision.

[2]Here we borrow the term Shah and Oppenheimer [2008] use to describe the calculation of an overall value of each alternative.

Kirsch [2019] identifies that the retrieval processes for updating cues and updating alternatives may influence each other. Because we treat these two update components separately, it may require implementers to interleave or cache the results of interactive information gathering mechanisms.

Component 3. ASSESS[3]: $(\mathbf{A}, \mathbf{W}) \rightarrow \mathbb{R}^{n \times m}$ (line 10) Any function producing the assessment matrix $M_W : \mathbb{R}^{n \times m}$, where each *column* of M_W is an assessment of the complete choice set C, using a different cue in the working set W.

Definition 5. Let an *evaluation* of the choice set, $\mathcal{E}_{IM}$, be a matrix of size $n = |C|$ alternatives by $k = |IM|$ incommensurable[4] measures $im_{1...k}(a_i)$ where each *row* is a judgment of a different alternative in C.

Component 4. AGGREGATE : $\mathbb{R}^{n \times m} \rightarrow \mathbb{R}^{n \times k}$ (line 11) Any function producing an evaluation of the choice set $\mathcal{E}_{IM}$ from an assessment matrix M_W.

Intuitively, the AGGREGATE component formulates a wholistic evaluation of each alternative by combining subsets of assessments from co-commensurable cues into one of k incommensurable measures.

Component 5. ORDER : $\mathbb{R}^{n \times k} \rightarrow \mathbb{Z}_n$ (line 12) Any function that produces a preference relation over the choice set $r_C \in \mathbb{Z}^n$ from an evaluation $\mathcal{E}_{IM}$, where r_C is encoded as a list of integer ranks (lower is better) corresponding to the alternative in the choice set with the same index. Alternatives of the same rank are of equal preference.

By ordering the alternatives, an agent takes a stance on incommensurable measures, prioritizing some over others, by collapsing the multi-axis evaluation into a single-axis preference relation (see discussion of social choice functions in Kirsch [2019]). This ensures all alternatives in the choice set are strongly connected, avoids ambiguous cyclical preferences, and can represent ties between alternatives.

Component 6. TAKE : $(\mathbf{A}, \mathbb{Z}^n) \rightarrow \mathbf{A}$ (line 13) Any function which takes the choice set C and a preference relation over it r_C as input, and outputs a choice $A' \subseteq C$. To promote early stopping, we add the restriction that there must exist some well-defined indicator function $\mathbf{1}_{A'} : A \rightarrow \mathbb{B}$ where $\forall(C), (a \in \text{TAKE}(C) \implies \mathbf{1}_{A'}(a) = \top)$ specifying the inclusion or exclusion of each alternative into the choice A'.

Applying this indicator function to each $a \in C$ gives TAKE linear computational complexity with respect to $|C|$.

By considering alternatives in order, TAKE functions can ensure that no alternative strictly preferred to those included in the choice is excluded and can stop early when decision problem size and fitness requirements are satisfied.

While the TAKE component can enforce fitness requirements on individual alternatives, it does not enforce acceptability requirements on the choice as a whole.

Component 7. ACCEPT : $(\mathbf{A}, \mathbb{R}^{n \times k}) \rightarrow \mathbb{B}$ (line 14) Any function indicating if a choice, A', has been accepted ($\top$) or rejected ($\bot$), optionally conditioned on the evaluation, $\mathcal{E}_{IM}$, used to construct it.

Iteration is bounded using some boolean acceptability criteria, called a stopping rule by Gigerenzer [2001], ensuring the choice meets all the requirements specified in the gap Q.

3.3 Assembling and Executing Heuristics

To make heuristic assembly possible, we apply the model-based approach to cognition of the CoreSense Architecture [Gabriel *et al.*, 2025]. Each heuristic is constructed as a reusable behavior tree [Colledanchise and Ögren, 2018], using a backward-chaining method to promote proactive explainability. We create independent programs for each heuristic component, and two models of them: i) behavior tree fragments for modular program composition and execution, and ii) semantic models in Typed First-order Form [Sutcliffe, 2024] that conform to the agent's model for planning cognitive behaviors.

Given a gap G captured in first order logic, we add a meta-model of the desired decision to the knowledge base and derive a valid assembly of heuristic components using the agent's cognitive behavior assembly system,[5] based on the Vampire Automated Theorem Prover[6] [Bártek *et al.*, 2025]. A valid heuristic ensures that a successful decision meets all of the problem requirements Q. The agent executes the assembly plan by composing the corresponding behavior tree fragments and executing the assembled tree.

[3]Kirsch calls this component FILLDECISIONMATRIX.

[4]Commensurable metrics can be judged by a single measure, making them comparable, where incommensurable metrics cannot.

[5]https://github.com/CoreSenseEU/coresense_understanding

[6]Vampire 4.9 Release, https://github.com/vprover/vampire

Decision	**Character**			**Cues**							**Components**						
	Assembly Time	**Abstraction**	**Cardinality**	/distance	/visited	/in_doorway	/is_book	/validity	/doorway_st	/habits	UPDATE ALTERNATIVES	UPDATECUES	ASSESS	AGGREGATE	ORDER	TAKE	ACCEPT
Decide-On-Room	R	L	1	?	?				?	?	$K_{A'}$	TTB	$A_{\|\|}$	I	Dom	EW_∞	Siz
Take-The-Best	P	H	n					x			K_0	Re_∞	$A_{\|\|}$	I	Lex	TB_1	Sat
Decide-On-Objects	P	L	n	x	x	x	x				K_0	Re_0	$A_{\|\|}$	Daw	Cop	TB_∞	All

Table 1: Decision Comparison

4 Evaluation

To evaluate our constructivist decision framework we have implemented it in a domestic robotic task. Domestic robotics confronts autonomous agents with hard-to-predict decision scenarios. We expect that constructivist approaches like our own increase robustness and adaptability for agents in such environments, due to their ability to adapt beyond preprogrammed decisions. Our evaluation focuses on two aspects: displaying our method's feasibility and the benefits of a modular approach.

The current state of the art of decision making in domestic robot systems consists mainly of predefined decision strategies (with fallback and recovery) developed for known task scenarios. We reproduce this setup with a predefined decision heuristic in one of our three evaluation scenarios. In the other two, we move beyond the state of the art to demonstrate decision heuristic assembly at runtime including using a predefined meta-level heuristic to decide on the cues in a non-meta-level heuristic.

4.1 Library of Components

This prototype, built on ROS 2 [Macenski *et al.*, 2022], uses a library of modular components based on heuristics found in cognitive science [Brandstätter *et al.*, 2006; Dawes, 1979; Gigerenzer and Brighton, 2009; Gigerenzer *et al.*, 1999; Kirsch, 2019; Payne *et al.*, 1993; Shah and Oppenheimer, 2008; Simon, 1956; Todd and Gigerenzer, 2000; Tversky, 1972] and robotics [Kotseruba and Tsotsos, 2020; Maes, 1989] literature.[7] Individual heuristic components are implemented as parametrized ROS nodes which communicate with the behavior tree via ROS actions and services. Before starting an assembled heuristic, the agent first prepares each component by adapting the parameters of each of the relevant ROS nodes.

The rest of this subsection briefly describes each component from this library used in the experiment (see Table 1).

UPDATEALTERNATIVES $K_{A'}$ keeps all alternatives chosen in the previous iteration and K_0 keeps none.

UPDATECUES TTB updates cues using the *Take-The-Best* heuristic [Gigerenzer and Goldstein, 1996] described in Section 4.3. Re_∞ reuses the same cues every iteration. Re_0 uses one cue per iteration without reuse.

ASSESS The same component, $A_{||}$ is used in each tested heuristic to compute each cue in parallel.

AGGREGATE I assumes that each cue directly measures an incommensurable value of the agent, equivalent to the identity function. Implementing Dawes' Rule [Dawes, 1979], Daw interprets assessments as only positive or negative and tabulates the difference.

ORDER Dom ranks alternatives by counting the number of dimensions in which each dominates the others. Lex imposes a lexicographical ordering, breaking ties with each subsequent evaluation axis. Cop uses the Copeland method: rank each alternative by the difference in the number of others that are worse than it and better than it for each evaluation axis.

[7] Source code and complete experimental results can be found in the supplementary material.

TAKE Each iteration EW_∞ eliminates all alternatives tied for worst, unless they are all tied.In contrast, TB_1 and TB_∞ take one or all of the alternatives tied for best.

ACCEPT Siz accepts a choice based on the number of chosen alternatives. This component is leveraged by heuristics that modify the size of the choice set between iterations. Sat is a satisficing ACCEPT component, accepting a choice if all chosen alternatives meet a threshold value, i.e. they are "good enough." All always accepts; heuristics using it will not iterate. All can be used when all problem requirements are (implicitly) guaranteed by the other heuristic components, e.g. TB_1 enforces a choice cardinality of 1.

4.2 Experiment Scenario

An autonomous robot is tasked with finding a specific book in a simulated domestic environment consisting of 4 rooms: kitchen, living room, bedroom, office, based on the Robocup@Home 2026 setting [Hart *et al.*]. Clues to the location of the book can be inferred from prior knowledge about the habits of the occupants, stored in a Prolog database, as well as specific items found in the environment, e.g. a dirty cup in the bedroom.

The robot has to make three decisions: 1) decide on a room to search next, and for that decision: 2) decide which cues are relevant; 3) decide which objects in that room to investigate.

We aim to showcase feasibility across a number of relevant decision-dimensions: **Assembly Time** of heuristic (**R**untime/**P**redefined), level of decision **Abstraction** (**L**ow/**H**igh), **Cardinality** of solution (**1/n**). Table 1 summarizes how the three decision tasks solutions cover this space.

4.3 Decisions

Using our framework, we have modeled cues and heuristic components, as well as the decisions with their alternatives in turn. For the predefined decision heuristics, we have manually defined the decision heuristic, i.e. associated the decision with heuristic components and cues. For the heuristic generated at runtime, the components are chosen by the cognitive behavior assembly system, which plans as part of this decision, another, meta-level cue decision to choose the cues.

Decide-On-Room

In the first decision problem, the gap is the lack of a target room; note that rooms could be investigated multiple times.

Listing 1: Snippet of the robot's Prolog knowledge base

```
gap(room_gap).
alternative_of(A, room_gap) :- room(A).
requirement_of(absolute_size_1_req,
    room_gap).
available_for('/visited', room_gap).
available_for('/habits', room_gap).
available_for('/doorway_status', room_gap).
available_for('/distance', room_gap).
validity('/visited', room_gap, 0.8).
validity('/habits', room_gap, 0.7).
validity('/doorway_status', room_gap, 0.6).
validity('/distance', room_gap, 0.6).
```

This problem contains two dilemmas: the amount of time taken to complete the mission, and the predicted probability of mission success upon searching the chosen room.

We model the gap with the Prolog clauses in lines 1–3 of Listing 1. The alternatives are naturally the four rooms in the apartment (line 2). Because the robot has to explore rooms sequentially, it adds a requirement (line 3) that the choice must have a cardinality of one (see Table 1 **Cardinality**) .

As there are no predefined cues, the UPDATECUES component triggers a meta-decision on which cues to use each iteration. In Table 1 this is reflected by the **R**untime label in the **Assembly Time** column and question marks denoting the alternatives for the meta-level decision.

Take-The-Best

The gap of *Take-The-Best* is the robot's lack of a set of relevant cues to commit to in the current iteration of *Decide-On-Room*. Dilemmas include how quickly the chosen cues can be executed, how likely they are to reduce the number of iterations of *Decide-On-Room* required, and how well they are expected to predict which room contains the book.

We use four cues as alternatives for this meta-decision: `/visited`, `/habits`, `/doorway_status`, `/distance` (lines 4–7).

Take-The-Best uses one predetermined (meta-)cue, *validity* (see Table 1 **Cues**), which measures the true positive rate of each cue in making past predictions. Perhaps this versatile heuristic was predefined, providing a fixed starting point to escape infinite regress. We assigned validities to each of the cues available for *Decide-On-Room* for demonstration purposes (lines 8–11).

We impose a satisficing threshold requiring cue validity to be at least 0.5, allowing the heuristic to operate when the full set of cues is not known in advance. This enables explicit modeling of decision failure, such as terminating the search for additional cues, improving robustness.

Decide-On-Objects

The gap of the third decision problem is the selection of a set of objects in the current room to investigate. Dilemmas are the amount of time taken to investigate each object and the predicted probability of those objects being the book.

This decision problem places no requirements on the size or quality of the choice, as any number of objects could be investigated so long as the most promising candidates are investigated first. Making a choice of unspecified size is made possible by the introduction of the TAKE component. We implemented the *Decide-On-Objects* heuristic by hand to demonstrate the modularity, versatility, and robustness of our model.

We simulate four cues the robot has identified as relevant for selecting objects: `/distance`, `/visited`, `/in_doorway`, and `/is_book`.

4.4 Results

We successfully simulated 50 runs of the example scenario with results broken down for each heuristic reported in Table 2. Total runtimes are the sum of one time costs (columns 5–6) and per iteration costs (columns 7–9) times the number of iterations. Note that the total runtime of *Decide-On-Room* had a bimodal distribution where trials taking 3 and 4 iterations had average runtimes of 3358 ms and 4189 ms respectively. Profiling was performed using Ubuntu 22.04 LTS on a consumer laptop featuring an Intel i9-12900H processor and 16 gigabytes of memory.

Naturally, as the *Decide-On-Room* heuristic uses *Take-The-Best* internally each iteration, its total runtime is much longer than the other two heuristics. Time spent assembling the *Decide-On-Room* heuristic only represented 8% of the total, demonstrating that it is inexpensive to create new heuristics at runtime for iterative decisions.

With total runtimes on the order of a few seconds, a robot could use our method to make decisions with medium to long term effects, but it is unsuitable for reactive decisions such as avoiding obstacles or having conversations.

The total average time taken for each decision was significantly larger than the sum of the time taken to execute the heuristic components themselves. Results presented in Table 2 show that the majority of decision time is spent setting up the behavior trees and communicating between heuristic components. This suggests that using ROS actions to communicate between the heuristic components and the behavior tree executor is a costly architectural design. For all but the ASSESS component, there was an insignificant difference in runtime per iteration between the three decisions. UPDATECUES and UPDATEALTERNATIVES took 5 ms each on average, and the AGGREGATE, ORDER, TAKE, and ACCEPT components ran for $1-3$ ms on average.

On the other hand, the ASSESS component was a clear outlier with a wide range of $12-54$ ms runtimes. This range can be largely explained by a difference in the number of alternatives and cues assessed each iteration.

These results suggest that with inexpensive cues, like those we simulated, agents concerned with decision speed can ignore minor performance differences between different heuristic components when assembling their heuristics. Instead, it is more effective for them to focus on minimizing the required number of iterations, such as by restricting the total number of cues and alternatives to be assessed.

By successfully executing the three heuristics, this model proves capable of representing a wide variety of problems and heuristics. The *Take-The-Best* heuristic demonstrates how decisions can be made at multiple levels of abstraction, while the incorporation into *Decide-On-Room* showcases integrated multi-level decision making. The model was able to execute both commensurable and incommensurable heuristics, resp. *Decide-On-Objects* and *Decide-On-Room*, and solve problems with differing cardinality and quality requirements.

This proof-of-concept demonstrates how our method displays high modularity. By using modular cues that share a common interface (see Table 1), *Decide-On-Room* was able to progressively eliminate rooms without changing any components. All three simulated heuristics were composed from a common library of 37 modular components. With just 18 finite-domain parameters, of which at most 13 will ever be used simultaneously by the heuristic assembly mechanism, this library can represent 15552 unique heuristics. This compactness reduces assembly effort whilst enabling agents to finely tune their decision strategies to their environment.

Heuristic	Number of Trials	Avg. Number of Iterations	Avg. Overall (ms)			Avg. Iteration (ms)		
			Total Runtime	Assembly	Behavior Tree Manipulation	Sum Heuristic Components	Component Communication	Self Adaptation
Decide-On-Room	50	3.4	3690	281	275	40	340	22
Take-The-Best	170	1	506	-	285	36	165	21
Decide-On-Objects	50	1	579	-	297	66	196	21

Table 2: Profiling results for each example heuristic overall & per iteration

Our prototype also demonstrates the versatility of cues and components in our model. As seen in Table 1, various heuristic components are shared between the three different heuristics. Of the 17 unique components making up the simulated heuristics, 3 are reused directly and 10 use the same ROS node as another, just with different parameters. The `/distance` and `/visited` cues were reused for multiple decisions as they reflect the same stance the agent has towards the alternatives. Limiting the number of parameters improves explainability and eases automated assembly.

5 Discussion

Our prototype for assembling and executing decision strategies at runtime shows how a constructivist approach is feasible for domestic autonomous robots to make decisions that align with the mission values, following the more realistic assumption of bounded rationality.

5.1 Model

Our model of decision making is designed to be applicable to many decision problems and domains via different implementations of the various decision components and cues. By leaving the precise nature of cues and components unspecified except for their interfaces, we allow users to adapt the model to their needs. However, this open design space also increases the effort required to apply it to specific decision making problems. Specific components and cues have to be defined, before a given decision can be explained.

With small modifications to our algorithm, partial heuristics could be processed in multiple parallel streams that are later recombined to complete the decision. For example, Kainen [2000] describes the *first-fit-decreasing* heuristic that uses two reasonable strategies in parallel but takes the result of whichever finishes first. There is evidence that humans use partial heuristics before abandoning them and jumping to another [Huber, 2000].

While the prototype we evaluated uses discrete alternatives, our approach could be extended to continuous choice sets with some reformulation. For example, if the choice is discrete but the choice set is continuous, then the assessment matrix, evaluation, and ordering become continuous maps with ranges of dimension $|W|$, $|IM|$, and 1 respectively. The TAKE component then samples some number of solutions from their composition. Alternatively, if the choice is also a continuous region then the problem could be reformulated as a decision on the bounds of that region.

5.2 Prototype

Our implementation uses and integrates into a pre-existing robot architecture. This provides some benefits (such as existing cognitive behavior planning) but also comes with limitations. In the following we list some desirable features still missing in this prototype.

Towards Solving The Selection Problem

Our prototype demonstrates how an agent can employ a meta-heuristic to decide which heuristic components to use for a given decision (the selection problem). This meta-heuristic is a seed from which the agent grows its library of heuristics. Under the hood, our heuristic assembly mechanism derives a collection of potential heuristics using a theorem prover. It then uses the *Availability* heuristic [Tversky and Kahneman, 1973], selecting the first acceptable alternative.

Further research is needed to design more ecologically rational selection problem heuristics and the cues they employ to tackle real world problems. We suspect that high modularity eases the tradeoff identified by Rieskamp and Otto [2006] between better decision quality and selection problem difficulty as agents gain access to more heuristics.

To improve their meta-heuristics, robots need mechanisms to generalize their understanding of the consequences of their decisions in relation to changing environmental structure. One potential solution with a developed literature is the Structure Mapping Engine (SME) which uses analogical reasoning for model generalization [Forbus *et al.*, 2017]. For example, ALIGN [McLure *et al.*, 2015] uses SME to progressively learn clustering cues by identifying and generalizing from near misses. The ALIGN algorithm could be used to develop meta-cues relating heuristic components to problem requirements and environmental conditions.

Cue-Modeling

A natural extension of our model is a mechanism for retrieving alternatives and cues from domain knowledge or deriving specific cues from abstract values. One possible step researchers could take in retrieving relevant cues is encoding a link between the agent's values and how each cue measures

those values for different classes of alternatives. Laird *et al.* [2012] approach this task in the Soar architecture by proposing an expected value function based on background knowledge, then refining it with reinforcement learning. Haan and Heer [2012] describe a tool called causal diagrams to connect observable properties to mission goals that we think merits further investigation.

Alternative Representations for Heuristic Assembly
We modeled heuristic components within the CoreSense framework, but alternative representations such as Qualitative Process Theory (QPT) [Forbus, 1984] could also facilitate heuristic assembly. QPT has so far been used for descriptive problems such as model formulation, limit analysis, and inference resolution [Forbus, 2019] but could be modified to permit operational value-based decision making. Where QPT analyzes causal dynamics by modulating the activation of model fragments, the CoreSense Understanding System generates strategies for the creation of model fragments using models of cognitive processes. Integrating existing QPT ontologies could provide an agent with better understanding of causal relationships between their decision heuristics, outcomes of their choices, and mission values.

5.3 Evaluation

We proved that adaptive heuristic assembly is possible, using a carefully selected, but small set of decision tasks in a single environment. Extension of this proof-of-concept to a complete robot mission is a natural next step in testing its strengths and weaknesses.

While generally applicable heuristics will by nature perform worse on any particular problem compared to fine-tuned ones, a comparison with such specialized systems is nevertheless useful. Such a comparison should however reflect robotic environments, i.e. not be hand-crafted to the specific decision capabilities implemented in the system under test, but include a wide range of problems [Kirsch, 2017]. This requires developing evaluation schemes which are difficult enough that they can demonstrate effects of complex environmental structures, such as *flat maximum* [Dawes, 1979] and *less-is-more* [Czerlinski *et al.*, 1999], while still being measurable enough to facilitate scientific analysis.

6 Conclusion

This work shows how constructivist approaches can increase agent generality by building complex capabilities from a set of simple components with common interfaces. We expect that explicitly modeling our desires, rather than inventing clever approximations, produces robots that share our values.

For robots to break out of the lab, we encourage the research community to embrace the lens of ecological rationality. With this fresh perspective, we present robots with a new kit of tools for completing their missions.

Modern robots make decisions in many ways, but rely on their designers to choose which strategies to employ and when. By using explicitly modeled values, taking cognitive short cuts, and tactfully leveraging information, they can robustly make novel and explainable decisions in an open and uncertain world.

Acknowledgement

This work has been supported by the European Union through the Horizon Europe CoreSense project (grant no. 10107025).

References

Eduard Brandstätter, Gerd Gigerenzer, and Ralph Hertwig. The priority heuristic: Making choices without trade-offs. *Psychological Review*, 113(2):409–432, 2006.

Seth Bullock and Peter M. Todd. Made to Measure: Ecological Rationality in Structured Environments. *Minds and Machines*, 9(4):497–541, November 1999.

F. Bártek, A. Bhayat, R. Coutelier, M. Hajdu, M. Hetzenberger, P. Hozzová, L. Kovács, J. Rath, M. Rawson, G. Reger, M. Suda, J. Schoisswohl, and A. Voronkov. The VAMPIRE Diary. In *Computer Aided Verification*, volume 15933, pages 57–71, Cham, 2025. Springer Nature.

Michele Colledanchise and Petter Ögren. *Behavior Trees in Robotics and AI: An Introduction*. CRC Press, July 2018.

Jean Czerlinski, Gerd Gigerenzer, and Daniel G. Goldstein. How good are simple heuristics? In *Simple heuristics that make us smart*, Evolution and cognition, pages 97–118. Oxford University Press, New York, NY, US, 1999.

Robyn M. Dawes. The robust beauty of improper linear models in decision making. *American Psychologist*, 34(7):571–582, 1979.

Hasra Dodampegama and Mohan Sridharan. Knowledge-based Reasoning and Learning under Partial Observability in Ad Hoc Teamwork. *Theory and Practice of Logic Programming*, 23(4):696–714, July 2023.

Aidan Feeney. Simple heuristics: From one infinite regress to another? *Behavioral and Brain Sciences*, 23(5):749–750, October 2000.

K. D. Forbus, R. W. Ferguson, A. Lovett, and D. Gentner. Extending SME to Handle Large-Scale Cognitive Modeling. *Cognitive Science*, 41(5):1152–1201, July 2017.

Kenneth D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24(1-3):85–168, December 1984.

Kenneth D. Forbus. *Qualitative Representations: How People Reason and Learn about the Continuous World*. The MIT Press, January 2019.

Alex Gabriel, Carlos Hernandez, Ricardo Sanz, Esther Aguado, Miguel Fernandez-Cortizas, G. GP-Lenza, I. Gonzalez, F.J. Rodriguez, and Francisco Martin. Specification of the CoreSense Architecture. Deliverable D2.2, The CORESENSE Project, May 2025.

Gerd Gigerenzer and Henry Brighton. Homo Heuristicus: Why Biased Minds Make Better Inferences. *Topics in Cognitive Science*, 1(1):107–143, January 2009.

Gerd Gigerenzer and Daniel G. Goldstein. Reasoning the fast and frugal way: Models of bounded rationality. *Psychological Review*, 103(4):650–669, 1996.

Gerd Gigerenzer, Peter M. Todd, and ABC Research Group. *Simple Heuristics That Make Us Smart*. Evolution and Cognition. Oxford University Press, 1999.

Gerd Gigerenzer. The Adaptive Toolbox. In R Selten and Gerd Gigerenzer, editors, *Bounded Rationality: The Adaptive Toolbox*. The MIT Press, February 2001.

Alexander de Haan and Pauline de Heer. *Solving complex problems: professional group decision-making support in highly complex situations*. Eleven International Publishing, The Hague, The Netherlands, 2012.

J. Hart, A. Moriarty, K. Pasternak, J. Kummert, M. Leonetti, L. Contreras, L. Ruegemer, A. Mitzutani, T. Ribeiro, A. Golding, T. Kang, and F. Pimentel. Robocup@home rulebook. https://robocupathome.github.io/RuleBook/rulebook/master.pdf#page=8.09. Revision 2026-01-14.

Ralph Hertwig, Ulrich Hoffrage, and Laura Martignon. Quick estimation: Letting the environment do the work. In *Simple heuristics that make us smart*, Evolution and cognition, pages 209–234. Oxford University Press, 1999.

Oswald Huber. What's in the adaptive toolbox: Global heuristics or more elementary components? *Behavioral and Brain Sciences*, 23(5):755–755, October 2000.

Peter Juslin, Sari Jones, Henrik Olsson, and Anders Winman. Cue abstraction and exemplar memory in categorization. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 29(5):924–941, 2003.

Paul C. Kainen. The role of mathematics in heuristic performance. *Behavioral and Brain Sciences*, 23(5):755–756, October 2000.

Konstantinos V. Katsikopoulos, Ian N. Durbach, and Theodor J. Stewart. When should we use simple decision models? A synthesis of various research strands. *Omega*, 81:17–25, December 2018.

Alexandra Kirsch. Heuristic decision-making for human-aware navigation in domestic environments. In *2nd global conference on artificial intelligence (GCAI)*, 2016.

Alexandra Kirsch. Lessons from human problem solving for cognitive systems research. *Advances in Cognitive Systems*, 5:13–24, 2017.

A. Kirsch. A unifying computational model of decision making. *Cognitive Processing*, 20(2):243–259, May 2019.

Iuliia Kotseruba and John K. Tsotsos. 40 years of cognitive architectures: core cognitive abilities and practical applications. *Artificial Intelligence Review*, 53(1):17–94, 2020.

John E. Laird, Nate Derbinsky, and Miller Tinkerhess. Online determination of value-function structure and action-value estimates for reinforcement learning in a cognitive architecture. *Advances in Cognitive Systems*, 2:221–238, 2012.

John Laird. *The Soar cognitive architecture*. MIT Press, Cambridge, Mass London, England, 2012.

Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, May 2022.

Pattie Maes. How to do the Right Thing. *Connection Science*, 1(3):291–323, January 1989.

Laura Martignon, Konstantinos V. Katsikopoulos, and Jan K. Woike. Categorization with limited resources: A family of simple heuristics. *Journal of Mathematical Psychology*, 52(6):352–361, December 2008.

Matthew McLure, Scott Friedman, and Kenneth Forbus. Extending Analogical Generalization with Near-Misses. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), February 2015.

Marvin Minsky. *Society Of Mind*. Simon and Schuster, March 1988.

John W. Payne, James R. Bettman, and Eric J. Johnson. *The Adaptive Decision Maker*. Cambridge University Press, 1 edition, May 1993.

Jörg Rieskamp and Philipp E. Otto. SSL: a theory of how people learn to select strategies. *Journal of experimental psychology: General*, 135(2):207, 2006.

Leonard J. Savage. *The foundations of statistics*. Dover publ, New York, 2nd rev. ed edition, 1972.

M. Scheutz and V. Andronache. Architectural Mechanisms for Dynamic Changes of Behavior Selection Strategies in Behavior-Based Systems. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, 34(6):2377–2395, December 2004.

Anuj K. Shah and Daniel M. Oppenheimer. Heuristics made easy: An effort-reduction framework. *Psychological Bulletin*, 134(2):207–222, 2008.

David R. Shanks and David Lagnado. Sub-optimal reasons for rejecting optimality. *Behavioral and Brain Sciences*, 23(5):761–762, October 2000.

Herbert A. Simon. Rational choice and the structure of the environment. *Psychological review*, 63(2):129, 1956.

Geoff Sutcliffe. Stepping Stones in the TPTP World. In Christoph Benzmüller, Marijn J.H. Heule, and Renate A. Schmidt, editors, *Automated Reasoning*, pages 30–50, Cham, 2024. Springer Nature Switzerland.

Kristinn R. Thórisson, David Kremelberg, Bas R. Steunebrink, and Eric Nivel. About Understanding. In Bas Steunebrink, Pei Wang, and Ben Goertzel, editors, *Artificial General Intelligence*, pages 106–117, Cham, 2016. Springer International Publishing.

Peter M. Todd and Gerd Gigerenzer. How can we open up the adaptive toolbox? *Behavioral and Brain Sciences*, 23(5):89–100, October 2000.

Amos Tversky and Daniel Kahneman. Availability: A heuristic for judging frequency and probability. *Cognitive Psychology*, 5(2):207–232, September 1973.

Amos Tversky. Elimination by aspects: A theory of choice. *Psychological Review*, 79(4):281–299, July 1972.

Diedrich Wolter, Felix Haase, and Alexandra Kirsch. AI Birds: Obstacles for Reaching Human-Level Performance and a New Role for Qualitative Reasoning. Technical report, University of Amsterdam, 2023.