ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη ενός ευφυούς πράκτορα λογισμικού για το παιχνίδι Super Mario Bros με τη χρήση νευροεξελικτικών μεθόδων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΜΙΧΑΗΛ Σ. ΡΩΜΑΙΟΣ

Επιβλέποντες : Ανδρέας-Γεώργιος Σταφυλοπάτης | Καθηγητής Ε.Μ.Π.
Γεώργιος Σιόλας | Ε.ΔΙ.Π Ε.Μ.Π.

Αθήνα, Ιούνιος 2016

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη ενός ευφυούς πράκτορα λογισμικού για το παιχνίδι Super Mario Bros με τη χρήση νευροεξελικτικών μεθόδων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΜΙΧΑΗΛ Σ. ΡΩΜΑΙΟΣ

Επιβλέποντες : Ανδρέας-Γεώργιος Σταφυλοπάτης | Καθηγητής Ε.Μ.Π.
Γεώργιος Σιόλας | Ε.ΔΙ.Π Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή τον Ιούνιο του 2016.

.....	.....	.....
Ανδρέας-Γεώργιος Σταφυλοπάτης	Στέφανος Κόλλιας	Γιώργος Στάμου
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2016

.....

ΜΙΧΑΗΛ Σ. ΡΩΜΑΙΟΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

ΠΕΡΙΛΗΨΗ

Τα παιχνίδια παρέχουν ένα ιδεώδες πεδίο δοκιμών για την ανάπτυξη και την αξιολόγηση αλγορίθμων τεχνητής νοημοσύνης. Αυτή η διπλωματική πραγματεύεται την χρήση νευροεξέλιξης για την ανάπτυξη ενός ευφυούς πράκτορα για το παιχνίδι Super Mario Bros.

Για την ανάπτυξη του χειριστή χρησιμοποιήθηκε η πλατφόρμα Mario AI Benchmark και ο συνδυασμός Τεχνητών Νευρωνικών Δικτύων με Γενετικούς Αλγορίθμους. Ερευνήθηκαν διάφορες μορφές του χώρου εισόδων, η τοπολογία των νευρώνων εισόδου και εξόδου καθώς και διάφορα καθεστώτα αξιολόγησης της απόδοσης των πρακτόρων. Δείχθηκε ότι είναι σχετικά εύκολο να μάθει ένας πράκτορας τις βασικές στρατηγικές για να περάσει επίπεδα πολύ χαμηλής δυσκολίας, αλλά παρουσιάζει προβλήματα στην γενίκευση σε επίπεδα που δεν έχει εκπαιδευτεί , ειδικότερα σε επίπεδα μεγαλύτερης δυσκολίας.

Λέξεις κλειδιά : Νευροεξέλιξη, Εξελικτικοί Αλγόριθμοι, Νευρωνικά Δίκτυα, Γενετικοί Αλγόριθμοι, Τεχνητή Νοημοσύνη, Παιχνίδια

Η σελίδα αυτή είναι σκόπιμα λευκή.

ABSTRACT

Games provide an ideal testbed for developing and testing artificial intelligence algorithms. This thesis explores the use of neuroevolution for developing an intelligent agent for the game Super Mario Bros.

For the development of the agent the Mario AI Benchmark is used in conjunction with Artificial Neural Networks and Genetic Algorithms. Different input spaces and input/output neuron topologies where investigated, as well as different regimes for calculating the fitness of the agents. It was shown that it is relatively easy for agents to learn the basic strategies required to pass levels of low difficulty, though the agents have generalization problems in untrained levels, especially in levels of higher difficulty.

Keywords: Neuroevolution, Evolutionary Algorithms, Neural Networks, Genetic Algorithms, Artificial Intelligence, Games

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΠΕΡΙΕΧΟΜΕΝΑ

ΛΙΣΤΑ ΕΙΚΟΝΩΝ

ΛΙΣΤΑ ΠΙΝΑΚΩΝ

ΚΕΦΑΛΑΙΟ 1

1 Εισαγωγή

Τα παιχνίδια εμφανίζουν προβλήματα με κλιμακούμενες προκλήσεις στις οποίες εμπλέκονται πολλές πτυχές της ανθρώπινης γνωστικής ικανότητας.

Η σχέση των παιχνιδιών με την επιστήμη της τεχνητής νοημοσύνης έχει ρίζες ακόμα και πριν από την καθιέρωση της ως ξεχωριστού πεδίου. Από πολύ νωρίς ευεργέτες του κλάδου της επιστήμης των υπολογιστών έγραφαν προγράμματα που παίζουν παιχνίδια για να δοκιμάσουν αν οι υπολογιστές μπορούν να λύσουν προβλήματα που χρειάζονται "νοημοσύνη".

Το σκάκι για παράδειγμα είναι πιθανώς το παλιότερο πεδίο δοκιμών τεχνητής νοημοσύνης και έχει παίξει ένα σημαντικό ρόλο στην έρευνα τεχνικών και μεθόδων τεχνητής και υπολογιστικής νοημοσύνης. Ο Alan Turing είχε προτείνει ότι το παιχνίδι μπορεί να παιχτεί αυτόματα χρησιμοποιώντας τον αλγόριθμο MiniMax (Turing 1950).

Στον διάσημο αγώνα Kasparov εναντίων Deep Blue το 1997, για πρώτη φορά ένα λογισμικό πρόγραμμα νίκησε έναν παγκοσμίου κύρους παίχτη κάνοντας το, τον καλύτερο σκακιστή παγκοσμίως (Newborn 1997). Αποδείχτηκε έτσι ότι ένας πράκτορας τεχνητής νοημοσύνης μπορεί να διαπρέψει σε ένα συγκεκριμένο παιχνίδι, χωρίς να έχει ευρύτερο φάσμα συμπεριφοράς ή χωρίς να είναι ικανός να προσαρμόζεται σε διαφορετικές προκλήσεις του πραγματικού κόσμου.

Έκτοτε πολλά επιτραπέζια παιχνίδια έχουν παρουσιάσει δυσκολότερες προκλήσεις για τον κλάδο της υπολογιστικής και τεχνητής νοημοσύνης. Τα τελευταία χρόνια μεγάλο ερευνητικό ενδιαφέρον έχει επικεντρωθεί στο παιχνίδι Go, όπου ο μεγάλος παράγοντας διακλάδωσης καθιστά τις κλασσικές μεθόδους αναζήτησης δέντρων μη αποτελεσματικές. Για πρώτη φορά το 2016 το AlphaGo του DeepMind (Google) κατάφερε να νικήσει έναν επαγγελματία παίχτη στο παιχνίδι, τον πρωταθλητή Ευρώπης Fan Hui, χρησιμοποιώντας μάθηση σε βάθος με νευρωνικά δίκτυα (deep learning in neural networks) (Gibney 2016). Μερικούς μήνες αργότερα σφράγισε την νίκη του νικώντας 4-1 τον 19 φορές παγκόσμιο πρωταθλητή Lee Sedol.

Τα επιτραπέζια και τα παιχνίδια καρτών παρουσιάζουν πολλές δύσκολες και ενδιαφέρουσες προκλήσεις για τους ανθρώπους και απαιτούν ικανότητες όπως, η συλλογιστική (reasoning) και ο σχεδιασμός (planning). Υπάρχουν όμως κι άλλες προσκλήσεις που δεν εμφανίζονται σε αυτά. Τα ψηφιακά παιχνίδια (συγκεκριμένα τα βιντεοπαιχνίδια) ενώ χρειάζονται σχεδιασμό και

συλλογιστική για να παίξει κανείς επιτυχώς, παρουσιάζουν και άλλες απαιτήσεις όπως η οπτική αναγνώριση προτύπων, ο χωρικός προσανατολισμός και η συλλογιστική, γρήγορες αντιδράσεις, βραχυχρόνια μνήμη, ικανότητα πρόβλεψης της μεταβολής του περιβάλλοντος, ικανότητα δράσης σε περιβάλλοντα ελλιπών πληροφοριών καθώς και την δυνατότητα χειρισμού χώρων καταστάσεων και χώρων δράσεων πολλών διαστάσεων. (Karakovskiy και Togelius 2012).

Τα παιχνίδια πλατφόρμας και τα βιντεοπαιχνίδια γενικότερα, μπορούν να αποτελέσουν εξαιρετικά πεδία δοκιμών δεδομένου του ευρέος φάσματος προκλήσεων που παρουσιάζουν (παρόμοια με την ρομποτική) και του ευχάριστου χαρακτήρα της φύσεως τους. Ένα ακόμα πολύ θετικό στοιχείο χρήσης τους ως πεδίων δοκιμών είναι ότι επειδή τρέχουν σε ένα ελεγχόμενο υπολογιστικό περιβάλλον, τα περισσότερα μπορούν να επιταχυνθούν εκατοντάδες ή ακόμα και χιλιάδες φορές σε σχέση με τον πραγματικό χρόνο, επιτρέποντας έτσι την χρήση αλγορίθμων μάθησης και την παραμετροποίηση τους στον επιθυμητό βαθμό δυσκολίας.

Αυτά τα χαρακτηριστικά επιτρέπουν και προάγουν έρευνα που θα ήταν πρακτικά αδύνατη στον κλάδο της ρομποτικής, με πολύ μικρότερο κόστος, χρόνο και πολυπλοκότητα. Η ανάπτυξη ενός πράκτορα ή τεχνικών σε ένα τέτοιο περιβάλλον θα μπορούσε επιπλέον να συνδυαστεί με την ρομποτική παράγοντας ενδιαφέροντα αποτελέσματα.

Μια τεχνική που είναι εφαρμόσιμη σε ένα μεγάλος πλήθος προβλημάτων μέσα στο πεδίο της τεχνητής και υπολογιστικής νοημοσύνης είναι η νευροεξέλιξη (neuroevolution). Η νευροεξέλιξη πραγματεύεται την δημιουργία ή/και την εκπαίδευση νευρωνικών δικτύων με την χρήση εξελικτικών αλγορίθμων, που είναι μία κατηγορία στοχαστικών μεθόδων αναζήτησης, βασισμένων σε πληθυσμούς, εμπνευσμένων από την Δαρβινική εξέλιξη.

Στόχος της διπλωματικής είναι η δημιουργία και η μελέτη των δυνατοτήτων ενός ευφυούς πράκτορα λογισμικού για το MarioAI χρησιμοποιώντας τεχνικές νευροεξέλιξης για την εκπαίδευση του πράκτορα. Η πλατφόρμα MarioAI δημιουργήθηκε ως πεδίο δοκιμών και είναι βασισμένη στο Infinite Mario Bros του Markus Persson που είναι βεβαίως εμπνευσμένο από το Super Mario Bros της Nintendo. Ο κόσμος του Mario είναι ένα μερικώς παρατηρήσιμο, δυναμικό, επεισοδιακό πρόβλημα και κατά συνέπεια αποτελεί μία ενδιαφέρουσα και χρήσιμη πλατφόρμα για την εφαρμογή και αξιολόγηση διαφόρων τεχνικών μηχανικής μάθησης.

Η πλατφόρμα του Mario AI και κατ' επέκταση το Mario AI Competition είναι χαρακτηριστικό παράδειγμα της στροφής των ερευνητών τεχνητής νοημοσύνης στην χρήση βιντεοπαιχνιδιών ως πεδίων δοκιμών (testbeds) για αλγορίθμους και τεχνικές τα τελευταία χρόνια. Σε πολλές περιπτώσεις έχουν οργανωθεί διαγωνισμοί όπου ερευνητές μπορούν να υποβάλλουν πράκτορες δίνοντας έτσι την ευκαιρία σύγκρισης πολλών διαφόρων προσεγγίσεων. Ο επαναλαμβανόμενος χαρακτήρας των διαγωνισμών δίνει την δυνατότητα στους συμμετέχοντες να βελτιστοποιούν και να βελτιώνουν τις λύσεις τους.

Παιχνίδια για τα οποία έχουν οργανωθεί τέτοιοι διαγωνισμοί περιλαμβάνουν το Super Mario Bros (Togelius et. al. 2013), StarCraft (Ontañon et. al. 2013), TORCS (Loiacono et. al. 2010), Ms Pac-Man (Rolfshagen και Lucas 2011), ένα γενικό παιχνίδι μάχης τύπου Street-Fighter (Lu et. al. (2013), Angry Birds (Renz 2015), Unreal Tournament (Hingston 2010) κ.α.. Μια πρόσφατη

εξέλιξη στον χώρο αυτό αποτελεί ο διαγωνισμός General Video Game Playing ο οποίος περιγράφεται στο κεφάλαιο 7.

1.1 Δομή της διπλωματικής

Στο κεφάλαιο 1 αναφέρεται η δυνατότητα χρήσης των παιχνιδιών ως πεδίων δοκιμών για την έρευνα στον κλάδο της τεχνητής και υπολογιστικής νοημοσύνης.

Στο κεφάλαιο 2 παρουσιάζεται η έννοια του πράκτορα (agent), οι κατηγορίες αυτών καθώς και μια περιγραφή των συνηθέστερων περιβαλλόντων μέσα στα οποία καλούνται να λειτουργήσουν.

Στο κεφάλαιο 3 περιγράφονται οι ιδέες και το θεωρητικό υπόβαθρο των τεχνικών και των πεδίων που εμπλέκονται στην διπλωματική, όπως τα τεχνητά νευρωνικά δίκτυα, το πεδίο της εξελικτικής υπολογιστικής, η νευροεξέλιξη και τα πλεονεκτήματα εφαρμογής της σε παιχνίδια.

Στο κεφάλαιο 4 περιγράφεται η πλατφόρμα MarioAI και τα χαρακτηριστικά της καθώς και διάφορες από τις προκλήσεις που παρουσιάζονται στο συγκεκριμένο πρόβλημα.

Στο κεφάλαιο 5 γίνεται αναλυτική παρουσίαση της δομής των διαφόρων πρακτόρων που εξελίχτηκαν. Αναλύονται οι διάφορες αρχιτεκτονικές και οι τρόποι αλληλεπίδρασης με το περιβάλλον, καθώς και οι παράμετροι του γενετικού αλγορίθμου και το πώς χρησιμοποιήθηκε για να εκπαιδευτούν οι πράκτορες.

Στο κεφάλαιο 6 παρουσιάζονται τα πειράματα που εκτελέστηκαν και τα αποτελέσματά τους. Μέσα από μια σειρά πειραμάτων αποφασίστηκε η τελική μορφή της αρχιτεκτονικής του δικτύου. Στην συνέχεια ερευνήθηκαν οι δυνατότητες εξελιγμένων πρακτόρων, με αυτήν την αρχιτεκτονική, σε πιο δύσκολα επίπεδα.

Αρχικά επιλέχθηκε η στρατηγική εκπαίδευσης (πόσες πίστες θα παίξει κάθε άτομο του πληθυσμού για να αποτιμηθεί η απόδοση του), χρησιμοποιώντας μία από της πιθανές αρχιτεκτονικές του δικτύου. Στην συνέχεια μέσα από μία σειρά εξελίξεων επιλέχθηκε ποια θα είναι η μορφή των εισόδων και αντίστοιχα εξόδων του τελικού δικτύου, σύμφωνα με την απόδοση του καλύτερου πράκτορα κάθε αρχιτεκτονικής. Αφού αποφασίστηκε η τελική μορφή, εξελίχθηκαν πράκτορες για επίπεδα διαφορετικής δυσκολίας και συγκρίθηκε η απόδοση τους με πράκτορες που εμπεριέχονταν στην πλατφόρμα. Τέλος ερευνήθηκαν οι δυνατότητες του πράκτορα σε σύνθετες μορφολογικά πίστες, καθώς και σε δυσκολότερα επίπεδα.

Στο κεφάλαιο 7 παρουσιάζονται τα συμπεράσματα της έρευνας και γίνεται ο απολογισμός του έργου. Επίσης γίνεται μια πρόταση για ερευνητές που τους κίνησε το ενδιαφέρον η χρήση

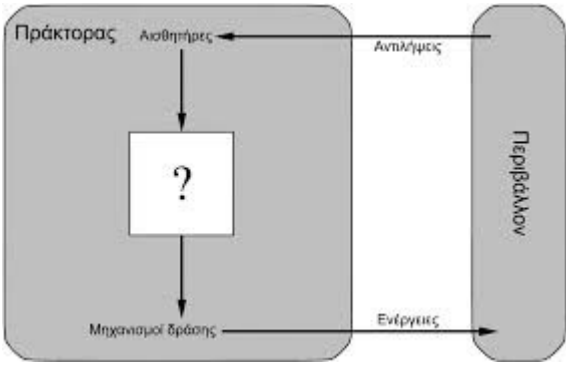
των παιχνιδιών ως πεδίων δοκιμών για αλγορίθμους μάθησης και ενδιαφέρονται να συνεχίσουν προς αυτή την κατεύθυνση.

ΚΕΦΑΛΑΙΟ 2

2 Ευφυείς πράκτορες

Ο σημαντικότερος στόχος της τεχνητής νοημοσύνης είναι η δημιουργία πλήρως αυτόνομων και λειτουργικών οντοτήτων, είτε σε φυσική μορφή (π.χ. ρομπότ, αυτοκινούμενα οχήματα κτλ.), είτε σε άυλη μορφή (π.χ. λογισμικό), που θα συνυπάρχουν στον πραγματικό κόσμο με τους ανθρώπους και με ομολόγους τους (Stone 2007, αναφορά απο τον Χατζηδημητρίου 2012). Μία από τις βασικότερες μεταφορές για την αναπαράσταση τέτοιου είδους συστημάτων είναι η έννοια του πράκτορα.

Πράκτορας είναι οτιδήποτε μπορεί να θεωρηθεί ότι αντιλαμβάνεται το **περιβάλλον** του (environment) μέσω **αισθητήρων** (sensors), και επενεργεί σε αυτό το περιβάλλον μέσω **μηχανισμών δράσης** (actuators). (Russel and Norvig 2003). Η ιδέα αυτή απεικονίζεται παρακάτω.



Εικόνα 2.1 : Γενικό μοντέλο ενός πράκτορα

Ένας ζωντανός πράκτορας μπορεί να είναι ένας σκύλος ο οποίος έχει την μύτη του, τα αφτιά του, τα μάτια του και άλλα ως αισθητήρες, και πόδια, στόμα (φωνητικό σύστημα) και άλλα ως μηχανισμούς δράσης. Ένας ρομποτικός πράκτορας μπορεί να έχει κάμερες και μαγνητόμετρο ως αισθητήρες, και διαφόρους κινητήρες ως μηχανισμούς δράσης. Ένας λογισμικός πράκτορας μπορεί να δέχεται πατήματα πλήκτρων, περιεχόμενα αρχείων και πακέτα δικτύου ως αισθητήριες εισόδους, και να ενεργεί στο περιβάλλον εμφανίζοντας στοιχεία στην οθόνη, γράφοντας σε αρχεία, εκτελώντας λογισμικές λειτουργίες, στέλνοντας πακέτα και άλλα.

Η έρευνα πάνω στην περιοχή των πρακτόρων έχει ένα πολύ ευρύ φάσμα. Μερικές είναι :

- **Κατανεμημένη επίλυση προβλημάτων (Distributed Problem Solving)**
  Η ιδέα των πρακτόρων μπορεί να χρησιμοποιηθεί για απλοποιήσει την λύση μεγάλων προβλημάτων με την διανομή αυτών σε έναν αριθμό συνεργαζόμενων μονάδων επίλυσης .
- **Πολυπρακτορικά συστήματα (Multi-Agent Systems)**
  Επιλύουν με κατανεμημένο τρόπο σύνθετα προβλήματα, χωρίς ο κάθε πράκτορας να έχει γνώση του συνολικού προβλήματος. Η έρευνα σε αυτόν τον κλάδο πραγματεύεται τους κατάλληλους τρόπους οργάνωσης των πρακτόρων. Αυτοί περιλαμβάνουν γενικές οργανωτικές έννοιες, την διανομή των καθηκόντων διαχείρισης, δυναμικές οργανωτικές αλλαγές, όπως το σχηματισμό ομάδων και των υποκείμενων μηχανισμών επικοινωνίας κ.α.
- **Αυτόνομοι πράκτορες**
  Η έρευνα στους αυτόνομους πράκτορες ασχολείται κυρίως με την υλοποίηση ενός και μόνο πράκτορα. Αυτό περιλαμβάνει θέματα όμως η ανίχνευση (sensing), τα μοντέλα του συναισθήματος (models of emotion), τα κίνητρα (motivation), την προσωπικότητα (personality), και την επιλογή δράσεων και σχεδιασμού (action selection and planning). (Nareyek 2001)

Τα είδη των πρακτόρων σύμφωνα με την αρχιτεκτονική τους μπορούν να διακριθούν σε :

- Αντανακλαστικούς πράκτορες (reactive agents)
- Πράκτορες βασισμένους σε στόχους (goal based agents)
- Πράκτορες βασισμένους στην χρησιμότητα (utility based agents)
- Πράκτορες με πεποιθήσεις, επιθυμίες, προθέσεις (BDI-Belief Desire Intention)
- Υβριδικές αρχιτεκτονικές
  (Løland 2008)

2.1 Κατηγορίες ευφυών πρακτόρων

Παρακάτω παρουσιάζονται μερικές από τις κύριες κατηγορίες ευφυών πρακτόρων.

2.1.1 Αντανακλαστικοί πράκτορες

Αποκλειστικά αντανακλαστικοί πράκτορες

Αυτό είναι το απλούστερο είδος πράκτορα. Αυτοί οι πράκτορες επιλέγουν ενέργειες με βάση την τρέχουσα αντίληψη, αγνοώντας το υπόλοιπο ιστορικό αντιλήψεων (το παρελθόν). Το πρόγραμμα του πράκτορα βασίζεται σε κανόνες *συνθήκης-ενέργειας.*

Αυτοί οι πράκτορες δουλεύουν μόνο όταν το περιβάλλον είναι πλήρως παρατηρήσιμο. Μερικοί αντανακλαστικοί πράκτορες μπορεί να έχουν και πληροφορίες για την τρέχουσα κατάσταση τους που τους δίνει την δυνατότητα να αγνοούν συνθήκες των οποίων οι μηχανισμοί δράσης έχουν ήδη ενεργοποιηθεί.

Οι ατέρμονες βρόχοι είναι συχνά αναπόφευκτοι για τους απλούς ανακλαστικούς πράκτορες που λειτουργούν σε μερικώς παρατηρήσιμα περιβάλλοντα. Αν ο πράκτορας μπορεί να

τυχαιοποιεί (randomize) τις ενέργειες του, μπορεί να είναι δυνατό να διαφύγει από ένα τούς ατέρμονα βρόχο. (Suganya 2014)

Αντανακλαστικοί πράκτορες βασισμένοι σε μοντέλο

Οι πράκτορες βασισμένοι σε μοντέλο μπορούν να διαχειριστούν ένα μερικώς παρατηρήσιμο περιβάλλον. Η τρέχουσα κατάστασή του αποθηκεύεται από τον πράκτορα τηρώντας κάποιο είδος δομής που περιγράφει τον τμήμα του κόσμου που δεν μπορεί να δει. Αυτή η γνώση για το "πώς δουλεύει ο κόσμος" καλείται μοντέλο του κόσμου.

Ο πράκτορας θα πρέπει να τηρεί κάποιο είδος εσωτερικής κατάστασης, η οποία εξαρτάται από το ιστορικό των αντιλήψεων και μέσω αυτών αντανακλά τουλάχιστον μερικές από τις μη παρατηρήσιμες απόψεις για την τρέχουσα κατάσταση. (Russel and Norvig 2003)

2.1.1 Πράκτορες βασισμένοι σε στόχους

Οι πράκτορες βασισμένοι σε στόχους επεκτείνουν της δυνατότητες των αντανακλαστικών πρακτόρων βασισμένων σε μοντέλο, χρησιμοποιώντας πληροφορίες για κάποιο στόχο. Αυτές οι πληροφορίες περιγράφουν καταστάσεις που είναι επιθυμητές. Το πρόγραμμα του πράκτορα συνδυάζει αυτές της πληροφορίες με εκείνες που αφορούν τα αποτελέσματα των δυνατών ενεργειών (τις ίδιες πληροφορίες που χρησιμοποιήθηκαν για να ενημερωθεί η εσωτερική κατάσταση στον αντανακλαστικό πράκτορα) προκειμένου να επιλέγει ενέργειες που επιτυγχάνουν τους στόχους. (Russel and Norvig 2003). Η επιλογή των ενεργειών είναι συχνά περίπλοκη και χρησιμοποιείται η αναζήτηση (searching) και ο σχεδιασμός (planning).

2.1.1 Πράκτορες βασισμένοι στην χρησιμότητα

Οι πράκτορες βασισμένοι σε στόχους μπορούν να διακρίνουν μεταξύ καταστάσεων στόχων και μη-στόχων. Είναι δυνατό να οριστεί ένα μέτρο για το πόσο είναι επιθυμητή μια συγκεκριμένη κατάσταση. Το μέτρο αυτό μπορεί να αποκτηθεί μέσω της χρήσης μίας συνάρτησης χρησιμότητας που αντιστοιχίζει μια κατάσταση σε ένα πραγματικό αριθμό που αντικατοπτρίζει ένα βαθμό ικανοποίησης που προκύπτει από αυτήν την κατάσταση. Ο πράκτορας διαλέγει ενέργειες με την μεγαλύτερη χρησιμότητα, προσπαθώντας έτσι να μεγιστοποιήσει την απόδοση του.

2.1.1 Ορθολογικότητα των πρακτόρων

Η ορθολογικότητα είναι η δυνατότητα για την επιλογή αποδοτικών ενεργειών βάση των διαθέσιμων πληροφοριών για το περιβάλλον και την γνώση για περιβάλλον στο οποίο βρίσκεται ο πράκτορας. Γνώση τόσο για το παρελθόν (προηγούμενη γνώση/εμπειρία), όσο και για το πώς το περιβάλλον συμπεριφέρεται και εξελίσσεται βάση τις ακολουθία των αντιλήψεων μέχρι στιγμής.

2.1.1 Πράκτορες με πεποιθήσεις, επιθυμίες, προθέσεις

Ο τρόπος λειτουργίας ενός πράκτορα BDI (Belief-Desire-Intention) στηρίζεται σε θεωρητικά μοντέλα ανθρώπινης συλλογιστικής.

- Οι **πεποιθήσεις** αντιπροσωπεύουν την αντίληψη του πράκτορα για τον κόσμο (συμπεριλαμβανομένου και του εαυτού του). Καινούργιες πεποιθήσεις δημιουργούνται από τις αντιλήψεις και τις παλιές πεποιθήσεις. Μία πεποίθηση μπορεί να είναι λάθος, καθώς και να αλλάξει στο μέλλον.
- Οι **επιθυμίες** αναπαριστούν τα κίνητρα που έχει ο πράκτορας. Είναι οι στόχοι ή οι καταστάσεις που ο πράκτορας θα ήθελε να επιτύχει ή να επιφέρει. Για παράδειγμα επιθυμία μπορεί να είναι : εύρεση της οικονομικότερη πτήσης.
- Οι **προθέσεις** αντιπροσωπεύουν το τι ο πράκτορας έχει επιλέξει και δεσμευτεί να κάνει. Οι προθέσεις επιλέγονται απο τον πράκτορα με βάση τις επιθυμίες και τις πεποιθήσεις.
    - Τα **πλάνα** (plans) είναι ακολουθίες ενεργειών που μπορεί ο πράκτορας να εκτελέσει για την επίτευξη μιας ή περισσοτέρων από τις πεποιθήσεις του. Τα πλάνα μπορεί να περιλαμβάνουν άλλα πλάνα: σε ένα παιχνίδι, το πλάνο να επισκεφτεί ο ήρωας έναν άλλο πλανήτη περιέχει το πλάνο να σχεδιάσει τη διαδρομή έως εκεί. Αυτό σημαίνει ότι αρχικά τα πλάνα έχουν ένα γενικό σχεδιασμό και, στη συνέχεια, συμπληρώνονται με στοιχεία ανάλογα με το πώς εξελίσσονται.

Η λειτουργία ενός πράκτορα BDI βασίζεται στην ύπαρξη συμβάντων που προκαλούν την αντίδρασή του:

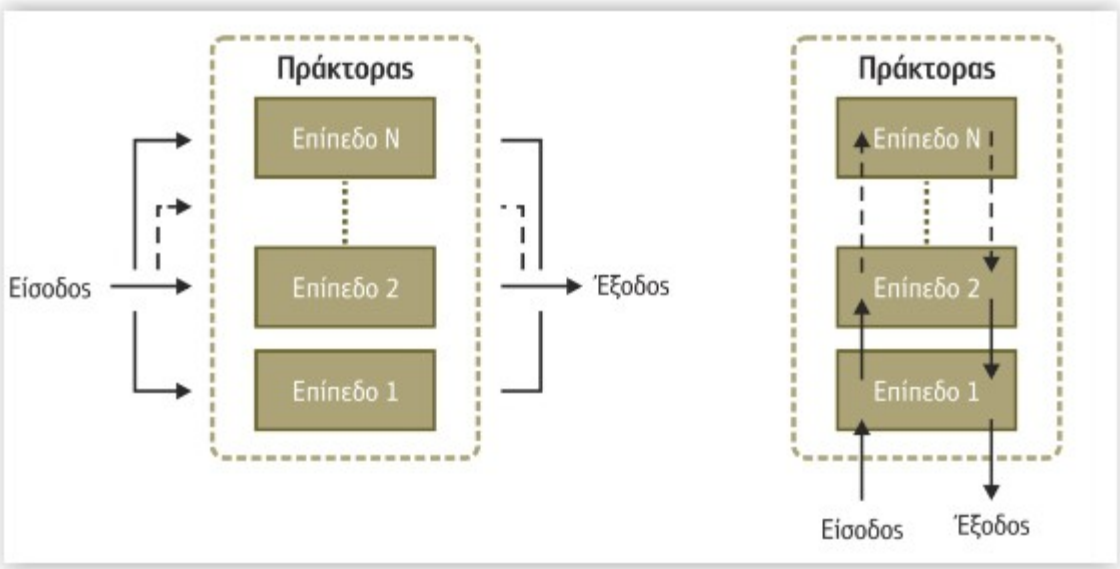
- Τα **συμβάντα** είναι ωθήσεις για αντιδραστική δραστηριότητα από τον πράκτορα. Ένα συμβάν μπορεί να ενημερώσει τις πεποιθήσεις, να προκαλέσει σχέδια ή να τροποποιήσει τους στόχους. Τα συμβάντα μπορούν να δημιουργηθούν είτε εσωτερικά από τον ίδιο τον πράκτορα, για να προκαλέσουν αποσυνδεδεμένες ενημερώσεις ή σχέδια δράσης, ή απο το ίδιο το σύστημα είτε εξωτερικά και να γίνουν αντιληπτά απο τους αισθητήρες. (Γεωργούλη 2016)

2.1.1 Υβριδικές αρχιτεκτονικές

Για να μπορέσει ένας πράκτορας να έχει αντιδραστική καθώς και προδραστική συμπεριφορά εισάγεται μια ιεραρχία αλληλεπιδρόντων υποσυστημάτων σε επίπεδα. Χρησιμοποιούνται κατ' ελάχιστο 2 επίπεδα. Ένα υπεύθυνο για την αντιδραστική συμπεριφορά και ένα για την συμπεριφορά με εσωτερική κατάσταση.
Διακρίνονται σε 2 κατηγορίες:

- **Οριζόντιες**
  Σε αυτό το μοντέλο, όλα τα επίπεδα είναι συνδεδεμένα σε παραλληλία στους αισθητήρες εισόδου και στους μηχανισμούς δράσης
- **Κάθετες**

Σε αυτό το μοντέλο, τα επίπεδα είναι συνδεδεμένα σε σειρά, όπου μόνο ένα επίπεδο είναι συνδεδεμένο στους αισθητήρες εισόδου και μόνο ένα στους μηχανισμούς δράσης. Αυτό το μοντέλο συχνά χρειάζεται μία συνάρτηση ελέγχου που αποφασίζει ποιο επίπεδο έχει τον έλεγχο του πράκτορα ανά πάσα στιγμή. Αυτή η συνάρτηση μπορεί να θεωρηθεί και το "σημείο συμφόρησης" στην διαδικασία λήψης αποφάσεων του πράκτορα.



Εικόνα 2.2 : Οριζόντια και κάθετη αρχιτεκτονική.

Οι κάθετες αρχιτεκτονικές μπορούν να χωριστούν επιπλέον σε 2 κατηγορίες:

- **Ενός περάσματος**
  Στην αρχιτεκτονική "ενός περάσματος" η ροή ελέγχου και τον πληροφοριών διατρέχει ακολουθιακά κάθε επίπεδο, εως ότου το τελευταίο επίπεδο παράγει μια δράση εξόδου.
- **Δύο περασμάτων**
  Σε αυτή την αρχιτεκτονική η ροη των πληροφοριών διατρέχει "προς τα πάνω" τα επίπεδα, και η ροή ελέγχου προς τα κάτω. Στην εικόνα 2.2. στα δεξιά απεικονίζεται ένα γενικό μοντέλο αρχιτεκτονικής "δύο περασμάτων".

2.2 Περιβάλλοντα πρακτόρων

Το περιβάλλον είναι ο κόσμος μέσα στον οποίο λειτουργεί ένας πράκτορας. Το περιβάλλον έχει πολλές ιδιότητες που μπορούν να κατηγοριοποιηθούν. Οι Russel και Norvig (2003) κατηγοριοποιούν τα περιβάλλοντα ως εξής:

- **Πλήρως προσβάσιμο ή μερικώς προσβάσιμο** (Fully observable / partially observable) όταν οι αισθητήρες ενός πράκτορα του παρέχουν πρόσβαση στην πλήρη κατάσταση του περιβάλλοντος σε κάθε χρονική στιγμή, τότε το περιβάλλον εργασιών καλείται πλήρως παρατηρήσιμο. Σε ένα πλήρως προσβάσιμο περιβάλλον, ένας πράκτορας δεν απαιτείται

να έχει εσωτερική κατάσταση για την αναπαράσταση του περιβάλλοντος. Ενα περιβάλλον μπορεί να είναι μερικώς παρατηρήσιμο λόγω του θορύβου και της ανακρίβειας των αισθητήρων ή επειδή κάποια μέρη της κατάστασης λειτουργίας λείπουν από τα δεδομένα των αισθητήρων.

- **Αιτιοκρατικό ή στοχαστικό** (Deterministic / stochastic)
  Αν η επόμενη κατάσταση του περιβάλλοντος προσδιορίζεται πλήρως από την τρέχουσα κατάσταση και από την ενέργεια που εκτελείται από τον πράκτορα, τότε το περιβάλλον χαρακτηρίζεται ως αιτιοκρατικό, αλλιώς, το περιβάλλον είναι στοχαστικό. Επιπλέον αν το περιβάλλον είναι αιτιοκρατικό με εξαίρεση τις ενέργειες άλλων πρακτόρων, τότε χαρακτηρίζεται ως **στρατηγικό** (strategic)

- **Επεισοδιακό ή ακολουθιακό** (Episodic / sequential)
  είναι το περιβάλλον στο οποίο η εμπειρία του πράκτορα αποτελείται από επεισόδια. Σε κάθε επεισόδιο, ο πράκτορας αντιλαμβάνεται και έπειτα πραγματοποιεί μια μεμονωμένη ενέργεια. Το επόμενο επεισόδιο δεν εξαρτάται καθοριστικά από τις ενέργειες που έγιναν στα προηγούμενα επεισόδια. Αντίθετα, σε στα ακολουθιακά περιβάλλοντα η τρέχουσα κατάσταση επηρεάζει όλες τις μελλοντικές αποφάσεις

- **Στατικό ή δυναμικό** (Static / dynamic)
  Αν το περιβάλλον μπορεί να αλλάζει ενώ ένας πράκτορας σκέφτεται , τότε χαρακτηρίζεται ως δυναμικό, αλλιώς χαρακτηρίζεται ως στατικό. Αν το περιβάλλον δεν αλλάζει με το πέρασμα του χρόνου, όμως αλλάζει η βαθμολογία της απόδοσης του πράκτορα, τότε το περιβάλλον χαρακτηρίζεται ως **ημιδυναμικό** (semidynamic)

- **Διακριτό ή συνεχές** (discrete / continuous)
  Η διάκριση μεταξύ συνεχούς και διακριτού μπορεί να εφαρμοστεί στην κατάσταση του περιβάλλοντος, στον τρόπο που αντιμετωπίζεται ο χρόνος, και στις αντιλήψεις και ενέργειες του πράκτορα. Συνηθέστερα λαμβάνεται υπόψιν η αντίληψη του χρόνου από πλευράς του πράκτορα μέσω των αισθητήρων του. Ένα παιχνίδι ντάμας για παράδειγμα, όπου οι κινήσεις εκφράζονται σε γύρους, αποτελεί ένα στατικό περιβάλλον. Ένα παιχνίδι στρατηγικής πραγματικού χρόνου απο την άλλη είναι συνεχές.

- **Μονοπρακτορικό ή πολυπρακτορικό** (single-agent / multiagent)
  Αν ένα πράκτορας δρα μόνος του σε ένα περιβάλλον, τότε αυτό χαρακτηρίζεται ως μονοπρακτορικό. Σε αντίθετη περίπτωση χαρακτηρίζεται πολυπρακτορικό.
  Τα πολυπρακτορικά περιβάλλοντα μπορεί να είναι ανταγωνιστικά (competitive) ή συνεργατικά (cooperative).

Όσο πιο περίπλοκο είναι το περιβάλλον, τόσο πιο δύσκολη είναι η ανάπτυξη ενός λειτουργικού πράκτορα που δρα μέσα σε αυτό. Η πολυπλοκότητα δεν είναι μια προκαθορισμένη

ιδιότητα του περιβάλλοντος, άλλα εξαρτάται σε μεγάλο βαθμό από τους αισθητήρες του πράκτορα που αλληλεπιδρά με αυτό. Δύο πράκτορες ίδια λειτουργίας αλλά διαφορετικών αισθητήρων οι οποίοι δρουν στο ίδιο περιβάλλον το αντιλαμβάνονται διαφορετικά. (Γεωργούλη 2016)

ΚΕΦΑΛΑΙΟ 3

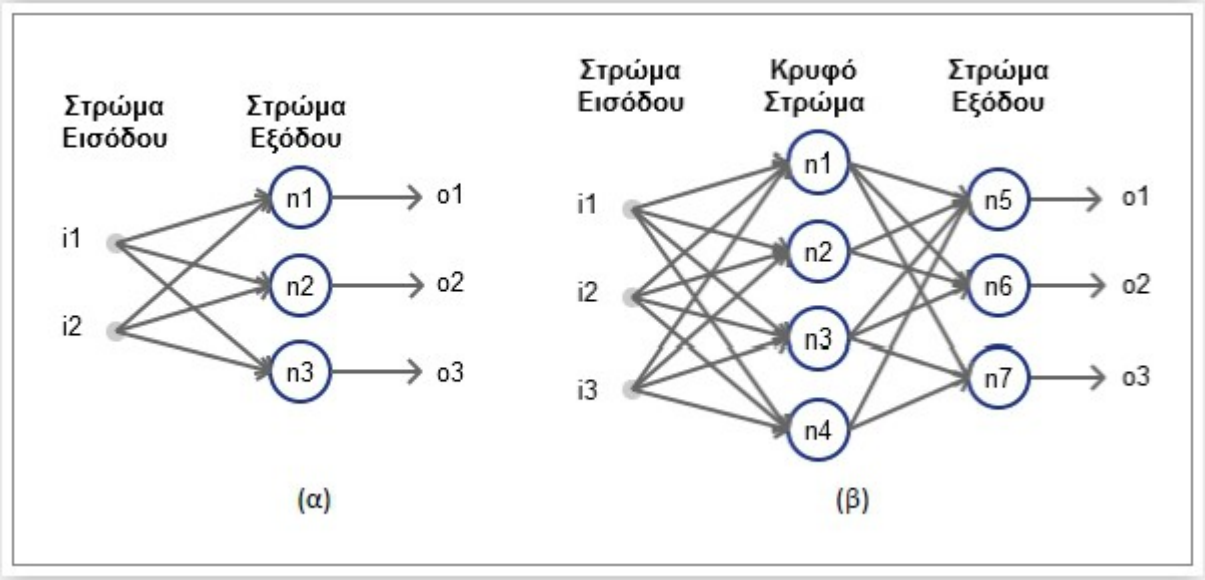
3 Θεωρητικό υπόβαθρο και συναφείς μελέτες

3.1 Τεχνητά νευρωνικά δίκτυα

Τα τεχνητά νευρωνικά δίκτυα είναι υπολογιστικά μοντέλα που προσπαθούν να προσομοιάσουν την συμπεριφορά και τα προσαρμοστικά χαρακτηριστικά των βιολογικών νευρωνικών δικτύων. Αυτά τα μοντέλα υλοποιούνται είτε με την χρήση λογισμικού, είτε χρησιμοποιώντας ηλεκτρονικά στοιχεία (hardware). Τα τεχνητά νευρωνικά δίκτυα χρησιμοποιούνται συχνά για την επίλυση προβλημάτων όπου είναι δύσκολο ή αδύνατο να δοθεί μια αναλυτική λύση.

Ένα τεχνητό νευρωνικό δίκτυο αποτελείται από επιμέρους μονάδες που καλούνται νευρώνες, οι οποίοι είναι διασυνδεδεμένοι μεταξύ τους με σταθμισμένες συνδέσεις. Οι νευρώνες είναι συνήθως χωρισμένοι σε επίπεδα και ο αριθμός των επιπέδων εξαρτάται από την αρχιτεκτονική του δικτύου.

Τα επίπεδα μπορούν να χωριστούν σε επίπεδο εισόδου, κρυφά επίπεδα και επίπεδο εξόδου. Οι νευρώνες στο επίπεδο εισόδου δέχονται πληροφορίες από το περιβάλλον, ενώ οι νευρώνες στο επίπεδο εξόδου παράγουν ένα σήμα εξόδου. Οι ενδιάμεσοι νευρώνες δεν αποτελούν μέρος ούτε της εισόδου, ούτε της εξόδου, για αυτό χαρακτηρίζονται ως κρυφοί. Τα περισσότερα τεχνητά νευρωνικά δίκτυα έχουν ένα ή περισσότερα κρυφά επίπεδα, ενώ είναι δυνατή και η ύπαρξη δικτύου χωρίς κανένα κρυφό επίπεδο. Παρακάτω παρουσιάζονται δύο γενικευμένες μορφές δικτύων πρόσθιας τροφοδότησης. Ένα χωρίς κρυφό επίπεδο, και ένα με ένα κρυφό επίπεδο.



Εικόνα 3.1 : Γενική μορφή τεχνητών νευρωνικών δικτύων.

Το απλό μοντέλο των νευρώνων έχει τα εξής χαρακτηριστικά:

- ένα σύνολο συνδέσεων (**συνάψεις**), κάθε μία από τις οποίες χαρακτηρίζεται απο ένα βάρος. Οι συνάψεις αποτελούν την είσοδο των διαφόρων σημάτων στον νευρώνα και μπορεί να προέρχονται από άλλους νευρώνες ή το περιβάλλον του δικτύου. Τα βάρη των συνάψεων λειτουργούν πολλαπλασιαστικά στα σήματα και μπορούν να έχουν οποιαδήποτε πραγματική τιμή.
- έναν **αθροιστή**, για την άθροιση των σημάτων εισόδου, πολλαπλασιασμένων με τα βάρη των συνάψεων.
- μία **συνάρτηση μεταφοράς**. Αυτή είναι μία μαθηματική συνάρτηση (γραμμική, κατωφλίου, σιγμοειδής κ.α.) που δέχεται ως είσοδο την έξοδο του αθροιστή και το αποτέλεσμα αποτελεί την έξοδο του νευρώνα στο υπόλοιπο σύστημα ή στο εξωτερικό περιβάλλον.

Το σήμα εξόδου του νευρώνα μπορεί συνεχές ή διακριτό.

Η αρχιτεκτονική των τεχνητών νευρωνικών δικτύων επηρεάζει την λειτουργικότητά τους. Οι πιο σύνηθες αρχιτεκτονικές είναι τα **δίκτυα πρόσθιας τροφοδότησης**, και τα **δίκτυα με ανάδραση**.

Τα δίκτυα πρόσθιας τροφοδότησης είναι χωρισμένα σε επίπεδα όπου η ροή της πληροφορίας ρέει μόνο προς την μία κατεύθυνση (βλ. Εικόνα 4.1). Ο κάθε νευρώνας τροφοδοτεί μόνο νευρώνες που ανήκουν στο επόμενο επίπεδο. Αυτή είναι η απλούστερη μορφή ΤΝΝ.

Τα δίκτυα με ανάδραση μπορεί να έχουν συνδέσεις μεταξύ των νευρώνων του ιδίου επιπέδου, καθώς και προηγούμενων. Η παρουσία αναδράσεων έχει μεγάλη επίπτωση στις δυνατότητες μάθησης του δικτύου καθώς και στην επίδοση του.

Η έξοδος του νευρωνικού δικτύου όταν δέχεται πληροφορίες από το περιβάλλον εξαρτάται από την συνάρτηση μεταφοράς των νευρώνων, τις συνδέσεις μεταξύ αυτών και τα βάρη των συνδέσεων.

Μία από τις βασικότερες δυνατότητες ενός ΤΝΝ είναι ότι μπορεί να μαθαίνει από το περιβάλλον και μέσα από την διαδικασία αυτή να βελτιώνει την απόδοσή του. Χρησιμοποιώντας μεθόδους εκμάθησης μπορούν να προσαρμοστούν οι συνδέσεις μεταξύ των νευρώνων έτσι ώστε να βελτιωθεί η απόδοση του και έτσι να "μάθει". Οι περισσότερες μέθοδοι εκπαίδευσης ΤΝΝ χρησιμοποιούν παραδείγματα και μπορούν να χωριστούν σε **εκπαίδευση με επίβλεψη** (supervised learning) και **εκπαίδευση χωρίς επίβλεψη** (unsupervised learning).

Οι μέθοδοι εκπαίδευσης χωρίς επίβλεψη, όπως για παράδειγμα ο αλγόριθμος back-propagation (Rumelhart et al. 1986), βασίζονται στην σύγκριση της εξόδου του ΤΝΝ με μία επιθυμητή έξοδο, για έναν δοσμένο σετ από εισόδους. Στην συνέχεια η διαφορά των δύο εξόδων (επιθυμητής και πραγματικής) θεωρείται η τιμή σφάλματος, και αυτό το σφάλμα διαδίδεται προς τα πίσω μέσα από το δίκτυο προσαρμόζοντας στην πορεία τα βάρη των συνάψεων βάση του κανόνα εκμάθησης που χρησιμοποιεί. Υπάρχει μία ειδική κατηγορία της επιβλεπόμενης μάθησης που καλείται **ενισχυτική μάθηση**. Σε αυτήν δεν παρουσιάζονται ζεύγη σωστών/λάθος αποτελεσμάτων, αλλά δίνεται μία αμοιβή σχετικά με το πόσο επιθυμητή είναι η κατάσταση στην οποία θα βρεθεί το σύστημα μετά την έξοδο. Έτσι η διαδικασία μάθησης εκτελείται μέσα από συνεχή αλληλεπίδραση με το περιβάλλον.

Η εκπαίδευση χωρίς επίβλεψη βασίζεται σε συσχετισμούς μεταξύ των τιμών εισόδου, χωρίς γνώση για τις σωστές τιμές εξόδου.

3.2 Εξελικτική υπολογιστική

Η εξελικτική υπολογιστική (Eiben και Smith 2003) είναι μία περιοχή που περιλαμβάνει υπολογιστικές διαδικασίες στοχαστικής εξεύρεσης λύσεων ακολουθώντας το παράδειγμα τις εξέλιξης των ειδών στο φυσικό περιβάλλον. Κύρια συστατικά της είναι εξέλιξη ενός πληθυσμού λύσεων, που ονομάζονται χρωμοσώματα (chromosomes), με τεχνικές βασισμένες στην φύση, όπως για παράδειγμα φυσική επιλογή με βάση την ικανότητα ενός οργανισμού, διασταύρωση χρωμοσωμάτων και μετάλλαξη γονιδίων κ.α.. Οι κυριότερες μεθοδολογίες σε αυτήν την περιοχή είναι οι γενετικοί αλγόριθμοι, ο γενετικός προγραμματισμός και οι εξελικτικές στρατηγικές. Η εφαρμογή αυτών των ιδεών επεκτείνεται και στην ρομποτική στην περιοχή της εξελικτικής ρομποτικής (evolutionary robotics) (Doncieux et al. 2015). Οι μεθοδολογίες της εξελικτικής υπολογιστικής μπορούν να διαχειρίζονται προβλήματα βελτιστοποίησης μεγάλου εύρους, πολύπλοκα, μη διαφορίσιμα, μη συνεχή, πολυτροπικά (multinodal) και θορυβώδη (Yao 1999, Eiben και Smith 2003, Igel και Sendhoff 2008, αναφορά από τον Χατζηδημητρίου 2012).

Παρακάτω παρουσιάζονται τα βασικότερα στοιχεία ενός αλγορίθμου εξελικτικής υπολογιστικής.

- **Πληθυσμός** (Population): Οι αλγόριθμοι εξελικτικής υπολογιστικής χρησιμοποιούν έναν πληθυσμό στον οποίο κάθε στοιχείο του είναι και μία πιθανή λύση στο πρόβλημα που εξετάζεται. Τα άτομα του πληθυσμού συχνά καλούνται χρωμοσώματα, και μπορεί να αναφέρονται και ως οργανισμοί (organisms) ή γονιδιώματα (genomes).

- **Γενιές** (Generations): Ένας εξελικτικός αλγόριθμος εξελίσσει τον πληθυσμό για έναν αριθμό γενεών.

- **Επιλογή** (Selection): Κατά την επιλογή προκρίνονται οι ικανότεροι οργανισμοί προκειμένου να διασταυρωθούν και να προκύψουν απόγονοι που θα συνδυάζουν τα κατάλληλα στοιχεία και από του δύο γονείς και κατ' επέκταση να αποτελέσουν μια βελτιωμένη λύση για το πρόβλημα. Υπάρχουν διάφορες στρατηγικές επιλογής. Μερικές είναι:

  - **Επιλογή Ρουλέτας** (Roulette wheel selection)
    Κάθε χρωμόσωμα επιλέγεται ως γονέας με πιθανότητα ανάλογη της ικανότητας του συγκρινόμενη με το άθροισμα των ικανοτήτων όλων των άλλων ατόμων του πληθυσμού. $P(k)=\frac{fitness(k)}{\sum_{i=1}^{n} fitness(i)}$.

  - **Επιλογή Κατάταξης** (Rank selection)
    Στην επιλογή κατάταξης πρώτα ταξινομείται ο πληθυσμός με βάση την απόδοση του, και έπειτα το κάθε χρωμόσωμα λαμβάνει μία τιμή καταλληλότητας σύμφωνα με την θέση κατάταξής του. Η πιθανότητα επιλογής είναι ανάλογη της τιμής καταλληλότητας.

  - **Επιλογή διαγωνισμού** (Tournament selection):
    Στην πιο απλή του μορφή επιλέγονται τυχαία *m* χρωμοσώματα από τον πληθυσμό, όπου *m* είναι μία παράμετρος που ονομάζεται μέγεθος διαγωνισμού (tournament size) και μπορεί να λάβει τιμές από 2 έως N(όπου N το μέγεθος του πληθυσμού. Στη συνέχεια τα επιλεγμένα χρωμοσώματα συμμετέχουν σε έναν διαγωνισμό ενός νικητή. Ο νικητής επιλέγεται ώς γονέας, και η διαδικασία επαναλαμβάνεται μέχρις ότου συμπληρωθεί ο αριθμός γονέων που απαιτείται.

  - **Ελιτισμός** (Elitism)
    Στην μέθοδο του ελιτισμού επιλέγεται να αντιγραφεί ένα μικρό ποσοστό από τα ικανότερα χρωμοσώματα αυτούσιο, χωρίς αλλαγές, στην επόμενη γενεά. Αυτό μπορεί να επηρεάσει σημαντικά την απόδοση του εξελικτικού αλγορίθμου, αποτρέποντας το χάσιμο της ικανότερης ως τώρα λύσης.

- **Αναπαραγωγή** (Reproduction)
  Μετά το τέλος της διαδικασίας της επιλογής οι επιλεγμένοι γονείς συνδυάζονται για την δημιουργία απογόνων (νέων ατόμων της επόμενης γενεάς του πληθυσμού). Η αναπαραγωγή πραγματοποιείται κυρίως μέσω δύο μηχανισμών, γνωστών και ώς γενετικοί τελεστές (genetic operators), την διασταύρωση και την μετάλλαξη.

  - **Διασταύρωση** (Crossover)
    Η διασταύρωση είναι μία διαδικασία σύνθεσης νέων ατόμων. Ο κάθε γονέας χωρίζεται σε δύο ή περισσότερα τμήματα, και στη συνέχεια συνδυάζονται αυτά τα τμήματα (“γενετικό υλικό”) για να δημιουργηθούν απόγονοι. Σκοπός είναι ότι με την διασταύρωση καλών ενδεχόμενων λύσεων, θα παραχθούν νέες, καλύτερες λύσεις.

  - **Μετάλλαξη** (Crossover)
    Η διαδικασία της μετάλλαξης αλλάζει με τυχαίο τρόπο κάποιο σημείο του γενετικού υλικού ενός χρωμοσώματος. Συνήθως πραγματοποιείται μετά την διασταύρωση. Δημιουργεί μικρές παραλλαγές στη λύση και κύριος στόχος της είναι να βελτιώσει την έρευνα στον χώρο αναζήτησης λύσεων, αποκολλώντας τον αλγόριθμο από τοπικά, μη βέλτιστα, σημεία.

Ένα ζήτημα στην σχεδίαση ενός εξελικτικού αλγόριθμου είναι η επιλογή της κατάλληλης αναπαράστασης των λύσεων σε χρωμοσώματα και παίζει μεγάλο ρόλο στην απόδοση του αλγορίθμου.

3.3 Νευροεξέλιξη

Η νευροεξέλιξη (neuroevolution) (Floreano et al. 2008, Yao 1999) αφορά την εφαρμογή εξελικτικών αλγορίθμων σε τεχνητά νευρωνικά δίκτυα και εστιάζει στην βελτιστοποίηση τους. Οι εξελικτικοί αλγόριθμοι μπορούν να χρησιμοποιηθούν για να αλλάξουν την συμπεριφορά ενός τεχνητού νευρωνικού δικτύου μεταβάλλοντας τα βάρη των συνάψεων, την αρχιτεκτονική του δικτύου, τους κανόνες εκμάθησης καθώς και τις συναρτήσεις μεταφοράς (Yao 1999). Ένας αλγόριθμος νευροεξέλιξης επιλέγεται αντί ενός αλγορίθμου μάθησης διότι : α) μπορεί να συνεξελίξει πολλά χαρακτηριστικά ενός νευρωνικού δικτύου, β) έχει πιο ευέλικτες συναρτήσεις αξιολόγησης από ότι οι συναρτήσεις σφάλματος ή ενέργειας και γ) μπορεί να συνδυαστεί με αλγορίθμους μάθησης όπως για παράδειγμα η Χεμπιανή μάθηση (Hebbian learning), ή ακόμα να χρησιμοποιηθεί για να δημιουργήσει καινούργιους αλγορίθμους μάθησης (Floreano et al. 2008).

Η εκπαίδευση ενός τεχνητού νευρωνικού δικτύου για την επίλυση ενός υπολογιστικού προβλήματος περιλαμβάνει και την εύρεση των κατάλληλων παραμέτρων του δικτύου, όπως η τοπολογία του δικτύου ή/και τα βάρη των συναπτικών συνδέσεων. Η βασική ιδέα πίσω από την νευροεξέλιξη είναι η εκπαίδευση ενός δικτύου με έναν εξελικτικό αλγόριθμο, που όπως έχει

αναφερθεί είναι μια στοχαστική κλάση μεθόδων αναζήτησης εμπνευσμένες από την Δαρβινική εξέλιξη. Μία σημαντική σχεδιαστική επιλογή που πρέπει να ληφθεί είναι η γενετική αναπαράσταση (δηλαδή του γονοτύπου) του νευρωνικού δικτύου (φαινοτύπου), πάνω στην οποία θα εφαρμοστούν οι γενετικοί τελεστές τις διασταύρωσης και τις μετάλλαξης.

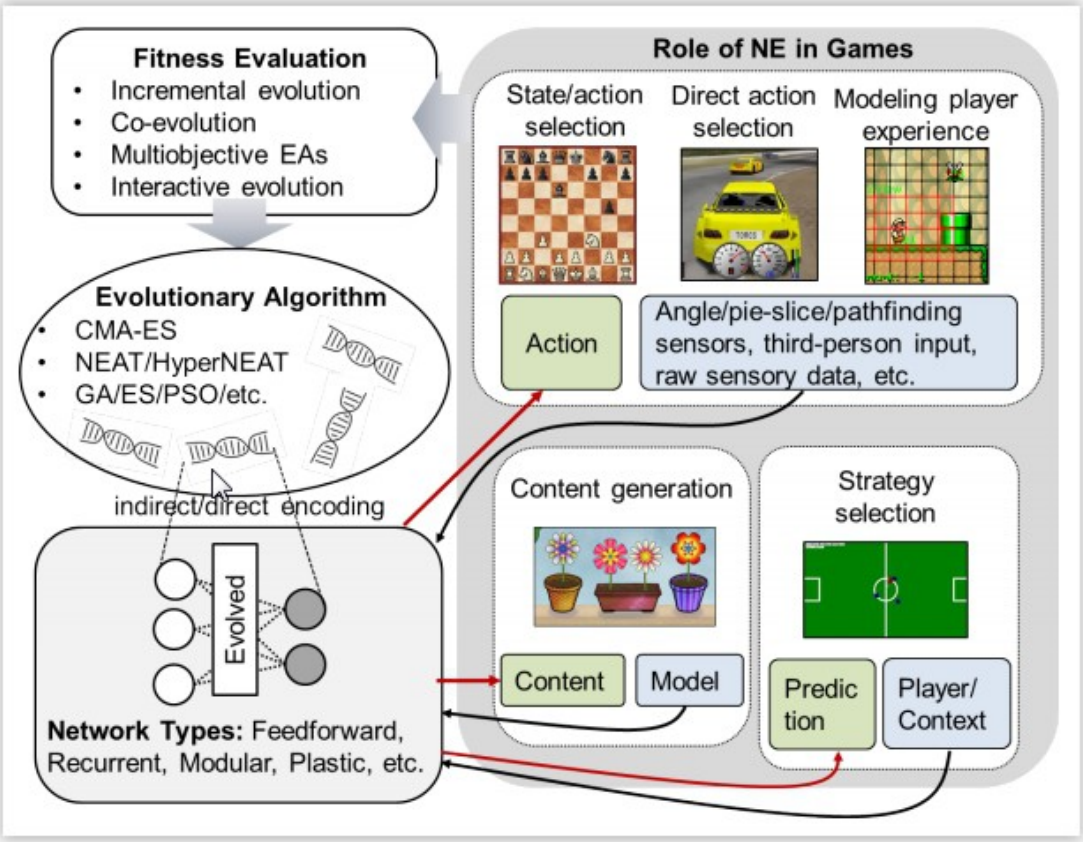
Για παράδειγμα, ένας από τους πιο απλούς και συνηθισμένους τρόπους εξέλιξης ενός νευρωνικού δικτύου είναι η εξέλιξη των βαρών των συνδέσεων του, ενώ κρατιέται σταθερή η αρχιτεκτονική του δικτύου και οι συναρτήσεις ενεργοποίησης. Αυτό γίνεται κωδικοποιώντας τα συναπτικά βάρη στους γονότυπους των ατόμων (συνήθως με την διάταξη των αριθμητικών τιμών των βαρών του δικτύου σε ένα διάνυσμα) του πληθυσμού και αναθέτοντας τιμές καταλληλότητας στα άτομα σύμφωνα με την απόδοση του φαινοτύπου τους (τεχνητού νευρωνικού δικτύου παραγόμενο από τον συγκεκριμένο γονότυπο). Χρησιμοποιώντας τους γενετικούς τελεστές δημιουργούνται δίκτυα με καινούργια συναπτικά βάρη.

Στον βασικό αλγόριθμο της μεθόδου της νευροεξέλιξης αρχικά δημιουργείται ένας πληθυσμός από χρωμοσώματα με τους αντίστοιχους γονότυπους να αντιπροσωπεύουν/κωδικοποιούν τεχνητά νευρωνικά δίκτυα και εξελίσσεται με σκοπό την εύρεση ενός δικτύου (βάρη ή/και τοπολογία) που μπορεί να λύσει ένα υπολογιστικό πρόβλημα.

Τυπικά κάθε γονότυπος μετασχηματίζεται σε ένα τεχνητό νευρωνικό δίκτυο, το οποίο με την σειρά του δοκιμάζεται σε ένα συγκεκριμένο έργο για κάποιο χρονικό διάστημα. Στην συνέχεια καταγράφεται η απόδοση του κάθε νευρωνικού δικτύου και αφού αντιστοιχιστούν οι τιμές αυτές στα αντίστοιχα άτομα του πληθυσμού, δημιουργείται ένας καινούργιος πληθυσμός μεταβάλλοντας ελαφρώς τους γονότυπους των ατόμων (μετάλλαξη) ή/και συνδυάζοντας διαφορετικούς γονότυπους (διασταύρωση). Γενικά τα άτομα με υψηλότερη απόδοση έχουν περισσότερες πιθανότητες να επιλεγούν για αναπαραγωγή και οι απόγονοι τους να αντικαταστήσουν άτομα που είχανε χαμηλότερη απόδοση. Αυτός ο κύκλος δημιουργίας – αξιολόγησης – αναπαραγωγής των γενεών επαναλαμβάνεται εκατοντάδες ή χιλιάδες φορές με σκοπό την εύρεση όλο και καλύτερα αποδιδόμενων δικτύων.

3.3.1 Εφαρμόζοντας νευροεξέλιξη στα παιχνίδια

Η νευροεξέλιξη είναι μια σημαντική μέθοδος που έχει αυξανόμενη δημοτικότητα τα τελευταία χρόνια και έχει ήδη πολλές εφαρμογές σε παιχνίδια και πολλούς άλλους τομείς. Μία ουσιώδεις διαφορά μεταξύ των προσεγγίσεων της νευροεξέλιξης είναι ο ρόλους που αυτή έχει σε ένα παιχνίδι, διότι είναι στενά συνδεδεμένη με τις εισόδους που δέχεται το εξελισσόμενο δίκτυο και τις εξόδους που παράγει. Ο ρόλος της νευροεξέλιξης επίσης επηρεάζει άμεσα τον τύπο της συνάρτησης καταλληλότητας που θα χρησιμοποιηθεί. Διαφορετικοί εξελικτικοί αλγόριθμοι υποστηρίζουν διαφορετικούς τύπους δικτύου, και μερικές μέθοδοι μπορεί να είναι πιο κατάλληλες από άλλες ανάλογα με τον τύπο αναπαράστασης των εισόδων.



Εικόνα 3.2 : Εφαρμογή της νευροεξέλιξη στα παιχνίδια

Οι Risi και Togelius (2015) αναφέρουν τα πλεονεκτήματα επιλογής της νευροεξέλιξης στον μηχανισμό των παιχνιδιών καθώς και των ρόλο που αυτή παίζει όταν εφαρμόζεται στα παιχνίδια.

Η νευροεξέλιξη είναι μια πολύ καλή γενική μέθοδος που μπορεί να εφαρμοστεί σε μια πληθώρα πεδίων στα παιχνίδια όπως στρατηγικές πρακτόρων, μοντελοποίηση των παιχτών, δυναμική δημιουργία περιεχομένου κ.α.. Σε όλες τις περιπτώσεις η εφαρμογή της μεθόδου παρουσιάζει ιδιαίτερο ενδιαφέρον. Μερικά από τα προτερήματα επιλογής της παρουσιάζονται παρακάτω.

1. **Επιδόσεις ρεκόρ (Record-beating performance)**
   Σε μερικά από τα προβλήματα, η νευροεξέλιξη παρέχει την καλύτερη επίδοση συγκρινόμενη με άλλες μεθόδους μάθησης (η επίδοση βεβαίως ορίζεται διαφορετικά ανάλογα με τον τύπου του προβλήματος). Ένα παράδειγμα αυτού μπορεί να εντοπιστεί στο πρόβλημα ισορρόπησης στύλων (pole balancing problem), ένα κλασσικό πρόβλημα ενισχυτικής μάθησης. Η μέθοδος νευροεξέλιξης CoSyNE 'νίκησε' όλες τις άλλες μεθόδους στις περισσότερες παραμέτρους του προβλήματος (Gomez et al. 2008, αναφορά από Risi και Togelius (2015)). Συγκεκριμένα βρήκε λύσεις στο πρόβλημα χρησιμοποιώντας λιγότερες δοκιμές από οποιονδήποτε άλλο αλγόριθμο.

2. **Ευρείας εφαρμοσιμότητας (Broad applicability)**
   Η νευροεξέλιξη μπορεί να εφαρμοστεί σε προβλήματα εκπαίδευσης με επίβλεψη, χωρίς επίβλεψη καθώς και ενισχυτικής μάθησης. Το μόνο που χρειάζεται είναι μία αριθμητική αξιολόγηση της ποιότητας των υποψήφιων δικτύων της. Σε αυτό το πλαίσιο, μοιάζει με αλγορίθμους ενισχυτικής μάθησης όπως για παράδειγμα η οικογένεια αλγορίθμων μάθησης χρονικών διαφορών. Εάν δίνεται ένα σετ δεδομένων με γνωστές τις επιθυμητές τιμές, τότε η νευροεξέλιξη μπορεί να χρησιμοποιηθεί σαν ένας αλγόριθμος επιβλεπόμενης μάθησης παρόμοιος με το τρόπο χρήσης του backpropagation.

3. **Επεκτασιμότητα (Scalability)**
   Συγκρινόμενη με άλλους τύπους ενισχυτικής μάθησης, η νευροεξέλιξη φαίνεται να μπορεί να χειριστεί πολύ μεγάλους χώρους δράσεων/καταστάσεων πολύ καλά, ειδικά όταν χρησιμοποιείται για άμεση επιλογή δράσης (direct action selection) (Risi και Togelius 2015).

4. **Ποικιλομορφία (Diversity)**
   Η νευροεξέλιξη μπορεί να βασιστεί πάνω στην πλούσια οικογένεια μεθόδων διατήρησης της ποικιλομορφίας (όπως το niching : τεχνικές που προωθούν τον σχηματισμό και την συντήρηση σταθερών υπό-πληθυσμών σε έναν γενετικό αλγόριθμο), καθώς και στις μεθόδους πολλαπλών στόχων (multiobjective methods) που έχουν αναπτυχθεί στην κοινότητα της εξελικτικής υπολογιστικής. Αυτό επιτρέπει στις μεθόδους νευροεξέλιξης να επιτύχουν διάφορες μορφές ποικιλομορφίας στις λύσεις και έτσι να παρέχουν σετ διαφορετικών ουσιωδών στρατηγικών, πρακτόρων, μοντέλων και/η περιεχομένου (Agapitos et. al. 2008, van Hoorn et al. 2009, αναφορά από Risi και Togelius 2015)

5. **Ανοιχτού τύπου μάθηση (Open-ended learning)**
   Παρότι η νευροεξέλιξη μπορεί να χρησιμοποιηθεί για ενισχυτική μάθηση με τον ίδιο τρόπο που χρησιμοποιείται η μέθοδος χρονικών διαφορών (Temporal Difference Learning), θα μπορούσε κανείς να υποστηρίξει ότι θα μπορούσε να φτάσει πιο μακρυά. Ειδικά σε περιπτώσεις όπου εξελίσσεται και η τοπολογία του δικτύου, η νευροεξέλιξη θα μπορούσε (στην θεωρία τουλάχιστον) να υποστηρίξει εξέλιξη ανοιχτού τύπου, όπου θα μπορούσαν να προκύψουν συμπεριφορές αυθαίρετης πολυπλοκότητας και εξειδίκευσης. Οι αλγόριθμοι νευροεξέλιξης συχνά ψάχνουν σε μεγαλύτερο χώρο από τους αλγόριθμους χρονικών διαφορών.

6. **Επιτρέπει την δημιουργία καινούργιων τύπων παιχνιδιών**
   Καινούργια παιχνίδια όπως το Galactic Arms Race (GAR: Hastings et. al. 2009), στο οποίο ο παίχτης μπορεί διαδραστικά να εξελίξει συγκεκριμένα όπλα, το NERO (Stanley et. al. 2005), που δίνει την δυνατότητα στον χρήστη να εξελίξει μια ομάδα από ρομπότ και να πολεμήσει με αυτά άλλους παίχτες, καθώς και το βιντεοπαιχνίδι Petalz (Risi et. al. 2012 & 2015), στο οποίο ο παίχτης μπορεί να αναθρέψει μια ατελείωτη ποικιλία

εικονικών λουλουδιών, θα ήταν δύσκολο να πραγματοποιηθούν με τις "παραδοσιακές" μεθόδους μάθησης. Η εξελικτική υπολογιστική εδώ παρέχει μοναδικές ευκαιρίες για σχεδιασμό παιχνιδιών. Παιχνίδια όπως το GAR και το Petalz εξαρτώνται από την συνεχή περιπλοκή του παραγόμενου περιεχομένου, η οποία είναι υποστηριζόμενη και αποτελεί αναπόσπαστο μέρος ορισμένων μεθόδων νευροεξέλιξης. Επιπλέον σε παιχνίδια όπως το Petalz, του οποίο ο βασικός πυρήνας του μηχανισμού είναι η καλλιέργεια και ανατροφή καινούργιων ειδών λουλουδιών, η χρήση μεθόδων εξελικτικής υπολογιστικής αποτελεί φυσική επιλογή. Το γεγονός ότι οι μέθοδοι νευροεξέλιξης ενσωματώνουν ένα μεγαλύτερο στοιχείο εξερεύνησης υποστηρίζοντας έτσι την open-ended μάθηση, τις καθιστά απευθείας εφαρμόσιμες σε καινούργιους τύπους παιχνιδιών.

3.3.2 Ο ρόλος της νευροεξέλιξης στα παιχνίδια

Στην πληθώρα των περιπτώσεων η νευροεξέλιξη χρησιμοποιείται είτε για την μάθηση του τρόπου παιξίματος ενός παιχνιδιού είτε για τον έλεγχο κάποιου NPC (Non-Player Character) στο παιχνίδι (Risi και Togelius 2015). Το νευρωνικό δίκτυο σε αυτές τις περιπτώσεις μπορεί έχει να έναν από τους παρακάτω δύο ρόλους: α) να αποτιμά την αξία των καταστάσεων ή δράσεων, έτσι ώστε στη συνέχεια κάποιος άλλος αλγόριθμος να αναλάβει την επιλογή της καταλληλότερης δράσης, β) να επιλέγει άμεσα τις δράσεις που πρέπει να γίνουν σε μια δεδομένη κατάσταση. Εκτός από αυτούς του δύο όμως ρόλους, η νευροεξέλιξη μπορεί να χρησιμοποιηθεί και για άλλους σκοπούς. Η διαδικαστική δημιουργία περιεχομένου (Procedural Content Generation – PCG) είναι μία ενεργή περιοχή έρευνας στον κλάδο της τεχνητής νοημοσύνης στα παιχνίδια, όπου τα εξελίξιμα νευρωνικά δίκτυα μπορούν να χρησιμοποιηθούν για να αναπαραστήσουν περιεχόμενο. Επιπλέον, η νευροεξέλιξη μπορεί να χρησιμοποιηθεί για να προβλεφτεί η εμπειρία ενός παίχτη καθώς και οι προτιμήσεις του, λαμβάνοντας έτσι ενεργό ρόλο στην μοντελοποίηση αυτών.

Παρακάτω παρουσιάζεται ένας συνοπτικός πίνακας που αναφέρει τους ρόλους της νευροεξέλιξης σε ένα πλήθος παιχνιδιών καθώς και τον τρόπο με τον οποίο αυτή χρησιμοποιήθηκε.

NE Role	Game	ANN Type	NE Methods	Fitness Evaluation	Input Representation
State/Action evalutation	Checkers	MLP	UD, GA	Coevolution	TP (piece type)
	Chess	MLP	UD, GA	PA (positional values)	TP (piece type)
	Othello	MLP	Marker-based	Cooperative Coevolution	TP (piece type)
	Go (7x7)	CPPN(MLP)	HyperNeat	PA (score+board size)	TP (piece type)
	Ms. Pac-Man	MLP	UD, ES	PA (average score)	Path-finding
	Simulated Car Racing	MLP	UD, ES	PA (waypoints visited)	Speed, pos, waypoints
	Ouake II	MLP	UD, GA	PA (kill count)	Visual Input (14x2)
	Unreal Tournament	Recurrent, LSTM	UD, GA, NSGA-II	MO (damage+accuracy)	Pie-slice, waypoints, etc.
Direct action selection	Go (7x7)	MLP	NEAT	Transfer Learning	Roving Eye (3x3)
	Simulated Car Racing	MLP	UD, ES	Incremental Evolution	Rangefinders, waypoints
	Keepaway Soccer	MLP	NEAT	Transfer Learning	Distances
	Battle Domain	MLP	NEAT, NSGA-II	MO+Incremental	Angle, straight line
	NERO	MLP	NEAT	Interactive Evolution	Rangefinders, pie-slice
	Ms. Pac-Man	Modular MLP	NEAT, NSGA-II	MO (pills+ghosts eaten)	Path-finding
	Simulated Car Racing	MLP	UD, GA	PA (distance)	Roving Eye (5x5)
	Atari	CPPN (MLP)	HyperNEAT	PA (game score)	Raw inpute (16x21)
	Creatures	Modular MLP	GA	Interactive Evolution	TP (e.g. type of object
Selection between strategies	Keepaway Soccer	MLP	NEAT	PA (hold time)	Angle and distance
	EvoCommander	MLP	NEAT	Interactive Evolution	Pie-slice, rangefinder
Modelling oponnent strategy	Texas Hold'em Poker	MLP	NEAT	PA (%hands won)	TP (e.g. size of pot, cost of a bet, etc.)
Content generation	GAR	CPPN (MLP)	NEAT	Interactive Evolution	Model
	Petalz	CPPN (MLP)	NEAT	Interactive Evolution	Model
Modelling player experience	Super Mario Bros	MLP, Perceptron	UD, GA	PA (player preference)	TP (e.g. gap width, number of deaths, etc.)

Πίνακας 3.1 : Οι ρόλοι της Νευροεξέλιξης σε επιλεγμένα παιχνίδια

ES = evolutionary strategy, GA = genetic algorithm, MLP = multi-layer perceptron, MO = multiobjective, TP = third-person (input not tied to a specific frame of reference, e.g. number of edible ghosts), UD = user-define topology, PA = performance alone

Για περισσότερες πληροφορίες σχετικά με τον ρόλο και την χρήση της νευροεξέλιξης στα παιχνίδια μπορεί κανείς να ανατρέξει στο άρθρο των Risi και Togelius (2015), *Neuroevolution in Games: State of the Art and Open Challenges,* καθώς και για μια πλήρη ανασκόπηση και ανάλυση των διαφορετικών πεδίων εφαρμογής της υπολογιστικής και τεχνητής νοημοσύνης στα παιχνίδια στο άρθρο των Yannakakis και Togelius (2014), *A Panorama of Artificial and Computational Intelligence In Games*.

3.4 Συναφείς μελέτες

Ο Lucas (2005) πρότεινε την εξέλιξη Νευρωνικών Δικτύων με Εξελικτικές Στρατηγικές για την αξιολόγηση των κινήσεων στο παιχνίδι Ms. PacMan.

Οι Cardamone, Loiacono και Lanzi (2009) παρουσίασαν ελεγκτές για το αγωνιστικό παιχνίδι αυτοκινήτων TORCS (The Open Car Racing Simulator) βασισμένους στην μέθοδο νευροεξέλιξης NEAT (NeuroEvolution of Augmenting Topologies). Χρησιμοποίησαν την NEAT για να εξελίξουν χωριστά δύο στρατηγικές (γρήγορη και αξιόπιστη οδήγηση στο οδόστρωμα & την ικανότητα προσπέρασης αντιπάλων και αποφυγής συγκρούσεων) τις οποίες συνδύασαν σε έναν χειριστή. Έδειξαν ότι ο χειριστής τους που συνδύαζε και τις δύο μεθόδους κατάφερε να παίξει καλύτερα από όλους τους διαθέσιμους χειριστές που είχε η πλατφόρμα και που είχαν κατατεθεί στο διαγωνισμό.

Οι Togelius et. al. (2009) χρησιμοποίησαν Τεχνητά Νευρωνικά Δίκτυα (Προσθιας τροφοδότησης καθώς και Αναδρομικά) με μ + λ Εξελικτικές Στρατηγικές , καθώς και την μέθοδο HyperGP (Buk et. al. 2009), για να εξελίξουν πράκτορες για την πλατφόρμα που ανάπτυξαν βασισμένη στο Infinite Mario Bros του Markus Persson.

ΚΕΦΑΛΑΙΟ 4

4 MarioAI Benchmark

Το Mario AI Benchmark (Togelius, Karakovskiy et. al. 2009) είναι μία πλατφόρμα ανοιχτού λογισμικού εμπνευσμένη από το κλασσικό παιχνίδι της Nintendo, Super Mario Bros, και βασισμένη στην υλοποίηση του Infinite Mario Bros του Markus Persson. Δημιουργήθηκε για να δώσει σε ερευνητές ένα πεδίο δοκιμών μεθόδων και αλγορίθμων σε ένα παιχνίδι που είναι απλό να μαθευτεί, δύσκολο να κατακτηθεί, οπτικά ευχάριστο και μπορεί να παιχτεί από πράκτορας καθώς και από ανθρώπους (Tønder και Olsen 2013).

Σκοπός του παιχνιδιού είναι να καθοδηγήσει ο παίχτης τον Mario μέσα από δισδιάστατα επίπεδα που παρακολουθούνται από το πλάι. Σε κάθε επίπεδο ο παίχτης ξεκινάει στο αριστερότερο σημείο της πίστας και πρέπει να την διανύσει έτσι ώστε να φτάσει στο τέλος. Ο Mario μπορεί να περπατήσει και να τρέξει προς τα αριστερά ή τα δεξιά, να εκτελέσει άλμα και (ανάλογα με την κατάσταση στην οποία βρίσκεται) να εκτοξεύσει φωτιές. Ο Mario μπορεί να βρίσκεται σε μία από τρεις καταστάσεις: α) Μικρός (στην αρχή του παιχνιδιού), β) Μεγάλος (αφού λάβει ένα μανιτάρι, σε αυτήν την κατάσταση μπορεί να σπάει τούβλα αν τα χτυπήσει από κάτω), γ) Φωτιάς (αφού λάβει ένα λουλούδι φωτιάς μπορεί να εκτοξεύει μπάλες φωτιάς.

Δευτερεύοντες στόχοι του παιχνιδιού αποτελούν οι συγκέντρωση μέγιστου αριθμού νομισμάτων, τελειώνοντας ένα επίπεδο στο μικρότερο δυνατό χρόνο, μεγιστοποιώντας του πόντους κάθε πίστας, σκοτώνοντας τους περισσότερους εχθρούς κα..

Περιπλέκοντας την κατάσταση κάθε επίπεδο μπορεί να έχει έναν αριθμό από πλατφόρμες, τούβλα, πηγάδια/τρύπες και κινούμενους εχθρούς με διαφορετικές ικανότητες και συμπεριφορές. Ο Mario χάνει μια ζωή εάν πέσει μέσα σε μία τρύπα ή εάν τελειώσει ο χρόνος τις πίστας. Εάν έρθει σε επαφή με έναν εχθρό ή εχθρικό αντικείμενο τότε πληγώνεται (εάν ήταν στην κατάσταση μικρός, πεθαίνει, διαφορετικά "υποβιβάζεται" στην προηγούμενη κατάσταση). Εκτός από τις καταστάσεις που αναφέρθηκαν προηγουμένως υπάρχουν και μερικές συμπληρωματικές όπως για παράδειγμα το κράτημα μιας χελώνας (υπάρχουν κάποιες χελώνες που την πρώτη φορά που θα προσγειωθεί επάνω τους ο Mario αυτές μπαίνουν στο καβούκι τους, και στην συνέχεια μπορεί να τις σηκώσει και χρησιμοποιήσει ως βλήμα, εκτοξεύοντας τες σε εχθρούς.)

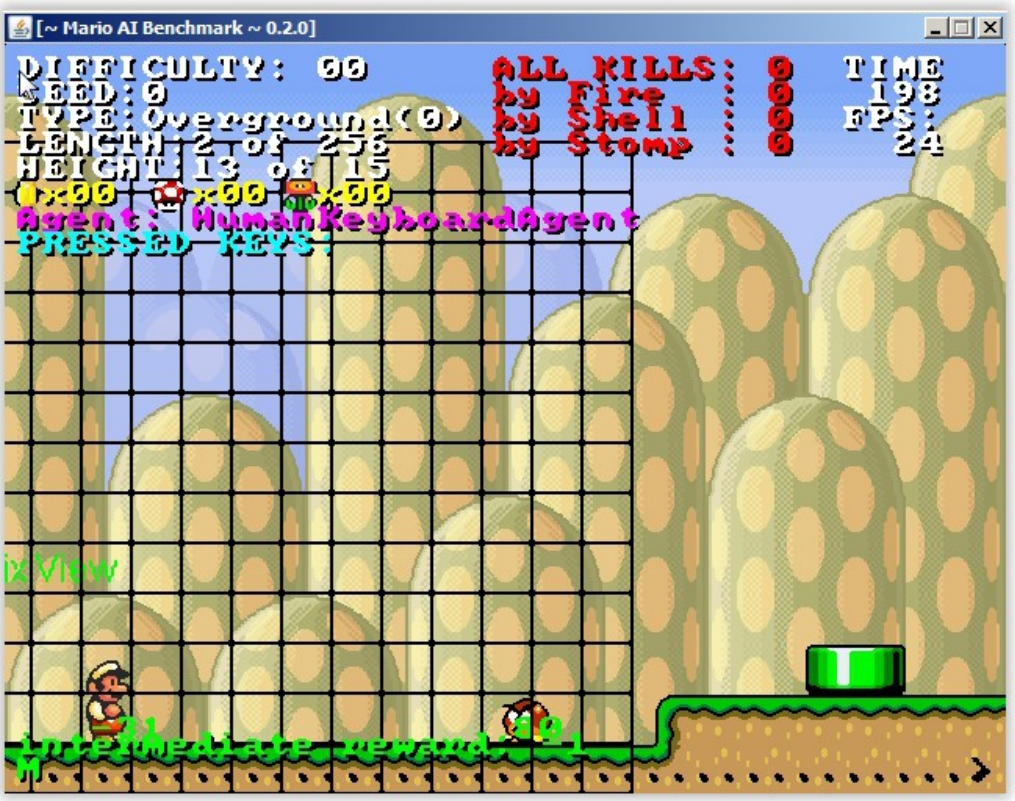
Στο Mario AI τα επίπεδα χαρακτηρίζονται πρωτίστως από το επίπεδο δυσκολίας τους. Αφού αυτό οριστεί δημιουργούνται τυχαία επίπεδα κάθε φορά που ξεκινάει το παιχνίδι φτιάχνοντας ένα ορισμένο σε μήκος επίπεδο και προσθέτοντας χαρακτηριστικά όπως τρύπες, τούβλα, εχθρούς κλπ. Το API του Mario AI επιτρέπει στον προγραμματιστή να παραμετροποιήσει

πλήρως τα παραγόμενα επίπεδα. Κάθε φορά που δημιουργείται τυχαία ένα επίπεδο αυτό χαρακτηρίζεται από τον σπόρο του (seed) έτσι δίνεται η δυνατότητα αναπαραγωγής κάποιου τυχαίου επιπέδου κατά βούληση. Επιπλέον μπορούν να οριστούν οι τύποι εχθρών που θα περιέχει, το αν θα περιέχει κενά η όχι, το μήκος της πίστας, η δυσκολία και πολλά άλλα.

Για παράδειγμα χρησιμοποιώντας την μέθοδο *MarioAIOptions* μπορεί κανείς να δημιουργήσει ένα αντικείμενο *options* και στην συνέχεια να παραμετροποιήσει το επίπεδο όπως αυτός θέλει.

```
MarioAIOptions option = new MarioAIOptions(new String[0]);
options.setLevelDifficulty(1); /*Επίπεδο δυσκολίας 1*/
options.setEnemies("le g"); /*Να περιέχει μόνο εχθρούς goomba*/
options.setLevelLength(300);/*Να έχει μήκος 300 κελιά*/
options.setFlatLevel(true); /*Να μην έχει κενά*/
κ.α.
```

Πολλά χαρακτηριστικά κάνουν το Super Mario Bros ένα ιδιαίτερα ενδιαφέρον παιχνίδι από την σκοπιά της Ενισχυτικής Μάθησης. Το πιο σημαντικό από αυτά είναι η πολύ πλούσια και υψηλή σε διαστάσεις αναπαράσταση του περιβάλλοντος. Όταν ένας άνθρωπος παίζει το παιχνίδι, βλέπει μόνο ένα μικρό μέρος της πίστας από το πλάι, με την οθόνη κεντρισμένη στον Mario. Ακόμα και αυτή η οπτική περιλαμβάνει πολλές δεκάδες αντικείμενα όπως τούβλα, εχθρούς και συλλέξιμα αντικείμενα. Αυτά τα αντικείμενα είναι μοιρασμένα σε έναν ήμι-συνεχή τρόπο: τα στατικά αντικείμενα περιβάλλοντος (όπως το γρασίδι, αγωγοί νερού, τούβλα κλπ.) και τα νομίσματα είναι διαμορφωμένα σε ένα "πλέγμα" (όπου η απλή οθόνη καλύπτει περίπου 15x15 κελιά), ενώ τα κινούμενα αντικείμενα (οι περισσότεροι εχθροί καθώς και τα μανιτάρια) κινούνται σχεδόν συνέχεια στο επίπεδο των pixels. (Togelius et. al. 2009)



Εικόνα 4.1 : Το πλέγμα μέσα στο οποίο ο Mario μπορεί να 'αισθάνεται' το περιβάλλον

Ένας πράκτορας έχει πρόσβαση σε πληροφορίες σχετικά με το περιβάλλον και την κατάσταση του παιχνιδιού σε κάθε χρονικό βήμα (time frame). Αυτές οι πληροφορίες μπορούν να ανακτηθούν με διάφορους τρόπους όπως μέσω παραμέτρων για την κατάσταση του Mario καθώς και από ένα οπτικό πλέγμα γύρω από τον Mario (βλ. Εικόνα 4.1). Αυτό το πλέγμα είναι ένας πίνακας από τιμές που βασίζονται στο τι περιέχεται σε κάθε κελί. Τα κελιά που είναι άδεια περιέχουν την τιμή 0, ενώ τα κελιά με τον Mario, εχθρούς, τούβλα, έδαφος κλπ. έχουν μη μηδενικές τιμές που εξαρτώνται από τον τύπο του αντικειμένου στο κάθε κελί.

Παραδείγματα μεθόδων που ανήκουν στην διεπαφή *Environment* που μπορούν να χρησιμοποιηθούν για την ανάκτηση αυτών των πληροφοριών δείχνονται παρακάτω:

```
public byte[][] getMergedObservationZZ(int ZlevelScene,int
ZLevelEnemies);
public byte[][] getEnemiesObservation(int ZLevelScene);
public byte[][] getLevelSceneObservationZ(int ZLevelEnemies);
public float[] getMarioFloatPos();
public float[] getEnemiesFloatPos();
public boolean isMarioOnGround();
public boolean isMarioAbleToJump();
public boolean isMarioAbleToShoot();
```

Η μεταβλητή Z (ZoomLevel) αντιπροσωπεύει τον βαθμό λεπτομέρειας της παρατήρησης. (Απο την πιό λεπτομερή, που κάθε αντικείμενο και εχθρός έχει την δικιά του τιμή, μέχρι μια πιο γενική όπου τα αντικείμενα ομαδοποιούνται σε κατηγορίες.)

Το Super Mario Bros είναι ένα παιχνίδι με δυναμικά μεταβαλλόμενο περιβάλλον, όπου ο παίχτης βλέπει μόνο ένα μικρό μέρος κάθε επιπέδου κάθε δεδομένη χρονική στιγμή. Αυτό σημαίνει ότι ο πράκτορας θα πρέπει να λειτουργεί σε ένα περιβάλλον που αλλάζει ραγδαία καθώς ο Mario διανύει το επίπεδο. Επίσης το παιχνίδι τρέχει σε πραγματικό χρόνο, δίνοντας στον πράκτορα πολύ λίγο χρόνο για να υπολογίσει ποια θα είναι η επόμενη του κίνηση, κάνοντας δύσκολη έτσι την αναζήτηση της πιθανής κίνησης κοιτάζοντας μπροστά πολλά βήματα. Το παιχνίδι έχει μια ομαλή καμπύλη μάθησης μεταξύ των επιπέδων όσον αναφορά τις συμπεριφορές που χρειάζονται για να περαστεί το κάθε επίπεδο. Τα πολύ απλά επίπεδα για παράδειγμα χωρίς εχθρούς και μόνο με μερικά εμπόδια και τρύπες απαιτούν από έναν πράκτορα να προχωράει συνέχεια προς τα δεξιά και να μπορεί να πηδάει όποτε βρίσκει μία τρύπα ή έναν εμπόδιο. Για να μπορέσει όμως να γίνει πολύ καλός στο παιχνίδι και να περνάει το κάθε επίπεδο, ένας πράκτορας πρέπει να μπορεί να μάθει τις διαφορετικές ιδιότητες του κάθε εδάφους και εχθρού και το πώς να τους περάσει και να αλληλεπιδράσει μαζί τους. Για παράδειγμα μερικοί εχθροί δεν σκοτώνονται άμα πατηθούν παρά μόνο με φωτιά, καθώς επίσης και τα τούβλα δεν σπάνε παρά μόνο όταν ο Mario είναι στην κατάσταση Μεγάλος ή Φωτιά και τα χτυπήσει από κάτω.

ΚΕΦΑΛΑΙΟ 5

5 Σχεδιασμός και ανάπτυξη του πράκτορα

5.1 Ελέγχοντας τον Mario με τεχνητά νευρωνικά δίκτυα

Ο κάθε χειριστής για το Mario AI ορίζεται από το αντίστοιχο τεχνητό νευρωνικό δίκτυο. Σε κάθε χρονικό βήμα (24 ανά δευτερόλεπτο) ο χειριστής λαμβάνει πληροφορίες για το περιβάλλον και την κατάσταση του παιχνιδιού από την διεπαφή "Environment" και ανταποκρίνεται με μία δράση. Αυτή η δράση αντιπροσωπεύει το αν θα ενεργοποιηθεί κάθε ένα από τα 6 διαθέσιμα κουμπιά. Τέσσερα κουμπιά είναι για τις κινήσεις αριστερά, δεξιά, πάνω, κάτω και δύο για να κάνει άλμα και να τρέξει/πυροβολήσει. Το κουμπί για την κίνηση επάνω, εφαρμόζεται μόνο στην περίπτωση που υπάρχει σκάλα για να ανέβει ο Mario. Η μελέτη περιορίστηκε στα αρχικά επίπεδα δυσκολίας που δεν περιλαμβάνουν σκάλες και η δράση για αυτό το κουμπί ήταν πάντα ΨΕΥΔΗΣ. Έτσι αντίστοιχα περιορίστηκαν και οι νευρώνες εξόδου του νευρωνικού δικτύου. Ο χειριστής χρησιμοποιεί το ΤΝΔ για να αποφασίσει ποια δράση θα εκτελέσει βασισμένος στις πληροφορίες που εισπράττει από το περιβάλλον.

```
-------------------------------------------------------
Αλγοριθμός Επιλογή Δράσης
-------------------------------------------------------
   Είσοδος: παρατήρηση σ1, νευρωνικό δίκτυο ν1
   Αρχικοποίηση έξοδοι = float[], είσοδοι = float[]
   είσοδοι = Observation(σ1)
   έξοδοι = ν1.activate(είσοδοι)
   Έξοδος Επιλογή δράσης(έξοδοι)
-------------------------------------------------------
```

Η επίδοση του πράκτορα επηρεάζεται από τις εισόδους του ΤΝΔ (δλδ τα στοιχεία που του δίνονται από την παρατήρηση του περιβάλλοντος) και τα βάρη του ΤΝΔ (δλδ τις συνδέσεις μεταξύ των νευρώνων). Η συμπεριφορά του πράκτορα και η πολιτική που ακολουθεί αλλάζει μεταβάλλοντας τα βάρη του ΤΝΔ.

Το ΤΝΔ είναι ένα Πολυεπίπεδο Δίκτυο Πρόσθιας Τροφοδότησης με το κάθε επίπεδο να είναι πλήρως συνδεδεμένο με το επόμενο και με ένα κρυμμένο επίπεδο 10 νευρώνων. Η αύξηση των νευρώνων του κρυμμένου επιπέδου δεν έδειξε να βελτιώνει την απόδοση του χειριστή. Ο αριθμός των νευρώνων εισόδου καθώς και εξόδου αποτέλεσε μέρος της μελέτης της Διπλωματικής. Παρακάτω θα παρουσιαστούν οι διαφορετικές υποψήφιες υλοποιήσεις και στο Κεφάλαιο 7 αναλυτικά η διαδικασία επιλογής της καταλληλότερης για την ανάπτυξη του πράκτορα.

5.1.1 Περιβάλλον και μέθοδοι

Όπως περιγράφτηκε στο Κεφάλαιο 4 , η πλατφόρμα του MarioAI δίνει την δυνατότητα στον χειριστή να πάρει πληροφορίες για το περιβάλλον με βάση ένα πλέγμα με ανάλυση 22x22 τετράγωνα γύρω από τον Mario. Ο Mario είναι πάντα στις συντεταγμένες 11,11 του πλέγματος. Για να μπορέσει να πάρει πληροφορίες για την κατάσταση του παιχνιδιού θα πρέπει να κοιτάξει τις τιμές του πλέγματος. Για παράδειγμα για να δει τι υπάρχει ακριβώς μπροστά από τον Mario ο χειριστής θα κοιτάξει στο τετράγωνο με συντεταγμένες 11,12. Η έξοδος του χειριστή θα είναι η δράση που θα πρέπει να εκτελέσει ο Mario σε μορφή δυαδικού πίνακα, πού θα δείχνει ποια κουμπιά αποτελούν την δράση.

Εάν χρησιμοποιούνταν ένας νευρώνας για κάθε τετράγωνο του πλέγματος θα είχαμε 484 νευρώνες εισόδου. Έγινε μια προσπάθεια μείωσης του χώρου αυτού, ενώ παράλληλα να μπορεί ο πράκτορας να λάβει αρκετές πληροφορίες για το τριγύρω περιβάλλον έτσι ώστε να μπορεί να βγάλει σωστά συμπεράσματα για την κατάλληλη δράση.

Για να μειωθεί ο χώρος των καταστάσεων εισόδου ορίστηκε η διεπαφή “Observation”. Μέσω αυτής επεξεργάζεται το περιβάλλον και δίνεται στους χειριστές μία διαίσθηση για το παιχνίδι παρόμοια με αυτή που θα είχε εάν έπαιζε ένας άνθρωπος. Ποιοι είναι οι κοντινότεροι εχθροί και πόσο κοντά είναι, εάν υπάρχει έδαφος από κάτω, εάν μπορεί ο Mario να εκτελέσει άλμα κλπ.

- ΕΧΘΡΟΙ
  Ορίστηκαν οι εξής μέθοδοι που δίνουν την δυνατότητα στον χειριστή να δει κοντινούς εχθρούς και να αξιολογήσει την επικινδυνότητα τους
  - `enemyInFrontDistance`
  - `enemyInFrontDistance_NStomp`
  - `enemyInFrontDistance_Stomp`
  - `enemyBehindDistance`
  - `enemyBehindDistance_NStomp`
  - `enemyBehindDistance_Stomp`
  - `enemyAboveDistance`
  - `enemyAboveDistance_NStomp`
  - `enemyAboveDistance_Stomp`
  - `enemyBelowDistance`

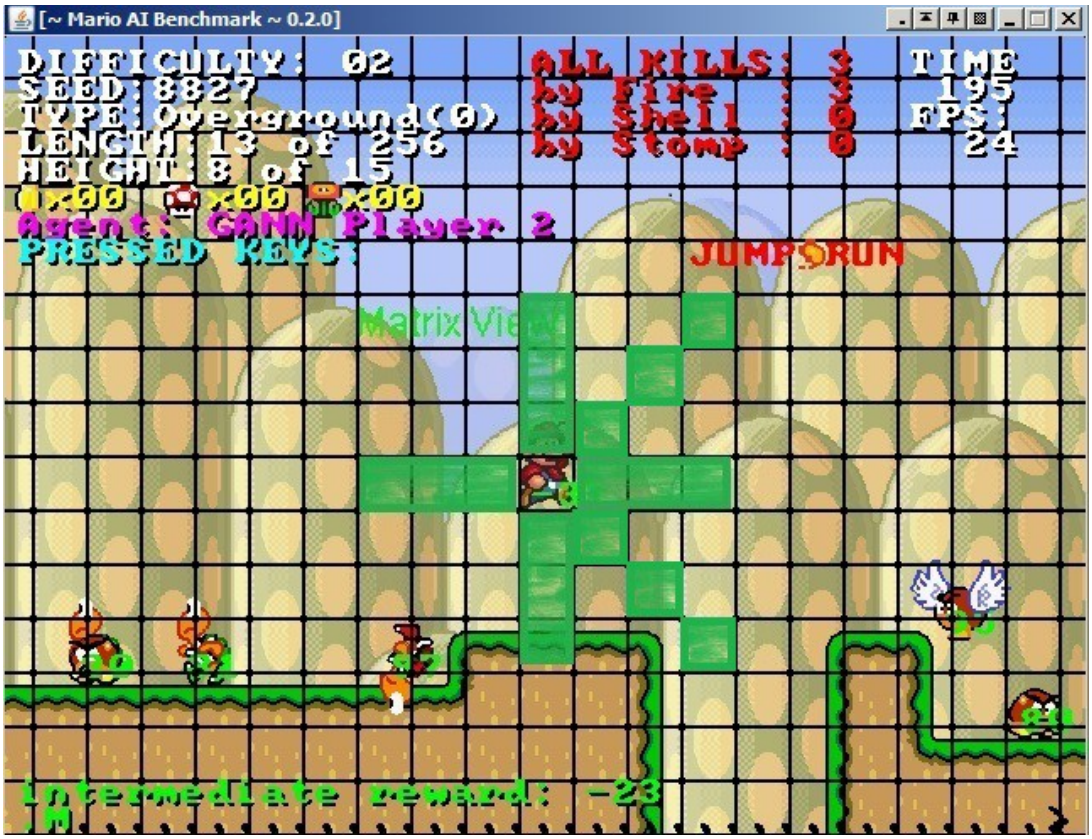
- enemyBelowDistance_NStomp
- enemyBelowDistance_Stomp
- enemyBellowKillable
- enemyInFrontKillable
- enemyRightDownKillable
- enemyUpRightDistance
- enemyDownRightDistance

Αυτές επιστρέφουν την απόσταση του εχθρού από τον Mario στην κατεύθυνση που υποδηλώνουν. Κοιτάνε τα τετράγωνα που απέχουν από τον Mario σύμφωνα με μια μεταβλητή viewdistance και επιστρέφουν μια κανονικοποιημένη τιμή στο διάστημα [0,1). Δοκιμάστηκαν διαφορετικές τιμές για την viewdistance και επιλέχτηκε η τιμή 3 βάση της ταχύτητας μάθησης. Εάν δεν υπάρχει εχθρός στα τετράγωνα που κοιτάζονται η τιμή επιστροφής της μεθόδου θα είναι 0. Όσο πιο κοντά βρίσκεται κάποιος εχθρός στην συγκεκριμένη κατεύθυνση τόσο πιο μεγάλη θα είναι η τιμή υποδηλώνοντας μεγαλύτερο κίνδυνο, με μέγιστη τιμή 0.99 στην περίπτωση που κάποιος εχθρός βρίσκεται στο αμέσως επόμενο τετράγωνο.

Οι _NStomp "κοιτάνε" μόνο για εχθρούς που δέν μπορεί να "πατήσει" ο Mario (δλδ να προσγειωθεί επάνω τους). Αντίστοιχα οι _Stomp επιστρέφουν τιμή μόνο στην περίπτωση που ο Mario μπορεί να πατήσει τον εχθρό εντός του πεδίου.

Η μέθοδοι enemyBellowKillable, enemyRightDownKillable, enemyInFrontKillable σε αντίθεση με τις NStomp και Stomp κοιτάζουν μόνο εάν ο εχθρός που βρίσκεται ακριβώς κάτω,διαγώνια κατω, μπροστα απο τον Mario μπορεί να πατηθεί και επιστρέφουν την τιμή 1, αλλιως επιστρέφουν 0.

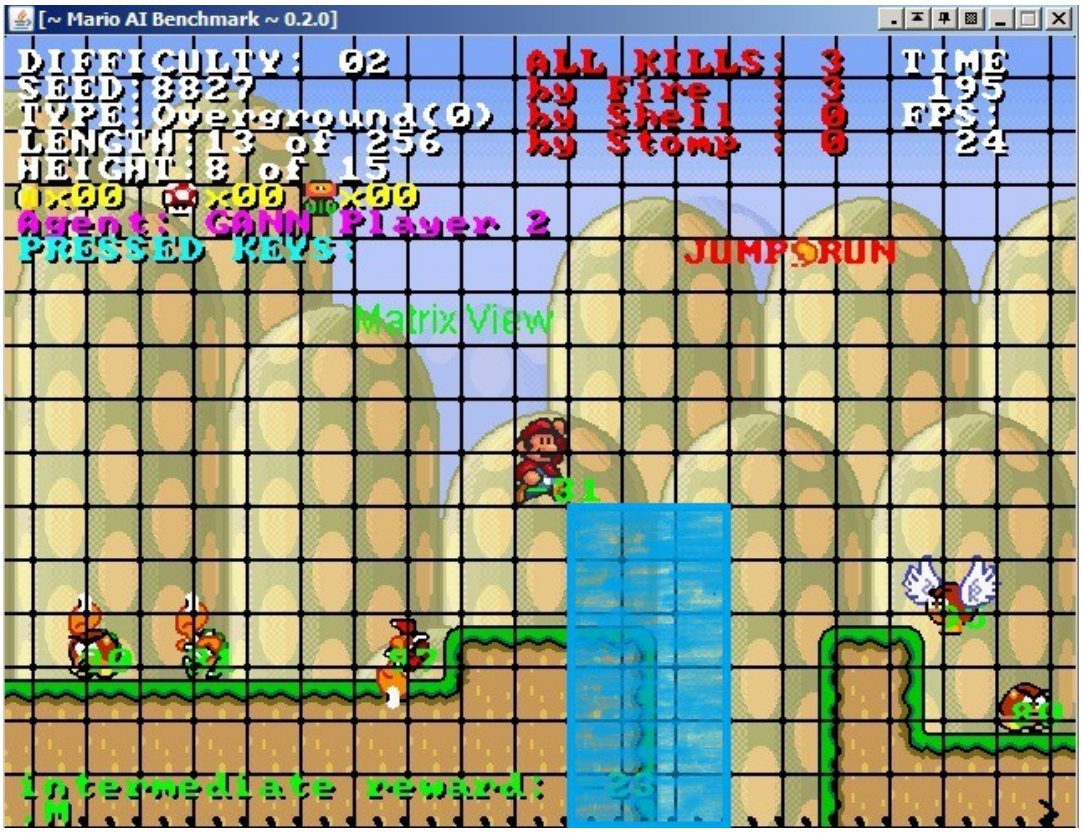
Παρακάτω βλέπουμε το πεδίο που μπορεί να δει κάποιος χειριστής με της μεθόδους.



Εικόνα 5.1 : Οπτικό Πεδίο Εχθρών

- ΠΕΡΙΒΑΛΛΟΝ
  Ορίστηκαν οι εξής μέθοδοι που δίνουν πληροφορίες στον χειριστή σχετικά με την μορφολογία της πίστας.
  - `obstacleDistance`
  - `obstacleHeight`
  - `obstacleAboveDistance`
  - `obstacleBelowDistance`
  - `gapDistance`
  - `sizeOfGap`
  - `flowerPotState`

  Οι μέθοδοι `*Distance` επιστρέφουν μια τιμή στο διάστημα [0,1) ανάλογα με την απόσταση του κοντινότερου εμποδίου/κενού. Οι μέθοδοι `sizeOfGap` και `obstacleHeight` επιστρέφουν μια τιμή στο διάστημα [0,1] ανάλογα με το μέγεθος του κενού/εμποδίου. Η `flowerPotState` επιστρέφει και αυτή μια τιμή στο διάστημα [0,1] κοιτάζοντας εάν υπάρχει μπροστά από τον Mario γλάστρα με λουλούδι και επιστρέφοντας την κατάστασή της. 0 σε περίπτωση που το λουλούδι βρίσκεται μέσα στην γλάστρα, 0.33 και 0.66 καθώς το λουλούδι βρίσκεται σε ενδιάμεσο στάδιο και 1 εάν είναι πλήρως έξω, και συνεπώς αποτελεί την μέγιστη απειλή για τον Mario.
  Παρακάτω βλέπουμε το πεδίο στο οποίο μπορούν να εντοπιστούν κενά στην πίστα.



Εικόνα 5.2 : Οπτικό Πεδίο Κενών

- ΚΑΤΑΣΤΑΣΗ ΤΟΥ ΜΑΡΙΟ
  Ορίστηκαν 2 μέθοδοι που μπορούν να δώσουν στον χειριστή την πληροφορία για το αν μπορεί ο Mario να πυροβολήσει, καθώς και για το αν είναι ικανός να κάνει άλμα. Σύμφωνα με τους κανόνες του παιχνιδιού εάν ο Mario βρίσκεται ήδη στον αέρα δεν του επιτρέπεται να εκτελέσει άλμα.
  - canMarioJump
  - canMarioShoot

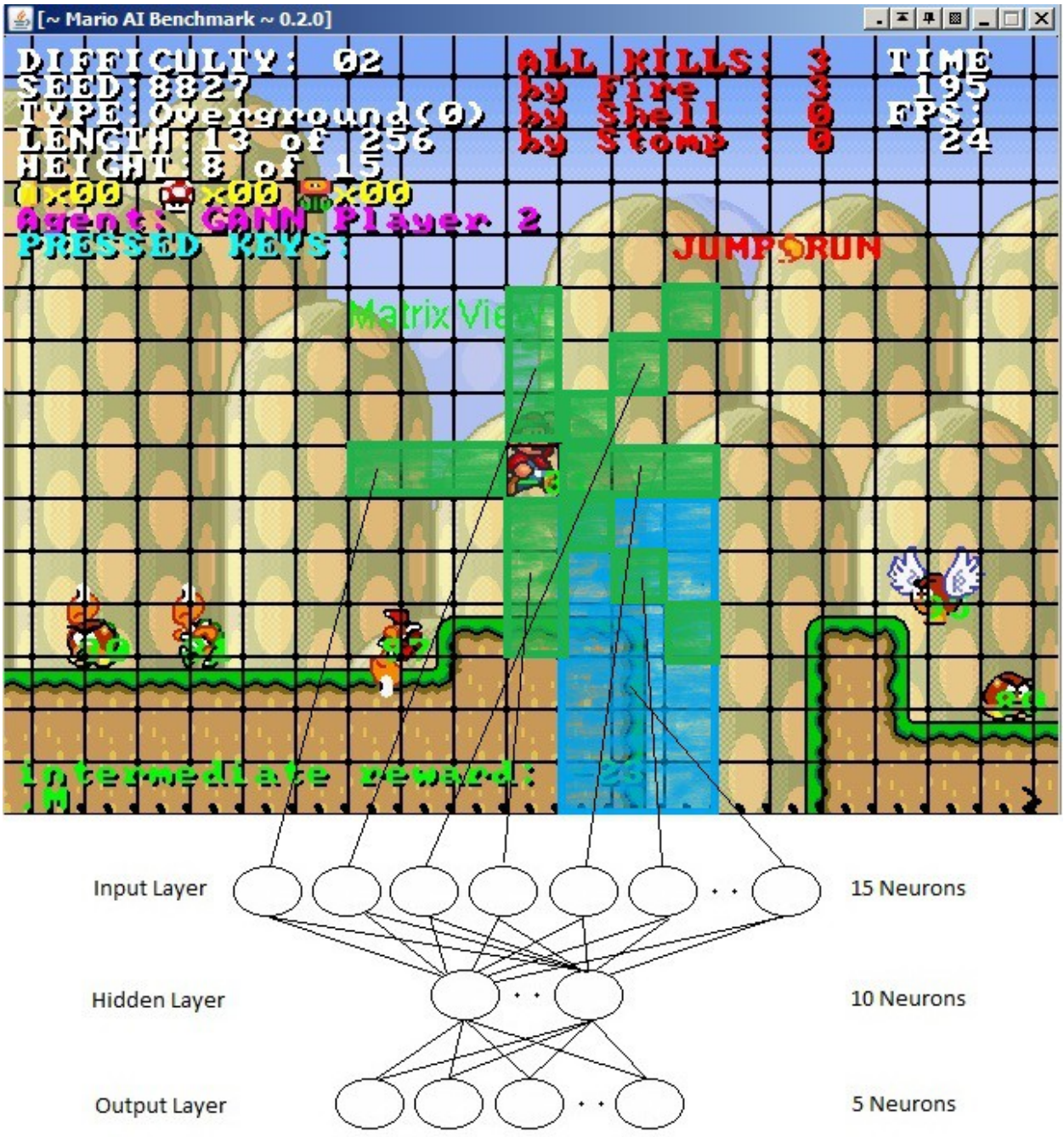
5.1.2 Είσοδοι των νευρωνικών δικτύων

Βάση των μεθόδων που ορίστηκαν παραπάνω καθώς και τις τιμές του πλέγματος γύρω απο τον Mario διαμορφώθηκαν έξι τύποι εισόδων.

```
-------------------------------------------------------
Νευρωνικό Δίκτυο GANN_1
-------------------------------------------------------
   15 Νευρώνες Εισόδου
   enemyInFrontDistance, enemyBehindDistance, enemyAboveDistance,
   enemyBelowDistance, enemyUpRightDistance, enemyDownRightDistance,
```

```
    flowerPotState, obstacleDistance, obstacleHeight,
    ObstacleBelowDistance, obstacleAboveDistance, gapDistance,
    sizeOfGap, canMarioJump, canMarioShoot
-------------------------------------------------------------
```

Με αυτές τις εισόδους ο χειριστής έχει την δυνατότητα να αντιληφθεί τους κοντινότερους εχθρούς (επίγειους καθώς και ιπτάμενους λόγω των `UpRightDistance` και `DownRightDistance` καθώς και τα κοντινά εμπόδια της πίστας. Δεν έχει την δυνατότητα να διαχωρίσει μεταξύ των διαφορετικών εχθρών. Επίσης έχει κάποιες 'νεκρές γωνίες' στο πεδίο.



Εικόνα 5.3 : Είσοδοι Νευρωνικού Δικτύου GANN_1

Το GANN_1 βασίστηκε στην υλοποίηση των Sandberg & Jørgensen [2009]. Τα GANN_2 & _3 επεκτείνουν την βασική υλοποίηση προσθέτοντας κι άλλους νευρώνες εισόδου για την αύξηση του πεδίου οπτικής του πράκτορα.

```
-----------------------------------------------------------
Νευρωνικό Δίκτυο GANN_2
-----------------------------------------------------------
   16 Νευρώνες Εισόδου
   enemyInFrontDistance, enemyBehindDistance, enemyAboveDistance,
   enemyBelowDistance, enemyUpRightDistance, enemyDownRightDistance,
   flowerPotState, obstacleDistance, obstacleHeight,
   ObstacleBelowDistance, obstacleAboveDistance, gapDistance,
   sizeOfGap, canMarioJump, canMarioShoot, enemyBelowKillable
-----------------------------------------------------------
```

Αυτό το νευρωνικό έχει μία παραπάνω είσοδο απο το **GANN_1** την `enemyBelowKillable`. Δοκιμάστηκε έτσι εάν θα μάθει να συμπεριφέρεται διαφορετικά ο πράκτορας έχοντας πληροφορία για το εάν ο εχθρός ακριβώς από κάτω μπορεί να σκοτωθεί εάν προσγειωθεί ο Mario επάνω του.

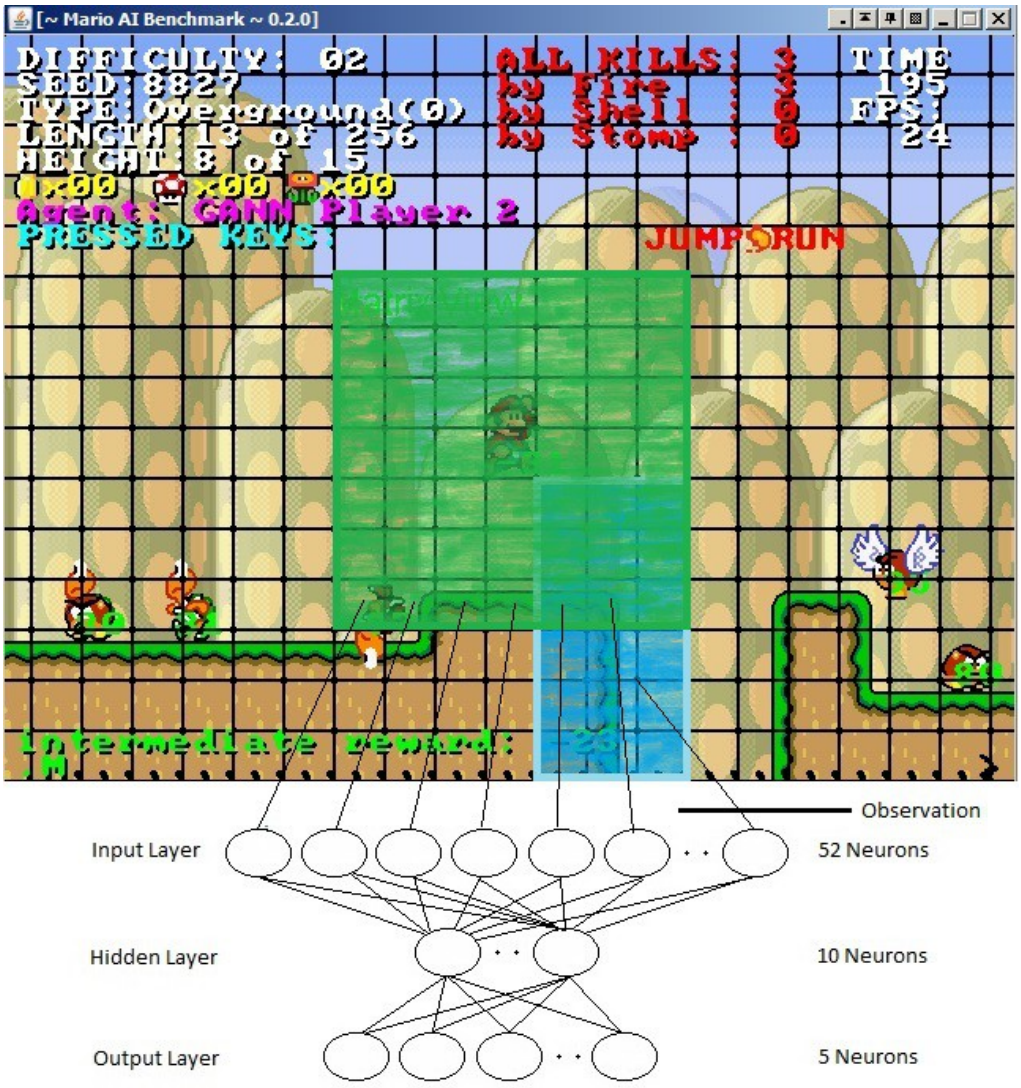
```
-----------------------------------------------------------
Νευρωνικό Δίκτυο GANN_3
-----------------------------------------------------------
   19 Νευρώνες Εισόδου
   enemyInFrontDistance, enemyBehindDistance, enemyAboveDistance,
   enemyBelowDistance, enemyUpRightDistance, enemyDownRightDistance,
   flowerPotState, obstacleDistance, obstacleHeight,
   ObstacleBelowDistance, obstacleAboveDistance, gapDistance,
   sizeOfGap, canMarioJump, canMarioShoot, enemyBelowKillable,
   enemyRightDownKillable, enemyInFrontKillable,
   obstacleAboveBreakable
-----------------------------------------------------------


-----------------------------------------------------------
Νευρωνικό Δίκτυο GANN_4
-----------------------------------------------------------
   20 Νευρώνες Εισόδου
   enemyInFrontDistance_Stomp, enemyInFrontDistance_Nstomp,
   enemyAboveDistance_Stomp, enemyAboveDistance_Nstomp,
   enemyBelowDistance_Stomp, enemyBelowDistance_Nstomp,
   enemyUpRightDistance_Stomp, enemyUpRightDistance_Nstomp,
   enemyRightDownDistance_Stomp, enemyRightDownDistance_Nstomp,
   flowerPotState, obstacleDistance, obstacleHeight,
   obstacleBelowDistance, obstacleAboveDistance, gapDistance,
   sizeOfGap, obstacleAboveBreakable, canMarioJump, canMarioShoot
-----------------------------------------------------------
```

```
-----------------------------------------------------------
Νευρωνικό Δίκτυο GANN_g52
-----------------------------------------------------------
   52 Νευρώνες Εισόδου
      48 Νευρώνες που παίρνουνε τιμές από το πλέγμα γύρω από τον
      Mario
      4 Νευρώνες canMarioJump, canMarioShoot, gapDistance,
      sizeOfGap
-----------------------------------------------------------
```

Οι 48 νευρώνες παίρνουν τιμές που προέρχονται απο το πλέγμα. Για κάθε κελί καλείται η συνάρτηση probe που επιστρέφει την τιμή του κελιού ανάλογα με το τί περιέχει (τύπο εχθρού, τύπο εδάφους κλπ)

```
input[square] = probe(i, j, scene)
```

Οι υπόλοιποι 4 νευρώνες παίρνουν τιμές όπως στους άλλους χειριστές. Δίνουν την πληροφορία στον χειριστή για το εάν μπορεί να εκτελέσει άλμα, εάν μπορεί να πυροβολήσει, εάν υπάρχει κενό και το πόσο μεγάλο είναι.



Εικόνα 5.4 : Είσοδοι Νευρωνικού Δικτύου GANN_g52

Νευρωνικό Δίκτυο GANN_g292

---

292 Νευρώνες Εισόδου

288 Νευρώνες. Κάθε κελί του πλέγματος αναλύεται στις κατηγορίες/είδος τιμής που μπορεί να περιέχει. Αυτές έχουνε ομαλοποιηθεί σε 6 κατηγορίες σύμφωνα με την συμπεριφορά του Mario απέναντι τους. Έτσι κάθε κελί αντιστοιχεί σε 6 νευρώνες.

4 Νευρώνες canMarioJump, canMarioShoot, gapDistance, sizeOfGap

---

Η διεπαφή Environment επιστρέφει για κάθε κελί πληροφορία για το τί περιέχει. Δίνεται η δυνατότητα μέσω της πλατφόρμας MarioAI να διαλέξει κανείς το επίπεδο της ανάλυσης που επιθυμεί σε αυτές τις τιμές. Επιλέχτηκε η τιμή Z1. Σε αυτό το επίπεδο ανάλυσης η τιμές του πλέγματος κατηγοριοποιούνται με βάση το είδος του εχθρού αντί να επιστρέφει διαφορετικές τιμές για όλα τα είδη των εχθρών. Κατηγοριοποιούνται σε αυτού που μπορούν να πατηθούν και σε αυτούς που δεν μπορούν.
Παρακάτω παρουσιάζονται όλες οι τιμές.

- ΕΧΘΡΟΙ
  - 0 κέρμα, λάμψη, Mario,
  - 3 λουλούδι φωτιάς
  - 2 μανιτάρι
  - 25 φωτιά
  - 80 bullet, goomba, winged goomba, green goomba, winged greek goomba shell, wave goomba
  - 93 spiky, enemyflower, winged spiky

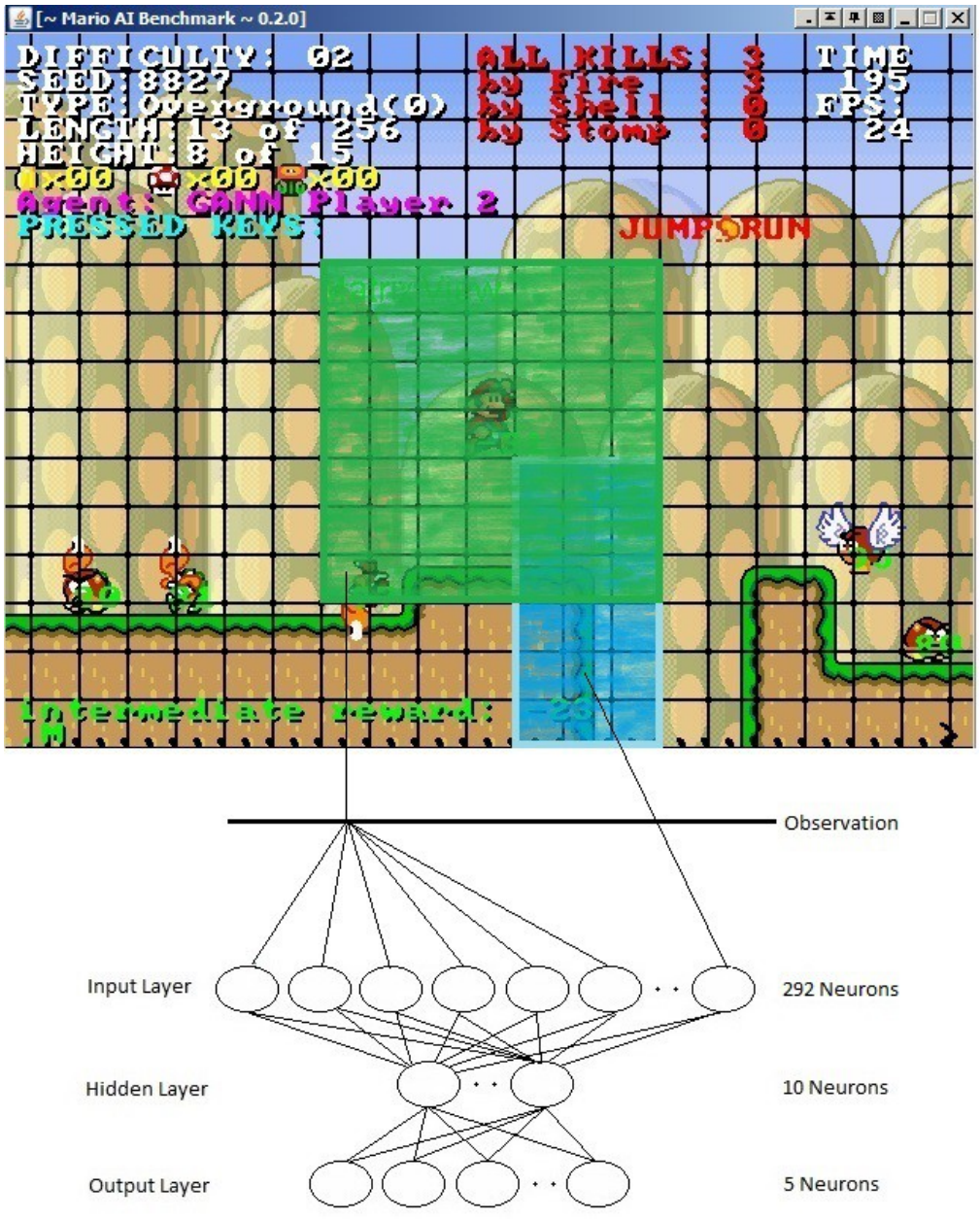
- ΠΕΡΙΒΑΛΛΟΝ / ΕΔΑΦΟΣ
  - 0 κρυμμένο τετράγωνο
  - -60 border cannot pass through
  - -62 border hill
  - -85 flower pot , cannon
  - 61 top of ladder
  - 5 princess.

Αυτές οι τιμές κατηγοριοποιήθηκαν επιπλέον για απλοποίηση του περιβάλλοντος. Ορίστηκε η συνάρτηση `probez` που μετατρέπει την τιμή του κελιού σε έναν δυαδικό αριθμό που αντιπροσωπεύει την κατηγορία του κελιού.

```
switch (scene[realX][realY])
{
//enemies
 case 80:
    return new float [] {1.0f,0f,0f,0f,0f,0f};
 case 93:
    return new float [] {0.0f,1f,0f,0f,0f,0f};
 case 2:
 case 3:
 case 25:
    return new float [] {0.0f,0f,1f,0f,0f,0f};
 case 0:
    return new float [] {0.0f,0f,0f,0f,0f,0f};
//environment
 case -24:
    return new float [] {0.0f,0f,0f,1f,0f,0f};
 case -60:
 case -62:
    return new float [] {0.0f,0f,0f,0f,0f,1f};
```

```
case -85:
   return new float [] {0.0f,0f,0f,0f,1f,0f};
default :
   return new float [] {0.0f,0f,0f,0f,0f,0f};
```

Το GANN_g51 & _292 βασίστηκε στην υλοποίηση των Togelius & Karakovskiy [2009] επεκτείνοντας το πλέγμα οπτικής καθώς και τους κωδικούς περιβάλλοντος που μπορεί να αντιληφθεί ο πράκτορας.



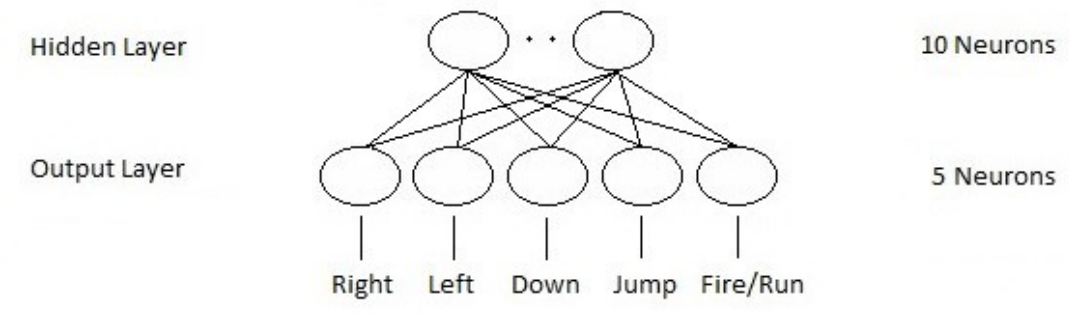
Εικόνα 5.5 : Είσοδοι Νευρωνικού Δικτύου GANN_g292

5.1.3 Έξοδοι των νευρωνικών δικτύων

5.1.3.1 Νευρώνες εξόδου τύπου I

Στην αρχή δοκιμάστηκαν όλα τα διαφορετικά νευρωνικά δίκτυα με τον ίδιους νευρώνες εξόδου. Αυτοί ήταν πέντε, ένας για κάθε κουμπί του παιχνιδιού. Σε κάθε χρονικό βήμα ο χειριστής λάμβανε τις επεξεργασμένες τιμές από το δεκτικό του πεδίο και τις έβαζε σαν εισόδους στο νευρωνικό του δίκτυο. Στην συνέχεια το νευρωνικό δίκτυο βγάζει μία έξοδο για κάθε κουμπί. Εφόσον αυτή η τιμή αυτή είναι πάνω από ένα κατώφλι, ο χειριστής θα πατήσει το αντίστοιχο κουμπί. Η τιμή του κατωφλίου είναι 0.5. Κάθε νευρώνας εξόδου έχει μια σιγμοειδής συνάρτηση ενεργοποίησης έτσι θα παράγει πραγματικές τιμές στο διάστημα [0.0,1.0]. Μια σχηματική αναπαράσταση μπορεί να βρεθεί στην εικόνα 5.3.

```
-----------------------------------------------------------
Νευρώνες Εξόδου Τυπου I
-----------------------------------------------------------
Δεδομένα: νευρωνικό δίκτυο ν1, είσοδοι ι1
Αρχικοποίηση outputs = float[], action = boolean[]
ν1.setInputs(ι1)
ν1.activate()
outputs = ν1.getOutputs()
for (int i = 0; i < action.length-1; i++)
{
   if( outputs > 0.5 )
      action[i] = true;
      else
      action[i] = false;
}
-----------------------------------------------------------
```



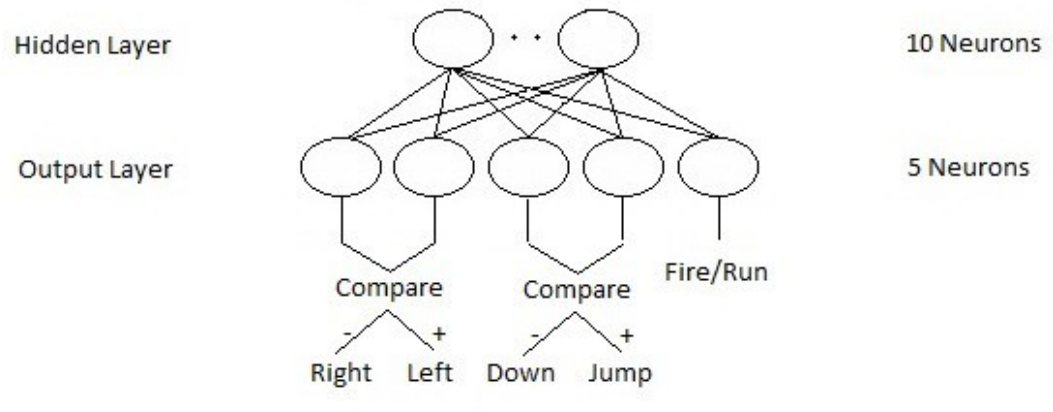
Εικόνα 5.6 : Έξοδοι Νευρωνικού Δικτύου Τύπου I

5.1.3.2 Νευρώνες εξόδου τύπου ΙΙ

Μια άλλη προσέγγιση που δοκιμάστηκε ήταν να υπάρχουν 5 νευρώνες εξόδου όπως προηγουμένως αλλά η απόφαση για το αν θα εκτελέσει ο πράκτορας κάποια κίνηση βασίζεται στην διαφορά των νευρώνων που αντιστοιχούν στην αντίστοιχη κατεύθυνση.

Έστω ότι ο νευρώνας εξόδου 1 αντιστοιχεί στο κουμπί αριστερά, και ο νευρώνας εξόδου 2 αντιστοιχεί στο κουμπί δεξιά. Υπολογίζεται η διαφορά των εξόδων των δύο νευρώνων. Εάν η διαφορά είναι αρνητική ο πράκτορας θα πατήσει το κουμπί δεξιά, εάν είναι θετική θα πατήσει το κουμπί αριστερά. Ομοίως για το κουμπιά άλμα και κάτω. Η απόφαση για το εάν θα πυροβολήσει/τρέξει υπολογίζεται πάλι βάση ενός κατωφλιού (0.5). Εάν η έξοδος του νευρώνα υπερβαίνει το κατώφλι θα εκτελεστεί η κίνηση.

```
-----------------------------------------------------------
Νευρώνες Εξόδου Τυπου ΙΙ
-----------------------------------------------------------
Δεδομένα: νευρωνικό δίκτυο ν1, είσοδοι ι1
Αρχικοποίηση outputs = float[], action = boolean[]
ν1.setInputs(ι1)
ν1.activate()
outputs = ν1.getOutputs()
if ((outputs[0]-outputs[1]) > 0)
   action[0]=true;
else
      action[1]=true;
if ((outputs[2]-outputs[3]) > 0)
      action[2]=true;
else
      action[3]=true;
if (outputs[4] > 0.5)
      action[4]=true;
   -----------------------------------------------------------
```



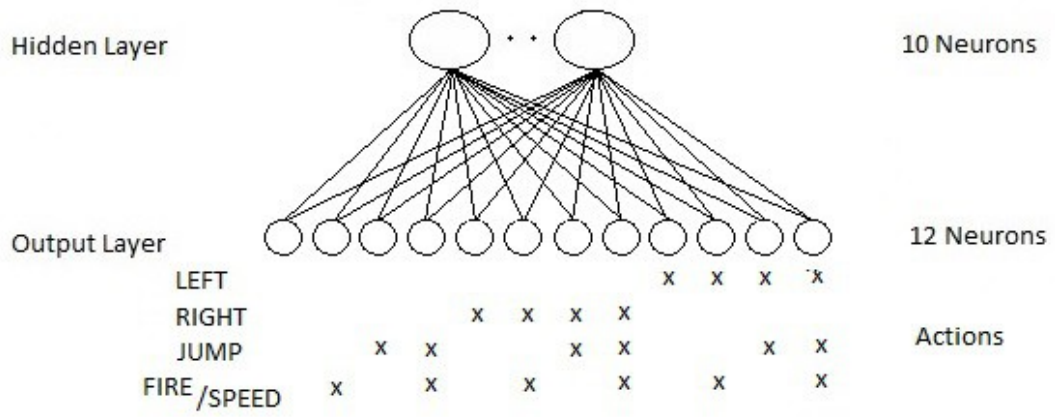
Εικόνα 5.7 : Έξοδοι Νευρωνικού Δικτύου Τύπου ΙΙ

5.1.3.3 Νευρώνες εξόδου τύπου III

Ο χώρος των δράσεων παρότι είναι διακριτός είναι και αυτός μεγάλος. Στο παιχνίδι Mario της Nintendo, ο παίχτης χειρίζεται τον Mario με ένα χειριστήριο που έχει κουμπιά κατεύθυνσης (δεξιά , αριστερά, πάνω, κάτω) και δύο κουμπιά δράσης (A,B). Το κουμπί A ξεκινάει ένα άλμα το οποίο είναι ανάλογο του χρόνου που κρατήθηκε πατημένο το κουμπί. Το κουμπί B κάνει τον Mario να τρέξει. Επιπλέον εάν ο Mario είναι σε κατάσταση Φωτιάς, με το κουμπί B πυροβολάει.

Αγνοώντας την κατεύθυνση πάνω που δεν θα χρησιμοποιήσουμε, η πληροφορίες που θα πρέπει να δώσει σε κάθε χρονικό βήμα ο χειριστής είναι πέντε bits, δλδ 2^5 πιθανοί συνδυασμοί δράσεων. Πολλοί από αυτούς τους συνδυασμούς είναι μη έγκυροι και δεν έχουν κάποιο νόημα, όπως π.χ. να πατηθεί το δεξιά με το αριστερά. Έτσι για να διευκολυνθεί η μάθηση του πράκτορα ορίστηκαν 12 έγκυροι συνδυασμοί πλήκτρων οι οποίοι είναι αρκετοί για να τερματίσει ο πράκτορας τα πρώτα επίπεδα δυσκολίας . Κάθε ένας από αυτούς αντιπροσωπεύεται από έναν νευρώνα εξόδου. Έτσι επιλέγεται ο νευρώνας με την μέγιστη έξοδο και εκτελούνται οι κινήσεις τις οποίες αντιπροσωπεύει.

```
------------------------------------------------------------
Νευρώνες Εξόδου Τυπου ΙΙ
------------------------------------------------------------
Δεδομένα: νευρωνικό δίκτυο ν1, είσοδοι ι1
Αρχικοποίηση outputs = float[], action = boolean[]
ν1.setInputs(ι1)
ν1.activate()
outputs = ν1.getOutputs()
action[MAX(outputs)] = true

------------------------------------------------------------
```



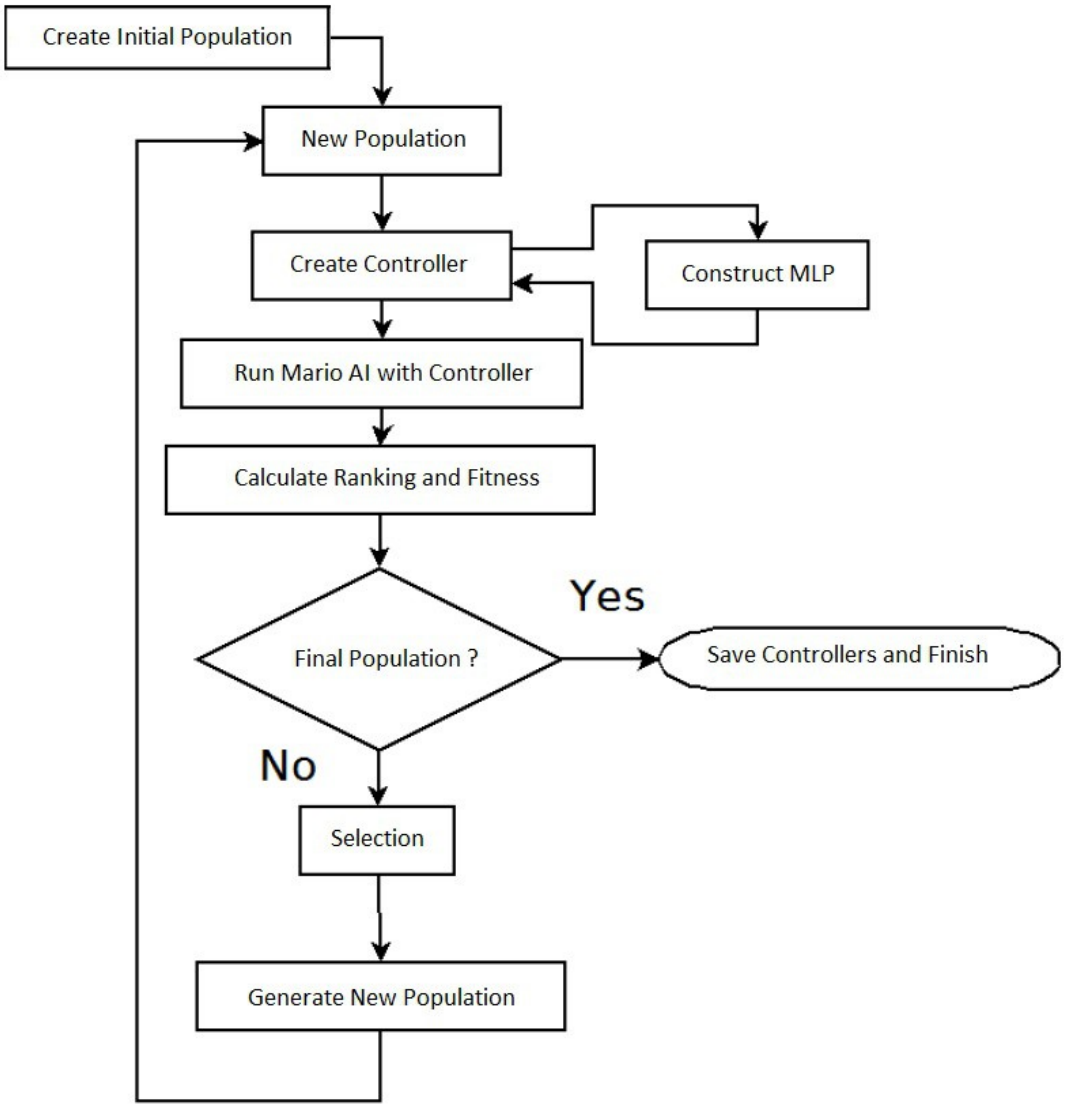
Εικόνα 5.8 : Έξοδοι Νευρωνικού Δικτύου Τύπου III

5.2 Εξελίσσοντας χειριστές με γενετικούς αλγορίθμους

Για να εφαρμοστεί ο Γενετικός Αλγόριθμος στην εξέλιξη των χειριστών θα πρέπει να γίνει μια κατάλληλη κωδικοποίηση του προβλήματος έτσι ώστε να γίνει η αντιστοιχία γονιδίων και χρωματοσωμάτων. Χρησιμοποιήθηκε η "άμεση κωδικοποίηση" (direct encoding [Whitley, 1992]). Σε αυτήν οι παράμετροι του νευρωνικού δικτύου όπως βάρη, συνδέσεις κλπ. αντιστοιχίζονται κατευθείαν από το γονιδίωμα, σε αντίθεση με την "έμμεση κωδικοποίηση" (indirect encoding) στην οποία κωδικοποιούνται κανόνες που περιέχουν πληροφορίες για το πώς θα κατασκευαστεί το δίκτυο.

Κάθε χειριστής χαρακτηρίζεται από ένα Τεχνητό Νευρωνικό Δίκτυο. Τα βάρη των Νευρωνικών Δικτύων θα αντιστοιχιστούν στα γονίδια του χρωματοσώματος έτσι κάθε χρωμόσωμα θα αντιπροσωπεύει ένα χειριστή. Ο Γενετικός Αλγόριθμος χρησιμοποιείται για να εκπαιδεύσει το Νευρωνικό Δίκτυο, βελτιστοποιώντας τις τιμές των γονιδίων των χρωμοσωμάτων, μεγιστοποιώντας έτσι την ορισμένη συνάρτηση καταλληλότητας που έχουμε θέσει για τον χειριστή. Η τοπολογία του δικτύου παραμένει σταθερή.

Για κάθε τύπο Νευρωνικού Δικτύου δημιουργήθηκε ένας εκπαιδευτής που αναλαμβάνει την εξέλιξη των χειριστών μέσω του γενετικού αλγορίθμου και ελέγχει όλες της παραμέτρους εκπαίδευσης.
Παρακάτω παρουσιάζεται το διάγραμμα ροής της δημιουργίας των χειριστών.



Εικόνα 5.9 : Διάγραμμα ροής δημιουργίας χειριστών

Ο Εκπαιδευτής ξεκινάει φτιάχνοντας έναν αρχικό πληθυσμό από άτομα με τυχαίες Γκαουσιανές τιμές που ακολουθούν κανονική κατανομή (μέσο όρο 0 και τυπική απόκλιση 1) στα γονίδια των χρωμοσωμάτων. Έτσι κάθε άτομο/χρωμόσωμα έχει ένα γονιδίωμα από κανονικές τιμές στο διάστημα [-1,1] που θα χρησιμοποιηθούν για βάρη στα ΤΝΔ.

Το κάθε φαινότυπο θα δημιουργηθεί χρησιμοποιώντας τις τιμές του αντίστοιχου γονότυπου σαν βάρη στο νευρωνικό του δίκτυο, έτσι υπάρχει ένα προς ένα αντιστοιχία μεταξύ των φαινοτύπων και γονότυπων. Το κάθε φαινότυπο χρησιμοποιείται ως χειριστής λοιπόν για το Mario AI παίζοντας πίστες και συλλέγοντας στατιστικά για την επίδοσή του. Βάση αυτών βγαίνουν τιμές για τους αντικειμενικούς στόχους του κάθε χειριστή. Στην συνέχεια του ανατίθεται μία τιμή σύμφωνα με την συνάρτηση καταλληλότητας που καθιστά την απόδοση του

χειριστή. Χρησιμοποιώντας την απόδοση του κάθε χειριστή/χρωμοσώματος γίνεται η επιλογή των ατόμων για τον νέο πληθυσμό, διασταυρώνοντας και μεταλλάσσοντας τα χρωμοσώματα. Τέλος μόλις φτάσει στην τελευταία γενιά του πληθυσμού, ο εκπαιδευτής αποθηκεύει όλο τον πληθυσμό σε ένα αρχείο. Το πρώτο άτομο του πληθυσμού θα είναι και ο βελτιστοποιημένος χειριστής που θα χρησιμοποιήσουμε αργότερα για να παίξει το παιχνίδι ή σαν βάση για να εξελίξουμε κάποιον πιο σύνθετο χειριστή.

Παρακάτω περιγράφονται οι παράμετροι του γενετικού αλγορίθμου και η μεθοδολογία εξέλιξης.

5.2.1 Πληθυσμός

Ένα απο τα πρώτα πράγματα που έπρεπε να οριστούν είναι το μέγεθος του πληθυσμού. Δοκιμάστηκαν διάφορα μεγέθη 50,67,100,150 άτομα. Επιλέχτηκε ο πληθυσμός να αποτελείται από 67 άτομα. Βρέθηκε ότι δεν διέφερε πολύ από τα 100, και επιλέχτηκε για να είναι ευέλικτο το σύστημα να εκτελεί πολλές προσομοιώσεις ενώ παράλληλα να έχει καλή εξερευνητική ικανότητα.

5.2.2 Επιλογή

Κάθε πράκτορας στον πληθυσμό αξιολογείται αφού παίξει κάποια επίπεδα του Mario AI (στο 6.2.5 περιγράφεται η προσέγγιση εκπαίδευσης) και του ανατίθενται μία τιμή με βάση την συνάρτηση καταλληλότητας στηριζόμενη στην επίδοση του στα επίπεδα. Αφού αξιολογηθούν όλοι οι πράκτορες του πληθυσμού, ταξινομούνται με βάση την απόδοση τους.

Η πλατφόρμα MarioAI δίνει την κλάση `SystemOfValues`

```
public class SystemOfValues
{
public int distance = 1;
public int win = 1024;
public int mode = 32;
public int coins = 16;
public int flowerFire = 64;
public int kills = 42;
public int killedByFire = 4;
public int killedByShell = 17;
public int killedByStomp = 12;
public int mushroom = 58;
public int timeLeft = 8;
public int hiddenBlock = 24;
public int greenMushroom = 58;
// For Intermediate rewards only
public int stomp = 10;
}
```

Επεκτείνοντας αυτήν μπορούμε να αναθέσουμε τιμές βαρύτητας στα επιτεύγματα του χειριστή σε κάθε επίπεδο που παίζει.
Μετά απο κάθε παίξιμο ενός επιπέδου μπορούμε να μετρήσουμε την απόδοση του χειριστή με βάση τις παραμέτρους που μας ενδιαφέρουν. Οι στατιστικές πληροφορίες αποθηκεύονται σε ένα αντικείμενο που μπορούμε να ανακτήσουμε με την μέθοδο `evaluator.getEvaluationInfo()`. Στην συνέχεια χρησιμοποιώντας την μέθοδο `computeWeightedFitness(SystemOfValues sov)` επιστρέφεται η τιμή της βαθμολογίας του χειριστή.

5.2.2.1 Συνάρτηση καταλληλότητας

Μία από τις σημαντικότερες παραμέτρους για έναν γενετικό αλγόριθμο είναι η συνάρτηση καταλληλότητας. Βασίζεται σε πληροφορίες της απόδοσης του χειριστή στο επίπεδο που έπαιξε και χρησιμοποιείται για να αξιολογηθούν οι διαφορετικοί νευρωνικοί χειριστές (δίκτυα).

Τα βασικά στοιχεία της συνάρτησης είναι η απόσταση που διανύθηκε (μέση τιμή επιπέδου 4096) , ο υπολειπόμενος χρόνος (ο χρόνος που έχει στην διάθεσή του ο mario είναι 200 δευτερόλεπτα στην αρχή της πίστας και μειώνεται), πόσοι εχθροί σκοτώθηκαν (διαφορετική τρόποι εξόντωσης των εχθρών επιφέρουν διαφορετική αμοιβή), η κατάσταση του mario (μικρός, μεγάλος, φωτιά), ο αριθμός μανιταριών που πήρε, ο αριθμός λουλουδιών φωτιάς που πήρε. Επιπλέον συμβάλει αρνητικά το εάν έπεσε ο mario σε κάποιο κενό, αφαιρώντας συνολικά βαθμούς που ισοδυναμούν με 3 τετράγωνα στην απόσταση που διένυσε.

Συνάρτηση Καταλληλότητας

Δεδομένα: αξιολόγηση v_i

Αρχικοποίηση *fitness* = 0

```
fitness +=  DistanceTravelled
fitness +=  win*1000
fitness +=  Mode*250
fitness +=  flowerFire*500
fitness +=  mushroom*500
fitness +=  kills*50
fitness +=  killedByFire*100
fitness +=  killedByShell*170
fitness +=  killedNyStomp*75
fitness +=  timeLeft
fitness -=  fallenInGap*48
```

Έξοδοι: *fitness*

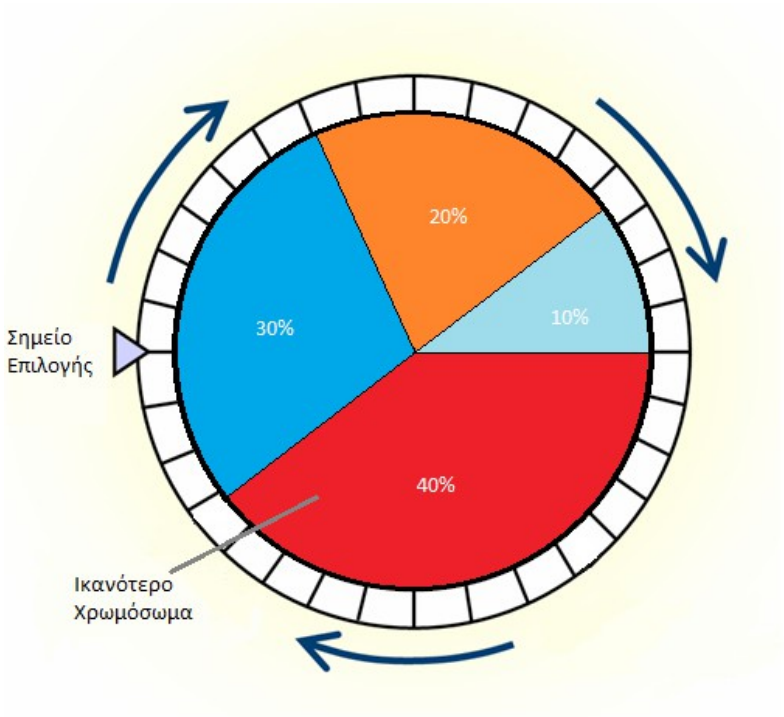
Δοκιμάστηκαν πολλές διαφορετικές παραλλαγές στην βαρύτητα των ανωτέρων τιμών τις συνάρτησης. Στην αρχή η συνάρτηση καταλληλότητας βασιζόταν μόνο στην απόσταση που

διένυσε ο πράκτορας. Αυτό είχε ως αποτέλεσμα την μικρή ποικιλομορφία ανάμεσα στους χειριστές που εξελισσόντουσαν. Ένα άλλο πρόβλημα ήταν ότι η συνάρτηση καταλληλότητας κάποιον πρακτόρων που είχαν πέσει μέσα σε κενό ήταν μεγαλύτερη από αυτών που είχαν κολλήσει κάπου πριν το κενό. Επίσης οι πράκτορες που τερμάτιζαν με ολόκληρη ζωή (μεγάλος mario, ή mario με φωτιά) είχαν τον ίδιο βαθμό με αυτούς που τερμάτιζαν με την χαμηλότερη ζωή (μικρός mario).

Στην προσπάθεια να ενσωματώσουν οι πράκτορες στοιχεία τακτικής συμπεριφοράς προστέθηκαν κι άλλες μεταβλητές στην συνάρτηση. Το ποσοστό "ζωής" του mario (μικρος, μεγάλος, με φωτιά) για να μπορέσει ο ΓΑ να ξεχωρίσει αυτούς που καταφέρανε να τερματίσουν με περισσότερη ζωή. Ο χρόνος που απομένει στην πίστα έτσι ώστε να ξεχωρίσουν οι πράκτορες που είχαν κολλήσει σε κάποιο σημείο. Ο αριθμός των εχθρών που σκοτώθηκαν, καθώς και διαφορετική αμοιβή ανάλογα με τον τρόπο που σκοτώθηκαν. Και τέλος έξτρα αμοιβή για το εάν τερμάτισε ή όχι την πίστα.

5.2.3 Διασταύρωση

Σε κάθε γενιά του πληθυσμού επιλέγετε ένας μέρος του για να διασταυρωθεί και να παράξει άτομα της επόμενης γενιάς. Μετά την ανάθεση της τιμής καταλληλότητας του κάθε ατόμου, ο πληθυσμός ταξινομείται σύμφωνα με την αντικειμενική αξία. Επιλέχτηκε η *μέθοδος σταθμισμένης ρουλέτας* (βλέπε Κεφάλαιο 3) για την επιλογή των γονέων.

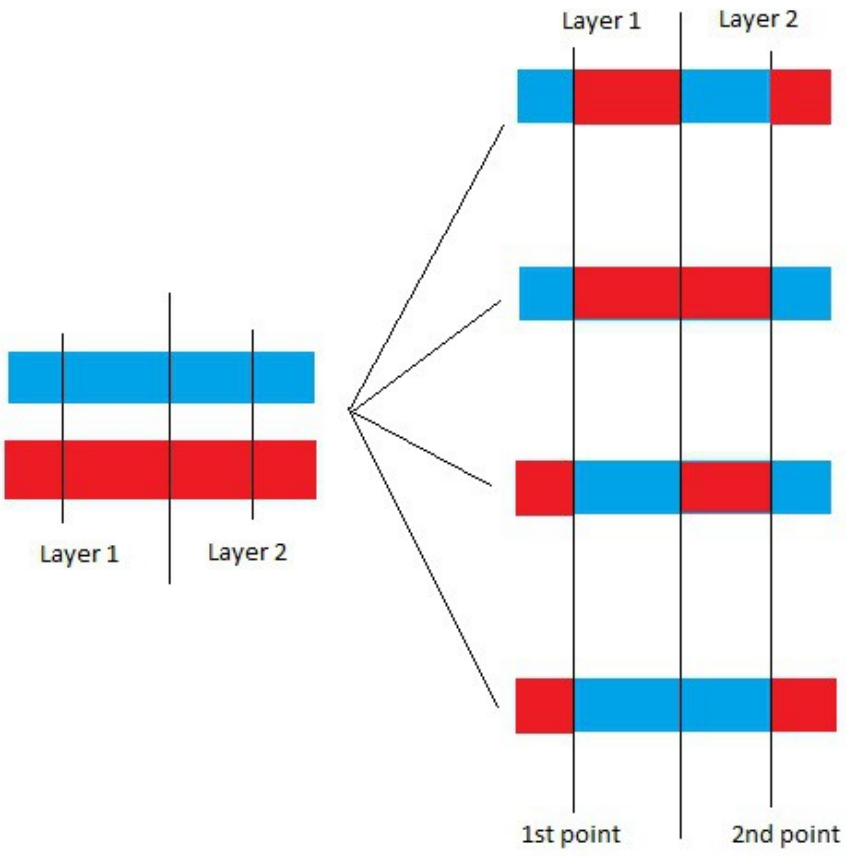


Εικόνα 5.10 : Διάγραμμα μεθόδου σταθμισμένης ρουλέτας

Έτσι η ικανότητα κάθε ατόμου εξαρτάται μόνο από την θέση του στην κατάταξη και όχι από την απόλυτη τιμή της συνάρτησης καταλληλότητας του. Κάθε χρωμόσωμα τοποθετείται σε ένα κομμάτι της ρουλέτας, του οποίου το μέγεθος είναι ανάλογο της θέσης του χρωμοσώματος στην κατάταξη. Στη συνέχεια η ρουλέτα γυρνάει δύο φορές και επιλέγονται δύο γονείς που διασταυρώνονται για να παραχθεί ένα νέο άτομο για τον καινούργιο πληθυσμό. Η διαδικασία αυτή επαναλαμβάνεται μέχρις ότου να συμπληρωθεί ο αριθμός των ατόμων που ορίζει έμμεσα η μεταβλητή `crossoverrate`.

Λόγω της φύσης της κωδικοποίησης των χρωμοσωμάτων αποφασίστηκε να εφαρμοστεί *γενική διασταύρωση δύο σημείων* καθώς και να περιοριστεί η εναλλαγή της πληροφορίας ανα επίπεδο νευρώνων. Έτσι επιλέγεται ένα τυχαίο σημείο στα βάρη μεταξύ των νευρώνων δύο επιπέδων και διασταυρώνονται μόνο τα βάρη μεταξύ εκείνων των επιπέδων.

Παρακάτω φαίνονται αναλυτικά η πιθανοί τρόποι διασταύρωσης δύο ατόμων.



Εικόνα 5.11 : Τρόποι Διασταύρωσης χρωμοσωμάτων

5.2.4 Μετάλλαξη

Αφού δημιουργηθούν τα νέα άτομα μέσω διασταύρωσης, επιλέγεται ένας τυχαίος αριθμός από αυτά για να μεταλλαχθούν. Σε κάθε χρωμόσωμα που επιλέγεται υπάρχει μία σταθερή πιθανότητα μετάλλαξης κάθε γονιδίου του. Η μετάλλαξη συμβαίνει προσθέτοντας στην τιμή του γονιδίου μία τυχαία μεταβλητή που ακολουθεί γκαουσιανή κατανομή με μέση τιμή 0 και τυπική απόκλιση 0.09.

Αργότερα στα πειράματα εφαρμόστηκε μεταβλητή πιθανότητα μετάλλαξης κάθε γονιδίου. Αυτή ήταν ανάλογη του αριθμού των γενεών, με αρχική τιμή 0.7 και τελική 0.09.

5.2.5 Διαδικασία εξέλιξης

Παρακάτω παρουσιάζεται πιο αναλυτικά η εικόνα 6.9 και περιγράφεται η λειτουργία του γενετικού αλγορίθμου.

1. Δημιουργείται ένας αριθμός από χρωμοσώματα ίσος με το μέγεθος του πληθυσμού.

2. Εάν υπάρχει αρχείο με αποθηκευμένα βάρη, φορτώνεται. Διαφορετικά δημιουργούνται χρωμοσώματα με τυχαία βάρη.
3. Αποθηκεύονται τα χρωμοσώματα στον πληθυσμό.
4. Όλα τα βάρη των χρωματοσωμάτων αντιγράφονται σε αντίστοιχα νευρωνικά δίκτυα.
5. Κάθε νευρωνικό δίκτυο αντιστοιχίζεται σε έναν πράκτορα.
6. Κάθε πράκτορας παίζει έναν προκαθορισμένο αριθμό φορών επίπεδα του παιχνιδιού και ο γενετικός αλγόριθμος θα αξιολογεί την απόδοση του σύμφωνα με την συνάρτηση καταλληλότητας.
7. Ταξινομείται ο πληθυσμός σύμφωνα με την απόδοση του κάθε χειριστή.
8. Δημιουργείται η επόμενη γενιά
   i. Οι καλύτεροι 9 πράκτορες (13%) αποθηκεύονται. Η διαδικασία της διασταύρωσης και της μετάλλαξης μπορεί να καταστρέψει ικανά χρωμοσώματα. Έτσι επιλέχθηκε να επιβιώνουν οι καλύτεροι κάθε γενιάς και στον επόμενο πληθυσμό. Η τεχνική αυτή ονομάζεται *ελιτισμός*.
   ii. Από όλο τον πληθυσμό, με την μέθοδο της σταθμισμένης ρουλέτας παράγονται 57 καινούργια άτομα (85% του πληθυσμού).
   iii. Από τα καινούργια άτομα επιλέγεται τυχαία ένα ποσοστό μεταξύ 0% - 30% για να μεταλλαχθεί.
   iv. Αποθηκεύονται τα καινούργια άτομα που πλέον αποτελούν τον καινούργιο πληθυσμό.
9. Επανάληψη της διαδικασίας από το βήμα 4 για προκαθορισμένο αριθμό γενεών.

5.3 Περιορισμοί

Υπάρχουν κάποιοι περιορισμοί δεδομένων των εισόδων των νευρωνικών δικτύων. Συγκεκριμένα για τα δίκτυα GANN_X, παρά τις εισόδους που αναφέρονται στο διαγώνιο οπτικό πεδίο, βλέπουμε ότι ο πράκτορας έχει κενά στην αντίληψή του για το περιβάλλον. Έτσι μπορεί να μην δει κάποιους ιπτάμενους εχθρούς, ή να μην αντιληφθεί αρκετά νωρίς κάποιον εχθρό που βρίσκεται διαγώνια πίσω του.

Λόγο της περιορισμένης δυνατότητας του πράκτορα να αντιληφθεί το περιβάλλον αναλυτικά, αποφασίστηκε να εκπαιδευτεί μέχρι επίπεδα δυσκολίας 2.

ΚΕΦΑΛΑΙΟ 6

6 Πειραματική διαδικασία και αποτελέσματα

6.1 Ελέγχοντας τον Mario με τεχνητά νευρωνικά δίκτυα

6.1.1 Πείραμα 1 : Δοκιμή σε επίπεδο χωρίς εχθρούς και εμπόδια

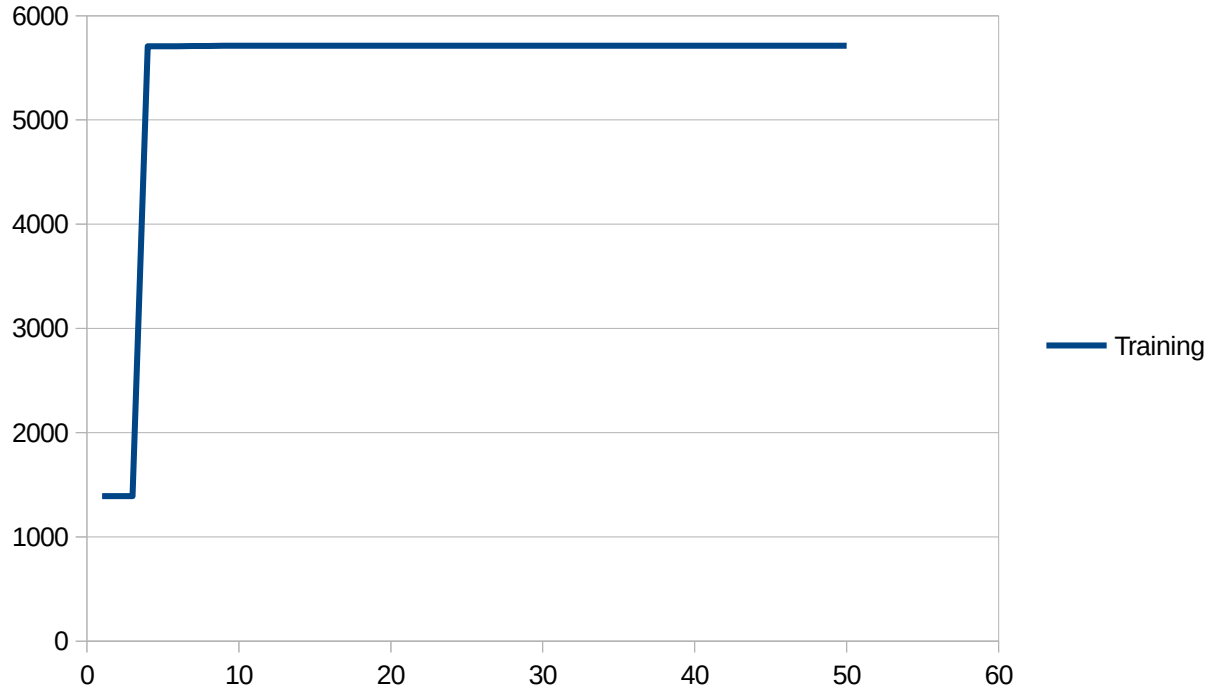
Σε αυτό το πείραμα δοκιμάστηκε το TNN 1 με τους νευρώνες εξόδου τύπου I (ένας νευρώνας για κάθε διαθέσιμο κουμπί) σε μία άδεια πίστα, χωρίς κενά, εμπόδια ή εχθρούς.

Η εκπαίδευση έγινε σε 1 συγκεκριμένο επίπεδο (seed 8827) για 50 γενεές. Παρατηρούμε ότι πολύ γρήγορα (είδη από την 4η γενεά) υπάρχουν πράκτορες που τερματίζουν την πίστα.

Generations	Training
1	1391
1	1391
2	1391
3	1391
4	5707
5	5707
6	5707
7	5709
8	5709
9	5713
10	5713
11	5713
12	5713
...	...
50	5713

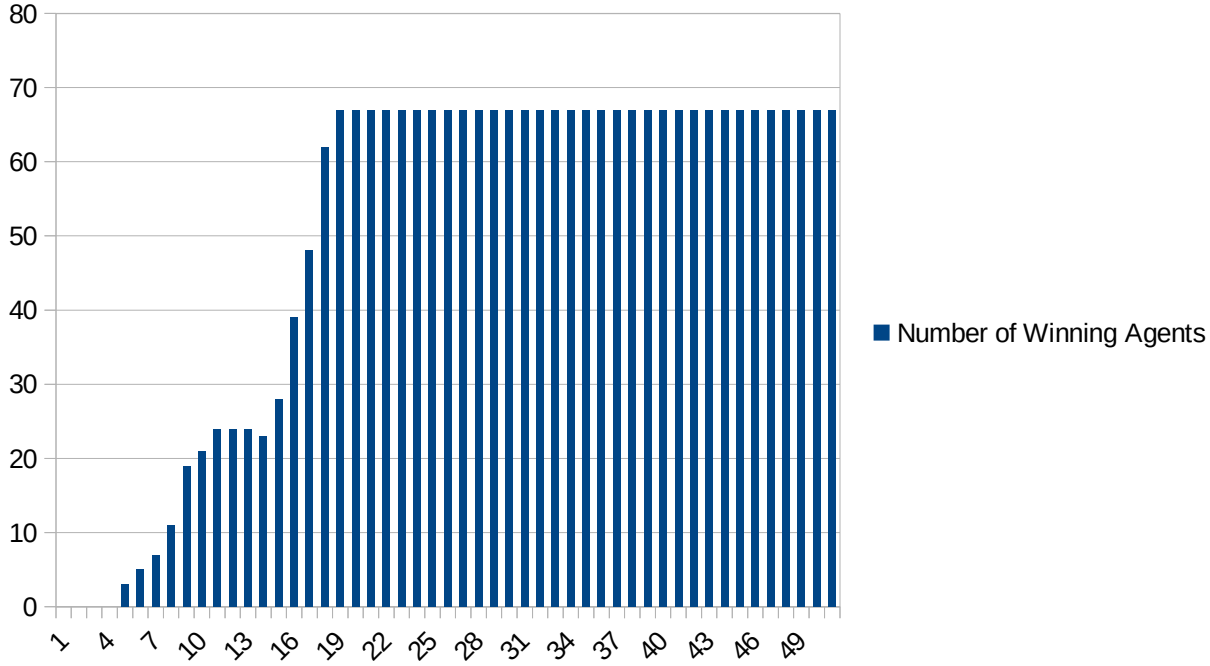
Πίνακας 6.1 : Τιμές Συνάρτησης Καταλληλότητας

Από την 9η γενιά ο αλγόριθμος έχει φτάσει σε ένα μέγιστο.



Εικόνα 6.1 : Καμπύλη Συνάρτησης Καταλληλότητας

Ξεκινώντας από την 4η γενιά , υπάρχουν πράκτορες που τερματίζουν την πίστα. Ολοένα ο αριθμός αυτόν μεγαλώνει καθώς ο αλγόριθμος συγκλίνει στο μέγιστο. Ακολουθώντας την διαδικασία της διασταύρωσης και τις μετάλλαξης βλέπουμε ότι εν τέλει όλα τα άτομα του πληθυσμού είναι ικανά να τερματίσουν το επίπεδο με την μέγιστη βαθμολογία.



Εικόνα 6.2 : Αριθμός πρακτόρων που τερμάτισαν το επίπεδο ανα γενεά

Παρακάτω φαίνεται ένα στιγμιότυπο από το παιχνίδι.

Εικόνα 6.3 : Mario παίζοντας ένα κενό επίπεδο

Βλέπουμε ότι ο πράκτορας έμαθε ότι πρέπει να κινείται προς τα δεξιά. Η τιμή της συνάρτηση καταλληλότητας αυξάνεται ανάλογα με το πόσος χρόνος έχει απομείνει στον mario στο τέλος της πίστας. Συναρτήσει αυτού ο πράκτορας έχει καταφέρει να μεγιστοποιήσει την συνάρτηση τρέχοντας και πηδώντας προς τα δεξιά.

Για το ίδιο επίπεδο θα χρησιμοποιήσουμε για σύγκριση δύο πράκτορες που περιέχονταν μαζί με την πλατφόρμα του Mario AI (Karakovskiy & Togelius , 2009)

- RandomAgent : Ένας πράκτορας που κάνει τυχαίες κινήσεις με μία τάση να κινείται δεξιά και να κάνει μακρυά άλματα.
- ForwardJumpingAgent : Ένας πράκτορας που τρέχει προς τα δεξιά και πηδάει εάν οι μπορεί.

GANN_1	Random Agent	Forward Jumping Agent
Evaluation lasted : 15661 ms	Evaluation lasted : 83843 ms	Evaluation lasted : 16112 ms
Weighted Fitness : 6576	Weighted Fitness : 5688	Weighted Fitness : 6568
Mario Status : WIN!	Mario Status : WIN!	Mario Status : WIN!
Time Spent(marioseconds) : 25	Time Spent(marioseconds) : 136	Time Spent(marioseconds) : 26
Time Left(marioseconds) : 174	Time Left(marioseconds) : 63	Time Left(marioseconds) : 173

Πίνακας 6.2 : Βαθμολογία Πρακτόρων στο κενό επίπεδο

Ο πράκτορας που εξελίχθηκε με τον γενετικό αλγόριθμο τερμάτισε γρηγορότερα 111 δευτερόλεπτα απο τον RandomAgent και 1 δευτερόλεπτο απο τον ForwardJumpingAgent.

6.1.2 Πείραμα 2 : Επιλογή καθεστώτος εκπαίδευσης

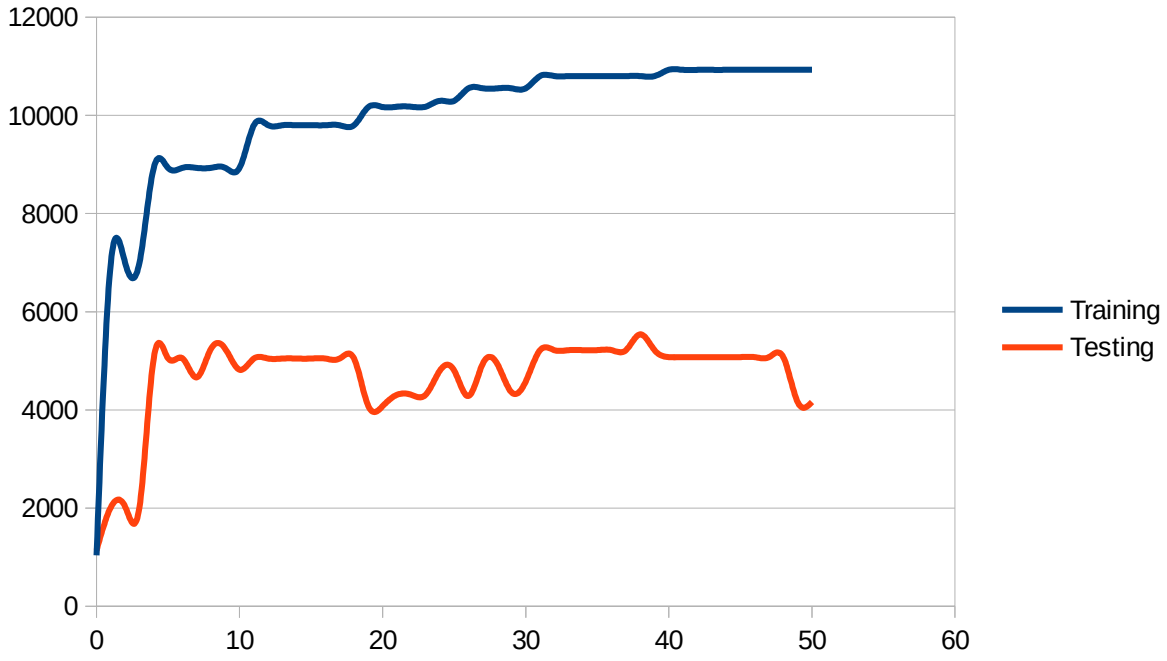
Είδαμε ότι ο πράκτορας που εκπαιδεύεται πολύ γρήγορα μαθαίνει να κινείται προς τα δεξιά σε μία κενή πίστα. Σε αυτό το πείραμα προσπαθήσαμε να επιλέξουμε το βέλτιστο τρόπο εκπαίδευσης του πράκτορα, όσον αναφορά τις πίστες στην οποίες εκπαιδεύεται.

Η μελέτη έγινε σε επίπεδα δυσκολίας 0.(Πίστες χωρίς κενά, με εμπόδια και εχθρούς που πατιούνται, καθώς και λουλούδια που πετάνε φωτιές)
Χρησιμοποιήθηκε το GANN_1 με νευρώνες εξόδου τύπου I.
Για να αποφασιστεί ποιος θα ήταν ο καλύτερος τρόπος εκπαίδευσης για να αυξηθεί η ικανότητα γενίκευσης του πράκτορα, μεταβλήθηκε ο αριθμός επιπέδων που έπαιζε ο κάθε πράκτορας (χρωμόσωμα) σε κάθε γενιά. Η τιμή της καταλληλότητας του κάθε χρωμοσώματος του πληθυσμού προέκυπτε από τον μέσο όρο της τιμής στα μεμονωμένα επίπεδα.

Στα διαγράμματα υπάρχουν δύο καμπύλες. Η μία προκύπτει απο την μέγιστη τιμή τις συνάρτησης καταλληλότητας ανά γενιά, και αφορά το επίπεδο εκπαίδευσης. Η δεύτερη είναι η απόδοση του καλύτερου πράκτορα της γενειάς σε μία σειρά από 10 τυχαία επίπεδα, ίδιου επιπέδου δυσκολίας.

Παρακάτω παρουσιάζονται οι διαφορετικές μέθοδοι εκπαίδευσης :

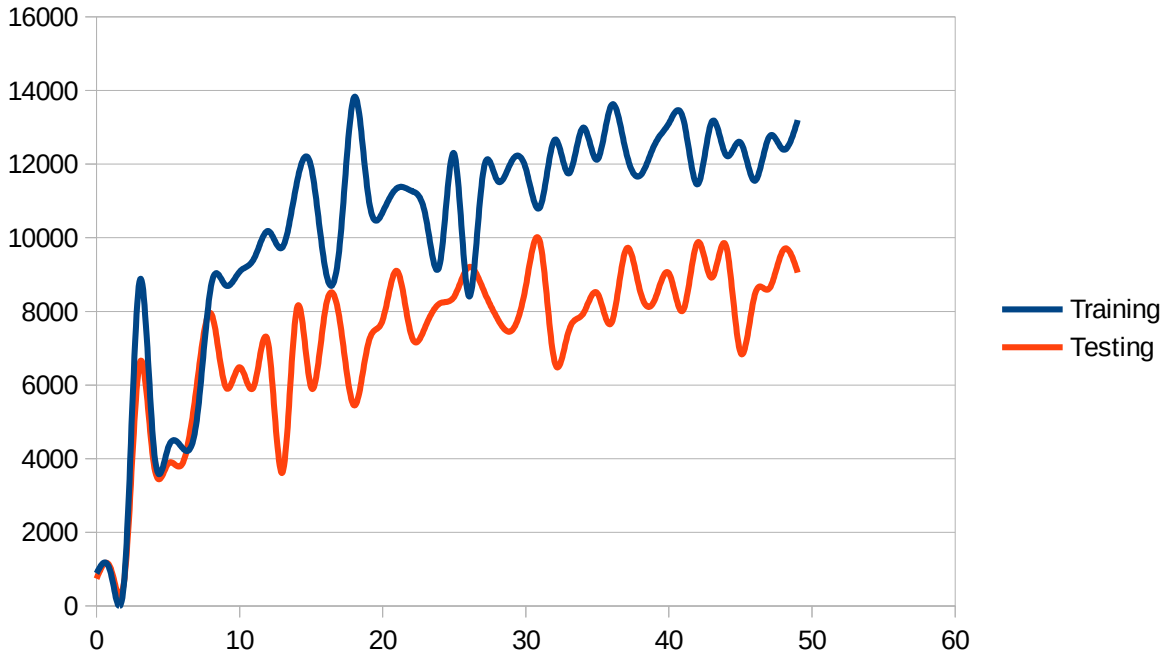
- Εκπαίδευση κάθε πράκτορα (χρωμοσώματος) σε ένα σταθερό επίπεδο. Αυτό το επίπεδο παρέμεινε σταθερό καθ όλη την διαδικασία εξέλιξης



Εικόνα 6.4 : Εκπαίδευση σε 1 σταθερό επίπεδο (seed 8827)

Βλέπουμε πως κοντά στην 10 γενιά αρχίζει και υπάρχει υπερπροσαρμογή (overfitting). Η δυνατότητα γενίκευσης του πράκτορα κρατιέται σε χαμηλά επίπεδα, ενώ αρχίζει και μεγιστοποιεί την απόδοση του στην συγκεκριμένη πίστα.

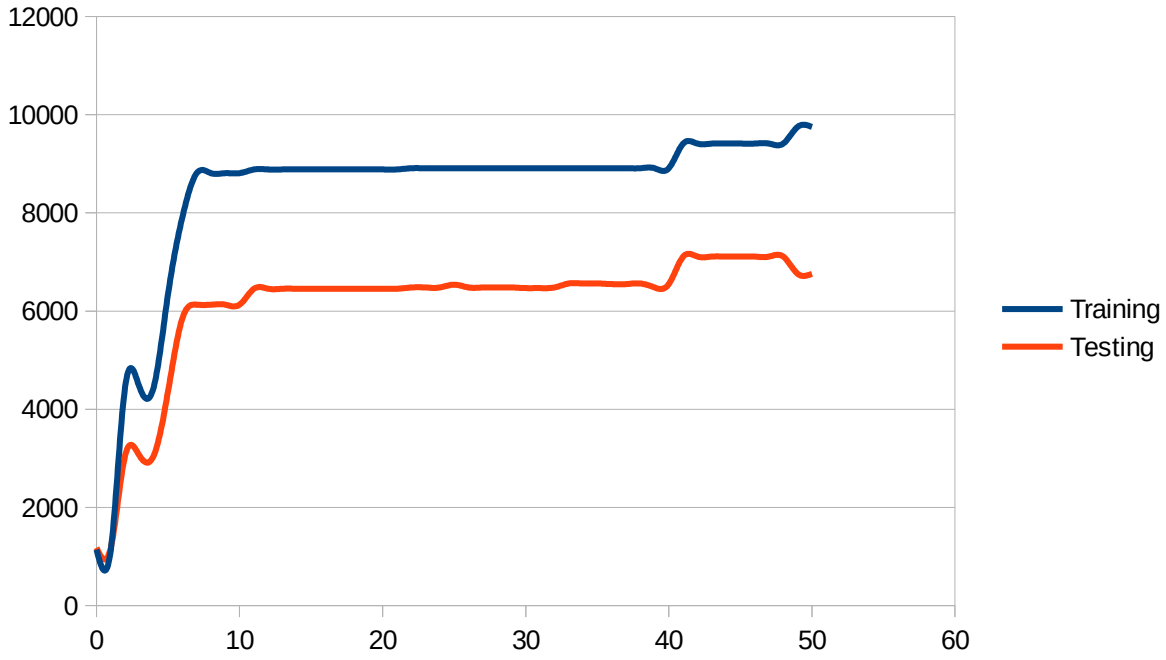
- Εκπαίδευση κάθε πράκτορα (χρωμοσώματος) σε ένα σταθερό επίπεδο ανα γενιά. Μόλις κάποιος πράκτορας απο την γενιά του πληθυσμού τερματίσει το επίπεδο, αυτό αλλάζει τυχαία για την επόμενη γενιά.



Εικόνα 6.5 : Εκπαίδευση σε 1 επίπεδο. Εναλλαγή επιπέδου σε περίπτωση νίκης

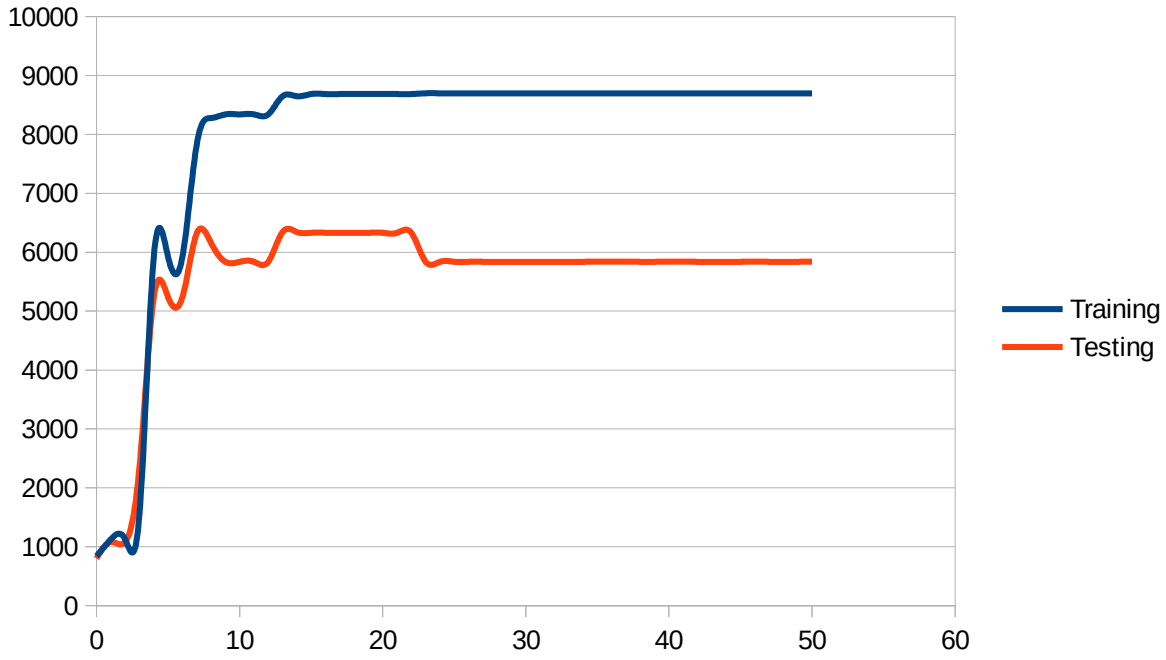
Στην περίπτωση εναλλαγής του επιπέδου εκπαίδευσης σε περίπτωση νίκης, βλέπουμε ότι υπάρχει μία ανοδική τάση στην απόδοση του πράκτορα σε τυχαία επίπεδα, καθώς επίσης και υψηλή απόδοση στο εκπαιδευόμενο επίπεδο.

- Εκπαίδευση κάθε πράκτορα (χρωμοσώματος) σε 5 επίπεδα ανα γενιά. Ως απόδοση του πράκτορα λαμβάνεται η μέση τιμή της απόδοσής του σε αυτά τα πέντε επίπεδα.



Εικόνα 6.6 : Εκπαίδευση σε 5 επίπεδα.

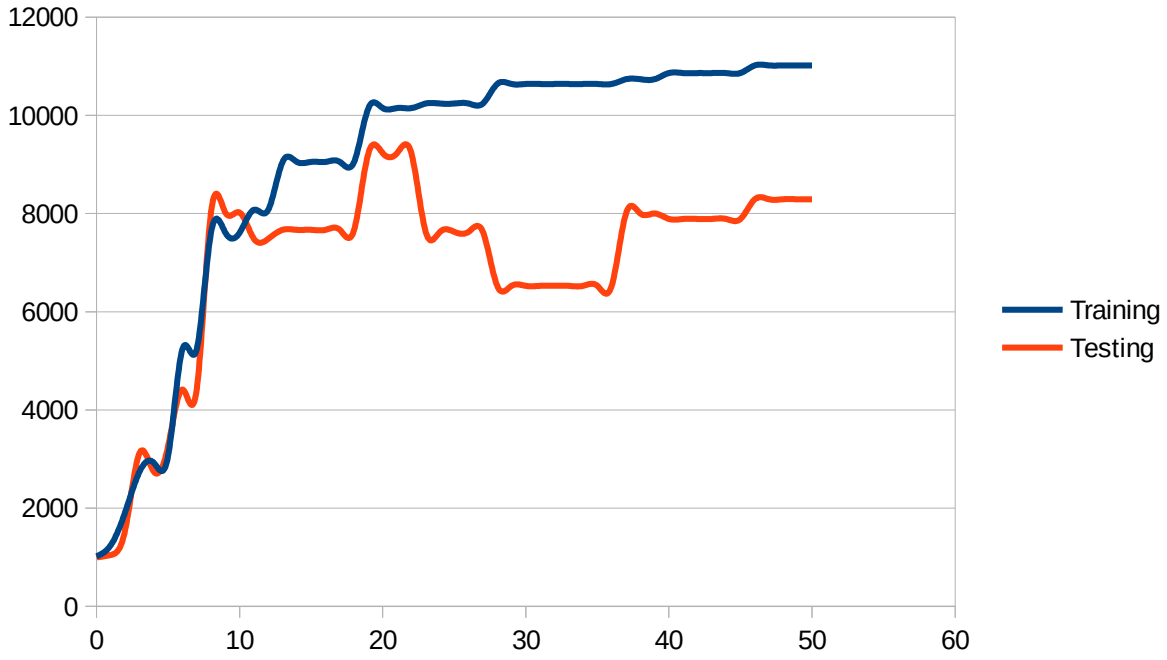
- Εκπαίδευση κάθε πράκτορα (χρωμοσώματος) σε 10 επίπεδα ανά γενιά. Ως απόδοση του πράκτορα λαμβάνεται η μέση τιμή της απόδοσης του σε αυτά τα δέκα επίπεδα. Όπως και στην προηγούμενη κατηγορία, τα επίπεδα αυτά κρατιούνται σταθερά και δεν αλλάζουν στην διάρκεια της εξελικτικής διαδικασίας.



Εικόνα 6.7 : Εκπαίδευση σε 10 επίπεδα

Στην περίπτωση των 5 καθώς και των 10 επιπέδων , βλέπουμε ότι ο πράκτορας βρίσκει ένα “τοίχο”. Κολλάει σε ένα τοπικό μέγιστο, και επειδή η απόδοση βγαίνει ως μέσος όρος δεν βοηθάει τον πράκτορα να βγάλει καινούργιες λύσεις καθώς η μικρή βελτίωση απορροφάται ως χειροτέρευση σε άλλα επίπεδα.

- Εκπαίδευση κάθε πράκτορα (χρωμοσώματος) σε 15 επίπεδα ανα γενιά. Ως απόδοση του πράκτορα λαμβάνεται η μέση τιμή της απόδοσης του σε αυτά τα 15 επίπεδα.



Εικόνα 6.8 : Εκπαίδευση σε 15 επίπεδα

Η αύξηση των επιπέδων εκπαίδευσης σε 15 φαίνεται να δίνει στον γενετικό αλγόριθμο μία ευκαιρία να φεύγει από τοπικά μέγιστα. Ίσως λόγο της ποικιλομορφίας των πολλών επιπέδων.

Συμπεράσματα

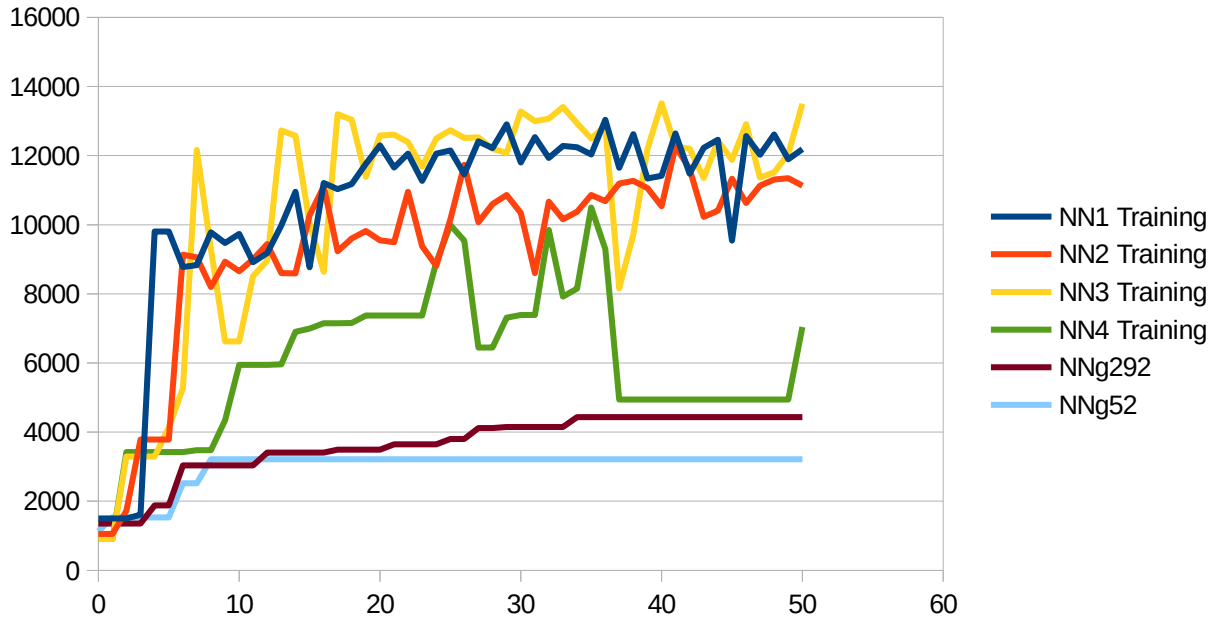
Καλύτερη γενίκευση σε συνδυασμό με καλή απόδοση στο εκπαιδευόμενο επίπεδο φαίνεται να έχουνε δύο τρόποι : α) οι εναλλαγή του επιπέδου εκπαίδευσης σε περίπτωση νίκης και β) οι εκπαίδευση σε 15 επίπεδα. Λόγω του ότι να εκπαιδεύεται κάθε χρωμόσωμα σε 15 επίπεδα είναι υπολογιστικά ακριβό, στα περισσότερα επόμενα πειράματα επιλέχτηκε οι μέθοδος του να αλλάζει το επίπεδο μόλις κάποιος πράκτορας από μία γενιά το περάσει.

6.1.3 Πείραμα 3 : Επιλογή εισόδων του νευρωνικού δικτύου

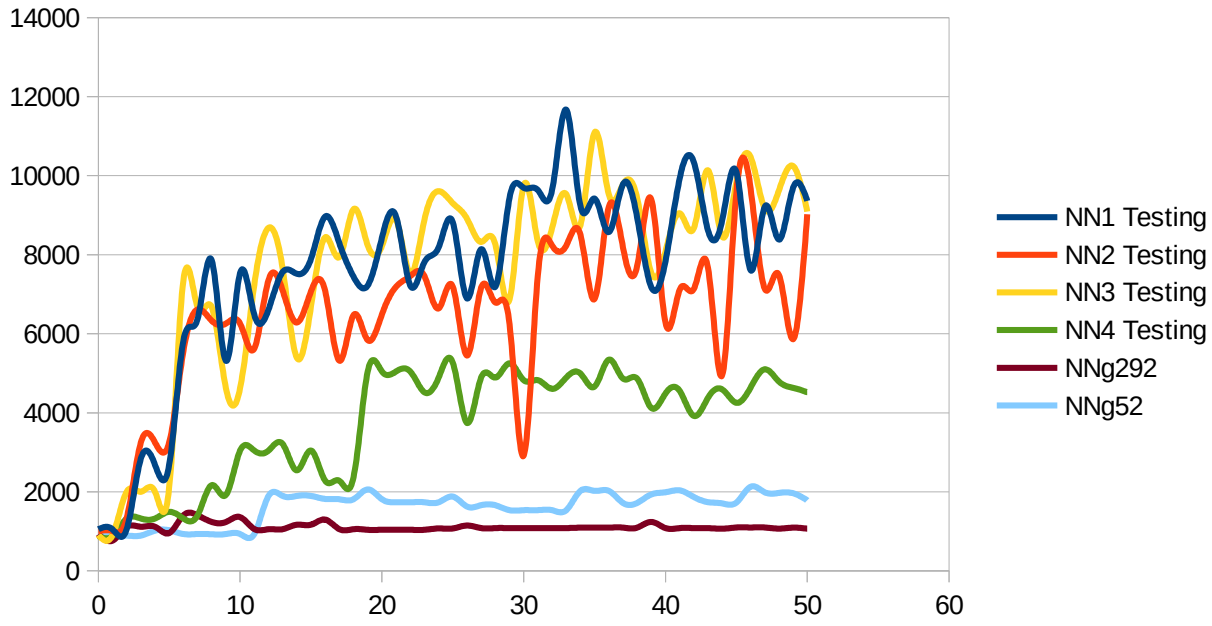
Στην προσπάθεια επιλογής της τοπολογίας του δικτύου για την εξέλιξη ενός καλού πράκτορα έπρεπε να επιλεχθούν οι είσοδοι του νευρωνικού δικτύου. Σε αυτό και το επόμενο πείραμα διεξήχθησαν μία σειρά από αξιολογήσεις των διαφόρων κατηγοριών εισόδων που ορίστηκαν στην παράγραφο 6.1.2 του κεφαλαίου 6. Σε όλες τις αξιολογήσεις ο αριθμός των νευρώνων εξόδου παρέμενε σταθερός, και ήταν τύπου I (ένας νευρώνας ανά διαθέσιμο κουμπί χειρισμού βλ. 6.1.3.1.) .

- Δοκιμάστηκαν οι 6 τύποι νευρωνικών δικτύων για 50 γενιές σε επίπεδα δυσκολίας 0.

Βλέπουμε παρακάτω τα διαγράμματα με την απόδοση των διαφόρων νευρωνικών ανά γενιά.



Εικόνα 6.9 : Απόδοση (σετ εκπαίδευσης) νευρωνικών χειριστών σύμφωνα με το είδος των νευρώνων εισόδου



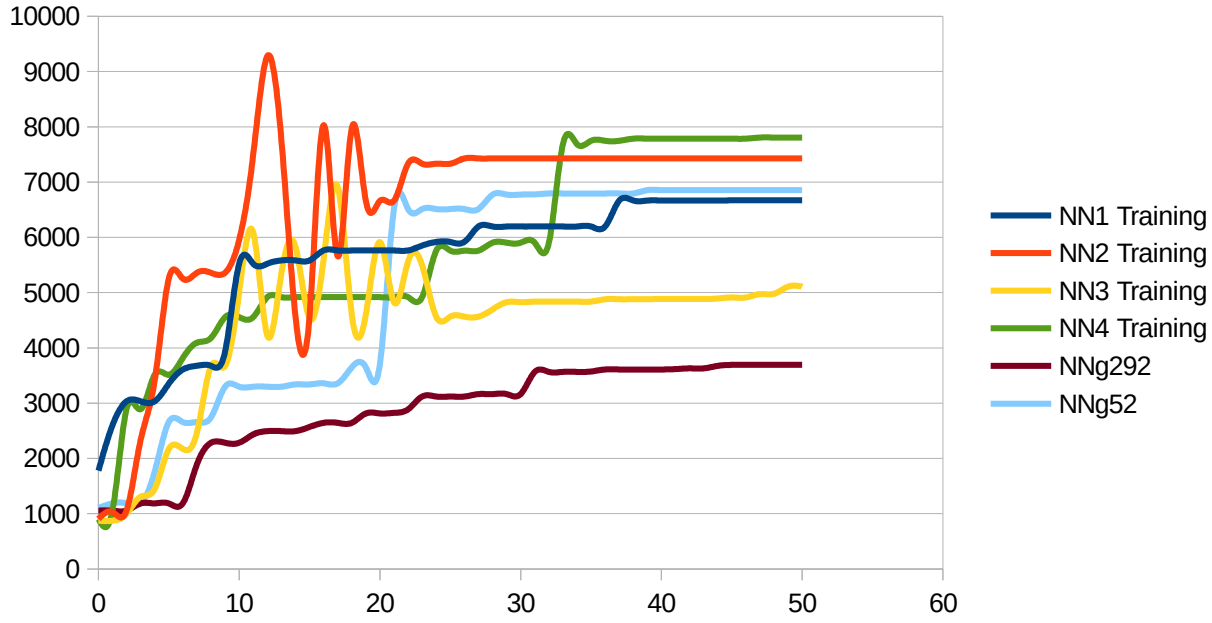
Εικόνα 6.10 : Απόδοση (σετ δοκιμασίας) νευρωνικών χειριστών σύμφωνα με το είδος των νευρώνων εισόδου

Στην εικόνα 7.9 βλέπουμε την απόδοση του καλύτερου πράκτορα ανά γενιά στο επίπεδο εκπαίδευσης. Όπως και στο προηγούμενο πείραμα, στο τέλος κάθε γενειάς ο καλύτερος πράκτορας αποτιμάται παίζοντας 10 τυχαία επίπεδα ίδιας δυσκολίας. Τα αποτελέσματα φαίνονται στην εικόνα 7.10

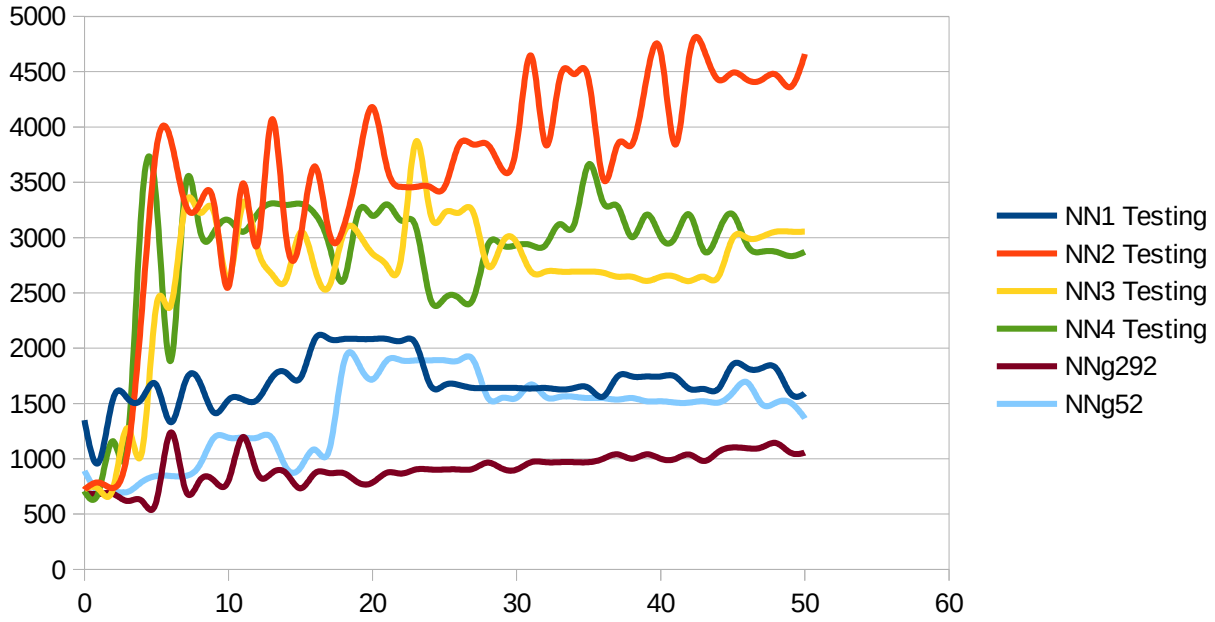
- Δοκιμάστηκαν οι 6 τύποι νευρωνικών δικτύων για 50 γενιές σε επίπεδα δυσκολίας 0 με την διαφορά ότι προστέθηκε ένα ακόμα είδος εχθρών. (spiky goombas). Αυτοί οι εχθροί δεν πεθαίνουν όταν ο mario προσγειώνεται επάνω τους.

Βλέπουμε παρακάτω ένα στιγμιότυπο από το επίπεδο δυσκολίας με αυτούς τους εχθρούς και στην συνέχεια τα διαγράμματα με την απόδοση των διαφόρων νευρωνικών ανά γενιά.

Εικόνα 6.11 : Στιγμιότυπο επιπέδου δυσκολίας 0 με spiky goombas



Εικόνα 6.12 : Απόδοση (σετ εκπαίδευσης) νευρωνικών χειριστών σύμφωνα με το είδος των νευρώνων εισόδου



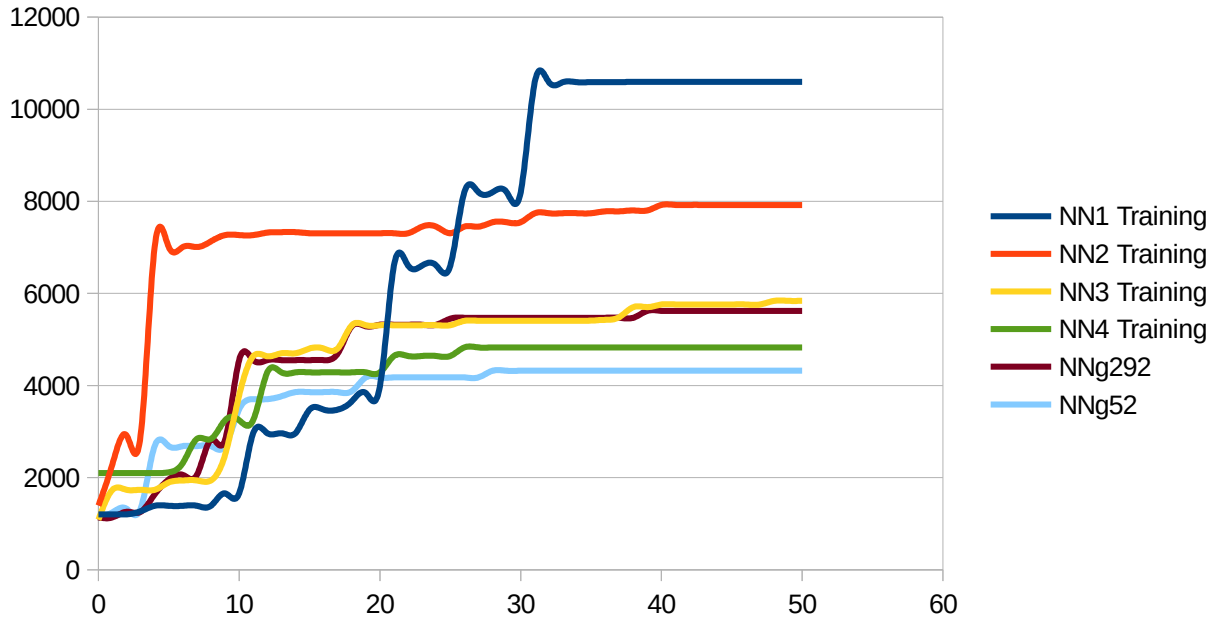
Εικόνα 6.13 : Απόδοση (σετ δοκιμασίας) νευρωνικών χειριστών σύμφωνα με το είδος των νευρώνων εισόδου

Καλύτερη απόδοση καθώς και καλύτερη γενίκευση παρουσιάζουν οι πράκτορες που βασίζονται σε νευρωνικά δίκτυα που μπορούν να αντιληφθούν την διαφορετική κατηγορία εχθρών, με το NN2 να ξεχωρίζει. Σε παρόμοιο πείραμα που είχε γίνει πρώτα εκπαίδευση σε επίπεδο 0 με τους απλούς εχθρούς και στην συνέχεια σε επίπεδο 0 με αγκαθωτούς εχθρούς καλή απόδοση παρουσίασε και το NN1 το οποίο έμαθε να κάνει άλματα και να πυροβολάει συνέχεια αποφεύγοντας τις συγκρούσεις.

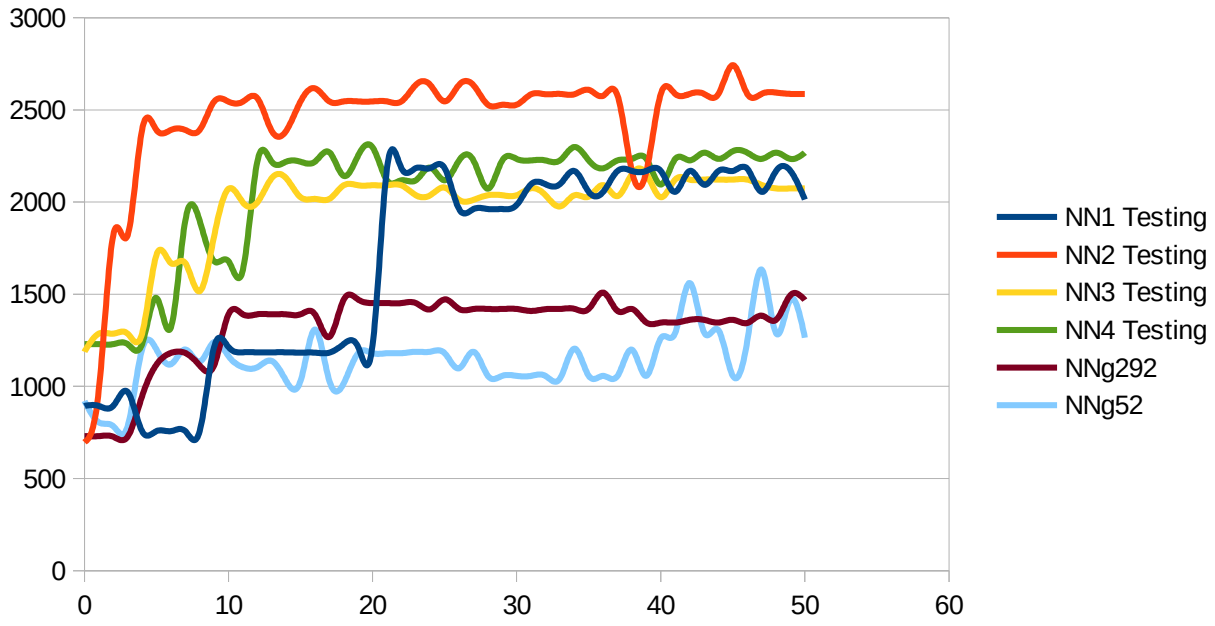
- Δοκιμάστηκαν οι 6 τύποι νευρωνικών δικτύων για 50 γενιές σε επίπεδα δυσκολίας 1. Σε αυτό το επίπεδο δυσκολίας υπάρχουν επιπρόσθετα από το επίπεδο δυσκολίας 0 :
    - 3 ειδών φτερωτοί εχθροί
    - κενά
    - πιο δύσκολη μορφολογικά πίστα
    - καινούργιοι επίγειοι εχθροί (χελώνες)

Παρακάτω βλέπουμε ένα στιγμιότυπο από αυτό το επίπεδο δυσκολίας καθώς και τα διαγράμματα απόδοσης.

Εικόνα 6.14 : Στιγμιότυπο επιπέδου δυσκολίας 1



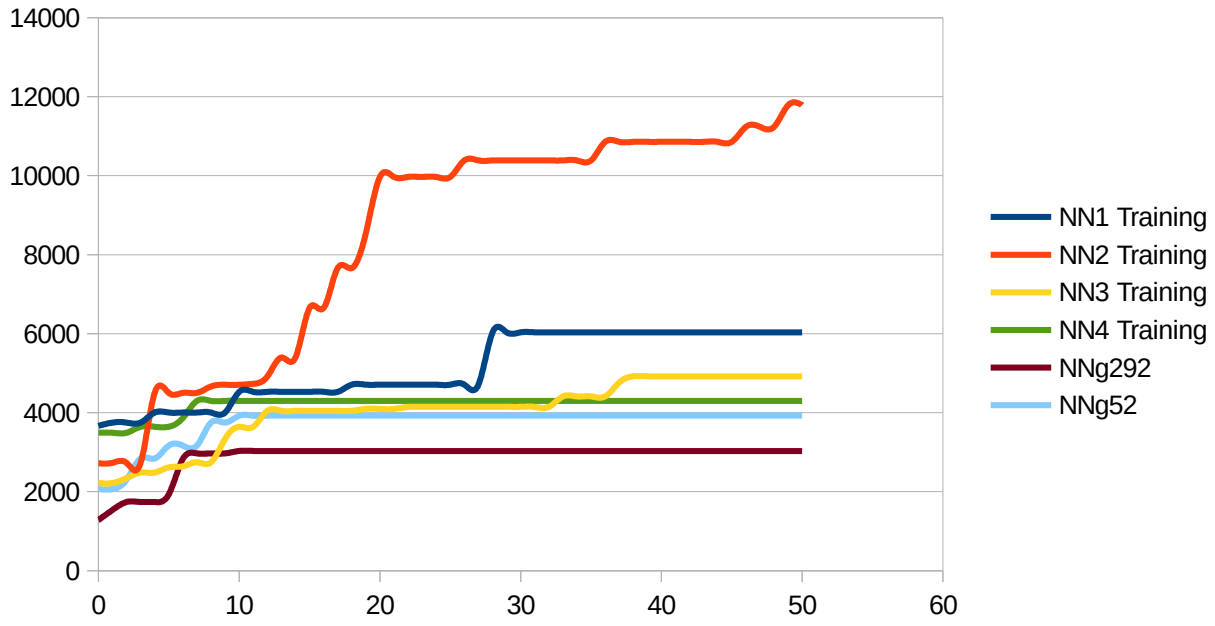
Εικόνα 6.15 : Απόδοση (σετ εκπαίδευσης) νευρωνικών χειριστών σύμφωνα με το είδος των νευρώνων εισόδου σε επίπεδο δυσκολίας 1



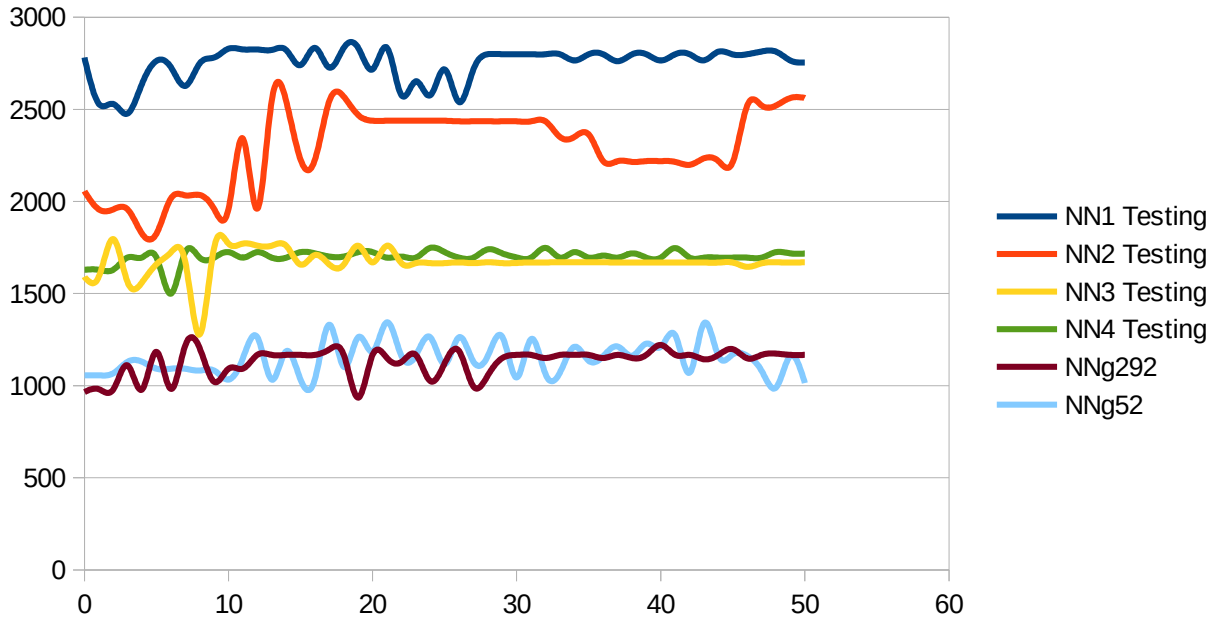
Εικόνα 6.16 : Απόδοση (σετ δοκιμασίας) νευρωνικών χειριστών σύμφωνα με το είδος των νευρώνων εισόδου σε επίπεδο δυσκολίας 1

Σε αυτό το αυξημένο επίπεδο δυσκολίας οι πράκτορες μαθαίνουν πολύ πιο αργά. Επίσης η ικανότητα τους γενίκευσης βελτιώνεται μόνο κατά 1000-2000 βαθμούς και μετά δεν υπάρχει βελτίωση. Ενδεχομένως αυτό να οφείλεται στο ότι κανένας πράκτορας δεν κατάφερε να τερματίσει το επίπεδο δυσκολίας 1 μέσα σε 50 γενιές, άλλα όπως θα δούμε και αργότερα όταν συμβαίνει αυτό το επίπεδο γενίκευσης φαίνεται να είναι σχεδόν μηδενικό.

- Δοκιμάστηκαν οι 6 τύποι νευρωνικών δικτύων για 50 γενιές σε επίπεδα δυσκολίας 1, αφού προηγουμένως είχαν εκπαιδευτεί για 50 γενιές σε επίπεδα δυσκολίας 0 (εναλλάσσοντας το επίπεδο εκπαίδευσης σε περίπτωση νίκης).



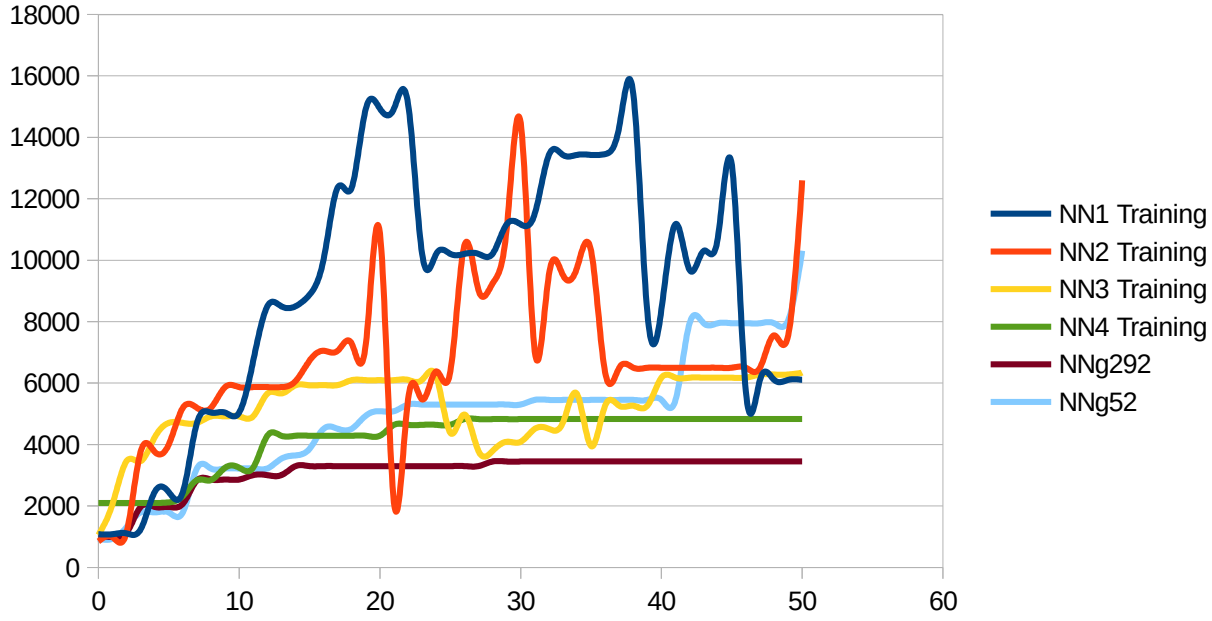
Εικόνα 6.17 : Απόδοση (σετ εκπαίδευσης) νευρωνικών χειριστών σύμφωνα με το είδος των νευρώνων εισόδου σε επίπεδο δυσκολίας 1 (με πρό εκπαίδευση σε επίπεδο 0)



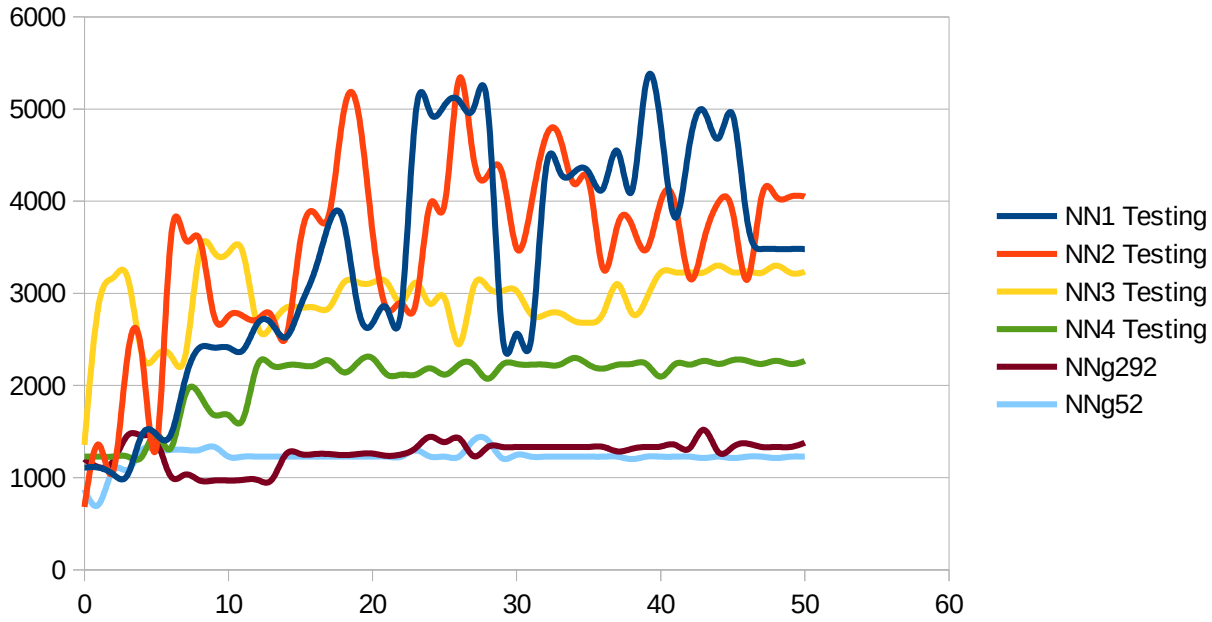
Εικόνα 6.18 : Απόδοση (σετ δοκιμασίας) νευρωνικών χειριστών σύμφωνα με το είδος των νευρώνων εισόδου σε επίπεδο δυσκολίας 1 (με προ εκπαίδευση σε επίπεδο 0)

Οι αποδόσεις των πρακτόρων αρχίζουν από υψηλότερες τιμές από ότι εάν δεν είχαμε προ-εκπαίδευση. Έτσι φαίνεται να μεταφέρεται ένα κομμάτι γνώσης από τα προηγούμενα επίπεδα δυσκολίας στα επόμενα.

- Δοκιμάστηκαν οι 6 τύποι νευρωνικών δικτύων για 50 γενιές σε επίπεδα δυσκολίας 1, τα οποία περιείχαν μόνο επίγειους εχθρούς (goomba) (εναλλάσσοντας το επίπεδο εκπαίδευσης σε περίπτωση νίκης).



Εικόνα 6.19 : Απόδοση (σετ εκπαίδευσης) νευρωνικών χειριστών σύμφωνα με το είδος των νευρώνων εισόδου σε επίπεδο δυσκολίας 1 (με εχθρούς επιπέδου 0)



Εικόνα 6.20 : Απόδοση (σετ δοκιμασίας) νευρωνικών χειριστών σύμφωνα με το είδος των νευρώνων εισόδου σε επίπεδο δυσκολίας 1 (με εχθρούς επιπέδου 0)

Τα νευρωνικά δίκτυα NN1 και NN2 παρουσιάζουν την καλύτερη απόδοση στα σετ εκπαίδευσης καθώς και στα σετ αξιολόγησης. Και τα δύο μάθανε να περνάνε πίστες με κενά. Βλέπουμε ότι μόλις αλλάξει κάποιο επίπεδο (αυτό φαίνεται από τις κορυφές στο διάγραμμα που ξεπερνάνε τις 10.000) μέσα σε 10-20 γενιές ο καλύτερος πράκτορας της γενιάς μπορεί να περάσει την πίστα. Το NN1 πέρασε συνολικά 4 πίστες , το NN2 πέρασε συνολικά 5 πίστες.

Σε αντίθεση ο μεγαλύτερος αριθμός νευρώνων των υπόλοιπων δικτύων καθ έστισε την εξέλιξη πολύ πιο αργή, χωρίς να έχουμε εμφανείς αποτελέσματα μέσα σε 50 γενιές.

Συμπεράσματα

Μόλις αλλάζει το επίπεδο δυσκολίας η αύξηση της πολυπλοκότητας και του χώρου των καταστάσεων είναι μεγάλη. Ενώ κάποιος πράκτορας μπορεί να αποδίδει καλά σε μία άδεια πίστα, σε μία πίστα μόνο με κενά ή σε μία πίστα μόνο με εχθρούς , όταν καλείται να παίξει μια πίστα με τον συνδυασμό αυτών δεν μπορεί να μάθει να γενικεύει καλά.

Το NN1 είχε την υψηλότερη απόδοση στην γενίκευση όταν είχε προηγηθεί εκπαίδευση σε επίπεδο 0. Αυτό το νευρωνικό δεν έχει την δυνατότητα να αντιληφθεί διαφορετικούς τύπους εχθρών έτσι ανέπτυξε μία πολιτική αποφυγής συγκρούσεων και άλματος σε συνδυασμό με φωτιά.

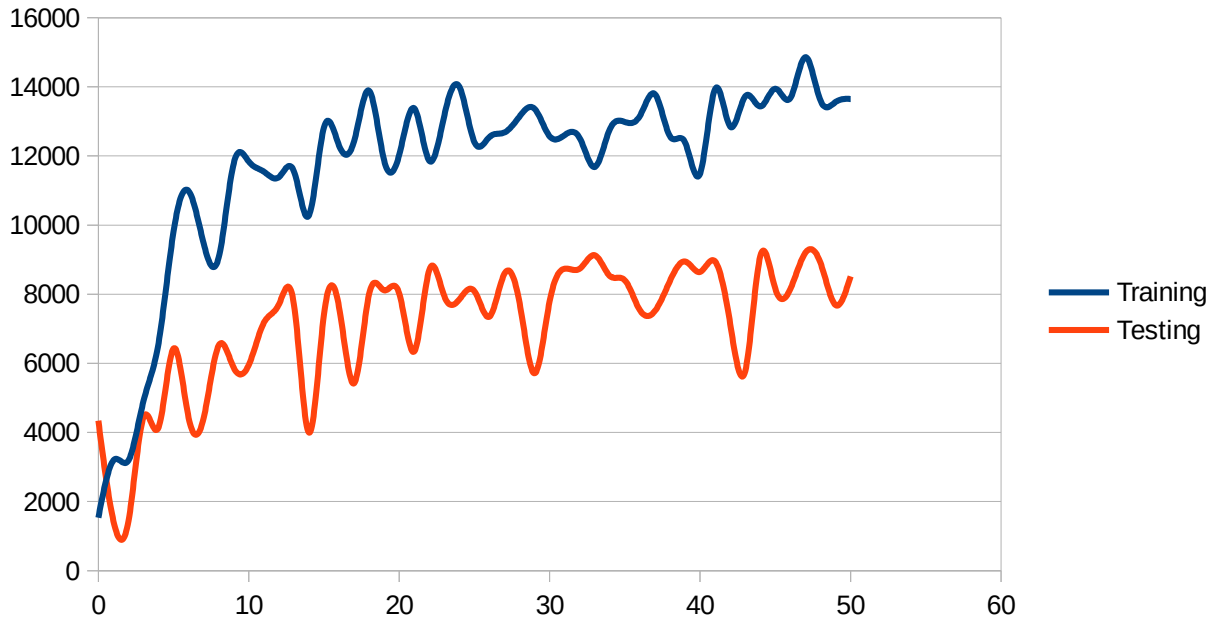
Το NN2 κατάφερε να έχει καλύτερη καμπύλη μάθησης από τα υπόλοιπα στο επίπεδο που προσθέσαμε εχθρούς που δεν πατιούνται. Επίσης ήρθε αρκετά κοντά στο να τερματίσει την πίστα του επιπέδου 1 όταν είχε προ εκπαιδευτεί στο επίπεδο 0. Επιλέχτηκαν οι είσοδοι του NN2 για τα επόμενα πειράματα, διότι παρόλο που το NN1 έχει καλύτερη απόδοση σε κάποιες κατηγορίες το NN2 φαίνεται να έχει την δυνατότητα να αντιλαμβάνεται εάν ο εχθρός που βρίσκεται κάτω από τον mario μπορεί να πατηθεί ή όχι.

6.1.4 Πείραμα 4 : Επιλογή εξόδων του νευρωνικού δικτύου

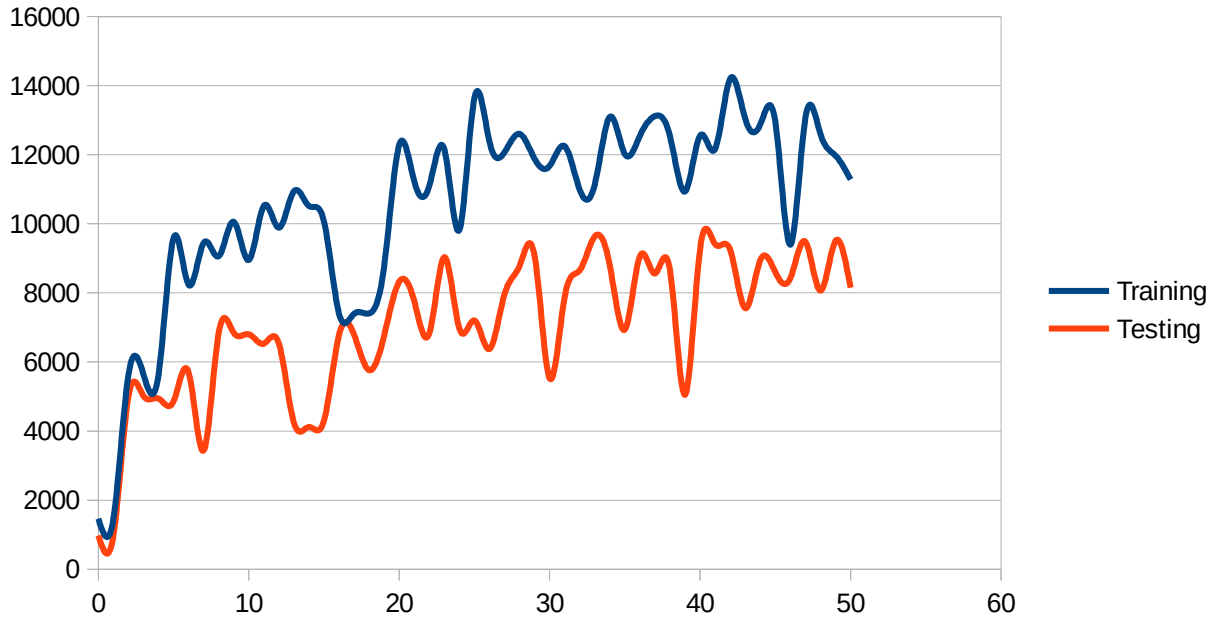
Αφού επιλέχτηκε το είδος των εισόδων των νευρωνικών δικτύων που θα εκπαιδευτούν με τον γενετικό αλγόριθμο, σε αυτό το πείραμα επιλέχτηκε το ειδος των νευρώνων εξόδου. Στην παράγραφο 6.1.3 του κεφαλαίου 6, περιγράφονται τα 3 είδη εξόδου που θα αξιολογήσουμε.

Για να μελετηθεί η απόδοση των διαφόρων ειδών , επιλέχτηκε να δοκιμαστούν σε 2 σετ από εκπαιδεύσεις.

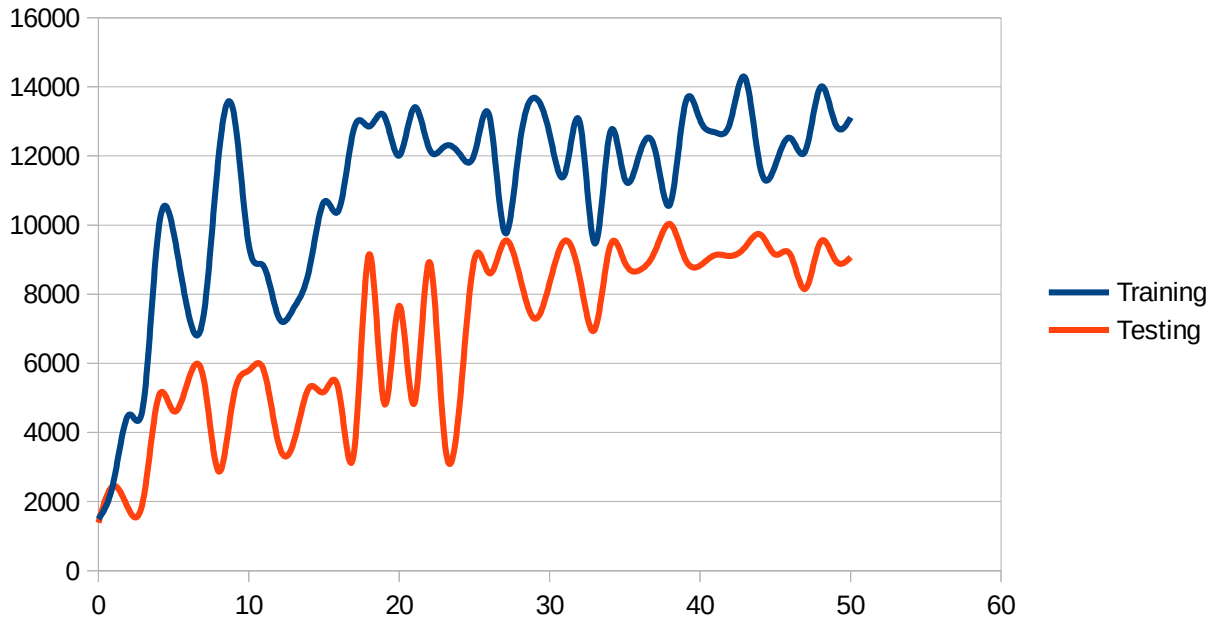
- Εκπαίδευση για 50 γενιές σε επίπεδα δυσκολίας 0. Κατά την διάρκεια εκπαίδευσης εάν κάποιος χειριστείς τερμάτιζε το επίπεδο , αυτό άλλαζε για την επόμενη γενιά. Παρακάτω παρουσιάζονται οι τιμές της συνάρτησης καταλληλότητας της κάθε κατηγορίας για το επίπεδο στο οποίο εκπαιδεύονταν καθώς και για ένα σετ δοκιμασίας που απαρτίζονταν από 10 τυχαία επίπεδα ίδιας δυσκολίας.



Εικόνα 6.21 : Απόδοση καλύτερου νευρωνικού χειριστή ανά γενιά σε επίπεδο δυσκολίας 0. (Τύπος νευρώνων εξόδου Ι)



Εικόνα 6.22 : Απόδοση καλύτερου νευρωνικού χειριστή ανά γενιά σε επίπεδο δυσκολίας 0. (Τύπος νευρώνων εξόδου ΙΙ)

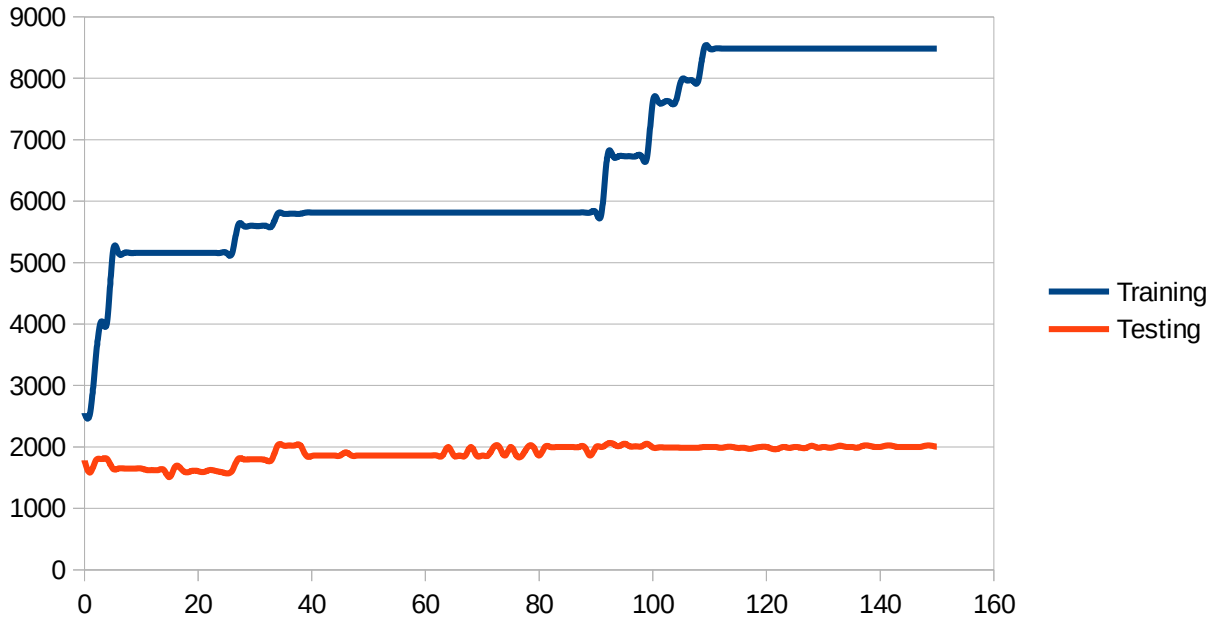


Εικόνα 6.23 : Απόδοση καλύτερου νευρωνικού χειριστή ανά γενιά σε επίπεδο δυσκολίας 0. (Τύπος νευρώνων εξόδου III)

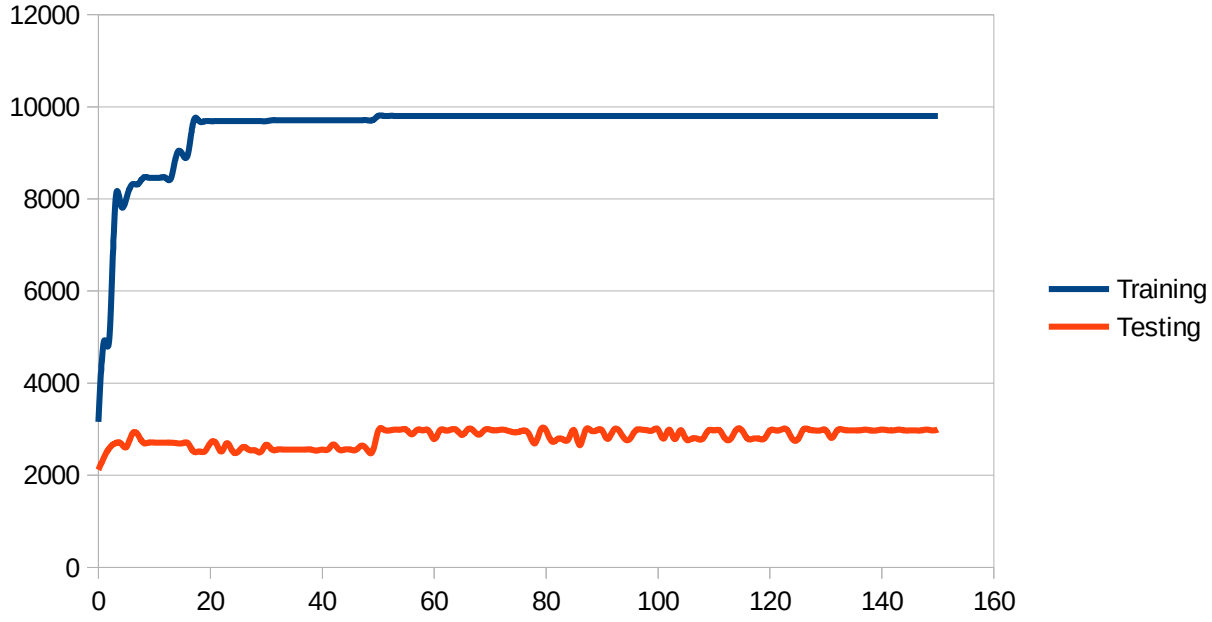
Παρατηρούμε ότι με τον τύπο I είχαμε απόδοση με λιγότερες διακυμάνσεις τόσο στο σετ εκπαίδευσής όσο και στο σετ δοκιμασίας. Πάραυτα ο βαθμός τις απόδοσης ήταν χαμηλότερος από του τύπου III. Επίσης στον τύπο I παρατηρείται η μεγαλύτερη διαφορά μεταξύ της απόδοσης στο σετ εκπαίδευσης και στο σετ δοκιμασίας.

Υψηλότερη απόδοση παρατηρείται στον τύπο III. Πετυχαίνει μεγαλύτερη απόδοση στο εκπαιδευόμενο κάθε φορά επίπεδο, και φαίνεται να γενικεύει καλύτερα.

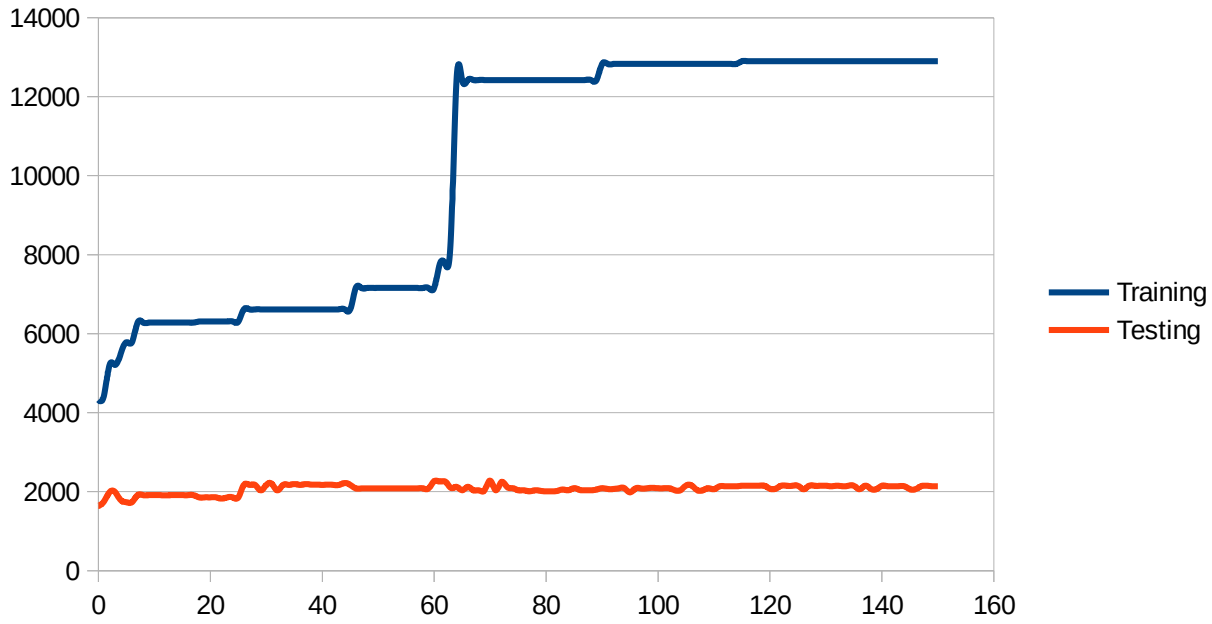
- Στην συνέχεια για κάθε τύπο, συνέχισε να εκπαιδεύεται ο πληθυσμός αυτή την φορά σε ένα μόνο επίπεδο δυσκολίας 1 για 150 γενιές.



Εικόνα 6.24 : Απόδοση καλύτερου νευρωνικού χειριστή ανά γενιά σε επίπεδο δυσκολίας 1. (Τύπος νευρώνων εξόδου I)



Εικόνα 6.25 : Απόδοση καλύτερου νευρωνικού χειριστή ανά γενιά σε επίπεδο δυσκολίας 1. (Τύπος νευρώνων εξόδου II)



Εικόνα 6.26 : Απόδοση καλύτερου νευρωνικού χειριστή ανά γενιά σε επίπεδο δυσκολίας 1. (Τύπος νευρώνων εξόδου III)

Σε όλους τους τύπους παρατηρούμε πολύ μικρή βελτίωση στην ικανότητα γενίκευσης του πράκτορα της τάξεως τον 100-600 πόντων. Κατά την εκπαίδευση φαίνεται ότι οι εξέλιξη των νευρωνικών με νευρώνες εξόδου τύπου II σύγκλινε πολύ γρήγορα σε ένα τοπικό μέγιστο, το οποίο δεν μπόρεσε να ξεπεράσει. Στον τύπο I βλέπουμε πιο αργή σύγκλιση και βελτίωση στην απόδοση των χειριστών. Την καλύτερη απόδοση στην εκπαιδευόμενη πίστα είχε ο πράκτορας που εκπαιδεύτηκε εξελίσσοντας νευρωνικά με νευρώνες εξόδου τύπου III (ένας νευρώνας ανά επιτρεπτό συνδυασμό κινήσεων) , και αυτός επιλέχτηκε για περαιτέρω μελέτη σε αυτήν την διπλωματική.

Έτσι καταλήξαμε στην μορφολογία του νευρωνικού που θα χρησιμοποιεί ο πράκτορας και θα εξετάσουμε στη συνέχεια την απόδοση του σε επίπεδα 0, 1 , 2.

- Δίκτυο εμπρόσθιας τροφοδότησης
- Νευρώνες εισόδου τύπου 2 (GANN_2)
- 1 κρυμμένο επίπεδο με 10 νευρώνες
- Νευρώνες εξόδου τύπου III

6.1.5 Πείραμα 5 : Απόδοση σε επίπεδα δυσκολίας 0

Σε αυτό το πείραμα εκπαιδεύτηκε ένας πληθυσμός 67 πρακτόρων για 50 γενιές σε επίπεδα δυσκολίας 0. Οι πράκτορες αποτελούνται από το νευρωνικό δίκτυο το οποίο αποφασίστηκε να μελετηθεί παραπάνω στο πείραμα 4.

Επειδή το αποτέλεσμα της εκπαίδευσης είναι μη-ντετερμινιστικό επαναλήφθηκε 5 φορές η διαδικασία εξέλιξης (από την αρχή κάθε φορά) και επιλέχθηκε ο καλύτερος πράκτορας.

Στη συνέχεια δοκιμάστηκε ο πράκτορας σε 100 επίπεδα δυσκολίας 0 των οποίων το μήκος κυμαινόταν από 0 – 511 και παραγόταν βάση του αριθμού του επιπέδου.

```
Level length = (200 + (i * 12) + (seed % (i + 1))) % 512
```

GANN_2	**Forward Jumping Agent**	**Random Agent**
Competition score: 3692.75	Competition score: 3085.16	Competition score: 1995.44
Number of levels cleared = 78.0	Number of levels cleared = 63.0	Number of levels cleared = 26.0
Additional (tie-breaker) info:	Additional (tie-breaker) info:	Additional (tie-breaker) info:
Total time left = 13457	Total time left = 16854	Total time left = 4846
Total kills = 1907	Total kills = 901	Total kills = 1535
Mario mode sum = 128	Mario mode sum = 83	Mario mode sum = 122
Number of disqualifications= 0	Number of disqualifications= 0	Number of disqualifications= 0

Πίνακας 6.3 : Βαθμολογία Πρακτόρων σε 100 επίπεδα δυσκολίας 0

Παραπάνω βλέπουμε την σύγκριση του πράκτορα σε σχέση με τον ForwardJumpingAgent και τον RandomAgents που περιέχονται στην πλατφόρμα Mario AI. Ο νευρωνικός χειριστής κατάφερε να περάσει 23,8% περισσότερα επίπεδα από το ForwardJumping Agent και 300% περισσότερα επίπεδα από τον Random Agent, περνώντας συνολικά 78 επίπεδα στα 100.

Ο GANN_2 σκότωσε συνολικά 1907 εχθρούς, διπλάσιους από τον ForwardJumpingAgent. Φαίνεται έτσι ότι έμαθε όχι μόνο να κάνει άλματα και να προσπερνάει εχθρούς, άλλα να σκοτώνει και όσους περισσότερους μπορεί στην πορεία.

Οι τρεις αυτοί πράκτορες αξιολογήθηκαν επιπλέον σε μία παραλλαγή του GamePlayTrack της πλατφόρμας Mario AI. Εδώ ομοίως με παραπάνω το μήκος των επιπέδων ήταν μεταβλητό. Επιπρόσθετα άλλαζε η μορφολογία της πίστας (με κενά, υπόγεια πίστα , πίστα χωρίς εχθρούς). Ο κάθε πράκτορες αξιολογήθηκε σε 20 πίστες επιπέδου δυσκολίας 0.

Παρακάτω παρουσιάζονται τα αποτελέσματα.

GANN_2	ForwardJumpingAgent
[~ Mario AI Benchmark ~ 0.2.0]	[~ Mario AI Benchmark ~ 0.2.0]
Evaluating agent GANN Player 2 with seed 1429766	Evaluating agent ForwardJumpingAgent with seed 1429766
[MarioAI] ~ Evaluation Results for Task: GamePlayTask	[MarioAI] ~ Evaluation Results for Task: GamePlayTask
Weighted Fitness : 156896	Weighted Fitness : 128592
Mario Status : 14	Mario Status : 10
Mario Mode : 30	Mario Mode : 21
Collisions with creatures : 12	Collisions with creatures : 25
Passed (Cells, Phys) : 5923 of 6397, 94826 of 102352 (92% passed)	Passed (Cells, Phys) : 4828 of 6397, 77348 of 102352 (75% passed
Time Spent(marioseconds) : 2006	Time Spent(marioseconds) : 1384
Time Left(marioseconds) : 1980	Time Left(marioseconds) : 2602
Coins Gained : 1026 of 4542 (22% collected)	Coins Gained : 811 of 4542 (17% collected)
Hidden Blocks Found : 3 of 364 (0% found)	Hidden Blocks Found : 2 of 364 (0% found)
Mushrooms Devoured : 0 of 0 found (0% collected)	Mushrooms Devoured : 0 of 1 found (0% collected)
Flowers Devoured : 0 of 23 found (0% collected)	Flowers Devoured : 0 of 21 found (0% collected)
kills Total : 347 of 792 found (43%)	kills Total : 180 of 792 found (22%)
kills By Fire : 232	kills By Fire : 99
kills By Shell : 0	kills By Shell : 0
kills By Stomp : 115	kills By Stomp : 81
disqualifications : 0	disqualifications : 0

Πίνακας 6.4 : Βαθμολογία Πρακτόρων (GANN_2 & ForwardJumpingAgent) σε 20 επίπεδα δυσκολίας 0 – GamePlayTrack (0)

Random Agent
[~ Mario AI Benchmark ~ 0.2.0]
Evaluating agent RandomAgent with seed 1429766
[MarioAI] ~ Evaluation Results for Task: GamePlayTask
Weighted Fitness : 65128
Mario Status : 0
Mario Mode : 25
Collisions with creatures : 23
Passed (Cells, Phys) : 2566 of 6397, 41226 of 102352 (40% passed)
Time Spent(marioseconds) : 3462
Time Left(marioseconds) : 533
Coins Gained : 552 of 4542 (12% collected)
Hidden Blocks Found : 4 of 364 (1% found)
Mushrooms Devoured : 2 of 5 found (40% collected)
Flowers Devoured : 5 of 17 found (29% collected)
kills Total : 245 of 792 found (30%)
kills By Fire : 165
kills By Shell : 0
kills By Stomp : 80
disqualifications : 0

Πίνακας 6.5 : Βαθμολογία Πράκτορα RandomAgent σε 20 επίπεδα δυσκολίας 0 – GamePlayTrack (0)

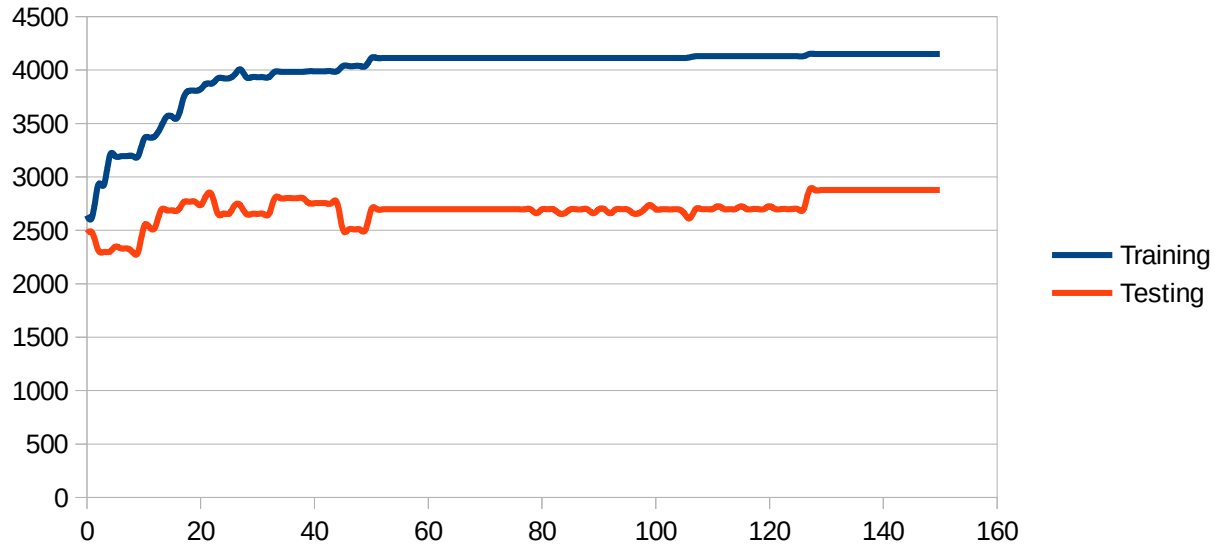
Και σε αυτήν την αξιολόγηση ο GANN_2 σκότωσε σχεδόν διπλάσιους εχθρούς (43%) αντί για (22%) που σκότωσε ο ForwardJumpingAgent. Από αυτούς σκότωσε πολλούς παραπάνω με φωτιά, που υποδεικνύει ότι έμαθε αποτελεσματικά να πυροβολάει έναντι του απλά να πηδάει προς τα δεξιά. Από το σύνολο του μήκους των πιστών κατάφερε να περάσει το 92% , έναντι 75% του ForwardJumpingAgent και 40% του RandomAgent. Επίσης ο αριθμός των συγκρούσεων του με πλάσματα ήταν μόλις 12, έναντι 25 του ForwardJumpingAgent και 23 του

RandomAgent. Βλέπουμε ότι ανέπτυξε και μία στρατηγική για να αποφεύγει την επαφή με τους εχθρούς.

Συνολικά παρατηρούμε ότι στα επίπεδα δυσκολίας 0 , ο πράκτορας καταφέρνει να γενικεύει καλά την γνώση που αποκομίζει κατά την εκπαίδευση και να την εφαρμόζει σε επίπεδα που δεν έχει δει.

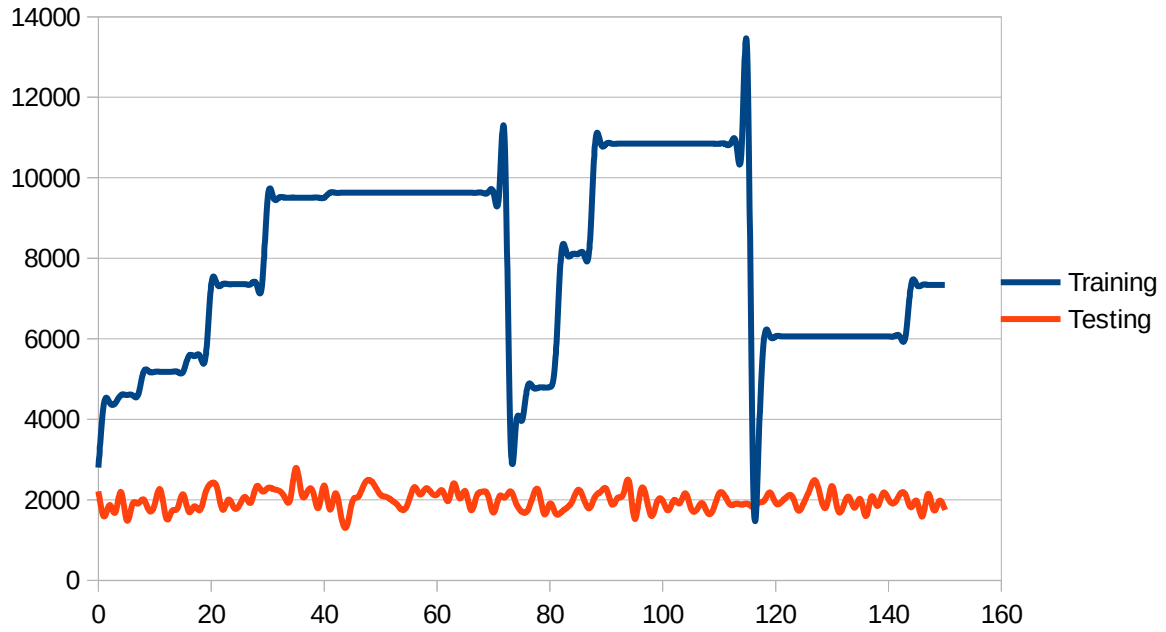
6.1.6 Πείραμα 6 : Απόδοση σε επίπεδα δυσκολίας 1

Όπως είδαμε και προηγουμένως (Πείραμα 4) , κανένας από τους πράκτορες δεν κατάφερε να γενικεύσει καλά σε επίπεδα δυσκολίας 1. Για να προσπαθήσουμε να βελτιώσουμε την ικανότητα του πράκτορα να περνάει επίπεδα δυσκολίας 1, πρώτα φτιάχτηκε ένας πληθυσμός βάσης ο οποίος εκπαιδεύτηκε για 150 γενιές. Το κάθε άτομο/πράκτορας της γενιάς έπαιζε 15 επίπεδα ανά γενιά και η απόδοση του έβγαινε ως ο μέσος όρος των αποδόσεων στα μεμονωμένα επίπεδα.



Εικόνα 6.27 : Απόδοση καλύτερου νευρωνικού χειριστή ανά γενιά σε επίπεδο δυσκολίας 1. Εκπαίδευση σε 15 επίπεδα.

Στην συνέχεια, ο πληθυσμός αυτός εκπαιδεύτηκε για ακόμα 150 γενιές, αυτήν την φορά σε ένα επίπεδο το οποίο άλλαζε εάν κάποιος πράκτορας κατάφερνε να το περάσει.



Εικόνα 6.28 : Απόδοση καλύτερου νευρωνικού χειριστή ανά γενιά σε επίπεδο δυσκολίας 1. Εκπαίδευση σε 1 επίπεδο, εναλλαγή σε περίπτωση νίκης.

Παρότι στο σετ δοκιμασίας ο πράκτορας συνεχίζει να μην έχει καλή απόδοση, χρησιμοποιώντας την βάση που φτιάχτηκε για να την εξελίξουμε περαιτέρω, βλέπουμε ότι μαθαίνει να περνάει το επίπεδο δυσκολίας 1 στο οποίο εκπαιδεύεται μέσα σε 50-70 γενιές.

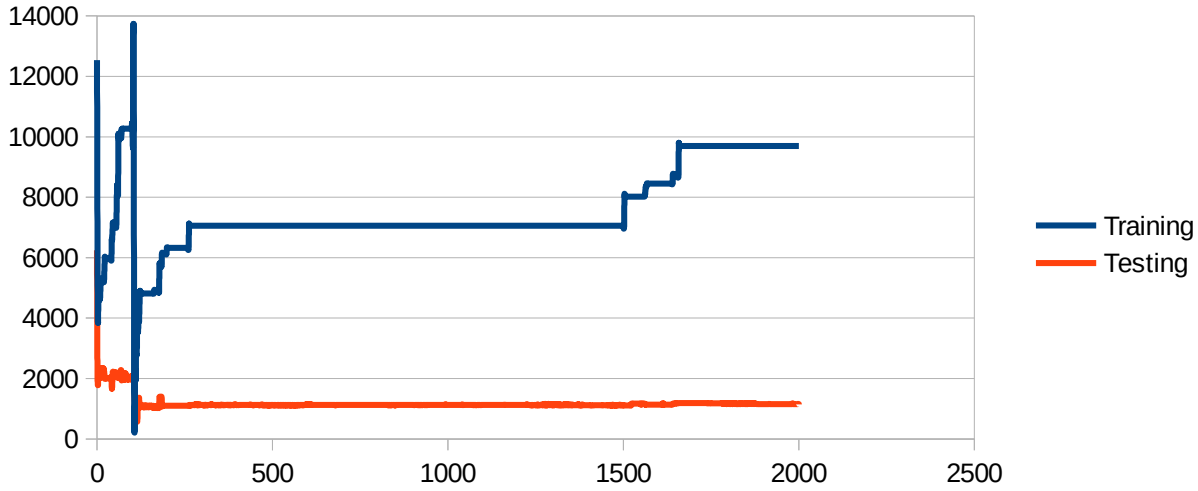
Μπορούμε να το συγκρίνουμε αυτό με την εικόνα 7.26 που μέσα σε 150 γενιές δεν κατάφερε να περάσει κάποιο επίπεδο δυσκολίας 1. Εδώ νικάει ένα επίπεδο στην 71η γενιά, και περνάει το επόμενο στην 115 γενιά.

Συμπεράσματα

Η εκπαίδευση του συγκεκριμένου νευρωνικού φαίνεται να μην μπορεί να φτιάξει έναν καλό γενικευμένο πράκτορα για ποιο δύσκολα επίπεδα. Δοσμένης όμως μίας καλής βάσης, μπορεί να εξελίξει και να βελτιστοποιήσει έναν χειριστή για ένα συγκεκριμένο επίπεδο.

6.1.7 Πείραμα 7 : Εναλλάσσοντας το επίπεδο δυσκολίας

Σε αυτό το πείραμα ξεκινώντας από την βάση που φτιάχτηκε στο πείραμα 6 (έναν πληθυσμό εκπαιδευμένο για 150 γενιές σε 15 επίπεδα ανά γενιά) , αφήσαμε τον γενετικό να τρέχει για 2000 γενιές. Μόλις κάποιος πράκτορας περνούσε ένα επίπεδο κάποιας δυσκολίας, άλλαζε το επίπεδο (τυχαίο) και αυξανόταν η δυσκολία κατά μία μονάδα.



Εικόνα 6.29 : Απόδοση καλύτερου νευρωνικού χειριστή ανά γενιά σε επίπεδο αυξανόμενης δυσκολίας

Το επίπεδο δυσκολίας 0 περάστηκε από την πρώτη κιόλας γενιά. Στην συνέχεια στην 104η γενιά περάστηκε και το επίπεδο δυσκολίας 1. Μετά για 1000 περίπου γενιές είχε παραμείνει σε κάποιο τοπικό μέγιστο, ώσπου μετά από αποτέλεσμα κάποιας μετάλλαξης πιθανόν σε συνδυασμό με κάποια διασταύρωση κατάφερε να το ξεπεράσει και να συνεχίσει την ανοδική πορεία. Παρά την μεγάλη διάρκεια της εκπαίδευσης δεν μπόρεσε κάποιος πράκτορας να περάσει το επίπεδο δυσκολίας 2.

Ένα μειονέκτημα της διαδικασίας εξέλιξης , είναι ότι καθώς αυξάνεται το επίπεδο δυσκολίας , ο πράκτορας όχι μόνο δυσκολεύεται αρκετά στο να προσαρμοστεί στο καινούργιο επίπεδο , αλλά ξεχνάει και οτιδήποτε έχει μάθει για τα προηγούμενα επίπεδα.
Για να το δείξουμε αυτό τρέξαμε μία αξιολόγηση του πράκτορα μετά το τέλος των 2000 γενιών.

Ο πράκτορας αξιολογήθηκε σε 10 επίπεδα ανά κατηγορία δυσκολίας.

Scoring controller thesis.GANN_2@2de8d9d with starting seed 17657
[~ Mario AI Benchmark ~ 0.2.0]

Difficulty 0	Difficulty 1	Difficulty 2
Competition score: 41.2	Competition score: 92.7	Competition score: 229.55
Number of levels cleared = 0.0	Number of levels cleared = 0.0	Number of levels cleared = 0.0
Additional (tie-breaker) info:	Additional (tie-breaker) info:	Additional (tie-breaker) info:
Total time left = 0	Total time left = 526	Total time left = 1468
Total kills = 19	Total kills = 45	Total kills = 57
Mario mode (small, large, fire) sum = 38	Mario mode (small, large, fire) sum = 34	Mario mode (small, large, fire) sum = 21
Number of disqualifications= 0	Number of disqualifications= 0	Number of disqualifications= 0

Πίνακας 6.6: Βαθμολογία Πράκτορα GANN_2 σε 10 επίπεδα δυσκολίας 0-2

Παρατηρούμε ότι έχει ξεχάσει ότι είχε μάθει για τα επίπεδα δυσκολίας 0 και δεν καταφέρνει να περάσει ούτε ένα επίπεδο.

Ένα χιουμοριστικό στοιχείο εμφανίζεται στην στρατηγική που έχει αναπτύξει ο πράκτορας. Ο πράκτορας τρέχει μόνιμα προς τα αριστερά και πολλές φορές σκυφτός (σαν να είναι φοβισμένος από τους εχθρούς που έρχονται) , και αρχίζει να προχωράει προς τα δεξιά όταν έρθει κάποιος εχθρός στην περιοχή όρασης του. Έτσι η πορεία του προς τα δεξιά μαγνητίζεται από τους εχθρούς μπροστά. Αυτό είναι καλό για να περάσει την πίστα δυσκολίας 2 στην οποία εκπαιδευόταν, όμως στις πίστες μικρότερης δυσκολίας που οι εχθροί είναι λιγότεροι και λιγότερο επιθετικοί δεν μπορεί να εφαρμοστεί.

Ένα στιγμιότυπο αυτής της συμπεριφοράς παρουσιάζεται παρακάτω.

Εικόνα 6.30 : Στιγμιότυπο από το παίξιμο στην πίστα δυσκολίας 2 που εκπαιδευόταν ο πράκτορας

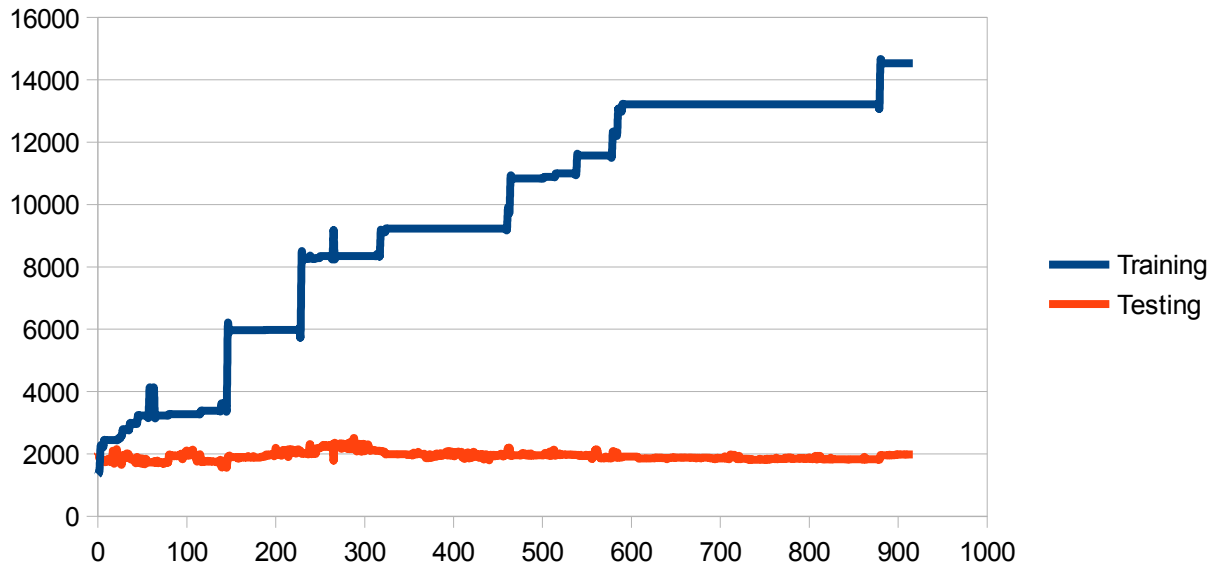
6.1.8 Πείραμα 8 : Επιχειρώντας επίπεδο δυσκολίας 2

Σε αυτό το πείραμα ξεκινήσαμε πάλι από την βάση πληθυσμού που φτιάχτηκε στο πείραμα 6. Επιπλέον μειώσαμε την τυχαία τιμή που προστίθεται στις μεταλλάξεις, για να είναι πιο ομαλές

οι μεταβολές. Πλέον οι τιμές θα επιλέγονται από μία γκαουσιανή κατανομή με μέση τιμή 0 και τυπική απόκλιση 0.075 αντί 0.1.

Στη συνέχεια αυξήθηκε η πιθανότητα μετάλλαξης κάθε γονιδίου σε αυτά τα χρωμοσώματα που επιλέγονταν για μετάλλαξη σε 0.7 από 0.1, στις αρχικές γενιές. Η πιθανότητα αυτή έφθινε από την τιμή 0.7 στην 0.1 γραμμικά στις πρώτες 500 γενιές, και κρατιόταν σταθερή για την υπόλοιπη εκπαίδευση.

Παρακάτω παρουσιάζεται το διάγραμμα της απόδοσης του καλύτερου πράκτορα ανά γενιά , για 920 γενιές σε επίπεδο δυσκολίας 2.



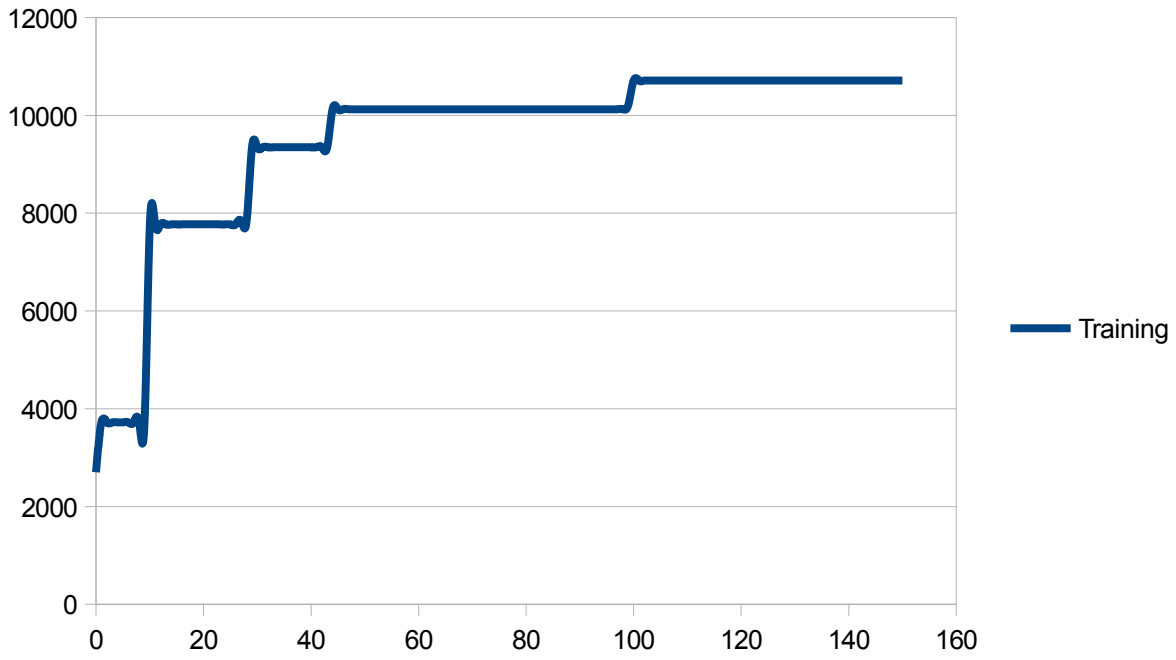
Εικόνα 6.31 : Απόδοση καλύτερου νευρωνικού χειριστή ανά γενιά σε επίπεδο δυσκολίας 2

Με αυτήν την αλλαγή στην μετάλλαξη κατάφερε να ξεπεραστούν τυχόν τοπικά μέγιστα και να εξελιχθεί ένας πράκτορας που μπορεί να απόδοση εξαιρετικά καλά σε αυτό το επίπεδο δυσκολίας 2. Πάραυτα δεν παρουσίασε καμία βελτίωση στην ικανότητα γενίκευσης του.

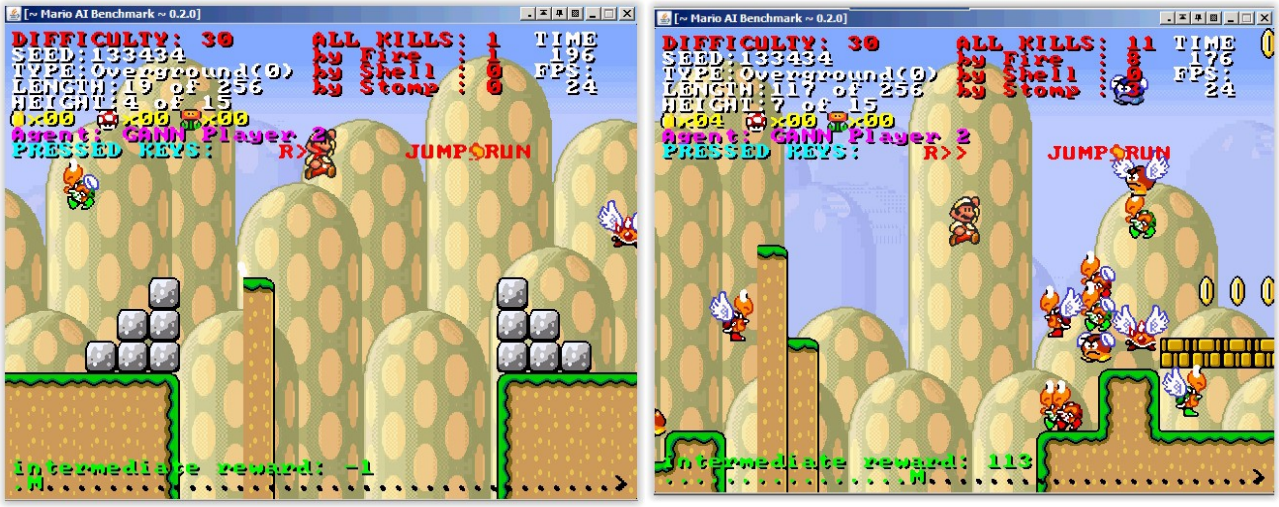
6.1.9 Πείραμα 9 : Επιχειρώντας επίπεδα δυσκολίας 5 & 30

Σε αυτό το πείραμα κυρίως εξετάστηκαν πολύ δύσκολες μορφολογικά πίστες (με παράξενα κενά). Επίσης επιχειρήθηκε μία προσπάθεια εκπαίδευσης για 150 γενιές σε ένα επίπεδο δυσκολίας 5.

- Δοκιμάστηκε να εξελιχθεί από τον πληθυσμό βάσης του πειράματος 4 ένας πράκτορας που να μπορεί να περάσει μία πίστα επιπέδου 30. Αυτό που θέλαμε να δούμε είναι εάν μπορεί να προσαρμοστεί στην δυσκολία της τοπολογίας της πίστας, έτσι ο Mario κατά την διάρκεια της εκπαίδευσης και της αξιολόγησης ήτανε ανίκητος (δεν δεχότανε ζημιά από εχθρούς). Η διάρκεια της εξέλιξης ήτανε 150 γενιές.



Εικόνα 6.32 : Απόδοση καλύτερου νευρωνικού χειριστή ανά γενιά σε επίπεδο δυσκολίας 30

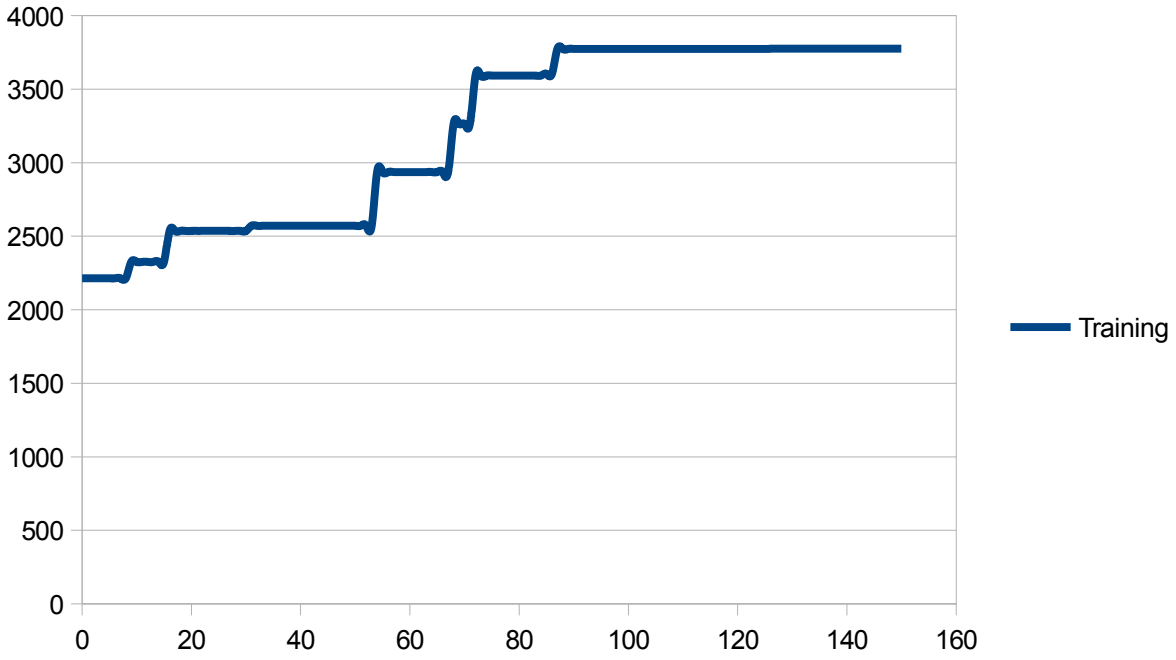


Εικόνα 6.33 : Στιγμιότυπο από το παίξιμο στην πίστα δυσκολίας 30 που εκπαιδευόταν ο πράκτορας

Εικόνα 6.34 : Στιγμιότυπο από το παίξιμο στην πίστα δυσκολίας 30 που εκπαιδευόταν ο πράκτορας

Παρατηρούμε ότι μέσα σε 33 γενιές έχει εξελιχθεί ένας πράκτορας που περνάει το επίπεδο. Επιπλέον συνεχίζει και βελτιώνεται σκοτώνοντας περισσότερους εχθρούς και στοιχεία του περιβάλλοντος (όπως κανονόμπαλες).

- Δοκιμάστηκε να εξελιχθεί από τον πληθυσμό βάσης του πειράματος 4 ένας πράκτορας που να μπορεί να περάσει μία πίστα επιπέδου 5. Η διάρκεια της εξέλιξης ήτανε 150 γενιές.



Εικόνα 6.35 : Απόδοση καλύτερου νευρωνικού χειριστή ανά γενιά σε επίπεδο δυσκολίας 5

Βλέπουμε ότι παρά την μεγάλη δυσκολία , υπάρχει εξέλιξη ακόμα και σε 150 γενιές (θυμίζετε ότι χρειάστηκε πάνω από 500 για να εξελιχθεί κάποιος χειριστής που να περνάει ένα επίπεδο δυσκολίας 2). Σίγουρα δεν πέρασε την πίστα , άλλα κατάφερε να σκοράρει αρκετά ψηλότερα από έναν απλό ανθρώπινο χειριστή σε αυτό το επίπεδο.

Παρακάτω βλέπουμε ένα στιγμιότυπο τις πίστας :

Εικόνα 6.36 : Στιγμιότυπο από το παίξιμο στην πίστα δυσκολίας 5 που εκπαιδευόταν ο πράκτορας

ΚΕΦΑΛΑΙΟ 7

7 Συμπεράσματα και μελλοντικές επεκτάσεις

7.1 Συμπεράσματα

Είδαμε από τα αποτελέσματα ότι ο ΓΑ (Γενετικός Αλγόριθμος) με τα ΤΝΝ (Τεχνητά Νευρωνικά Δίκτυα) είναι μία καλή προσέγγιση για να βρεθεί μια καλή λύση που ταιριάζει σε ένα συγκεκριμένο τύπο επιπέδου ή προβλήματος. Δοσμένων κατάλληλων συνθηκών (π.χ. ένα επίπεδο μόνο με κενά, ένα επίπεδο μόνο με εχθρούς που πατιούνται κλπ.) μπορεί να εκπαιδευτεί ένας χειριστής που γίνεται αρκετά γρήγορα καλός στο να περνάει το επίπεδο. Ακόμα και σε συνδυαστικά επίπεδα μικρής δυσκολίας, με την πάροδο τον γενεών η καμπύλη καταλληλότητας αυξανόταν και εξελισσόταν ένας πράκτορας που συμπεριφερόταν καλύτερα από κάποιον τυχαίο πράκτορα (RandomAgent) ή κάποιον πράκτορα που προχωράει μπροστά με έξυπνο τρόπο (ForwardAgent / ForwardJumpingAgent).

Παρατηρούνται όμως προβλήματα στην γενίκευση. Η γνώση πού αποκτάει ο χειριστής είναι στοχευμένη στο να αυξήσει την καταλληλότητα του στο συγκεκριμένο επίπεδο ή περιβάλλον. Αυτό δεν σημαίνει ότι η γνώση αυτή μπορεί να χρησιμοποιηθεί σε άλλα περιβάλλοντα, ακόμα και ίδιας φύσεως. Αυτό φαίνεται και από τα σετ εκπαίδευσης όπου ο καλύτερος χειριστής μαθαίνει να εφαρμόζει και να προσαρμόζει την γνώση που έχει για την συγκεκριμένη πίστα αυξάνοντας έτσι “γρήγορα” την τιμή της καταλληλότητας του αλλά χωρίς να βελτιώνει την απόδοση του στο σετ δοκιμής.

Όταν καλείται να εκπαιδευτεί σε επίπεδα που συνδυάζουν πολλά στοιχεία υπάρχει ο κίνδυνος να ξεχάσει ότι ήδη έχει μάθει από προηγούμενα επίπεδα στον απώτερο σκοπό του να μάθει να λειτουργεί στα καινούργια επίπεδα. Επειδή η διαδικασία επιλογής των ατόμων και οι εξελικτική διαδικασία (μετάλλαξη & διασταύρωση) είναι αυτοματοποιημένες και τυχαίες δεν μπορεί να βελτιωθεί πολύ η δυνατότητα γενίκευσης των χειριστών. Δοσμένου αρκετού χρόνου θα μπορούσε ενδεχομένως να εξελιχθεί ένας χειριστείς ικανός να γενικεύει την γνώση που μαθαίνει, αλλά αυτό θα έπαιρνε πολύ χρόνο. Μία πιο ρεαλιστική προσέγγιση της συγκεκριμένης εφαρμογής του γενετικού αλγορίθμου θα μπορούσε να ήταν η εξέλιξη ενός ήδη εκπαιδευμένου χειριστή (και όχι τυχαίου). Έτσι στα βάρη του ΝΝ θα ήταν ήδη αποθηκευμένη γνώση για

διάφορες καταστάσεις και όλη η διαδικασία εξέλιξης θα εστίαζε στο να βρει καινούργιες πιο ενδιαφέροντες λύσεις. Αυτό θα είχε ως αποτέλεσμα την μείωση του χρόνου εκπαίδευσης μέχρι να φτάσει στο σημείο ενός "καλού" πράκτορα, και την εξέλιξή του από αυτό το σημείο.

Ένας άλλος παράγοντας που φαίνεται να επηρεάζει την ικανότητα γενίκευσης είναι το καθεστώς εκπαίδευσης, κάτι που μελετήθηκε στα πλαίσια της διπλωματικής. Συνδυάζοντας πολλά επίπεδα στην εκπαίδευση του κάθε πράκτορα και αποτιμώντας την καταλληλότητα του κάθε χειριστή ως τον μέσο όρο της επίδοσης του σε αυτά φαίνεται να καλυτερεύουν την ικανότητα γενίκευσης.

Ένας παράγοντας που θέτει κατώφλι στην δυνατότητα εξέλιξης κάποιου πιο ικανού χειριστή είναι η μορφή των νευρώνων εισόδου. Η επιλογές που μελετήθηκαν δεν προσφέρονται για την μοντελοποίηση καταστάσεων σε επίπεδα με πολλούς εχθρούς. Ο χωρισμός του οπτικού πεδίου σε ζώνες όπως κάνανε οι Tonder & Olsen (2013) θα πρόσθετε μία επιπλέον διαίσθηση για το πότε ο Mario πρέπει να προχωρήσει / εκτελέσει άλμα / πυροβολήσει κλπ.

Κλείνοντας αξίζει να σημειωθεί το ιδιαίτερο ενδιαφέρον που παρουσιάζει η χρήση της Νευροεξέλιξης. Δείχθηκε ότι και στην πιο απλή μορφή της (εξελίσσοντας τα βάρη ενός νευρωνικού δικτύου με έναν απλό Γενετικό Αλγόριθμο με σταθερή την αρχιτεκτονική του δικτύου) εξελίχτηκε ένας χειριστής με καλύτερες επιδόσεις από τον τυχαίο πράκτορα και συγκρινόμενο με αρχάριο παίχτη στην γενικευμένη μορφή του. Συγκεκριμένα στα επίπεδα εκπαίδευσης είχε επιδόσεις καλού ανθρώπινου παίχτη. Η χρήση των παιχνιδιών ως πεδίων δοκιμών είναι επίσης ένας πολλά υποσχόμενος κλάδος που έμμεσα παιχνιδοποιεί (gamifies) την ερεύνα πάνω στους αυτόνομους πράκτορες (και όχι μόνο) και την χρήση διαφόρων τεχνικών τεχνητής και υπολογιστικής νοημοσύνης, προάγοντας έτσι την έρευνα σε ένα διασκεδαστικό περιβάλλον.

7.2 Μελλοντικές επεκτάσεις

Περαιτέρω μελέτη του συγκεκριμένου προβλήματος θεωρείται πεπερασμένη σύμφωνα με όλη την βιβλιογραφία και το γεγονός ότι ο διαγωνισμός MarioAI έχει πλέον καταργηθεί.

Αυτό που παρουσιάζει μεγάλο ενδιαφέρον είναι η χρήση νευροεξελικτικών μεθόδων στο General Video Game Playing.

Ο διαγωνισμός General Video Game Playing διεξάχθηκε πρώτη φορά στο συνέδριο της IEEE για την "Υπολογιστική Νοημοσύνη και Παιχνίδια" το 2014. Ο διαγωνισμός παρουσιάζει την πρόκληση της δημιουργίας ελεγκτών για γενικευμένα παιχνίδια, όπου ένας πράκτορας θα πρέπει να μπορεί να είναι σε θέση να μάθει να παίζει πολλά διαφορετικά παιχνίδια, αρκετά από τα οποία θα είναι άγνωστα προς τους συμμετέχοντες κατά την διάρκεια υποβολής των ελεγκτών τους. Η δοκιμή αυτή μπορεί να θεωρηθεί ως προσέγγιση της "Γενικής Τεχνητής Νοημοσύνης" (General Artificial Intelligence), καθώς ο αριθμός των ευριστικών τεχνικών προσαρμοσμένος για συγκεκριμένα παιχνίδια θα πρέπει να είναι πολύ περιορισμένος.

Τα παιχνίδια που χρησιμοποιούνται είναι στοχαστικά σενάρια πραγματικού χρόνου (όπου ο διαθέσιμος χρόνος για να διαλέξει ο χειριστής την επόμενη κίνηση μετριέται σε χιλιοστά του δευτερολέπτου) με διαφορετικές συνθήκες νίκης, μηχανισμούς βαθμολόγησης, είδη από οπτικά

στοιχεία και διαθέσιμες για τον παίχτη κινήσεις. Υπεύθυνοι είναι οι πράκτορες να ανακαλύψουν τους μηχανισμούς του κάθε παιχνιδιού, τις απαιτήσεις για την απόκτηση υψηλής βαθμολογίας και τις προϋποθέσεις για να επιτευχθεί τελικά η νίκη. Περισσότερες πληροφορίες για τον πρώτο διαγωνισμό του 2014 μπορούν να βρεθούν στο άρθρο των Perez et al. (2015) καθώς και στον ιστότοπο του διαγωνισμού www.gvgai.net.

Για την δημιουργία των παιχνιδιών χρησιμοποιείται μία προσαρμογή σε java της Video Game Definition Language (VGDL) ονόματι java-vgdl. Η VGDL είναι μία γλώσσα σχεδιασμένη από τον Tom Shaul, αρχικά υλοποιημένη σε Python. Η VGDL περιγράφει παιχνίδια με έναν πολύ συνοπτικό τρόπο, κάνοντας κάθε παιχνίδι να καταλαμβάνει μερικές δεκάδες γραμμές απλού κειμένου. Ο ορισμός ενός παιχνιδιού χωρίζεσαι σε τέσσερα τμήματα.

Παρακάτω φαίνεται ο ορισμός του παιχνιδιού Sokoban

```
BasicGame frame_rate=30
    SpriteSet
        hole   > Immovable color=DARKBLUE img=hole
        avatar > MovingAvatar #cooldown=4
        box    > Passive img=box
    LevelMapping
        0 > hole
        1 > box
    InteractionSet
        avatar wall > stepBack
        box avatar  > bounceForward
        box wall    > undoAll
        box box     > undoAll
        box hole    > killSprite scoreChange=1
    TerminationSet
        SpriteCounter stype=box    limit=0 win=True
```

Εικόνα 7.1: Ορισμός του παιχνιδιού Sokoban σε VGDL

- SpriteSet : ορίζει όλα τα διαθέσιμα αντικείμενα για το παιχνίδι, συμπεριλαμβανομένων των παραμέτρων αυτών καθώς και τις οπτικές ρυθμίσεις αναπαράστασης.
- LevelMapping : ορίζει τις σχέσεις μεταξύ χαρακτήρων, ορισμένων στους ορισμούς των επιπέδων, και των διαθέσιμων αντικειμένων.
- InteractionSet : Καθορίζει τι γεγονότα συμβαίνουν όταν δύο αντικείμενα συγκρούονται στο παιχνίδια
- TerminationSet : Καθορίζει τις συνθήκες τερματισμού του παιχνιδιού, που δείχνουν και αν ο παίχτης νίκησε ή έχασε.

Βιβλιογραφία

1. Agapitos, A., Togelius, J., Lucas, S.M., Schmidhuber, J., Konstantinidis, A., (2008), "Genereting diverse opponents with multiobjective neuroevolution", In Computational Intelligence and Games, CIG 08, IEEE Symposium On, IEEE, p. 135-142
2. Buk, Z., Koutnik, J., Snorek, M., (2009), “NEAT in HyperNEAT substituted with genetic programming", in Proceedings of the International Conference on Adaptive and Natural Computing Algorithms (ICANNGA)
3. Cardamone, L., Loiacono, D, Lanzi, P.L., (2009), "Evolving competitive car controllers for racing games with neuroevolution", in Proceedings of the 11th Annual conference on Genetic and Evolutionary Computation(GECCO '09), NY, USA, ACM, p. 1179-1186
4. Doncieux, S, Bredeche, N, Mouret, J-B, Eiben, AE, (2015), "Evolutionary robotics: what, why, and where to", Front, Robot, AI 2:4, doi: 10.3389/frobt.2015.00004, διαθέσιμο στο: http://dx.doi.org/10.3389/frobt.2015.00004
5. Eiben, A.E., Smith, James E. (2003), "Introduction to Evolutionary Computing", Natural Computing Series, Springer-Verlag Berlin Heidelberg
6. Gibney, E., (2016), “Google masters Go” [article], Nature, Volume 529, p. 445-446
7. Gomez, F., Schmidhuber, J., Miikkulainen, R., (2008), "Accelerated neural evolution through cooperatively coevolved synapses", The Journal of Machine Learning Research, p. 937–965
8. Hastings, E.J., Guha, R.K., Stanley, K.O., (2009), "Automatic content generation in the galactic arms race video game", Computational Intelligence and AI in Games, IEEE Transactions on, Volume 1, Issue 4, IEEE,p. 245-263
9. Hingston, P., (2010), "A new design for a Turing Test for Bots", Computational Intelligence and Games (CIG), 2010 IEEE Symposium on, IEEE, p. 245-250
10. Igel, Christian, Sendhoff, Bernhard, (2008), "Genesis of organic computing systems: Coupling evolution and learning", In R. P. Würtz, editor, "Organic Computing. Understanding Complex Systems", chapter 7, Springer-Verlag Berlin Heidelberg
11. Jørgensen, L. D., Sandberg, T. W.,(2009), "Playing Mario using advanced AI techniques"[online], IT University of Copenhagen, Available at: http://twsandberg.dk/media/4615/playing%20mario%20using%20advanced%20ai%20techniques.pdf, Accessed on: 16.10.2013)
12. Karakovskiy, S., Togelius, J.,(2012), "The Mario AI Benchmark and Competitions", IEEE Transactions on Computational Intelligence and AI in Games (TCIAG), volume 4 issue 1, p. 55-67

13. Loiacono, D., Lanzi, P.L., Togelius, J., Onieva, E., Pelta, D.A. et al., (2010), "The 2009 Simulated Car Racing Championship", IEEE Transactions on Computational Intelligence and AI in Games, Volume 2,Issue 2, p. 131-147
14. Løland, Karl Syvert, (2008), "Intelligent agents in computer games", Norwegian University of Science and Technology, Department of Computer and Information Science, Master's Thesis
15. Lu, F., Yamamoto, K, Nomura, L.H., Mizuno, S., Lee, Y.M., Thawonmas, R., (2013), "Fighting Game Artificial Intelligence Competition Platform", in the Proc. of the 2nd IEEE Global Conference on Consumer Electronics(GCCE 2013), Tokyo, Japan, p. 320-323
16. Lucas, S.M., (2005), "Evolving a Neural Network Location Evaluator to Play Ms. Pac-Man", in Proceedings of the 2005 IEEE Symposium on Computational Intelligence and Games (CIG 2005), IEEE, p. 203-210
17. Nareyek, Alexander, (2000), "Review: Intelligent Agents for Computer Games", Conference: Computer and Games, Second International Conference, CG 2000, Hamamatsu, Japan, Revised Papers, DOI: 10.1007/3-540-45579-5_28
18. Newborn, M., (1997), “Kasparov Versus Deep Blue: Computer Chess Comes of Age”, Springer-Verlag
19. Onta˜non, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D. et al., (2013), "A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft", IEEE Transactions on Computational Intelligence and AI in games,IEEE Computational Intelligence Society, Volume 5, Issue 4, pp.1-19. <hal-00871001>
20. Perez, D., Samothrakis, S., Togelius, J., Schaul, T., Lucas, S., Couetoux, A., Lee, J., Lim, C.-U., and Thompson, T., (2015), “The 2014 General Video Game Playing Competition”, IEEE Transactions on Computational Intelligence and AI in Games, DOI: 10.1109/TCIAIG.2015.2402393
21. Renz, J., (2015), " AIBIRDS: The Angry Birds Artificial Intelligence Competition", in the Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI, p. 4326-4327
22. Risi, S., Lehman, J., D'Ambrosio, D.B., Hall, R., Stanley, K.O., (2012), "Combining search-based procedural content generation and social gaming in the Petal video game", In Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)
23. Risi, S., Lehman, J., D’Ambrosio, D.B., Hall, R., Stanley, K.O., (2015),"Petalz: Search-based procedural content generation for the casual gamer", Computational Intelligence and AI in Games, IEEE Transactions on, Volume PP, Issue 99, IEEE, p. 1–1, ISSN 1943-068X. doi: 10.1109/TCIAIG.2015.2416206.
24. Rohlfshagen, P., Lucas, S.M., (2011), "Ms Pac-Man versus Ghost Team CEC 2011 Competition", in Evolutionary Computation (CEC), 2011 IEEE Congress on, IEEE, p. 70-77
25. Rumelhart, D.E., Hinton, G.E., Williams, R.J., (1986), "Learning representations by back-propagation of errors", Nature, Volume 323, p. 533–536
26. Russel, Stuart, Norvig, Peter (2003) (2005 Ελληνική Έκδοση), "Τεχνητή Νοημοσύνη, Μία σύγχρονη προσέγγιση", Κλειδάριθμος, Αθήνα, σελ. 64-88.

27. Schaul, T., (2013), “A Video Game Description Language for Model-based or Interactive Learning”, in Proceedings of the IEEE Conference on Computational Intelligence in Games, Niagara Falls: IEEE Press, p. 193–200.
28. Stanley, K.O., Bryant, B.D., Miikkulainen, R., (2005), "Real-time neuroevolution in the NERO video game",Evolutionary Computation, IEEE Transactions on, Volume 9, Issue 6, IEEE p.653-668
29. Stone, Peter, (2007), "Learning and multiagent reasoning for autonomous agents", Proceedings of the 20th International Joint Conference on Artificial Intelligence, p. 13-30
30. Suganya, K., (2014), "A Review of Intelligent Agents", International Journal of Engineering Research and General Science Volume 2, Issue 2, ISSN 2091-2730, p. 112-117
31. Togelius, J., Karakaovskiy, S., Koutnik, J., Schmidhuber,J,(2009), “Super mario evolution,”, in Proceedings of IEEE Symposium on Computational Intelligence and Games (CIG)
32. Togelius, J., Shaker, N., Karakovskiy, S., Yannakakis, G.N., (2013), "The Mario AI Championship 2009-2012", in AI MAGAZINE, Volume 34, Issue 3, p. 89-92, DOI: http://dx.doi.org/10.1609/aimag.v34i3.2492
33. Tønder, L. S., Olsen, O.-P., (2013), "Multi-objective Neuroevolution in Super Mario Bros", Norwegian University of Science and Technology, Department of Computer and Information Science, Master's Thesis
34. Turing, A.M., (1950), “Computing machinery and intelligence”, Mind, Volume 59, p.433-460
35. van Hoorn, N., Togelius, J., Wierstra, D., Schmidhuber J., (2009), "Robust player imitation using multiobjective evolution", In Evolutionary Computation, CEC '09, IEEE Congress On, IEEE, p. 652-659
36. Whitley, D., Schaffer, J. D., Eshelman, L. J.,(1992), “Combinations of genetic algorithms and neural networks: A survey of the State of the Art”, in Proceedings of the International Workshop on Combinations of genetic algorithms and neural networks, Baltimore, IEEE, p. 1-37
37. Yannakakis, G.N., Togelius, J., (2014), "A Panorama of Artificial and Computational Intelligence in Games", in IEEE Transactions on Computational Intelligence in Games, Volume 7, Issue 4, IEEE, p.317-335
38. Yao, Xin, (1999) "Evolving artificial neural networks", Proceedings of the IEEE, Volume 87, Issue 9, p. 1423–1447
39. Γεωργούλη, Α., (2016), "Νοήμονες Πράκτορες" [Κεφάλαιο Συγγράμματος], Στο "ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ" [ηλεκτρ. βιβλ.], Αθήνα:Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών, κεφ 6, Διαθέσιμο στο: http://hdl.handle.net/11419/3384
40. Χατζηδημητρίου, Κυριάκος, (2012), "Μηχανισμοί ενισχυτικής μάθησης και εξελικτικής υπολογιστικής για αυτόνομους πράκτορες, ΑΠΘ, Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών, Διδακτορική διατριβή