SIGEVOlution

newsletter of the ACM Special Interest Group on Genetic and Evolutionary Computation

Winter 2007
Volume 2 Issue 4

in this issue

Editorial

Space Invaders was the first video game I ever played. There were invaders from outer space and I defended the Earth with a small cannon and the little cover provided by a few small buildings. The graphics were plain black and white; yellow, red, green and blue tapes were stuck on the screen to obtain colors. Today, computer games are so advanced that I cannot play them anymore and they have stunning graphics. They are becoming so complex that in the future we may need evolutionary computation to build artificial players that can challenge human players—at least, this is what Steffen Priesterjahn suggests in the first article of this issue. If artificial world and computer games are not your cup of tea, you can compete in the real world against artificially evolved drivers using 1:24 scale cars. In the second article, Ivan Tanev and Katsunori Shimohara show how genetic algorithms can provide you with controllers for scale cars that may challenge human drivers. The issue is completed with the usual columns including the summary of the workshop on the petroleum applications of evolutionary computation held at GECCO-2007, information about the Rubik's cube competition at GECCO-2008, the list of GECCO-2008 best paper nominees, a report on the simulated car racing, Ms PacMan, and Othello competitions at WCCI-2008, the new issues of EC journals, and the calendar of EC events.

The cover photo is a screenshot of Quake III (©1999, id software) taken by Steffen Priesterjahn.

This is the fourth and final issue of 2007! The second volume of SIGEVOlution is now completed and the first issue of the third volume is already a work in progress. I would like to thank the people who made this issue possible, Steffen Priesterjahn, Ivan Tanev, Katsunori Shimohara, Terry Soule, Robert B. Heckendorn, Steven Armentrout, Alexandre Castellini, Charles Guthrie, Burak Yeten, Tina Yu, Daniele Loiacono, Julian Togelius, Simon M. Lucas, Larry Bull, Martin V. Butz, Kumara Sastry, and board members Dave Davis and Martin Pelikan. Without them and without you reading, I would not be here writing.

I hope to see you all in Atlanta during GECCO and if you have suggestions, comments, ideas, or criticisms, please drop an email to editor@sigevolution.org or stop by for a chat at GECCO.

Pier Luca
June 16th, 2008

SIGEVOlution
Winter 2007, Volume 2, Issue 4

Newsletter of the ACM Special Interest Group on Genetic and Evolutionary Computation.

Contributors to this Issue

Steffen Priesterjahn
Ivan Tanev
Katsunori Shimohara

Contents

ISSN: 1931-8499

Imitation-Based Evolution of Artificial Game Players

Steffen Priesterjahn, University of Paderborn, Department of Computer Science, Germany, spriesterjahn@upb.de

Because of the rapid progress of commercial computer games in recent years the development of artificial characters that inhabit the presented game worlds has become a challenging task with very specific requirements. A particular computer game specific requirement is that, as the objective of computer games is the entertainment of the player, the artificial intelligence should not only be competitive but also show intelligent and human-like behaviours. Therefore, the following article proposes the usage of imitation techniques to generate more human-like behaviours in an action game, whereas the imitation is achieved by recording players and by using these recordings as the basis of an evolutionary learning approach.

Introduction

Many modern computer games show complex, highly dynamic virtual worlds with realistic physics. However, the artificial intelligence routines of the characters that inhabit these game worlds - usually called game AI - is often static and relies on pre-written scripts [4]. Scientific research in the area of game AI has therefore concentrated on the deployment of learning methods to create competitive agents - often with very high performing results [9, 10].

However, when creating game AI for a real game one should keep in mind that the primary objective of a game is to entertain the player. Therefore, gaming characters should not be as good as possible or be almost invincible [6]. They should show some sophisticated human-like behaviors that at least looks intelligent [7].

For example in an action game, the agents should not just aggressively try to inflict as much damage as possible. It is much more desirable that they try to use the map structure for taking cover or try to trick their opponents. The ultimate goal of game AI should be to create game characters that are almost indistinguishable from other human players.

The question is how such a behavior can be achieved. As we said above, a pure learning approach that is based on the optimisation of the agents is inappropriate because it usually optimizes the raw performance of the game agents. Instead, we propose that to behave human-like, an agent should base its behaviour on how human players play the game and try to imitate them.

In computer games, human and artificial players meet at the same level. They play in the same virtual world and have the same abilities. Therefore, it is quite simple to record the behaviour of a human player and to use it as a source for imitation. The advantages of imitation are that the imitating player automatically plays at about the same performance level as its imitator and that, if a human player is imitated, the artificial player will automatically show more believable, sophisticated and human-like behaviours.

Imitation involves a role model and an imitator. Technically, imitation can be achieved by first recording how the role model reacts to its encountered situations. Then, this recording can be analysed to generate a controller that shows imitative behaviours. To achieve the most perfect imitation by basically creating a copy of the role model, a supervised learning method can be used to generate a controller, that minimises the imitation error and that maps the current situation or state to the action that fits best to the ones that can be found in the recording.

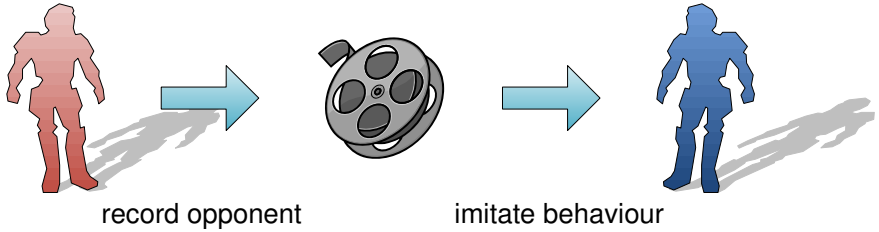


Fig. 1: Imitation scheme

However, such a pure imitation method is often not enough to generate competitive performance. In fact, our experiences from several experiments [12] rather indicates that the performance of the imitator is always significantly lower than the performance of its role model. The reason for that lies in inherent errors that are made in the process of imitation. For example by assuming a certain state and action model that does not necessarily fit to the role model or by choosing a certain agent controller for the imitator. Therefore, we propose the usage of imitation-based learning which uses an optimisation method on top of a representation that is based on recorded player behaviour to obtain competitive and imitating agents.

This article presents an imitation-based approach that uses an evolutionary algorithm as the described optimisation method to successfully train agents for combat in Quake III (©1999, id software) - a popular three-dimensional action game[1]. It is based on an evolutionary learning approach that we have published in 2006 [14]. However, in this approach the evolutionary process is mainly not used to create new knowledge, but to select the right combination of imitated behaviour pieces and to smooth the resulting behaviour. We will show that this approach is able to generate successfully performing as well as imitating agents that show sophisticated behaviours. This article presents a continuation of the raw approach that we have presented in [13] and not only adds refinements to the method itself but also a much more detailed discussion of the results and a proper analysis of the resulting game agents.

[1] The game Quake III offers fast paced and very dynamic multiplayer-based gameplay. As the source code of Quake III was completely published in 2005, it is highly modifiable and was thus chosen for the experiments.

Fig. 2: Quake III

The usage of the imitation of human players has become more and more common in the game AI research field in the most recent years. There, imitation is used as a method to create pure imitators that behave more human-like [1, 15, 5, 16] or as an approach to support a learning method [2, 3, 8, 11] to achieve more believable but also better performing results. Our approach fits best into the latter category, as its primary objective is to create competitive but also believable combat agents. One approach that bears a strong resemblance are the so-called case-injected genetic algorithms from Louis et al. [8], which also use recorded gaming data in a real-time strategy game to improve the learning process. However, our approach is more focused on the actual imitation of the presented behaviour, instead of its utilisation to achieve a higher performance.

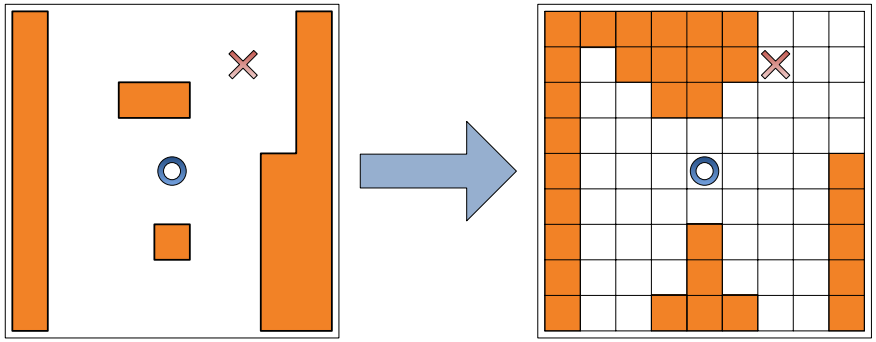


Fig. 3: Grid computation

States & Actions

As this approach is based on our previous evolutionary method [13, 14], the basic modeling resembles this work in most respects. The agents use regular grids for their state representation and use rule lists to encode their behaviour. A grid describes the current game situation as a mapping of the current vicinity of an agent to quadratic areas. The grid is always aligned along the view direction of the observed game character. The grid cells can have three different values - empty, filled and opponent - that are based on their content. The agent is not able to look behind walls. Figure 3 presents an example for the construction of such a grid. The behaviour of an agent is encoded in a rule list that contains rules that map grids to actions. Hence, each rule contains a grid that represents the state in which it should be applied and an action that proposes the next move by specifying if the agent should move left or right, forward or backward and by which degree its view angles should be changed as well as if the agent should attack or not. According to the current situation the best fitting rule of the rule list is determined by computing the Euclidean distance between the currently sensed grid and the grids that are proposed by the rules. For a better determination of the similarity between the grids, all grids are smoothed by a Gaussian filter before they are compared. The basic operating loop of an agent is described in algorithm 1. The loop is executed ten times per second.

Algorithm 1 Agent operating loop

```
1: loop
2:     sense the current state and compute the corresponding grid
3:     find the best fitting rule in the rule list
4:     execute the proposed action
5:     wait until next time frame
6: end loop
```

The performance of an agent in combat is measured by letting it play for a fixed amount of time and by computing the amount of damage that it inflicted on the other game characters minus the damage that it received in this timespan.

As we have stated above, the rule lists of the agents are optimised by using an evolutionary algorithm. However, the optimisation is based on the performance of a whole agent or rule list and not on individual rule utilities like typical reinforcement learning or learning classifier system approaches. The reason for that is the high uncertainty of the game environment. Not only is the outcome of a movement affected by friction and other randomised effects, the behaviour of the in-game opponents is also controlled by a randomised algorithm. We conducted several prior experiments using Q-Learning [12], which were not successful because the volatility of the environment makes it very difficult to compute reliable rule utilities. Therefore, we use a population-based approach in which several agents with individual rulesets are tested in terms of their overall performance. An evolutionary algorithm then discards the low performing agents and replaces them by mutated recombinations of the high performing agents. By doing this, the method has a statistically more stable foundation for its learning process.

Creating the Rule Base

To achieve imitative behaviour, we generate the initial rule lists of the population by recording players. This is simply done by letting them play against each other and by recording their grid-to-command matches for each frame of the game. Each of these matches represents a rule which is then stored in a rule database. We just put the rules into the database without any preprocessing. Thus, rules which are executed more often and, hence, should be more important are put into the rule base more often.

In the first step of training, certain behaviours of the players will be imitated by our agents. Then, the selection of the appropriate rules from the rule base and the performance of the agents is optimised by the evolutionary algorithm. This approach has the advantage that certain behaviours can be presented to the agent, from which it learns to use the best in relation to its fitness function. In this way an agent can be trained to show a certain behaviour without programming it manually and still be competitive.

The Evolutionary Algorithm

In most aspects, the underlying evolutionary algorithm presents a very straightforward approach. The individuals are game agents whose behaviour is encoded in the above described rule lists. In each generation each agent is evaluated for a certain timespan. Then selection, recombination and mutation is applied to generate new rule lists out of the best performing ones.

The evolutionary algorithm is used to find the most important rules in the rule base and to put them together in a fitting list. Therefore, the recombination operator plays an important role. However, as we already explained above, there is still the need for optimisation and fine tuning to generate competitive agents. Therefore, we still use mutation to adapt the rules to receive the desired gaming performance. In the following we will present the used selection and variation operators.

Parent Selection. Concerning the population structure and the selection scheme, we use a $(\mu+\lambda)$ evolutionary algorithm. The size of the parental population is $\mu \in \mathbb{N}$. In each generation $\lambda \in \mathbb{N}$ offspring individuals are produced by applying the variation operators recombination and mutation. In contrast to the comma selection scheme, the plus selection scheme lets the parents survive and be a part of the new generation. Therefore, the population size is always $\mu+\lambda$. The survivor selection itself just selects the μ best rule lists according to their fitness. We do not use fitness-proportional selection to achieve a better exploitation. The parents are kept in the population for several reasons. First, the evolutionary process is stabilised by keeping the good solutions. As our variation operators - especially recombination - apply very strong changes to achieve better exploration, this balances the learning process. Second, it helps to reduce the effects of a volatile fitness function. The performance of an agent can be affected by several incidents. For example, the agent or the opponent could have made a lucky shot or have made a very bad decision that got them into a corner. Therefore, the agents are reevaluated in each generation. To stay in the population they have to prove their value again and again. This results in more generalised behaviours and the surviving agents are better equipped to handle unseen situations.

Survivor Selection. From the survivors, the parents are selected randomly with uniform distribution.

Recombination. For the recombination, two parents are chosen randomly with uniform distribution from the parental population. Let $(R_1,...,R_k) \in \mathcal{R}^k$ and $(R'_1,...,R'_k) \in \mathcal{R}^k$ be the rule lists of the parents. Then, the rule list of an offspring $(O_1,...,O_k) \in \mathcal{R}^k$ is created by randomly choosing each rule O_i from $\{R_i, R'_i\}$ with uniform distribution. Hence, recombination affects the structure of the rule lists. The operator resembles uniform crossover. We chose this operator in contrast to a one-point crossover to increase the variety of the produced offspring.

Mutation. In contrast to crossover, the mutation operator effects the structure of the rules itself. In contrast to the evolution from scratch [14] the mutation operator is changed so that it only affects the command but not the grid of a rule. We assume that a recorded rule base that is large enough already contains all important game states. There is no need to create new ones. Furthermore, if the grids are not mutated the resulting rules remain readable over the course of the evolution. Thus, making it possible to easily identify the situation that is represented by a grid by simply looking at it. All changes that are introduced by mutation are made with the same probability π. For the forward and lateral movement as well as the attack value the old value is replaced by a random new value. The view angle change is mutated by adding a small random angle. We use a Gaussian distribution with mean zero and standard deviation α to realise this.

Evaluation. The fitness of each agent is evaluated by letting it play and by applying its list of rules for a fixed simulation period. As we already stated above, the cumulative damage that was applied to the opponents and received by the agent are counted and integrated into the fitness function

$$f = \eta \cdot \text{applied damage} - (1-\eta) \cdot \text{received damage} \quad (\eta \in [0,1]).$$

Applied damage increases and taken damage decreases the fitness of the agent. The weight η determines the influence of each value. We call η the aggressiveness value because it determines the aggressiveness of the agent. If η equals 0.5, attack and defense will be considered in balance. If η is smaller than 0.5, the defense will be emphasised. Finally, if η is larger than 0.5, the fitness will be more strongly affected by the attack capability of the agents.

In preliminary experiments we noticed that a fitness calculation by $f =$ applied damage $-$ received damage (with $\eta = 0.5$) could lead to an undesirable gaming behaviour. In some experiments the agents learned to run away from the opponent and got stuck in this behaviour. Therefore, running away seems to be a local optimum. It minimises the own health loss. Once caught in this behaviour, it is not easy to learn that the fitness can be even further increased, if the opponent is attacked. As the agent will make itself more vulnerable, if it moves into an attack position, changing the behaviour would first result in a deterioration of the fitness.

However, when we chose higher aggressiveness values, like η = 2/3, we created agents that tended to behave almost suicidal. Therefore, we introduced a dynamic fitness calculation. At the beginning, we start with a rather high value for η. After each generation, a discount rate $q \in]0,1[$ is applied to η until it reaches 0.5 This means that η is multiplied by q after each generation to determine the new η value.

To distinguish between the fitness of an agent and its actual gaming result, we distinguish between the *fitness* and the *performance* of an agent, where the performance is simply computed by

$$\text{performance} = \text{applied damage} - \text{received damage}.$$

Experimental Setup

For the setup of the experiments to evaluate the approach, we could rely on our previous results regarding the evolution of game players from scratch [13, 14]. However, though these results helped with the choosing of well performing parameters, the different initialisation of the algorithm made it necessary to examine the results of different grid and rule list sizes again. Most of the other parameters were chosen according to the former experiments. Table 1 shows these parameters. We again started with an aggressiveness value of $\eta = \frac{2}{3}$ to avoid the generation of fleeing agents. The aggressiveness discount rate q was again chosen so that $\eta = 0.5$ is reached after 30 generations.

parameter	value
population size $\mu+\lambda$	60
number of selected parents μ	10
number of generated offspring λ	50
yaw angle mutation range α	5°
evaluation timespan	60 seconds per agent (1 hour per generation)
aggressiveness η	starts at $\frac{2}{3}$
aggressiveness discount rate q	0.99
termination	after 3 days (72 generations)
runs per experiment	20
rule base size	4000 rules (ca. 6:40 min of gameplay)

Tab. 1: Parameter setup

Table 2 shows the final setup of our experiments. We added further experiments which use grid mutation and different mutation rates to see if the new mutation operator has an effect on the gained performance and if it reacts differently to changes to the mutation rate. The experiments were run by playing against the built-in Quake III agent[2] on its default difficulty setting. The Quake III agent plays at a constant performance level and can therefore be used as a benchmark. However, in the past we have also made experiments on using coevolution that also led to successful though slower results [14]. For a better judgment of the learned behaviour we also chose the Quake III agent as the role model to see if the imitators are able to compete with their role models. Furthermore, the Quake III agents have a very distinctive behaviour that helps to judge the quality of the shown imitation and to see if some new behaviours have been generated. We grouped the experiments in several sets, whereas each set examines the influence of one parameter. All sets were based on one single base experiment (the underlined one), whereas all other experiments in each set provided derivations of the base experiment in one parameter.

[2] As a matter of fact we chose the final and hardest opponent of the game "Xaero" as the opponent.

#	grid size	rule list size	mutation rate	grid mutation
1.1	**11 × 11**	100	0.01	no
1.2	**15 × 15**	100	0.01	no
1.3	**21 × 21**	100	0.01	no
2.1	15 × 15	**10**	0.01	no
2.2	15 × 15	**50**	0.01	no
2.3	15 × 15	**100**	0.01	no
2.4	15 × 15	**400**	0.01	no
3.1	15 × 15	100	**0.01**	no
3.2	15 × 15	100	**0.1**	no
4.1	15 × 15	100	0.01	**no**
4.2	15 × 15	100	0.01	**yes**
4.3	15 × 15	100	**0.1**	**yes**

(base experiment 1.2 = 2.3 = 3.1 = 4.1)

Tab. 2: Experimental Setup

With the new initialisation, the foundation of the learning process had changed. Therefore we again examined the influence of the grid size in set 1 to see if it has a different effect on the performance of the agents. Without imitation, a grid size of 15×15 provided the best results. Therefore, we used it in the base setup. The grid cell size was changed according to the changes to the grid size so that the area the agent sees stays at about 15 by 15 metres. Because of the new initialisation, we also reexamined the influence of the rule list size and the mutation rate. In set 2, the size of the rule list was varied to see if more or less rules as in the imitation-less approach are needed. According to the best setup of the former approach, the base setup used a rule list size of 100 rules. Set 3 varied the mutation rate. The base setup used a mutation rate of 0.01, which differs from the 0.1 that we used in the imitation-less approach [14]. However, the imitation-based approach is already initialised at a search space location that provides rules for effective gaming behaviour. Therefore, less exploration and more exploitation is needed to find the best rules from the rule base.

As we already explained above, basing the approach on recorded rules makes it possible and also reasonable to only mutate commands but not grids. To find out, if omitting grid mutation does not handicap the learning process, set 4 consisted of experiments that used and did not use grid mutation. In this set we also used grid mutation with different mutation rates to detect the influence of that parameter in this case.

Results

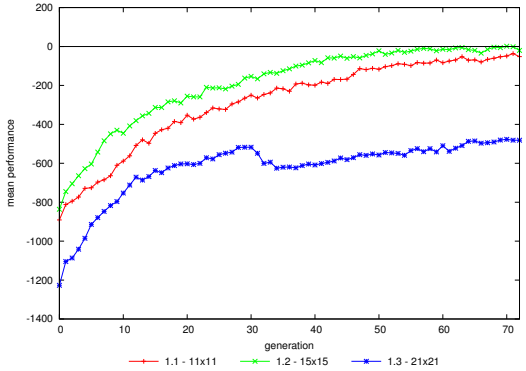
As the overview of the maximum performance in each generation of all experiments in Figure 4e shows, the imitation-based approach is able to successfully create agents that outperform their opponents. They do this by using their own strategies against them and by improving upon these strategies.

Because of the extensive setup of the experiments we obtained several results. To give a clearer presentation of the results we will only show the plots which we find particularly interesting. In the following we will present mainly figures that show the mean performance of the respective experiments because they allow to draw more statistically valid conclusions. In addition, our results indicate that the mean and the maximum performance are correlated.

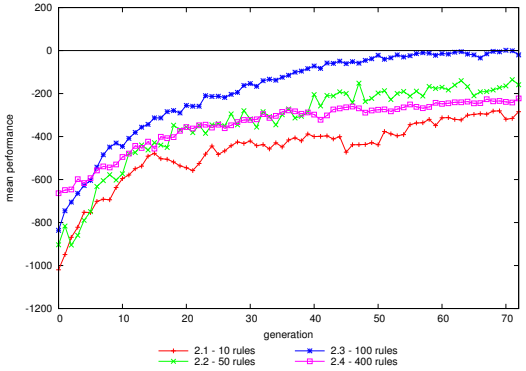
The striking result of the experiments is that the imitation-based initialisation has a strong effect on the performance and the behaviour of the evolved agents. The reached maximum performance (see Figure 4e) is considerably lower than the results of the pure evolution[3] in [14]. Therefore, the evolution of competitive behaviour when starting from an imitation rule base seems to be a harder problem. However, it should be expected that the performance of an imitating agent is closer to the level of its role model.

Concerning the influence of the parameters, one result is that we can only detect a significant influence of the grid size in the case that it was set to 21×21 (see Figure 4a). The experiments using 11×11 and 15×15 grids provided a similar performance. This indicates that a grid size of 11×11 is still sufficient to generate competitive behaviour. Of course, significantly different results could be obtained by setting the grid size to more extreme values (e.g., 1×1 or 100×100). Using a grid of 21×21 cells decreased the performance of the agents significantly. This result

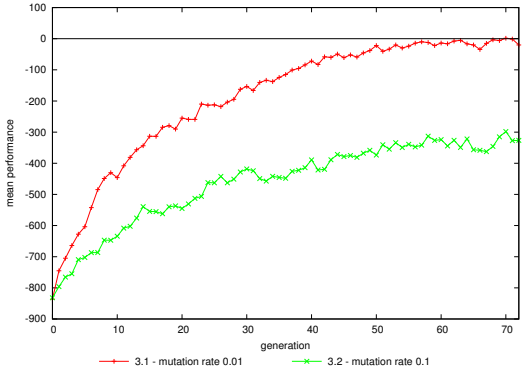
[3] The best agents reached a performance of above 2500



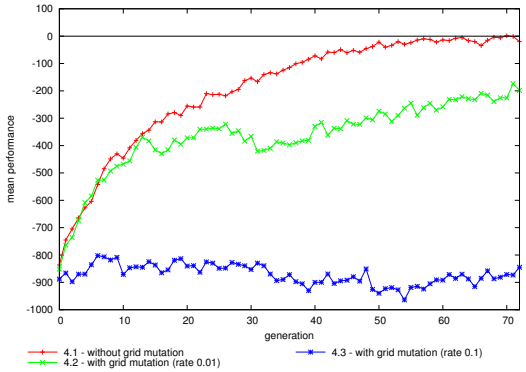
(a) Set 1 (variation of the grid size)



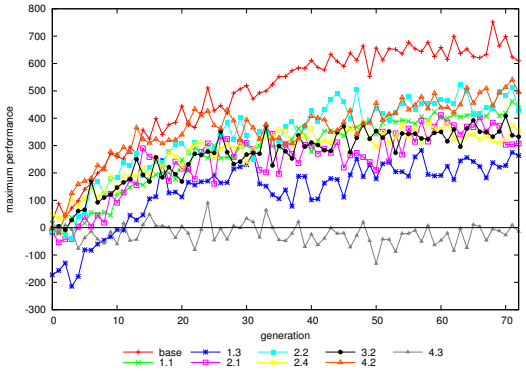
(b) Set 2 (variation of the grid size)



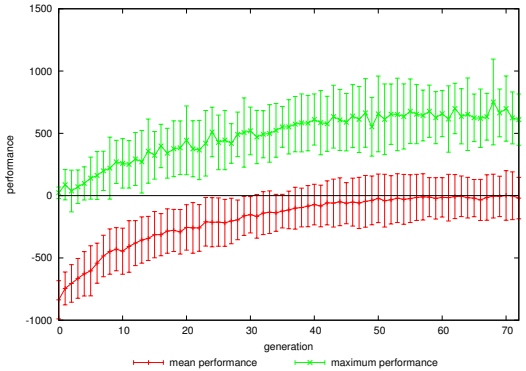
(c) Set 3 (variation of the grid size)



(d) Set 4 (variation of the grid size)



(e) Maximum performance of all sets



(f) Results of the best setup with standard deviation

Fig. 4: Mean performance of all sets

is the same as in the imitation-less approach. If the grid size is too big, the agents can differ more states which leads to a larger search space. In addition, the computation of the distances between the current situation and all grids in the rule list becomes more time consuming and increases the reaction time of the agent.

However, when watching the gameplay of the respective agents, it can be seen that the level of imitation and sophistication of the shown behaviour is higher with the more detailed grids. As a higher grids size leads to more distinguishable states, it also makes it possible to encode more complex behaviour. Therefore, the grid size has to be chosen reasonably big, but not too big.

It should be noted that the setup of set 1 can not be seen as completely fair because each experiment started with a different rule base of recorded rules with the respective grid size. Though we did our best to achieve a high similarity between the recorded rule bases by generating them under the completely same conditions and by making them reasonably big, we can not guarantee that there might exist a small difference in their quality.

Figure 4b shows the mean performances of the experiments from set 2, which examines the influence of the rule list size. The variation of the rule list size has a significant effect on the performance. As in the random-based experiments a rule list size of 10 is too small to perform well. This has several reasons. First, 10 rules are simply not enough to encode a diverse gaming behaviour as it is provided by the rule base. In the imitation-based case more rules are needed to encode the mimicking behaviours. Second, the number of rules in the first generation is considerably lower and less diverse as with a higher rule list size. Therefore, many of the experiments with a rule list size of 10 never produced a well playing agent or, in contrast to that, some of the experiments even converged to rather well performing agents that resembled the purely evolved agents and did not show imitative behaviour.

The results also show that increasing the rule list size results in a higher performance until a certain threshold is reached. If the rule list size is too big, the search space is enlarged and the agents simply need too much time to go through the rule list.

Figure 4c shows the influence of the mutation rate. Using a mutation rate of 0.1 significantly diminished the reached performance. The imitation-based approach does not need much mutation to work well. It mainly uses recombination to find out the best mix of rules. Mutation is only needed to make slight adjustments, to create more fluent and successful behaviours. If the mutation rate is too big, the learning process starts to make bigger steps in the search space and to move away from the imitation-based behaviours.

As depicted in Figure 4d, using grid mutation led to a more chaotic learning process and resulted in a lower performance. In addition, the structures in the grids that resemble real map structures were destroyed. When grid mutation with a mutation rate of 0.1 was used, the approach even failed to create valid agents at all. This is very surprising as this setup exactly resembled the best performing setup in the same approach without imitation [14].

To provide a better basis for the judgment of the significance of the above statements, Figure 4f provides the mean and maximum performance of the base experiment with the respective standard deviations. The base setup also provided the best results in terms of mean and maximum performance.

Concerning the gaming behaviour of the agents, the result is that they very closely imitated the Quake III agents.[4] In the first generations the right combination of rules had to be sorted out and the agents behaved quite randomly. Though, they already showed a much more valid gaming behaviour than a randomly initialised agent. Then - beginning with approximately the fifth generation - the agents started to closely mirror the Quake III agent in its movements. Later, in the course of the evolution, the agents took more and more freedom in their movements. For example, some agents started to take cover behind the column while their weapon reloaded. This behaviour was not present in the rule base and represents a level of sophistication in the learned behaviour that was not shown by the pure learning approach.

We also conducted several experiments to check if the approach is able to imitate other players. To do this we created a rule base which contained the behaviour of a human player. The results were also satisfying and showed imitative behaviour. Though it was difficult to evaluate the quality of imitation, it could be clearly seen that the agents copied behaviours which were performed by the human players.

[4] See https://chaos.cs.upb.de/imitation.avi for a demonstration.

Analysis

To further investigate what the learning method has produced, we made a statistical analysis of the well performing agents. In addition to the examination of the rule selection probabilities and its standard deviation, we were especially interested in second order statistics. The following definition presents a so-called co-occurrence matrix which holds the probabilities that after rule R_i another rule R_j is executed. This should not be confused with the transition probabilities of a Markov decision process which is a conditional probability. The sum of *all elements* in the co-occurrence matrix is one, whereas the sum of *each line* of the transition probability matrix is one.

Definition 1 (Co-occurrence Matrix)
Let $(R_1,...,R_n) \in \mathcal{R}^n$ *be a rule list of an agent. Then, the* co-occurrence Matrix C *is defined as* $C = (p_{i,j})_{1 \leq i,j \leq n}$, *where*

$$c_{i,j} = Pr(R_{t-1} = R_i \wedge R_t = R_j)$$

denotes the probability that rule R_j *is executed directly after the execution of* R_i.

Given a co-occurrence matrix the transitivity and the reflexivity of the used rule list can be computed according to the following definition.

Definition 2 (Reflexivity ρ, Transitivity τ)
For a given co-occurrence matrix $C = (p_{i,j})_{1 \leq i,j \leq n}$, *the value* $\rho \in [0,1]$ *with*

$$\rho = \sum_{i=1}^{n} p_{i,i}$$

is called the reflexivity of C. *The value* $\tau \in [0,1]$ *with*

$$\tau = 1 - \rho$$

is called the transitivity of C.

Table 3 shows the standard deviation σ for choosing a rule as well as the reflexivity ρ and the transitivity τ of the best performing randomly initialised agents from [14] and the best performing imitation-based agents. Both are typical for the results that were obtained by the respective methods. Interestingly, the values from the imitation-based rule list are very similar to the other ones, except the standard deviation. This indicates that there is a similar structure in the imitation-based rule list but the work is distributed onto a higher number of important rules.

agent	standard deviation σ	reflexivity ρ	transitivity τ
random-based	0.34	28%	72%
imitation-based	0.06	31%	69%

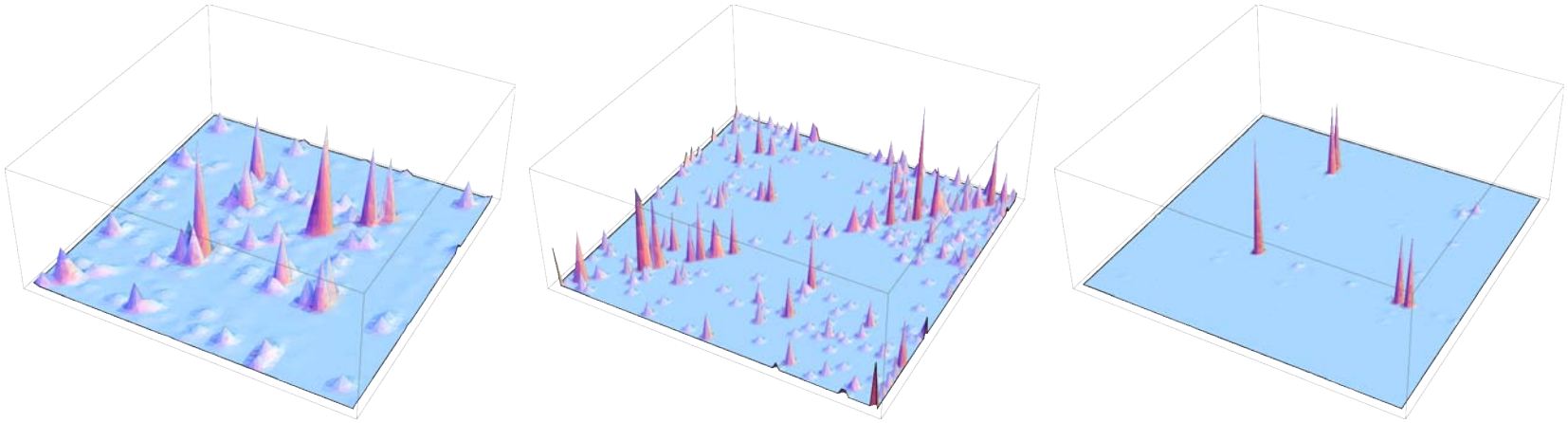
Tab. 3: Statistical analysis

Figures 5a and 5b show the co-occurrence matrices of two of the best agents which we produced by the imitation-based approach. There is a significant difference to the matrices of the agents that were produced by pure evolution (Figure 5c), as the evaluation of the standard deviation already indicated above. Much more rules are used and there exists a bunch of special rules for special events and behaviours which enable the agents to show more sophisticated and human-like behaviours.

To further examine the differences between the resulting rule lists of both methods, Figure 6 shows the most important rules of the best performing agents from the random and imitation-based experiments. The value of a rule was computed by detecting the damage that was applied and taken while the respective rule was executed. The random-based rule clearly shows that the surrounding map structure does not have a high influence on the state. The cells are rather randomly empty or filled. This indicates that the random-based agents usually base their actions on the position of the opponent. The benefit of the imitation-based initialisation is that the rule base automatically consists of states that already take the map structure into account. Therefore, the decision to restrict the mutation operator to mutating the commands but not the grids is important for the generation of more sophisticated behaviours.

Conclusion

In the experiments, our agents were able to behave in the same way as the original players already after few generations. They were also able to improve their performance beyond their basis and to develop new behaviours. Therefore, the presented system can be used to train certain aspects of the behaviour of an artificial opponent based on the imitation of other players and to emphasise desired behaviours. Our approach has also turned out to prevent disadvantageous behaviours, because they impair the fitness of the agent. Such behaviours, e.g. getting stuck in



(a) Imitation-based agent, 50 rules (b) Imitation-based agent, 100 rules (c) Purely evolved agent, 100 rules

Fig. 5: Co-occurrence matrices

corners or standing still, have been eliminated in all experiments after at most 20 to 30 generations. The generated agents, though having a lower performance, showed a much higher level of sophistication in their behaviour and appeared much more human-like as the agents that were generated by using plain evolution. However, it should be noted that the presented approach is only able to base its results on the imitation of the respective role model but not to fully imitate it because of the unsupervised nature of the method. In addition, the method can not be applied to an online scenario in an ongoing game because it often generates defective agents. To achieve this more cautious variation operators would be needed, as we applied in [14].

Bibliography

[1] C. Bauckhage, C. Thurau, and G. Sagerer. Learning Human-like Opponent Behaviour for Interactive Computer Games. In *Pattern Recognition*, Lecturenotes in Computer Science 2781, pages 148–155. Springer, 2003.

[2] B. D. Bryant. *Evolving Visibly Intelligent Behavior for Embedded Game Agents*. PhD thesis, Department of Computer Sciences, The University of Texas, Austin, TX, USA, 2006.

[3] B. D. Bryant and R. Miikkulainen. Acquiring Visibly Intelligent Behavior with Example-Guided Neuroevolution. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pages 801–808, 2007.

[4] M. Buckland. *Programming Game AI by Example*. Wordware Publishing, 2005.

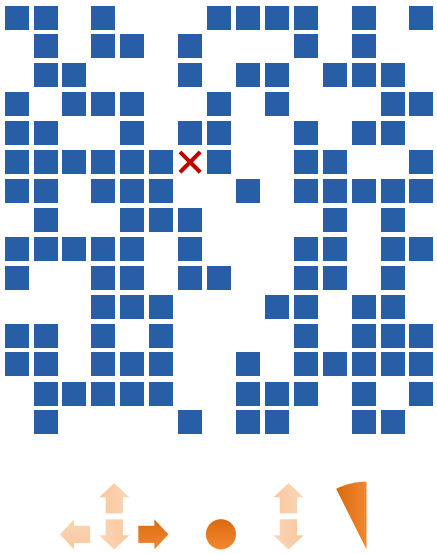
[5] B. Gorman, C. Thurau, C. Bauckhage, and M. Humphrys. Bayesian Imitation of Human Behavior in Interactive Computer Games. In *Proceedings of the International Conference on Pattern Recognition (ICPR'06)*, volume 1, pages 1244–1247. IEEE Press, 2006.

[6] L. Lidén. Artificial Stupidity: The Art of Making Intentional Mistakes. *AI Game Programming Wisdom*, 2:41–48, 2004.

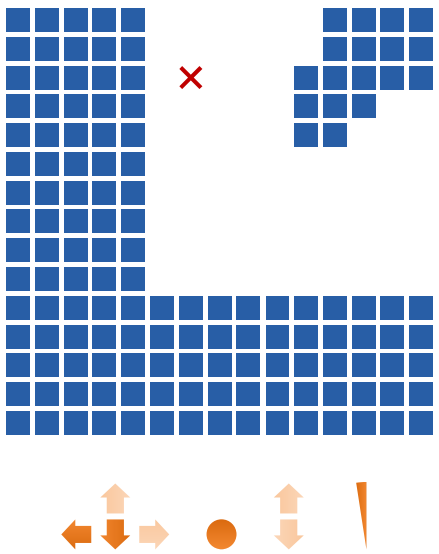
[7] D. Livingstone. Turing's Test and Believable AI in Games. *Computers in Entertainment*, 4(1):6, 2006.

[8] S. J. Louis and C. Miles. Learning to Play Like a Human: Case-Injected Genetic Algorithms for Strategic Computer Gaming. In *Proceedings of the 2nd Workshop on Military and Security Applications of Evolutionary Computation*, pages 6–12. IEEE Press, 2006.

[9] S. M. Lucas and G. Kendall. Evolutionary Computation and Games. *IEEE Computational Intelligence Magazine*, 1:10–18, 2006.



(a) The best rule of the best random-based agent [14]



(b) The best rule of the best imitation-based agent

Fig. 6: Best rules of the best agents of both approaches

[10] R. Miikkulainen, B. D. Bryant, R. Cornelius, I. V. Karpov, K. O. Stanley, and C. H. Yong. Computational Intelligence in Games. *Computational Intelligence: Principles and Practice*, pages 155–191, 2006.

[11] C. Miles and S. Louis. Case-Injection Improves Response Time for a Real-Time Strategy Game. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG'05)*, pages 149–156. IEEE Press, 2005.

[12] S. Priesterjahn. *Online Adaptation and Imitation in Modern Computer Games*. PhD thesis, University of Paderborn, 2008.

[13] S. Priesterjahn, O. Kramer, A. Weimer, and A. Goebels. Evolution of Reactive Rules in Multi-Player Computer Games Based on Imitation. In *Proceedings of the International Conference on Natural Computation (ICNC'06)*, volume 2, pages 744–755. Springer, 2005.

[14] S. Priesterjahn, O. Kramer, A. Weimer, and A. Goebels. Evolution of Human-Competitive Agents in Modern Computer Games. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'06)*, pages 777–784. IEEE Press, 2006.

[15] C. Thurau, C. Bauckhage, and G. Sagerer. Imitation Learning at All Levels of Game-AI. In *Proceedings of the International Conference on Computer Games, Artificial Intelligence, Design and Education*, pages 402–408, 2004.

[16] J. Togelius, R. D. Nardi, and S. M. Lucas. Towards Automatic Personalised Content Creation for Racing Games. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG'07)*, 2007.

About the author

Steffen Priesterjahn was born in 1979 in Blankenburg, Germany. He began to study computer science with minor subject mathematics at the Clausthal University of Technology (TU Clausthal) in 1998. In 2003 he graduated and joined the University of Paderborn as a Ph.D. student. As a part of the research group on Knowledge-Based Systems, under Prof. Dr. Hans Kleine Büning he has just recently received his Ph.D. in natural sciences for his thesis "Online Imitation and Adaptation in Modern Computer Games". His main research interest is the development of believable and competitive artificial players for modern computer games that are able to adapt in real time. To achieve this he has developed adaptation methods that are based on the imitation of other players and that use evolution and social learning to ground the learning process on the exploitation of the experiences of a population of game-playing agents.

Homepage: http://www.upb.de/cs/ag-klbue/en/staff/spriesterjahn
Email: spriesterjahn@upb.de

Towards Human Competitive Driving of Scale Model of a Car

Ivan Tanev, Doshisha University, Japan, itanev@mail.doshisha.ac.jp
Katsunori Shimohara, Doshisha University, Japan, kshimoha@mail.doshisha.ac.jp

The success of the computer playing board games (e.g., chess [8], checkers [4], backgammon, tic-tac-toe, etc.) has long served as an indication of the progress in the field of artificial intelligence (AI). The expanding scope of applicability of AI, when the latter is employed to control the individual characters (agents) which are able to "learn" the environment, often including opponents, and to adopt an adaptive optimal (rather than a priori preprogrammed) playing tactics and strategy include soccer [15], motocross and car racing [3, 21], etc. [4], [6]. In this article, we focus on the domain of car racing, and consider the problem of designing a driving agent, able to remotely control a scale model of a racing car, which runs in a human-competitive, fast and consistent way. Adhering to the commonly recognized criteria of the human competitiveness of automatically evolved solutions, we attempt to verify that the evolved driving agent "holds its own or wins a regulated competition involving human contestants" [11].

Our work is motivated by the opportunity to develop an agent, able to address some of the challenges faced by a human racer. First of all, in order to provide fastest laps times around the circuit, the driver needs to define the best driving (racing) line, or the way the car enters, crosses the apex, and exits the turns of the circuit. Moreover, in order to realize the once defined optimal line, the driver has to make a precise judgment about the current state (i.e., position, orientation and velocity) of the car and the environment, and it has to react timely and precisely.

The aim of our work is the automated evolutionary design of the functionality of a driving agent, able to remotely operate a scale model of racing car (hereafter referred to as "car") in a human-competitive, fast and consistent way around predefined circuits. An agent with such capabilities would open up an opportunity to build a framework of adaptive racing games in which a human competes against a computer with a matching capabilities, with both of them remotely operating scale models, rather than software modeled cars. The proposed evolutionary approach could be also applied for automated design of the control software of remotely operated vehicles capable to find an optimal solution to various tasks in different environmental situations. To achieve our objective, two tasks should be solved:

- The automated determination of the best driving style. In order to solve this task, we have (i) to formalize the driving style and to define the parameters that describe it; and (ii) to employ an algorithm paradigm for automated determination of the fastest driving style by setting its parameters to their optimal values.
- The adequate control of the car enabling it to realize the defined driving style. This implies that the driving agent should be able to control the fast moving car via closed control loop with finite feedback latency.

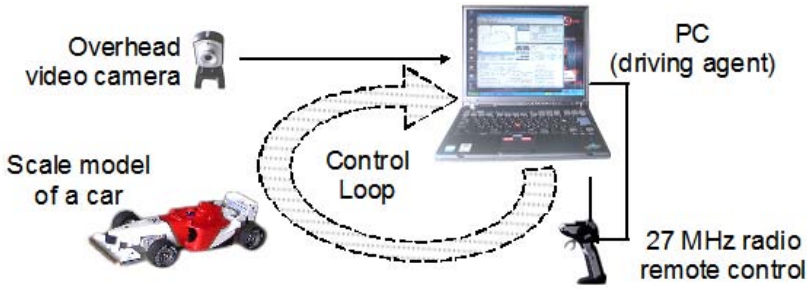


Fig. 1: System configuration

In the related work, Suzuki and Floreano [17] have demonstrated the feasibility of active vision for landmark navigation of a scaled vehicle. Wloch and Bentley [21] applied genetic algorithms for automated optimization of the setup of the simulated racing car. However, neither the adaptation of the driving style to the setup of the car (e.g., co-evolution of the driving style and the setup) nor the use of a physical (scale) model of a car was considered. Togelius and Lucas [18] used scale models of cars in their research to demonstrate the ability of the artificial evolution to develop optimal neurocontrollers with various architectures. However, neither [21] nor [18] considered the human competitiveness of the obtained results. In our previous work [19], we used an evolutionary approach to optimize the controller of a scale model of a car and to automatically optimize the avoidance of a priori known, immobile obstacle [20]. Although we did discuss the feedback latency and proposed a way to alleviate its detrimental effect on the drivability of the car, we did not consider the implications of the proposed approach on the human competitiveness of the evolved driving agent.

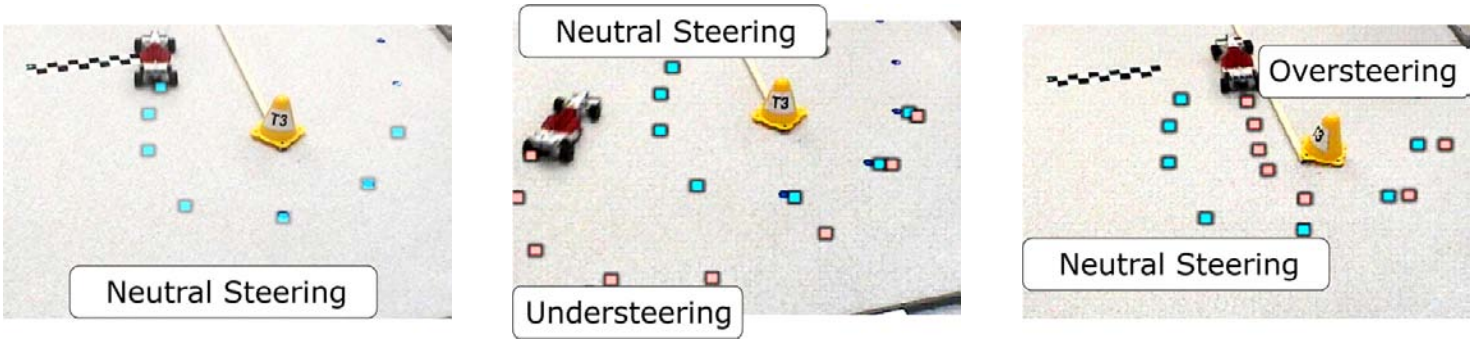
System Configuration

The Car

In our work, we used a 1:24 scaled model of an F1 racing car, with the bodywork repainted red for more reliable image tracking, as shown in Figure 1. This off-the-shelf car features a simple two-channel radio remote control (RC) with functionality including "forward", "reverse", and "neutral" throttle control commands and "left", "right" and "straight" steering controls. The car has the three very interesting features: (i) a wide steering angularity, (ii) a spring suspension system in both front and rear wheels, and (iii) a differential drive. The first feature implies a small turning radius, and consequently, high maneuverability of the car. The torsion spring of the rear suspension of the car functions as an elastic buffer, which absorbs the shocks, caused by the sharp alterations in the torque generated by the car's rear wheel drive motor. These torque alterations occur during the pulse-width modulation (PWM) of the throttle, by means of which the driving agent regulates the speed of the car within the range from zero to the maximum possible value. In addition, torque alterations occur when the "reverse" throttle command is applied for braking of the car that still runs forward. The absorption of these shocks is relevant for the smooth transfer of the torque from the motor to the driving wheels of the car without an excessive wheelspin, achieving a good traction both under braking and acceleration. Moreover, the absorptions of the shocks caused by the frequent torque alterations are important for the longevity of the gear transmission of the car. The third feature, the differential rear wheels drive.implies that the torque of the motor is split and delivered to the rear wheels in a way that allows them to rotate at different angular speeds when necessary, e.g., under cornering. Therefore, the car turns without a rear wheels spin, which results in a smooth entrance into the turns and a good traction at their exits. The main mechanical characteristics of the car are elaborated in detail in [19].

Handling Attitudes of the Car on Cornering

The tires of the turning car, operated at, and beyond the limits of the friction (grip, adhesion) forces, slide to some degree across the intended direction of traveling. The dynamic weight redistribution causes the grip levels at the front and rear wheels to vary as the turning car accelerates on "forward" or decelerates on either "neutral" or "reverse" throttle commands [7]. This, in turn, yields different sliding angles for the front and rear wheels, causing the car that turns too fast to feature either a neutral steering (the slide angles of both axles assume the same values, and the car turns with a nominal or slightly smaller turning radius, as shown in Figure 2a), an understeer (the slide angles of the front wheels are greater than those of the rears — the car turns with a wider radius, as shown in Figure 2b) or oversteer (slide angle of the front wheels are narrower than that of the rear ones — the car turns with a narrower turning radius, as depicted in Figure 2c).



(a) (b) (c)

Fig. 2: The three possible handling attitudes of a sliding car on cornering: (a) neutral steering – the sliding angles of the wheels of both the front and rear axle are nearly the same; (b) understeer – the sliding angles of the front wheels are greater than those of the rear wheels; (c) oversteer – the sliding angles of the front wheels are narrower than those of the rear.

In addition to the degradation of the maneuverability of the car, the sliding of the wheels results in a significant braking forces which, in turn, reduce the velocity of the car. Moreover, the increased actual turning radius due to sliding of the understeering car means that the car might enter the run-off areas of the track or even hit the guardrails on tight corners of the track, which, in turn, might result either in a damage of the car, lost of momentum, or both. Therefore, the sliding of the understeering car limits the average velocity on cornering car (due to the lower than intended speeds along longer than intended arcs), which may have a detrimental effect on the overall lap times. On the other hand, the vector sum of the tangential braking and radial centrifugal forces applied to the rear wheels of a turning car under braking (e.g., when a "reverse" throttle command is applied on cornering) may exceed the reduced (due to the weight transfer) grip limits of the rear wheels, causing the car to oversteer. Depending on the severity and duration of oversteer, the car might either turn into the corner smoothly with slightly lower turning radius, or spin out of control. Conversely to the understeer, the oversteer is usually self sustained and therefore, difficult to control. Indeed, the excessive sliding of the rear wheels causes a significant braking momentum which, due to the weight transfer effect, keeps the rear axle of the car lightweight, resulting in a constantly low grip of the rear tires. Moreover, the sliding of the rear tires reduces the actual turning radius of the car, which in turn may result in sustained, significant centrifugal forces despite the decrease of the speed of the oversteering car. The complexity of the effects of the various handling attitudes of the car on the lap time makes the task of optimizing the driving style of the agent even more challenging, which additionally motivated us to consider an automated heuristic approach to address it.

Perceptions and Action of the Driving Agent

The only perceptions of the agent are obtained from the overhead-mounted video camera. The camera features a CCD sensor and a lens with a wide field of view (66 degrees) which covers an area of about 2800mm x 2100mm from an altitude of about 2200mm. The camera operates at 320x240 pixels mode with a video sampling interval of 30ms. The camera is connected to the notebook personal computer (PC) through a PCMCIA-type video capture board.

The agent's actions (a series of steering and throttle commands) are conveyed to the car via a standard two-channel radio control transmitter operating in 27MHz band. The four mechanical buttons (two buttons per channel) of the transmitter are electronically bypassed by the transistor switches activated by the controlling software. Transistors are mounted on a small board, connected to the parallel port of the computer.

Following Sample Routes

In order to verify the very basic concepts of applying the agency for remote operation of the car, we devised experiments with the car following sample routes marked by apexes of the turns. Analogous to the real-world vehicle navigation using GPS waypoints, we assume that the driving agent is a priori aware of the positions of these apexes. They are conveniently set-up via graphical drag-and-drop user interface, which facilitates their augmentation into the scene and uploading of their coordinates into the agent's memory. The three routes are (i) an O-shaped circuit featuring two right, single-apex turns, (ii) 8-shaped circuit with a right and a left, double-apex turns, and (iii) S-shaped circuit with a series of right and left turns.

At each time step, the agent receives the live video feed from the camera, tracks the car, calculates the current state (i.e., position, orientation and speed) of the car and depending on the values of these parameters issues a series of corresponding throttle- and steering controlling commands. The controlling commands correspond to the very basic, hand-crafted functionality needed to follow the route by homing to the apexes of the turns at 20 degrees with a relatively slow speed of about 800mm/s (scaled speed of about 70km/h). The speed- and angular differences (errors) between the desired and actual speed and homing angle result in a corresponding throttle and steering commands, respectively. The resulting trajectories, indicated by the traces of the perceived geometrical center of the car on O-, 8-, and S-shaped circuits are shown in Figure 3. As Figure 3 illustrates, the actual lines (shown in Figures 3a, 3b and 3c) differ significantly from the expected ones (Figures 3d, 3e and 3f). In the next section we (i) elaborate on the problem causing the discrepancy between the expected driving lines and the really observed ones and (ii) discuss the approach we propose to address it.

Anticipatory Modeling

Outdated Perceptions

The delays introduced in the feedback control loop (as depicted in Figure 1) by the latency of the video feed imply that the actions (i.e., steering and throttle commands to the car) of the driving agent are based on outdated perceptions, and consequently, outdated knowledge about the state of the car (position, orientation and velocity) and the surrounding environment (bearing of- and distance to the apexes). For the hardware used in our system, the aggregated latency is about 90ms, which results in a maximum error of perceiving the position of the car of about 180mm (scaled error of 4.3m) when the later runs at its maximum speed of 2000mm/s (scaled speed of 172km/h). The latency also causes an error in perceiving the orientation (bearing) and the speed of the car. The cumulative effect of these errors makes the tasks of precisely following simple routes, shown in Figure 3, hardly solvable. The driving lines, shown in figures 3a, 3b and 3c, conversely to the expected lines shown in 3d, 3e and 3f, respectively, illustrate the detrimental effects of the feedback latency on the precision of the control.

Anticipating the State of the Car and the Environment

In order to investigate the detrimental effect of latency on the performance of the driving agent, and to verify the effectiveness of the proposed approach for its alleviation, we developed a software simulator of the car and the tracks. The additional rationales behind the simulator include (i) the possibility to verify the feasibility of certain circuit configurations without the need to consider the risks of possible damage to the environment or the car (or both), and (ii) the opportunity to "compress" the runtime of the fitness evaluation in the eventual implementation of agent's evolution [10, 13]. Furthermore, while operating the real car, the driving agent continuously applies the kernel of the developed simulator — the internal model of the car and the environment in order to anticipate the car's intrinsic state from currently available (outdated) perceptions. The software simulator takes into consideration the Newtonian physics of the (potentially sliding) car and the random uniform noise of +/-1 pixel (equal to the experimentally obtained value) incorporated in the "tracking" of the modeled car.

In the proposed approach of incorporating an anticipatory model [16], the driving agent considers its current actions based on anticipated intrinsic (rather than currently available, outdated) state of the car and surrounding environment. The agent anticipates the intrinsic state of the car (position, orientation, and speed) from the currently available outdated (by 90ms) state by means of iteratively applying the history of its own most recent actions (i.e., the throttle and steering commands) to the internal model of the car. It also anticipates the perception information related to the surrounding environment, (e.g., the distance and the bearing to the apex of the next turn) from the viewpoint of the anticipated intrinsic position and orientation of the car. The approach is related to the dead reckoning in GPS-based vehicle navigation [1].

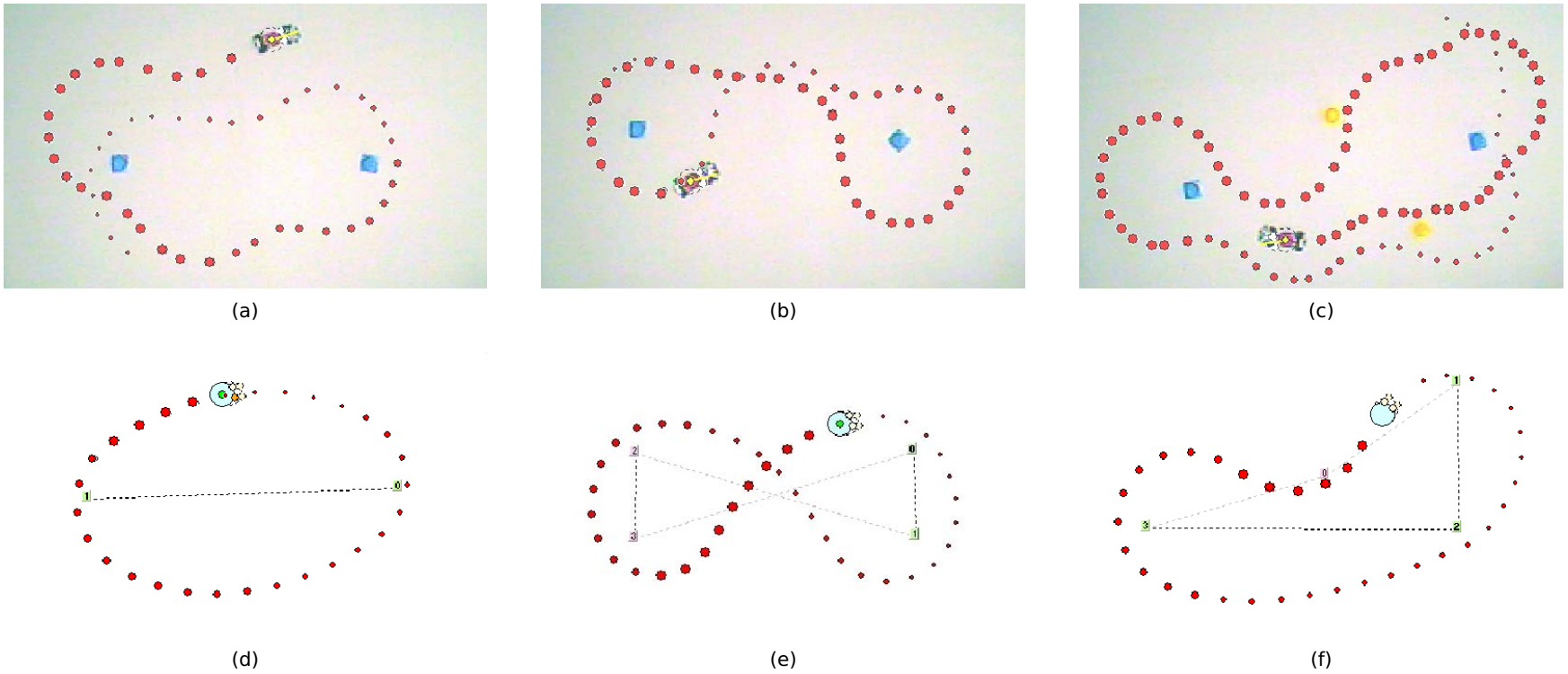


Fig. 3: Driving lines of the scaled model of the car (top) and the expected lines of the modeled car (bottom), controlled by agent in O-shaped (a and d), 8-shaped (b and e), and S-shaped (c and f) circuits, respectively. The actual driving lines (a, b, and c) differ significantly from the expected ones on the same circuits (d, e, and f, respectively).

Following Simple Routes with Anticipatory Modeling

The emerged driving lines (superposed over four consecutive laps) of the scaled model of the car employing an anticipatory model are shown in Figure 4. As Figure 4 illustrates, the driving lines of the car controlled by an agent employing anticipatory modeling in a system with latency feedback (figures 4a, 4b, and 4c) are much similar to the expected driving lines of the modeled car with non-latency feedback (Figures 3d, 3e and 3f, respectively). These results experimentally verify the compensatory effect of the anticipation on the performance of the driving agent.

Moreover, the superposition of the driving lines obtained over four consecutive laps demonstrates the consistence of trajectory of the car. Indeed, as Figure 4 illustrates, the maximal lateral deviation of the center of the car (shown as dark trailing circles) from the imaginable average is less than 60% of the width of the car, i.e., less than 55mm. In the most challenging S-circuit, the small variations in both the actual length of the lap and the average lap speed result in a relative standard deviation of the lap time of about 2.2% (120ms of the average 5300ms). We view this result as an important prerequisite for the human competitiveness of the agent in terms of its consistence. Moreover, this consistence is relevant for the efficiency of the optimization of the driving style of a fast moving car. As we should elaborate later, the minimization of the lap times of the car around the predefined circuits can be achieved via evolutionary optimization of the parameters of the driving style. In such an optimization, the achieved lap time (being the fitness value of the evolved driving agent) is the only feedback obtained from the interaction of the agent and the environment. An inaccurate evaluation of the fitness due to inconsistent lap times would result in a noisy fitness, which in turn would have a detrimental effect on the computational effort (convergence) of the evolutionary algorithms [14]. Because the noisy fitness value can be viewed as a sum of the real fitness plus a random, unbiased (mean of zero) noise component, increasing the number of the laps during the fitness evaluation of the evolving agent would cancel, to some extent, the random noise component and consequently, reduce the fitness noise. Such an approach, however, would increase the runtime of the fitness evaluation which would reduce the computational performance of the simulated evolution.

The above-discussed small inconsistencies in both the driving lines and the lap times are caused by the combination of the following inaccuracies:

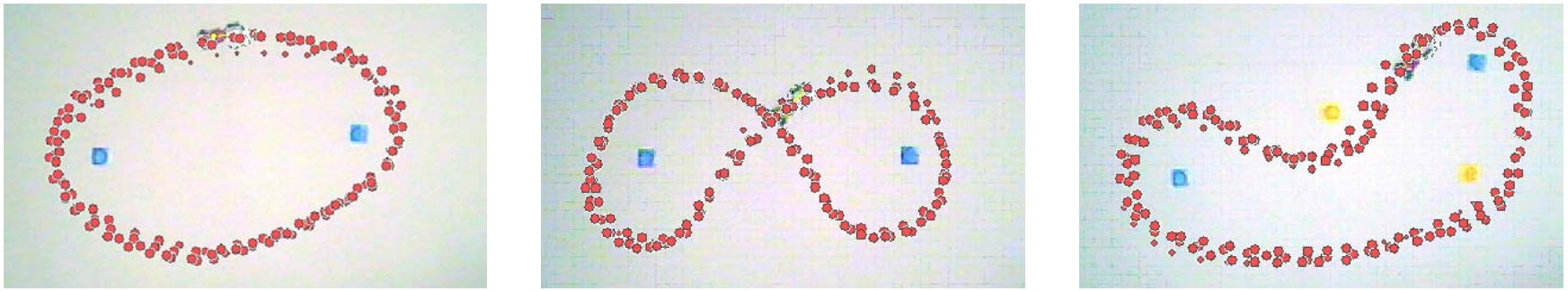
- Inaccuracy of the computer vision: imprecise, noisy determination of the position of the car due to both the irregularities of colors (i.e. shadows) in the perceived image of the car and the variable foreshortening. The problem is alleviated by Kalman filter [10], which exploits the Newtonian mechanics of the car, as defined in its internal model.
- Sampling error, caused by the discrete perceptions-actions control loop of the agent with a finite sampling interval.

The small differences between the modeled driving lines (Figure 3d, 3e, and 3f) and the real ones with anticipatory modeling (Figure 3a, 3b and 3c, respectively) is caused by the inaccuracy of the internal model of the car. The perspective distortion of the camera yields a variable scale of the scene (mm per pixel), and consequently, a variation in the perceived mechanical characteristics of the car (acceleration, maximal speed, turning radius, etc.) depending on the actual distance between the camera and the car. In our current implementation, neither the vision subsystem nor the model of the car attempt to correct this distortion.

Evolution of Driving Styles

Attributes of the Driving Style

We consider the driving style as the driving line, which the car follows before, around, and after the turns in the circuits combined with the speed, at which the car travels along this line. Our choice of driving styles' parameters is based on the view, shared among the high-performance drivers from various racing teams in different formulas, that (i) the track can be seen as a set of consequent turns they need to optimize divided by simple straights, and that (ii) the turns with the preceding and following straights should be treated as a single whole [2, 5]. Based on these standpoints, we introduce the following key attributes of the driving style, pertaining to each of the turns of the circuit: (i) straight-line gear — the gear at which the car approaches the turn, (ii) turning gear, (iii) throttle lift-off zone – the distance from the apex at which the car begins slowing down from the velocity corresponding to the straight line gear to the velocity of the turning gear, (iv) braking velocity — the threshold above which the car being in the throttle lift-off zone applies brakes (i.e., reverse throttle command) for slowing down, and (v) approach (homing) angle — the constant bearing of the apex of the turn. Higher values of the latter parameter yield wider driving lines featuring higher turning radiuses.



(a) (b) (c)

Fig. 4: Driving lines superposed over four consecutive laps in O- (a), 8- (b), and S-shaped (c) circuits respectively. The agent employs an anticipatory model to compensate the feedback latency. The car, shown in the top left corner of (a), (b) and (c) quantitatively illustrates the scale of the snapshots.

Viewing the desired values of these attributes as values that the agent has to maintain, the functionality of the agent can seen as issuing such a control sequence that result in the perceived state of the car and environment to match the desired values of the corresponding parameters. The control algorithm of the driving agent is elaborated in details in [19].

Evolution of Driving Styles

Assuming that the key parameters of optimal driving style around different turns of a circuit will feature different values, our objective of automatic design of optimal driving styles can be rephrased as an automatic discovery of the optimal values of these parameters for each of the turns in the circuit. This section elaborates (i) on the genetic algorithm (GA), proposed for the automatic discovery of these optimal values on the software simulator of the car and (ii) on the adaptation of the evolved solution to the concrete physical characteristics of the real scaled model of the car on the real track.

Representation. The genotype in the proposed genetic algorithm encodes for the evolving optimal values of the key parameters of the driving style for each of the turns of a given circuit. In order to allow for the crossover operation to swap not only the values of a particular parameter, but also the complete set of driving style parameters associated with particular turn with the complete set of parameters of another turn, and consequently, to protect the higher granularity building blocks from the destructive effects of crossover, we implement a hierarchical, tree-based representation of the genotype as a parsing tree, as usually employed in genetic programming. A sample genotype, which encodes the driving style parameters for a four-turn circuit, represented as XML/DOM formatted text, is shown in Figure 5. The main parameters of GA are shown in Table 1. The sample circuit considered in our experiments of evolving driving styles of the agent operating both the software model and the real scale model of the car is shown in Figure 6a. The circuit features a combination of one high-speed (turn #3), one medium-speed (turn #1) and two low-speed hairpin turns (turn #2 and turn #4), represented in the figure with their respective apexes. The series of turns 4-1-2 form a challenging, technical S-shaped sector of right, left, and right turn. The length of the track, measured between the apexes of the turns is about 3800mm. The walls (“guardrails”) are virtual in that they are not physically constructed on the track. Consequently “hitting” the walls in has no effect the dynamics of either the simulated or the real car. However, each “crash” is penalized with 0.4s (about 10% of the expected lap time), added to the actual lap time. This reflects our intention to evolve driving agents that avoid the potentially dangerous crashes into the eventual real walls rather than trying to exploit the occasional benefits of bouncing from them.

```
<?xml version="1.0" ?>
- <GP xmlns:xs="http://.../XMLSchema-instance" ...>
- <DStyle>
+ <Turn>
- <Turn>
    <StraightLineGear>4</StraightLineGear>
    <ApproachingMode>0</ApproachingMode>
    <ApproachingAngle>8</ApproachingAngle>
    <ApproachingAngleThreshold>7
    </ApproachingAngleThreshold>
    <ThrottleLiftOffZone_x10>29
    </ThrottleLiftOffZone_x10>
    <BrakingVelocity_x10>186</BrakingVelocity_x10>
    <TurningGear>3</TurningGear>
    <DistToCurrSwitchToNext_x10>
       12
    </DistToCurrSwitchToNext_x10>
    </Turn>
    +<Turn >
    +<Turn >
 </DStyle>
 </GP>
```

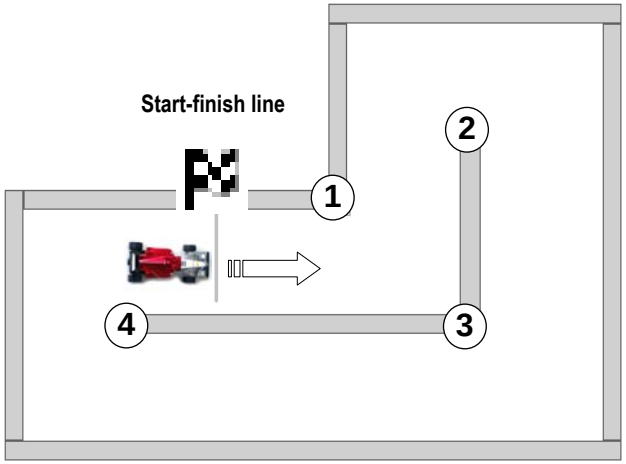
Fig. 5: Sample genotype represented as XML/DOM-formatted text. The sub-tree with the values of attributes of the second turn of the four-turn circuit is shown expanded.

Category	Value
Population size	100 individuals
Selection	Binary tournament, selection ratio 0.1, reproduction ratio 0.9
Elitism	Best 4 individuals
Mutation	Random sub-tree mutation, ratio 0.01
Trial interval	Single flying lap for the software model and two flying laps for the real car
Fitness	Average lap time in milliseconds
Termination criteria	Number of generations = 40

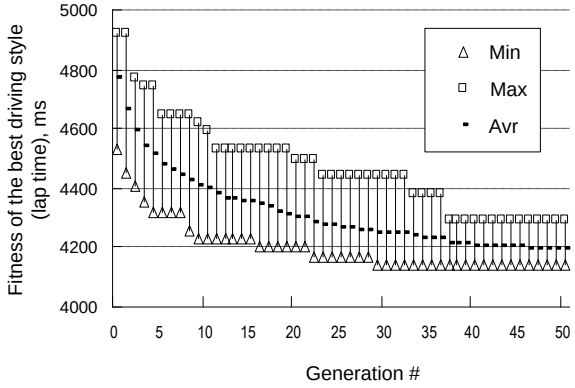
Tab. 1: Main parameters of GA

Offline Evolution of Driving Styles. The fitness convergence results of the offline evolution on the software anticipatory model of the car, aggregated over 50 independent runs of GA, are shown in Figure 6b. As Figure 6b illustrates, the best lap time average over all runs of GA improved from 4770ms to about 4200ms (i.e., about 14%) within 40 generations, which for a single run of GA consumes about 24 minutes of runtime on a PC featuring 3GHz CPU, 512MB RAM and Windows XP OS. For the measured average speed of about 1100mm/s the achieved average reduction of lap time by 570ms corresponds to an advantage of about 63cm (more than 3 lengths of the car) per lap.

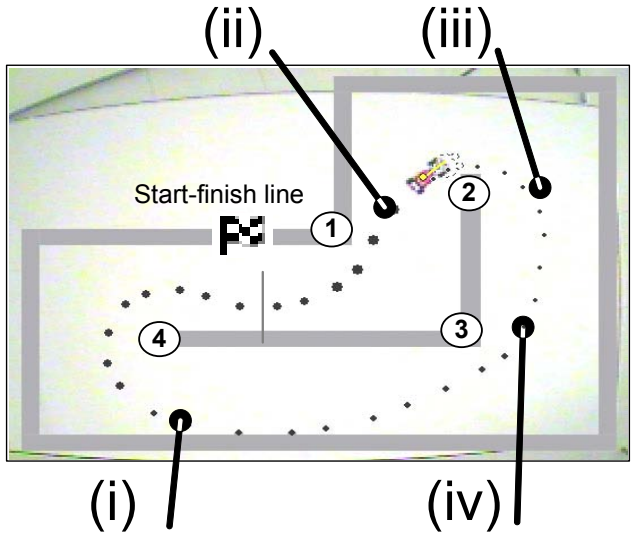
Porting the Evolved Solution to the Real Car. This step can be viewed as a process of adaptation to the changes in the fitness landscape of the task. In our approach we employ the same GA framework we used for offline evolution also for adapting a set of good solutions to the real car. At the beginning of the adaptation the GA is initialized with a population comprising 20 best-of-run driving styles obtained from the offline evolution. In order to address the challenges of (i) guaranteeing an equal initial conditions for the time trials of all candidate solutions and (ii) positioning the real car before each time trial automatically, we devise the time trials to comprise an out-lap followed by a series of flying timed laps, and finally, an in-lap. The approach is similar to the current qualifying format in the car racing formulas. After crossing the start-finish line (shown between turn #4 and turn #1 in Figure 6a) completing the final timed lap governed by the current driving style, the car enters the in-lap and slows down under “Neutral” throttle command. Depending on the speed at the start-finish line, the car comes to a rest at a point somewhere between the turn #1 and turn #2. At this point, which can be seen as an improvised pit stop, the next driving style which has to be evaluated is loaded into the agent’s controller, and the car starts its out-lap. Controlled by the new driving style, the car negotiates turns #2, #3 and #4. During the out lap the car covers a distance from the pit stop to the start-finish line, which is sufficient enough to cancel out any effect of the previously evaluated driving style on the performance of the currently evaluated driving style. In order to compensate for the eventual small inconsistence of the lap time, a total amount of 2 timed laps are conducted during the time trial of each driving style, and the average lap time is considered as a fitness value.



(a)



(b)



(c)

Fig. 6: Sample circuit used for evolution of driving style of agent (a), the fitness convergence characteristics (b) of offline evolution of driving styles on this circuit and (c) the emergent features of an online evolved best driving style.

The online evolution of the initial population of 20 best-of-run solutions obtained offline was allowed to run until no improvement in fitness value of the best driving style have been registered for 4 consecutive generations. A single run has been completed, and improvement of the aggregated fitness value of the best solution from the initial value 4930ms (due to the initial hitting of the "walls" by the fast agents evolved offline and currently operating the scale model of the car) to 4140ms has been observed within 10 generations. The emergent features of the evolved best driving style of the anticipatory agent are shown in Figure 6c. As illustrated in the figure, (i) the car starts its flying lap entering the turn #4 relatively wide and exiting it close to the apex, which allows (ii) negotiating the turn #1 and the following turn #2 using the shortest possible driving line. The car exits the turn #2 in a way (iii) that allows for a favorable orientation at the entrance of the following turn #3 and (iv) an early acceleration well before its apex, contributing to the achievement of the faster speed down the back straight between turns #3 and #4. The car uses the full width of the track and enters the turn #4 wide and exits it close to its apex preparing for the first turn of the next flying lap.

Human Competitiveness of the Evolved Driving Agent

In order to evaluate the human competitiveness of the evolved driving agent we collected the experimental results of the lap times of a sample best driving agent over 300 consecutive laps (30 runs of 10 laps each) in the sample circuit shown in Figure 6a, and compared them with the analogous results achieved by human drivers. The circuit is cleaned after each of the runs to eliminate the effect of the variable friction coefficient caused by rubber dust (from the sliding tires of the car) accumulated from the previous runs on the lap times of the following runs. In each of the series of 30 runs (of 10 laps each), the human operates the car in three different modes as follows: human perceives the scene (i) directly, (ii) via the PC-visualized video feed of the same overhead camera as used by the driving agent, and (iii) via PC-visualized video feed of a camera (with wide field of view of 66 degrees) mounted onboard the car. The later two cases attempt to mimic the real-world situations when a human operator remotely controls a moving artifact via inherently delayed video feedback. In both of the considered cases, the video stream is outdated by the same 90ms as the stream which provides the driving agent with an environmental feedback. The human operator conveyed the commands to the car via the standard radio RC unit.

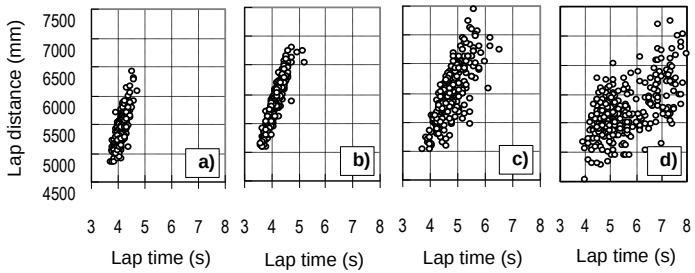


Fig. 7: Lap times vs. lap distance of 300 laps (30 runs of 10 laps each) of the driving agent (a), and a human operator perceiving the scene directly (b), via the overhead camera (c) and onboard camera (d), respectively.

The single human operator, competing with the driving agent in the presented experiment, is the winner among four candidates (including the first author) in a series of qualifying runs with the car. Despite that all the candidates are by far not novices in the world of the RC cars, the qualifying runs were preceded by few days of practice runs which allowed the candidates to better understand both the track and the car. This in turn helped the candidates to discover their own, best driving style.

The distribution of the lap times vs. the lap distance for the 300 runs of the driving agent and human, operating the car in the three modes as elaborated above, is shown in Figure 7a, 7b, 7c and 7d, respectively. The aggregated results of these runs, as shown in Table 2, indicate that the human is nearly equally competitive to the agent in terms of average lap time (4.15s for human vs. 4.14s for computer) only when the human perceives the scene directly. However, even in this case, the consistence of the lap times of the human is much inferior to the driving agent (0.27s for human vs. 0.16s for the agent). With the introduction of the latent video feedback from an overhead camera, both the lap time and the consistence deteriorate (4.86s, and 0.65s, respectively). The onboard camera, providing the driver not only with an outdated, but also incomplete information about immediate surrounding in the direction of the traveling of the car, yields additional degradation in terms of both the lap time and it's consistence (5.63s, and 1.05s, respectively). In the case of direct view of the scene, human surpass the agent in the average velocity around the track (1404mm/s for human vs. 1304mm/s for the agent). However, this speed is also associated with increased lap

distanced (5828mm for human vs. 5400mm for the agent) suggesting a poorer control of the understeering. As we previously elaborated, the car experiences an understeer with an increased turning radius and significant braking momentum due to the sliding front wheels if it enters the corner too fast. The understeering also temporarily compromises the controllability of the car which in turn contributes to the inferior consistence of lap distance, lap velocity, and resulting lap times of the human operator.

The results of the experiments on consistence of the lap times during the runs indicate that the deviation of the average lap times of the driving agent is quite small, and could be contributed to the factors as elaborated earlier. In addition, we noticed a slight, but steady increase of the lap time towards the end of the runs of agent, which is caused by the reduced friction coefficients of the track around turns #2 and #4 (as shown in Figure 6), due to the rubber dust, scrubbed off the sliding tires of the slightly understeering car. Conversely, the results achieved by human drivers indicate the inferiority of the initial lap(s) due to the lack of adequate concentration (e.g., warm-up effect). In addition, the deviations of the lap time of the human increase during the second half of the run, which could be attributed to some tiredness. Moreover, the detrimental effect of tiredness on both the absolute value and consistence of the lap time in the experiment of controlling the car via onboard camera seem to be so strong that it overcompensates the usual improvement of the human's performance after the initial lap(s). In both cases with latency feedback, which requires an anticipation of the intrinsic state of the car, the human is unable to mach his own speed achieved with the direct observation of the scene (Figure 6a).

The comparison of the several relevant characteristics of the driving agent and a human operator is shown in Table 3. The indicated range of the reaction time of a human driver is as presented in [12]. The value of the maximal frequency of control of the human, shown in Table 3, is as we measured during the experiments. As Table 3 indicates, in all of the features, listed in the table, the agent is either equally good or surpasses the human. While most of the superiority should be attributed to the advantages of the very concept of the computer-based control (e.g., consistence, reaction time, and precision of control), the ability to optimize the driving style and the anticipatory abilities, both equally relevant for the human competitiveness of the agent, are achieved as a result of the approaches discussed in this work: the evolutionary optimization and the anticipatory modeling, respectively.

Conclusions

The objective of this work is an evolutionary design of driving agent, able to remotely operate a scale model of racing car running in human competitive way. The agent's actions are conveyed to the car via simple remote control unit. The agent perceives the environment from live video feed of an overhead camera. In order to cope with the inherent video feed latency we implemented an anticipatory model in which the agent considers its current actions from the anticipated intrinsic state of the car and its surrounding. We formalized the notion of driving style and defined the key parameters, which describe it, and applied genetic algorithms to evolve the optimal values of these parameters. The optimized driving style, employed by the agent is human competitive in that it yields both faster and more consistent lap times around the predefined circuit. Presented work can be viewed as a step towards the automated design of the control software of remotely operated vehicles capable to find an optimal solution to various tasks in a priori known environmental situations. The results can be seen also as a verification of the feasibility of developing a framework of racing games in which a human competes against at least an equally strong computerized opponent.

In our future work we plan to incorporate additional cars and challenging obstacles that should be negotiated in a robust way regardless of their number, their a priori unknown locations, car moving directions or velocities.

Acknowledgment

We would like to thank the passionate "drivers" of the remotely controlled scale model of a car M.Joachimczak, T.Nakagawa and S.Prudent.

Bibliography

[1] E. Abbott, and D. Powell, *Land-vehicle Navigation Using GPS*, The Proceedings of the IEEE, 1999, vol. 87, No 1, pp.145-162.

[2] R. Bentley, *Speed Secrets: Professional Race Driving Techniques*, Motorbooks International, 1998.

[3] B. Chaperot and C. Fyfe, *Improving Artificial Intelligence in a Motocross Game*, IEEE Symposium on Computational Intelligence and Games, CIG'06, May 2006, pp.1435 - 1500

Parameter	Operator of the Scale Model of a Car			
	Driving Agent	Human, Direct View	Human, Overhead Camera View	Human, Onboard Camera View
Average lap time, s	4.14	4.15	4.86	5.63
Standard deviation of lap time, s	0.16	0.27	0.65	1.05
Average lap distance, mm	5400	5828	6055	5708
Standard deviation of lap distance, mm	267	368	520	467
Average Speed, mm/s	1304	1404	1246	1014
Scaled Average Speed, km/h	113	121	108	88

Tab. 2: Human competitiveness of the evolved driving agent.

Characteristic	Driving Agent	Human Operator
Ability to optimize the driving style	Equally good	
Ability to consistently follow the optimized driving line	Good	Relatively poor (warm-up, tiredness, slower reaction time)
Anticipatory abilities	Good	Relatively poor
Reaction time (perception, recognition, decision and physical response)	0.03s	0.4 – 2.7s
Precision of control: Max frequency of commands in pulse width modulation	$30s^{-1}$	$4 – 6s^{-1}$

Tab. 3: Comparative relevant characteristics of the evolved driving agent and human operator.

[4] D. B. Fogel , *Blondie24 : Playing at the Edge of AI*, Morgan Kaufmann, 2001.

[5] P. Frere, *Sports Cars and Competition Driving*, Bentley Publishing, 1992.

[6] J. D. Funge, *Artificial Intelligence for Computer Games*, Peters Corp., 2004.

[7] T. Gillespie, *Fundamentals of Vehicle Dynamics*, Society of Automotive Engineers International, 1992.

[8] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addision-Wesley, 1989.

[9] IBM Corporation: Deep Blue, 1997, URL: http://www.research.ibm.com/deepblue/

[10] N. Jacobi, *Minimal Simulations for Evolutionary Robotics*, Ph.D. thesis, School of Cognitive and Computing Sciences, Sussex University, 1998.

[11] J.Koza, *Human Competitive Machine Intelligence by Means of Genetic Algorithms*, Festschrift in honor of John H. Holland, Center for the Study of Complex Systems, Ann Arbor, USA, 1999, pp.15-22.

[12] K. Ma, and I. Andréasson, *Driver reaction Time Estimation from Real Car Following Data and Application in General Motor-type Model Evaluation*, Transportation Research Record, No.1965, 2006, pp.130-141.

[13] L. Meeden, and D. Kumar, *Trends in Evolutionary Robotics*, Soft Computing for Intelligent Robotic Systems, edited by L.C. Jain and T. Fukuda, Physica-Verlag, New York, NY, 1998, pp. 215-233

[14] B. L. Miller, and D. E. Goldberg, *Genetic Algorithms, Tournament Selection, and the Effects of Noise*, Complex System, 9(3), 1995, pp.193-212

[15] Robocup, 2005 URL: http://www.robocup.org/

[16] R. Rosen, *Anticipatory Systems*, Pergamon Press, 1985.

[17] M.Suzuki, and D.Floreano, *Active Vision for Neural Development and Landmark Navigation*, In 50th Anniversary Summit of Artificial Intelligence, 2006, pp. 247-248.

[18] J. Togelius and S. M. Lucas, *Evolving Controllers for Simulated Car Racing*, Proceedings of IEEE Congress on Evolutionary Computations (CEC-2005), Edinburgh, UK, 2005, pp.1906-1913.

[19] I. Tanev, M. Joachimczak, H. Hemmi. and K. Shimohara, *Evolution of the Driving Styles of Anticipatory Agent Remotely Operating a Scaled Model of Racing Car*, Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC-2005), Edinburgh, UK, 2005, pp.1891-1898.

[20] I. Tanev, M. Joachimczak and K. Shimohara, *Evolution and Adaptation of an Agent Driving a Scale Model of a Car with Obstacle Avoidance Capabilities*, Proceedings of the Ninth International Conference on the Simulation of Adaptive Behavior (SAB 2006), LNAI 4095, Springer-Verlag Berlin Heidelberg, pp.619-630.

[21] K. Wloch, and P. Bentley, *Optimizing the Performance of a Formula One Car Using a Genetic Algorithm*, Proceedings of the 8th International Conference on Parallel Problem Solving from Nature, Birmingham, UK, 2004, pp.702-711.

About the authors

Ivan Tanev received M.S. (with honors) and Ph.D. degrees from Saint-Petersburg State Electrotechnical University, Russia in 1987 and 1993 respectively, and Dr.Eng. degree from Muroran Institute of Technology, Japan in 2001. He has been with the Bulgarian Space Research Institute (1987), Bulgarian Central Institute of Computer Engineering and Computer Technologies (1988-1989), Bulgarian National Electricity Company (1994-1997), Synthetic Planning Industry Co.Ltd., Japan (2001-2002), and ATR Human Information Science Laboratories (2002-2004), Japan. Since April 2004 he has been with the Department of Information Systems Design, Faculty of Science and Engineering, Doshisha University, Kyoto, Japan. Dr. Tanev's research interests include evolutionary computations, multi-agent systems and socioinformatics. He is a member of IEEE and SIGEVO.

Homepage: isd-si.doshisha.ac.jp/itanev/
Email: itanev@mail.doshisha.ac.jp

Katsunori Shimohara received the B.E. and M.E. degrees in Computer Science and Communication Engineering and the Doctor of Engineering degree from Kyushu University, Fukuoka, Japan, in 1976, 1978 and 2000, respectively. He was Director of the Network Informatics Laboratories and the Human Information Science Laboratories, Advanced Telecommunications Research Institute (ATR) International, Kyoto, Japan. He is currently a Professor at the Department of Information Systems Design, Faculty of Science and Engineering, ant the Graduate School of Engineering, Doshisha University, Kyoto, Japan. His research interests include human communication mechanisms, evolutionary systems, human-system interactions, genome informatics and socio-informatics. Dr. Shimohara is a member of IEEE, the Institute of Electronics, Information and Communication Engineers, the Japanese Society of Artificial Intelligence, the Human Interface Society, and the Society of Instrument and Control Engineers.

Email: kshimoha@mail.doshisha.ac.jp

GECCO-2007 Highlights

Petroleum Applications of Evolutionary Computation Workshop

Alexandre Castellini, Chevron Energy Technology Company, USA
Charles Guthrie, Chevron Energy Technology Company, USA
Burak Yeten, Chevron Energy Technology Company, USA
Tina Yu, Memorial University of Newfoundland, Canada

Workshop webpage: WWW

Overview

The GECCO conference is the largest annual conference in the field of Genetic and Evolutionary Computation. Workshops on specific topics are organized every year. In July 2007, Chevron and Tina Yu from the University of Newfoundland organized the first workshop on Petroleum Applications. Speakers from industry and academia shared their recent work on a variety of subjects and a panel discussion addressed the gaps and opportunities for Oil and Gas companies.

Dominant themes

Cullick (Landmark, USA) presented an integrated simulation-optimization framework applied to field development, well locations and number, drilling scheduling, facilities planning, and valve operation. The stability of solutions is consistently tested and subsurface uncertainty is incorporated in the process with multiple earth models. An efficient history-matching strategy is also illustrated with the use of non linear proxy models in the form of neural networks.

Guyaguler (Chevron, UK) addressed the weaknesses of Genetic Algorithms and presented solutions that combine GAs with other mathematical tools. The well placement and rate optimization of an off-shore field is achieved via a hybrid algorithm that consists of GA combined with hill-climbers. Well placement under uncertainty is achieved through a customized GA that works on a number of realizations of the model at the same time. GAs, together with response surfaces are used to history match a model to achieve probabilistic forecasts. Finally a customized GA is used to find the underlying model of a well-test response.

Davis (VGO Associates, USA) focused on the problem of scheduling oil well workover operations in the Permian Basin in West Texas. He explained how the workover team learned to see the workover process as a probabilistic process. A work over simulation contains events that occur with specified probabilities. The genetic algorithm for optimizing workover schedules is described and the transformations in the workover process caused by the new approach are presented.

Petrovska (Imperial College, UK) presented a Population-Based Incremental Learning algorithm (PBIL) applied to history-matching. An initial population of models is generated from the prior distribution of all uncertain parameters. In each iteration promising solutions are selected and a corresponding probability model is constructed. A user-defined fraction of information is preserved from one generation to the other. The new probabilistic model of promising solutions is then used to sample new candidate solutions.

Cruz (Applied Computational Intelligence Laboratory, Brasil) applied a GA to optimize the scheduling of crude oils in refineries for Petrobras. To efficiently produce the contracted oil products such as petrol, jet fuel, fuel oil, a number of tasks had to be optimized: allocating terminal tanks to unload the vessels considering their scheduled arrival times; scheduling the pipeline transfers, which include allocating the terminal and refinery tanks and finally scheduling the crude distillation units feeds.

Fayez (Cairo University, Egypt) looked at optimizing the pipe size for natural gas networks for Egypt Gas Co. The objective was to minimize the total cost of the network while satisfying a set of practical constraints including pressure head at the demand node above a given limit and flow rate continuity at junction nodes.

Panel discussion topics

- Although most problems require multiple variables to be optimized simultaneously, a single objective function is almost always considered. What is preventing us from doing Multiple Objective function optimization?
- The Petroleum industry has been generally slow to accept EC techniques. What is the bottleneck? What are the suggestions to bridge the gap between industries? More specifically, Genetic Algorithms and Evolutionary Strategies account for most techniques employed in Oil and Gas applications. Are other techniques not practical?
- For a given system, uncertainty in input parameters leads to uncertainty in output variables. Rigorous uncertainty assessment typically precludes the use of optimization techniques. Can we have the best of both worlds?
- Opportunities for graduates with special knowledge of Evolutionary Strategies in Petroleum oriented organizations.

Key Takeaways

- Evolutionary computation techniques are currently applied to a broad range of domains in the industry and are not limited to the upstream realm: field development, history-matching, well work-over operations, well testing, scheduling of crude oil in refineries, natural gas pipe network design, cyclic steam injection. . .
- In complex non-linear problems, global optimizers such as Genetic Algorithms (GA) and Evolutionary Strategies (ES) are necessary to achieve good performance.
- High performance computing capabilities is essential for a practical use of such optimization techniques.
- Some practitioner use meta-heuristic techniques like scatter and tabu search but most prefer GA or ES. One key characteristic that differentiates ES from GA is the ability to process self-adaptive mutations. In general, Evolution Strategies require a smaller number of evaluations and converge faster to optimum solution but it is done at the detriment of a robust uncertainty assessment (poorer coverage or the parameter space).
- Most GAs weaknesses can be overcome when combined with other tools/optimizers: Neural Network, non-linear response surfaces, hill-climbers can accelerate the convergence of GA and improve the search for an optimum.
- GA or ES come in many sizes and shapes and have tuning parameters that can be customized for a given problem.
- In many cases, carefully designed optimization strategies have produced better results (faster, lower cost, smaller objective function) than those with a heuristic approach based on human experience and judgment. This led to improved learning about the system and behavioral changes in organizations.
- Most optimization problems are multiple objective function problems (e.g. maximize oil produced and minimize water produced). Looking for solutions on the Pareto front is still fairly academic. Building a single objective function with appropriate weights is more practical. Several iterations and sensitivity analysis needed to set up weights. One solution is to normalize all responses in dollar terms.
- Slow acceptance of Evolutionary Computation techniques:
  - Lack of customizable toolkits. There is a need for a GA for Excel plug-in. A lot of efforts wasted in improving techniques when the user needs easy-to-use packages.
  - Problem of trust: good collaboration needed between user/operator who knows the physical problem, the heuristics and the EC specialist who knows what is in the black box.
  - Computer cluster availability.
  - Operation Research people control the brain waves in this domain and the petroleum industry has an image problem that prevents good collaboration.
  - Additional layer of complexity to the already difficult task of deciding on the right parameters and ranges.
- Alternative to GA or ES: Particle Swarm. Easy to implement, used successfully on a limited set of problems. Best to use GA with integer parameters and Particle Swarm with continuous variables (Note: Particle Swarms are being investigated in Stanford's SUPRIB research program of which Chevron is a sponsor).

- Population-Based Incremental Learning algorithms (PBIL) look promising but more real-life field studies need to be performed. The main difference between PBIL and other evolutionary optimization techniques is that instead of evolving a population of models the algorithm evolves a probability model.
- Optimization vs. uncertainty: Evolutionary techniques naturally provide multiple solutions but they tend to be clustered in a limited subspace which is not, well, optimal for robust uncertainty assessment. Some stochastic methods can improve the process. Declustering techniques can also help at the post-processing stage.
- Opportunities for EC graduates? Very small in this industry so far. Companies have to be less conservative, more open to new ideas and build more bridges with universities. Most of the time, a PE develops an interest for the EC field at the graduate level, rarely a computer scientist will be attracted by a career in our industry. Internships are good opportunities for them to discover the Oil and Gas business.

Material

The workshop webpage contains links to biographies, abstracts, presentations prepared by the speakers and the complete recording of the panel discussion.

Information about the organizers: WWW
Workshop webpage: WWW
GECCO-2007 conference webpage: WWW

If you have any question, please contact one of the organizers.

GECCO-2008 New Contest Problem!

Solving Rubik's Cube ($1,000 prize)

In conjunction with the 2008 Genetic and Evolutionary Computation Conference (GECCO), Parabon Computation, the leading on-demand computation utility, announced today that it will sponsor a $1,000 prize competition in which contestants are challenged to evolve a program that can solve an arbitrarily scrambled Rubik's Cube. The competition is designed to demonstrate the combined capability of two complementary computing technologies provided by Parabon Computation. The Origin(tm) Evolutionary SDK (software development kit) uses computer-based evolutionary processes to derive (evolve) desired results and the Frontier Grid Platform harnesses the computational power of thousands of computers to deliver supercomputing as a service.

Rather than writing a computer program to solve an arbitrarily scrambled Rubik's Cube (several such programs already exist), contestants must instead write a program that will evolve another program that is able to solve the cube in a minimal number of twists. The evolved programs start off with little or no capability, however, they are shaped by algorithms that mimic biological evolution — for example, algorithms for selecting "parent" solvers from a population, breeding them and possibly mutating offspring solvers. Gradually, over a sufficient number of generations, a population of solvers will emerge that not only solve the cube, but do so with ever improving efficiency. For such evolutionary programs to be effective, large-scale computation must be applied, and that's where Parabon Computation's grid service comes into play. The company's on-demand computing utility will provide contestants with high-performance computing capacity in the weeks leading up to the conference to demonstrate the ease and affordability of the service. Contestants will submit their best solvers for judging ahead of the conference and the winner will be announced at GECCO 2008, which is to be held July 12-16 in Atlanta.

Werner Randelshofer has developed a Java applet of an interactive Rubik's cube which can be found here.

Contest Deadline: July 6th, 2008

Contest Rules are available here.

To participate, contestants will need to:

1. Sign up for a Frontier account
2. Download and install the Frontier SDK. (You will be asked to provide your Username and Password again.)
3. Download and install the Origin Evolutionary SDK

Documentation for Parabon's Frontier technologies can be found under the Dev Center tab.

Winners will be announced during GECCO.

General Competition organizers

- Dr. Terry Soule
  Department of Computer Science
  University of Idaho
  Moscow, ID, 83844-1010
  Email: tsoule@cs.uidaho.edu
- Dr. Robert B. Heckendorn
  Department of Computer Science
  University of Idaho
  Moscow, ID, 83844-1010
  Email: heckendo@cs.uidaho.edu

Contact for the Rubik's cube contest

- Dr. Steven Armentrout
  Parabon Computation, Inc.
  11260 Roger Bacon Dr, Ste 406
  Reston, VA 20190
  Email: steve@parabon.com

GECCO-2008 Best Papers Nominees

In 2002, ISGEC created a best paper award for GECCO. As part of the double blind peer review, the reviewers were asked to nominate papers for best paper awards. We continue this tradition at GECCO-2008. The Track Chairs, Editor in Chief, and the Conference Chair nominated the papers that received the most nominations and/or the highest evaluation scores for consideration by the conference. The winners are chosen by secret ballot of the GECCO attendees after the papers have been orally presented at the conference. Best Paper winners are posted on the conference website. The titles and authors of all papers nominated are given below:

Ant Colony Optimization, Swarm Intelligence, and Artificial Immune Systems

Collective Intelligence and Bush Fire Spotting
David Howden (Swinburne University of Technology)
Tim Hendtlass (Swinburne University of Technology)

Convergence Behavior of the Fully Informed Particle Swarm Optimization Algorithm
Marco A. Montes de Oca (Université Libre de Bruxelles)
Thomas Stützle (Université Libre de Bruxelles)

Evolutionary Swarm Design of Architectural Idea Models
Sebastian von Mammen (University of Calgary)
Christian Jacob (University of Calgary)

Theoretical and Empirical Study of Particle Swarms with Additive Stochasticity and Different Recombination Operators
Jorge Peña (Université de Lausanne)

Artificial Life, Evolutionary Robotics, Adaptive Behavior, Evolvable Hardware

The Influence of Scaling and Assortativity on Takeover Times in Scale-Free Topologies
Joshua L Payne (University of Vermont)
Margaret J Eppstein (University of Vermont)

Designing Multi-Rover Emergent Specialization
Geoff Nitschke (Vrije Universiteit)
Martijn Schut (Vrije Universiteit)

A Multi-scaled Approach to Artificial Life Simulation With P Systems and Dissipative Particle Dynamics
James Smaldon (University of Nottingham)
Jonathan Blakes (University of Nottingham)
Natalio Krasnogor (University of Nottingham)
Doron Lancet (Weizmann Institute of Science)

Modular Neuroevolution for Multilegged Locomotion
Vinod K Valsalam (The University of Texas at Austin)
Risto Miikkulainen (The University of Texas at Austin)

Bioinformatics and Computational Biology

An Efficient Probabilistic Population-Based Descent for the Median Genome Problem
Adrien Goeffon (INRIA)
Macha Nikolski (CNRS/LaBRI)
David J. Sherman (INRIA)

Structure and Parameter Estimation for Cell Systems Biology Models
Francisco J. Romero-Campero (Univeristy of Nottingham)
Hongqing Cao (University of Nottingham)
Miguel Camara (University of Nottingham)
Natalio Krasnogor (University of Nottingham)

Mask Functions for the Symbolic Modeling of Epistasis Using Genetic Programming
Ryan J Urbanowicz (Dartmouth College)
Nate Barney (Dartmouth College)
Bill C White (Dartmouth College)
Jason H Moore (Dartmouth College)

Coevolution

An Empirical Comparison of Evolution and Coevolution for Designing Artificial Neural Network Game Players
Min Shi (Norwegian University of Science and Technology)

Estimation of Distribution Algorithms

Using Previous Models to Bias Structural Learning in the Hierarchical BOA
Mark W Hauschild (University of Missouri - St. Louis)
Martin Pelikan (University of Missouri - St. Louis)
Kumara Sastry (University of Illinois at Urbana-Champaign)
David E. Goldberg (University of Illinois at Urbana-Champaign)

On the Effectiveness of Distributions Estimated by Probabilistic Model Building
Chung-Yao Chuang (National Chiao Tung University)
Ying-ping Chen (National Chiao Tung University)

From Mating Pool Distributions to Model Overfitting
Claudio F Lima (University of Algarve)
Fernando G Lobo (University of Algarve)
Martin Pelikan (University of Missouri at St. Louis)

Evolution Strategies, Evolutionary Programming

Why Noise May be Good
Silja Meyer-Nieberg (Universitaet der Bundeswehr Muenchen)
Hans-Georg Beyer (Vorarlberg University of Applied Sciences)

Functionally Specialized CMA-ES: A Modification of CMA-ES based on the Specialization of the Functions of Covariance Matrix Adaptation and Step Size Adaptation
Youhei Akimoto (Tokyo Institute of Technology)
Jun Sakuma (Tokyo Institute of Technology)
Shigenobu Kobayashi (Tokyo Institute of Technology)
Isao Ono (Tokyo Institute of Technology)

Aiming for a theoretically tractable CSA variant by means of empirical investigations
Jens Jägersküpper (TU Dortmund)
Mike Preuss (TU Dortmund)

Evolutionary Combinatorial Optimization

A Study of NK Landscapes' Basins and Local Optima Networks
Gabriela Ochoa (University of Nottingham)
Marco Tomassini (University of Lausanne)
Sebastien Verel (CNRS-University of Nice)
Christian Darabos (University of Lausanne)

Crossover Can Provably be Useful in Evolutionary Computation
Benjamin Doerr (Max-Planck-Institut für Informatik)
Edda Happ (Max-Planck-Institut für Informatik)
Christian Klein (Max-Planck-Institut für Informatik)

Evolutionary Multiobjective Optimization

A New Memetic Strategy for the Numerical Treatment of Multi-Objective Optimization Problems
Oliver Schuetze (CINVESTAV-IPN)
Gustavo Sanchez (Simon Bolivar University)
Carlos A. Coello Coello (CINVESTAV-IPN)

Introducing MONEDA: Scalable Multiobjective Optimization with a Neural Estimation of Distribution Algorithm
Luis Martí (Universidad Carlos III de Madrid)
Jesús García (Universidad Carlos III de Madrid)
Antonio Berlanga (Universidad Carlos III de Madrid)
José Manuel Molina (Universidad Carlos III de Madrid)

Pattern Identification in Pareto-Set Approximations
Tamara Ulrich (ETH Zurich)
Dimo Brockhoff (ETH Zurich)
Eckart Zitzler (ETH Zurich)

Benefits and Drawbacks for the Use of epsilon-Dominance in Evolutionary Multi-Objective Optimization Problems
Christian Horoba (Technische Universität Dortmund)
Frank Neumann (Max-Planck-Institut für Informatik)

Formal Theory

Computing Minimum Cuts by Randomized Search Heuristics
Frank Neumann (Max-Planck-Institut für Informatik)
Joachim Reichel (TU Berlin)
Martin Skutella (TU Berlin)

Memetic Algorithms with Variable-Depth Search to Overcome Local Optima
Dirk Sudholt (TU Dortmund)

Precision, Local Search and Unimodal Functions
Martin Dietzfelbinger (Technische Universitat Ilmenau)
Jonathan E Rowe (University of Birmingham)
Ingo Wegener (Technische Universitat Dortmund)
Philipp Woelfel (University of Calgary)

Generative and Developmental Systems

Generative Encoding for Multiagent Systems
David B. D'Ambrosio (University of Central Florida)
Kenneth O. Stanley (Univeristy of Central Florida)

A Cellular Model for the Evolutionary Development of Lightweight Material with an Inner Structure
Till Steiner (Honda Research Institute Europe GmbH)
Yaochu Jin (Honda Research Institute Europe GmbH)

Genetic Algorithms

Optimal Sampling of Genetic Algorithms on Polynomial Regression
Tian-Li Yu (National Taiwan University)
Wei-Kai Lin (National Taiwan University)

Rank Based Variation Operators for Genetic Algorithms
Jorge Cervantes (Universidad Autónoma Metropolitana)
Christopher Rhodes Stephens
(Instituto de Ciencias Nucleares UNAM)

Theoretical Analysis of Diversity Mechanisms for Global Exploration
Tobias Friedrich (Max-Planck-Institut fuer Informatik)
Pietro S. Oliveto (University of Birmingham)
Dirk Sudholt (TU Dortmund)
Carsten Witt (TU Dortmund)

Rigorous Analyses of Fitness-Proportional Selection for Optimizing Linear Functions
Edda Happ (Max-Planck-Institut Informatik)
Daniel Johannsen (Max-Planck-Institut Informatik)
Christian Klein (Max-Planck-Institut Informatik)
Frank Neumann (Max-Planck-Institut Informatik)

Genetic Programming

Parsimony Pressure Made Easyød
Riccardo Poli (University of Essex)
Nicholas Freitag McPhee (University of Minnesota, Morris)

The Impact of Population Size on Code Growth in GP: Analysis and Empirical Validation
Riccardo Poli (University of Essex)
Nicholas Freitag McPhee (University of Minnesota, Morris)
Leonardo Vanneschi (University of Milano-Bicocca)

Rapid Prediction of Optimum Population Size in Genetic Programming Using a Novel Genotype — Fitness Correlation
David C Wedge (University of Manchester)
Douglas B Kell (University of Manchester)

Learning to Recognise Mental Activities: Genetic Programming of Stateful Classifiers for Brain-Computer Interfacing

Alexandros Agapitos (University of Essex)
Matthew Dyson (University of Essex)
Simon M Lucas (University of Essex)
Francisco Sepulveda (University of Essex)

Genetics-Based Machine Learning

Context-Dependent Predictions and Cognitive Arm Control with XCSF

Martin V Butz (University of Würzburg)
Oliver Herbort (University of Würzburg)

Real-World Applications

Speeding Online Synthesis via Enforced Selecto-Recombination

Shunsuke Saruwatari
(University of Illinois at Urbana-Champaign)
Xavier Llorá (University of Illinois at Urbana-Champaign)
Noriko Imafuji Yasui (University of Illinois at Urbana-Champaign)
Hiroshi Tamura (Hakuhodo Inc)
Kumara Sastry (University of Illinois at Urbana-Champaign)
David E. Goldberg (University of Illinois at Urbana-Champaign)

Evolved Bayesian Networks as a Versatile Alternative to Partin Tables for Prostate Cancer Management

Ratiba Kabli (The Robert Gordon University)
John McCall (The Robert Gordon University)
Frank Herrmann (The Robert Gordon University)
Eng Ong (Aberdeen Royal Infirmary)

Genetic Algorithms for Mentor-Assisted Evaluation Function Optimization

Omid David-Tabibi (Bar-Ilan University)
Moshe Koppel (Bar-Ilan University)
Nathan S. Netanyahu (Bar-Ilan University)

Multiobjective Robustness for Portfolio Optimization in Volatile Environments

Ghada Hassan (UCL)
Christopher D. Clack (UCL)

Search-Based Software Engineering

Empirical Analysis of a Genetic Algorithm-based Stress Test Technique

Vahid Garousi (University of Calgary)

Fitness Calculation Approach for the Switch-Case Construct in Evolutionary Testing

Yan Wang (Software Engineering Institute, Xidian University)
Zhiwen Bai (Software Engineering Institute, Xidian University)
Miao Zhang (Software Engineering Institute, Xidian University)
Wen Du (Software Engineering Institute, Xidian University)
Ying Qin (Software Engineering Institute, Xidian University)
Xiyang Liu (Software Engineering Institute, Xidian University)

Searching for Liveness Property Violations in Concurrent Systems with ACO

Francisco Chicano (University of Málaga)
Enrique Alba (University of Málaga)

Simulated Car Racing Competition

Daniele Loiacono, Politecnico di Milano, Italy
Julian Togelius, IDSIA, Switzerland
Competition webpage: http://cig.dei.polimi.it/

The Simulated Car Racing Competition was organized in conjunction with WCCI-2008, the IEEE World Congress on Computational Intelligence. The competition was the spiritual successor of the previous car racing competition held at CIG-2007 (the IEEE Conference on Computational Intelligence and Games) and CEC-2007 (the IEEE Congress on Evolutionary Computation). The goal of this year's competition was to learn, or otherwise design a controller for The Open Racing Car Simulator (TORCS), a state-of-the-art, open source, car racing simulator, which could be able to race for a certain number of laps on a set of three unknown tracks, alone or against other drivers.

The Open Car Racing Simulator

The use of a car racing simulator with a powerful graphical engine (see the screenshot in the opposite column) was the first major innovation with respect to the previous editions which involved a simpler Java simulator developed by Julian Togelius specifically for the competition. The second major innovation was the introduction of a real-time client-server architecture. In the previous Java simulator, as well as in the original TORCS, controllers are embedded in the simulator and therefore a slow controller would slow down the race without receiving any penalization for that. In this competition, the simulator worked on a client server basis: controllers run as external programs that communicate with the simulator server through UDP connections. Controllers had to act quickly based on the most recent sensory information to properly control the car; a slow controller would be inherently penalized since it would be working on lagged information.

The controller perceived the racing environment through a number of sensor readings which would reflect the surrounding environment (the tracks and the opponents) and the current game state and it could invoke basic driving commands to control the car. The sensors include rangefinders to perceive the nearby track limits as well as the distance of nearby opponents, the current speed, the engine's RPM, the current gear, the fuel level, etc. The driving commands would control the steering wheel, the gas pedal, the brake pedal, and the gear change. To simplify the development of racing controllers, simple APIs were provided for C++ and Java languages and for Windows, Mac, and Linux operative systems.

The competition officially started at the beginning of February and ended on May 25th. Five controllers were submitted. **Matt Simmerson** submitted a neural controller evolved using NEAT4J, his Java implementation of *NEAT*, a well-known neuroevolution framework introduced by Kenneth O. Stanley and Risto Miikkulainen. **Leonard Kinnaird-Heether** under the supervision of Robert G. Reynolds from Wayne State University applied a rule-based approach: car effectors were controlled by a set of rules whose parameters were optimized using a cultural algorithm which viewed the racing simulation as a social event. **Simon Lucas** from the University of Essex submitted a manually programmed controller in which

human domain knowledge was used to improve one of the simple controllers provided as an example to the competitors. **Tan Chin Hiong** from the National University of Singapore designed a parametrized controller based on the principle of finding the direction with maximum free distance. He then applied evolutionary strategies to find the optimal set of parameters. Finally, **Diego Pérez Liébana** under the supervision of Yago Sáez and Pedro Isasi from the University Carlos III, in Madrid also followed a rule-based evolutionary approach. He represented the controller as a set of condition-action rules and then he applied a genetic algorithm to search for the optimal set of rules.

The entries were scored through a two stage process which involved three tracks that were unknown to the competitors. The first (warm up) stage was aimed at eliminating particularly bad performing controllers. Each controller raced alone in each of the three tracks and its performance was measured as the distance covered in 10000 game tics (approximately, 3 minutes and 20 seconds of actual game time). All the five submitted controllers performed well on this first task so that no controller was eliminated from the race. In the second stage, all the five controllers competed together in each of the three tracks. In this case, the goal was to complete three laps and each controller was scored based on its arrival order using the same point system used in F1: 10 points to the first controller that completed the three laps, 8 points to the second one, 6 to the third one, 5 to the fourth, and 4 to the fifth one. Ten runs for each track were performed. The score of a controller on one track was computed as the median of the scores obtained during the ten runs. The final score for each controller was finally computed as the sum of the points collected on each track. Matt Simmerson won the competition (congratulations!) and the final scoreboard was: Matt Simmerson 26 points, Leonard Kinnaird-Heether 22 points, Simon Lucas 20 points, Tan Chin Hiong 15 points, and Diego Pérez 14 points.

This has been the first racing competition based on TORCS and it has also been the first competition using a real-time client-server architecture. The performance of the five controllers is encouraging: two evolved controllers clearly outperformed the hand-coded ones that were provided by the organizers as examples. However, there is still a lot of work to do since the best submitted controllers are far from being competitive with human players. In particular, controllers that took part in the competition need to improve their overtaking capabilities their reliability on tracks that are different from the ones used for training.

And the winner is...

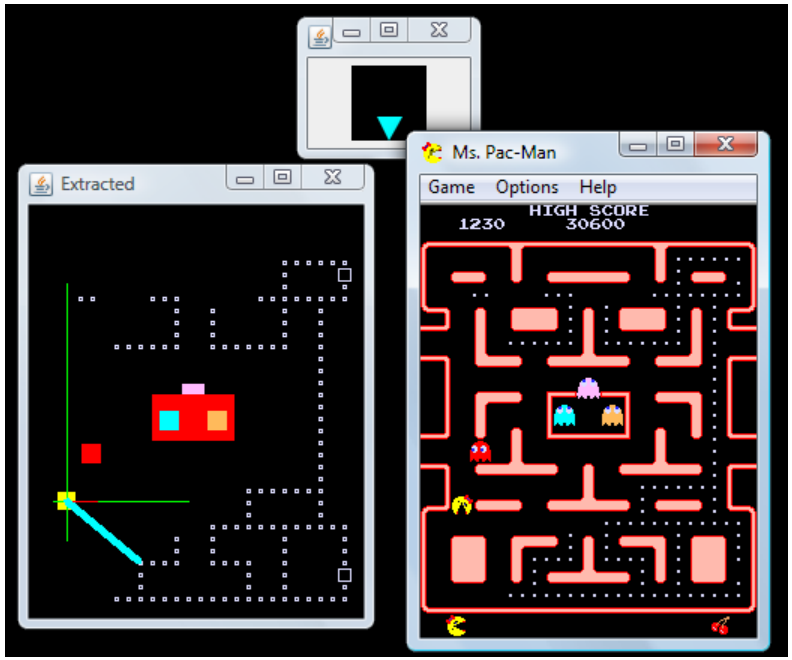
Although technical problems delayed the start of the competition, a number of good controllers were submitted. Most important, a large number of people attended the competition session of the conference and the several comments we received show that there is a large interest in this kind of competition. We believe that to be useful, for instance as a benchmark, the competition has to be run regularly so that the entrants can improve on their controllers and the same approaches can be tested on more challenges. Therefore, we are working on the organization of three new competitions, based on the same car racing simulator, that will be held during major conferences: one will be run this December during CIG-2008 (the IEEE Conference on Computational Intelligence in Games), one is planned for CEC-2009 (the IEEE Conference on Evolutionary Computation), and one is planned for GECCO-2009 (the ACM Genetic and Evolutionary Computation Conference). We hope that the next editions will attract more submissions especially from all the people who expressed their interested for this first competition but due to the tight schedule could not meet the deadline. We also hope to see the current entrants again in the next editions. Interestingly, the competition has also been assigned as homework to students, for instance, by Phil Hingston and Bobby Bryant. Therefore, we would be very glad to provide support to anyone who is interested in using our extension of TORCS for their courses. More information, including the documentation and the source code of the entries submitted to the competition, is available at competition web page.

The new edition of this competition will be held at CIG-2008 and it will be open for submission by the end of August!

Ms Pac-Man Competition

Simon M. Lucas, University of Essex, U.K.
Competition webpage: [WWW]

IEEE WCCI 2008 in Hong Kong played host to the latest Ms Pac-Man competition, organised by Simon Lucas as an activity of the IEEE CIS Games Technical Committee. The competition attracted 11 entries from teams all around the world, with the winning entry by Alan Fitzgerald, Peter Kemeraitis, and Clare Bates Congdon from the University of Southern Maine (USM) achieving a high-score of 15,970.



The agents play the standard game in screen capture mode (the screen is captured many times per second, the software agent has to process this to find the main game objects, and then interact with the game by generating key events). The figure illustrates the supplied software kit in operation, and shows the original game window, the game objects being extracted, a vector from the agent to the nearest food pill, and the currently selected direction.

The supplied controller provides a baseline performance: it simply always tried to head for the nearest food pill (nearest in terms of Euclidean distance). In the CEC 2007 competition, none of the entries were able to outperform this simple baseline on the day of the competition. The current entries have come a long way since then, and the leading entries regularly score around 15,000 points. That's significantly better than novice human players.

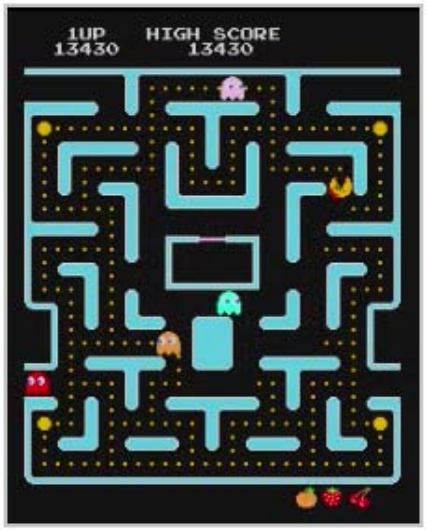
It's interesting to observe that the two leading entries have adopted very different strategies. The USM entry makes little effort to eat the ghosts when edible, whereas the second place entry (Gan, Liu and Bao) uses ghost-eating to great effect. While both entries achieve similar scores on average, I personally found the Gan entry more fun to watch. It frequently takes outrageous risks and often escapes from seemingly impossible situations, whereas the USM entry takes a more business-like approach to efficiently clear each level.

The leading entries didn't use any machine learning, though a good deal of hand-tuning has been used to make them operate effectively, and many of the entrants plan to use machine learning techniques in the future. The USM entry has all the hooks in place to use evolutionary algorithms to adapt the parameters of its controller, so hopefully we'll see that put to good effect for the CIG 2008 contest.

While the leading entries (especially Gan) play exceedingly well at times, they often lose lives through making apparently stupid decisions - running straight into ghosts for example, when not even trapped. However, when judging the behaviour it's important to realise that the software agents operate on a slightly delayed version of the game (having been through the screen capture and image parsing process). Therefore, a decision that appears to be incomprehensible may have made good sense 20 milliseconds ago!

These entries significantly outperform previous attempts at developing software agents for the game based on machine learning (e.g. [1] and [2]). While it should be noted that both [1] and [2] used different implementations of Ms. Pac-Man, both of these were (in the opinion of the author) easier and less entertaining versions than the original. Also, both [1] and [2] had direct access to the game state, and did not suffer the timing uncertainties inherent in the screen capture mode of operation.

In the screenshot below the Gan agent is about to commit suicide (give up the ghost) by running straight into "pinky", under no threat whatsoever! Prior to this the agent demonstrated significant skill in eating all four ghosts, and in surviving some death-defying situations.



While these entries represent the state of the art in software agent Ms. Pac-Man, they are still light-years away from the best human players. The two leading human players have high scores of over 900,000. Expert human players have internalised excellent models of the ghosts, and are able to predict their next steps rapidly and with a high degree of accuracy, even though the ghosts in Ms Pac-Man move non-deterministically. No one really knows how hard it is in terms of human endeavour, for example in comparison to being a chess grand-master. However, anyone who has played the game extensively will probably agree scoring over 100,000 is difficult – 900,000 is truly phenomenal. Only time will tell how hard a challenge it proves to be for software agents, but we await the IEEE CIG 2008 (December, Perth, WA) competition with great anticipation!

For more information we refer the reader to the competition website.

Bibliography

[1] Simon M. Lucas, *Evolving a Neural Network Location Evaluator to Play Ms. Pac-Man*, IEEE Symposium on Computational Intelligence and Games (2005), pages: 203–210

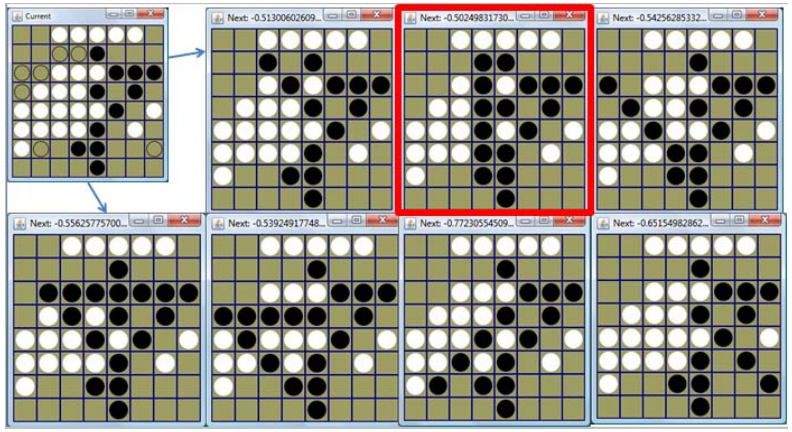
[2] Szita and A. Lorincz, *Learning to Play Using Low-Complexity Rule-Based Policies: Illustrations through Ms. Pac-Man*, Journal of Artificial Intelligence Research (2007), Volume 30, pages 659-684.

Neural Network Othello Competition

Simon M. Lucas, University of Essex, U.K.
Competition webpage: [WWW]

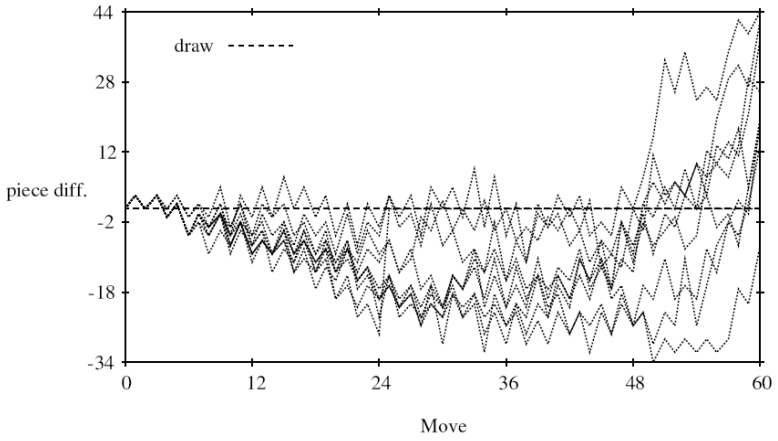
IEEE WCCI 2008 also saw the latest instalment of the Neural Network Othello competition. The aim of this is to find the best performing neural network (or more generally, any form of value function) for playing Othello. The mode of operation is as follows. Entrants submit their trained architectures to the competition web server, which evaluates them against a standard heuristic player, and immediately updates a league table with the performance of the submitted player. The leading entry by each entrant is then played off in a round robin league against the other leading entries to determine the competition winner. The competition server has been running for over two years now, and has received well over 1,000 entries. The IEEE CEC 2006 entry was won by an MLP (with a single hidden layer), submitted by Kyung-Joong Kim and Sung-Bae Cho of Yonsei University, Seoul, South Korea. In the league table below this entry is referred to as CEC 2006 Champ.

The mode of operation of each entry is as follows. An Othello engine plays out the game at 1-ply i.e. each move is made by expanding the game tree one level (by applying all possible legal moves from the current board state), then choosing the move that the current ‘player’ gives the highest value to. This is illustrated in the next figure. The current board state is shown in the top left, with black to move. All legal seven legal moves are applied from this position to form the set of seven after-states, and each of these states is evaluated by the value function playing as black. The move leading to the board outlined in red is chosen, as this has the highest value out of the seven. This process is repeated with the black and white function approximators taking turns until the game is concluded.



This mode of operation places all the emphasis on the computational intelligence involved, compared to a more general Othello competition where much of the playing strength stems from the game-tree search algorithms.

One of the aims of the competition is to evaluate the performance of various types of function approximation architecture. So far we have seen MLPs clearly outperform weighted piece counters, but in the 2008 competition we now witnessed other architectures clearly outperform MLPs. Othello is an ideal game for evaluating computational intelligence methods, as the rules are extremely simple, but as yet the game is unsolved.



Furthermore, it has a very deceptive nature, whereby novice players can appear to do well during the middle of the game, only to be thrashed in the end game. The previous figure illustrates this, showing the piece difference between a weak player and a strong player during 10 games (the strong player's piece difference is shown positive). In this case to play well the strong player has to learn to give it's pieces away during this mid game!

The runner-up entry (Spatial MLP) was submitted by Łukasz Czarzasty, Łukasz Dunal, Dagmara Kurpiewska, Marcin Walczyk and Jacek Mańdziuk from the Warsaw University of Technology, Poland, and consisted of a spatial MLP similar to the type used in [1] and [2], trained with co-evolution. The Warsaw team made a slight modification from [1] and [2], in that the piece difference value (passed straight to the output neuron) was scaled by an evolvable weight – in the standard version it is passed unscaled.

The winning entry (Ed Big) was by Ed Manning of Brookdale College, New Jersey. He used the approach described in [3] — a symmetric N-tuple network trained using temporal difference learning. The entries were played off against each other with three values of epsilon (the probability of a forced random move) – 0, 1%, and 10%. The results below are for 10%, but Ed Big came top of the league for all three cases. Simple SNT was the sample symmetric N-tuple network supplied on the competition web site. In this league Ed Big just scrapes ahead of Simple SNT (not statistically significant), but overall it is a significantly better player when taking the other leagues into account.

```
10% Random Moves
0   700   500   19   181   Ed Big
1   700   498   16   186   Simple SNT
2   700   387   20   293   Spatial MLP
3   700   353   17   330   CEC 2006 Champ
4   700   346   25   329   MLP Test
5   700   272   18   410   Heuristic(0)
6   700   205   30   465   Symmetric WPC
7   700   152   29   519   CEC 2007 Entry
```

In conclusion, the results show that the choice of function approximation architecture plays a major role in how well a system can learn to play Othello, a result which is likely to transfer to other games. So far, the best results by far have been obtained with symmetric N-tuple networks.

This is an interesting result in itself, given the fact that most researchers in this area are still using more conventional MLP type neural networks to solve this and similar problems. The learning algorithm of course plays a critical part, and the use of temporal difference learning probably played a key role in this entry's success.

The competition web server will continue to accept entries, and researchers are free to propose new types of architecture. For more details of the current architectures and the competition results, please refer to the competition website.

Bibliography

[1] K. Chellapilla and D. Fogel, *Evolving an expert checkers playing program without using human expertise*, IEEE Transactions on Evolutionary Computation, vol. 5, pp. 422 – 428, 2001.

[2] S. Y. Chong, M. K. Tan, and J. D. White, *Observing the evolution of neural networks learning to play the game of Othello*, IEEE Transactions on Evolutionary Computation, vol. 9, pp. 240 – 251, 2005.

[3] Lucas, S.M. *Learning to Play Othello with N-Tuple Systems*, Australian Journal of Intelligent Information Processing Systems, Special Issue on Game Technology, Vol 9, No. 4 pp 01-20, 2007.

New Issues of Journals

Evolutionary Intelligence 1(2) (www)

- **Special issue on artificial immune systems**, Uwe Aickelin, pp 83-84 (pdf)
- **The DCA: SOMe comparison: A comparative study between two biologically inspired algorithms**, Julie Greensmith, Jan Feyereisl and Uwe Aickelin, pp 85-112 (pdf)
- **Improving the reliability of real-time embedded systems using innate immune techniques**, Nicholas Lay and Iain Bate, pp 113-132 (pdf)
- **Evolutionary algorithms to simulate the phylogenesis of a binary artificial immune system**, Grazziela P. Figueredo, Luis A. V. de Carvalho, Helio J. C. Barbosa and Nelson F.F. Ebecken, pp 133-144 (pdf)
- **Frequency analysis for dendritic cell population tuning**, Robert Oates, Graham Kendall and Jonathan M. Garibaldi, pp 145-157 (pdf)
- **Exploratory data analysis with artificial immune systems**, Ying Wu and Colin Fyfe, pp 159-169 (pdf)

Calls and Calendar

July 2008

GECCO 2008 - Genetic and Evolutionary Computation Conference

July 12-16, 2008, Atlanta, Georgia, USA

Homepage: http://www.sigevo.org/gecco-2008

Conference Program

July 12	Pre-conference free workshops and tutorials
July 13	Pre-conference free workshops and tutorials; in the evening, opening reception
July 14	Presentations: reviewed papers, late breaking papers, Evolutionary Computation in Practice, competitions
July 15	Presentations: reviewed papers, late breaking papers, Evolutionary Computation in Practice, competitions; in the evening, poster session and reception
July 15	Poster Session and reception
July 16	Presentations: reviewed papers, late breaking papers, Evolutionary Computation in Practice, competitions

September 2008

New Journal: IEEE Transactions On Computational Intelligence And AI In Games

Homepage: http://ieee-cis.org/pubs/tciaig/ (available soon)

Submissions open August/September 2008

The IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI in GAMES (T-CIAIG), published four times a year, publishes archival journal quality original papers in computational intelligence and related areas in artificial intelligence applied to games, including but not limited to video games, mathematical games, human-computer interactions in games, and games involving physical objects. Emphasis will also be placed on the use of these methods to improve performance in and understanding of the dynamics of games, as well as gaining insight into the properties of the methods as applied to games. It will also include using games as a platform for building intelligent embedded agents for the real world. Papers connecting games to all areas of computational intelligence and traditional AI will be considered.

The journal is co-sponsored by the IEEE Computational Intelligence Society, the IEEE Computer Society, the IEEE Consumer Electronics Society and the IEEE Sensors Council. It is technically co-sponsored by the IEEE Systems, Man, and Cybernetics Society, the IEEE Instrumentation and Measurement Society, the IEEE Robotics and Automation Society, and the IEEE Communications Society.

The Journal will begin accepting submissions in August/September 2008, with the first issue to be published in March 2009. The journal is expected to rapidly establish itself as the leading publication in the field, following in the tradition of many other high quality IEEE Transactions.

For more information contact the Editor in Chief, Simon M. Lucas, University of Essex, UK, sml@essex.ac.uk.

PPSN 2008 - Parallel Problem Solving from Nature

September 13-17, 2008, Dortmund, Germany

Homepage: http://www.ppsn2008.org/

Call for paper: download

PPSN X will showcase a wide range of topics in Natural Computing including, but not restricted to: Evolutionary Computation, Quantum Computation, Molecular Computation, Neural Computation, Artificial Life, Swarm Intelligence, Artificial Ant Systems, Artificial Immune Systems, Self-Organizing Systems, Emergent Behaviors, and Applications to Real-World Problems.

Paper Presentation

Following the now well-established tradition of PPSN conferences, all accepted papers will be presented during small poster sessions of about 16 papers. Each session will contain papers from a wide variety of topics, and will begin by a plenary quick overview of all papers in that session by a major researcher in the field. Past experiences have shown that such presentation format led to more interactions between participants and to a deeper understanding of the papers. All accepted papers will be published in the Proceedings.

ICES 2008 - 8th International Conference of Evolvable Systems: From Biology to Hardware

September 21-24, 2008. Prague, Czech Republic

Homepage: http://www.fit.vutbr.cz/events/ices2008

The 8th International Conference of Evolvable Systems (ICSE 2008) which will be held in Prague, September 21-24, 2008. Topics to be covered include, but are not limited to: Evolutionary hardware design Evolutionary circuit diagnostics and testing, Self-reconfiguring/repairing and fault tolerant systems, co-evolution of hybrid systems, generative and developmental approaches, embryonic hardware, hardware/software co-evolution, intrinsic and extrinsic evolution, real-world applications of evolvable hardware, on-line hardware evolution, MEMS and nanotechnology in evolvable hardware, evolutionary robotics, formal models for bio-inspired hardware systems adaptive computing, novel devices/testbeds/tools for evolvable hardware.

Sixth International Conference on Ant Colony Optimization and Swarm Intelligence

September 22–24, 2008. Brussels, Belgium

Homepage: http://iridia.ulb.ac.be/ants2008/

Swarm intelligence is a relatively new discipline that deals with the study of self-organizing processes both in nature and in artificial systems. Researchers in ethology and animal behavior have proposed many models to explain interesting aspects of social insect behavior such as self-organization and shape-formation. Recently, algorithms inspired by these models have been proposed to solve difficult computational problems.

An example of a particularly successful research direction in swarm intelligence is **ant colony optimization**, the main focus of which is on discrete optimization problems. Ant colony optimization has been applied successfully to a large number of difficult discrete optimization problems including the traveling salesman problem, the quadratic assignment problem, scheduling, vehicle routing, etc., as well as to routing in telecommunication networks. Another interesting approach is that of **particle swarm optimization**, that focuses on continuous optimization problems. Here too, a number of successful applications can be found in the recent literature. [...]

ANTS 2008 will give researchers in swarm intelligence the opportunity to meet, to present their latest research, and to discuss current developments and applications.

The three-day conference will be held in Brussels, Belgium, on September 22–24, 2008. Tutorial sessions will be held in the mornings before the conference program.

Further Information

Up-to-date information will be published on the web site http://iridia.ulb.ac.be/ants2008/. For information about local arrangements, registration forms, etc., please refer to the above-mentioned web site or contact the local organizers at the address below.

Conference Address

ANTS 2008
IRIDIA CP 194/6 — Tel +32-2-6502729
Université Libre de Bruxelles — Fax +32-2-6502715
Av. F. D. Roosevelt 50 — http://iridia.ulb.ac.be/ants2008
1050 Bruxelles, Belgium — email: ants@iridia.ulb.ac.be

December 2008

CIG 2008 - IEEE Symposium on Computational Intelligence and Games

December 15-18, 2008, Perth, Australia

Homepage: http://www.csse.uwa.edu.au/cig08/

Deadline July 15, 2008

Submission website: [WWW]

Games have proven to be an ideal domain for the study of computational intelligence as not only are they fun to play and interesting to observe, but they provide competitive and dynamic environments that model many real-world problems. This symposium, sponsored by the IEEE Computational Intelligence Society with technical co-sponsorship from the IEEE Consumer Electronics Society, aims to bring together leading researchers and practitioners from both academia and industry to discuss recent advances and explore future directions in this field.

- Learning in games
- Coevolution in games
- Neural-based approaches for games
- Fuzzy-based approaches for games
- Opponent modelling in games
- Theoretical or empirical analysis of computational intelligence techniques for games
- Comparative studies (e.g. evolved players versus human-designed players or other learning algorithms)
- Multi-agent and multi-strategy learning
- Applications of game theory
- Board and card games
- Economic or mathematical games
- Imperfect information and non-deterministic games
- Evasion (predator/prey) games
- Console and video games
- Realistic games for simulation or training purposes
- Player satisfaction in games
- Games for mobile or digital platforms
- Games involving control of physical objects
- Games involving physical simulation

The symposium will consist of a single track of oral presentations, tutorial and workshop/special sessions, and live competitions. The proceedings will be published by the IEEE and made freely available on this website after the symposium.

The paper submission is now open.

January 2009

FOGA X - Foundations of Genetic Algorithms

January 9-11, 2009, Orlando, Florida USA

Homepage: http://www.sigevo.org/foga-2009

Deadline August 18, 2008

We invite submissions of extended abstracts for the tenth Foundations of Genetic Algorithms, sponsored by ACM SIGEVO. FOGA focuses on theoretical foundations of evolutionary computation. It will be held from Friday, January 9 until Sunday January 11, 2009 in Orlando, Florida in the USA. Attendance will be limited to individuals who have submitted papers, or those requesting attendance in advance. Students are particularly encouraged to participate. Submissions should address theoretical issues in evolutionary computation. Papers that consider foundational issues, place analysis in the context of the wider community of theoretical computer science, or focus on bridging the gap between theory and practice are particularly encouraged. These topics do not preclude the acceptance of papers that use an experimental approach, but such work should be directed toward validation of suitable hypotheses concerning foundational matters.

Extended abstracts should be between 10-12 pages long. To submit, please email a compressed postscript or a PDF file to foga09@ist.ucf.edu no later than Monday, August 18, 2008. In their submission message, authors should provide the title of the paper, as well as the name, address and affiliation of the author(s).

Submitted papers should use standard spacing and margins, with 11pt or 12pt font for the main text. Authors using LATEX should either use the standard article style file or the FOGA style file which can be found at the conference web-site. A double-blind reviewing process will be employed, so authors are asked to remove references to themselves from their paper. Notification will be in early November, and drafts of the full paper will be due one month after notification. These drafts will be distributed as part of a preprint to participants at FOGA. Authors of papers presented at the FOGA workshop will be asked to contribute final versions of their papers (based on discussion/feedback at the meeting) as part of the final volume.

Further Information

Extended abstracts due	August 18, 2008
Requests for attendance due	September 1, 2008
Notification of acceptance	early November, 2008
Full papers due	early December, 2008
FOGA Workshop	January 9-11, 2009

Organizing Committee

Thomas Jansen,	Thomas.Jansen@tu-dortmund.de
Ivan Garibay,	igaribay@cs.ucf.edu
R. Paul Wiegand,	wiegand@ist.ucf.edu
Annie S. Wu,	aswu@cs.ucf.edu

Further Information

Enquiries and submissions: foga09@ist.ucf.edu

2009 EVO*
www.evostar.org
tübingen::germany

April 2009

Evostar 2009 - EuroGP, EvoCOP, EvoBIO and EvoWorkshops
April 15-17, 2009, Tübingen, Germany
Homepage: www.evostar.org
Important Dates for all events are:
Deadline November 5, 2008

The EuroGP, EvoCOP and EvoBIO conferences and the workshops collectively entitled EvoWorkshops compose EVO*: Europe's premier co-located events in the field of Evolutionary Computing. Featuring the latest in theoretical and applied research, EVO* topics include recent genetic programming challenges, evolutionary and other meta-heuristic approaches for combinatorial optimisation, evolutionary algorithms, machine learning and data mining techniques in the biosciences, in numerical optimisation, in music and art domains, in image analysis and signal processing, in hardware optimisation and in a wide range of applications to scientific, industrial, financial and other real-world problems.

EuroGP

Twelfth European Conference on Genetic Programming: high quality papers are sought on topics strongly related to the evolution of computer programs, ranging from theoretical work to innovative applications.

EvoCOP

Ninth European Conference on Evolutionary Computation in Combinatorial Optimisation: practical and theoretical contributions are invited, related to evolutionary computation techniques and other meta-heuristics for solving combinatorial optimisation problems.

EvoBIO

Seventh European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics: the emphasis is on evolutionary computation and other advanced techniques addressing important problems in molecular biology, proteomics, genomics and genetics, that have been implemented and tested in simulations and on real-life datasets.

EvoWorkshops

The twelve workshops which make up this event are focused on the use of Evolutionary Computation in different application areas:

- EvoCOMNET: Telecommunication networks and other parallel and distributed systems
- EvoENVIRONMENT: Environmental issues
- EvoFIN: Finance and economics
- EvoGAMES: Games
- EvoHOT: Design automation
- EvoIASP: Image analysis and signal processing
- EvoINTERACTION: Interactive evolution and humanized computational intelligence
- EvoMUSART: Music, sound, art and design
- EvoNUM: Continuous parameter optimisation
- EvoPHD: Graduate student workshop on evolutionary computation
- EvoSTOC: Stochastic and dynamic environments
- EvoTRANSLOG: Transportation and logistics

In 2009, the event will take place in Tübingen, a traditional university town in Baden-Württemberg, Germany, situated on a ridge between the Neckar and Ammer rivers in the southwest of the country, about 30 kms southwest of Stuttgart. EVO* 2009 will be hosted at Eberhard Karls University in Tübingen, founded in 1477 and one of the oldest universities in Germany.

EVO* 2009 proceedings will be published by Springer Verlag in the Lecture Notes in Computer Science series.

The website www.evostar.org offers information relevant to all events, including calls for papers, deadlines, organising committees, submission requirements, local information and a thorough view on the previous editions.

May 2009

2009 IEEE Congress on Evolutionary Computation (CEC 2009)

May 18-21, 2009, Trondheim, NORWAY

Homepage: www.cec-2009.org

Deadline November 1, 2007

The 2009 IEEE Congress on Evolutionary Computation (CEC 2009) will be at the Nova Conference Centre and Cinema, Trondheim, Norway during May 18-21, 2009. Sponsored by the IEEE Computational Intelligence Society, co-sponsored by the Evolutionary Programming Society and the Institution of Engineering and Technology, CEC 2009 continues the successful sequence of World-class events going back to 1999.

CEC 2009 will feature a world-class conference that will bring together researchers and practitioners in the field of evolutionary computation and computational intelligence from all around the globe. Technical exchanges within the research community will encompass keynote speeches, special sessions, tutorials, panel discussions as well as poster presentations. On top of these, participants will be treated to a series of social functions, receptions and networking sessions, which will serve as a vital channel to establish new connections and foster everlasting friendship among fellow researchers. The annual IEEE Congress on Evolutionary Computation (CEC) is one of the leading events in the area of evolutionary computation. CEC covers all topics in evolutionary computation, including, but not limited to:

- Ant colony optimization
- Artificial immune systems
- Artificial life
- Autonomous mental & behaviour development
- Bioinformatics & bioengineering

- Coevolution & collective behaviour
- Cognitive systems & applications
- Combinatorial & numerical optimization
- Computational finance & economics
- Constraint & uncertainty handling
- Estimation of distribution algorithms
- Evolutionary data mining
- Evolutionary design
- Evolutionary games
- Evolvable hardware & software
- Evolutionary learning systems
- Evolving neural networks & fuzzy systems
- Molecular & quantum computing
- Particle swarm intelligence
- Representation & operators

Researchers are invited to contribute high-quality papers to CEC 2009. All papers are to be submitted electronically through the Congress website by November 1, 2008. All submitted papers will be refereed by experts in the fields based on the criteria of originality, significance, quality, and clarity. In addition, we are looking for high quality proposals for Special Sessions and Tutorials for the Congress. More details on all of all of these are below.

Call for Contributed Papers

Prospective authors are invited to contribute high-quality papers to CEC2009. All papers are to be submitted electronically through the Congress website. For general inquiries, please contact General Chair Andy Tyrrell at amt@ohm.york.ac.uk. For program inquiries, contact Program Chair Pauline Haddow at Pauline.Haddow@idi.ntnu.no.

Call for Special Sessions

CEC 2009 Program Committee solicits proposals for special sessions within the technical scopes of the congress. Special sessions, to be organised by internationally recognised experts, aim to bring together researchers in special focused topics. Papers submitted for special sessions are to be peer-reviewed with the same criteria used for the contributed papers. Researchers interested in organising special sessions are invited to submit formal proposals to the Special Session Chair Jon Timmis at jt517@ohm.york.ac.uk. A special session proposal should include the session title, a brief description of the scope and motivation, names, contact information and brief CV of the organisers.

Call for Tutorials

CEC 2009 will also feature pre-congress tutorials covering fundamental and advanced computational intelligence topics. A tutorial proposal should include title, outline, expected enrollment and presenter biography. Tutorials are expected to run for 2 hours each. Researchers interested in organising tutorials are invited to submit formal proposals to the Tutorial Chair Stephen Smith at sls@ohm.york.ac.uk.

Important Dates:

- Special Session proposals: September 1, 2008
- Paper submissions: November 1, 2008
- Tutorial proposals: December 1, 2008
- Notification of acceptance: January 16, 2009
- Final paper submission: February 16, 2009

More information can be found at: www.cec-2009.org.

For general inquiries, please contact General Chair Andy Tyrrell at amt@ohm.york.ac.uk.

About the Newsletter

SIGEVOlution is the newsletter of SIGEVO, the ACM Special Interest Group on Genetic and Evolutionary Computation.

To join SIGEVO, please follow this link [WWW]

Contributing to SIGEVOlution

We solicit contributions in the following categories:

Art: Are you working with Evolutionary Art? We are always looking for nice evolutionary art for the cover page of the newsletter.

Short surveys and position papers: We invite short surveys and position papers in EC and EC related areas. We are also interested in applications of EC technologies that have solved interesting and important problems.

Software: Are you are a developer of an EC software and you wish to tell us about it? Then, send us a short summary or a short tutorial of your software.

Lost Gems: Did you read an interesting EC paper that, in your opinion, did not receive enough attention or should be rediscovered? Then send us a page about it.

Dissertations: We invite short summaries, around a page, of theses in EC-related areas that have been recently discussed and are available online.

Meetings Reports: Did you participate to an interesting EC-related event? Would you be willing to tell us about it? Then, send us a short summary, around half a page, about the event.

Forthcoming Events: If you have an EC event you wish to announce, this is the place.

News and Announcements: Is there anything you wish to announce? This is the place.

Letters: If you want to ask or to say something to SIGEVO members, please write us a letter!

Suggestions: If you have a suggestion about how to improve the newsletter, please send us an email.

Contributions will be reviewed by members of the newsletter board.

We accept contributions in LaTeX, MS Word, and plain text.

Enquiries about submissions and contributions can be emailed to editor@sigevolution.org.

All the issues of SIGEVOlution are also available online at www.sigevolution.org.

Notice to Contributing Authors to SIG Newsletters

By submitting your article for distribution in the Special Interest Group publication, you hereby grant to ACM the following non-exclusive, perpetual, worldwide rights:

- to publish in print on condition of acceptance by the editor
- to digitize and post your article in the electronic version of this publication
- to include the article in the ACM Digital Library
- to allow users to copy and distribute the article for noncommercial, educational or research purposes

However, as a contributing author, you retain copyright to your article and ACM will make every effort to refer requests for commercial use directly to you.