Evolution Strategy for Optimizing Parameters in Ms Pac-Man Controller ICE Pambush 3

Ruck Thawonmas, *Senior Member, IEEE* and Takashi Ashida

***Abstract*— This paper describes an application of Evolutionary Strategy to optimizing ten distance parameters and seven cost parameters in our Ms Pac-Man controller, ICE Pambush 3, which was the winner of the IEEE CIG 2009 competition. Targeting at the first game level, we report our results from 14 possible optimization schemes; arising from combinations of which initial values to chose, those originally used in ICE Pambush 3 or those randomly assigned, and which parameter types to optimize first, the distance parameters, the cost parameters, or both. We have found that the best optimization scheme is to first optimize the distance parameters, with their initial values set to random values, and then the cost parameters with their initial values set to random values. The optimized ICE Pambush 3 using the resulting parameters from this optimization scheme has an improvement of 17% in the performance for the first game level, compared to the original ICE Pambush 3.**

I. Introduction

Ms Pac-Man serves as a challenging platform for research in computational intelligence and artificial intelligence. Main reasons for this include (1) the four different semi-random moving ghosts, (2) the short response time of 67 milliseconds to interact with the game, and (3) the inaccessibility of game circumstance information available in the game engine. A series of Ms Pac-Man controller competitions [1] have been held since 2007 for providing an opportunity to compete among the state-of-the-art controllers and to share recent research findings.

ICE Pambush 3 [2] is our rule-based controller that won the IEEE CIG 2009 competition [3]. As also stated at the competition site, the causes of our victory are due to (1) effective and efficient game-object extraction from the screen, (2) an implementation for luring the ghosts to a location near a power pill and then ambushing them, and (3) some luck. Luck was in fact a crucial factor in manual selection of all parameters used in ICE Pambush 3, through repetitions of heuristic setting of those parameters and then testing them. Although techniques in computational intelligence or artificial intelligence were not fully exploited, ICE Pambush 3 can serve as a baseline controller for advanced controllers exploiting such techniques to compare with.

In this paper, we present an effective application of Evolutionary Strategy (ES) to optimizing ten distance parameters and seven cost parameters in ICE Pambush 3. The contributions of this work are as follows:

The authors are with the Intelligent Computer Entertainment Laboratory, Graduate School of Science and Engineering, Ritsumeikan University, Shiga, Japan (phone: +81-77-561-5048; fax: +81-77-561-5203; email: ruck@ci.ritsumei.ac.jp).

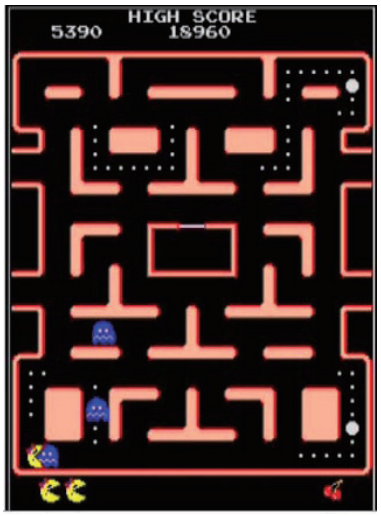

Fig. 1. Screenshot of Ms Pac-Man when she is eating an edible ghost

1) Description of the best optimization scheme whose performance is superior to that of the original ICE Pambush 3,
2) The values of the resulting distance and cost parameters, and
3) The outline of the controller itself.

The remainder of this paper is organized as follows. First, we present an overview of Ms Pac-Man and relevant related work. Next we describe an outline of ICE Pambush 3 and its parameter optimization schemes. Then we present and discuss the evaluation results.

II. Ms Pac-Man and Related Work

Ms Pac-Man (cf. Fig. 1), whose main character is a female one wearing a red ribbon on its head, was released in 1981 as a variant of Pac-Man. The ghosts in Ms Pac-Man behave almost non-deterministically, making it difficult to use preset patterns to clear each game level where pills as well as four power-pills reside, and fruits appear. Major game rules are as follows:

- In order to finish a given game level and proceed to the next one, Ms Pac-Man must eat all available pills and power-pills.
- When Ms Pac-Man is hit by a ghost, its life value, initially set to 3, is decremented, and this value is added by one when the score reaches ten thousand,
- If Ms Pac-Man eats a power pill, all available ghosts outside of the ghost cage will become edible for a

978-1-4244-6297-1/10/$26.00 ©2010 IEEE

certain period, during which the more edible ghosts that Ms Pac-Man eats, the higher score will be earned from each of them.
- If Ms Pac-Man eats a fruit, a score will be earned according to the fruit type.

As a pioneer work on Ms Pac-Man controllers, Evolutionary Strategy (cf. [4]) was used to train a neural network controller [5]. The same group recently compared Temporal Difference Learning and Evolution approaches for this task in [6]. Fuzzy techniques were also exploited together with Evolutionary Strategy and Q-Learning in [7] and [8], respectively. Other recent work includes research focusing on map circumstances in [9] and [10], a controller based on simple-tree-search in [11], and an attempt to learn low-complexity play policies in [12].

A very recent work in [13] used Grammatical Evolution for evolving both rules and parameters of Ms Pac-Man. The performance of this approach on level 1 against three different ghost teams running on a simulator was impressing even with only one life. The performance on the real game, however, remains unknown.

III. Outline of ICE Pambush 3

Our controller moves Ms Pac-Man along the path that has the lowest cost between its current grid position and the current target location. ICE Pambush 3 reuses the image processing mechanism in the previous controller ICE Pambush 2 [14], which won the IEEE CEC 2009 competition, for extraction of the current game circumstance information from the game screen. For path finding, in order to suppress the search time, we adopted two versions of Depth-First Search (DFS): DFS-A and DFS-B.

DFS-A and DFS-B use different maximum depths m and different number of factors considered in cost computation. DFS-A searches with the smaller m but considers more factors while DFS-B searches with the larger m but considers less factors. The former is invoked when "a ghost is nearby" AND "the situation is critical" while the latter is used when ghosts are far from Ms Pac-Man. At each iteration, the following nine rules for determining the target location are examined in this order, and the first rule that holds will be fired, where $D_1, D_2, \ldots, D_{10}$, denote the distance parameters.

Rule 1:
IF
$d(nearest\_power\_pill) \leq D_2$
AND
$D_3 \leq d(nearest\_ghost) \leq D_1$
AND
$d(ghost\_nearest\_to\_nearest\_power\_pill) \geq D_4$,

THEN set the "ambush state" to ON and stop moving at the corner or the cross point near the nearest power pill waiting for a ghost to come closer, where $d(nearest\_power\_pill)$ is the distance from Ms Pac-Man to the nearest power pill, $d(nearest\_ghost)$ the distance from Ms Pac-Man to the nearest ghost, and $d(ghost\_nearest\_to\_nearest\_power\_pill)$ the distance from Ms Pac-Man to the ghost nearest to the power pill nearest to Ms Pac-Man.

Rule 2:
IF
at least one power pill exists
AND
no edible ghost exists
AND
$d(nearest\_ghost) \leq D_1$
AND
the "ambush state" is ON
AND
$(d(nearest\_ghost) \leq D_3$
OR
$d(ghost\_nearest\_to\_nearest\_power\_pill) \leq D_4)$,

THEN move to the nearest power pill with DFS-B.

Rule 3:
IF
at least one power pill exists
AND
no edible ghost exists
AND
$d(nearest\_ghost) \leq D_1$
AND
$d(nearest\_power\_pill) \leq D_5$,

THEN set the "ambush state" to OFF and move to the nearest power pill with DFS-B.

Rule 4:
IF
at least one power pill exists
AND
no edible ghost exists
AND
$d(nearest\_ghost) \leq D_1$,

THEN set the "ambush state" to OFF and move to the nearest power pill with DFS-A.

Rule 5:
IF
at least one edible ghost exists
AND
$d(nearest\_ghost) \leq D_1$
AND
$d(nearest\_edible\_ghost) \leq D_6$,

THEN move to the nearest edible ghost with DFS-A, where $d(nearest\_edible\_ghost)$ is the distance from Ms Pac-Man to the nearest edible ghost.

Rule 6:
IF
at least one pill exists
AND
$d(nearest\_ghost) \leq D_1$,

THEN set the "ambush state" to OFF and move to the nearest pill with DFS-A.

Rule 7:
IF
at least one edible ghost exists
AND
$d(nearest\_ghost) > D_1$
AND
$d(nearest\_edible\_ghost) \leq D_7$,

THEN move to the nearest edible ghost with DFS-B.

Rule 8:
IF
at least one pill exists
AND
$d(nearest\_ghost) > D_1$
AND
(no pill# exists
OR $(d(nearest\_ghost) \geq D_8$
AND
$d(nearest\_power\_pill) \geq D_9$
AND
$d(pill) \leq D_{10}))$,

THEN move to the nearest pill with DFS-B, where pill# indicate those pills not lying between any pair of cross points whose connected path contains a power pill.

Rule 9:
IF
at least one pill exists
AND
$d(nearest\_ghost) > D_1$,

THEN move to the nearest pill# with DFS-B.

The parameter m in DFS specifies the search space covering all paths from the current grid of Ms Pac-Man to the corner or cross points (henceforth, called nodes) at level m. At a node at level i, where $i < m$, the current path will be expanded into four directions; i.e., north, east, south, or west; where the reverse direction and those towards the wall are excluded. In particular, m is set to 5 and 10 for DFS-A and DFS-B, respectively.

To find a path, DFS-A considers the distance cost, the ghost cost, and the corner cost while DFS-B considers only the distance cost. The Dijkstra distances between all grids were computed and stored in the related files in advance. At the beginning of each game, this distance information is loaded and used in our search algorithm. Our cost definitions, with seven cost parameters, are given as follows:

Ghost Cost for each of the four ghosts at node X away from the ghost:

$$= C_i / distance(Y\_to\_X)^2,$$

where C_i for $i = 1$ to 4 is the ghost cost parameter for Blinky, Pinky, Inky, and Sue, respectively, X the jth node from the ghost, $j = 1$ and 2, and $distance(Y\_to\_X)$ is the distance from the ghost to node X.

Ghost Cost II at a node where a ghost resides:

$$= C_i,$$

where C_i is the same parameter as above.

Ghost Cost III at a node behind a ghost chasing Ms Pac-Man on the same corridor:

$$= C_5,$$

where C_5 is the ghost cost III parameter.

Distance Cost at node X:

$$= C_6[distance(X)$$
$$+distance(X\_to\_target) - distance(target)],$$

where C_6 is the distance cost parameter, X is the ith-level node from Ms Pac-Man, $i = 1$ to 5 (DFS-A) or 1 to 10 (DFS-B), $distance(X)$ is the distance from Ms Pac-Man to X, $distance(X\_to\_target)$ is the distance from X to the target location, and $distance(target)$ is the distance from Ms Pac-Man to the target location.

Corner Cost at each corner:

$$= C_7,$$

where C_7 is the corner cost parameter.

IV. Parameter Optimization Schemes

For parameter optimization, we must find an answer to each of the following questions .

1) Which should we use between the real game and a simulator?
2) Should a same set of parameters be used for all game levels or a different set for each level?
3) How do we evaluate a given set of parameters (or parameter candidates)?
4) Which algorithm should be employed for this task?
5) Should we exploit the original parameters used in ICE Pambush 3 and optimize simultaneously the distance parameters and the cost parameters?

We adopt a web-version [15], which is one of the two versions recommended at the contest site, of Ms Pac-Man in this study. A number of research groups used a simulator in their work [10-13]. Although, the information of all necessary objects are readily accessible and the game time can be manipulated to decrease the wall clock time in parameter optimization, most simulators do not have exactly the same behavior as the two game versions used in the competition. Our primary reason for using the aforementioned web-version is that the performance of the controller with this version implies the performance at the competition.

We plan to use a different set of parameters for each game level. A same set of parameter was used in ICE Pambush 3. However, a capable player would readily notice that they

need to slightly change their tactics for each game level due to changes in ghost speeds, mazes, etc. In order to be able to use a different set of parameters for each game level, we needed and thus implemented an image processing module for detecting the beginning of each game level. In this work, we limit ourselves to parameter optimization for the first game level; according to our experience, the wall clock time to complete the first game level takes about 1 minute and 30 seconds.

We evaluate a given set of parameters for 10 games (thus taking approximately 15 minutes) and 100 games (approximately 150 minutes) for parameter optimization and for performance testing, respectively. For parameter optimization, our performance index is the median because it is more robust against outlier scores. However, for performance testing of a resulting set of optimized parameters, we use the minimum of the 90 maximum scores (Min-Max score); where each maximum score is the maximum score of each overlapping slice, consisting of 10 games and overlapping with the subsequent slice by 9 games. We consider that the Min-Max score is a performance index that should indicate the lower-bound performance at the competition, where the maximum score of 10 games is used.

We adopt Evolutionary Strategy (ES) [4], which only uses mutation, for parameter optimization. Our primary reason for this is that each parameter has a specific role and thus other evolutionary operations such as crossover might not fit for this problem. Because the time to proceed from one generation to the subsequent generation linearly increases with the number of ES members, we need to restrict the number of ES members in each generation. In particular, we use (1+1) ES where a mutated child, whose mutation rate is controlled by the 1/5 rule, is evaluated and compared with its single parent in order to decide who will survive for the next generation.

We divide parameter optimization into two stages. At the first optimization stage either the distance parameters, denoted by D, or the cost parameters, denoted by C, are optimized with two possible initial values: the ICE Pambush 3's original values, denoted by I, or random values, denoted by R. In addition, if the optimization targets are the distance parameters, the cost parameters are fixed to the ICE Pambush 3' original values, and vice versa. At the second optimization stage, the cost parameters are optimized if the first stage's optimization targets were the distance parameters, and vice versa. The resulting optimized parameters from the first optimization stage are fixed and the targeted parameters here are initialized either by the ICE Pambush 3's original values or random values.

Let $O(x, y)$ denote optimization of the x parameters with the initial values set to y, where x are either D or C, and y are either I or R. In addition, let $F(x, y)$ represent fixing of the x parameters with their values set to y, where x are either D or C, and y are either I (applicable only at the first optimization stage) or $V(z)$ (applicable only at the second optimization stage), $V(z)$ having the values optimized at the first optimization stage with the initial values z set to I or R. All possible optimization schemes are summarized by a pair of $F(x, y)$ and $O(x, y)$ as follows:

First Optimization Stage

1-1 $F(D, I)$ and $O(C, I)$
1-2 $F(D, I)$ and $O(C, R)$
1-3 $F(C, I)$ and $O(D, I)$
1-4 $F(C, I)$ and $O(D, R)$

Second Optimization Stage

2-1 $F(C, V(I))$ and $O(D, I)$
2-2 $F(C, V(I))$ and $O(D, R)$
2-3 $F(C, V(R))$ and $O(D, I)$
2-4 $F(C, V(R))$ and $O(D, R)$
2-5 $F(D, V(I))$ and $O(C, I)$
2-6 $F(D, V(I))$ and $O(C, R)$
2-7 $F(D, V(R))$ and $O(C, I)$
2-8 $F(D, V(R))$ and $O(C, R)$

Note that 2-1 and 2-2 follow 1-1, 2-3 and 2-4 follow 1-2, 2-5 and 2-6 follow 1-3, 2-7 and 2-8 follow 1-4. We also conduct the following two optimization schemes 0-1 and 0-2 where the distance parameters and cost parameters are optimized simultaneously.

0-1 $O(D, I)$ and $O(C, I)$
0-2 $O(D, R)$ and $O(C, R)$

V. Evaluation and Discussions

In our evaluation, all distance parameters have an integer value between 0 and 49, and all cost parameters have an integer value between 0 and 1000000. Each value is converted from a real value x having the range between 0 and 1. The initial value of x is randomly assigned. During the evolution process, x is mutated at each generation as follows:

$$x = x + N(0, \sigma^2)$$

where σ is set to 0.77. As mentioned in the previous section, the 1/5 rule is used for controlling the mutation rate or σ. In order to prevent too early changing of σ, we fix σ until the median score among 10 games of a child at a current generation, say, the ith generation, is greater than the average between the median score of the first generation parent and 14400 (the maximum score of level 1). From the ith generation, σ is updated as follows:

- If the number of times that a child survived is less than or equal to one for the last ten generations, then $\sigma = 0.85\sigma$;
- Otherwise, $\sigma = (1/0.85)\sigma$.

Figures 2, 3, and 4 show the median score among 10 games for each generation of the four schemes at the first optimization stage and the first four schemes as well as the last four schemes at the second optimization stage, respectively. Figure 5 shows the results for 0-1 and 0-2. At the first stage, scheme 1-4, in which the distance parameters were optimized from randomly assigned initial values, shows

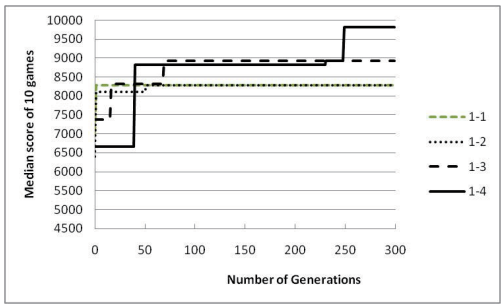

Fig. 2. Median-score evolution at the first optimization stage

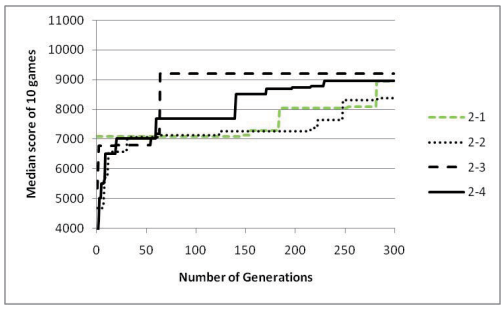

Fig. 3. Median-score evolution at the second optimization stage (the first four schemes)

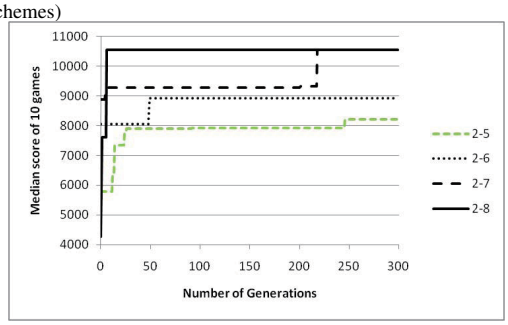

Fig. 4. Median-score evolution at the second optimization stage (the last four schemes)

the best performance with the median score of 9820 at the 300th generation. At the second stage, schemes 2-7 and 2-8, whose distance parameters were fixed to the values optimized by scheme 1-4 at the first stage, have the median score of 10545 and 10555, respectively. From these results, one could anticipate that schemes 2-7 and 2-8 would give a satisfactory result in performance testing.

Figure 6 shows the Min-Max score of each scheme. As expected, schemes 2-7 and 2-8 achieve a satisfactory performance with the Min-Max score of 8870 and 9480, respectively. In addition, scheme 2-8 is of the best performance and has an improvement of 17% in the performance, compared to the original ICE Pambush 3 whose Min-Max score is 8120. However, this figure also indicates that some schemes, such as 1-3, have lower performances than the original one although their performances during evolution monotonically increase. We conjecture that each of those

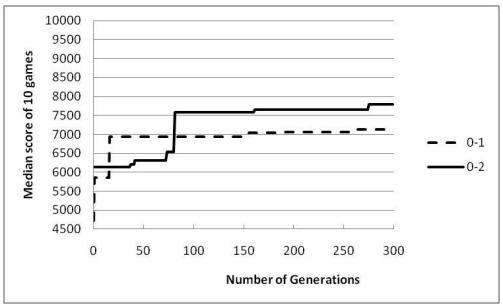

Fig. 5. Median-score evolution of 0-1 and 0-2

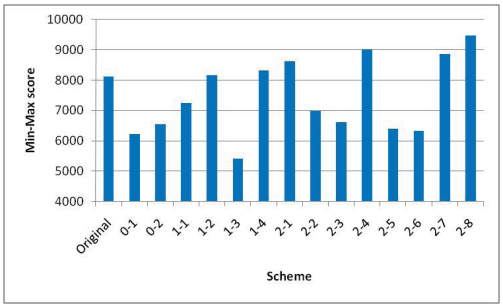

Fig. 6. The testing performance in terms of the Min-Max scores

schemes was trapped in a local optimum.

Figure 7 compares the score distributions, out of 100 games, of the original ICE Pambush 3 and scheme 2-8. It can be seen that the score distribution of scheme 2-8 concentrates more on the upper half while that of the original ICE Pambush 3 on the lower half. This result also confirms the effectiveness of scheme 2-8.

Table I shows the set of distance parameters used in the original ICE Pambush 3 and the one resulting from scheme 2-8 at the 300th generation while Table II shows the cost parameters. Surprisingly, because D_1 is less than D_3 for scheme 2-8, rule 1 will never be fired. This indicates that ICE Pambush 3 using the set of parameters from this scheme does no longer perform a stop and ambush at the first game level. Regarding the cost, because C_7 has been drastically increased, ICE Pambush 3 with the parameters from scheme 2-8 will go to a corner with a much lower frequency than the original ICE Pambush 3. As a result, although the same set of rules is used, optimization of the distance and cost parameters leads to another playing style.

VI. Conclusions and Future Work

This work described an attempt to increase the performance of our Ms Pac-Man controller ICE Pambush 3 by optimizing its parameters using Evolutionary Strategy. Our findings indicate that the best performance is from ICE Pambush 3 with the set of parameters obtained from scheme 2-8. In addition, this set of parameters makes Ms Pac-Man behave differently from the original one at the first game level.

TABLE I
DISTANCE PARAMETERS

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
Original	8	5	4	6	4	10	10	9	20	10
2-8	5	16	9	41	44	3	31	38	10	31

TABLE II
COST PARAMETERS

	C_1	C_2	C_3	C_4	C_5	C_6	C_7
Original	500000	500000	500000	500000	500000	1000	5000
2-8	493141	765404	196308	741495	287633	6296	179269

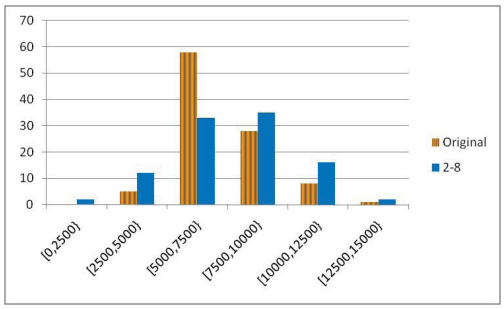

Fig. 7. Score distribution of ICE Pambush 3 using the original parameter set and the one resulting from scheme 2-8

We are currently extending parameter optimization to other subsequent levels and attempting to circumvent the local optima issue. As future work, we plan to evolve some, if not all, rules using a hybrid of computational intelligence techniques. In addition, we plan to explore supervised learning techniques for training the controller in certain situations, such as critical situations.

ACKNOWLEDGMENTS

The authors would like to thank all other members of the Ms Pac-Man group in the laboratory; in particular, Hiroshi Matsumoto, the leading member of the ICE Pambush 3 team, for their helps and fruitful comments. This work was supported in part by the MEXT Global COE Program – Digital Humanities Center for Japanese Arts and Cultures, Ritsumeikan University.

REFERENCES

[1] Ms Pac-Man Competition, http://cswww.essex.ac.uk/staff/sml/pacman/PacManContest.html
[2] H. Matsumoto, T. Ashida, Y. Ozasa, T. Maruyama, and R. Thawonmas, "ICE Pambush 3," *Controller description paper*, http://cswww.essex.ac.uk/staff/sml/pacman/cig2009/ICEPambush3/ICE%20Pambush%203.pdf
[3] Ms Pac-Man Competition IEEE CIG 2009 Results, http://cswww.essex.ac.uk/staff/sml/pacman/CIG2009Results.html
[4] M. Dianati, I. Song, and M. Treiber, "An Introduction to Genetic Algorithms and Evolutionary Strategies," *Technical Report*, Dept. of Electrical and Computer Engineering, University of Waterloo, 2002.
[5] S.M. Lucas, "Evolving a neural network location evaluator to play Ms. Pac-Man," in *IEEE Symposium on Computational Intelligence and Games*, 2005, pp. 203–210.
[6] P. Burrow, S.M. Lucas, "Evolution versus temporal difference learning for learning to play Ms. Pac-Man," in *Proc. of the 5th international symposium on Computational Intelligence and Games*, pp. 53–60, 2009.
[7] H. Handa, M. Isozaki, "Evolutionary Fuzzy Systems for Generating Better Ms. Pac Man Players," in *Proc. of the 2008 IEEE World Congress on Computational Intelligence*, pp. 2182–2185, 2008.
[8] L.L. DeLooze , W.R. Viner, "Fuzzy Q-learning in a nondeterministic environment: developing an intelligent Ms. Pac-Man agent," in *Proc. of the 5th international symposium on Computational Intelligence and Games*, pp. 162–169, 2009.
[9] H. Handa, "Constitution of Ms. Pac-Man Player with Critical-Situation Learning Mechanism," in *Proc. 4th International Workshop on Computational Intelligence and Applications*, pp. 48–53, 2008.
[10] N. Wirth and M. Gallagher, "An influence map model for playing Ms. Pac-Man," in *Proc. of the 4th international symposium on Computational Intelligence and Games*, pp. 228–233, 2008.
[11] D. Robles and S.M. Lucas, "Simple Tree Search Method for Playing Ms. Pac-Man," in *Proc. of the 5th international symposium on Computational Intelligence and Games*, pp. 249–255, 2009.
[12] I. Szita and A. Lorincz, "Learning to Play Using Low-Complexity Rule-Based Policies: Illustrations through Ms. Pac-Man," *Journal of Artificial Intelligence Research*, vol. 30, no. 1, pp. 659–684, 2007.
[13] E. Galvan-Lopez, J.M. Swafford, M. O'Neill and A. Brabazon, "Evolving a Ms. Pacman Controller using Grammatical Evolution," in *Applications of Evolutionary Computation, EvoApplications 2010: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC*, vol. 6024 of LNCS, Springer , Part I, pp. 161–170, Istanbul, Turkey 7-9 April, 2010.
[14] R. Thawonmas and H. Matsumoto, "Automatic Controller of Ms. Pac-Man and Its Performance: Winner of the IEEE CEC 2009 Software Agent Ms. Pac-Man Competition," in *CD-ROM Proc. Asia Simulation Conference 2009 (JSST 2009)*, 2009, Ritsumeikan University, Shiga, Japan (4 pages).
[15] Ms Pac-Man web version, http://www.webpacman.com/mspacman.htm