WCCI 2012 IEEE World Congress on Computational Intelligence
June, 10-15, 2012 - Brisbane, Australia IEEE CEC

Evolving the way of doing the right thing

Oscar J. Romero Lopez
Decoroso Crespo Lab
Universidad Politecnica de Madrid
Madrid, Spain
Email: ojrlopez@hotmail.com

Angelica de Antonio
Decoroso Crespo Lab
Universidad Politecnica de Madrid
Madrid, Spain
Email: angelica@fi.upm.es

Abstract—**An attractive question which still remains on Intelligent Systems reseach field is how can we build autonomous agents whose internal cognition process can be self-configured over time? Our paper proposes a self-organized model for decision making, which is a robust evolutionary extension of typical Behaviors Network model. Given an initial set of meaningless and unconnected units (behaviors), our system is able to evolutionarily build well-defined and robust behavior networks which are adapted and specialized to concrete internal agent's needs and goals. As a result, several properties of self-organization and adaptability emerged when the proposed model was tested in a robotic environment, using a multi-agent platform.**

I. Introduction

An autonomous agent is a self-contained program which is able to control its own decision making process, sensing and acting autonomously in its environment, and by doing so realizes a set of goals or tasks for which it is designed. These goals and tasks use to change dynamically through time as a consequence of internal and external (environmental) perturbations, and ideally, the agent should adapt its own behavior to these perturbations. Maes [1]–[3] proposed a model for building such an autonomous agent, which includes a mechanism for action selection (MASM) in dynamic and unpredictable domains based on so-called behavior networks. This model specifies how the overall problem can be decomposed into subproblems, i.e. how the construction of the agent can be decomposed into the construction of a set of component modules (behaviors) and how these modules should be made to interact. The total set of modules and their interactions provides an answer to the question of how the sensor data and the current internal state of the agent determine the actions (effector outputs) and future internal state of the agent through an activation spreading mechanism that determines the best behavior to be activated at each situation.

One of the most relevant weakness of original Maes model is that the whole network model is fixed, so it requires to be pre-programmed both a network structure (e.g., modules, spreading activation links, etc.) and global parameters that define the characteristics of a particular application (e.g., goal-orientedness vs. situation-orientedness, etc.), and hence that the agent has no complete autonomy over its own decision making process. As an intent of resolving this problem, in [4] are proposed two mechanisms depending on real world observations: a learning mechanism for adding/deleting links in the network, and an introspection mechanism for tuning global parameters. The main problem with the former is that it does not use a real machine learning algorithm, but rather a simple statistical process based on observations, so a lot of hand-coded instructions are still required. In respect to the latter, it proposes a meta-network (another behavior network) which controls the global parameter variation of first network through time, but the problem still remains: who is in charge of dynamically adapting the global parameters of the meta-network? it seems to be similar to what is well known in cognitive psychology as the homunculus problem, or as the russian nested dolls (matryoska dolls) effect in colloquial terms.

The present work proposes a novel model based on Gene Expression Programming (GEP) [5] that allows the agent to self-configure both the topological structure and the functional characterization of each behavior network without losing the required expressiveness level. This approach confers a high level of adaptability and flexibility, and always produces, as a consequence, syntactically and semantically valid behavior networks.

The remainder of this paper is organized as follows. Section II describes roughly the operation of the behavior network model. Section III explains in detail how the behavior network model is extended using GEP. Section IV outlines and discusses the results of the experiments. The concluding remarks are shown in Section V.

II. Behavior Networks Model

In the following, we describe the behavior network formalism. Since we do not need the full details for our purposes, the description will be sketchy and informal at some points.

A Behavior Network (BN) is a mechanism proposed by Maes [1] as a collection of competence modules which works in a continuous domains. Action selection is modeled as an emergent property of an activation/inhibition dynamics among these modules. A behavior i can be described by a tuple $\langle c_i, a_i, d_i, \alpha_i \rangle$. c_i is a list of preconditions which have to be fulfilled before the behavior can become active, and $e = \tau(c_i, s)$ the executability of the behavior in situation s where $\tau(c_i, s)$ is the truth value of the precondition in situation s. a_i and d_i represent the expected (positive and negative) effects of the behavior's action in terms of an *add list* and a *delete list*. Additionally, each behavior has a level of activation α_i. If the proposition X about environment is true and X is in the precondition list of the behavior A, there is an active link from the state X to the action A. If the goal Y has an activation greater than zero and Y is in the add list of the behavior A, there is an active link from the goal Y to the action A.

Internal links include predecessor links, successor links, and conflicter links. There is a successor link from behavior A to behavior B (A has B as successor) for every proposition p that is member of the add list of A and also member of the precondition list of B (so more than one successor link between two competence modules may exist). A predecessor link from module B to module A (B has A as predecessor) exists for every successor link from A to B. There is a conflicter link from module A to module B (B conflicts with A) for every proposition p that is a member of the delete list of B and a member of the precondition list of A. The following is the procedure to select an action to be executed at each step:

1) Calculate the excitation coming in from the environment and the goals.
2) Spread excitation along the predecessor, successor, and conflicter links, and normalize the behavior activations so that the average activation becomes equal to the constant π.
3) Check any executable behaviors, choose the one with the highest activation, execute it, and finish. A behavior is executable if all the preconditions are true and if its activation is greater than the global threshold. If no behavior is executable, reduce the global threshold and repeat the cycle.

Additionally, the model defines five global parameters that can be used to "tune" the spreading activation dynamics of the BN and thereby, they affect the operation of the behavior network:

1) π: the mean level of activation
2) θ: the threshold for becoming active. θ is lowered with 10% each time none of the modules could be selected. It is reset to its initial value when a module could be selected.
3) ϕ: the amount of activation energy a proposition that is observed to be true injects into the network.
4) γ: the amount of activation energy a goal injects into the network.
5) δ: the amount of activation energy a protected goal takes away from the network.

In Maes' BN model, all of the internal links and all of the global parameters as well must be tuned by hand. In the following section, we describe a mechanism to evolve the BN topology in order to adapt it to continuous changing goals and states of the environment.

III. Evolutionary Behavior Networks

We propose an extended version of Maes model described above, which incorporates an evolutionary mechanism addressed by Gene Expression Programming (GEP) [5] in charge of evolving the BN topology, in other words, the

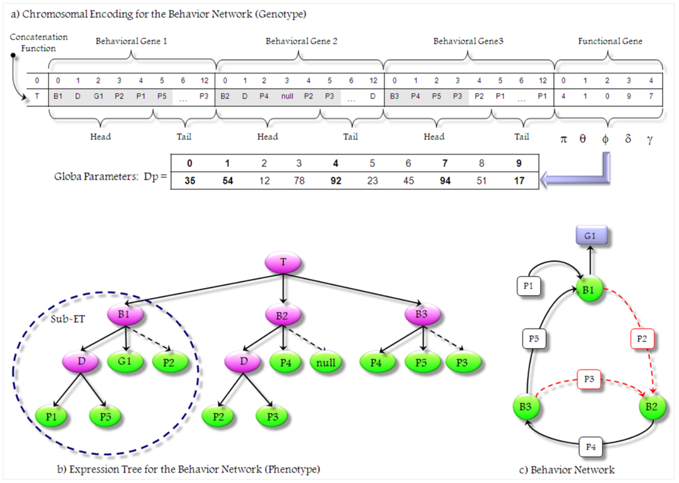


Fig. 1: GEP translation of a Behavior Network

activation/inhibition links among behaviors, the preconditions of each behavior, and the algorithm's global parameters. This section explains how the chromosomes of GEP can be modified so that a complete BN, including the architecture, the activation/inhibition links, and the global parameters, could be totally encoded by a linear chromosome, though may be expressed as non-linear structures such as expression trees. The main advantage of using GEP is that no matter how much or how profoundly the chromosomes are modified, the algorithm always guarantees the production of valid solutions.

A. *Genetic encoding of Behavior Networks*

The network architecture is encoded in the familiar structure of head and tail [6]. The head contains both special *functions* that activate the units and *terminals* that represent the input units. The tail contains obviously only *terminals*. Let us now analyze an example about how the BN is encoded into a GEP chromosome.

In Figure 1.a, a linear multigenic chromosome is initially generated in a random way and modified by genetic operators after that. Each multigenic chromosome defines several *Behavioral Genes* and just one *Functional Gene*. Each *Behavioral Gene* encodes a different behavior's structure whereas the *Functional Gene* encodes the global parameters of the BN. We propose a multigenic chromosomal structure, which is more appropriate to evolve good solutions to complex problems, because they permit the modular construction of complex hierarchical structures, where each gene encodes a smaller and simpler building block (a behavior). The details of the encoding process will be explained further along.

After that, the multigenic chromosome could be translated into the whole Expression Tree of Figure 1.b, using the conversion process described in [6]. Here, it is possible to identify three kind of functions: **B**, **D**, and **T**. The **B** function is used for representing each behavior of the net, and it has an arity of three: the first branch is a set of preconditions, the second one is a set of activation links that connects to other behaviors, and the third one is a set of inhibition links that connects to other behaviors (dashed arrows). The **D** and **T** functions are connectivity functions that join two or three elements, respectively, of the same nature (e.g., behaviors, preconditions, goals, etc.). It is important to notice that the Expression Tree is composed of several sub-expression trees (sub-ETs), where each one of these represents the structure of an unique behavior in the net, and hence that each sub-ET has a particular organization that is encoded into one separated *Behavioral Gene*, and the whole Expression Tree (ET) models the entire behavior network.

In Figure 1.c is depicted a basic BN with three behaviors *(B1, B2, and B3)*, where the solid arrows denote excitatory activation connections and the dashed arrows denote inhibition connections among behaviors. P_1, P_2, P_3, P_4 and P_5 denotes the preconditions for behaviors. In order to simplify the picture, each behavior only defines few preconditions. However, in the real implementation, the preconditions set for each behavior might be composed by: a set of sensory inputs (internal and external), a set of active working memory elements, a set of current sub-goals, and a set of motivational states (drives, moods and emotions). G_1 is an agent's global goal pursued by behavior B_1.

In the example of Figure 1.a, for each *Behavioral Gene*, positions from 0 to 3 encode the head domain (so both functions and terminals are allowed), and positions from 4 to 12 encode the tail domain (where only terminals are allowed). Due to each Behavior defines variable sets of preconditions, activation links, and inhibition links, the corresponding genetic encoding spans along regions of different sizes into the behavioral gene. These regions are called *Open Reading Frames* (ORF) [5]. In GEP, what changes is not the length of genes, but rather the length of the ORF. Indeed, the length of an ORF may be equal to or less than the length of the gene.

Each sub-ET can be generated straightforwardly from chromosomal representation as follows: first, the start of a gene corresponds to the root of the sub-ET, forming this node the first line; second, depending on the number of arguments to each element (functions may have a different number of arguments, whereas terminals have an arity of zero), in the next line are placed as many nodes as there are arguments to the functions in the previous line; third, from left to right, the nodes are filled, in the same order, with the elements of the gene; and fourth, the process is repeated until a line containing only terminals is formed.

Due to the process is "bidirectional, inversely each Behavioral Gene can be easily inferred from the corresponding sub-ET as follows: first, the behavior function (B) of the sub-ET is encoded, and after that, the algorithm makes a straightforward reading of the sub-ET from left to right and from top to bottom (exactly as one reads a page of text). For instance, sub-ET for the behavior B_1 is encoded as: **B1-D-G1-P2-P1-P5**, and this is the ORF of its corresponding Behavioral Gene (i.e., the shadowy region for this gene in Figure 1).

On the other hand, the *Functional Gene* encodes an additional domain called D_p, which represents the global parameters of the BN. For the *Functional Gene*, position 0 encodes π (the mean level of activation), position 1 encodes θ (the threshold for becoming active), position 2 encodes ϕ (the amount of energy for preconditions), position 3 encodes δ (the amount of energy for protected goals), and position 4 encodes γ (the amount of energy for goals). The values of global parameters are kept in an array and are retrieved as necessary. The number represented by each position in the parameters domain indicates the order in the array D_p. For example, position 0 in the Functional Gene (π) encapsulates the index "4" which corresponds to the value 92 in the D_p array (in bold), and so on. For simplicity, Figure 1 only shows an array of ten elements for parameters domain D_p, but in our implementation we are using an array of one hundred elements, where each position encodes one numeric value between 0 and 100. Genetic operators guarantee that global parameters are always generated inside the domain of D_p array.

B. Special Genetic Operators

The operators of the basic gene expression algorithm [5] are easily transposed to behavior-net encoding chromosomes, and all of them can be used as long as the boundaries of each domain are maintained so alphabets are not mixed up. Mutation was extended to all the domains so every different gene (behavioral or functional) is modified following its respective domain constraints (e.g., not replacing terminal nodes by function nodes in the tail region, etc.). Insertion Sequence (IS) and Root Insertion Sequence (RIS) transposition were also implemented in behavioral genes and their action is obviously restricted to heads and tails. In functional gene we define only an IS operator (because RIS operator is not applicable here) that works within D_p domain, ensuring the efficient circulation of global parameters in the population. Another special operator, the parameters' mutation, was also defined in order to directly introduce variation in the functional gene (i.e., global parameters region) selecting random values from D_p array.

The extension of recombination and gene transposition to GEP-nets is straightforward, as their actions never result in mixed domains or alphabets. However, for them to work efficiently (i.e., allow an efficient learning and adaptation), we must be careful in determining which behavior's structure elements and global parameters insert into which region after the splitting of the chromosomes, otherwise the system is incapable of evolving efficiently. Therefore, for our multigenic system, a special intragenic two-point recombination was used so that the recombination is restricted to a particular gene (instead of interchanging genetic material with other kind of genes in the chromosome).

In summary, in order to guarantee the generation of valid BNs, all genetic operators have to comply with the following constrains:

- In the first position of behavioral genes, it can only be inserted a *B* (Behavior) node.
- For head region in Behavioral Genes:

  - It just can be mutated by connectivity functions (D and T), and by terminals such as preconditions (P_n) and goals (G_n).
  - Transposition (IS and RIS) and one-point and two-point Recombination operators must follow the same syntactic validations than the mutation operator.
- For tail region in Behavioral Genes:
  - Terminals of this gene just can be mutated, transposed and recombined using elements from tail domain, such as preconditions (P_n) and goals (G_n). No syntactic validations are required.
- For global parameters in Functional Gene:
  - Terminals of this gene just can be mutated, transposed and recombined using numeric values from parameters domain D_p, that means, numeric values between 0 and 100. No additional syntactic validations are required.

C. Fitness Functions for BN-Chromosomes

In this section we describe how Behavior-Network chromosomes are evaluated, so they have more or less probability to be replicated in the next generation of the evolutionary process. For the fitness evaluation we have took into account the theorems proposed by [7], and additionally we have identified a set of necessary and sufficient conditions that make behavior networks goal converging. It is important to notice that all reinforcement parameters used in the next fitness functions are self-generated by the system from changes observed in agent's internal states, so that they don't require a priori encoding nor manual adjustment made by a designer.

First of all, we define two fitnesses functions: one evaluates how well-defined the behavior-network structure is, and the other one evaluates the efficiency and functionality of the behavior network. The fitness function for evaluating the behavior-network structure is:

$$FFS_i = A_i + B_i + C_i + D_i + E_i + F_i \tag{1}$$

Where i is a chromosome encoding a specific BN, and each term is:

a) A_i: is there at least one behavior of the net accomplishing with a goal?, such that:

$$A = \begin{cases} a1, & \text{if } \exists\, beh \in i \mid a_{beh} \cap G(t) \neq \emptyset \\ a2, & \text{otherwise} \end{cases} \tag{2}$$

where beh is any behavior, a_{beh} is the add list of beh, $G(t)$ is a set of global goals, $a1$ is a positive reinforcement (+100), and $a2$ is a negative reinforcement (-100).

b) B_i: are all behaviors of the net well-connected? such that:

$$B = n_{cp} \cdot b1 + n_{up} \cdot b2 \tag{3}$$

where n_{cp} is the number of behaviors correctly connected to others through successor and predecesor links (self inhibitory connections are incorrect). n_{up} is the number of unconnected behaviors (no propositions at *add list* neither at *delete list*). $b1$ is a positive reinforcement (+10) and $b2$ is a negative reinforcement (-20).

c) C_i: is there any deadlock loops defined by the BN?, such that:

$$C = \begin{cases} (n_p \times c1) + (n_{np} \times c2), & \text{if the BN has associated a global goal} \\ c3, & \text{otherwise} \end{cases} \tag{4}$$

where n_p is the number of behaviors that define at least one path connecting to the global goal, $c1$ is a positive reinforcement (+20), n_{np} is the number of behaviors without a path between them and the global goal, and $c2$ and $c3$ are negative reinforcements (-10 and -50).

d) D_i: are all propositions (preconditions, add list, and delete list) of each behavior unambiguous? (e.g., the precondition set is ambiguous if it has propositions p and $\neg p$ at the same time), such that:

$$D = \sum_{i=0}^{k} (n_{na} \times d1) + (n_a \times d2) \tag{5}$$

where k is the total number of behaviors, n_{na} is the number of unambiguous propositions, $d1$ is a positive reinforcement (+10), n_a is the number of ambiguous propositions, and $d2$ is a negative reinforcement (-20).

e) E_i: are all add-list propositions non-conflicting? (e.g., a proposition that appears both in the add list and in the delete list – for the same behavior – is a conflicting proposition), such that:

$$E = \sum_{i=0}^{k} (n_{nca} \times e1) + (n_{ca} \times e2) \tag{6}$$

where k is the total number of behaviors of the BN, n_{cna} is the number of non-conflicting add-list propositions, $e1$ is a positive reinforcement (+10), n_{ca} is the number of conflicting add-list propositions, and $e2$ is a negative reinforcement (-20).

f) F_i: are all delete-list propositions non-conflicting? such that:

$$F = \sum_{i=0}^{k} (n_{ncd} \times f1) + (n_{cd} \times f2) \tag{7}$$

where k is the total number of behaviors of the BN, n_{cnd} is the number of non-conflicting delete-list propositions, $f1$ is a positive reinforcement (+10), n_{cd} is the number of conflicting delete-list propositions, and $f2$ is a negative reinforcement (-20).

On the other hand, the fitness function for evaluating network functionality is:

$$FFE_i = G_i + H_i + I_i + J_i + L_i + M_i + N_i \tag{8}$$

Where i is a chromosome encoding a specific BN, and each term is defined as:

g) G_i: this term determines if γ (the amount of energy for goals) is a well-defined parameter. Due to the parameter γ must reflect the "goal-orientedness" feature of the BN, then:

$$G=\begin{cases} 100-g1, & \text{if } (R_{freqG}>0 \wedge R_\gamma>0) \vee (R_{freqG}<0 \wedge R_\gamma<0) \\ g2, & \text{otherwise} \end{cases} \tag{9}$$

where R_{freqG} is the absolute variation rate which determines how often a goal is activated by the internal agent's motivational sub-system. R_γ is the absolute variation rate for parameter γ. $g1$ is the absolute difference among the variation rates: $g1 = \mid R_{freqG} - R_\gamma \mid$; and $g2$ is a negative reinforcement (-100). Intuitively, when the frequency of activated goals increases through time, the global parameter γ should increase proportionally too.

h) H_i: this term determines if ϕ (the amount of energy for preconditions) is a well-defined parameter. Due to the parameter ϕ must reflect the "situation relevance" and "adaptivity" features of the BN, then:

$$H=\begin{cases} 100-h1, & \text{if } (R_{freqC}>0 \wedge R_\phi>0) \vee (R_{freqC}<0 \wedge R_\phi<0) \\ h2, & \text{otherwise} \end{cases} \tag{10}$$

where $h1$ is the absolute difference among the absolute variation rates: $h1 = \mid R_{freqC} - R_\phi \mid$. R_{freqC} is a variation rate (between a current and prior states) which determines how often the environmental perturbations are perceived by the agent. R_ϕ denotes the absolute variation rate for parameter ϕ. And $h2$ is a negative reinforcement (-100).

i) I_i: this term determines if π (the mean level of activation) is a well-defined parameter. Due to the parameter π must reflect the "adaptivity" and "bias to ongoing plans" features of the BN, then:

$$I=\begin{cases} 100-i1, & \text{if } (R_{freqSG}>0 \wedge R_\pi>0) \vee (R_{freqSG}<0 \wedge R_\pi<0) \\ i2, & \text{otherwise} \end{cases} \tag{11}$$

where $i1$ is the absolute difference among the absolute variation rates: $i1 = \mid R_{freqSG} - R_\pi \mid$. R_{freqSG} is a variation rate (between a current and prior states) which determines the activation frequency of the sub-goals set that are associated to a current global goal. R_π denotes the absolute variation rate for parameter π. $i2$ is a negative reinforcement (-100). Absolute variation rates are treated quite similar as in term G. Intuitively, if the environment requires the agent to address its actuation to the achievement of a hierarchical set of goals, the BN must increase the value of π through the time; otherwise, if the environment is quite dynamic and an adaptive behavior is required, the parameter π should decrease.

j) J_i: this term determines if δ (the amount of energy for protected goals) is a well-defined parameter. Due to the parameter δ must reflect the "avoiding goal conflicts" feature of the BN, then:

$$J=\begin{cases} 100-j1, & \text{if } (R_{auto}>0 \wedge R_\delta<0) \vee (R_{auto}<0 \wedge R_\delta>0) \\ j2, & \text{otherwise} \end{cases} \tag{12}$$

where $j1$ is the absolute difference among the absolute variation rates: $j1 = \mid R_{auto} - R_\delta \mid$. R_{auto} is the absolute variation rate for the number of self-referenced loops identified by the agent between current and prior states (e.g., when the system identifies a circular reference of behavior activation such as: $a \rightarrow b, b \rightarrow c, c \rightarrow a$). R_δ denotes the absolute variation rate for parameter δ. $j2$ is a negative reinforcement (-100). Intuitively, if R_{auto} increases, then R_δ should decrease proportionally, and vice versa.

k) L_i: this term determines if θ (the threshold for becoming active) is a well-defined parameter. Due to the parameter θ must reflect the "bias to ongoing plans", "deliberation", and "reactivity" features of the BN, then:

$$L=\begin{cases} 100-l1, & \text{if } (R_{freqCE}>0 \wedge R_\theta<0) \vee (R_{freqCE}<0 \wedge R_\theta>0) \\ l2, & \text{otherwise} \end{cases} \tag{13}$$

where $l1$ is the absolute difference among the absolute variation rates: $l1 = \mid R_{freqCE} - R_\theta \mid$. R_{freqCE} is the absolute variation rate for the number of changing environmental elements between current and prior states (e.g., novel objects that coming into the perception field, or perceived objects that change physically, etc.). R_θ denotes the absolute variation rate for parameter θ. $l2$ is a negative reinforcement (-100). Intuitively, if R_{freqCE} increases (that means, the environment is more dynamic), then R_θ should decrease proportionally (making the BN more reactive); but if R_{freqCE} decreases, then R_θ should increase in order to make the BN more deliberative.

l) M_i: this term validates the add-list efficiency of each behavior. If the current state includes a proposition that corresponds to any add-list's proposition of any behavior, the behavior will receive a positive reinforcement:

$$M = m1 \cdot \sum_{i=0}^{k} eev \tag{14}$$

where $m1$ is a positive reinforcement (+100). k is the number of propositions defined by the add-list of the activated behavior (a_{beh}), and eev is a function that determines if the expected effect is included into the current state ($S(t)$), in other words, it validates if the following condition is true: $\exists\, p \in S(t) \mid p \cap a_{beh} \neq \emptyset$.

m) N_i: this term validates the delete-list efficiency of each behavior. If the current state includes a proposition that corresponds to any delete-list's proposition of any behavior, the behavior will receive a negative reinforcement:

$$N = n1 \cdot \sum_{i=0}^{k} een \tag{15}$$

where $n1$ is a negative reinforcement (-200). k is the number of propositions defined by the delete-list of the activated behavior (d_{beh}), and een is a function that determines if the non-expected effect is not included into the current state ($S(t)$), in other words, it validates if the following condition is true: $\exists\, p \in S(t) \mid p \cap d_{beh} = \emptyset$.

Finally, the whole fitness for each BN is calculated as follows:

$$FFT_i = FFS_i + FFE_i \quad (16)$$

All the elements of the function exert different (and in some cases, opposing) evolutionary and selective pressures. On the one hand, we have defined a function element for each of the most typical structural problems identified in behavior networks (such as terminating and dead-end networks, monotone networks, non-converging acyclic networks, ambiguous and conflicting links, and so on). On the other hand, we have defined a function element for each kind of functional characterization of the behavior network (such as goal orientedness vs. situation orientedness, bias towards ongoing plans vs. adaptivity, deliberation vs. reactivity, and sensitivity to goal conflicts). So the whole fitness function try to model a multi-objective problem where probably the best suited solution should be met in an intermediate point.

IV. Experimentation

A simulated robotic environment was proposed in order to evaluate the proposed evolutionary decision-making model. In the simulated environment, the robot will have to collect different kind of objects and then deliver them in specific storages. The robot will have to coordinate different kind of tasks such as object search, object recognition, route planning, obstacle avoidance, battery recharging, object piling up, and so forth. The simulated robotic environment was designed using the Player/Stage platform[1]. Stage simulates a population of mobile robots moving in and sensing a two-dimensional bitmapped environment. In our simulation, the robot (agent) is provided with four kind of sensor interfaces: sonar sensors, gps sensor, laser sensors, and fiducial sensors; and two kind of actuator interfaces: position interface and gripper interface. In order to measure the convergence rates of different aspects of the evolutionary functional design of BNs, we propose various experimental cases where global parameters were continuously adapted to the situations: **Case 1**: this case measures the adaptation rates of goal-orientedness vs. situation-orientedness aspects of BNs. In this experiment, the robot senses one orange box and seven green boxes around it, where `<Collect-All-Orange-Boxes>` goal is the current goal. In spite of the fact that the robot receives more activation energy from situation (i.e., the seven green boxes), it must learn to pursue current goals and avoid changes of attention focus (e.g., it must focus on collecting orange boxes instead of collecting green boxes). See Figure 2.

Case 2: this case measures the adaptation rates of deliberation vs. reactivity aspects of BNs. Initially, the robot has to store an observed box in a specific storage and, for this, it has to gradually accumulate activation energy and activate sequentially a set of behaviors which accomplish the `<Store-Box>` goal (deliberation). Suddenly, during the task execution, an unexpected situation is presented to the robot: some obstacles are dynamically moved around, so the robot

[1]The open source platform is available on http://playerstage.sourceforge.net/

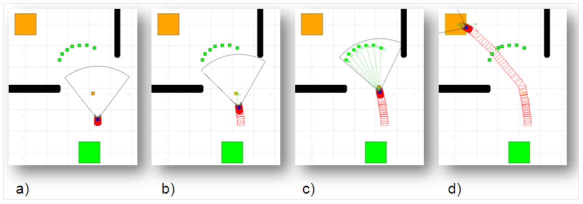


Fig. 2: Case 1. a) the robot senses an orange box. b) the robot activates `<Pick-up-orange-box>` behavior. c) the robot senses other seven "green" boxes. d) the robot does not change the attention focus even though it receives more activation from situation (the seven green boxes).

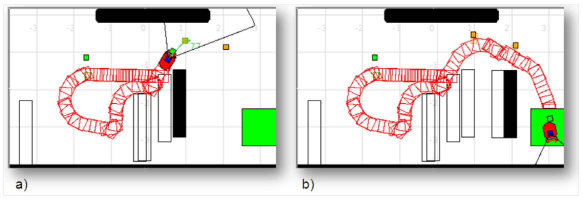


Fig. 3: Case 2. a) the robot reactively avoids a moving obstacle in front of it while it is carrying a box. b) the robot finally stores the box in spite of the multiple distracting obstacles.

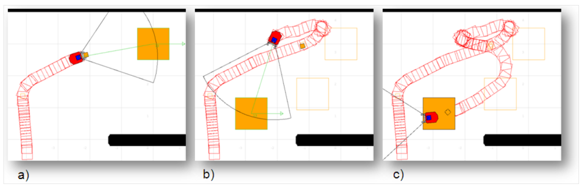


Fig. 4: Case 3. a) the robot is transporting a box until the storage. b) the location of the storage is moved, so the robot changes its initial plan (dropping the grasped box and starting to look for the new location of the storage). c) after the robot finds the new location of the storage it retakes the `<LOOK-FOR-BOX>` behavior and then stores the box.

will have to react on time with an evasive action (reactivity) and retake the control after that. See Figure 3.

Case 3: this case measures the adaptation rates of bias towards ongoing plans vs. adaptivity aspects of BNs. In this experiment the robot has to store a box in a storage situated in a specific point of the environment. For this, the robot has to previously make a plan in order to achieve the `<Store-Box>` goal. When the robot is getting closer to the storage, this latter will be displaced to another location, so the robot won't be able to store the box and it will have to start looking for the new location of the storage. The aim of this experiment is to validate the speed of the best evolved BN to replan a new problem-solving strategy on runtime. See Figure 4.

Case 4: this case measures the adaptation rate of sensitivity to goal conflicts aspect of BNs. In this experiment we take into account the anomalous situation example of the blocks world [8]. In this classical conflicting goals example there are three blocks (`A`, `B`, and `C`) which must be piled up in a specific order. The initial state of the world is `S(0) = (`

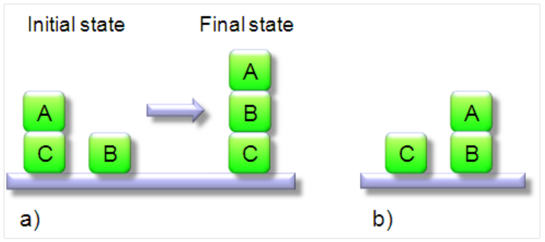
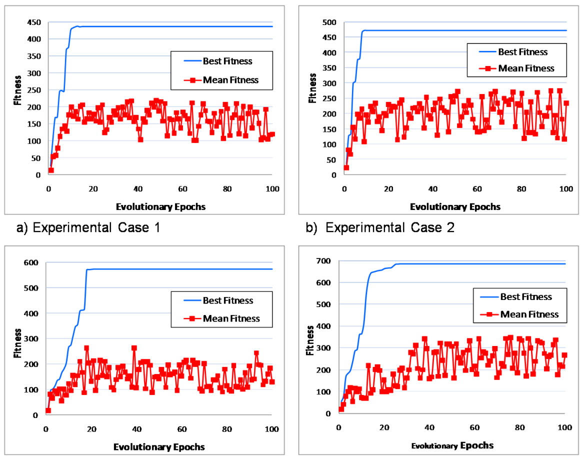


Fig. 5: Case 4. a) initial and final states of block world problem. b) deadlock situation that avoids to undo the already achieved goal `<A-on-B>`.



a) Experimental Case 1 b) Experimental Case 2 c) Experimental Case 3 d) Experimental Case 4

Fig. 6: Behavior of the GEP population best and mean fitnesses during the evolutionary epochs for cases 1, 2, 3, and 4.

`<clear-B>`, `<clear-A>`, `<A-on-C>`) and the goals are `G(0) = (<A-on-B>, <B-on-C>)`. The robot should first achieve the goal `<B-on-C>` and then the goal `<A-on-B>`. It is tempted however to immediately stack `A` onto `B` which may bring it in a deadlock situation (not wanting to undo the already achieved goal). Some of the behaviors used for this experiment were: `<stack-A-on-B>`, `<stack-B-on-C>`, and `<take-A-from-C>`. See Figure 5.

Figure 6 shows the convergence curves for the experimental cases 1, 2, 3 and 4 using the harmonic mean value for 100 runs. It is important to notice that the evolutionary process always converged for all cases, although not always at the same speed. For case 4 the convergence speed was slower because the GEP algorithm had to adjust all the global parameters, whereas in other cases were necessary to adjust only some parameters: π, γ, and ϕ for case 3; θ, γ, and ϕ for case 2; and γ and ϕ for case 1.

In table I are shown the statistical data from the experiments. The CE (convergence epoch) column indicates the mean epoch when the algorithm converged in every experiment. The MSE is the mean square error for 100 runs of the corresponding experiment, where an error is considered as a wrong behavior activation produced in every execution step by the behavior network (in the MSE_{gep} column is used the BN with the best fitness generated by the GEP algorithm

TABLE I: Statistics for BN functional evolution. CE: Convergence Epoch of the curve. MSE$_{gep}$: Mean Square Error of the case using the GEP algorithm. MSE$_{maes}$: MSE using only the original Maes' BN model. MSE rate is equivalent to $[1 - (MSE_{gep}/MSE_{maes})]$.

	Min	Max	Average	Std dev	CE	MSE$_{gep}$	MSE$_{maes}$	MSE rate
Case 1	27	437	416.10	70.90	15	45.78	118.03	61.21%
Case 2	35	472	455.44	69.10	9	41.39	98.36	57.91%
Case 3	28	573	515.18	137.73	21	56.15	145.87	61.50%
Case 4	53	683	629.01	143.98	27	73.02	233.28	68.69%

TABLE II: Evolved Global Parameters

	δ	γ	ϕ	π	θ
Case 1	85	68	23	92	95
Case 2	87	57	64	93	15
Case 3	88	37	61	18	93
Case 4	48	17	63	18	41

and in the MSE_{maes} column is used the original Maes' BN model without an evolutionary mechanism). The MSE rate presents the performance relationship between MSE_{gep} and MSE_{maes}. It is important to notice that, for the experiments executed, the proposed evolutionary BN model improves the performance results obtained by the original Maes' BN model a rate between 58% and 69%. This improvement is due to the capability of the proposed evolutionary BN model to self-adjust the global parameters on runtime, whereas the original Maes' BN model was always restricted to a fixed configuration of these parameters.

For all the experiments, original Maes' BN model was configured with the following fixed global parameters: $\delta = 90$, $\gamma = 50$, $\phi = 90$, $\pi = 90$, and $\theta = 100$. In contrast with these fixed values, table II shows the global parameters discovered by the proposed evolutionary mechanism for each experimental case. From these results, it is important to remark some observations:

Case 1: the proposed evolutionary mechanism discovered that in order to keep the balance between "goal-orientedness" and "situation-orientedness" aspects, γ must be roughly between 29% and 34% greater than ϕ. For a very high value of γ or a very low value of ϕ, the agent (robot) could not adaptively re-drive its attention focus towards more interesting goals when they were presented.

On the other hand, a value of ϕ greater than γ would avoid the robot to achieve none of the goals because it would be continuously changing its attention focus.

Case 2: the proposed evolutionary mechanism found out that in order to keep the balance between "deliberation" vs. "reactivity" aspects, the value of ϕ must be a little bit greater than γ (roughly between 9% and 15%). With this configuration the agent is not only able to keep its attention focused on current goals, but is also able to react against unexpected or dangerous situations. Additionally, when reactive behavior was required, the proposed evolutionary mechanism discovered that a low value of θ allows a fast behavior activation due to the BN takes less activation loops and, as a consequence, the amount of deliberative processing was considerably decreased.

Case 3: the proposed evolutionary mechanism revealed that in order to keep the balance between "bias towards ongoing plans" vs. "adaptivity" aspects, the value of π must be roughly between 46% and 52% greater than γ, and roughly between 72% and 77% lesser than ϕ. If these global parameters are kept between such ranges, the agent will not continually be "jumping" from goal to goal and in turn it will be able to adapt to changing situations. From the wrong solutions it is possible to infer that a value of π too much greater than γ and ϕ makes the BN more adaptive although less biased towards ongoing plans, wherewith the agent will continually be changing the current goals without keeping the focus on none of them.

Case 4: the proposed evolutionary mechanism discovered that in order to preserve the "sensitivity to goal conflicts", the value of δ must be roughly between 51% and 65% greater than γ. For a rate lesser than 50%, the BN does not take away enough activation energy from conflictive goals, whereas a value of γ grater than δ causes the BN to go in deadlocks due to the inability of the BN to undo already achieved goals. Furthermore, the evolutionary mechanism found out that the value of ϕ must be roughly between 62% and 73% greater than δ, otherwise the BN will not be able to activate the behavior sequence that resolves the goal conflict (i.e., `<take-A-from-C>`, then `<stack-B-on-C>`, and then `<stack-A-on-B>`) because of it will not receive enough activation from observed state. Finally, the evolutionary mechanism revealed that in goal-conflicting situations the value of θ must be lesser than δ, otherwise the BN will be more deliberative, and therefore, it will execute more activation loops during which the behaviors that promise to directly achieve a goal (e.g., `<stack-B-on-C>` and `<stack-A-on-B>`) will accumulate more activation energy than those behaviors that solve conflictive goals through sub-goaling (e.g., `<take-A-from-C>`).

From the obtained results of the above experiments, it is evident that the global parameters setting is a "multi-objective" problem, wherewith a single BN solution can not define a proper setting for all the proposed experimental cases. So, the solution to this problem requires multiple BNs competing among them in order to survive into the population, where only the best BN will be activated according to the situation observed by the agent.

V. Conclusion

In the present paper we have described a hybrid decision-making approach for autonomous agents which is supported by the robustness of both Behavior Networks model and Gene Expression Programming. The proposed model is able to adaptively build complex decision-making structures as a result of interacting evolutionary dynamics.

Specifically, the proposed evolutionary model focuses on the on-line *"development"* of Behavior Networks (BN) as task-oriented decision-making structures. BN have been proposed in previous works [1], [9], [10] as "control mechanisms for selective focusing of attention", and we tried to follow the same philosophy in our approach. However, we argue that this kind of decision-making structures must have the capacity of being adaptables and flexibles throughout the agent life cycle, instead of being pre-wired, hand-coded, and fixed structures. Thus, we proposed an evolutionary mechanism based on GEP which evolves the BNs depending on both the dynamic environmental interactions experienced by the agent and the internal changing states of itself. From the experimentation, it is possible to infer that the evolutionary approach was suitably able to evolve different kinds of BNs. This diversity can be seen from both a structural perspective (e.g., diverse net topologies determined by varying relationships of competition and cooperation among behaviors), and a functional perspective (e.g., different global parameters configurations determine diverse features of BN such as goal orientedness vs. situation relevance, adaptivity vs. bias to ongoing plans, deliberation vs. reactivity, sensitivity to goal conflicts, etc.).

In our approach it is possible to identify two levels of planning: (1) a short-term planning carried out by the Anticipatory Classifier System and Behaviors, which predicts the outcomes of action executed in the prior state, so the agent can opportunely react; and (2) a long-term planning driven by spreading and accumulation of energy dynamics of BNs, through which a set of expectations, goals and sub-goals are pursued. All these levels allow the agent to exhibit a deliberative behavior focused on goals achievement, although with the ability to reactively replan and change the course of action when (internal and external) perturbations from state require it. Therefore, it is the own system who is in charge of producing its own plans, mentally execute them (through internal simulation), check against its real world execution, and what it is more important, evolve them insofar as time passed by and feedback is received.

References

[1] P. Maes, "How to do the right thing," *Connection Science Journal*, vol. 1, pp. 291–323, 1989.

[2] ——, "Situated agents can have goals," *Robotics and Autonomous Systems*, vol. 6, no. 1, pp. 49–70, 1990.

[3] ——, "Learning behavior networks from experience," in *Proceedings of the First European Conference on Artificial Life*, 1992, pp. 48–57.

[4] ——, "The agent network architecture (ana)," *SIGART Bull.*, vol. 2, pp. 115–120, 1991.

[5] C. Ferreira, "Gene expression programming: A new adaptive algorithm for solving problems, complex systems," *Cognitive Science*, vol. 13, pp. 87–129, 2001.

[6] R. Poli, W. Langdon, and N. McPhee, *A Field Guide to Genetic Programming*. Massachusetts, USA: Standford University, 2008.

[7] B. Nebel and Y. Babovich-Lierler, "When are behaviour networks well-behaved?" in *Proceedings of ECAI*, 2004, pp. 672–676.

[8] G. Sussman, "A computer model of skill acquisition," Ph.D. dissertation, Elsevier Science Inc., 1975.

[9] S. Franklin, *The LIDA architecture: Adding new modes of learning to an intelligent, autonomous, software agent*. Cambridge: In Proc. of the Int. Conf. on Integrated Design and Process Technology, 2006.

[10] M. P. Shanahan and B. Baars, "Applying global workspace theory to the frame problem," *Cognition*, vol. 98, no. 2, pp. 157–176, 2005.