Activity-based Credit Assignment (ACA) Heuristic for Simulation-based Stochastic Search in a Hierarchical Model-base of Systems

Alexandre Muzy, Bernard P. Zeigler

Activity-based Credit Assignment (ACA) Heuristic for Simulation-based Stochastic Search in a Hierarchical Model-base of Systems

Alexandre Muzy*, Bernard P. Zeigler‡

*I3S UMR CNRS 7271, Bio-info team, CS 40121 - 06903 Sophia-Antipolis Cedex, France, Email: alexandre.muzy@cnrs.fr.

‡RTSync Corp. and Arizona Center for Integrative Modeling and Simulation, AZ, United-States of America, Email: zeigler@rtsync.com.

***Abstract*—Synthesis of systems constitutes a vast class of problems. Although machine learning techniques operate at the functional level, little attention has been paid to system synthesis using a hierarchical model-base. This paper develops an original approach for automatically rating component systems and composing them according to the experimental frames in which they are placed. Components are assigned credit by correlating measures of their participation (activity) in simulation runs with run outcomes. These ratings are employed to bias component selection in subsequent compositions.**

I. Introduction

In general system theory, theoretical frameworks have been defined by Klir through *systems problem solving* [1] and by Wymore through *system design and engineering* [2]. In Electronics, *system synthesis* [3] is often used from behavioral to structural domains. Specification is given as a set of subsystems finally implemented as digital circuits at hardware level. Broadly, the whole process can be embedded in a three-step implementation of an *intelligent algorithm* (as described in [4]): (i) Problem solving specification (of the objective model), (ii) Generated configurations (of system architectures), and (iii) Optimization execution (at simulation level[1]). More than specifying the parameter space of intelligent algorithms, dealing explicitly with systems requires re-investigating their structure and providing a new generic simulation-based search algorithm operating in the set of subsystems. This is the main objective of this article.

In supervised learning [11], considering a set of input-output pairs: $\{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, an "intelligent" algorithm determines, in a set of functions G, a function $g : X \rightarrow Y$, where X is the input set, and Y is the output set. However, when studying a complex system, a modeler has to alternate between both behavior (input-output function) and structure (states and interactions between system's/network's components) to specify its dynamics. According to the state of the system, for a particular input, some components are activated (or not) and participate to next global state change and output (or not). The *rate of change* (the *activity*) of these components can be conceived as their *re-activation* through many trials. Another way saying it is that there is a correlation between the activity of these components and the behavior achieved at composition level.

From a formal perspective, *systems theory* [12] offers a framework for the specification of both dynamics and topology of a system from input-output functions to a network of systems. In a network, systems activate each other through their inputs/outputs. These interface-based interactions allow determining precisely the activity locations in the network. Besides, the dynamics of systems can be discretized through discrete-events. Using discrete-events allows precisely focusing on the state changes (activity) [13] in the system. Hence, both discrete-events and systems theory offer the right explicit tools to deal with activity, structurally and behaviorally.

The *main contribution* of this work is to propose an original approach for automatically rating component systems and composing them according to the experimental frames in which they are placed. To achieve this goal, a simulation-based evaluation of components is performed. Activity-based performances consist of the correlation between component usage and network behaviors. Performances of hierarchical components are stored in a hierarchical model-base. A simulation-based stochastic search is achieved based on the performance model-base. Correlation between the activity of a component and corresponding composition outcome is referred to the *credit assignment problem*. The credit of components is used to bias their selection.

Activity-based credit assignment (ACA) proves here to:

1) Apply to any level in the hierarchy of components within any experimental frame,
2) Converge on good components/compositions faster than a repository-based random search,
3) Automatically synthesize systems from a model-base enabling reusability of highly rated components in compositions.

The manuscript is organized as follows. In Section 2, a state-of-the-art of the activity concept in simulation is presented and other approaches from systems science are discussed. In Section 3, the method for correlating both component usage and composition behavior is introduced. In Section 4, the main features, possible real world applications and the complexity levels of systems building are proposed. In Section

[1]For more information on the relationships between optimization and simulation, please refer to [5], [6], [7], [8], [9], [10].

5, discrete-event systems, activity, credit, and model-base, are defined at both formal and operational levels. In Section 6, the ACA stochastic search algorithm is described. In Section 7, a hierarchical model example is simulated and results are discussed. Section 8 compares activity-based approach and other credit assignment approaches. Section 9 discusses healthcare as an application in information and communication technologies. Finally, conclusion and perspectives are drawn.

II. State-of-the-art

In [14], Balci desribed the *four major conceptual frameworks* [(i) *event scheduling*, (ii) *activity scanning*, (iii) *three-phase approach*, and (iv) *process interaction*] that were in use to implement discrete event simulation kernels. These conceptual frameworks (also named *simulation structures* or *simulation strategies* or *world views*) guide scientists in the design and the development of their simulation model. Among these frameworks, *activity scanning* is also called two-phase approach, the first phase being dedicated to simulation time management, the second phase to the execution of conditional activities (*e.g.*, during scanning, execution of simulation functions depend on the fulfillment of specific conditions). The *three phase approach* is an optimization of the activity scanning approach. This optimization is interesting for systems in which potential activities can be detected at each time step. The first phase is the same as in the activity scanning approach. The second phase is different since it handles the execution of all unconditional activities (avoiding rules scanning for rules known to always be fired). The third phase is then similar to the second phase of regular activity scanning (an activity is considered and executed if the corresponding rule can be fired). In artificial intelligence activity scanning is known as *rule-based programming* (also known as *rule-based systems* or *expert systems*) [15]. Buxton and Laski introduced this approach in the simulation field with the *Control and Simulation Language* (CSL) [16]. In CSL, when a rule is "fired" a corresponding action is taken and the system state is updated. This approach is often considered to be dual with the event scheduling method.

From a theoretical point of view [12], a discrete-event specification of the system mathematical structure introduced by Wymore [17] can be defined. The *Discrete Event System Specification* (DEVS) consists of basic elements: discrete events and models as components composing networks of coupled models (models being atomic or coupled). Usual infomal definitions [mainly extracted from [14]] of *activity* and *event* consist of:

- An *activity* is an operation that transforms the state of a system over time. It begins with an event and ends by producing another event (linked to the termination of the activity). Some definitions in the simulation community consider that an activity is thus a period of time with a known duration.
- An *event* is what causes a change in the state of the system (eventually composed of many components).

In usual definitions, *activity* consists of a *qualitative aspect* of a system's behavior. *E.g.*, if the system is a *waiter agent*, his activities consist of "serving", "opening a bottle", "walking", etc. Executing the agent's activities is equivalent to executing some transitions in the corresponding dynamic system. Let's now consider that a modeler disposes of a collection of systems sharing the same activities. However, when testing these systems in different contexts, some systems are engaged in some activities while others are not, leading to different performances at the global level.

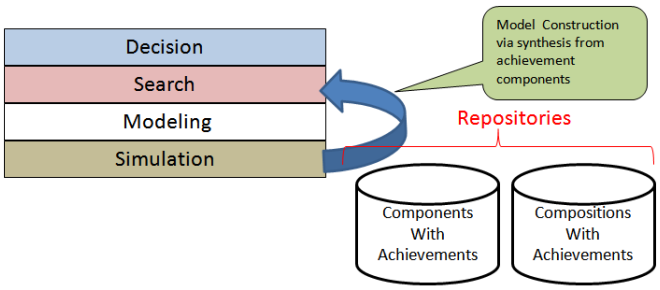


Fig. 1. Decision and search in modeling and simulation.

In this paper, the main idea is to consider *activities* as a *quantitative aspect* of a system's behavior, correlating the number of executions of dynamic system's transitions (its *quantitative activity*) with the perfomances at global level. This correlation is then equivalent to a measure of component credits and can be used as a guide to solution.

From a systems science perspective, *systems problem solving* [1] and *system design and engineering* [2] are the approaches closest to the approach presented here. However, these approaches remain at modeling level and do not use activity at component level for building systems at network level. Concerning other credit assignment methods a comparison is presented in Section VIII.

III. Correlating component usage with composition behavior

Figure 1 sketches the position of the new decision and search process layers on the top of usual modeling and simulation processes. Model construction is biased via the synthesis from achievement components stored in repositories (or model-bases).

Figure 2 presents the basic modeling and simulation entities used in activity-based credit assignment. A *correlator* entity is in charge of correlating *performance evaluations* obtained in an *experimental frame* and *simulation* results for *credit assignment* at *component-based level*.

The underlying concept formulates the approach in terms of two types of measurements: *component activity levels* and *collaboration outcome scores*. In brief, the idea is to correlate activity levels with scores to obtain evaluations of component contributions to collaboration outcomes. As illustrated, in Figure 3, each component is assigned a credit value for the outcome of a collaboration execution, or trial. This is done by multiplying its level of activity in the collaboration during that execution by the evaluation of the collaboration outcome, or score. These credit assignments are then accumulated as executions are encountered with possibly different subsets of

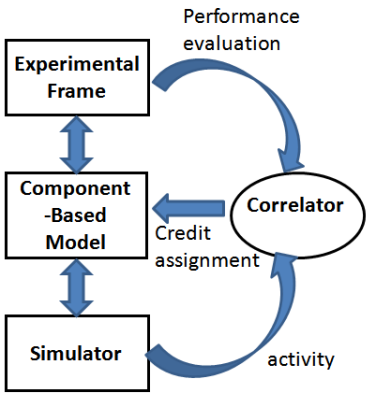


Fig. 2. Basic modeling and simulation entities in activity-based credit assignment.

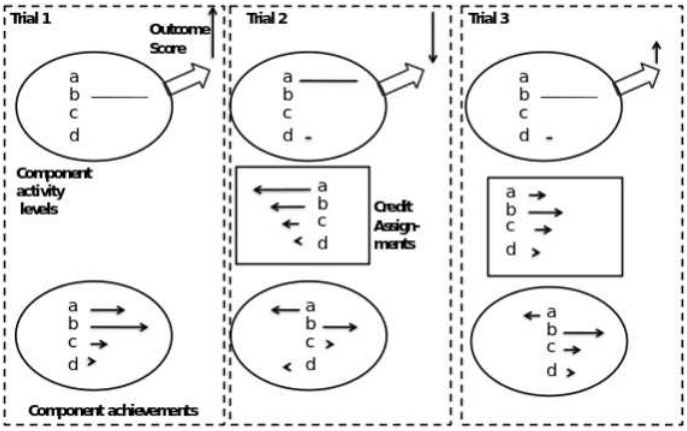


Fig. 3. Correlation algorithm for computing component achievements.

components. Over time, these credit assignments are accumulated to result in numerical ratings, called component achievements, that represent individual contributions to collaboration outcomes. For example, in the three trials shown, *Component b* emerges with a high positive achievement because it has been very active in positive outcomes (*Trials 1* and *3*) and quite inactive in those with negative outcomes (*Trial 2*). On the other hand, *Component a* emerges with a net negative achievement because it has been active when the outcomes were bad and had only minor activity when the outcomes were good. Other *components*, such as *c* and *d*, may not have been sufficiently active to accumulate meaningful achievement evaluations.

As can be appreciated from this brief exposition, the correlation scheme encompasses at least two dimensions in which variations must be addressed. There may be various ways to determine the degree of activity of a component in a collaboration and to measure the outcome of a collaboration in such a way as to support evaluating individual contributions. The two core concepts that must be successfully characterized to enable the proposed approach to work are the *activity level* and the *scoring of outcomes* to support efficient and effective credit assignment. We now discuss briefly these issues.

Activity level - Perhaps the most rudimentary measurement of component participation in a collaboration is whether or not the component was employed in the collaboration. However, this binary measure would allow components that are rarely used throughout the execution to be considered equally with ones that carry the burden of processing. A more refined metric, measures the utilization, i.e., the fractional time that a component is actually engaged in processing during some execution under consideration. In many computer system applications, such a measure is straightforward to obtain with access to the executing process thread and the system clock. However, in many contexts, the interface to the system might not allow direct access to its execution threads and clocks. In any case, the most veridical indications of activity of a component in collaboration derive from measures of its actual processing activity. That is why we formulate the basic definition to be the number of transitions in execution as a model component in a coupled simulation. This count of transitions includes both internal and external transitions, hence measures both internal computations as well as message exchanges with other components. As we will show, in simulation based on DEVS, these counts can be obtained by functions that are built into the simulator and can become part of the simulation methodology [18].

Scoring outcomes for credit assignment – these derive basically from measures of performance for the task undertaken by the collaboration – how well the task is performed by the collaboration. Such measures would typically be task-dependent but well known. However, additional constraints on the measures employed are necessary to adequately guide, or improve, the credit assignment process. For example, outcome scores that range over both positive and negative values appear to be better than ones that range only over positive values. Consider a scale from 0 to 10, after multiplication by positive activity levels, low resulting values confound low activity levels with low scores. In contrast, when using a scale ranging from -5 to $+5$, after multiplication by activity level, the relevant performance information (positive and negative) is retained as well as near zero values representing low activity and/or average performance. Thus, theory development is needed to study appropriate techniques for converting a performance measure into a score that has the best properties for informative credit assignment.

IV. From Mathematical Modeling to Real World Applications

Table I presents the *characteristic features* of the general problem of building high performing systems from system components (as well as their hockey team and multi-disciplinary physician team manifestations). In optimization, finding the right components to compose a global time state trajectory function turns out to be rapidly NP-complete as the number of time points, connections and components increase. Increasing levels of complexity of systems building can be defined considering both the qualitative (topology types) and quantitative (number of connections, components, etc.) aspects of the systems to be studied:

1) Mathematical modeling: Proofs (of ordered search set, convergence, etc.) can be established for simple parallel and series connections of components [19].
2) Simulation-based heuristics: The combination of parallel and series connections of components - for a "reasonable" number of behaviors, connections and components can studied at simulation level (cf. hockey team model described after) still without using real data but reasoning about the main mechanisms of ACA.
3) Real world applications: The explicit use of activity allows better managing real systems. E.g., fire spread sensors can more efficiently use their battery [20], or as we will see, US healthcare system can be improved.

At level 1, series and parallel connections allow generalizing simulation search results in probability theory. The fundamental aspects of ACA search (learning speed, convergence, etc.) can be mathematically studied. Having this formal basis, more complex structures (of level 2 and level 3) can be modeled and simulated.

At level 2, Table I describes how a collective sport such as hockey is representative of the general problem of building high performing systems from system components. Collective sports exhibit the interestig characeristic feature of *collaboration* (the whole being more than the parts). Indeed a good team is more than only good players. As components exchange information, players have to collaborate together to achieve a (global) team behavior. Finding a good team requires evaluating the players in the context of team games. Building a good team can be a very complex task requiring statistical modeling (e.g., think to the movie Moneyball for baseball). Here, hockey team modeling is used as an system archetype which exhibits all the characteristic features of system building. It is a good application of a global target behavior (game) at network level (team) requiring hierarchical components (lines, players…) and evaluations in context (team and opponents).

At level 3, US healthcare can be considered as a good example of real case application. US healthcare, the most expensive in the world, has been diagnosed as an assemblage of subsystems embedded in a market economy that promotes price setting by components independently and without reference to the end-to-end quality of care and cost delivered to patients. Reforming such a system requires methods to model large scale distributed complex systems using systems of systems engineering approaches [21], [22], [23], [24]. Porter [25], [26]] advocates radical reform of health care that requires that physicians re-organize themselves into Integrated Practice Units (IPUs) moving away from care that is currently based on specialties with associated hospital departments – geriatrics, obstetrics, etc. An IPU is centered on a medical condition defined as an interrelated set of patient medical circumstances best addressed in an integrated way. Examples of IPUs are those centered on asthma, diabetes, congestive heart failure, and so on. These target a cluster of related adverse health conditions and includes the most common co-occurring complications. As such, the IPU may bring together a host of specialists and services needed to treat the target in an integral manner – as a team rather than as a collection of individual entities. This assemblage of individual independent entities into a single collaborative organization fits the pattern of system-of-systems and motivates research to provide a firm basis for such integration. The IPU delivers all the services needed for the target condition which are organized into an end-to-end interaction with the patient called a full cycle of care covering a Care Delivery Value Chain (CDVC). Here "Value" is defined as health outcomes achieved per dollar of cost compared to competition and the Value Chain is a set of activities that increase value (i.e., contribute to the outcomes) from the initiation of the care cycle to its termination.

Although Porter's formulation likens healthcare delivery to manufacturing, a systems engineering approach requires developing specific system of systems concepts and supporting tools for modeling, simulation, and activity planning and scheduling of health care services. This is due to the presence of complex patient flows, numerous human resources, dynamic evolution of patient's health state [27]. Using recently developed modeling and simulation methodology and tools [28], [29], [30], Zeigler [31] discusses systems-of-systems formalization of Porter's IPU concept laying the groundwork for application of the methodology presented here to continuous improvement of such systems.

Table I exhibits the features of collaboration manifested in the IPU, as a multi-disciplinary team, which necessitate the application of the activity-based component selection methodology.

V. Activity and credit in discrete event systems

System specification and activity are linked here through activity-based credit assigment.

A. Discrete Event System Specification (DEVS)

The structure of both network and basic discrete event systems is presented here.

Definition 1. A basic Discrete Event System Specification (*DEVS*) is a structure:

$$DEVS = (X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta)$$

Where, X is the *set of input events*, Y is the *set of output events*, S is the *set of partial states*, $\delta_{ext} : Q \times X \to S$ is the *external transition function* with $Q = \{(s, e) \mid s \in S, 0 \leq e \leq ta(s)\}$ the set of total states, $\delta_{int} : S \to S$ is the *internal transition function*, $\lambda : S \to Y$ is the *output function*, and $ta : S \to \mathbb{R}_{\infty}^{0,+}$is the *time advance function.*

Definition 2. *A $DEVS$ network* is a structure:

$$N = (X, Y, D, \{M_d\}, \{I_d\}, \{Z_{i,d}\}, Select)$$

Where X is the *set of input events*, Y is the *set of output events*, D is the *set of component names*, for each $d \in D$, M_d is a *basic model* (whose structure differs from one *DEVS*-based formalism to another), for each $d \in D \cup \{N\}$, I_d is the *set of influencers* of d such that $I_d \subseteq D \cup \{N\}, d \notin I_d$ and for each $i \in I_d$: $Z_{i,d}$ is a *coupling function*, the $i-$ to $-d$

Characteristic feature	Hockey team manifestation	Multi-disciplinary physician team manifestation
collaboration requirement	team must work together no single player is sufficient	Each physician must provide his/her service to the assure successful treatments
modularity	6 distinct positions on ice skilled players can be arbitrarily plugged in to such roles	Physicians within the same discipline can be interchanged to play the same role in a team
specialization	each position has its own skill set	Physicians specialize via disciplines to play specific roles in a team
variety in compositions Substitution alternatives	18 team players 6 on ice, players get tired and replaced farm club and trades furnish additional alternatives	Physicians schedules and participation in multiple teams provide variety in compositions
system composition problem	coach must select 3 subsets of 6 that work best together to win games	Some physicians work well with others, some do not. So selecting the best team composition for a given patient care full cycle is a challenge.
what constitutes a single trial	game = 60 minutes	A single full cycle of care rendered to a patient
activity of component in a trial	player's minutes on ice	Time spent by physician in full cycle of care rendered to a patient
evaluation of trial	game outcome, e.g. goals scored – goals allowed	Value (outcome per unit cost) as defined in text

TABLE I
GENERAL PROBLEM OF BUILDING HIGH PERFORMING SYSTEMS FROM SYSTEM COMPONENTS, HOCKEY TEAM AND HEALTH CARE COORDINATION APPLICATIONS.

output translation, defined for: (i) *external input couplings*: $Z_{self,d} : X_{self} \to X_d$, with $self$ the network name, (ii) *internal couplings*: $Z_{i,j} : Y_i \to X_j$, and (iii) *external output couplings*: $Z_{d,self} : Y_d \to Y_{self}$, and $Select : 2^D - \{\emptyset\} \to D \cup \{\emptyset\}$ is the *sequential select function* (to select one component to execute its transition/output functions, among imminent components). Considering a set of components C candidate for internal transition, the sequential select function has constraint $Select(C) \in C \cup \{\emptyset\}$, *i.e.*, only one component or no components can be selected among candidates.

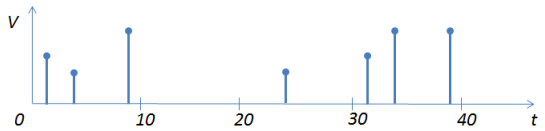


Fig. 4. An example of event trajectory.

B. Activity

Definition 3. *Event-based activity* $A_\xi(t'-t)$ [32] of an *event set* ξ consists of :

$$A_\xi(t'-t) = |\{ev_i = (t_i, v_i) \mid ev_i \in \xi, t \leq t_i < t'\}|$$

Where t, t_i, t' are *time-stamps* and $v_i \in V$ is an *event value*. *Average event-based activity* consists then of $\overline{A_\xi(t'-t)} = \frac{A_\xi(t'-t)}{t'-t}$.

For example, assuming the event trajectory depicted in Figure 4, the average event-based activity of the system corresponds to the following values for different time periods: $\overline{A_\xi(10)} = 0.3$, $\overline{A_\xi(20)} = 0.15$, $\overline{A_\xi(30)} \simeq 0.133$, $\overline{A_\xi(40)} = 0.175$.

Definition 4. *Average external activity* $\overline{A_{ext}}$, related to the counting n_{ext} of *external transitions* $\delta_{ext}(s,e,x)$, over a time period $[t,t']$ consists of:

$$\begin{cases} s' \leftarrow \delta_{ext}(s,e,x) \Rightarrow n'_{ext} \leftarrow n_{ext} + 1 \\ \overline{A_{ext}(t'-t)} = \frac{n_{ext}}{t'-t} \end{cases}$$

Definition 5. *Average internal activity* $\overline{A_{int}}$, related to the counting n_{int} of *internal transitions* $\delta_{int}(s)$, over a time period $[t,t']$ consists of:

$$\begin{cases} s' \leftarrow \delta_{int}(s,e) \Rightarrow n'_{int} \leftarrow n_{int} + 1 \\ \overline{A_{int}(t'-t)} = \frac{n_{int}}{t'-t} \end{cases}$$

Definition 6. *Total average simulation activity* $\overline{A_s(t'-t)}$ is equal to:

$$\overline{A_s(t'-t)} = \overline{A_{ext}(t'-t)} + \overline{A_{int}(t'-t)}$$

Definition 7. *Average simulation activity of a hierarchical composition (network) named* $k \in K$ consists of the sum of average simulation activities of components $i \in D$ in the network: $\overline{A_{s,k}(t'-t)} = \Sigma_{i \in D} \overline{A_{s,i}(t'-t)}$.

Here is the modification of usual abstract simulators for atomic models [12]:

Algorithm 1 Activity-based abstract simulators.

variables
  tl — time of last event
  tn — time of next event
  n_{int} — number of internal transitions
  n_{ext} — number of external transitions

when receive i-message (i, t) at time t
  $tl \leftarrow t$–e
  $tn \leftarrow tl + ta(s)$
when receive *-message $(*, t)$ at time t
  if $(t = tn)$ **then**
    $y \leftarrow \lambda(s)$
    send y-message (y, t) to parent coordinator
    $s' \leftarrow \delta_{int}(s)$
    $n'_{int} \leftarrow n_{int} + 1$
when receive x-message (x, t)
  if $(x \neq \oslash$ and $tl \leq t \leq tn)$ **then**
    $s' \leftarrow \delta_{ext}(s, x, e)$
    $n'_{ext} \leftarrow n_{ext} + 1$
$tl \leftarrow t$
$tn \leftarrow tl + ta(s)$

C. Activity credit assignment (ACA)

Definition 8. The *hierarchical candidate set* H contains all the names of candidate components. It is an indexed family of sets noted $H = \{H_l\}_{l=0}^{p}$ where each set H_l contains the names of candidate components from top network component at level $l = 0$ until atomic components at (last) level $l = p$.

Definition 9. The *evaluation function* is defined as $e_v : \Omega \times P \times P^* \rightarrow \mathbb{R}$ and operates at *top level* $l = 0$, based on:

- The *input segment* $\omega \in \Omega$ (a partial function $\omega : [t, t'] \rightarrow X$) of both *top candidate network* $k_0 \in H_0$ and *top solution network* $k_0^* \in H_0$,
- The *output segment* $\rho \in P$ (a partial function $\rho : [t, t'] \rightarrow Y$) of a top candidate network $k_0 \in H_0$, and
- The *solution output segment* $\rho^* \in P^*$ (a partial function $\rho^* : [t, t'] \rightarrow Y^*$) of the *top solution network* $k_0^* \in H_0$.

For each input segment $\omega \in \Omega$, the evaluation function e_v computes the *distance* between each output segment $\rho \in P$ (obtained by simulation) and each solution output segment $\rho^* \in P^*$. The evaluation function thus compares simulation results between a top candidate network $k_0 \in H_0$ and the top solution network $k_0^* \in H_0$, i.e., $e_v(k_0, k_0^*)$, noted $e_v(k_0)$ for short hereafter.

Definition 10. *A local optimum* $\hat{k}_0 \in H_0$ consists of $e_v(\hat{k}_0) < e_v(k_0^*)$, where $k_0^* \in H_0$ is the global optimum (corresponding to the top solution network) and $e_v(k_0^*) = v^*$ with $v^* \in \mathbb{R}$ the *maximum evaluation value*.

Definition 11. The *trial credit* (*achievement*) of an atomic component $i \in D$, at *trial* $0 < r \leq R$ (with $r \in \mathbb{N}$), over a simulation duration $[t, t']$, consists of $c_{i,r}(t' - t) = \overline{A_{s,i}(t' - t)} \times e_v(k_0)$, where $\overline{A_{s,i}(t' - t)}$ is the *average simulation activity* of the atomic component and $e_v(k_0)$ is the evaluation function of a top candidate network $k_0 \in H_0$.

Definition 12. Over a number of trials $R \in \mathbb{N}$, the *accumulated credit (achievement)* of an atomic component $i \in D$ is the sum of the trial credits at each *trial* $0 < r \leq R$: $c_{i,R}(t' - t) = \Sigma_{r=1}^{R} c_{i,r}(t' - t)$.

Definition 13. The *trial credit* of a network named $k \in H_l$ of structure $N_k \in N_l$ with N_l the set of candidate network structures at level l with $N_k = (X_k, Y_k, D_k, \{M_d\}_k, \{I_d\}_k, \{Z_{i,d}\}_k, Select_k)$, and over a simulation of duration $[t, t']$, at a trial $0 < r \leq R$, is equal to $c_{k,r}(t' - t) = e_v(k_0)\Sigma_{i \in D_k} \overline{A_{s,i}(t' - t)}$, where $\overline{A_{s,i}(t' - t)}$ are the *average simulation activities* of the (atomic or network) components composing the network. By definition, it is important to notice that candidate networks at a certain level l constitute undecomposable and *unique* chunks of solutions with their own credit - independent from the credit of their components. For example, in a hockey team, the credit of a line is not the sum of the credits of the constitutive players because players can play in other lines and other games. This is why level l is a non-trivial parameter of *ACA* search algorithms.

Definition 14. The *hierachical model-base*[2] $\mathcal{M}$ is defined as an indexed family of sets and noted $\mathcal{M} = \{\mathcal{M}_l\}_{l=0}^{p}$, where $\mathcal{M}_l$ is a *model relation* [33] *at level* l defined as $\mathcal{M}_l \subseteq H_l \times A_l \times C_l \times \mathcal{R}_l \times \mathcal{V}_l \times F_l$, where H_l is the *set of names of candidate components*, $A_l \subseteq \mathcal{A}_{ext} \times \mathcal{A}_{int} \times \mathcal{A}_s$ is the *activities relation* (with $\mathcal{A}_{ext}$ the *set of average external activities*, $\mathcal{A}_{int}$ the *set of average internal activities*, and $\mathcal{A}_s$ the *set of total average simulation activities*), C_l is the *set of accumulated credits*, $\mathcal{R}_l$ is the *set of trial numbers*, $\mathcal{V}_l \subseteq \mathcal{P}(V_{R \in \mathcal{R}})$ is the *set of sets of evaluation*s (to keep track about evaluations history for each model), F_l is the *set of frames (contexts)*. The hierachical model-base $\mathcal{M}$ contains the names of evaluated components, i.e., it is a subset H_l^v of the names of candidate components H_l such that $H_l^v = \{m \in H_l \,|\, e_v(k_0) \neq \emptyset \wedge m \in N_0\}$ with $k_0 \in H_0$.

D. Structure assembly

Figure 5 describes the elements $\mathcal{M}_l$ of the model-base $\mathcal{M}$. The simple hierarchical model example of hockey team is used. The *team* is composed of two *lines* each line being composed of two *players* each player having (being composed) of three abilities (*stop*, *attack* and *pass*). Each instance of both network and atomic models is stored in the model-base with corresponding simulation-based metrics (credit, activity, evaluations, etc.) According to these metrics, intances can be assembled together or not - at each level - to build a new team.

[2]Notice that model-base structure and behavior can be represented in the System Entity Structure formalism (cf. [12], chapter 18, for more details).

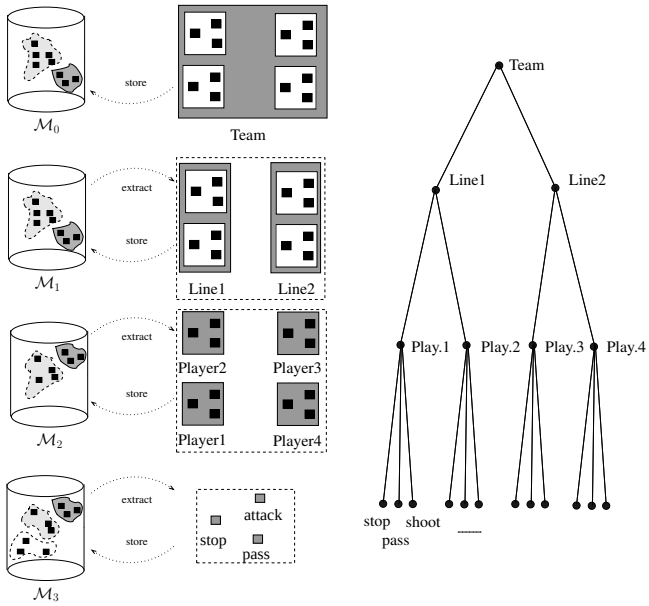


Fig. 5. Description of the hierarchical model-base.

VI. ACA STOCHASTIC SEARCH ALGORITHM

To increase robustness and to do not be trapped in local optima, some components are selected randomly. Three types of search algorithms have been implemented for comparison: (i) *Purely random search*, (ii) *Model-based random search*, and (iii) *ACA-based search. Purely random search* is used both for comparison and before *biased search*. Two kinds of *search biases* are used: (i) *memory bias* (using model-base), and (ii) *performance bias* (using credits). *Model-based random search* uses a memory bias: After some trials, some components are randomly extracted from the model-base and combined with other randomly generated components. *ACA search* uses both memory and performance biases: After some trials, some (best) components are extracted from the model-base (according to their credit) and combined with other randomly generated components.

Remark 15. *Model-based random search* is used as a benchmark to test the impact of the size of the model-base compared to the size of the solution set. Indeed, depending on the search level initially chosen, the number of candidate components varies and impact performances of the search algorithm. This will also be discussed in Section VII (*cf.* Table II).

Algorithm 2 summarizes the main $hierarchicalSearch$ function:

- Lines 7-9: For each trial below b value, an unbiased composition of network structure N_{k_0} is generated randomly, i.e., selecting uniformly non-evaluated components using the *purelyRandomSearch* function.
- Lines 10-12: For each trial above b value, a biased composition of network structure N_{k_0} is generated (*cf.* description of Algorithm 3 hereafter).
- Line 13: Network N_{k_0} is simulated until simulation time t_{end}.
- Line 14: Performances of N_{k_0} and of each of its subcomponent*s* are stored,i.e., at each level $0 \leq l \leq p$, where p is the atomic level. Components are ranked according to their credit.
- Lines 15-18: If the *maximum evaluation value* v^* is obtained optimal network name $k_0^*$is returned.

Algorithm 2 $hierarchicalSearch(type, b, t_{end}, l_s)$

1: $k_0 \in H_0$: Name of top candidate network
2: $N_{k_0} \in N_0$: Structure of top candidate network
3: b: Trial value to start using a bias
4: t_{end}: Total simulation time
5: l_s: Search level with $1 \leq l_s \leq p$
6: $type$: Type of search algorithm
7: **for** each trial $r \leq R$ **do**
8: **if** $r < b$ **then**
9: $N_{k_0} \leftarrow purelyRandomSearch()$ ▷ random non evaluated components
10: **else**
11: $N_{k_0} \leftarrow biasedRandomSearch(type, l_s)$ ▷ use model-base
12: **end if**
13: **simulate** network N_{k_0} until t_{end}
14: **recursively store and rank** each sub-component $i \in D_{k_l}$ for each level $0 \leq l \leq p$ in $\mathcal{M}_l$
15: **if** $e_v(k_0) = v^*$ **then**
16: **return** k_0^*
17: **end if**
18: **end for**

Algorithm 3 summarizes the $biasedRandomSearch$ function:

- Line 5: The network structure N_0 is initialized.
- Line 6: The number of best components to select in $\mathcal{M}_{l_s}$ is drawn using a uniform law.
- Lines 7-12 concern the *model-base random search*, which simply (randomly and uniformly) selects a number n_b of best components from model-base $\mathcal{M}_{l_s}$ while other components are non-evaluated components.
- Lines 13-22 concern the *ACA search*, which first computes the selection probability of each component in $\mathcal{M}_{l_s}$, through the *credit-based selection probability relation*:

$$\mathrm{SP}_{\mathrm{l_s,r}} = \{(a,b) \in P(a) \times H_l^v \mid a,b \in \mathcal{M}_{l_s}, \quad P(a) = \frac{c_{a,R_a}(t'-t)}{\Sigma_{b \in \mathcal{M}_{l_s}} c_{b,R_b}(t'-t)}\}$$

where $P(a)$ is the *selection probability* at trial r of a component $a \in \mathcal{M}_{l_s}$ computed as its own accumulated credit (through a number of trials $R_a \leq R$) over the sum of the accumulated credits of all the components in model-base $\mathcal{M}_{l_s}$.

- Line 23: Finally the new assembled network is returned.

Algorithm 3 $biasedRandomSearch(type, l_s)$

β: A selected best component
n_{l_s}: Number of components $|D_{l_s}|$ at search level l_s
n_b: Number of best components to select in $\mathcal{M}_{l_s}$
$type$: Type of search algorithm
$N_0 \leftarrow \emptyset$
$n_b \leftarrow \mathcal{U}(0, n_{l_s})$
if $type$ = "model-based" **then**
  for each $i \leq n_b$ **do**
    select randomly and uniformly β in $\mathcal{M}_l$
    add β to N_0
  end for
  complete N_0 with non-evaluated components
else
  if $type$ = "ACA" **then**
    compute relation $\mathrm{SP}_{\mathrm{l_s,r}}$
    for $i \leq n_b$ **do**
      select β in $\mathcal{M}_{l_s}$ according to relation $\mathrm{SP}_{\mathrm{l_s,r}}$
      add β toN_0
    end for
    complete N_0 with non-evaluated components
  end if
end if
return N_0

VII. Hierarchical modeling and simulation

ACA is applied here to the hockey team archetype model. In the first subsection, the Hockey team model is introduced. In the second subsection, simulation analysis is presented.

A. Hockey team model

A picture of the whole composition is provided in Figure 6.

This model has the following characteristics:

- Only two kinds of position are considered: Defense and attack,
- Only four players are selected at a time, two in defense and two in attack,
- There are 3 *abilities* (stop, pass, shoot) that can be *good* or *bad.* Therefore, there are $2^3 = 8$ possible players at each position,
- There are $8^2 = 64$ possible lines in defense/attack.
- There are $64^2 = 4096$ possible teams.

Figure 6 shows also the coupling of the experimental frame to the model. During a trial, the experimental frame sends one attack (puck external event) to each defender. When a defender receives the puck he must stop it, pass it to the other defender who must forward it to one attacker. The latter must pass it to the other attacker who must shoot, in oder to score a goal. Any break in this chain results in no score. To ensure a score, the abilities at every point of the sequence are required to be "good". Therefore, since there are two such attacks, the maximum score is two. Finding the best hockey team (i.e., the global optimum) consists in finding the best players (the local optima). However, once a best player is selected it is not possible to know it is a best player as only the team is evaluated.

In collective sports, many statistics aim at evaluating teams and players performances. In ice hockey [34], *plus–minus statistic* is increased by one ("plus") for those players on the ice when the team scores a goal; the plus–minus statistic is decreased by one ("minus") for those players on the ice when the team allows a goal. In basket ball [35], *player efficiency rating* correlates (among other parameters) both the *number of minutes played* and the *number of goals*.

Using credit assignment a finer evaluation of players can be achieved correlating both the activity of players (representing the number of actions they achieve during a play) and team's goals at the end of the game. Here, at the function evaluation level, the best team is found for score (maximum evaluation value) $v^* = 2$ corresponding to the global optimum. Local optima correspond to teams with partial score (intermediate evaluation value) $v = 1$.

B. Model simulation

First, the metrics and simulation process are introduced. After, an analysis of simulation results is presented.

1) Metrics and description of parallel pseudorandom simulation: Pseudorandom simulation is used through replica and seeds for statistical analysis. Simulation is fully parallelized for efficiency reasons. A simulation is determined by: a random number seed, a hierarchical level, and a bias start value (start trial at which search is biased). For each bias start value, *30* replications (random seeds) have been run in parallel on a Symmetric Multiprocessing (*SMP*) with *8* quadcore-processors (*32* cores). Each level runs roughly *40* biases for *30* replications each[3]. There are four hierarchical levels: Level 0 is the top team level, level 1 is the lines level, level 2 is the players level, and finally, level 3 is the abilities level (atomic level). Refering to Algorithm 2, first the bias start value is set to infinity and the simulation is run (without any connection to the activity credit assignment), until the maximum score is reached. The resulting number of trials is $R_{\overline{b}}$ where $\overline{b}$ indicates a *unbiased search*. This is the number of trials required for that seed at that level to find purely randomly the team that can score two goals. After, for each trial from *1* to $R_{\overline{b}}$, the simulation is run again using each biased search algorithm and the number of trials required at that level (to find the team that can score two goals) is obtained and noted R_b where $\overline{b}$ indicates a *biased search*.

For each bias start value, *trial speed-up* is defined as $\sigma = \frac{R_{\overline{b}}}{R_b}$. Note that $\sigma \geqslant 1$ for any algorithm that is not worse than random search. Also $\sigma < 1$ is possible. The same way, *activity reduction* is defined as $\alpha = \frac{A_{\overline{b}}}{A_b}$, where $A_{\overline{b}} = \Sigma_{r=1}^{R_{\overline{b}}} \overline{A_{s,r}(t_{end})}$ is the *accumulated average activity over trials for unbiased search* and $A_b = \Sigma_{r=1}^{R_b} \overline{A_{s,r}(t_{end})}$ is the *accumulated average activity over trials for biased search*. Note that $\alpha \geqslant 1$ for any algorithm that is not worse than random search. Also $\alpha < 1$ is possible.

2) Results: The behavior of *trial speed-up* is a function of *bias start values* as shown in Figure 7 for each search level $1 \leq l \leq p$. Each curve shows (the same) consistency. For

[3]It takes approximatively one hour to run one level. Approximately, it would take a day, on a sequential machine, to do the same thing.

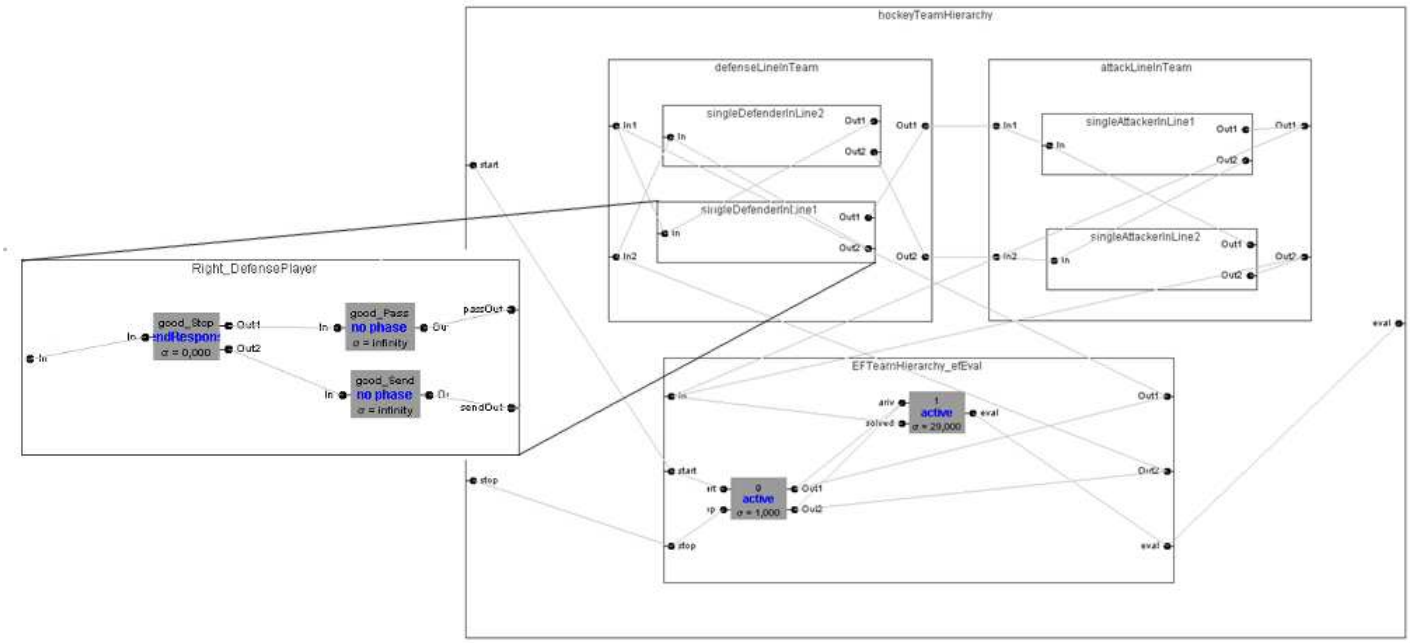


Fig. 6. Hockey Team Components. On the bottom is the experimental frame sending and receiving events (attacks and eventual goals). On the top are the coupled models: *attack and defense lines* , each line is composed of two players. Each player is composed of three abilities (cf. "Right_DefensePlayer" decomposition example on the left, with good abilities).

example, for coupled level 1 (attack and defense lines), the average speed up increases until a bias start value of *800* and then decreases. This shows that starting biased search just after *800* trials is an optimal choice (enough information has been collected). Notice too that the lower bound of the confidence interval never goes below *2* (showing that bias algorithm is always better than only randomly selecting components without any use of repository).

Table II shows the size of each component and the number of components for each level. Increasing in levels, the search space increases with the number of candidate components.

Level	Number of components	Size of each component
3 - atomic abilities	12	2
2 - players	4	8
1 - lines	2	64

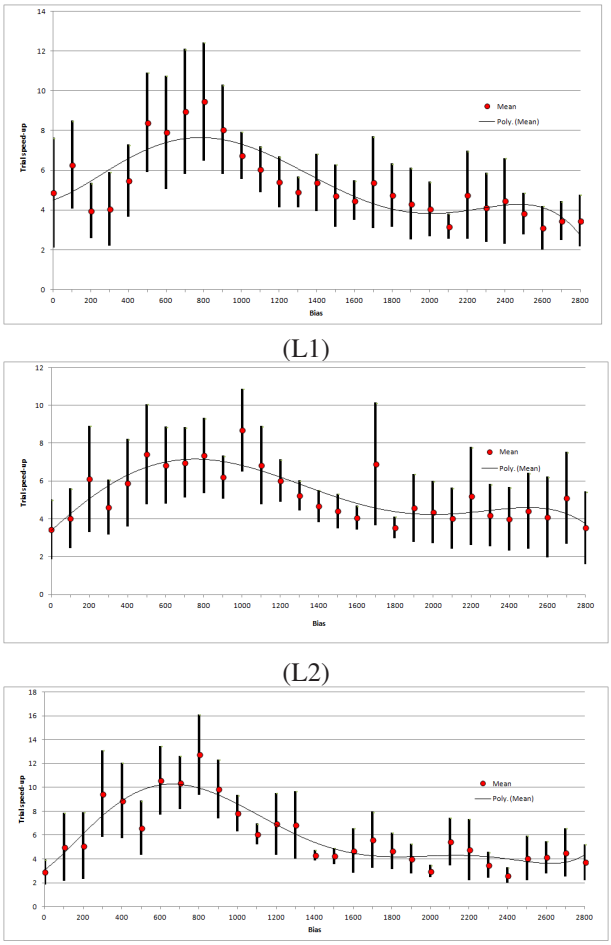
TABLE II
NUMBER OF COMPONENTS AND SIZE OF EACH COMPONENT, FOR EACH LEVEL.

Table III summarizes, at each level, the average speed-up (for all bias start values and replications) for both *random model-base* (RMB) and ACA methods. It can be seen that ACA is always more efficient. Also, the performances of RMB decrease with the size of the search set, *i.e.*, going down the hierarchy. For ACA it is the opposite: performances increase with search set size. *At level 3*, each ability is considered different, e.g., shooting in defense player in defense line is different from shooting in attack player in attack line. If the abilities were considered the same, there would be only *3* different abilities no matter what the context they were in. So in ACA *12* different abilities are considered and correlated with outcome eventually finding the *12* good versions (because they correlate well with the outcome). RMB does not take advantage of the outcome (and correlated activity) knowledge and takes longer. *At level 2*, each of the *4* players are considered different and there are *8* versions of each player. A particular version of player is more rarely seen than a particular combination of abilities. Then, ACA accumulates less information about each player than at level *3* for abilities. Hence, the difference between RMB and ACA performances reduces. *At level 1*, each of the *2* lines is considered different and there are *64* versions of each line. Since a particular version is rarely seen, ACA cannot accumulate much more information about it than RMB (notice that p-value indicates that results at level *1* are not statistically significant. . .)

method/level	1	2	3
RMB	4.5±1.88	4.0±1.43	3.5±1.71
ACA	5.1±1.72	5.1 ±1.74	5.8±1.92
p-value (%)	7,97	0,02	0,0042

TABLE III
TRIAL SPEED-UP FOR BOTH *random model-base* AND *ACA* METHODS. CONFIDENCE INTERVAL OF 95% IS INDICATED WITH ± SIGN.

Finally, Table IV summarizes, at each level, the average activity reduction for both RMB and ACA methods. First, notice that, as for speed-up, lines search set is not large enough to enable relevant ACA performances and statistical significance (cf. p-value). At levels *2* and *3*, It can be seen that both methods consume less activity than purely random one. Also, A*CA* uses less activity than RMB. In conclusion, ACA



(L1)

(L2)

(L3)

Fig. 7. Accross *30* replications, trial speed-up for each bias start value at: (l1) Coupled level 1 (attack and defense lines), (l2) Coupled level 2 (attack and defense lines), (l3) Atomic level 3 (stop, pass and shoot abilities). 95% confidence intervals are indicated for each point as vertical lines.

reduces the number of trials without inducing an increase of activity. This is confirmed by experimental results, e.g., at level 3, the average activity per trial of purely random approach is equal to *0.0366* while ACA one is *0.0370*.

method/level	**1**	**2**	**3**
RMB	4.5±1.83	3.9±1.46	3.5±1.7
ACA	5±1.66	4.9±1.65	5.8±1.81
p-value (%)	16,57	0,05	0,0035

TABLE IV
AVERAGE ACTIVITY REDUCTION FOR BOTH *random repository* AND *ACA* METHODS. CONFIDENCE INTERVAL OF 95% IS INDICATED WITH ± SIGN.

Comparing Tables III and IV, it can be noticed that activity reduction values are almost equal to speed-up values. This shows that both RMB and ACA methods do not require supplementary activity of components to increase search speed-up. Also, it indicates that activity seems to be the right measure to quantify the computational effort of components involved in a search process.

VIII. COMPARISON WITH OTHER CREDIT ASSIGNMENT METHODS

The *credit assignment problem* [36] consists in assigning partial credit to sub-decisions leading to a complete task. This allows investigating new rule paths even if the first steps do not provide immediate reward (e.g., at chess, you can chose to loose a piece to win the game...)

A first solution to credit assignment problem consists in the *bucket brigade algorithm* [37]. In the bucket brigade algorithm, the basic entities are *classifier systems*. A *learning classifier system* interacts with the environment through an I/O interface using condition-action rules. The rule-base consists of a population of many condition-action rules (*classifiers*). The rule conditions and actions are strings of characters from the *ternary alphabet:* $\{0, 1, \#\}$, with # being as "don't care" when appearing in the condition part. According to the reward of an action (changing the state of the environment) a rule reinforcement is performed. Rules are generated by a *genetic algorithm*. In the bucket brigade algorithm, "the highest bidding classifiers may place their message on the message list of the next cycle, but they have to pay with their bid which is distributed among the classifiers active during the last time step which set up the triggering conditions. (...) The central idea is that classifiers which are not active when the environment gives payoff but which had an important role for setting the stage for directly rewarded classifiers can earn credit by participating in 'bucket brigade chains'." [38]. The first classifiers of a classifier chain gets a partial reward even if the action performed on the environment does not provide the maximum reward expected. Another solution to credit assignment problem consists in the *profit-sharing plan* [39]. Bucket brigade algorithm focuses on incremental schemes. Profit-sharing plan focuses on reward schemes waiting for external rewards. In this approach, problem solving is divided into episodes delimited by the receipt of external reward. A rule is active during an episode if it wins a bidding competition. Whereas bucket brigade is better adapted to rules firing in parallel, profit-sharing is more adapted for single active chains [40].

We describe now the main differences/similarities between both activity-based and genetic algorithms(*GA*)-based credit assignments, starting from usual GA vocabulary:

- *Rules*: *GA* produces rules, i.e., pairs of *(conditions,actions)* of a classifier system, expressed in bits (or #), GA selects the best actions to make new rules. In contrast, ACA applies to any DEVS hierarchical components and compositions, where the inputs of the component/composition correspond to the conditions from the environment, and outputs of the component/composition correspond to actions.
- *Selection*:
  - GA, at every generation: 1. Rank individuals according to their fitness value, and 2. Keep a percentage of best individuals. On the other hand, ACA: 1. Ranks components and compositions according to their performances in one or more enviroment, 2. Maintains a model-base of components and compo-

sitions according to their ranking.

– *GA*, e.g., in bucket brigade, select a $rule\ i$ to fire using a bid-based probability distribution:

$$P(i) = \frac{bid_i}{\Sigma_{i=1}^{n} bid_i}$$

where n is the number of rules. On the other hand, ACA selects components and compositions using a credit-based probability distribution:

$$P(a) = \frac{c_{a,R_a}(t'-t)}{\Sigma_{b \in \mathcal{M}_{l_s}} c_{b,R_b}(t'-t)}$$

where $P(a)$ is the *selection probability* at trial r of a component $a \in \mathcal{M}_{l_s}$ computed as its own accumulated credit (through a number of trials $R_a \leq R$) over the sum of the accumulated credits of all the components in model-base $\mathcal{M}_{l_s}$.

- *Avoiding traps in local optima*: GA use mutation, i.e., a bit can be randomly changed in a rule, likewise ACA, in the model-base, uses a combination of highly ranked or randomly selected components/compositions.
- *Combination of sub-solutions*: GA use crossover genetic operator to combine parts of two parent chromosomes to make a new child chromosome. ACA, using hierarchical composition and at any level, combines (according to the above policies) components from the next lower level. Similarly to crossover both highly ranked and randomly selected lower level components are used.

IX. Discussion

Application of the ACA methodology has applications to many information and communication technologies in healthcare among other areas. In healthcare, Pathway care coordination models, modeled in the DEVS formalism, lend themselves to support critical features of such learning systems. A pathway keeps track of individuals' traversal through the IPU The activity of a pathway over a time interval is measured by the number of state transitions that occurred in the interval. The activity of the overall system is estimated by the aggregation of all individual pathway activities. A measure of a component's activity is obtained by aggregating pathway activity over all individuals that traversed the component. Since pathways include outcome measurement they enable correlation of activity and outcome per individual and aggregation over individual traversals of components to obtain component performance outcome measures. Components or variants that do not perform well in this measure are candidates for replacement by other alternatives that can be interchanged with them. Moreover, payment incentives are shown to have a significant effect in actual application where pathways with direct payment linked to outcome are shown to be executed more successfully than those without such direct linkage. Application of the activity-based payment methodology here would enable distributing rewards to team members that are not necessarily linked directly to payment but who nevertheless participate in producing high value outcomes.

X. Conclusion and perspectives

As described in sub-section VIII, *bucket brigade* algorithm is better adapted to rules firing in parallel, *profit-sharing* is more adapted for single active chains. On the other hand, ACA can be applied to models embedding both series and parallel chains, as well as cycles (feedback loops) of components. This is a reflection of the generality of the DEVS formalism as representing arbitrary dynamic systems in computational and discrete event form [12]. Although chains were employed in the hockey experiment, these were totally transparent to the ACA (there was nothing in the method that relied on, or exploited such knowledge). This generality and transparency to coupling structure suggests that ACA is worth considering as a foundation for a general framework for automated, hierarchical and efficient problem solving at the system level. Future research should test this conjecture in application to more complex systems interacting with multiple environments such as a hockey team facing diverse opponents and physician teams tackling multiple manifestations of the same medical condition.

Finally, in further research, more theoretical work can be done to justify the fact that there is an underlying credit ranking that any simulation using ACA would converge to - extending the approach to unsupervised learning. However, the results provided here still indicate that significant speed ups were achieved for ACA at each level of the hierarchy and executed within feasible time frame on relatively inexpensive multiprocessor. For ACA, an implementation and study of context independence can be worth. For example, considering hockey team archetype, abilities could be considered to be the same and independent of context so there are only *3* abilities. Then, the difference between ACA and RMB should increase.

Acknowledgements

This material is based in part upon work supported by the Centre National de la Recherche Scientifique and the National Science Foundation under Grant Number CMMI-1235364. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] George J. Klir and Doug Elias. *Architecture of Systems Problem Solving*. Springer, 2003.

[2] A.W. Wymore. *Model-Based Systems Engineering*. Systems Engineering. Taylor & Francis, 1993.

[3] Philippe Coussy and Adam Morawiec. *High-Level Synthesis: From Algorithm to Digital Circuit*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[4] Fei Tao, Yuanjun Laili, Yilong Liu, Ying Feng, Qining Wang, Lin Zhang, and L. Xu. Concept, principle and application of dynamic configuration for intelligent algorithms. *Systems Journal, IEEE*, 8(1):28–42, March 2014.

[5] Michael C. Fu. Feature article: Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing*, 14:2002.

[6] M.C. Fu, F.W. Glover, and J. April. Simulation optimization: a review, new developments, and applications. In *Simulation Conference, 2005 Proceedings of the Winter*, pages 13 pp.–, 2005.

[7] Michael C. Fu, Sigrún Andradóttir, John S. Carson, Fred Glover, Charles R. Harrell, Yu-Chi Ho, James P. Kelly, and Stephen M. Robinson. Integrating optimization and simulation: Research and practice. In *Proceedings of the 32Nd Conference on Winter Simulation*, WSC '00, pages 610–616, Orlando, Florida, 2000. Society for Computer Simulation International.

[8] L.J. Hong and B.L. Nelson. A brief introduction to optimization via simulation. In *Simulation Conference (WSC), Proceedings of the 2009 Winter*, pages 75–85, 2009.

[9] J.R. Swisher, P.D. Hyden, S.H. Jacobson, and L.W. Schruben. A survey of simulation optimization techniques and procedures. In *Simulation Conference, 2000. Proceedings. Winter*, volume 1, pages 119–128 vol.1, 2000.

[10] J.F. Santucci and L. Capocchi. Discrete optimization via simulation of catchment basin management within the devsimpy framework. In *Winter Simmulation Conference*, pages 205–216, 2013.

[11] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. MIT Press, 1998.

[12] H. Praehofer B. P. Zeigler, T. G. Kim. *Theory of Modeling and Simulation*. Academic Press, 2000.

[13] Alexandre Muzy, Franck Varenne, Bernard P. Zeigler, Jonathan Caux, Patrick Coquillard, Luc Touraille, Dominique Prunetti, Philippe Caillou, Olivier Michel, and David R. C. Hill. Refounding of the activity concept? towards a federative paradigm for modeling and simulation. *Simulation*, 89(2):156–177, 2013.

[14] Osman Balci. The implementation of four conceptual frameworks for simulation modeling in high-level languages. In *WSC '88: Proceedings of the 20th conference on Winter simulation*, pages 287–295, San Diego, California, United States, 1988. ACM.

[15] Carlos Iván Ches nevar, Ana Gabriela Maguitman, and Ronald Prescott Loui. Logical models of argument. *ACM Comput. Surv.*, 32(4):337–383, 2000.

[16] J. N. Buxton and J. G. Laski. Control and simulation language. *The Computer Journal*, 5(4):e194–199, 04 1962.

[17] Tuncer I Oren and Bernard P Zeigler. System theoretic foundations of modeling and simulation: a historic perspective and the legacy of a wayne wymore. *SIMULATION*, 2012.

[18] Santucci, J.F. and Capocchi, L. Implementation and analysis of devs activity-tracking with devsimpy. *ITM Web of Conferences*, 1:01001, 2013.

[19] Alexandre Muzy and Bernard P. Zeigler. A conjecture from learning simulation of parallel and series connections of components. In *26th European Modeling and Simulation Symposium (EMSS) - Model Engineering Workshop, accepted for publication*, 2014.

[20] Xiaolin Hu and Bernard P. Zeigler. Linking information and energy - activity-based energy-aware information processing. *Simulation*, 89(4):435–450, 2013.

[21] Valdez RS, Ramly E, and Brennan PF. Industrial and systems engineering and health care: Critical areas of research - final report. Technical Report 10-0079, Rockville, MD: Agency for Healthcare Research and Quality, 2010.

[22] Mo Jamshidi. *Systems of Systems - Innovations for the 21st Century*. Wiley, 2008.

[23] Saurabh Mittal, Bernard P. Zeigler, Jose L. Risco Martin, Ferat Sahin, and Mo Jamshidi. *Modeling and Simulation for Systems of Systems Engineering*, pages 101–149. John Wiley & Sons, Inc., 2008.

[24] Bernard P. Zeigler and Hessam S. Sarjoughian. *Guide to Modeling and Simulation of Systems of Systems (Simulation Foundations, Methods and Applications)*. Methods and Applications. Springer, 2012.

[25] Michael E. Porter and Elizabeth Olmsted Teisberg. *Redefining Health Care: Creating Value-based Competition on Results*. Harvard Business Review Press, 2006.

[26] Michael E. Porter. Measuring health outcomes. *New England Journal Of Medicine*, (363):2477–81, 2010.

[27] Vincent Augusto and Xiaolan Xie. A modeling and simulation framework for health care systems. *IEEE T. Systems, Man, and Cybernetics: Systems*, 44(1):30–46, 2014.

[28] B.P. Zeigler, E. Carter, C. Seo, C. K Russell, , and B. A. Leath. Methodology and modeling environment for simulating national health care. In *Autumn Simulation Multi-Conference (AutumnSim'12)*, pages 30–46, 2012.

[29] B.P. Zeigler, Chungman Seo, and Doohwan Kim. System entity structures for suites of simulation models. *International Journal of Modeling, Simulation, and Scientific Computing*, 4(3):1–11, 2013.

[30] B.P. Zeigler. The role of modeling and simulation in coordination of health care, keynote lecture. In *4th International Conference on Simulation and Modeling Methodologies, Technologies, and Applications*, Vienna, Austria, 2014.

[31] B.P. Zeigler. Formalizing porter's integrated practice unit with system-of-systems modeling and simulation, keynote lecture. In *10th International & International Conference of the Hellenic Society for Systemic Studies*, Athens, Grece, 2014.

[32] Alexandre Muzy, Luc Touraille, Hans Vangheluwe, Olivier Michel, Mamadou Kaba Traoré, and David R. C. Hill. Activity regions for the specication of discrete event systems. In *Spring Simulation Multi-Conference Symposium On Theory of Modeling and Simulation (DEVS)*, pages 176–182, 2010.

[33] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, June 1970.

[34] Timo Seppa, Ryan Schwepfinger, and Jesse Spector. *Hockey Prospectus 2013-14*. CreateSpace Independent Publishing Platform, 2013.

[35] S.M. Shea and C.E. Baker. *Basketball Analytics: Objective and Efficient Strategies for Understanding How Teams Win*. CreateSpace Independent Publishing Platform, 2013.

[36] Marvin Minsky. Steps toward artificial intelligence. In *Computers and Thought*, pages 406–450. McGraw-Hill, 1961.

[37] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, 1992.

[38] J. Schmidhuber. A local learning algorithm for dynamic feedforward and recurrent networks. *Connection Science*, 1(4):403–412, 1989.

[39] John J. Grefenstette. Credit assignment in rule discovery systems based on genetic algorithms. *Mach. Learn.*, 3:225–245, October 1988.

[40] A. H. Gilbert, Frances Bell, and Christine L. Valenzuela. Adaptive learning of process control and profit optimisation using a classifier system. *Evolutionary Computation*.

Alexandre Muzy is chargé de recherche at CNRS. His research mainly concerns activity-based theory of modeling and simulation. He is associate editor of Simulation journal and member of many scientific committees of international conferences.

Bernard P. Zeigler is Emeritus Professor at Arizona Center of Integrative Modeling and Simulation and Chief Scientist at RTSync Corp, Arizona and Maryland. He is internationally known for his seminal contributions in modeling and simulation theory.