Generalization in an Autonomous Agent

Zs. Kalmár, Cs. Szepesvári and A. Lörincz

***Abstract*— In this article we present an extension of a previously defined model [8]. This model was introduced to govern an agent in a goal-programmed fashion in a previously unknown environment. The extension allows generalization in the input space, which reduces the memory requierements as well as the time requierements of the algorithm.**

I. Introduction

Tremendous information coming steadily from a changing environment requires the selection of important features. Restricting examination to these features saves time and is more efficient in storage. The basis of generalization is this selection. The formulation based on building neural network for the representation of events and chains of events [8] makes possible to formulate generalization as subgoal generation.

II. Architecture

Our starting point, or recipe, in the design of self-organizing autonomous agents is to take self-organizing artificial neural networks (ANN) for sizing down the tremendous amount of sensory input information from the external world to a set of 'high' or 'low' bits, and then to build up an internal representation from events and chains of events experienced during the lifetime of the agent. The agent has the following subsystems: (i) the dimension reducing ANN system, (ii) an operator, or action system, (iii) a subsytem that builds up a dynamic internal representation of sensory events, (iv) a prewired goal system, (v) the algorithm that designs action plans, and (vi) a decision system (see Fig. 1).

• The dimension reducing system (DRS)

Systems considered here are categorizing systems of soft competition, such as the 'where system' [6, 3] with spatial filters that reduce the high resolution image to a few positions and we can assume a grid system. The recognition of objects may be performed by several methods, the 'what system' [1, 2, 4, 3]. Both subsystem have 'high' and 'low' outputs and the agent should find out the meaning of these bits.

Cs. Szepesvári and A. Lörincz are with the Department of Photophysics, Institute of Isotopes, The Hungarian Academy of Sciences, Budapest, Hungary 1525. Cs. Szepesvári and Zs. Kalmár are with the Department of Mathematics, Attila József University of Szeged, Szeged, Hungary 6720

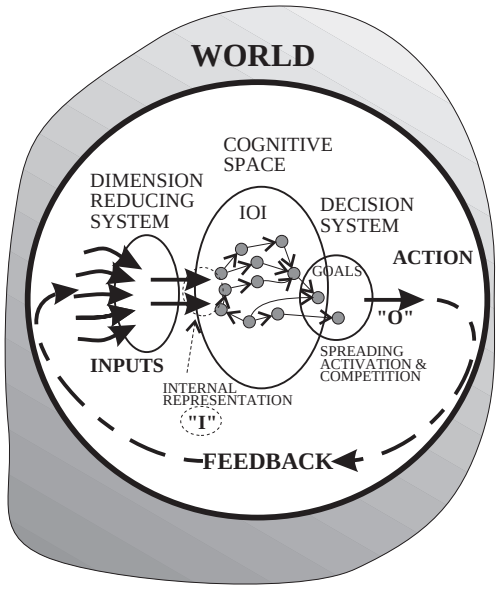

Figure 1: Architecture of the agent

• Operator, or action system

The system is equipped with a set of inherited operators, that allow to make macroscopic changes in the environment. It is assumed that these operators fit the grid of the 'where system'. At this point the model is not general. Generality could be improved if the starting point for the operator system were the self-learned differential operator set described in [7], and if the algorithm of [5] were used to develop useful macroscopic operators in a self-organized way.

• Internal representation of events (the cognitive space or CS)

The system collects 'events' of the world. An event is composed of a given internal representation of the sensory inputs (I), the operator (O) applied in state I, and state I that the system reached after executing operator O. The collection of these three entities shall be called an IOI triplet. During the course of life the agent experiences chains of triplets: one

event is followed by another. We collect these in the form of a directed graph where nodes are the events, and arrows correspond to the flow of time. This representation reflects that operators are not mappings on the I world but relations, since both the sensory inputs and the DRS may loose information; the system is a doubly weighted directed graph (or directed neural network (DNN)), where weight of node I_1OI_2 is upgraded to tell how reliably takes operator O from initial state I_1 to final state I_2. Weight of directed arrows between triplets $I_aO_1I_b$ and $I_cO_2I_d$ should reflect how often is operator O_1 is followed by operator O_2 [8].

• The goal system

The agent is equipped with a prewired goal subsystem. A 'goal' can be either 'high' or 'low'. If 'high' then the agent tries to set it 'low'. When it becomes 'low' then the event satisfies the goal and that is remembered.

• Designing action plans and decision making

When a goal becomes high then activation is spread from triplets, that had satisfied that goal, opposite to the arrows of the directed graph. Spreaded activation is monitored in nodes that have initial states identical to the present output of the DRS. The highest activity node is chosen and the activity is compared to a threshold. If the spreaded activity is below threshold then a move is chosen randomly. Otherwise the operator of the node is executed, the new I state is experienced, and CS is upgraded.

The system may have inherited ANN subsystems. Inherited ANN may be connected to inherited knowledge based system (KBS). The prewired macroscopic operators and the goals of the system may be thought of as KB systems too.

III. Generalizing ability: from relations to mappings

It was noted that operators are just relations, not mappings on the space of the I states: Thus the result of an action is in general ambiguous. It is then attractive to transform the space of the I states to make operators mappings on the transformed space. If operators are mappings and the transformation keeps the learnt information to reach the goals then we say that 'all the important features are kept'. Important informations then may be viewed as *'emerging concepts'* and the transformation as generalization.

Our working hypothesis about the world and the agent is that 'emerging concepts' are *'quasi independent'*, i.e. the effect of operators may be estimated by considering just a few concepts in a given situation, or I state. A concept $\mathcal{C}$ may be given as a subset of $\mathcal{I}$, the space of all possible I-states. Our I states may be given as elements of the $\{0,1\}^n$ set. Here, we restrict the set of possible concepts to Holland's Universe of $\mathcal{I}$, i.e. to $\{0,1,*\}^n$, where string $''01*''$ e.g. denotes the two element set $\{''010'',''011''\}$.

IV. Concept generation

Concepts are important features to reach the goal and are thus formed from the direction of the goal. Creating an important feature to reach the goal may be viewed as the creation of a subgoal. I.e. goals are the zeroth level concepts. A heuristic algorithm may be given as follows: First, an IOI' triplet that reaches the goal is viewed as $IO\mathcal{C}_g$. Union of concepts (and thus I-states too) is constructed by saving identical bits and replacing different bits by $''*''$.

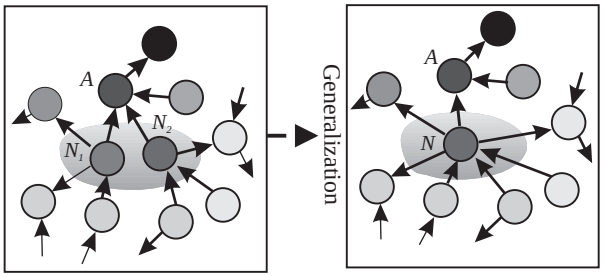

Figure 2: Generation of a new concept

Two nodes N_1 and N_2, which may be used for the generation of a new concept, their common ancestor A and the generated new concept N are shown in the figure. If $A = \mathcal{C}_1^A O^A \mathcal{C}_2^A$ and $N_i = \mathcal{C}_1^{N_i} O \mathcal{C}_2^{N_i}$, then $N = (\mathcal{C}_1^{N_1} \cup \mathcal{C}_1^{N_2}) O \mathcal{C}_1^A$.

Now, let us take two nodes of the DNN that satisfy the following conditions: (i) they have the same operator, (ii) they have the same ancestor, (iii) the ancestor is a generalized triplet of $\mathcal{C}O\mathcal{C}'$ form, where $\mathcal{C}$ is a non-trivial concept, and (iv) the ancestor is reliable (see Fig. 2). In this case a new node N is added to the DNN subject to all refreshing procedures given in [8]. New connection set being the union of the connection sets of the old nodes is generated to N. N has the joint operator of the old nodes. Initial concept of N is the union of the respective initial concepts of the old nodes. Final concept of N is the initial concept of the ancestor. To be able to start the algorithm nodes that satisfy the same goal are thought of as having a common ancestor that satisfies conditions (iii) and (iv).

V. Results

Performance difference between algorithms with (WG) and without generalization (WOG) is striking. Problems that were easily solvable for the WG algorithm led to an explosion in either memory requirement

or learning time for the WOG algorithm. Operators of the system were: 'up', 'down', 'left' 'right', and 'suck'. Hitting the food or any of the obstacles resulted in pain and the goal 'remove pain' was activated. Time-to-time the agent found the goal 'eat' high. To fulfil that goal the agent had to stop in front of a food and had to use operator 'suck'. The agent was pretrained in a space of a few obstacles and a few food. After pretraining the agent was put in a dense space with many obstacles. The WG agent could fit the emerged concepts to the new world and 'survived'. The WOG agent had an enormous network learnt for special rare space cases, unuseful for the new situation and was unable to survive. In order to make a useful comparison we had to simplify the problem to a middle sized grid of dimensions 9×14 and detected area of 3×3. In a series experiments with medium density of food and obstacles the two agents were allowed to learn and information densities of their respective memories were measured. The memory information density (MID) was taken as proportional to the performance and inversely proportional to the size of the DNN. For this small problem the MID ratio was already an order of magnitude in favor of the WG agent.

VI. Conclusions

Agent with generalization capabilities show striking improvement in performance. To make generalization one needs the information saved by the DNN, such as the representation given in the form of events, and the directed, doubly weighted property of the graph.

References

[1] G. Carpenter and S. Grossberg. Massively parallel architecture for self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, 37:54–115, 1987.

[2] P. Földiák. Learning invariance from transformation sequences. *Neural Computation*, 3(2):194–200, 1991.

[3] T. Fomin and A. Lörincz. Positioning and feature extraction with Hebbian and anti-Hebbian networks. *Neural Networks*, 1993. submitted.

[4] K. Fukushima. Character recognition with neural networks. *Neural Computing*, 4:221–233, 1992.

[5] S. Kovács, G. Tóth, C. Szepesvári, and A. Lörincz. Schematic algorithm. in preparation.

[6] C. Szepesvári, L. Balázs, and A. Lörincz. Topology learning solved by extended objects: a neural network model. *Neural Computation*, 1993. in press.

[7] C. Szepesvári, T. Fomin, and A. Lörincz. Self-organizing neurocontrol. In *ICANN'94: Int. Conf. on Artificial Neural Networks*, 1994. submitted.

[8] C. Szepesvári and A. Lörincz. Behavior of adaptive self-organizing autonomous agent working with cues and competing concepts. *Adaptive Behavior*, 1994. in press.