A Nontraditional Approach for a Highly Interactive Collective-Adaptive System

Emergent Lightweight Learning and Collective Adaptation

Dávid Lányi
Department of Telecommunications
Budapest University of Technology and Economics
Budapest, Hungary
dlanyi@hit.bme.hu

Borbála Katalin Benkő
Department of Telecommunications
Budapest University of Technology and Economics
Budapest, Hungary
bbenko@hit.bme.hu

***Abstract*—In this paper, we describe a nontraditional approach for realizing a highly interactive adaptive system with a minimal level of intrinsic knowledge; which opens up the way for a more open and active adaptation to the rules and dynamic changes of the environment. First, we discuss a lightweight, incremental adaptation model where knowledge and strategy emerge during (and throughout) the system's normal operation. Then, this model is extended with a collective mechanism: a society of learners makes use of each other's findings in order to converge faster. Finally, principles are evaluated with simulation in a theoretical showcase ('Connect-5') world.**

Keywords — collective-adaptive systems, open learning

I. INTRODUCTION

Emergence is one of nature's strongest weapons for handling complex, dynamic and large scale problems. The application of surprisingly simple algorithms, when done in multitudes, may lead to a different quality of global results. Certain fields of emergent behavior have already been utilized in computer science (such as self-organization), while other possible application areas are yet untouched. In this paper we propose to face the challenge of open adaptive systems with an emergent solution.

Adaptiveness–in our context–means that the system is able to dynamically respond to the changes of the environment. A traditional approach for achieving adaptivity is to maintain a self- and/or environment model, detect when changes occur, and explicitly start an adjustment process within the feedback loop. [1] Powerful tools like reasoning, semantics and ontology guarantee that the adaptation is effecttive and convergent.[3] However, the success of this approach largely relies on the accuracy of the system's *explicit knowledge*: on the completeness of the world *model* and on the efficiency of built-in *adjustment mechanisms*. As long as environmental changes are in line with it, the system is guaranteed to adapt efficiently; but it is theoretically impossible to react to changes that are not included in the model, or to choose adjustment strategies that were not encoded previously.[4,5,6] Our research focuses on the question: how much can we avoid this burdening explicit knowledge in an adaptive system, thus, unbind the learning process, and let the system openly find its way for achieving adaptation (instead of following what scientist taught it to do)? An extremity may be a system without any pre-injected model or strategy, an idea which opens up dimensions yet unknown in dynamic adaptation. In this paper we propose a knowledge-poor adaptation model (the amount of explicit, pre-injected model is kept very low) that we prove to be open and effective for a complex dynamic problem case.

Adaptivity is often required in distributed situations, where autonomous building blocks of the system need to find their optimum locally, without central help or control. Collective adaptive systems make use of the connectedness of individual blocks–through communication–in order to help converging faster or globally better. The presence of an explicit world model facilitates the collective behavior in this case, as members of the society share a common and well-defined understanding of the world and of possible strategies. The question we asked was: is it possible to establish meaningful cooperation between independent adaptive systems without sharing an explicit world model; is it possible to share ones expertise and help others to adapt better even if their–independently developed–knowledge and strategy is highly different? We describe a collective self-evaluation and expertise sharing mechanism for the society of knowledge-poor adaptive systems, and discuss the effects.

The structure of the paper is the following. Section II defines the problem space: a theoretical world with an adaptation challenge and easy measurability. Section III introduces the knowledge-poor learning and adaptation model. Section IV generalizes the model from a collective perspective, where independent learners share knowledge with each other. Section V discusses further aspects, consequences and limitations of the pro-posed models and algorithms. Section VI discloses evaluation results about the goodness of the individual and collective algorithms. Section VII concludes.

II. PROBLEM STATEMENT

The problem space tackled with in this paper is a world with actors who observe their environment and make actions from time to time. Certain states of the world mean reward to the actor who reaches it, while other states result in penalty. The world is only roughly modeled by the actors–the set of ob-servable factors is limited–and actors don't have pre-injected knowledge or assumptions about rules, requirements or strategies of the world. We focus on scenarios which can be described in means of discrete time-steps, one or more actors take part, and their actions modify to the world's state.

 ISBN: 978-1-61208-103-8

The environment is not only dynamic because of the presence of other actors (who also act), but also in means of general requirements or rules which may change from time to time. Actors get known with the actual requirements implicitly, through reward or penalty provided by the environment (reinforcement learning).

While numerous cases can be imagined satisfying the above specification; we choose a showcase in which the basic abilities of the system can be demonstrated and efficiently evaluated. This is actually a generalization of a simple two-player, deterministic, fully observable, zero-sum board game, often known as connect-5, gomoku, or amoeba.

A. Starting point: a basic Connect-5 World

Connect-5 is a simple board game, an extension of tic-tac-toe for bigger sized boards and longer combinations. There are two players in the game, one with mark X and the other one with mark O. The game starts with a board of tabular arranged empty square cells. Players make steps intermittently; each one places their mark onto an empty cell. A player wins the game, if five of their signs is placed consecutively in one row, column, or in a diagonal line on the board. When a player wins, the other one looses. Tie is reached, if the board has no more empty cells, but no one has won.

More formally, the *state* of the game is represented by the board itself (cells and their contents). The transition between states is the *action* of an actor, and the game is basically a time series of game states. Only one actor is allowed to perform action in each particular state. The action is mandatory, so if there is an empty cell on the board, the upcoming actor must act. After each action, the new state is evaluated by the environment, and if winner or tie state is reached, actors receive *feedback*.

Actors are able to observe a rough model of the actual environment at any time; and are also able to receive feedback about winning/losing/tie state. They also may accumulate a local knowledge base from these observations.

The goal of the actor is to perform actions that lead to a winning state, while the environment is dynamically modified by the opponent from time to time.

B. Generalized Connect-5 World

While the basic connect-5 world incorporates several important properties of our problem space, we decided to generalize it in order to include further aspects of dynamism and collectiveness.

- ***Collectiveness***. Instead of a single actor-opponent pair, the world consists of a society of players, engaged in multitudes of parallel games. The members of the society are still autonomous actors–with individual experience, strategy and decision ability –, but they also possess the ability of communicating with each other. Actors may share their expertise with other actors, or learn from others' shared knowledge. Please note that it's not guaranteed that the members of the society face the same problem instance, e.g., same opponent style or the same rules –; nor do we say that the knowledge of any individual agent is guaranteed to be of help for others. However, the pure ability of sharing one's dynamically built knowledge is an important attribute for a collective system, also from the theoretical point of view. Pair-wise knowledge sharing may also–but not necessarily–lead to the emergence of society level "common knowledge".
- ***Dynamic opponent style and strength***. The strategy, goodness and consistency of the opponent may change over time, resulting in dynamically changing environmental requirements from the actor's point of view. Extremities may be a random opponent (just picking random steps), and an analytically optimal opponent (using a mathematically optimal strategy).
- ***Changing game rules***. We also allow the game rules to be changed dynamically during the actor's life cycle. For example, the competitive aspect may be removed, so, the player gets rewarded for their own 5-long series regardless of the other player's moves. Another way of changing the rules is to modify the length of the required series: e.g., 4 or 10 items long series may mean victory.

The generalized model keeps the following attributes of the basic world (a) The states of the world form a time series. (b) Actors are able to observe the world's state and perform actions. They may receive feedback from the environment in certain states. (c) The world changes because of the actor's action or because of factors that are outside of the actor's control (e.g., opponent's action). Besides that, we also made the assumption that the actor has no pre-injected knowledge of the rules of the world or about the goal to reach–it has to reach (positive) game states by 'learning by doing'. [9] This may be a selfish requirement under static conditions (where explicit world mode-ling could result in optimal behavior from the startup), but our goal here is to ensure the openness of the system and its dynamic adaptation ability for immensely new requirements.

The state space in the showcase example–supposing a limited board size–is finite. However, we don't see that as a hard limitation from the theoretic side, because the observation ability of an agent is finite by definition (so any board size larger than the player's vision would work as infinite), and the mathematical model we use for learning can be easily extended for non-binary (multi-value or continuous interval) cases.

Summarizing the above, the agent's job is to learn the dynamically changing rules of the world through a feedback mechanism in order to select actions that lead to success; plus, to do this on-the-fly, without having had any pre-injected knowledge or preliminary training session; and possibly in a collective manner (by knowledge sharing).

III. BASIC LEARNING MODEL

This section describes the basic learning and adaptation model used within the open autonomous agents. The model combines known basic models (Markov Decision Process and Temporal Difference Learning) with specific extensions.

A. Markov Decision Process

The inspiration of our model comes from *reinforcement learning* (RL), a general approach that tackles with problems very similar to our problem statement. RL is not one specific

 ISBN: 978-1-61208-103-8

mechanism but a dynamically improving domain of machine learning models. The common approach [7] focuses on finding actions in an actual world state, in order to achieve a goal desired in that context. It is assumed, that an agent completing this task is able to sense the environment to some extent, is able to perform actions which influence that state, and is able to receive feedback from the environment about its success.

A widely used mathematical model describing reinforcement learning problems is the *Markov Decision Process* (MDP). It is defined as a five-tuple (S, A, R, P, γ), where S is the set of world states, A is a set of actions, P is a state transition probability function $P : S \times A \times S \mapsto [0, 1]$, (where $P(s', a, s)$ tells the probability of reaching state s' after performing action a in state s), R is the reward function $R \mapsto \mathbb{R}$, and γ is a discount factor from interval (0, 1]. It's important to note that the observation capacity of the agent is limited; hence, the state observed may significantly differ from the real world state. At this stage, S refers to the real world state (later, it will be replaced by the agent's perception).

RL models are often equipped with value functions and policies to help making automatic decisions. We follow the terminology and notation of [10]. There is a *value function* defined for states which is able to better describe the real utility of a state than the reward function itself (which would only tell whether state is terminal). The value function is associated with a *policy* π, which is a mapping from states to actions, $\pi : S \mapsto A$. A value function for a given policy, $V^{\pi} : S \mapsto \mathbb{R}$ is defined as the expected discounted sum of rewards received when starting (in t=0) from state s, and following policy π: $V^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) | s_0 = s, \pi\right]$. It can be shown [2] that this value function satisfies Bellman's equation, and may be expressed in the following way:

$$V^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} P(s', a^{\pi}, s) V^{\pi}(s')$$

If the reward function R and transition probability function P are both known, this formula can be analytically solved as a linear system. However, in most RL settings–including our case–the probability function P is not known; the agent only has access to a subset of state transitions (own experience) and to the feedback coming from the reward function. [8]

To limit the size of $|S|$ –to keep computations on an easy-to-handle level –, often, estimations or approximations are used instead of exact models or values.

We also introduced a feature extraction step between the raw perceived state and the state principle used within the model. Feature extraction helps highlighting important properties of a state, by preprocessing the raw observation before learning. The exact realization of this feature extraction step is an important attribute of the agent, as the extracted information deeply influences the learning process. In the basic model, the feature extraction mechanism is wired-in (and this is all the knowledge–even though being implicit, we call it knowledge– the agent gets at startup). In general versions of the model, features may be introduced or removed on the fly.

In means of terminology, a feature based linear approximator [2] for the value function is defined as:

$$V^{\pi}(s) \approx w^T \phi(s)$$

where $\boldsymbol{\Phi} \in \mathbb{R}^k$ is a feature vector based abstraction belonging to the state s, and $w \in \mathbb{R}^k$ is a parameter vector.

While the usage of feature vectors was originally suggested in order to keep the computational complexity under control and to be able to deal with large or even infinite state spaces, we use it for two other purposes, respectively: (a) to facilitate convergence with the selection and usage of relevant and meaningful features, and (b) to bring openness into the model through the possibility of dynamically adding and removing features–hence refreshing the implicit world model of the agent.

With these approximations we lose the applicability of Bellman's equation, but there are other efficient ways for finding the solution, such as the LSTD.

B. Least-Squares Temporal Difference Method

The Least-Squares Temporal Difference (LSTD) algorithm provides way for finding a parameter vector w that approximately satisfies Bellman's equation. Without the full deduction of the method discussed in [10] and recalled in [2] we denote the main formulae, and review it from the aspect of our setting. LSTD attempts to find a fixed point of the approximation

$$w = \tilde{f}(w) = \underset{u \in \Re^k}{\operatorname{argmin}} \left\| \Phi u - (\tilde{R} + \gamma \Phi' w) \right\|^2$$

in which Φ and Φ' are matrices containing m samples of observed state transitions from s to s' in their rows, represented with $\Phi(s)^T$ and $\Phi(s')^T$ in each row; $\tilde{R}$ is a vector containing the obtained reward r_i for each of the m transitions. Because the term to be minimized contains Euclidian norms only, the optimal fixed point can be analytically determined by solving a linear system $\tilde{A}^{-1}\tilde{b}$, where

$$\tilde{A} = \sum_{i=1}^{m} \phi(s_i)(\phi(s_i) - \gamma\, \phi(s_i'))^T \qquad \tilde{b} = \sum_{i=1}^{m} \phi(s_i)\, r_i$$

In other words, the only knowledge required by the agent for selecting the desirable next state is only a vector (b) and a matrix (A).

- ***Vector b*** gives a picture about the perceived goodness of each state, based on the total (positive or negative) reward experienced there.
- ***Matrix A*** describes the experienced state transition pairs. Transitions model the effect of the agent's action along with the effect of the opponent's action, in one unit. The discount factor helps in distinguishing between states immediately preceding an end state and states that are far away. Please note that this abstraction does not include any preconception about the number or nature of opponents, so the model is also applicable for n > 2 players.

From our point of view, the most important property of vector b and matrix A is that they can be constructed iteratively; each new experience means a minor addition to them.

The agent may use two different approaches for translating the knowledge (matrix A, vector b) into an action:

(1) Calculate the expected effect of each possible action, and select the most desirable one based on the value of the result state. (If the number of actions is too large–or infinite–, a sampled subset of the possible actions may be used, with hierarchical refinement.)
(2) Analytically identify the most desirable result state and then search for an action that leads to it. This approach is more problematic because (a) it's not guaranteed that the state space is connected enough so an arbitrary end state can be reached from the current state, and (b) transitions are not deterministic because of the effect of the opponent, so we may end up in a very different end state than desired.

We used the approach (1) in the model.

IV. COLLECTIVE LEARNING MODEL

In case of the collective learning model autonomous agents share their locally developed knowledge with each other. Knowledge sharing is realized as multitude of pair-wise shares, in a self-organizing manner, without central control and without stashing common knowledge centrally.

This requires the followings: (a) Knowledge import model: a mechanism to integrate external knowledge into the one's own knowledge base, (b) Self-evaluation mechanism: a metric for the agent to evaluate the goodness of its knowledge in the actual environment, and (c) Sharing and acceptance mechanism: a mechanism that initiates and controls knowledge sharing / acceptance. Participants of the share–donor and receptor–are autonomous elements, so it's their free decision what, when and with whom to share or accept.

A. Knowledge Import Model

The import model uses the previously described matrix A and vector b as the manifestation of the knowledge; this is what the donor shares with the receptor. As shown previously, A and b are built up iteratively, thus, they're additive.

We defined the knowledge import mechanisms the following way: the new knowledge of the receptor is a weighted combination of its old and the donor's shared knowledge. Weights are denoted by c_1 and c_2.

$$A_{new} = c_1 A_{original} + c_2 A_{import} \qquad b_{new} = c_1 b_{original} + c_2 b_{import}$$

Weights define the influence of the imported elements. An extreme case is when the original knowledge gets zero weight meaning that the imported knowledge replaces the receptor's own knowledge (suppressive import). In this case the receptor becomes the donor's equal copy or clone. This may be desirable if the imported knowledge is guaranteed to be of high value while the knowledge of the receptor is clearly non-performing. However, suppressive import may easily lead to a drastic drop in the population's diversity which may become dangerous when the environment changes.

Non-suppressive import of good knowledge may perform somewhat weaker on the short term, but it keeps the population diverse which is a useful property on the long term. Combinatory import may also accumulate a more general knowledge than suppressive one, because it tends to store information about uncommon parts of the state space (which may become handy when usual strategies stop working).

B. Self-Evaluation Mechanism

The self-evaluation mechanism of the agents is based on a sliding window memory about the outcome of the last few games. Contents of the sliding window are summarized: a game won counts as +1, a lost game counts as -1 and tie counts as zero. (This is just a despotic choice, more complex models could also consider trends, the goodness of the opponent etc.)

C. Sharing and Acceptance Decisions

We didn't define explicit triggers for knowledge sharing because we believe it's not possible to say that a certain knowledge instance is guaranteed to be helpful or unhelpful for others. Instead, a sharing protocol was defined: when a donor is ready to share, it contacts another agent who–if decides so–becomes the receptor. We examined the following sharing patterns:

- Random sharing model. Agents initiate/accept the transaction with a given probability.
- Self-confidence based model. Agents with high self-evaluation values offer their knowledge, and agents with low self-evaluation values accept it.
- Knowledge density (completeness) based model. The goodness of the knowledge is measured by its completeness (e.g., number of filled cells in A).
- Opponent-based model. The receptor prefers knowledge that contains data about its current opponent.

Our model extends the state of the art in the followings: (a) applies temporal difference (TD) learning for the problem of adaptive systems where requirements change dynamically over time (b) introduces the possibility of on-the-fly, automatic feature injection (c) brings TD learning into a collective dimension.

V. DISCUSSION

This section discusses consequences, generalization directions and limitations of the pervious models.

The descriptive properties of the model were partially covered in Section III: the model is suitable for problems where the state of the environment changes from time to time and the actor is able to perform actions picked from a finite or infinite set of possible actions. Opponents and environmental rules are not directly modeled within the agent's knowledge, so the model is generally applicable for multi-actor situations. The learning process is knowledge-poor, so, except for the initial features, the agent does not need pre-injected knowledge.

Learning happens naturally, during the agent's normal activity; which is in contrast with today's popular adaptive system approaches, where the learning phase precedes the phase of normal operation.

The most important factors influencing the learnability of a problem are (a) what the agent perceives from the world, hence feature extraction, and (b) how well the actual problem case is presented, hence, the behavior of the opponent.

Opponents may significantly influence the convergence of the learning process, especially for upexperienced agents. When playing against a dummy (e.g., random) opponent, the

 ISBN: 978-1-61208-103-8

agent easily spends significant amount of time in irrelevant sections of the problem space–as none of the players knows how to become successful. In case of a strong opponent the agent learns fast what to avoid and, probably, also what to do to win (see Section VI). It's an interesting question whether the strongest opponent is the best, or it's more optimal to learn against consequent but imperfect players. The advantage of an imperfect opponent is that it leaves space for the agent to learn how to correct errors and how to make use of the other's mistakes. We believe, and tests indicate, that the variety of opponent styles leads to the best kind of knowledge for static and dynamic cases, respectively.

Feature extraction is the heart of the agent's learning model. A novelty in our model is that features are not bound to be static: they may be introduced or removed dynamically, during runtime.

- Features may be introduced (a) at knowledge import, where the donor agent does not only transfer its knowledge but also the mechanism how the new feature can be calculated, or (b) through systematic generation where the agent uses data mining based techniques to generate new features or to derive them by combining existing ones. (The exact mechanism of data mining based feature generation is outside of the scope of this paper.)
- Irrelevant features may be removed in order to minimize the problem space, thus, facilitate convergence. The trigger of that may be (a) knowledge import where the agent may decide to remove those features that are missing from some/ any of the knowledge bases (b) in an explicitly executed feature optimization step which may be a mathematical matrix minimization method (e.g., principal component analysis or singular value decomposition) or the society may use a genetic algorithm based feature selection. (Experiments confirmed both directions, but the details are again outside of the scope of this paper.)

Too fast convergence in the knowledge may be dangerous because it develops over-specialized strategies that work well against the current opponent but may not help if the environment changes. To avoid overspecialization, the agent may choose prevention strategies, such as picking second-best directions. Such a strategy leads to a better coverage of the problem space, which may be suboptimal in the current game, but could help against future opponents with yet unknown strategies.

The challenge of the agent is not just to learn adapting to (playing well within) the current environment; but to *adjust to the dynamic changes of the environment*. Changes may range from mild (slightly different opponent style) to drastic (essential rule change in the game). Agents may face this challenge alone or as a society.

- Standalone adaptation strategies include: (a) for slight changes: prevention of over fitting by better problem space coverage, and (b) for drastic changes: self-evaluation triggered knowledge deprecation –when the agent feels a significant performance drop it devalues or completely clears up its existing knowledge.
- Collective adaptation strategies include: (a) when the problem space is locally homogeneous: learning from neighbors, (b) knowledge generalization through sharing and combination and (c) for drastic changes: fast, population-wide propagation of the up-to-date knowledge.

Collective mechanisms may be biased by distortions of the self-evaluation metrics. Self-evaluation is intrinsically subjective; the agent possesses empirical information only which means that the metrics is biased by its experience–opponents–by definition. Agents facing weak opponents may overvalue themselves, while agents with strong opponents may do the opposite. The evaluation bias may gain attention when it comes to sharing the knowledge: a receptor in a hard environment, so with low self-confidence, may overvalue the weak-environmental donor's knowledge just because of its false self-confidence. However, it would be heedless to say that receiving such knowledge–unless suppressive–is guaranteed to be unhelpful. When treating it (choosing c_1 and c_2) with caution, even that kind of import may prove to be useful, because it covers a different, yet un-known sub-domain of the problem space (see Section VI).

Altogether, we think that the descriptive power and generalizability of the model is high. We can also imagine a possible convergence between this knowledge-poor approach and today's knowledge intensive directions, where the two may strengthen each other (e.g., in feature generation).

VI. EVALUATION

Models were evaluated through simulation. This section includes the most important results about the standalone learning, adaptation ability, and the collective dimension.

The *standalone model* was evaluated along two lines. First we wanted to see, how efficient is the on-line learning ability in the connect-5 world, with no initial experience, trained against opponents with different strengths. In this setting, we also measured, how the self-evaluation mecha-nism performs compared to an objective evaluator. After this we examined how trained agents react to drastic environmental changes.

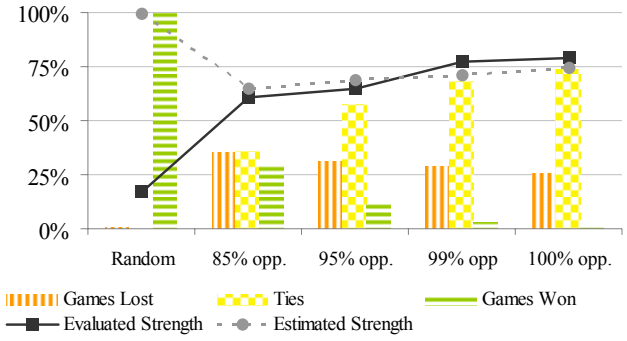

Figure 1. Learning characteristics and self-evaluation vs. opponent style.

On-line learning with no initial knowledge. The experiment consisted of an adaptation phase and an objective evaluation phase. (1) First, the untrained agent plays 350 games against a fixed-algorithm opponent, and uses its learning mechanism to adapt. Win/lost/tie statistics were also collected here. We evaluated five identical agents, each playing with a different opponent, namely: one random player; and the four opponents with mathematically optimal strategies but with

some–15%, 10%, 1% and 0%–chance of making an error (failing to choose the perfect action). (2) In the evaluation phase, learning was switched off in order to get an unbiased picture about the knowledge of each agent. Agents were allowed to use their existing knowledge, and were evaluated by playing 100 games against the "perfect" (0% failure rate) opponent, as an absolute measure. Their preliminary self-evaluation (based on the training phase) was compared to the actual measured strength. (Strength is defined as the percentage of non-lost games.)

Columns in Figure 1 visualize the outcome of the adaptation phase, while the curves refer to the self-evaluated and objectively measured strength. Training results show that the number of games won by the agent falls as opponents get stronger. Surprisingly, the number of lost games does not increase with stronger opponents; instead, games tend to end more often with a tie. Evaluation results indicate that the real gameplay strength is higher for agents trained against stronger opponents. Please note, that although the agent trained with the random player holds the lowest strength, it could also fray out a tie in 17 percent of the games against the strongest opponent. The difference between the self-estimated strength and the actual strength is unexpectedly small, expect for the divergent (random) training environemnt.

Adaptivity. The second experiment examines the level of adaptivity to world changes. We used a trained agent, which had a training session of 50 connect-4 games (a game with the same rules as connect-5, except that the combination of four is enough for the victory). Then, the agent had to play connect-5 against the strongest opponent, without any notification or adjustment regarding the rule change. Figure 2 shows that in the first 25 games the agent had serious problems using the experience gathered earlier, resulting in a defeat rate of almost 96 percent. Although, after 50 games it could defend with 50 percent accuracy, and after 125 games, it reached almost the same strength level, as in the previous on-line learning test. Tests with other rule change schemes (C-5 to C-4, no competitiveness) brought similar results.

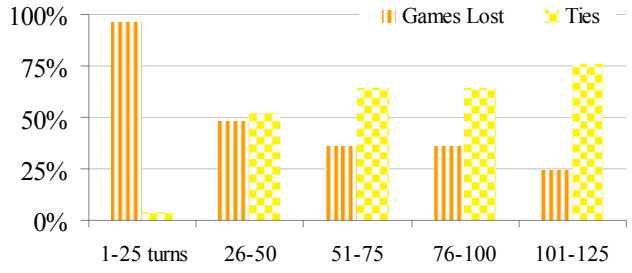

Figure 2. Adaptation to changing rules (Connect-4 to Connect-5).

The ***collective dimension*** was evaluated along various aspects, here we present one. Differently trained agents got knowledge injections, and then, got evaluated off-line. Listed combinations are: empty (untrained) receptor + best trained donor (trained with the strongest opponent), randomly trained receptor + best trained donor, Connect-4 trained receptor + best trained donor, and best trained receptor + random donor. Figure 3 shows that knowledge injection had positive effects in all cases. This is not surpising in the first three cases when the injected knowledge was clrealy more accurate than the agent's own. In the last case, a good agent received "worse" knowledge, and still, this helped it to gain winning which was out of ques-tion beforehand (however, general strength dropped slightly). This effect can be explained with the nonlinearity of the problem space.

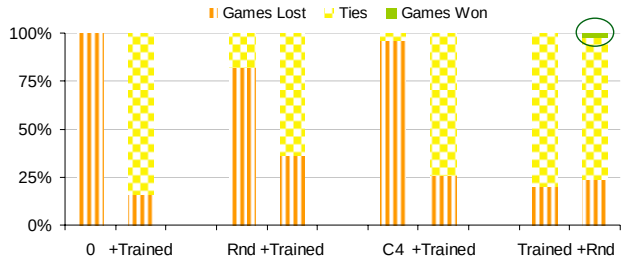

Figure 3. Collective effects (evaluated against the 100% opponent).

VII. SUMMARY

We described an emergent, knowledge-poor and open approach for adaptive systems where adaptation emerges from simple steps during the system's normal operation, even amongst drastically changing environmental rules. Dynamic features and the lack of mandatory and too explicit semantics bring real openness. The cooperative knowledge sharing mechanism brings the adaptation of knowledge-poor learners to a collective level.

REFERENCES

[1] Brun Y. et al., "Engineering self-adaptive systems through feedback loops," LNCS 5525/2009. Springer, Heidelberg, 2009.

[2] Kolter J.Z. and Ng A.Y., "Regularization and feature selection in least-squares temporal difference learning," Proc. of the 26th Annual International Conference on Machine Learning (ICML 09), vol. 382, 2009, pp. 521-528.

[3] Goldsby H.J. et al., "Goal-based modeling of dynamically adaptive system requirements," Proc. of 15th Annual IEEE International Conference on the Engineering of Computer Based Systems (ECBS), 2008.

[4] Benkő B. K., Brgulja N., Höfig E., and Kusber R., "Adaptive services in a distributed environment," Proc. of 8th International Workshop on Applications and Services in Wireless Networks, Kassel, 2008.

[5] Nelly Bencomo N. et al. "Genie: supporting the model driven development of reflective, component-based adaptive systems," Proc. of the 30th International Conference on Software Engineering, Leipzig, 2008, pp 811-814.

[6] Harel D and Marelly R., "Come, let's play: scenario-based programming using LSCs and the play-engine. Springer, Heidelberg, 2005.

[7] Sutton R.S. and Barto A.G., "Reinforcement learning: an introduction," The MIT Press, 1998.

[8] Sutton R. S., "Learning to predict by the methods of temporal differences," Machine Learning 1988/3, 1988, pp. 9-44.

[9] Baxter J. and Weaver A., "TDLeaf(lambda): Combining temporal difference learning with game-tree search," In Proceedings of the 9th Australian Conference on Neural Networks, 1998, pp. 39-43.

[10] Bradtke S.J. and Barto A.G., "Linear least-squares algorithms for temporal difference learning," Machine Learning 1996/22, 1996, pp. 33-57.

 ISBN: 978-1-61208-103-8