Evolution-inspired incremental development of complex autonomous intelligence

Torbjørn S. Dahl
Norwegian Defence Research Establishment
2027 Kjeller, Norway
Torbjorn-Semb.Dahl@ffi.no

Christophe Giraud-Carrier
ELCA Informatique SA
Lausanne 13, Switzerland
cgc@elca.ch

Abstract

This article presents Hypothetical Evolutionary Paths (HEP), an evolution-inspired methodology for incremental development of complex adaptive robot control algorithms. HEP produces algorithms that adapt without learning centralized hypotheses in the same way that behavior-based approaches in general avoid the use of centralized world models. The algorithms produced adapt rapidly without suffering from the large search spaces and long convergence times that have made traditional learning algorithms unattractive in situated systems. By applying multiple problem solving strategies concurrently, the algorithms keep a high performance-level during learning. We present two sets of experiments in simulation demonstrating rapidly adapting algorithms developed using HEP in the problem domains of mapping and conflict resolution.

1 Introduction

One of the promising prospects of behavior-based (BB) artificial intelligence was that complex autonomous intelligent systems could be built incrementally, mirroring the way evolution has produced such systems through a process of stepwise additions and adjustments [2]. The work presented here expands on this aspect of BB development. The focus is in particular on the role of memory in adaptation.

Consider an animal that competes with other members of its own species for a limited resource. Such an animal can have a number of strategies for resolving conflicts. One very basic, perhaps implausible, strategy would be to fight until victory or death. Increasingly complex physiologies and cognitive abilities can support increasingly sophisticated conflict resolution strategies such as, yielding, adherence to pecking order, and coalition building. The more environmental, social and historical elements an animal can utilize in a strategy, the more successful it can be.

For clarity, and in order to be in-line with the terminology used in biology, we use the term behavior to denote only externally observable displays. We call the solution to a problem provided by certain behaviors a behavioral strategy or just strategy. We use the term behavioral layer or layer to denote the internal structures of a robot involved in supporting a specific strategy.
Hypothetical Evolutionary Paths (HEP) is a development methodology inspired by evolution, that tries to deliver on the promise of incremental development while also allowing a developer to leverage functionality already implemented in the underlying implementations. HEP suggests the design of a set of concurrent, increasingly sophisticated, problem solving strategies that can be implemented incrementally. By recognizing key elements or schemas [1] in the different behavioral layers, HEP minimizes the size of any increments and maximizes the leverage given by underlying solutions. These schemas represent salient features of the robot’s inputs, and are made available to other layers as a way of biasing their related learning problems. The HEP methodology has two main strengths. First, the adaptation is highly restricted. Highly restricted adaptation allows the algorithms to adapt quickly after minimal amounts of experience. Second, HEP produces algorithms that employ multiple strategies of different sophistication concurrently to solve a given problem. During learning, under stress, i.e., when dealing with urgent problems, and in the case of limited hardware failures in distributed systems, such algorithms retain a higher level of performance than algorithms that use centralized hypotheses. The main weakness of HEP is that the algorithms produced are highly specialized compared to existing learning algorithms.

2 Schema Classes

Schema theory [1] represents cognitive structures distributed, communicating, concurrently executing nodes or schemas. We have implemented a library, plancs, for object oriented programming that provides a set of classes called programmable schemas (PSs). A PS implements the major schema properties of the abstract schema language [13]: delegation, wrapping, heterogeneity, encapsulation, and reusability, and also support subsumption style communication.

To support HEP, we used a modular model of behavioral layers based on a number of schema classes. The reason for using this model was to facilitate inter-layer sharing of highly abstracted perceptory and motory skills. A typical configuration of the different schema classes within a layer is presented in Figure 1. Arrows indicate communication channels. Inhibitions are indicated by lines with a circle in one end.

Figure 1: The Programmable Schema Classes

Sensor schemas and actuator schemas encapsulate low-level application programming interfaces (APIs). Sense schemas and competence schemas are high-level sensor and actuator abstractions respectively. Memory schemas are used to store data and last, drive schemas handle the logic of the interactions between multiple senses and competences within a layer and interact with other layers through inhibition.
3 Hypothetical Evolutionary Paths

An evolutionary origin for a behavioral strategy implies that it should be possible to construct a tree of supporting strategies linking the top-level or root strategy to a set of minimally adaptive starting points or leaf strategies. We call such a tree a strategy tree. We call each level of the strategy tree a generation, with the leaf nodes forming the first generation. We have extracted the main elements of the construction of such hypothetical evolutionary paths into a set of rules for developing adaptive robot algorithms:

1. **Complete Path Specification:**
   i. Identify perceptory and motory skill requirements.
   ii. Identify supporting strategies, delegate skills and refine requirements.
   iii. Identify the inhibition structure and shared schemas.
   iv. Specify a fitness metric that captures the evolutionary advantage and define an environment in which the advantage is expressed.

2. **Stepwise Path Verification:**
   i. Test the feasibility of the design through implementation.
   ii. Test the viability of the strategies through experimental comparison of fitness.
   iii. Revise Path.

**Complete Path Specification**  HEP starts with a strategy tree containing only the envisioned top-level strategy which solves the problem the robot is facing. For each strategy in the tree that is not directly implementable, HEP then identifies a set of supporting strategies. The supporting strategies are added to the strategy tree as branches from the supported strategy. Each supporting strategy should implement a complete solution to a problem or sub-problem facing the robot and in the process provide schemas that are useful for the supported strategy. The process of identifying supporting strategies is then repeated for each of the new strategies until all the leaf-strategies are directly implementable.

Shared schemas are schemas that are general enough to be useful for multiple layers, e.g., an opponent sense schema or an approach object competence schema. Some psychological theories emphasize how memories have an origin in an animals sensory or cognitive structures [4]. Based on this we limit the possible states for memory schemas by restricting their content to copies of sense or competence schemas already implemented in supporting layers.

Each step along the hypothetical evolutionary path should provide a new, superior solution to a problem relevant to the root strategy. In order to verify this improvement, it is necessary to define a measure of fitness that can objectively discriminate between the quality of two solutions.
Stepwise Path Verification Whereas the HEP design phase is mainly top-down, the test phase is strictly bottom-up. The strategy tree provides abstract descriptions of solutions that are not necessarily robust or efficient enough to work in situated systems. In order to avoid flights of fancy we suggest that each generation is fully implemented and tested for performance before any work is done on later generations.

To provide evidence for the evolutionary viability of the suggested strategies we suggest testing them against their predecessors. This kind of testing provides experimental evidence indicating whether a new strategy performs consistently better than its predecessor with respect to a pre-defined performance metric and in a pre-defined environment. If a new strategy cannot pass a viability test it is not likely to be a part of the evolutionary path that lead to the root strategy. From a developer’s viewpoint, testing for viability helps weeding out inefficient solutions early in the development process.

4  HEP for Foraging and Mapping

Foraging or locating and gathering resources, is a fundamental problem that all animals face. When the solution to a foraging problem includes remembering and revisiting the positions of the resources, the problem becomes one of mapping. We decided to study these two domains because there is a natural progression from one to the other, and because mapping is a domain where biologists have recognized multiple strategies in animals both through behavioral [3] and neurological evidence [9].

4.1  Path Specification for Mapping

We started with a general mapping strategy in mind, where a robot would explore an environment and be able to revisit previously observed food sources or feeders. For simplicity, we assumed an environment where all landmarks were globally visible. The metric for comparing the viability of the strategies was the feeding rate, i.e., the number of times the robot reached a feeder during a set time period. The application of the HEP rules as presented in Section 3 is indicated by their ascribed roman numerals.

Approach General Feeder Position (i) In order to follow this strategy we recognized that the robot needed to know its current location and the location of the observed feeders. It also had to know how to return to any given location. Finally, it needed to know its energy level in order to head for a feeder at the optimal time. (ii) The problem of returning to a known location was divided into two strategies: the approach general feeder position strategy, which turned the robot toward a known location, and the approach visible feeder position strategy, which approached a known location when it was visible. In order to orientate the robot toward a known food location, it was necessary to know the robot’s current position and pose. This localization skill was delegated to a layer supporting a structured exploring strategy. This strategy used the pose information to do efficient exploration. The ability to remember the feeder locations was delegated to the layer supporting the approach visible feeder approach strategy. The
skill left to be implemented by the top layer was the ability to turn toward given locations. (iii) The approach general feeder position strategy inhibited the underlying search strategies, i.e., the random walk strategy and the structured exploration strategy. It was inhibited by the underlying strategies for approaching feeders when found, i.e., the feeder approach element of a random walk strategy and the approach visible feeder position strategy. It was also inhibited by the obstacle avoidance element of the random walk strategy. The requirements analysis identified three key schemas to be provided by the supporting strategies, a low energy sense schema, a position sense schema and a feeder position memory schema. (iv) The ability to immediately orientate toward known food locations at optimal times rather than waiting for these locations to become visible should increasing the feeding rate. This advantage is not dependent on any environmental features other than those needed by the supporting strategies.

**Approach Visible Feeder Location** This strategy brought the robot back to a known feeder location whenever such a location was visible and the robot’s energy levels were below a given threshold. (i) To follow this strategy the robot needed to be able to remember and recognize feeder locations. This again meant that it needed to know its own location when it encountered feeders. Last, it needed to know how to approach visible things. A robot’s location was defined in terms of the visible landmarks. Hence, the information available in a position percept also indicated which locations were currently visible. (ii) The localization skill was already delegated to the structured exploration layer. The feeder recognition and approach object competence was delegated to the random walking layer that would also implement the feeder recognition and approach skills. The skills to be implemented by this layer was the ability to sense the current energy level and the ability to remember feeder positions. (iii) Like the approach general feeder position strategy, this strategy inhibited the underlying structured exploration strategy for searching. It was likewise inhibited by the feeder approach and obstacle avoidance elements of the random walk strategy. In addition to the position sense schema presented by the structured exploration layer, a full feeder sense schema and an object approach competence schema would be presented by the random walk layer. (iv) The ability to return to known feeders when energy got low rather than continuing exploration should increase the feeding rate. This advantage was not dependent on any environmental features other than those needed by the supporting strategies.

**Structured Exploration** The structured exploration strategy lead the robot around the given environment according to a pre-defined pattern. (i) We found that this strategy needed localization skills and also needed to know how to follow a set pattern of exploration. (ii) Approaching the landmarks that defined the exploration pattern could be handled by the object approach competence schema already delegated to the random walk layer. The skills left to be implemented by this layer was the localization and the ability to follow a set sequence of landmarks forming a pattern of exploration. (iii) Like the other search strategies, this strategy was inhibited by the feeder approach and obstacle avoidance elements of the random walk strategy. No new shared schemas were identified. (iv) A structured search is a more efficient way of exploring an environment that a random walk, and the ability to do this should increase the feeding rate.
This strategy needed landmarks to be present in the environment so that a pattern of exploration could be defined.

**Random Wandering**  The random wandering strategy made the robot go forward until an obstacle was encountered and then made it turn around. The sensor and actuator noise made this strategy relatively unpredictable. This layer was further divided into four basic layers: move, approach full feeders, touch full feeders, and avoid obstacles, but for brevity we here present these layers as one.

### 4.2 Step Verification for Mapping

Our implementations used a Khepera robot with six infrared (IR) proximity sensors and a K6300 color camera, simulated by the Webots robot simulator. We used a square 1.5m by 1.5m simulated environment. The environment contained a feeder, two fixed obstacles and a Khepera robot. Each corner was marked with a unique color and functioned as a landmark. The complete design for the final solution is presented in Figure 2. Shared schemas are outlined in gray.

![Figure 2: The Behavioral Layers of Foraging and Mapping](image)

We ran 20 trials with each of the controllers and measured the average feeding rate over the first two minutes. At the start of each experiment the robot had an energy level of 1.0. Each 64 ms the energy level decreased by 0.02. When the robot was in contact...
with a feeder, the energy level rose to 2.0. If the energy level fell to zero, the robot was considered dead and immediately removed from the environment. The performance of all the different strategies are presented graphically in Figure 3 with the standard deviation indicated by the error bars for each data point.

![Figure 3: Foraging and Mapping Strategy Performances](image)

On top of the architecture presented here, we have later developed a set of strategies for conflict resolution with the following strategies in order of increasing sophistication: *uncritical fighting*, *reactive yielding*, *pecking order adherence* and *stylized hierarchy formation*. The performance criteria was survival time and our experiments showed an increase in performance comparable to the one presented here for foraging and mapping.

## 5 Related Work

Learning from feedback is an alternative method of producing complex intelligence in BB systems. Learning has been applied in this way to sets of pre-specified behaviors [5], and to combining pre-programmed behaviors [6, 10]. Self-organizing systems have also been implemented on top of BB systems as a way of producing complex intelligence [7, 12]. There is a growing number of successful real-time learning behaviors for situated systems. These systems, however, generally have either highly specialized adaptive mechanisms or they handle data that are highly specialized for the given problem spaces. Our aim for HEP is to develop it into a generally applicable methodology for developing specialized adaptive strategies with the same range of sophistication that is currently found in animals.

The approach to robotics that is closest to ours in that it describes a methodology is evolutionary robotics (ER) [8]. While ER proceeds by reproducing the *process* of evolution, HEP proceeds by reproducing the *results* of evolution, and in order to do so, takes inspiration from the process. We see HEP as a pragmatic alternative to ER, important in order to explore solutions that cannot currently be produced through ER methods.
6 Conclusions and Future Work

The viability tests showed that the algorithms produced using the HEP methodology adapt quickly, with a minimum of experience. This shows that the HEP methodology can produce complex yet efficient adaptive algorithms.

Today, ML is typically applied in unknown or highly-complex domains with dynamics that are difficult or impossible to model. HEP algorithms cannot, due to lack of generality, be used for this kind of automated system optimization. Humans and animals have a mixture of evolved and learned structures for behavioral control [11]. Our long-term aim is to apply constructivist neural networks on top of our pre-programmed neural schemas in order to identify a good balance between these two development processes and produce algorithms that are applicable also in such domains.

References