Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Ανάπτυξη ευφυών πρακτόρων ηλεκτρονικών παιχνιδιών με τη χρήση Γενετικών Αλγορίθμων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΣΤΑΣΙΟΣ Δ. ΠΑΠΑΓΙΑΝΝΗΣ

Επιβλέπων Καθηγητής : Ανδρέας-Γεώργιος Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.

Επιβλέπων : Γεώργιος Αλεξανδρίδης
Διδάκτορας Ε.Μ.Π.

Αθήνα, Οκτώβριος 2017

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Ανάπτυξη ευφυών πρακτόρων ηλεκτρονικών παιχνιδιών με τη χρήση Γενετικών Αλγορίθμων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΣΤΑΣΙΟΣ Δ. ΠΑΠΑΓΙΑΝΝΗΣ

Επιβλέπων Καθηγητής : Ανδρέας-Γεώργιος Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.

Επιβλέπων : Γεώργιος Αλεξανδρίδης
Διδάκτορας Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 30η Οκτωβρίου 2017.

.....	.....	.....
Ανδρέας-Γεώργιος Σταφυλοπάτης Καθηγητής Ε.Μ.Π.	Παναγιώτης Τσανάκας Καθηγητής Ε.Μ.Π.	Γεώργιος Στάμου Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2017

Αναστάσιος Δ. Παπαγιάννης

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

Περίληψη

Την τελευταία δεκαετία, τα ηλεκτρονικά παιχνίδια αποτελούν ένα ιδιαίτερα δημοφιλές πεδίο εφαρμογής των σύγχρονων μεθόδων του ευρύτερου κλάδου της τεχνητής νοημοσύνης. Καθώς οι απαιτήσεις για ευφυή συμπεριφορά στα παιχνίδια αυξάνονται συνεχώς, η εφαρμογή αυτών των τεχνικών ξεπερνά το πλαίσιο δοκιμαστικού χαρακτήρα και αποκτά πρακτικούς σκοπούς. Ευριστικές μέθοδοι, νευρωνικά δίκτυα, δέντρα αναζήτησης και εξελικτικοί αλγόριθμοι είναι οι σημαντικότερες τεχνικές που έχουν χρησιμοποιηθεί μέχρι σήμερα στην προσπάθεια ανάπτυξης νοημοσύνης, ικανής να είναι ανταγωνιστική για τους ανθρώπους.

Στόχος της συγκεκριμένης διπλωματικής εργασίας είναι η μελέτη της χρήσης εξελικτικών αλγορίθμων, συγκεκριμένα των γενετικών αλγορίθμων, για τη δημιουργία πρακτόρων ελέγχου του βασικού χαρακτήρα σε ηλεκτρονικά παιχνίδια. Οι Γενετικοί Αλγόριθμοι βασίζονται στη Δαρβινική Θεωρία της εξέλιξης, προσομοιώνοντας στον υπολογιστή τη λογική της αναπαραγωγής και της επιβίωσης των ικανότερων ατόμων, όπως συμβαίνει στη φύση. Πρόκειται για μία μέθοδο δοκιμής-σφάλματος μέχρι την εύρεση ικανοποιητικής λύσης για ένα πρόβλημα. Στην παρούσα εργασία, εξετάζεται μία εναλλακτική προσέγγιση της εφαρμογής ΓΑ σε ηλεκτρονικά παιχνίδια, όσον αφορά την κωδικοποίηση του προβλήματος. Η βασική ιδέα είναι η κωδικοποίηση των καταστάσεων του κόσμου του παιχνιδιού, σε αντίθεση με τις κλασσικές τεχνικές στις οποίες κωδικοποιείται η ακολουθία των κινήσεων.

Οι πράκτορες υλοποιήθηκαν στην πλατφόρμα του Γενικού Διαγωνισμού Ηλεκτρονικών Παιχνιδιών με χρήση Τεχνητής Νοημοσύνης (GVG-AI), η οποία διαθέτει το κατάλληλο υπόβαθρο για την ανάπτυξη τέτοιων εφαρμογών. Σχεδιάστηκαν τρεις διαφορετικές εκδοχές ενός πράκτορα για το παιχνίδι Zelda που περιλαμβάνεται στη συγκεκριμένη πλατφόρμα. Σε όλες τις εκδοχές στόχος είναι η εύρεση της βέλτιστης κίνησης ανάλογα με την κατάσταση του κόσμου, διαφέρουν όμως ως προς τα σημεία του κόσμου που λαμβάνονται υπόψιν. Αρχικά τα σημεία εξετάστηκαν ένα προς ένα ενώ στη συνέχεια δοκιμάστηκε η κωδικοποίηση τους ανά ομάδες.

Στο τέλος αξιολογούνται τα αποτελέσματα και παρουσιάζονται μελλοντικές κατευθύνσεις για την επέκταση του ίδιου πράκτορα σε ένα δεύτερο παιχνίδι και για τα περιθώρια γενίκευσης αυτής της τεχνικής.

Λέξεις κλειδιά

Ηλεκτρονικά Παιχνίδια, Τεχνητή Νοημοσύνη, Γενετικοί Αλγόριθμοι, Κωδικοποίηση Καταστάσεων, GVG-AI

Abstract

During the last decade, video games have become a very popular field of application of modern methods in the broader Artificial Intelligence industry. As the demand for intelligent behaviour in games is constantly increasing, the application of these techniques goes beyond the testing framework and pursues practical purposes. Heuristics, neural networks, search trees and evolutionary algorithms have been the most significant techniques used so far in an attempt to develop intelligence competitive for humans.

The purpose of this diploma thesis is to study the use of evolutionary algorithms, namely genetic algorithms, in the creation of agents that will control the avatar in video games. Genetic Algorithms are based on Darwin's Theory of Evolution, simulating on the copmuter the reproduction and survival of the most capable individuals, as is the case in nature. This is a trial and error method, which terminates when a satisfactory solution is achieved. The present thesis sets out to investigate an alternative approach to the application of genetic algorithms on video games concerning problem encoding. The main idea is to encode the state of the world of the game, instead of the the sequence of actions which is encoded in most of the classic approaches.

The agents were implemented on the General Video Game Artificial Intelligence (GVG-AI) Competition's framework, which provides a suitable background for the development of this kind of applications. The implementation consists of three different versions of an agent, designed to control the avatar in Zelda, a game which is available through the particular framework. The goal of each agent version is to select the optimum action according to the world's state. Each version differs, however, in the world's blocks taken into consideration. Initially all the blocks were tested separately and subsequently they were encoded and tested in groups.

Finally the results are evaluated and future directions for extending the same agent to a second game are presented along with the scope for generalization of the described technique.

Key words

Video Games, Artificial Intelligence, Genetic Algorithms, State Encoding, GVG-AI

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε στο πλαίσιο του προπτυχιακού προγράμματος σπουδών της Σχολής Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου και σηματοδοτεί την ολοκλήρωση των σπουδών μου ενώ συγχρόνως αποτελεί το ερέθισμα για περαιτέρω έρευνα στο συγκεκριμένο αντικείμενο.

Πριν από οποιαδήποτε αναφορά στη διαδικασία που ακολουθήθηκε και στα αποτελέσματα που προέκυψαν, θα ήθελα να ευχαριστήσω θερμά τους ανθρώπους με τους οποίους συνεργάστηκα και συνέβαλαν στην ολοκλήρωση αυτής της εργασίας.

Κατ' αρχάς απευθύνω τις ευχαριστίες μου στον επιβλέποντα κ. Ανδρέα-Γεώργιο Σταφυλοπάτη, Καθηγητή Ε.Μ.Π, για την δυνατότητα που μου προσέφερε να εργαστώ σε ένα αντικείμενο ιδιαίτερα ελκυστικό για μένα και να διευρύνω τις επιστημονικές μου γνώσεις. Παράλληλα θα ήθελα να ευχαριστήσω τους κ. Γεώργιο Στάμου, Αναπληρωτή Καθηγητή Ε.Μ.Π και κ. Παναγιώτη Τσανάκα, Καθηγητή Ε.Μ.Π που με τίμησαν με την παρουσία τους στην επιτροπή εξέτασης της διπλωματικής εργασίας.

Επίσης οφείλω ιδιαίτερες ευχαριστίες στον κ. Γεώργιο Αλεξανδρίδη, Διδάκτορα Ε.Μ.Π. για το χρόνο που αφιέρωσε και την θεμελιώδη του συνεισφορά στην εκπόνηση της συγκεκριμένης εργασίας. Τόσο η επιστημονική όσο και η πνευματική του στήριξη ήταν ιδιαίτερα σημαντικές για εμένα κατά τη διάρκεια αυτής της πορείας και η καθοδήγησή του βοήθησε τα μέγιστα στην επίτευξη ενός πολύ σημαντικού για εμένα στόχου. Η εμπειρία και οι γνώσεις του στάθηκαν καθοριστικές και η συνεργασία μας θεωρώ πως ήταν άκρως επιτυχημένη και εποικοδομητική.

Τέλος, με εξίσου μεγάλη θέρμη θέλω να αναφερθώ και να ευχαριστήσω τους φίλους και την οικογένεια μου, που ήταν δίπλα μου σε όλη τη διάρκεια της ακαδημαϊκής μου πορείας, ο καθένας με τον ξεχωριστό του τρόπο. Για την αμέριστη υποστήριξή τους θέλω να εκφράσω την ευγνωμοσύνη μου στους γονείς μου, οι οποίοι ήταν κοντά μου και με στήριξαν όλα αυτά τα χρόνια και να ευχηθώ κάθε επιτυχία στα αδέρφια μου στις νέες προκλήσεις που τους παρουσιάζονται.

Αναστάσιος Δ. Παπαγιάννης,

Αθήνα, 30η Οκτωβρίου 2017

Η εργασία αυτή είναι επίσης διαθέσιμη ως Τεχνική Αναφορά , Εθνικό Μετσόβιο Πολυτεχνείο, Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών, Εργαστήριο Τεχνολογίας Λογισμικού, Οκτώβριος 2017.

URL: `http://www.softlab.ntua.gr/techrep/`
FTP: `ftp://ftp.softlab.ntua.gr/pub/techrep/`

Περιεχόμενα

Κατάλογος πινάκων

Κατάλογος σχημάτων

Κεφάλαιο 1

Εισαγωγή

Η *Τεχνητή Νοημοσύνη* (Artificial Intelligence – AI) είναι ο κλάδος της πληροφορικής που έχει ως αντικείμενο την ανάπτυξη ευφυών συμπεριφορών, με δυνατότητες μάθησης και εκπαίδευσης, παρόμοιες με την ανθρώπινη συμπεριφορά.

Η εφαρμογή της στον τομέα των ηλεκτρονικών παιχνιδιών (*Game AI*), αποτελεί υποκατηγορία του ευρύτερου πεδίου της, παρότι υπήρχε αρχικά η άποψη ότι πρόκειται για δύο ξεχωριστούς κλάδους με ορισμένα κοινά σημεία [Yann12]. Αυτή η άποψη στηρίχθηκε στο γεγονός ότι στις πρώτες απόπειρες εισαγωγής τεχνητής νοημοσύνης σε παιχνίδια, χρησιμοποιούνταν κυρίως προκαθορισμένες συμπεριφορές ανάλογα με τις ενέργειες των παιχτών και κάποιες απλές ευριστικές μέθοδοι, που απείχαν αρκετά από την έννοια της "πραγματικής" τεχνητής νοημοσύνης. Ωστόσο, τα τελευταία χρόνια έχει υπάρξει σημαντικό ενδιαφέρον από ερευνητές προς αυτή την κατεύθυνση, καθιστώντας πιο ευδιάκριτη την σύνδεση μεταξύ των δύο κλάδων.

Τα πρώτα παιχνίδια στα οποία εφαρμόστηκε τεχνητή νοημοσύνη χρονολογούνται ήδη από τις αρχές της δεκαετίας του 1950. Πιο συγκεκριμένα, το 1951, υλοποιήθηκαν οι πρώτοι *πράκτορες* (agents) για το παιχνίδι Nim, οι οποίοι ήταν ικανοί να νικήσουν παίχτες υψηλού επιπέδου. Παράλληλα, την ίδια εποχή αναπτύχθηκαν πράκτορες για το παιχνίδι checkers και για το σκάκι. Η συνέχεια της έρευνας σε αυτή την κατεύθυνση, οδήγησε το 1997 στην περίφημη νίκη του Deep Blue (υπολογιστή που αναπτύχθηκε από την IBM ειδικά για την επιλογή των κινήσεων σε μία παρτίδα σκακιού) επί του παγκόσμιου πρωταθλητή στο σκάκι Garry Kasparov [McCo04], κάνοντας με τον πλέον εμφατικό τρόπο ορατές τις δυνατότητες που προσφέρει η τεχνητή νοημοσύνη στα παιχνίδια.

Αρχικά, όλα τα ηλεκτρονικά παιχνίδια σχεδιάζονταν για δύο παίχτες, καθώς δεν υπήρχε η δυνατότητα ελέγχου του ενός εκ των δύο από τον υπολογιστή. Η επιλογή αυτή εμφανίστηκε για πρώτη φορά το 1974 στο παιχνίδι Speed Race της Taito και στα παιχνίδια Pursuit και Qwak της Atari. Στις δεκαετίες που ακολούθησαν, η ιδέα του ελέγχου χαρακτήρων από τον υπολογιστή (*Non-Player Characters-NPCs*), άρχισε να εξαπλώνεται σε ολοένα και περισσότερα παιχνίδια, με βασικότερα παραδείγματα το Space Invaders (1978) και το Pacman (1980).

Από τις αρχές της δεκαετίας του 1990 άρχισαν να χρησιμοποιούνται κλασσικές μέθοδοι της τεχνητής νοημοσύνης όπως *μηχανές πεπερασμένων καταστάσεων* (finite state machines), καθώς η τεχνική που χρησιμοποιούνταν κατά κύριο λόγο μέχρι τότε και βασιζόταν σε προκαθορισμένα *πρότυπα* (patterns) δεν ήταν αποτελεσματική στα πιο πολύπλοκα προβλήματα που παρουσίαζαν τα νεότερα παιχνίδια.

Πλέον, στα σύγχρονα παιχνίδια έχουν εφαρμοστεί πολλές εξελιγμένες τεχνικές τεχνητής νοημοσύνης, όπως δέντρα αναζήτησης, εξελικτικοί αλγόριθμοι, νευρωνικά δίκτυα κ.ά. Ωστόσο, ακόμα υπάρχουν αρκετά περιθώρια βελτίωσης, τα οποία μάλιστα με την πάροδο του χρόνου αυξάνονται μαζί με τις απαιτήσεις των παιχνιδιών για περισσότερες δυνατότητες. Αυτή η εκτίμηση επιβεβαιώνεται και από το γεγονός ότι σε πολλά παιχνίδια, προκειμένου να είναι ανταγωνιστικά για τον παίχτη, χρησιμοποιούνται τεχνικές που "κλέβουν" (*cheating*) [Scot02]. Με αυτό τον όρο περιγράφεται η πρόσβαση σε πληροφορίες που στην πραγματικότητα δεν θα έπρεπε να είναι διαθέσιμες σε ένα NPC, όπως για παράδειγμα η θέση του παίχτη στον κόσμο του παιχνιδιού όταν αυτός είναι κρυμμένος. Άλλες τέτοιες τεχνικές είναι η αύξηση των ικανοτήτων τους (πχ ταχύτητα) πέρα από τα φυσιολογικά όρια και η εμφάνισή τους σε ευνοϊκά σημεία της πίστας. Όλα αυτά προκύπτουν από την αδυναμία των γνωστών έως τώρα μεθόδων να δημιουργήσουν ανταγωνιστικούς NPCs για τους ανθρώπους, χωρίς τη χρήση

επιπλέον πληροφοριών.

Σήμερα, εκτός του ελέγχου των NPCs υπάρχουν πολλές άλλες προκλήσεις για την τεχνητή νοημοσύνη στα video games. Οι βασικότερες είναι η *πλοήγηση* (navigation) και η εύρεση διαδρομής, η προσαρμογή του επιπέδου δυσκολίας στις ικανότητες του χρήστη-παίχτη σε πραγματικό χρόνο και η *συλλογή χρήσιμων δεδομένων* (data mining) από τη συμπεριφορά των παιχτών σχετικά με τα σημεία του παιχνιδιού που απολαμβάνουν περισσότερο και τους λόγους για τους οποίους μπορεί να δυσαρεστηθούν ή να βαρεθούν ένα παιχνίδι. Τελευταία ιδιαίτερο ενδιαφέρον συγκεντρώνει επίσης, η ιδέα ανάπτυξης γεννητριών περιεχομένου υπό-συνθήκη. Αυτό συνεπάγεται τη συλλογή δεδομένων από την πορεία του χρήστη στο παιχνίδι και τη δημιουργία νέων στόχων ανάλογα με τη συμπεριφορά του, δηλαδή την προσαρμογή ουσιαστικά της εξέλιξης του παιχνιδιού στον κάθε χρήστη.

Άλλη μία πρόκληση που τα τελευταία χρόνια έχει αποκτήσει πιο επίσημο χαρακτήρα μέσω του διαγωνισμού General Video Game AI (GVG-AI), είναι η ανάπτυξη πρακτόρων που χρησιμοποιούν τεχνητή νοημοσύνη για τον έλεγχο των *χαρακτήρων* (avatars) σε περισσότερα από ένα ηλεκτρονικά παιχνίδια. Πρόκειται για μία προσπάθεια γενίκευσης της λειτουργίας των πρακτόρων, που θα μπορούσε στη συνέχεια να βρει πολλές εφαρμογές και σε άλλα πεδία εκτός των ηλεκτρονικών παιχνιδιών, όταν φτάσει στο επιθυμητό επίπεδο.

Ο GVG-AI προσφέρει μία *πλατφόρμα* (framework) για την ανάπτυξη πρακτόρων σε περισσότερα από 80 διαφορετικά video games παρόμοια με κλασσικά δημοφιλή παιχνίδια της δεκαετίας του 1980. Οι πράκτορες μπορούν να δοκιμαστούν σε αυτά τα παιχνίδια, τα περισσότερα εκ των οποίων είναι τύπου *NES* (Nintendo Entertainment System), και καλούνται να αποφασίσουν σε πραγματικό χρόνο την επόμενη κίνηση του avatar.

Το framework είναι υλοποιημένο στη γλώσσα προγραμματισμού Java και βασίζεται σε μία παραλλαγή της Video Game Description Language (VGDL) για Java (Java-VGDL). Η VGDL είναι μία γλώσσα υλοποιημένη σε Python, ειδικά για το σχεδιασμό παιχνιδιών τέτοιου τύπου, στην οποία όλες οι οντότητες αντιμετωπίζονται ως *αντικείμενα* (objects). Για τον ορισμό οποιουδήποτε παιχνιδιού σε VGDL απαιτείται ο προσδιορισμός των αρχείων *Περιγραφής Παιχνιδιού* (Game Description File) και *Περιγραφής Επιπέδου* (Level Description File).

Στη συγκεκριμένη διπλωματική εργασία, στόχος είναι η ανάπτυξη ενός πράκτορα ικανού να ελέγχει το avatar στο παιχνίδι Zelda, το οποίο είναι διαθέσιμο από το GVG-AI framework. Στο Zelda, ο παίχτης πρέπει αρχικά να βρει το κλειδί που είναι τοποθετημένο σε κάποιο σημείο της πίστας και στη συνέχεια αφού το πάρει να κατευθυνθεί προς την έξοδο-πόρτα, ελέγχοντας παράλληλα τις κινήσεις των εχθρών, τους οποίους πρέπει είτε να αποφεύγει είτε να εξολοθρεύσει.

Για την υλοποίηση του πράκτορα, επιλέχθηκε η χρήση γενετικών αλγορίθμων (ΓΑ). Οι ΓΑ είναι μία κατηγορία εξελικτικών αλγορίθμων που βασίζονται στη θεωρία της βιολογικής εξέλιξης. Οι παράμετροι του προβλήματος κωδικοποιούνται αρχικά σε χρωμοσώματα και στη συνέχεια δημιουργείται ένα σύνολο τυχαίων λύσεων που συνδυάζονται μεταξύ τους για την παραγωγή νέων. Έπειτα η διαδικασία της αναπαραγωγής επαναλαμβάνεται για αρκετές γενιές μέχρι να βρεθεί ικανοποιητική λύση για το πρόβλημα. Στην παρούσα εργασία, η ιδέα είναι να κωδικοποιηθούν μέσω των γενετικών αλγορίθμων οι διαφορετικές καταστάσεις του κόσμου του παιχνιδιού και με βάση αυτή την κωδικοποίηση να επιλέγει ο πράκτορας την βέλτιστη ενέργεια.

1.1 Δομή της Εργασίας

Η παρούσα διπλωματική εργασία έχει την παρακάτω δομή. Στην Εισαγωγή (1^{ο} Κεφάλαιο) αναφέρονται οι χρήσεις της τεχνητής νοημοσύνης στο πεδίο των ηλεκτρονικών παιχνιδιών και η εξέλιξή τους τις τελευταίες δεκαετίες. Επίσης περιγράφεται ο σκοπός της συγκεκριμένης εργασίας και η μέθοδος που εφαρμόστηκε περιληπτικά.

Στο 2^{ο} Κεφάλαιο περιγράφονται οι *ευφυείς πράκτορες* (intelligent agents) και τα βασικότερα στοιχεία τους. Ιδιαίτερη αναφορά γίνεται στα περιβάλλοντα των πρακτόρων, στις κατηγορίες τους, καθώς και στις σημαντικότερες εφαρμογές τους.

Στο 3^{ο} Κεφάλαιο περιγράφονται οι Γενετικοί Αλγόριθμοι και η γενική μέθοδος υλοποίησης τους.

Επίσης αναλύονται η δομή, οι τελεστές επιλογής και μετάλλαξης και τα κριτήρια τερματισμού. Τέλος παρουσιάζονται τα βασικά πλεονεκτήματά τους έναντι άλλων μεθόδων και το πεδίο εφαρμογής τους.

Στο 4ο Κεφάλαιο γίνεται η παρουσίαση της πλατφόρμας του διαγωνισμού GVG-AI που χρησιμοποιήθηκε στην πειραματική διαδικασία της εργασίας και του API που προσφέρει. Επίσης περιγράφεται ο τρόπος με τον οποίο υλοποιήθηκαν οι γενετικοί αλγόριθμοι στο συγκεκριμένο περιβάλλον και αναλύονται οι πράκτορες που αναπτύχθηκαν.

Στο 5ο Κεφάλαιο ορίζονται τα μέτρα απόδοσης των διαφορετικών πρακτόρων που υλοποιήθηκαν και δοκιμάστηκαν και παρουσιάζονται γραφικά τα αποτελέσματα για κάθε περίπτωση.

Τέλος, στο 6ο Κεφάλαιο αναφέρονται τα συμπεράσματα που προέκυψαν από τα αποτελέσματα της παρούσας εργασίας καθώς και κάποιες ιδέες για περαιτέρω ενασχόληση με την εφαρμογή γενετικών αλγορίθμων στο πεδίο των ηλεκτρονικών παιχνιδιών.

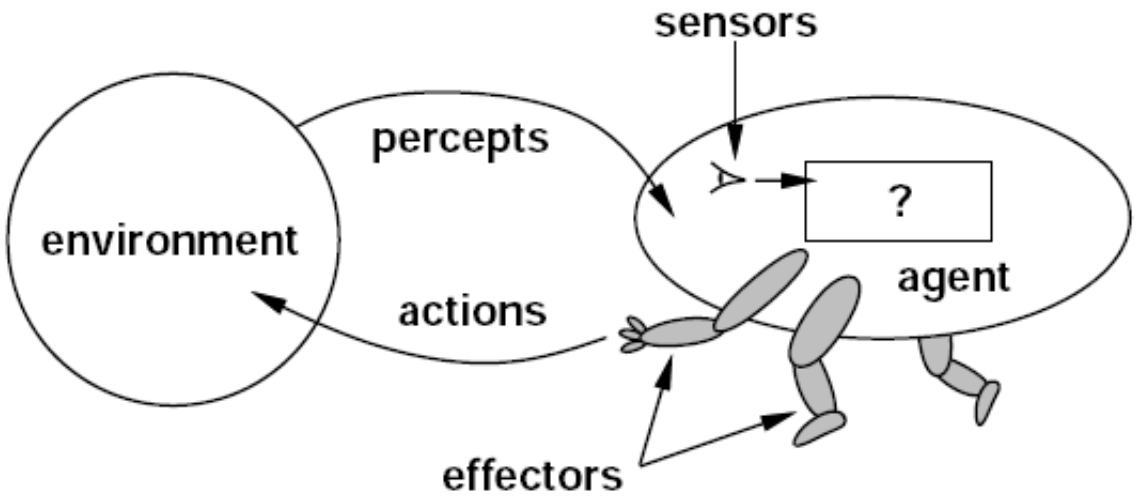
Κεφάλαιο 2

Ευφυείς Πράκτορες

2.1 Εισαγωγή

Στο πεδίο της *Τεχνητής Νοημοσύνης* (Artificial Intelligence – AI), ευφυές σύστημα είναι αυτό που επεξεργάζεται εσωτερική πληροφορία, προκειμένου να εκπληρώσει έναν στόχο [Mill]. Τέτοια μπορεί να είναι άνθρωποι ή ζώα αλλά και τεχνητά συστήματα όπως αισθητήρες και ρομπότ. Τα τελευταία χρόνια έχει αναπτυχθεί ιδιαίτερο ενδιαφέρον στην κατεύθυνση των τεχνητών ευφυών συστημάτων καθώς αποτελούν ένα πεδίο μελέτης με πολλές ανεξερεύνητες διαστάσεις. Μία κατηγορία ευφυών συστημάτων με πολύ σημαντική απήχηση είναι αυτή των *Αυτόνομων Τεχνητών Ευφυών Πρακτόρων* (Artificial Autonomous Intelligent Agents), οι οποίοι παρουσιάζουν πληθώρα εφαρμογών.

Στην ορολογία της τεχνητής νοημοσύνης, ο *Πράκτορας* (Agent) ορίζεται ως οποιαδήποτε οντότητα είναι ικανή να αντιλαμβάνεται το περιβάλλον μέσω *αισθητήρων* (sensors) και να επιδρά σε αυτό μέσω *ενεργοποιητών* (actuators) [Russ95]. *Ευφυής Πράκτορας* (Intelligent Agent - IA) είναι ένας πράκτορας ικανός να αποφασίσει ποιες ενέργειες πρέπει να κάνει, βασιζόμενος στην εμπειρία που έχει αποκτήσει [Mill].



Σχήμα 2.1: Σχηματική αναπαράσταση ενός Πράκτορα

Η *είσοδος* (input) που συλλέγει ο πράκτορας με τη χρήση των αισθητήρων του σε μία δεδομένη στιγμή καλείται *αντίληψη* (percept) του περιβάλλοντος. Για την συνολική πληροφορία που έχει συγκεντρώσει, χρησιμοποιείται ο όρος *αντιληπτική ακολουθία* (percept sequence) και είναι πρακτικά ό,τι χρειάζεται για να αποφασίσει τις επόμενες ενέργειες του.

Συνήθως ο πράκτορας αρχικοποιείται με μία προκαθορισμένη βασική γνώση για το περιβάλλον, η οποία θα τον βοηθήσει στα πρώτα στάδια που η αντιληπτική ακολουθία είναι σχετικά μικρή. Όσο όμως η πληροφορία που αντλεί από το περιβάλλον αυξάνεται θα πρέπει να είναι σε θέση να την αξιοποιεί για τις αποφάσεις του και να μην στηρίζεται μόνο στην αρχική γνώση που είχε. Αυτή είναι η έννοια της *αυτονομίας* (autonomy), η οποία με την πάροδο του χρόνου ενισχύεται, καθώς λαμβάνει

περισσότερες εισόδους.

Έχοντας κωδικοποιήσει την πληροφορία, ο πράκτορας χρειάζεται έναν τρόπο να την επεξεργαστεί ώστε να επιλέξει την κατάλληλη δράση. Μία τεχνική είναι ο σχηματισμός ενός πίνακα που θα περιγράφει την επιθυμητή ενέργεια για κάθε πιθανή αντιληπτική ακολουθία. Κάτι τέτοιο, προφανώς δεν είναι αποδοτικό για συστήματα με πολλές διαφορετικές εισόδους και καθώς η αντιληπτική ακολουθία αυξάνεται με την πάροδο του χρόνου, καθίσταται αδύνατο σε ένα πραγματικό σύστημα. Στη γενική περίπτωση, χρησιμοποιείται μία μέθοδος, η οποία μπορεί να αντιστοιχίσει μία οποιαδήποτε αντιληπτική ακολουθία σε μία *ενέργεια* (action). Αυτή η μέθοδος καλείται *συνάρτηση πράκτορα* (agent function) και είναι αυτή που ορίζει τη συμπεριφορά του στο περιβάλλον [Russ95]. Η συνάρτηση πράκτορα είναι μία μαθηματική περιγραφή της αντιστοίχισης και στην πράξη υλοποιείται με ένα *πρόγραμμα πράκτορα* (agent program) που αποτελεί συστατικό του στοιχείο.

Το πρόγραμμα είναι το ένα από τα δύο βασικά συστατικά της υλοποίησης ενός πράκτορα και εκτελείται συνεχώς σε κάποια υπολογιστική συσκευή. Η υπολογιστική συσκευή μαζί με το μηχανικό μέρος, που αποτελείται πρακτικά από τους αισθητήρες και τους ενεργοποιητές, συνιστούν την αρχιτεκτονική του, η οποία είναι το δεύτερο βασικό συστατικό.

Ο στόχος ενός ευφυούς πράκτορα είναι να μπορεί να συμπεριφέρεται με τέτοιο τρόπο, ώστε να προκαλεί τις επιθυμητές αλλαγές στο περιβάλλον του. Ένας πράκτορας, ο οποίος επιδρά επιτυχώς στο περιβάλλον κατ' αυτόν τον τρόπο ονομάζεται *Ορθολογιστικός* (Rational).

Η *λογικότητα* (rationality) ενός πράκτορα εξαρτάται από τις εξής παραμέτρους [Russ95]:

1. Το βαθμό επίδοσής του
2. Το σύνολο των δυνατών ενεργειών του
3. Την αντιληπτική ακολουθία
4. Την αρχικοποιημένη γνώση του για το περιβάλλον

Ο βαθμός επίδοσης μπορεί να προκύψει από ένα *μέτρο επίδοσης* (performance measure) για την αξιολόγηση των καταστάσεων στις οποίες μεταβαίνει το περιβάλλον έπειτα από κάθε ενέργεια. Το μέτρο, φυσικά, διαφέρει ανάλογα με τα χαρακτηριστικά του περιβάλλοντος και είναι ευθύνη του σχεδιαστή να το ορίσει κατάλληλα.

Ένας πράκτορας, ο οποίος για κάθε πιθανή αντιληπτική ακολουθία επιλέγει την ενέργεια που μεγιστοποιεί το βαθμό επίδοσής του, στηριζόμενος στην αρχική του γνώση και την πληροφορία που έχει μέσω της αντίληψης καλείται *Ιδανικός Ορθολογιστικός Πράκτορας* (Ideal Rational Agent).

Για την πλήρη περιγραφή ενός πράκτορα απαιτείται ο προσδιορισμός πέντε παραμέτρων [Fran97].

Περιβάλλον (Environment) Για την περιγραφή του περιβάλλοντος χρειάζεται να οριστούν όλες οι πιθανές καταστάσεις που μπορούν να υπάρξουν και η μέθοδος με την οποία γίνονται οι μεταβάσεις μεταξύ των καταστάσεων με την πάροδο του χρόνου. Επίσης πρέπει να οριστεί η *αρχική κατάσταση* (initial state).

Αισθητήριες Δυνατότητες (Sensing Capabilities) Περιγράφονται οι αισθητήρες του πράκτορα και οι αντίστοιχες είσοδοι από το περιβάλλον. Οι είσοδοι έχουν συνήθως τη μορφή διανύσματος, και καθεμία αντιστοιχίζεται σε μία από τις διαστάσεις του περιβάλλοντος.

Ενέργειες (Actions) Η κάθε ενέργεια προσδιορίζεται από την αλλαγή που προκαλεί στην κατάσταση του περιβάλλοντος όταν εφαρμόζεται. Πολύπλοκες ενέργειες μπορούν να προκύψουν ως ακολουθίες των μεμονωμένων βασικών ενεργειών που υποστηρίζει ο πράκτορας.

Αρχική Γνώση (Drives – Preferences) Είναι η ενσωματωμένη "εμπειρία" που έχει ο πράκτορας όταν τοποθετείται στο περιβάλλον. Η αλληλεπίδραση των κανόνων που προκύπτουν από αυτή τη γνώση εξαρτάται από την κατάσταση του περιβάλλοντος αλλά και του ίδιου του πράκτορα, συνεπώς μπορεί να αλληλοσυμπληρώνονται αλλά και να συγκρούονται μεταξύ τους.

Μέθοδος Επιλογής Ενεργειών (Action Selection Architecture) Περιγράφει τον τρόπο με τον οποίο αποφασίζεται ποια ενέργεια θα εκτελεστεί. Η μέθοδος επιλογής λαμβάνει υπόψιν τόσο την αρχική γνώση, όσο και την γνώση που έχει αποκτήσει ο πράκτορας από την εμπειρία του, και είναι ιδιαίτερα σημαντική για την απόδοσή του.

Ο σωστός σχεδιασμός ενός πράκτορα προϋποθέτει την μελέτη και την ορθή κατανόηση αυτών των παραμέτρων. Το περιβάλλον και η μέθοδος επιλογής ενεργειών, είναι αυτά που επηρεάζουν περισσότερο την απόδοση του καθώς παρουσιάζουν τη μεγαλύτερη ποικιλία και θα εξεταστούν αναλυτικά στη συνέχεια.

2.2 Περιβάλλοντα Πρακτόρων

Το περιβάλλον περιλαμβάνει όλα τα στοιχεία με τα οποία μπορεί να αλληλεπιδράσει ο πράκτορας. Οι είσοδοι λαμβάνονται από το περιβάλλον με τη χρήση των αισθητήρων, και οι ενεργοποιητές δρουν σε αυτό αλλάζοντας την κατάστασή του. Όλα τα περιβάλλοντα έχουν ορισμένες διαστάσεις/ιδιότητες με βάση τις οποίες μπορούν να κατηγοριοποιηθούν. Η υλοποίηση των πρακτόρων πρέπει να είναι προσαρμοσμένη σε αυτές τις ιδιότητες για την επίτευξη της καλύτερης δυνατής απόδοσης. Η κατηγοριοποίηση των περιβαλλόντων γίνεται με βάση τις παρακάτω ιδιότητες [Russ95].

Παρατηρησιμότητα (Obervability) Με τον όρο παρατηρησιμότητα ορίζεται η δυνατότητα να λαμβάνονται οι είσοδοι του περιβάλλοντος από τον πράκτορα. Με βάση αυτή τη δυνατότητα τα περιβάλλοντα διακρίνονται σε *πλήρως παρατηρήσιμα* (fully observable) και *μερικώς παρατηρήσιμα* (partially observable). Σε ένα πλήρως παρατηρήσιμο περιβάλλον ο πράκτορας έχει πρόσβαση σε όλες τις παραμέτρους/εισόδους που χρειάζεται για να πάρει μία απόφαση ενώ σε ένα μερικώς παρατηρήσιμο, δεν είναι διαθέσιμη ολόκληρη η κατάσταση του περιβάλλοντος. Αυτό μπορεί να οφείλεται είτε σε θόρυβο, ο οποίος μειώνει την αντίληψη του πράκτορα, είτε σε αρχιτεκτονικούς περιορισμούς, σε περίπτωση για παράδειγμα που δεν υπάρχουν όλοι οι απαραίτητοι αισθητήρες ή κάποιοι δεν λειτουργούν σωστά. Πιο σπάνια συναντάται και η περίπτωση μη παρατηρήσιμων περιβαλλόντων, στα οποία ο πράκτορας δεν έχει τη δυνατότητα να λάβει καμία είσοδο, χωρίς ωστόσο αυτό να συνεπάγεται απαραίτητα αδυναμία επίτευξης του στόχου.

Πλήθος Πρακτόρων Ανάλογα με τον αριθμό των πρακτόρων που συνυπάρχουν σε ένα περιβάλλον, αυτό μπορεί να είναι *ενός-πράκτορα* (single-agent) ή *πολλών-πρακτόρων* (multi-agent). Για να είναι ξεκάθαρος αυτός ο διαχωρισμός πρέπει να είναι δυνατή η διάκριση των πρακτόρων από τα απλά αντικείμενα του περιβάλλοντος, διαφορετικά μπορεί να υπάρξει σύγχυση. Στα περιβάλλοντα πολλών-πρακτόρων, οι στόχοι τους είναι δυνατόν να είναι κοινοί ή αλληλοσυγκρουόμενοι, οπότε ανάλογα με τη συμπεριφορά τους, τα περιβάλλοντα μπορούν να διακριθούν επιπλέον σε *συνεργατικά* (cooperative) και *ανταγωνιστικά* (competitive) αντίστοιχα.

Προβλεψιμότητα Με την έννοια της προβλεψιμότητας ορίζεται η εξάρτηση της μετάβασης του περιβάλλοντος σε μία κατάσταση από οποιονδήποτε παράγοντα πέραν της τρέχουσας κατάστασης και της ενέργειας που εκτελεί ο πράκτορας. Ένα περιβάλλον του οποίου οι καταστάσεις δεν εξαρτώνται από κανέναν τέτοιο παράγοντα ονομάζεται *ντετερμινιστικό* (deterministic), ενώ σε αντίθετη περίπτωση *στοχαστικό* (stochastic). Αν ένα περιβάλλον δεν είναι πλήρως παρατηρήσιμο ή ντετερμινιστικό καλείται *αβέβαιο* (uncertain). Στη γενική περίπτωση ένα πραγματικό περιβάλλον πρέπει να αντιμετωπίζεται ως στοχαστικό γιατί υπάρχουν πολλές παράμετροι με απρόβλεπτη συμπεριφορά, που μπορούν να επηρεάσουν την μετάβαση των καταστάσεων.

Μνήμη Τα περιβάλλοντα που έχουν την ιδιότητα της μνήμης, δηλαδή αυτά στα οποία μία μελλοντική κατάσταση επηρεάζεται από τις προηγούμενες, ονομάζονται *ακολουθιακά* (sequential). Για παράδειγμα σε ένα παιχνίδι στρατηγικής, μία ενέργεια του πράκτορα μπορεί να είναι καθοριστική για

την εξέλιξη της πορείας του παιχνιδιού. Στα περιβάλλοντα χωρίς μνήμη, ο πράκτορας δεν χρειάζεται να αποθηκεύει την αντιληπτική ακολουθία. Η αντίληψη των εισόδων γίνεται ανά επεισόδια, τα οποία είναι ανεξάρτητα μεταξύ τους, και γι' αυτό ονομάζονται *επεισοδιακά* (episodic) περιβάλλοντα. Τα επεισοδιακά είναι σαφώς απλούστερα από τα ακολουθιακά, καθώς απαιτούν σημαντικά λιγότερους υπολογισμούς για την επιλογή της ενέργειας του πράκτορα.

Δυναμικότητα Ένα περιβάλλον χαρακτηρίζεται *δυναμικό* (dynamic) αν μπορεί να αλλάζει καταστάσεις κατά τη διάρκεια που ο πράκτορας εκτελεί τους απαραίτητους υπολογισμούς για να πάρει μία απόφαση. Αντίστοιχα καλείται *στατικό* (static) όταν προϋπόθεση για να αλλάξει κατάσταση είναι η εκτέλεση κάποιας ενέργειας από τον πράκτορα. Τα δυναμικά περιβάλλοντα είναι πιο πολύπλοκα, αφού ο πράκτορας πρέπει συνεχώς να λαμβάνει υπόψιν τις διάφορες αλλαγές που συμβαίνουν.

Συνέχεια Η συνέχεια είναι μία ιδιότητα που χαρακτηρίζει ένα περιβάλλον με βάση τον τρόπο διαχείρισης του χρόνου. Έτσι τα περιβάλλοντα κατατάσσονται σε *διακριτά* (discrete) και *συνεχή* (continuous). Διακριτά είναι αυτά στα οποία μπορεί να οριστεί ξεκάθαρα ένα χρονικό διάστημα για κάθε μία κατάσταση μεμονωμένα. Αντίθετα σε ένα συνεχές περιβάλλον δεν υπάρχει σαφής αριθμός καταστάσεων, καθώς αυτές μεταβάλλονται αδιάλειπτα. Ένας πράκτορας για παράδειγμα ο οποίος λαμβάνει με τους αισθητήρες του ένα ηχητικό σήμα για ένα χρονικό διάστημα ανήκει σε συνεχές περιβάλλον, ενώ ένας ο οποίος αποφασίζει τις κινήσεις σε μία παρτίδα πόκερ, σε διακριτό.

Γνώση Κανόνων Η γνώση των κανόνων είναι μία ιδιότητα που επηρεάζει έμμεσα το περιβάλλον. Πρόκειται στην πραγματικότητα για την αρχικοποιημένη γνώση που έχουν οι πράκτορες σχετικά με τις θεμελιώδεις αρχές του περιβάλλοντος. Αν οι πράκτορες γνωρίζουν τους βασικούς κανόνες που το διέπουν, τότε το περιβάλλον είναι *γνωστό* (known) διαφορετικά είναι *άγνωστο* (unknown). Σε ένα άγνωστο περιβάλλον, ο πράκτορας μαθαίνει τελικά τους κανόνες μέσω της εμπειρίας που αποκτά, οπότε η διαφορά του με ένα γνωστό είναι πρακτικά η αρχική κατάσταση του πράκτορα.

2.3 Κατηγορίες Πρακτόρων

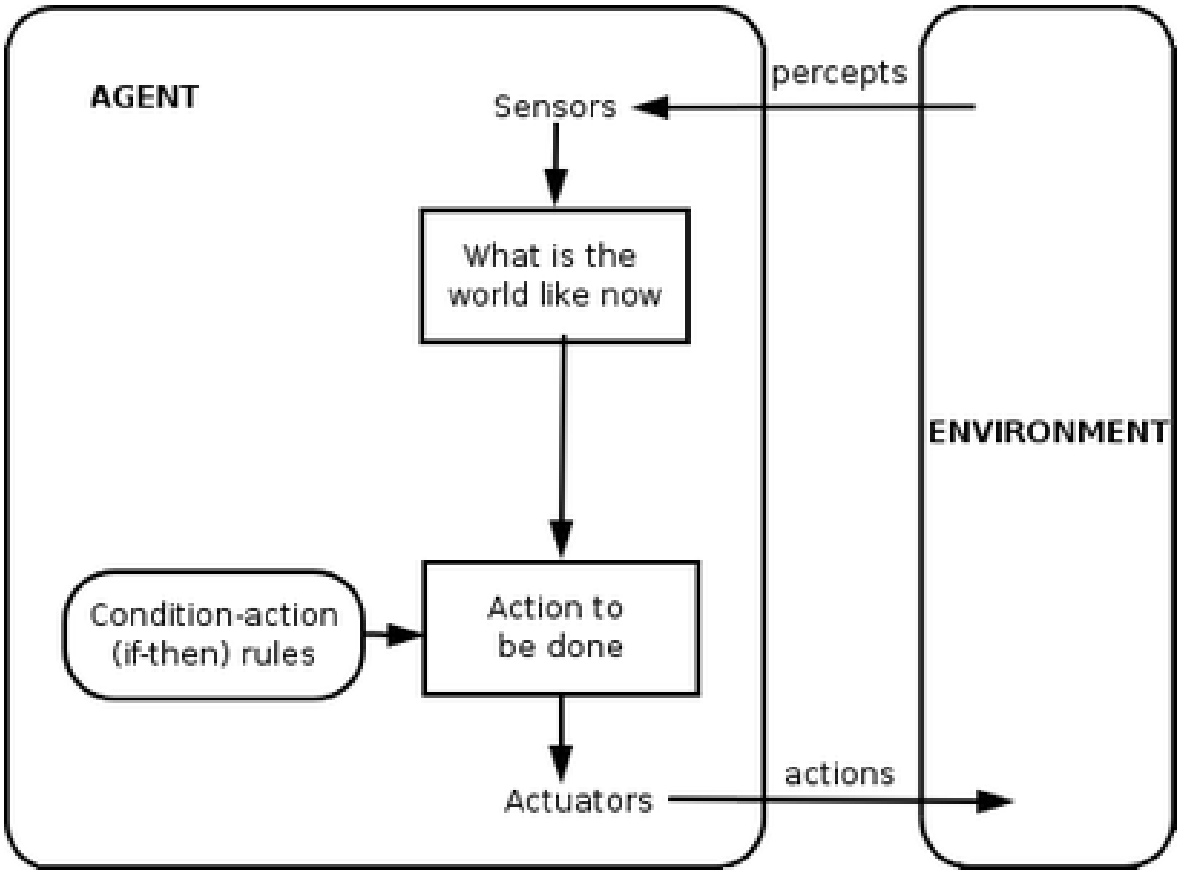
Ο δεύτερος παράγοντας που επηρεάζει την αποτελεσματικότητα ενός πράκτορα εξίσου σημαντικά με το περιβάλλον του, είναι η συνάρτηση επιλογής ενεργειών. Ιδανικά η συνάρτηση επιλογής πρέπει να μπορεί να περιγράψει ένα γενικό σχήμα που να καλύπτει όλες τις πιθανές αντιληπτικές ακολουθίες με όσο το δυνατόν απλούστερη λογική και λιγότερους υπολογισμούς. Με βάση τον τύπο του προγράμματος που υλοποιεί αυτή τη συνάρτηση οι πράκτορες μπορούν να καταταχθούν σε τέσσερις κατηγορίες [Russ95].

2.3.1 Απλοί Πράκτορες Αντανάκλασης (Simple Reflex Agents)

Σε αυτή την κατηγορία ανήκουν οι πράκτορες που στηρίζουν κάθε φορά την επιλογή τους μόνο στην αντίληψη της δεδομένης κατάστασης. Για το σκοπό αυτό λειτουργούν με κανόνες που αντιστοιχίζουν κάποιες συνθήκες σε προκαθορισμένες ενέργειες. Οι κανόνες αυτοί ονομάζονται *κανόνες συνθήκης-ενέργειας* (condition-action rules).

Αρχικά ένας απλός πράκτορας αντανάκλασης λαμβάνει τις εισόδους από το περιβάλλον με τη χρήση των αισθητήρων του. Έπειτα το πρόγραμμα πράκτορα διατρέχει το σύνολο των κανόνων και ελέγχει τις συνθήκες που ορίζονται σε αυτό. Για την πρώτη που ικανοποιείται, αν υπάρχει τέτοια, εκτελεί την αντίστοιχη – σύμφωνα με τον κανόνα – ενέργεια, μέσω των ενεργοποιητών. Αυτή η λειτουργία απεικονίζεται στο Σχήμα 2.2.

Οι πράκτορες αυτού του τύπου είναι εξαιρετικά απλοί στην υλοποίηση τους, καθώς δεν ενδιαφέρονται για τις καταστάσεις του παρελθόντος και συνεπώς δεν χρειάζεται να αποθηκεύουν και να επεξεργάζονται ολόκληρη την αντιληπτική ακολουθία, παρά μόνο την τρέχουσα αντίληψη. Αυτό,



Σχήμα 2.2: Απλός πράκτορας αντανάκλασης

ωστόσο, έχει αρνητικές συνέπειες στην αποτελεσματικότητά τους, που οφείλονται στη μη αξιοποίηση χρήσιμων πληροφοριών από τις προηγούμενες καταστάσεις.

Βασική προϋπόθεση για να έχει ένας απλός πράκτορας αντανάκλασης την επιθυμητή συμπεριφορά, είναι να βρίσκεται σε πλήρως παρατηρήσιμο περιβάλλον. Αν το περιβάλλον είναι μερικώς παρατηρήσιμο είναι πιθανό η τρέχουσα αντίληψη να μην αρκεί για την ανίχνευση μίας συνθήκης κάποιου κανόνα, η οποία στην πραγματικότητα ικανοποιείται. Ένας άλλος κίνδυνος που μπορεί να προκύψει σε τέτοιο περιβάλλον είναι να εγκλωβιστεί ο πράκτορας σε έναν ατέρμονο βρόχο. Αυτό μπορεί να αντιμετωπιστεί με την εισαγωγή τυχαιότητας στις κινήσεις του πράκτορα, όμως δεν ενδείκνυται γιατί στην πλειοψηφία των περιπτώσεων οι πράκτορες που εφαρμόζουν αυτή την τεχνική δεν είναι ορθολογιστικοί.

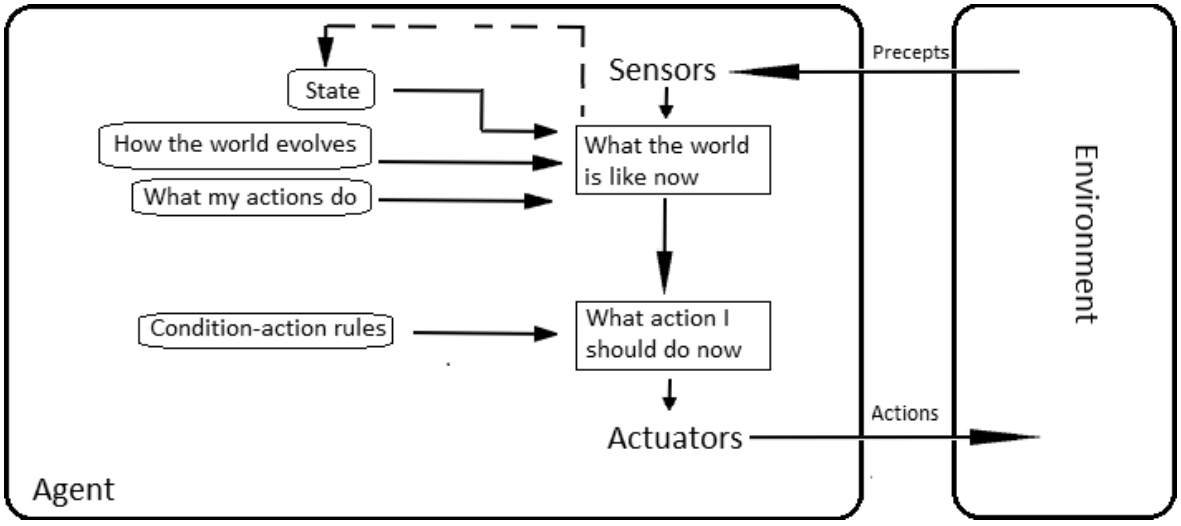
2.3.2 Πράκτορες Αντανάκλασης Βασισμένοι σε Μοντέλο (Model-based Reflex Agents)

Αυτοί οι πράκτορες αποτελούν επέκταση των απλών πρακτόρων αντανάκλασης, έτσι ώστε να είναι ικανοί να παίρνουν τη σωστή απόφαση και σε μερικώς παρατηρήσιμα περιβάλλοντα. Για να το πετύχουν αυτό διατηρούν ανά πάσα στιγμή μία *εσωτερική κατάσταση* (internal state), που βασίζεται και σε προηγούμενες εισόδους της αντιληπτικής ακολουθίας, προκειμένου να αντισταθμίσουν έτσι την απουσία κάποιων από τις εισόδους της τρέχουσας κατάστασης.

Η εσωτερική κατάσταση ανανεώνεται με κάθε νέα αντίληψη του περιβάλλοντος από τον πράκτορα. Για να μπορεί να γίνεται η ανανέωση επιτυχώς, απαιτείται επιπλέον γνώση για το περιβάλλον σε δύο βασικές κατευθύνσεις. Η πρώτη αφορά την εξέλιξη του περιβάλλοντος ανεξάρτητα από την ύπαρξη και τη δραστηριότητα του πράκτορα, και προϋποθέτει τη γνώση των βασικών κανόνων-νόμων που ισχύουν και δρουν μόνιμα σε αυτό. Η δεύτερη απαραίτητη πληροφορία σχετίζεται με την αλλαγή της κατάστασης του περιβάλλοντος σε συνάρτηση με τις ενέργειες του πράκτορα.

Η γνώση για τον τρόπο με τον οποίο εξελίσσεται ο "κόσμος" στον οποίο λειτουργεί ο πράκτορας όπως περιγράφηκε, ονομάζεται μοντέλο και οι πράκτορες που την αξιοποιούν για να ενεργήσουν κατάλληλα, *πράκτορες βασισμένοι σε μοντέλο* (model-based agents). Οι πράκτορες αντανάκλασης βασισμένοι σε μοντέλο συνδυάζουν την αντίληψη με την εσωτερική κατάσταση και τη γνώση για την λειτουργία του περιβάλλοντος, τόσο ανεξάρτητα όσο και υπό την επίδραση των ενεργειών τους, για να σχηματίσουν την νέα κατάσταση. Στη συνέχεια λειτουργούν όπως οι απλοί πράκτορες αντανάκλασης χρησιμοποιώντας αυτή την κατάσταση.

Παρακάτω παρουσιάζεται αυτή η διαδικασία με τη μορφή διαγράμματος (Σχήμα 2.3).



Σχήμα 2.3: Πράκτορας αντανάκλασης βασισμένος σε μοντέλο

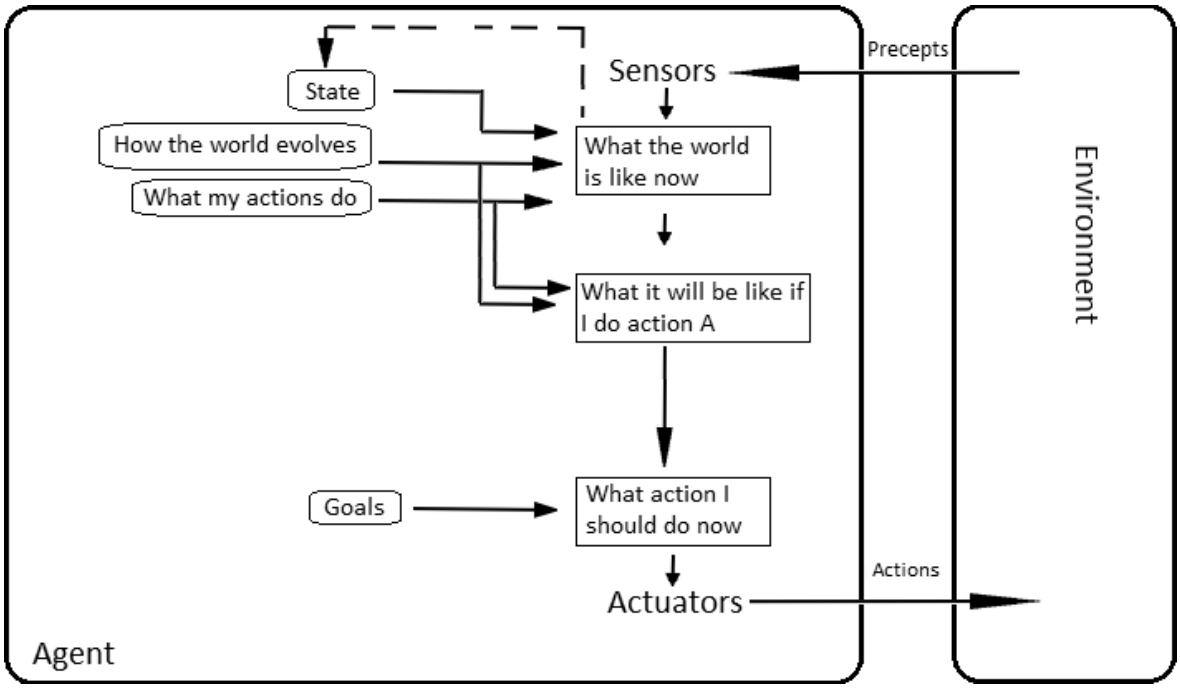
2.3.3 Πράκτορες Βασισμένοι σε Στόχους (Goal-based Agents)

Οι πράκτορες που βασίζονται σε στόχους, επιλέγουν κάθε φορά την κίνηση που σύμφωνα με τους υπολογισμούς τους, θα τους φέρει πιο κοντά στην επίτευξή αυτών.

Κάποιες φορές ενδέχεται μία ενέργεια να αρκεί για την πραγματοποίηση του στόχου, αλλά συνήθως απαιτείται μία ακολουθία ενεργειών (είναι πιθανόν να υπάρχουν περισσότερες από μία τέτοιες διαφορετικές ακολουθίες). Αυτό σημαίνει ότι ο πράκτορας πρέπει να είναι σε θέση να κάνει έναν σχεδιασμό αυτής της ακολουθίας και να προαποφασίσει τουλάχιστον κάποιες από τις μελλοντικές του ενέργειες. Για τον σχεδιασμό, χρησιμοποιεί το μοντέλο του "κόσμου", ώστε να μπορεί να προβλέψει και να αξιολογήσει τις καταστάσεις που θα προκύψουν από τις πιθανές ενέργειές του και να επιλέξει αυτές που εξυπηρετούν τους στόχους του.

Σε σύγκριση με τους πράκτορες αντανάκλασης, οι οποίοι αντιστοιχίζουν μία κατάσταση σε κάποια ενέργεια, οι πράκτορες που βασίζονται σε στόχους είναι αρκετά πιο πολύπλοκοι καθώς ακόμα και σε περιπτώσεις που τελικά καταλήγουν στην ίδια απόφαση έχουν χρησιμοποιήσει πολύ περισσότερους υπολογιστικούς πόρους. Όμως έχουν το πλεονέκτημα ότι μπορούν και προσαρμόζονται στις αλλαγές του περιβάλλοντος, κάτι που δεν ισχύει για τους πράκτορες αντανάκλασης, εφόσον οι ενέργειές τους είναι προκαθορισμένες ανάλογα με την κατάσταση.

Επίσης είναι πολύ εύκολο για έναν βασισμένο σε στόχους πράκτορα να χρησιμοποιηθεί για την επίλυση διαφορετικών προβλημάτων. Το γεγονός ότι μπορεί να σχεδιάσει το πλάνο του χρησιμοποιώντας κάποια "λογική", του επιτρέπει να αναπροσαρμόζεται σε ένα πρόβλημα, αρκεί να ορίζονται κατάλληλα οι εκάστοτε στόχοι. Αντίθετα σε έναν πράκτορα αντανάκλασης θα έπρεπε να οριστούν από την αρχή όλοι οι κανόνες και σε κάθε συνθήκη να αντιστοιχιστεί η νέα επιθυμητή ενέργεια.



Σχήμα 2.4: Πράκτορας βασισμένος σε στόχους

2.3.4 Πράκτορες Βασισμένοι στην Ωφελιμότητα (Utility-based Agents)

Οι πράκτορες που βασίζονται σε στόχους, μπορούν πολλές φορές να τους πετύχουν με αρκετούς διαφορετικούς τρόπους. Στην περίπτωση που ελέγχεται μόνο η επίτευξή τους, όλες οι ακολουθίες ενεργειών που το καταφέρνουν κρίνονται επιτυχείς, αυτό όμως δεν σημαίνει ότι είναι όλες το ίδιο αποδοτικές. Αν εξεταστούν και άλλες παράμετροι όπως για παράδειγμα ο απαιτούμενος χρόνος για την ολοκλήρωση της διαδικασίας, τότε κάποιες ακολουθίες είναι "καλύτερες" από κάποιες άλλες. Αυτή η διαβάθμιση μεταξύ των διαφορετικών ακολουθιών που οδηγούν στην επίτευξη των στόχων εκφράζεται με την έννοια της ωφελιμότητας.

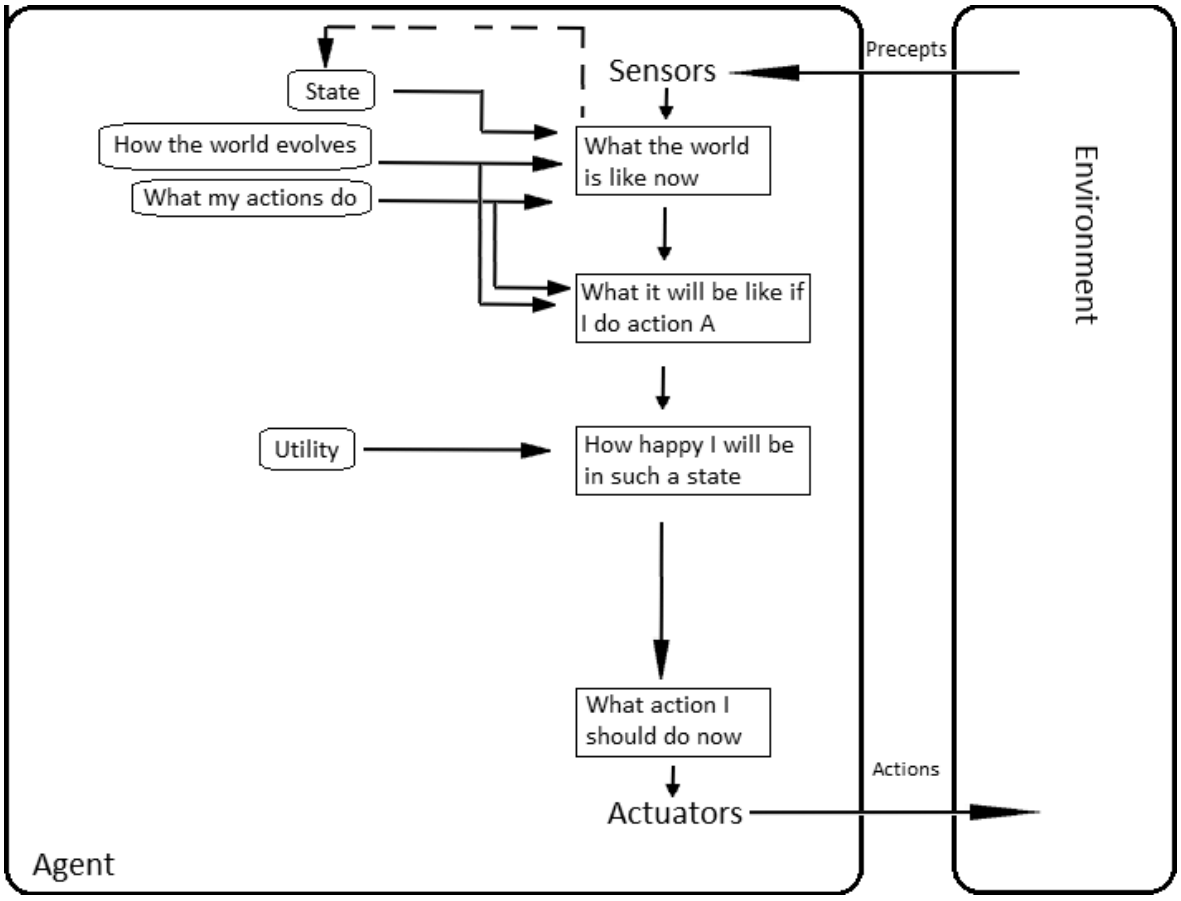
Οι πράκτορες που βασίζονται στην ωφελιμότητα χρειάζονται ένα μέτρο ώστε να μπορούν να επιλέξουν την καλύτερη δυνατή ακολουθία. Για το σκοπό αυτό χρησιμοποιείται μία μέθοδος, η οποία καλείται συνάρτηση ωφελιμότητας και επιστρέφει ένα τέτοιο μέτρο για κάθε πιθανή ενέργεια. Για να είναι ορθολογιστικός ο πράκτορας, θα πρέπει η συνάρτηση ωφελιμότητας να συμβαδίζει με το μέτρο επίδοσης του πράκτορα, δηλαδή οι ενέργειες που επιλέγονται να οδηγούν σε καταστάσεις που μεγιστοποιούν το βαθμό επίδοσης.

Ένας πράκτορας ωφελιμότητας ακολουθεί όλα τα βήματα που κάνει και ένας πράκτορας που βασίζεται σε στόχους και ελέγχει επιπλέον την ωφελιμότητα της κάθε ενέργειας προτού αποφασίσει. Στην πραγματικότητα, επειδή τα περισσότερα περιβάλλοντα είναι στοχαστικά, ελέγχει την προσδοκώμενη ωφελιμότητα αφού δεν μπορεί να προσδιοριστεί με βεβαιότητα.

Εκτός απ' το ότι είναι πιο αποδοτικοί, οι πράκτορες που βασίζονται στην ωφελιμότητα έχουν δύο επιπλέον πλεονεκτήματα.

Το πρώτο αφορά τους αλληλοσυγκρουόμενους στόχους. Τέτοιοι στόχοι είναι πιο συχνοί σε περιβάλλοντα πολλών πρακτόρων, όμως μπορεί να υπάρχουν και σε περιβάλλον με έναν μοναδικό πράκτορα. Σε αυτές τις περιπτώσεις ο πράκτορας μπορεί εύκολα να αποφασίσει, με τη χρήση της συνάρτησης ωφελιμότητας, ποιοι στόχοι είναι πιο σημαντικοί και πρέπει να προτιμηθούν έναντι άλλων οι οποίοι θα πρέπει να απορριφθούν.

Το δεύτερο πλεονέκτημά τους σχετίζεται επίσης με περιβάλλοντα στα οποία υπάρχουν περισσότεροι από έναν στόχοι χωρίς όμως απαραίτητα να είναι αντικρουόμενοι. Όταν δεν υπάρχει βεβαιότητα ότι κάποιος από τους στόχους μπορεί να επιτευχθεί, η ωφελιμότητα των στόχων μπορεί να "υποδείξει" αυτούς στους οποίους πρέπει να δοθεί προτεραιότητα με βάση την πιθανότητά τους να πραγματοποι-



Σχήμα 2.5: Πράκτορας βασισμένος στην ωφελιμότητα

ηθούν.

2.3.5 Πράκτορες Εκμάθησης (Learning Agents)

Η ανάπτυξη του προγράμματος ενός πράκτορα "χειροκίνητα" είναι μία πολύ απαιτητική και καθόλου αποδοτική διαδικασία. Αφενός είναι ιδιαίτερα χρονοβόρα, αφετέρου είναι δύσκολο να προβλεφθούν όλες οι πιθανές καταστάσεις του "κόσμου" και είναι πιθανό να υπάρχουν κενά στην περιγραφή του προγράμματος.

Η τεχνική που χρησιμοποιείται κατά κύριο λόγο σήμερα είναι η δημιουργία πρακτόρων ικανών να εκπαιδευτούν και να προσαρμοστούν αυτόματα στις συνθήκες του περιβάλλοντος. Αυτοί οι πράκτορες ονομάζονται *πράκτορες εκμάθησης* (learning agents).

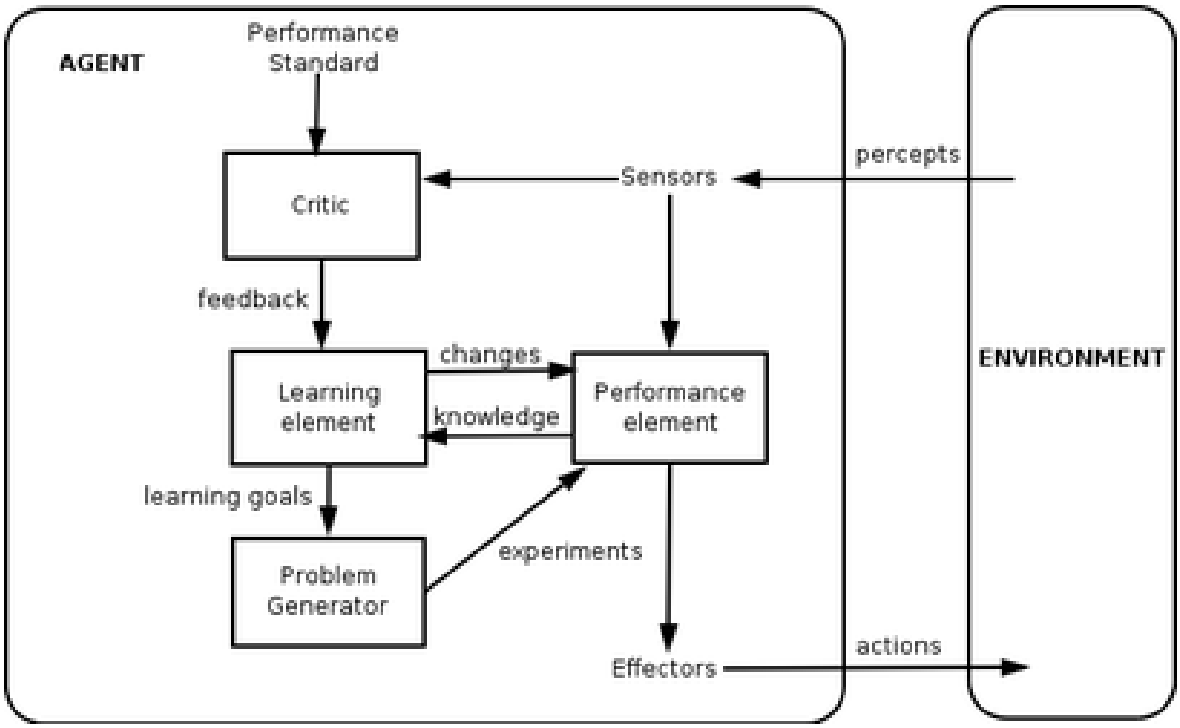
Η δομή ενός πράκτορα εκμάθησης φαίνεται στο Σχήμα 2.6. Αποτελείται από τέσσερα βασικά στοιχεία [Russ95], τα οποία είναι:

Στοιχείο-Κριτής (Critic) Αυτό το στοιχείο αξιολογεί την επίδοση του πράκτορα σύμφωνα με κάποιο *πρότυπο επίδοσης* (performance standard) και ενημερώνει ανάλογα το στοιχείο εκμάθησης. Είναι απαραίτητο γιατί η αντίληψη που λαμβάνει ο πράκτορας μέσω των αισθητήρων περιγράφει μεν την τρέχουσα κατάσταση, χωρίς όμως να παρέχει κάποια πληροφορία για το αν η κατάσταση αυτή είναι επιθυμητή ή όχι.

Στοιχείο Εκμάθησης (Learning Element) Το στοιχείο εκμάθησης είναι υπεύθυνο για την προσαρμογή του πράκτορα στα νέα δεδομένα, με βάση την *ανάδραση* (feedback) που παίρνει από το στοιχείο-κριτή. Ουσιαστικά υλοποιεί την λογική της βελτίωσης του μοντέλου που χρησιμοποιεί ο πράκτορας για να πάρει τις αποφάσεις του.

Στοιχείο Επίδοσης (Performance Element) Πρόκειται για το τμήμα του πράκτορα που υλοποιεί τη συνάρτηση επιλογής των ενεργειών και δρα στο περιβάλλον μέσω των ενεργοποιητών.

Γεννήτρια Προβλημάτων (Problem Generator) Με τα μέχρι στιγμής ορισμένα στοιχεία, ο πράκτορας επιλέγει κάθε φορά την ενέργεια που του υποδεικνύει το στοιχείο επίδοσης, έτσι όπως έχει διαμορφωθεί από την αλληλεπίδραση του με το στοιχείο εκμάθησης. Όμως στο στάδιο της εκπαίδευσης, στόχος είναι ο πράκτορας να ανακαλύψει όσο το δυνατόν περισσότερες πιθανές καταστάσεις και να τις κωδικοποιήσει κατάλληλα για να μπορεί να τις αναγνωρίσει στο μέλλον. Αυτό το ρόλο αναλαμβάνει η γεννήτρια προβλημάτων που συνδυάζεται με το στοιχείο εκμάθησης και προτείνει ενέργειες που θα οδηγήσουν σε νέες καταστάσεις, οι οποίες δεν έχουν εξερευνηθεί ακόμα, επιδιώκοντας μακροπρόθεσμα τη βελτίωση της συμπεριφοράς του πράκτορα.



Σχήμα 2.6: Δομή ενός πράκτορα εκμάθησης

Προηγουμένως θεωρήσαμε ότι το στοιχείο-κριτής χρησιμοποιεί κάποια γνωστή μέθοδο με την οποία η αντίληψη μίας κατάστασης εκλαμβάνεται ως *ανταμοιβή* (reward) ή *ποινή* (penalty) σύμφωνα με το πρότυπο επίδοσης. Η υπόθεση αυτή όμως σε ένα πραγματικό περιβάλλον δεν ισχύει για δύο λόγους. Πρώτον ο προσδιορισμός μίας τέτοιας *συνάρτησης ανταμοιβής* (reward function) γίνεται εμπειρικά και κατά συνέπεια μπορεί να αποδειχθεί λανθασμένος και δεύτερον στις περισσότερες περιπτώσεις απαιτεί το συνυπολογισμό πολλών παραμέτρων, των οποίων η βαρύτητα δεν είναι δυνατόν να οριστεί εκ των προτέρων [Russ98].

Συνεπώς για να είναι δυνατή η υλοποίηση ενός πράκτορα εκμάθησης είναι αναγκαίο να μπορεί να καθοριστεί η συνάρτηση ανταμοιβής με τη χρήση:

- μέτρων της συμπεριφοράς του πράκτορα υπό διάφορες συνθήκες
- μέτρων των εισόδων του πράκτορα από τους αισθητήρες του
- του μοντέλου του περιβάλλοντος στο οποίο λειτουργεί

Αυτή η διαδικασία είναι γνωστή ως *Εκμάθηση Αντίστροφης Ενίσχυσης* (Inverse Reinforcement Learning) και αποτελεί μέχρι στιγμής πεδίο έρευνας όσον αφορά τις προϋποθέσεις, την πολυπλοκότητα και τους πιθανούς αλγορίθμους επίλυσης.

2.4 Εφαρμογές Ευφυών Πρακτόρων

Οι ευφυείς πράκτορες έχουν εφαρμογή σε πολλούς τομείς όπως η βιομηχανία, η ιατρική ακόμα και η ψυχαγωγία, γεγονός που δικαιολογεί και την αυξανόμενη τάση για την ανάπτυξη τους. Παρακάτω παρουσιάζονται ορισμένες από αυτές τις εφαρμογές.

Ηλεκτρονικό Εμπόριο Καθώς πολύ μεγάλο μέρος των αγορών γίνεται πλέον ηλεκτρονικά, οι διαδικτυακές σελίδες των καταστημάτων χρησιμοποιούν σε μεγάλο βαθμό ευφυείς πράκτορες για τη διαχείριση των απαραίτητων ενεργειών. Η κυριότερη εφαρμογή τους στο ηλεκτρονικό εμπόριο αφορά την επεξεργασία του ιστορικού και το συσχετισμό μεταξύ πελατών με παρόμοιες επιλογές, με στόχο την εύρεση και σύσταση προϊόντων ανάλογα με τις προτιμήσεις του κάθε πελάτη. Επίσης σε πολλές περιπτώσεις, πράκτορες χρησιμοποιούνται για τη διαχείριση της διαδικασίας των πληρωμών σε ηλεκτρονικά καταστήματα [Pivk00].

Εξόρυξη Δεδομένων (Data Mining) Η εξόρυξη δεδομένων είναι ένας κλάδος της πληροφορικής που σχετίζεται με την εξεύρεση χρήσιμης πληροφορίας σε έναν πολύ μεγάλο όγκο δεδομένων (πχ μία βάση δεδομένων) [Han12] και αποτελεί αντικείμενο συνεχώς αυξανόμενης έρευνας τα τελευταία χρόνια. Ευφυείς πράκτορες χρησιμοποιούνται σε μεγάλο βαθμό για το φιλτράρισμα της διαθέσιμης πληροφορίας και την αυτοματοποίηση αυτής της διαδικασίας.

Ιατρικές Εφαρμογές Οι πράκτορες εμπλέκονται σε όλο και περισσότερες διαδικασίες που αφορούν την περίθαλψη ασθενών καθώς η τεχνολογία διεισδύει όλο και περισσότερο στην ιατρική. Πλέον υπάρχουν πράκτορες που αναλαμβάνουν τον προγραμματισμό των επισκέψεων των ασθενών, ο οποίος είναι αρκετά πολύπλοκος σε μεγάλες ιατρικές μονάδες, και έχει αποδειχτεί ότι ανταποκρίνονται καλύτερα από τις χειροκίνητες μεθόδους. Επιπλέον, διαδικασίες που εμπλέκουν περισσότερους από έναν ασθενείς, όπως για παράδειγμα η μεταμόσχευση οργάνων, αναπαριστώνται με συνεργατικά περιβάλλοντα πολλών-πρακτόρων, επιτυγχάνοντας πολύ ταχύτερο σχεδιασμό της ροής των εργασιών από τις παραδοσιακές μεθόδους [Neal02].

Οικονομικά Μοντέλα Μία από τις κύριες εφαρμογές των πρακτόρων είναι η προσομοίωση οικονομικών μοντέλων. Η υλοποίησή των μοντέλων περιλαμβάνει τη δημιουργία ενός εικονικού περιβάλλοντος οικονομίας, στο οποίο συμμετέχουν πράκτορες που μπορούν να αλληλεπιδρούν μεταξύ τους και να παίρνουν αποφάσεις με σκοπό την μεγιστοποίηση των κερδών τους. Τέτοια μοντέλα ονομάζονται *υπολογιστικά οικονομικά μοντέλα βασισμένα σε πράκτορες* (agent-based computational economic models) και έχουν το πλεονέκτημα ότι ξεκινάνε χωρίς προϋπάρχουσα γνώση και συνεπώς δεν επηρεάζονται από κάποιο ιστορικό.

Management Επιχειρήσεων Οι πράκτορες έχουν εφαρμογή και στη διαχείριση των επιχειρηματικών διαδικασιών. Σε μία επιχείρηση, ένας διευθυντής καλείται να πάρει αποφάσεις συνδυάζοντας πληροφορίες από πολλά διαφορετικά τμήματα. Ωστόσο, η συγκέντρωση όλων των αναγκαίων πληροφοριών σε μία μεγάλη εταιρεία, είναι πολύ επίπονη και χρονοβόρα διαδικασία. Για την αντιμετώπιση αυτού του προβλήματος, κάποιες επιχειρήσεις έχουν αναπτύξει πράκτορες, καθένας από τους οποίους αντιπροσωπεύει ένα τμήμα. Οι πράκτορες μπορούν να παρέχουν υπηρεσίες και να επικοινωνούν μεταξύ τους και τελικά να προτείνουν στους διευθυντές τις ενδεδειγμένες αποφάσεις ή ακόμα και να αποφασίζουν οι ίδιοι [Jenn98].

Messaging Τελευταία, αρκετές εταιρείες χρησιμοποιούν πράκτορες για την επικοινωνία με τους πελάτες τους μέσω γραπτών μηνυμάτων. Πρόκειται για πράκτορες που εκπαιδεύονται κατάλληλα ώστε να μπορούν να "καταλαβαίνουν" τουλάχιστον τις απλές, συνηθισμένες ερωτήσεις και να απαντάνε

αντίστοιχα, χωρίς να χρειάζεται ανθρώπινη παρέμβαση. Αυτή η τεχνολογία επιδέχεται πολλές βελτιώσεις, καθότι βρίσκεται ακόμα σε αρχικό στάδιο, αλλά ήδη εφαρμόζεται σε πολλές περιπτώσεις και αποτελεί ιδιαίτερα δημοφιλές αντικείμενο έρευνας.

Ηλεκτρονικά Παιχνίδια Σε ένα *ηλεκτρονικό παιχνίδι* (video game), ο πράκτορας παίρνει τη θέση του παίχτη, αναλαμβάνοντας να πάρει αποφάσεις για τις κινήσεις του χαρακτήρα (ή των χαρακτήρων) του παιχνιδιού. Στην πράξη, τέτοιοι πράκτορες μπορούν να χρησιμεύσουν είτε για τον έλεγχο των υπόλοιπων χαρακτήρων του παιχνιδιού –συνήθως των αντιπάλων του παίχτη– είτε για *δοκιμές* (testing) σε νέες πίστες ή σε διαφορετικά επίπεδα δυσκολίας. Μία χαρακτηριστική τέτοια υλοποίηση πράκτορα σε video game έχει γίνει για το ευρέως γνωστό παιχνίδι Tetris [Jenn98].

Κεφάλαιο 3

Γενετικοί Αλγόριθμοι

3.1 Εισαγωγή

Οι *Γενετικοί Αλγόριθμοι* (ΓΑ) είναι μία κατηγορία αλγορίθμων που στηρίζεται σε βασικές έννοιες της εξελικτικής διαδικασίας στη φύση και ανήκει στην ευρύτερη κατηγορία των *Εξελικτικών Αλγορίθμων* (ΕΑ). Η κύρια λογική γύρω από την οποία αναπτύχθηκαν είναι αυτή της βιολογικής εξέλιξης, κατά την οποία με την πάροδο του χρόνου, σε ένα σύστημα επιβιώνουν και αναπαράγονται τα άτομα με τα πιο ”ισχυρά” χαρακτηριστικά, σε σημαντικά υψηλότερο ποσοστό από τα “αδύναμα” άτομα. Αυτό έχει ως αποτέλεσμα οι απόγονοι που θα προκύψουν να κληρονομήσουν σε μεγαλύτερο βαθμό τα συγκεκριμένα χαρακτηριστικά και κατά συνέπεια οι νεότερες γενιές να είναι πιο “ισχυρές” από τις προηγούμενες.

Η εισαγωγή στην έννοια των εξελικτικών αλγορίθμων έγινε τη δεκαετία του 1950. Πρώτος ο Alan Turing χρησιμοποίησε τον όρο “learning machine” για να περιγράψει ένα μοντέλο που θα προσομοίωνε τις βασικές αρχές της εξέλιξης [TURI50]. Στην πράξη αυτή η προσομοίωση της εξέλιξης με τη χρήση υπολογιστή υλοποιήθηκε το 1954 από τον Nils Aall Barricelli και ακολούθησαν αρκετές προσπάθειες ανάπτυξης ΕΑ στις δεκαετίες 1950-1960 τόσο στο πεδίο της βιολογίας όσο και σε αυτό της τεχνητής νοημοσύνης. Το 1965 ο Rachenberg παρουσίασε τις *Στρατηγικές Εξέλιξης* (Evolution Strategies) [Schw81], και ένα χρόνο αργότερα οι Fogel, Walsh και Owens ανέπτυξαν την ιδέα του *Εξελικτικού Προγραμματισμού* (Evolutionary Programming), συστήματα που αποτελούν τις κύριες υποκατηγορίες εξελικτικών αλγορίθμων μαζί με τους γενετικούς.

Οι γενετικοί αλγόριθμοι αναπτύχθηκαν στις αρχές της δεκαετίας του 1970 από τον John Holland και τους συνεργάτες του στο Πανεπιστήμιο του Michigan και διαδόθηκαν ευρέως από το βιβλίο του “Adaptation in Natural and Artificial Systems”. Η διαφορά σε σχέση με τις Εξελικτικές Στρατηγικές και τον Εξελικτικό Προγραμματισμό είναι ότι ο Holland δεν επεδίωξε να εφαρμόσει τους ΓΑ σε ένα συγκεκριμένο πρόβλημα, αλλά τον ενδιέφερε να εισάγει τον μηχανισμό της εξέλιξης όπως συμβαίνει στη φύση, στον υπολογιστή [Mitc98]. Ο ίδιος ανέπτυξε τη μέθοδο με την οποία αναπαράγονται οι γενιές και προκύπτουν οι νέοι πληθυσμοί, και όρισε τους βασικούς τελεστές επιλογής, διασταύρωσης και μετάλλαξης που απαιτούνται για αυτή τη διαδικασία, και θεωρείται ως εκ τούτου ο “πατέρας” των ΓΑ.

Ωστόσο, η ουσιαστική έρευνα στο πεδίο των ΓΑ ξεκίνησε στα μέσα της δεκαετίας του 1980, όταν πραγματοποιήθηκε το πρώτο Διεθνές Συνέδριο για Γενετικούς Αλγορίθμους στο Πίτσμπουργκ της Πενσυλβάνιας. Έκτοτε το Διεθνές Συνέδριο λαμβάνει χώρα μία φορά κάθε χρόνο, έχοντας αυξήσει σημαντικά τις ερευνητικές δραστηριότητες γύρω από τους ΓΑ οι οποίοι αποτελούν πλέον μία ιδιαίτερα δημοφιλή κατηγορία.

Το πεδίο εφαρμογής τους αποτελείται κυρίως από προβλήματα βελτιστοποίησης, στα οποία δεν υπάρχει αναλυτική μέθοδος αναζήτησης που να εγγυάται την εύρεση επιθυμητής λύσης. Αυτό συμβαίνει συνήθως σε προβλήματα με πολύ μεγάλο αριθμό παραμέτρων/διαστάσεων όπου το μέγεθος του χώρου αναζήτησης δεν επιτρέπει την αξιολόγηση όλων των δυνατών συνδυασμών και το σχηματισμό του βέλτιστου.

Η βασική διαφορά των ΓΑ από άλλες μεθόδους επίλυσης αυτού του είδους προβλημάτων είναι ότι διατηρούν ανά πάσα στιγμή έναν πληθυσμό πιθανών λύσεων, ο οποίος τους επιτρέπει να ελέγχουν ταυτόχρονα πολλές κατευθύνσεις του *χώρου καταστάσεων* (search space). Αντίθετα σε τεχνικές στις

οποίες σε κάθε βήμα εξετάζεται μία μεμονωμένη λύση, είναι πιο εύκολο ο αλγόριθμος να περιοριστεί σε μία συγκεκριμένη κατεύθυνση λύσεων και να εγκλωβιστεί σε τοπικά μέγιστα (ή ελάχιστα ανάλογα με τη φύση του προβλήματος) [Γε99].

3.2 Βασικά Συστατικά και Υλοποίηση

3.2.1 Περιγραφή και Δομή

Σε έναν γενετικό αλγόριθμο στόχος είναι ο προσδιορισμός των τιμών κάποιων παραμέτρων μέσα από μια διαδικασία που ακολουθεί το μοντέλο της βιολογικής εξέλιξης . Αρχικά οι παράμετροι κωδικοποιούνται ώστε να μπορούν να αναπαρασταθούν από μία ακολουθία δυαδικών ψηφίων. Στη συνέχεια δημιουργείται (συνήθως με τυχαίο τρόπο) ένας αρχικός πληθυσμός από άτομα καθένα από τα οποία περιλαμβάνει μία τέτοια ακολουθία και αποτελεί λύση του προβλήματος. Τα άτομα αξιολογούνται σύμφωνα με μία *συνάρτηση ποιότητας* (fitness function), προσαρμοσμένη στο συγκεκριμένο πρόβλημα, και επιβιώνουν αυτά με την υψηλότερη τιμή, τα οποία τελικά αναπαράγονται και δημιουργούν την επόμενη γενιά λύσεων. Αυτή η διαδικασία επαναλαμβάνεται μέχρι να βρεθεί ένα άτομο με ικανοποιητική τιμή της συνάρτησης ποιότητας, που θα είναι και η τελική λύση του προβλήματος.

Η κωδικοποιημένη αναπαράσταση μίας πιθανής λύσης σε έναν ΓΑ αποτελείται από τα ακόλουθα δομικά στοιχεία [Kuma16]:

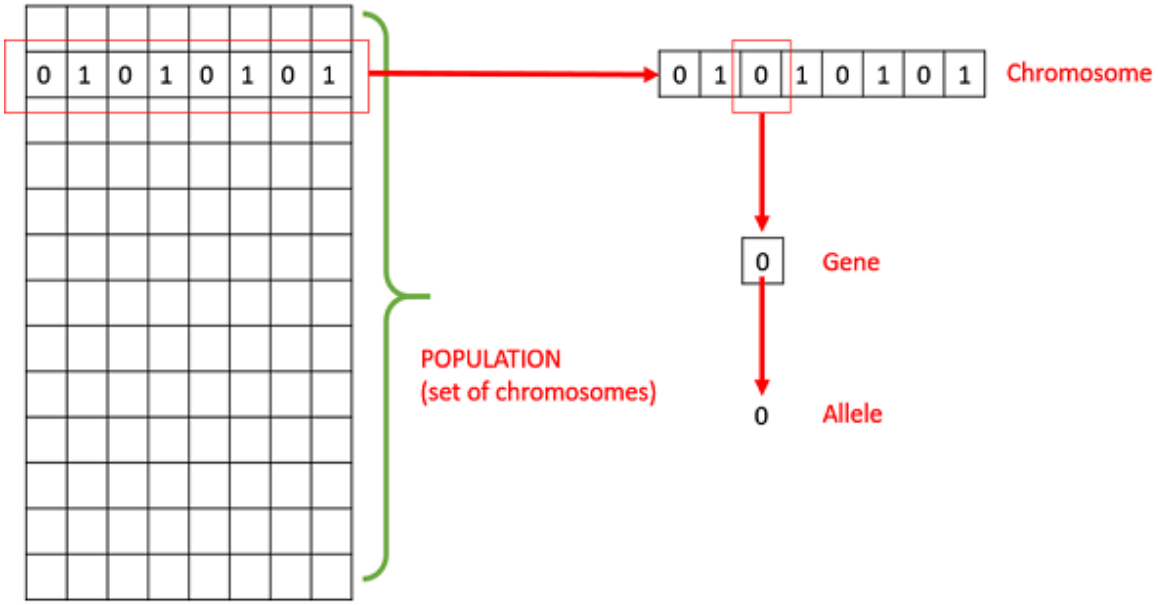
Γονίδιο (Gene) Είναι το πιο απλό δομικό συστατικό του γενετικού αλγορίθμου και είναι αυτό που περιέχει την πληροφορία, καθώς κάθε παράμετρος του προβλήματος που θέλουμε να επιλύσουμε κωδικοποιείται κατάλληλα και αντιστοιχίζεται σε ένα γονίδιο. Συνήθως τα γονίδια είναι δυαδικά στοιχεία τα οποία μπορούν να πάρουν τις τιμές 0 ή 1, όμως αυτό δεν είναι απαραίτητο. Ανάλογα με την υλοποίηση του αλγορίθμου τα γονίδια είναι δυνατόν να παίρνουν περισσότερες τιμές ή ακόμα και τιμές άλλου τύπου (πχ πραγματικές) όμως σε κάθε υλοποίηση πρέπει όλα τα γονίδια να είναι του ίδιου τύπου.

Χρωμόσωμα (Chromosome) Είναι μία ακολουθία γονιδίων, που αποτελείται από τουλάχιστον ένα γονίδιο. Στη γενική περίπτωση ένας γενετικός αλγόριθμος περιλαμβάνει έναν σημαντικό αριθμό γονιδίων τα οποία ομαδοποιούνται στα χρωμοσώματα. Συνήθως όλη η απαραίτητη πληροφορία περιλαμβάνεται σε ένα χρωμόσωμα, όμως είναι πιθανόν να υπάρχουν περισσότερα από ένα είτε για λόγους οργάνωσης είτε αν υπάρχουν γονίδια με διαφορετικό φάσμα τιμών, τα οποία δεν μπορούν να κωδικοποιηθούν στο ίδιο χρωμόσωμα. Κάθε χρωμόσωμα μπορεί να έχει διαφορετικό μήκος (δηλαδή να αποτελείται από διαφορετικό αριθμό γονιδίων) όπως και διαφορετικό φάσμα τιμών για τα γονίδιά του αλλά όλα πρέπει να περιλαμβάνουν γονίδια του ίδιου τύπου (πχ δυαδικά, ακέραια κλπ).

Γενότυπος (Genotype) Είναι το σύνολο των χρωμοσωμάτων που κωδικοποιούν την πληροφορία και ουσιαστικά κάθε γενότυπος αντιστοιχεί σε μία πιθανή λύση του προβλήματος, με παραμέτρους τις τιμές των χρωμοσωμάτων του.

Φαινότυπος (Phenotype) Είναι κάθε συνδυασμός ενός γενοτύπου με την συνάρτηση ποιότητας και αντιστοιχεί στην αποκωδικοποιημένη πληροφορία.

Γενιά (Generation) Αποτελείται από έναν αριθμό ατόμων/λύσεων με κοινούς προγόνους και περίοδο δημιουργίας. Η πρώτη γενιά δημιουργείται με κάποιον ξεχωριστό μηχανισμό (συνήθως τυχαία) και συνεπώς τα άτομα που την απαρτίζουν δεν έχουν προγόνους. Ο αριθμός των γενεών δεν είναι σταθερός και εξαρτάται τόσο από τον ρυθμό της εξέλιξης όσο και από το κριτήριο τερματισμού, με αποτέλεσμα να διαφέρει ακόμα και μεταξύ διαφορετικών στιγμιοτύπων της ίδιας υλοποίησης, καθώς πάντα υπάρχει ο τυχαίος παράγοντας που μπορεί να επηρεάσει την ταχύτητα της σύγκλισης στην τελική λύση.



Σχήμα 3.1: Η δομή των ατόμων ενός πληθυσμού

3.2.2 Βασικά Συστατικά

Όλοι οι γενετικοί αλγόριθμοι, ανεξάρτητα από τις πιθανές παραλλαγές και τις διαφορές που ενδέχεται να παρουσιάζουν μεταξύ τους, έχουν ορισμένα κοινά βασικά συστατικά, τα οποία είναι απαραίτητα σε οποιαδήποτε μορφή υλοποίησης. Αυτά είναι [Carr14]:

Πληθυσμός (Population) Είναι το σύνολο των ατόμων ανά γενιά κατά την διαδικασία της εξέλιξης. Ο πληθυσμός παραμένει σταθερός κατά την εκτέλεση του ΓΑ.

Συνάρτηση Ποιότητας (Fitness Function) Δίνει ένα μέτρο που υποδεικνύει πόσο "ορθά" προσεγγίζει μία πιθανή λύση την επιθυμητή. Συνήθως είναι κανονικοποιημένη στο διάστημα $[0, 1]$ με τη σύμβαση οι μεγαλύτερες τιμές να υποδηλώνουν αποτελεσματικότερη συμπεριφορά. Η συνάρτηση ποιότητας είναι νευραλγικής σημασίας για την αποδοτικότητα του γενετικού αλγορίθμου γιατί καθορίζει σε σημαντικό βαθμό ποια άτομα θα επιβιώσουν και θα αναπαραχθούν στις επόμενες γενιές.

Επιλογή (Selection) Η επιλογή των ατόμων-γονέων που θα συνδυαστούν για την παραγωγή των απογόνων μπορεί να γίνει με πολλούς διαφορετικούς τρόπους που θα εξεταστούν αναλυτικότερα παρακάτω. Σχεδόν σε όλες τις γνωστές μεθόδους όμως, λαμβάνεται υπόψιν η συνάρτηση ποιότητας, κάτι που σημαίνει ότι όσο μεγαλύτερος είναι ο *βαθμός καταλληλότητας* (fitness) ενός ατόμου, τόσο πιθανότερο είναι να επιλεχθεί ως γονέας.

Διασταύρωση (Crossover) Είναι η διαδικασία κατά την οποία δύο γενότυποι-γονείς συνδυάζονται για την δημιουργία ενός γενοτύπου-απογόνου. Ο απόγονος κληρονομεί σε αυτό το στάδιο γονίδια και από του δύο γονείς με αναλογία που μπορεί να ποικίλλει ανάλογα με την μέθοδο υλοποίησης. Το ποσοστό των γονιδίων που κληρονομούνται από κάθε γονέα μπορεί να διαφέρει ακόμα και σε διαφορετικές εφαρμογές της ίδιας μεθόδου, καθώς συνήθως χρησιμοποιούνται πιθανοτικά μοντέλα. Στην περίπτωση που ο γενότυπος περιλαμβάνει χρωμοσώματα με διαφορετικό μήκος, διασταυρώνονται μεταξύ τους μόνο χρωμοσώματα που έχουν την ίδια σειρά στους γενότυπους-γονείς, δηλαδή τα χρωμοσώματα που κωδικοποιούν ουσιαστικά την ίδια παράμετρο του προβλήματος.

Μετάλλαξη (Mutation) Η μετάλλαξη είναι το τελικό στάδιο της αναπαραγωγής. Εκτελείται τυχαία σε κάποιους από τους απογόνους αμέσως μετά την ολοκλήρωση της διασταύρωσης και αλλάζει κάποια γονίδια τους. Τα γονίδια αυτά επιλέγονται κατά βάση τυχαία, ενώ το ποσοστό τους επί του συνόλου των γονιδίων του χρωμοσώματος ανήκει στο εύρος $[1\% - 10\%]$, έτσι ώστε η τυχαιότητα που εισάγεται να μην επηρεάσει αρνητικά την απόδοση του αλγορίθμου.

3.2.3 Υλοποίηση

Αφού κωδικοποιηθεί το πρόβλημα και αντιστοιχιστούν οι προς προσδιορισμό παράμετροι σε γονίδια και χρωμοσώματα, ακολουθείται η εξής διαδικασία [Mitc98]:

1. Το πρώτο βήμα είναι η αρχικοποίηση του πληθυσμού. Ο αλγόριθμος δημιουργεί (κατά βάση με τυχαίο τρόπο) την πρώτη γενιά του πληθυσμού αρχικοποιώντας γενότυπους με γονίδια τυχαίων τιμών.

2. Για κάθε ένα άτομο της γενιάς εκτελείται η συνάρτηση ποιότητας που του αντιστοιχίζει ένα μέτρο ανάλογα με το πόσο προσεγγίζει την επιθυμητή λύση.

3. Όταν εκτιμηθούν τα μέτρα όλων των λύσεων της γενιάς, ξεκινάει η αναπαραγωγή της επόμενης, με την εφαρμογή κατά σειρά των τελεστών Επιλογής, Διασταύρωσης και Μετάλλαξης.

4. Τα δύο παραπάνω βήματα επαναλαμβάνονται ώσπου να ικανοποιηθεί το κριτήριο τερματισμού του αλγορίθμου και όταν η διαδικασία ολοκληρωθεί, επιστρέφεται η βέλτιστη λύση/γενότυπος που έχει παραχθεί μέχρι εκείνη τη στιγμή.

Σχήμα 3.2: Διαγραμματική υλοποίηση Γενετικών Αλγορίθμων

3.3 Τελεστές (Operators)

Σε αυτό το τμήμα θα εξεταστούν αναλυτικά οι βασικοί τελεστές των Γενετικών Αλγορίθμων. Τρεις είναι οι κύριες κατηγορίες τελεστών:

1. Επιλογής (Selectors)

2. Διασταύρωσης (Crossover)

3. Μετάλλαξης (Mutation)

3.3.1 Τελεστές Επιλογής (Selectors)

3.3.1.1 Επιλογή Γονέων

Η επιλογή των ατόμων που θα συνδυαστούν για την αναπαραγωγή των απογόνων είναι καθοριστική για τον γενετικό αλγόριθμο και έχει αποτελέσει πεδίο έρευνας για πολλά χρόνια. Ως εκ τούτου έχουν αναπτυχθεί πολλοί *τελεστές επιλογής* (selectors), χωρίς όμως να υπάρχει κάποιος αποδεδειγμένα πιο αποτελεσματικός για όλα τα προβλήματα. Φυσικά, βασική προϋπόθεση για όλες τις μεθόδους είναι να έχει εκτελεστεί πρώτα η συνάρτηση ποιότητας και να έχει υπολογιστεί ο βαθμός καταλληλότητας των ατόμων. Παρακάτω αναφέρονται οι σημαντικότεροι selectors [Alab12, Blic96].

Monte Carlo Selector Η μέθοδος επιλογής *monte carlo* είναι η απλούστερη μέθοδος καθώς επιλέγει τα άτομα εντελώς τυχαία. Προφανώς η απόδοσή της είναι πολύ χαμηλή και για αυτό δεν εφαρμόζεται στην πράξη, μπορεί όμως να χρησιμοποιηθεί ως πρότυπο για τον έλεγχο και την αξιολόγηση άλλων μεθόδων.

Tournament Selector Ο *tournament selector* ταξινομεί ένα τυχαίο υποσύνολο των ατόμων του πληθυσμού, με βάση το fitness τους και επιλέγει το άτομο με το υψηλότερο fitness. Έπειτα επαναλαμβάνει αυτή τη διαδικασία $n-1$ φορές ώστε να επιλεγούν τα n άτομα που θα χρησιμοποιηθούν για την αναπαραγωγή. Το μέγεθος των υποσυνόλων μπορεί να ποικίλλει και είναι μία παράμετρος που επίσης απαιτεί μελέτη, καθώς μπορεί να επηρεάσει το ποσοστό των πιο "αδύναμων" ατόμων που θα προκριθούν. Χαρακτηριστικό αυτής της μεθόδου είναι ότι ο γενότυπος με το μεγαλύτερο fitness θα επιλεγεί σίγουρα και αυτός με το μικρότερο θα απορριφθεί.

Truncation Selector Ο *truncation selector* (μέθοδος περικοπής) έχει την ακόλουθη πολύ απλή λειτουργία. Αρχικά ταξινομεί όλα τα άτομα σε φθίνουσα σειρά με βάση το βαθμό καταλληλότητάς τους και στη συνέχεια επιλέγει τα n πρώτα. Η μέθοδος αυτή οδηγεί γρήγορα σε άτομα με υψηλό fitness, όμως έχει το μειονέκτημα ότι μπορεί να αποκλείσει νωρίς χρήσιμα γονίδια αν τα χρωμοσώματα που τα περιέχουν δεν επιλεγούν, και γι' αυτό δεν χρησιμοποιείται συχνά.

Roulette-Wheel Selector Ο *roulette-wheel* είναι ένας από τους πιο ευρέως χρησιμοποιούμενους selectors, ο οποίος ανήκει στην κατηγορία των πιθανοτικών μεθόδων επιλογής. Για κάθε ένα άτομο/λύση i, υπολογίζει την πιθανότητα:

$$P(i) = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{3.1}$$

όπου f_i είναι ο βαθμός καταλληλότητας του ατόμου i και N ο αριθμός των ατόμων του πληθυσμού και έπειτα δημιουργεί n τυχαίους αριθμούς στο διάστημα $[0, 1]$. Οι πιθανότητες που υπολογίστηκαν χρησιμοποιούνται για το σχηματισμό των εξής διαστημάτων:

$$[0, P(1)], [P(1), P(1)+P(2)], \ldots, [P(1)+P(2)+\ldots+P(n-1), P(1)+P(2)+\ldots+P(n) = 1].$$

Ανάλογα με το διάστημα στο οποίο ανήκει ο κάθε ένας από τους τυχαίους αριθμούς, επιλέγεται το αντίστοιχο άτομο για να αναπαραχθεί στην επόμενη γενιά.

Linear Rank Selector Ο *linear rank* είναι ένας επίσης πιθανοτικός selector. Αρχικά ταξινομεί τα άτομα με βάση το fitness τους και προσδίδει στο καθένα ένα *βαθμό* (rank) από 1 μέχρι N (όπου N το πλήθος των ατόμων), ξεκινώντας από το μικρότερο προς το μεγαλύτερο. Τελικά η πιθανότητα να επιλεγεί το i άτομο είναι:

$$P(i) = \frac{1}{N}\left(n^- + (n^+ - n^-) * \frac{\text{rank}(i) - 1}{N-1}\right) \tag{3.2}$$

Η πιθανότητα επιλογής του καλύτερου ατόμου (δηλαδή του ατόμου με τον μεγαλύτερο βαθμό καταλληλότητας) είναι $\frac{n^+}{N}$ και του χειρότερου $\frac{n^-}{N}$. Η υλοποίηση του είναι παρόμοια με αυτή του random-wheel selector και βασίζεται στη δημιουργία τυχαίων αριθμών με μόνη διαφορά τις διαφορετικές πιθανότητες που αντιστοιχούν στα άτομα/γενότυπους.

Exponential Rank Selector Στον *exponential selector* τα άτομα επίσης ταξινομούνται σε φθίνουσα σειρά και αντιστοιχίζονται σε τιμές από N έως 1. Η πιθανότητα του ατόμου i να επιλεγεί είναι:

$$P(i) = \frac{c^{N-\text{rank}(i)}}{\sum_{j=1}^{N} c^{N-\text{rank}(j)}} \tag{3.3}$$

όπου c παράμετρος που ανήκει στο διάστημα $[0, 1)$ και ορίζεται από τον σχεδιαστή. Όσο η τιμή της παραμέτρου μειώνεται, αυξάνεται η πιθανότητα επιλογής του καλύτερου ατόμου, ενώ όσο αυξάνεται, οι πιθανότητες όλων των ατόμων τείνουν να γίνουν ίσες.

Stochastic Universal Selector Ο *stochastic universal* είναι μία παραλλαγή του random-wheel που στοχεύει στην πιο ομαλή επιλογή των απογόνων για να αποφευχθεί η απότομη εξέλιξη, η οποία έχει μεν ταχύτερη σύγκλιση αλλά αποκλείει γρήγορα λύσεις με μικρό fitness, που ενδεχομένως περιέχουν κάποια χρήσιμα γονίδια. Για την πραγματοποίηση της επιλογής γίνεται ταξινόμηση των ατόμων σε φθίνουσα σειρά με βάση το fitness τους και σχηματίζονται τα διαστήματα :

$$[0, f(1)], [f(1), f(1) + f(2)], \ldots, [0 + f(1) + \ldots + f(n-1), 0 + f(1) + \ldots + f(n)].$$

Στη συνέχεια υπολογίζεται ο μέσος όρος μ των fitnesses και δημιουργείται ένας τυχαίος αριθμός r στο διάστημα $[0, \mu]$. Σε αυτόν τον τυχαίο αριθμό προστίθεται ο μέσος όρος και ανάλογα με το διάστημα στο οποίο ανήκει το άθροισμά τους επιλέγεται το αντίστοιχο άτομο. Έπειτα προστίθεται ο μέσος όρος στο προηγούμενο άθροισμα και αυτή η διαδικασία επαναλαμβάνεται $n-1$ φορές ώστε να επιλεχθούν και τα υπόλοιπα $n-1$ άτομα. Τονίζεται ότι το πρώτο άτομο έχει επιλεχθεί ως αντίστοιχο του διαστήματος που περιλαμβάνει τον τυχαίο αριθμό r, και εφόσον $r \leq \mu < f(1)$, το άτομο με το μεγαλύτερο fitness, δηλαδή το πρώτο άτομο μετά την ταξινόμηση, επιλέγεται πάντα.

3.3.1.2 Επιλογή Επιζώντων

Κάθε νέα γενιά σε έναν ΓΑ αποτελείται από άτομα-απογόνους που προκύπτουν από τον συνδυασμό ατόμων-γονέων της προηγούμενης γενιάς καθώς και από άτομα-επιζώντες τα οποία περνούν αμετάβλητα από την προηγούμενη γενιά στην επόμενη. Ο τρόπος με τον οποίο θα γίνει η επιλογή των επιζώντων της κάθε γενιάς μπορεί επίσης να ποικίλλει και είναι εξίσου κομβικός για την αποδοτικότητα του αλγορίθμου. Δύο είναι οι κυριότερες κατηγορίες μεθόδων επιλογής [Jeba13].

Ηλικιακή Επιλογή (Age-based Selection) Με αυτή τη μέθοδο προκρίνονται στην επόμενη γενιά, τα άτομα/λύσεις με τη μικρότερη ηλικία, δηλαδή αυτά που έχουν περάσει αυτούσια στις λιγότερες γενιές σε σχέση με τα υπόλοιπα. Ο βαθμός καταλληλότητας των ατόμων δεν λαμβάνεται υπόψιν.

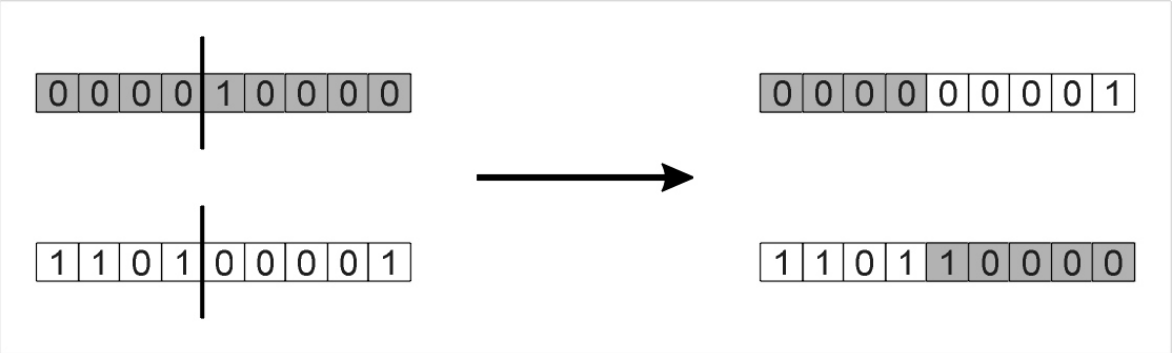
Επιλογή βασιμένη στον Βαθμό Καταλληλότητας (Fitness-based Selection) Είναι η πιο συχνά χρησιμοποιούμενη μέθοδος και στηρίζει την επιλογή των επιζώντων στο βαθμό καταλληλότητας τους. Συνήθως ακολουθεί την αρχή του ελιτισμού κατά την οποία το άτομο με τον υψηλότερο βαθμό καταλληλότητας επιβιώνει πάντα στην επόμενη γενιά, ώστε να εξασφαλιστεί ότι αυτή δεν θα είναι ποτέ χειρότερη από την προηγούμενη. Για την επιλογή των υπόλοιπων ατόμων μπορεί να χρησιμοποιηθεί οποιαδήποτε από τις τεχνικές που περιγράφηκαν παραπάνω και για την επιλογή των γονέων.

3.3.2 Τελεστές Διασταύρωσης (Crossover)

Παρακάτω παρουσιάζονται οι πιο σημαντικές τεχνικές διασταύρωσης για χρωμοσώματα με δυαδικά και με αριθμητικά γονίδια [Soni14, BUKH14].

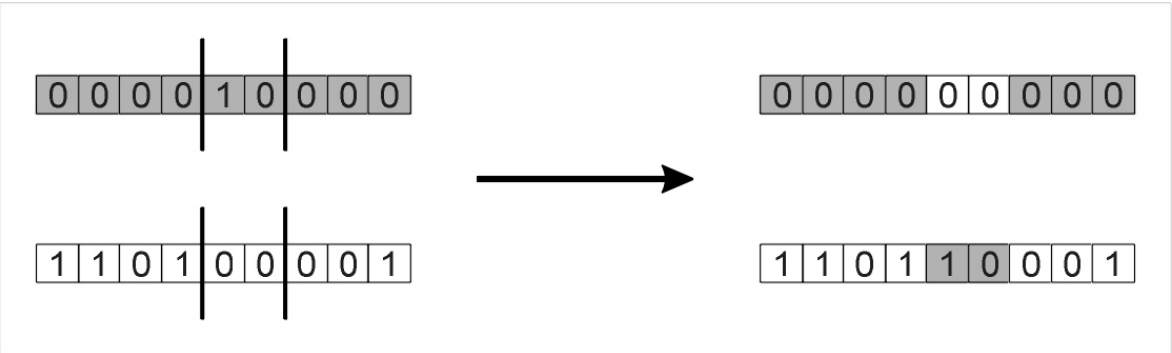
3.3.2.1 Δυαδικά Γονίδια

Single-point Crossover Σε αυτή την τεχνική διασταύρωσης, επιλέγεται τυχαία ένα σημείο μέσα στο χρωμόσωμα, και το χρωμόσωμα-απόγονος κληρονομεί όλα τα γονίδια του ενός χρωμοσώματος-γονέα μέχρι αυτό το σημείο και όλα τα γονίδια του άλλου γονέα από αυτό το σημείο και έπειτα. Το μειονέκτημα αυτής της τεχνικής είναι ότι καθυστερεί αρκετά την εξέλιξη και πρέπει να αναπτυχθούν πολλές γενιές για να προκύψουν άτομα-λύσεις που συνδυάζουν πολλά διαφορετικά γονίδια.



Σχήμα 3.3: Single-point crossover

Multi-point Crossover Επιλέγεται ένας αριθμός από σημεία μέσα στο χρωμόσωμα (συνήθως 2) και στα διαστήματα που ορίζονται από αυτά τα σημεία, ο απόγονος κληρονομεί ακολουθίες γονιδίων εναλλάξ από τα δύο χρωμοσώματα-γονείς. Επί της ουσίας το single-point crossover που περιγράφηκε παραπάνω είναι η τετριμμένη υποπερίπτωση του multi-point όπου επιλέγεται ένα μόνο σημείο.



Σχήμα 3.4: Multi-point crossover

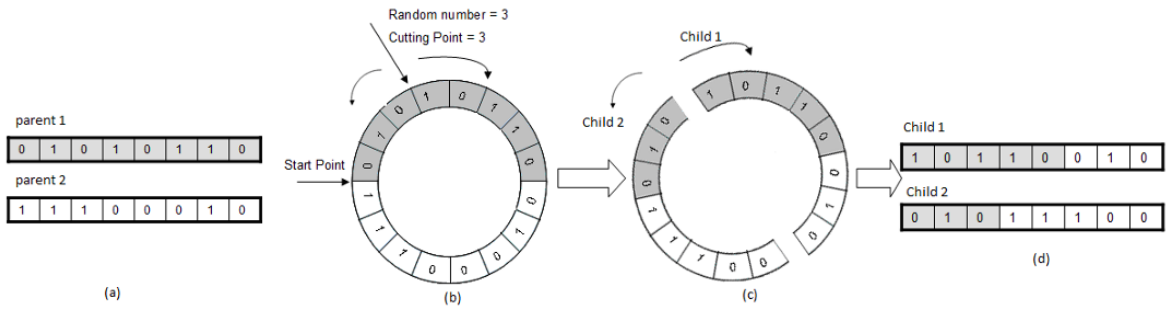
Uniform Crossover Είναι μία τεχνική παρόμοια με το multi-point crossover, με τη διαφορά ότι δεν υπάρχουν συγκεκριμένα διαστήματα στα οποία κληρονομούνται γονίδια από τον κάθε γονέα, αλλά είναι συγκεκριμένη η αναλογία. Αυτό πρακτικά σημαίνει ότι τα γονίδια των δύο χρωμοσωμάτων-γονέων εξετάζονται ένα προς ένα και κληρονομείται με πιθανότητα p το γονίδιο του ενός γονέα και με πιθανότητα $1-p$ το γονίδιο του άλλου. Η πιθανότητα p είναι τέτοια ώστε να κληρονομείται τελικά το επιθυμητό ποσοστό γονιδίων από τον κάθε γονέα.

Half-uniform Crossover Αποτελεί υποπερίπτωση του uniform crossover, όπου 50% των γονιδίων κληρονομούνται από τον ένα γονέα και 50% από τον άλλο.

Shuffle Crossover Πρόκειται για μία παραλλαγή του single-point crossover με στόχο να εξαλειφθεί η εξάρτηση από τη θέση του κάθε γονιδίου μέσα στο χρωμόσωμα, η οποία παίζει σημαντικό ρόλο στην κλασσική υλοποίηση που περιγράφηκε παραπάνω. Αυτή η τεχνική υλοποιείται σε τρία βήματα. Αρχικά αναδιατάσσει τυχαία τα γονίδια στα χρωμοσώματα των γονέων, πραγματοποιώντας όμως την ίδια αναδιάταξη και στους δύο γονείς ώστε τα γονίδια που βρίσκονται στην ίδια θέση μέσα στο χρωμόσωμα να αντιστοιχούν πάντα στην ίδια κωδικοποιημένη παράμετρο του προβλήματος. Έπειτα εκτελεί στα αναδιατεταγμένα χρωμοσώματα single-point crossover και τελικά χρησιμοποιεί την αντίστροφη αναδιάταξη για να επαναφέρει τα γονίδια στην αρχική τους θέση.

Reduced Sarrogate Crossover Άλλη μία παραλλαγή της μεθόδου single-point. Σε αυτή τη μέθοδο στόχος είναι να αποφευχθεί η διασταύρωση σε σημεία που τα γονίδια των δύο γονέων είναι ίδια, καθώς είναι περιττή. Η υλοποίηση της πραγματοποιείται σε δύο στάδια. Στο πρώτο στάδιο γίνεται σύγκριση όλων των γονιδίων των γονέων ένα προς ένα ώστε να αποκλειστούν τα σημεία των χρωμοσωμάτων όπου τα δύο γονίδια είναι ίδια, και στο δεύτερο επιλέγεται τυχαία ένα από τα σημεία στα οποία τα γονίδια είναι διαφορετικά και εκτελείται single-point crossover.

Ring Crossover Στο *ring crossover* τα χρωμοσώματα των γονέων αρχικά ενώνονται σχηματίζοντας ένα ενιαίο χρωμόσωμα σε μορφή δακτυλίου με τρόπο τέτοιο ώστε τα πρώτα γονίδια τους γονίδια να είναι διαδοχικά όπως και τα τελευταία σχηματίζοντας ουσιαστικά έναν δακτύλιο (Σχήμα 3.5). Ακολούθως επιλέγεται ένα τυχαίο σημείο του δακτυλίου στο οποίο γίνεται η πρώτη τομή, και το αντιδιαμετρικό του για την δεύτερη, ώστε τα δύο χρωμοσώματα που θα προκύψουν από τον διαχωρισμό να έχουν το ίδιο μήκος, το οποίο είναι φυσικά ίδιο με το μήκος των χρωμοσωμάτων-γονέων.



Σχήμα 3.5: Ring crossover

3.3.2.2 Αριθμητικά Γονίδια

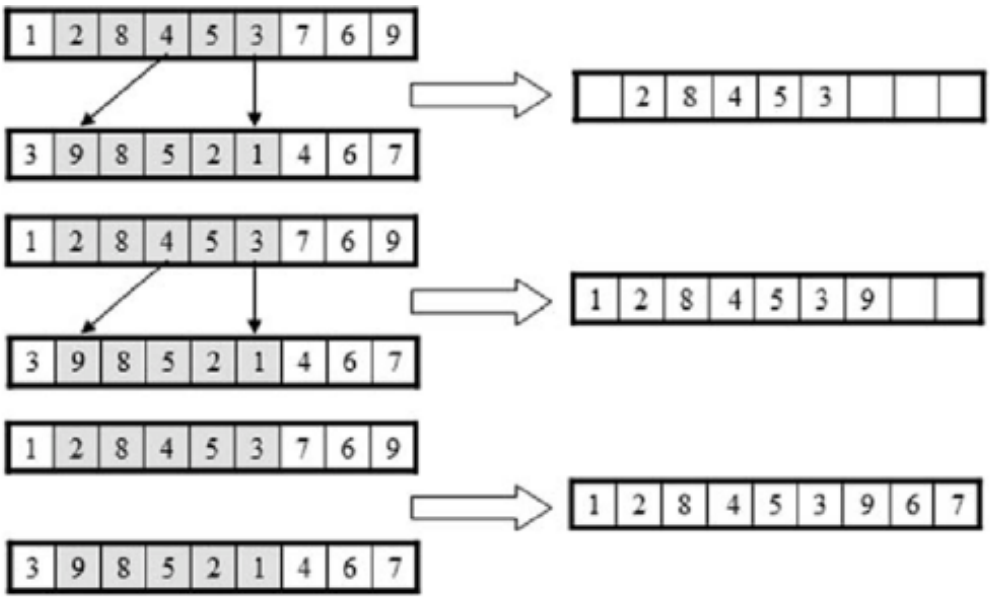
Σε αυτό το σημείο περιγράφονται οι κλασσικές τεχνικές διασταύρωσης για αριθμητικά γονίδια [Umba15, AlHa09].

Average Crossover Είναι μία από τις απλούστερες τεχνικές διασταύρωσης για γενότυπους που περιλαμβάνουν αριθμητικά γονίδια. Στο *average crossover* κάθε γονίδιο i του απογόνου παίρνει ως τιμή τον μέσο όρο των τιμών των αντίστοιχων γονιδίων i_1 και i_2 των γονέων του.

Flat Crossover Στην τεχνική διασταύρωσης *flat*, για κάθε γονίδιο του απογόνου δημιουργείται ένας τυχαίος αριθμός στο διάστημα που ορίζεται από τις τιμές των δύο γονιδίων των γονέων (της αντίστοιχης θέσης μέσα στο χρωμόσωμα).

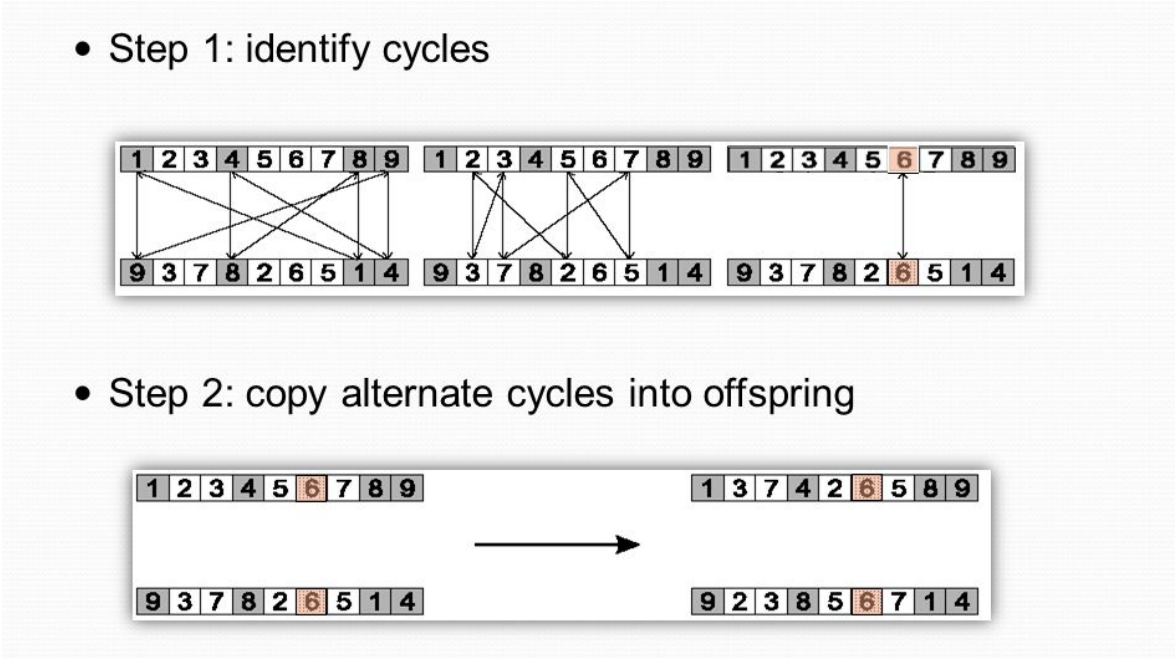
Partially Matched Crossover (PMX) Είναι μία από τις πιο διαδεδομένες μεθόδους διασταύρωσης για αριθμητικά γονίδια όπου απαιτείται όλα τα γονίδια του κάθε χρωμοσώματος να είναι διαφορετικά μεταξύ τους. Αρχικά, επιλέγονται δύο τυχαία σημεία στα χρωμοσώματα των γονέων και τα γονίδια που βρίσκονται ανάμεσα σε αυτά αντιγράφονται απευθείας από τον πρώτο γονέα στο χρωμόσωμα του απογόνου. Κατόπιν ελέγχονται ένα προς ένα τα γονίδια του δεύτερου γονέα που ανήκουν στο ίδιο διάστημα, έστω S, και για όσα από αυτά δεν έχουν περάσει ήδη στον απόγονο (δηλαδή δεν υπήρχαν γονίδια στο S με αυτή την τιμή στο χρωμόσωμα του πρώτου γονέα) ακολουθείται η παρακάτω διαδικασία.

Το γονίδιο i_2 (του δεύτερου γονέα) αντιστοιχίζεται στο γονίδιο i_1 του πρώτου. Έπειτα εντοπίζεται η θέση j του γονιδίου του δεύτερου γονέα που έχει την ίδια τιμή με το i_1, και αν το j δεν ανήκει στο διάστημα S, η τιμή του γονιδίου i_2 αντιγράφεται στη θέση j του χρωμοσώματος του απογόνου. Στην περίπτωση που το j ανήκει στο S, το γονίδιο j_2 αντιστοιχίζεται στο j_1 και επαναλαμβάνεται η προηγούμενη διαδικασία μέχρι να βρεθεί θέση n στο χρωμόσωμα του δεύτερου γονέα η οποία δεν ανήκει στο S. Σημειώνεται ότι όσες φορές και αν χρειαστεί να επαναληφθεί αυτό το βήμα, το γονίδιο που τελικά θα αντιγραφεί στο χρωμόσωμα του απογόνου είναι αυτό που είχε επιλεχθεί αρχικά από το διάστημα S του δεύτερου γονέα δηλαδή το i_2.



Σχήμα 3.6: Partially matched crossover

Cycle Crossover Το *cycle crossover* είναι μία τεχνική που επίσης εξασφαλίζει ότι δεν υπάρχουν δύο ίδια γονίδια στο χρωμόσωμα-απόγονο. Η λειτουργία του είναι αρκετά απλή. Ένα γονίδιο gi_1 του χρωμοσώματος του πρώτου γονέα αντιστοιχίζεται στο γονίδιο του χρωμοσώματος του δεύτερου γονέα που βρίσκεται στην ίδια θέση gi_2. Το γονίδιο του δεύτερου γονέα gi_2 αντιστοιχίζεται στο γονίδιο gj_1 του πρώτου γονέα που έχει την ίδια τιμή. Η διαδικασία αυτή ξεκινάει από το πρώτο γονίδιο του πρώτου γονέα g_{11} και ολοκληρώνεται όταν βρεθεί το γονίδιο gn_2 του δεύτερου γονέα με την ίδια τιμή. Τελικά όσα γονίδια προσπελάστηκαν κατά τη διάρκεια της παραπάνω διαδικασίας, αντιγράφονται στον απόγονο από τον πρώτο γονέα, ενώ τα υπόλοιπα από τον δεύτερο.



Σχήμα 3.7: Cycle crossover

Εκτός από αυτές τις τεχνικές μπορούν να χρησιμοποιηθούν και οι τεχνικές που ισχύουν για τα δυαδικά χρωμοσώματα με την κατάλληλη προσαρμογή.

3.3.3 Τελεστές Μετάλλαξης (Mutation)

Στόχος της μετάλλαξης είναι να εισαχθεί το στοιχείο της τυχαιότητας στην επόμενη γενιά ώστε να γίνει εξερεύνηση σε μεγαλύτερο χώρο καταστάσεων. Ωστόσο η πιθανότητα με την οποία ένα άτομο θα υποστεί μετάλλαξη πρέπει να είναι αρκετά μικρή, διαφορετικά τα άτομα-λύσεις θα καταλήξουν να κινούνται σε έναν τυχαίο χώρο καταστάσεων και η λειτουργία του αλγορίθμου θα εκφυλιστεί. Για τον ίδιο λόγο, είναι μικρός και ο αριθμός των γονιδίων του επιλεγμένου χρωμοσώματος που αλλάζουν κάθε φορά.

Για να είναι η διαδικασία της μετάλλαξης αποδεκτή πρέπει να πληροί ορισμένες προϋποθέσεις που εξασφαλίζουν ότι δεν θα αλλοιωθούν οι βασικές αρχές ενός γενετικού αλγορίθμου.

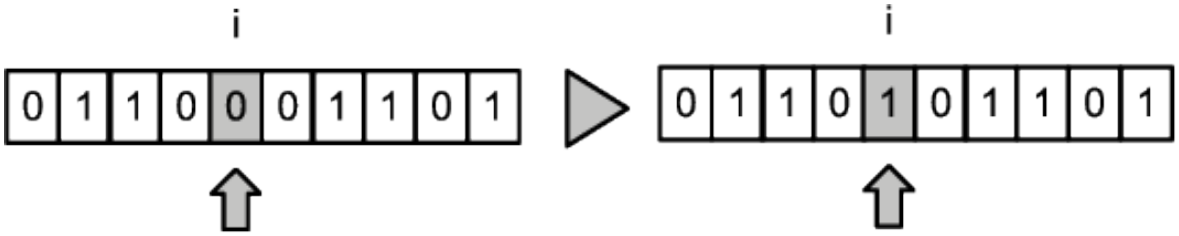
Κατ' αρχάς πρέπει όλες οι πιθανές καταστάσεις του υπό εξερεύνηση χώρου να είναι προσπελάσιμες ανά πάσα στιγμή έστω και με πολύ μικρή πιθανότητα. Σε περίπτωση που η μετάλλαξη εισάγει περιορισμούς που δεν ικανοποιούν αυτή την συνθήκη υπάρχει το ενδεχόμενο να μην βρεθεί ποτέ η βέλτιστη λύση, καθώς μπορεί να έχει αποκλειστεί από το χώρο αναζήτησης. Η συγκεκριμένη περίπτωση απαιτεί την ορθή κατανόηση του προβλήματος και την επιλογή της κατάλληλης μεθόδου μετάλλαξης από την αρχή, καθώς δεν γίνεται εύκολα ανιχνεύσιμη κατά την εκτέλεση του αλγορίθμου ή μετά την ολοκλήρωσή του.

Η δεύτερη σημαντική προϋπόθεση είναι η μετάλλαξη να μην ευνοεί την αναζήτηση προς μία συγκεκριμένη κατεύθυνση, σε προβλήματα που δεν υπάρχουν αντίστοιχοι περιορισμοί. Αυτό γίνεται πιο εύκολα αντιληπτό αν καθώς εξελίσσεται ο αλγόριθμος προκύπτουν πολλές παρόμοιες λύσεις, οι οποίες δεν προσεγγίζουν την επιθυμητή, κάτι που όμως δεν εγγυάται προβληματική μετάλλαξη, καθώς μπορεί να οφείλεται και σε άλλους παράγοντες.

Η μετάλλαξη μπορεί να υλοποιηθεί με διαφορετικές τεχνικές, ανάλογα και με τη φύση του προβλήματος, οι βασικότερες από τις οποίες παρουσιάζονται στη συνέχεια [Soni14, Kram17, DaRo14].

Bit Flip Mutation Είναι μία μέθοδος μετάλλαξης για προβλήματα με δυαδική κωδικοποίηση. Ένας μικρός αριθμός από bits επιλέγονται τυχαία μέσα στο χρωμόσωμα και αλλάζουν στο συμπλήρωμα

τους $(0 \Rightarrow 1, 1 \Rightarrow 0)$. Συνήθως το κάθε bit i έχει πιθανότητα να μεταλλαχθεί $P(i) = \frac{1}{N}$ για χρωμόσωμα με N bits.

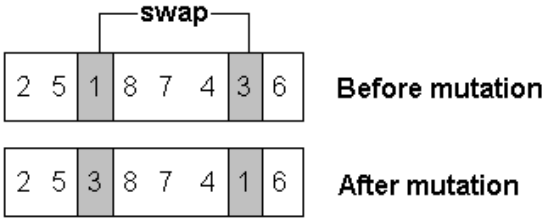


Σχήμα 3.8: Bit-flip mutation

Uniform Mutation (Random Resseting) Βασίζεται στη λογική του bit flip mutation, την οποία επεκτείνει για προβλήματα με κωδικοποίηση ακέραιων αριθμών. Σε αυτή την περίπτωση, τα επιλεγμένα γονίδια παίρνουν τυχαία μία από τις δυνατές τιμές που έχουν οριστεί κατά την κωδικοποίηση.

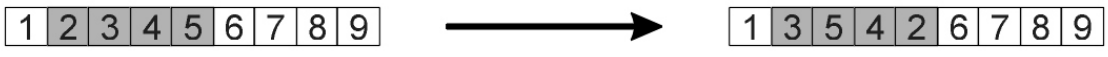
Boundary Mutation Χρησιμοποιείται σε προβλήματα με αριθμητική (ακέραια ή πραγματική) κωδικοποίηση και αντικαθιστά το γονίδιο προς μετάλλαξη είτε με την μικρότερη είτε με την μεγαλύτερη δυνατή τιμή τυχαία.

Swap Mutation Σε αυτή τη μέθοδο γίνεται ανταλλαγή δύο τυχαίων γονιδίων μέσα στο χρωμόσωμα. Εφαρμόζεται σε προβλήματα με *κωδικοποίηση μετάθεσης* (permutation encoding) όπου αναζητείται η βέλτιστη σειρά των γονιδίων μέσα στο χρωμόσωμα.



Σχήμα 3.9: Swap mutation

Scramble Mutation Εφαρμόζεται επίσης σε προβλήματα με κωδικοποίηση μετάθεσης και είναι παρόμοιο με το swap mutation με τη διαφορά ότι αντί για δύο γονίδια επιλέγεται ένα μπλοκ, και τα γονίδια που περιλαμβάνονται σε αυτό αναδιατάσσονται τυχαία.



Σχήμα 3.10: Scramble mutation

Inverse Mutation Είναι μία υποπερίπτωση του scramble mutation, στην οποία η σειρά των γονιδίων του επιλεγμένου μπλοκ δεν αλλάζει τυχαία, αλλά αντιστρέφεται.

Insertion Mutation Επιλέγεται ένα γονίδιο από το χρωμόσωμα και μετατίθεται σε μία άλλη τυχαία θέση, μετακινώντας και όλα τα γονίδια που βρίσκονται μεταξύ της αρχικής και της τελικής του θέσης, μία θέση δεξιά ή αριστερά ανάλογα με την κατεύθυνση της κίνησης του.

Displacement Mutation Λειτουργεί όπως το insertion mutation μετακινώντας ένα μπλοκ γονιδίων.

3.4 Κριτήρια Τερματισμού

Όπως έχει ήδη αναφερθεί το μέγεθος του πληθυσμού παραμένει σταθερό σε όλες τις γενιές και δεν μειώνεται κατά την εξέλιξη ενός γενετικού αλγορίθμου. Αυτό σημαίνει ότι ο αλγόριθμος μπορεί να συνεχίσει να εκτελείται επ' άπειρον αν δεν οριστεί από πριν κάποια συνθήκη, που όταν ικανοποιηθεί, να τον τερματίζει. Αυτές οι συνθήκες ονομάζονται κριτήρια τερματισμού, και ανάλογα με το πρόβλημα μπορούν να έχουν διαφορετική υλοποίηση. Επίσης μπορούν να οριστούν περισσότερα από ένα κριτήρια τερματισμού, οπότε ο ΓΑ θα ολοκληρωθεί όταν ικανοποιηθεί το πρώτο από αυτά.

Τα πιο δημοφιλή κριτήρια τερματισμού είναι τα ακόλουθα [Jain01]:

Προκαθορισμένης Γενιάς Είναι το απλούστερο κριτήριο τερματισμού το οποίο ικανοποιείται όταν παραχθεί ένας συγκεκριμένος αριθμός γενιών που έχει οριστεί ως κατώφλι. Συνήθως εφαρμόζεται συνδυαστικά μαζί με κάποιο άλλο κριτήριο, το οποίο ενδέχεται να μην μπορέσει να ικανοποιηθεί, για να εξασφαλίσει ότι ο αλγόριθμος θα τερματίσει σίγουρα.

Σταθερού Βαθμού Καταλληλότητας Συνήθως στα πρώτα στάδια ενός ΓΑ παρατηρείται σημαντική αύξηση του βαθμού καταλληλότητας των ατόμων/λύσεων από γενιά σε γενιά, η οποία μειώνεται σταδιακά καθώς οι γενιές αυξάνονται. Πολλές φορές, μάλιστα, από ένα σημείο και μετά το fitness της βέλτιστης λύσης σταματάει να βελτιώνεται και παραμένει σταθερό. Το κριτήριο σταθερού βαθμού καταλληλότητας αντιμετωπίζει αυτό το φαινόμενο τερματίζοντας τον αλγόριθμο αν το μέγιστο fitness παραμένει σταθερό για έναν συγκεκριμένο αριθμό συνεχόμενων γενιών θεωρώντας ότι δεν υπάρχουν άλλα περιθώρια βελτίωσης και η καλύτερη δυνατή λύση έχει ήδη βρεθεί.

Χρονικού Ορίου Εξέλιξης Σε αυτή τη μέθοδο ορίζεται ένα χρονικό όριο το οποίο όταν ξεπεραστεί σηματοδοτεί την ολοκλήρωση της εκτέλεσης του ΓΑ. Αυτό το κριτήριο εφαρμόζεται σε χρονικά ευαίσθητα περιβάλλοντα όπου είναι αναγκαία η εύρεση λύσης μέσα σε ένα συγκεκριμένο χρονικό πλαίσιο.

Ορίου Βαθμού Καταλληλότητας Τερματίζει τον αλγόριθμο όταν ξεπεραστεί ένα ελάχιστο ή μέγιστο κατώφλι βαθμού καταλληλότητας, για προβλήματα ελαχιστοποίησης και μεγιστοποίησης αντίστοιχα, το οποίο έχει οριστεί στην αρχή. Για την αποτελεσματική εφαρμογή αυτού του κριτηρίου είναι απαραίτητο να είναι εκ των προτέρων γνωστό το αναμενόμενο διάστημα στο οποίο θα ανήκει η βέλτιστη λύση, διαφορετικά η τελική λύση δεν θα προσεγγίζει την επιθυμητή.

Σύγκλισης Πληθυσμού Ο αλγόριθμος ολοκληρώνεται όταν σε μία γενιά ο μέσος όρος του βαθμού καταλληλότητας όλων των ατόμων/λύσεων απέχει από τον μέγιστο, λιγότερο από μία προκαθορισμένη τιμή (σε μορφή ποσοστού).

Μέγιστου-Ελαχίστου Είναι μία μέθοδος που επίσης εξετάζει το σύνολο του πληθυσμού της κάθε γενιάς και τερματίζει τον αλγόριθμο αν η διαφορά μεταξύ της καλύτερης και της χειρότερης λύσης είναι μικρότερη από το επιλεγμένο κατώφλι.

3.5 Βασικά Χαρακτηριστικά – Πλεονεκτήματα

Τα βασικότερα χαρακτηριστικά των Γενετικών Αλγορίθμων είναι τα εξής [Γε99]:

1. Επιλύουν δύσκολα προβλήματα για τα οποία δεν υπάρχει γνωστή αναλυτική μέθοδος. Οι κυριότερες εφαρμογές των ΓΑ είναι σε πεδία με πολλές διαστάσεις/παραμέτρους που καθιστούν μη αποδοτική ή ακόμα και υπολογιστικά αδύνατη την εφαρμογή πολλών άλλων μεθόδων, που χρησιμοποιούνται για την επίλυση τέτοιας φύσης προβλημάτων.

2. Μπορούν να εφαρμοστούν σε πολύ μεγάλο εύρος προβλημάτων. Σχεδόν οποιοδήποτε πρόβλημα μπορεί να προσεγγιστεί από έναν γενετικό αλγόριθμο με την κατάλληλη κωδικοποίηση. Φυσικά αυτό δεν σημαίνει ότι οι ΓΑ αποτελούν βέλτιστη επιλογή για κάθε είδος προβλήματος, καθώς ανάλογα με τις εκάστοτε απαιτήσεις είναι πιθανό να υπάρχουν πιο αποτελεσματικές μέθοδοι, ωστόσο σε αυτές τις περιπτώσεις μπορούν να χρησιμοποιηθούν ως μέτρο σύγκρισης.

3. Ενδείκνυνται για παράλληλη υλοποίηση. Η φύση των ΓΑ προσφέρει τη δυνατότητα εκμετάλλευσης της παραλληλίας που διαθέτουν οι σύγχρονες υπολογιστικές μηχανές στο μέγιστο, καθώς σε κάθε γενιά εξετάζεται ένα σύνολο ατόμων που είναι ανεξάρτητα μεταξύ τους και συνεπώς μπορούν να εκτελούνται παράλληλα οι απαραίτητοι υπολογισμοί.

4. Εξερευνούν ταυτόχρονα πολλές διαφορετικές κατευθύνσεις του χώρου αναζήτησης. Αυτό είναι ένα σημαντικό πλεονέκτημα των ΓΑ έναντι των περισσότερων μεθόδων επίλυσης παρόμοιων προβλημάτων, που εξερευνούν λύσεις προς μία συγκεκριμένη κατεύθυνση σε κάθε στάδιο της εκτέλεσης.

5. Χρησιμοποιούν πιθανοτικές μεθόδους σε διάφορα σημεία της υλοποίησής τους. Με αυτόν τον τρόπο καταλήγουν πιο εύκολα σε πιθανές λύσεις που δεν θα ήταν τόσο προφανείς αν βασίζονταν μόνο σε ντετερμινιστικές τεχνικές, χωρίς να υπάρχει το στοιχείο της τυχαιότητας.

6. Είναι ευέλικτοι, δηλαδή προσαρμόζονται σχετικά εύκολα στα διάφορα *προγραμματιστικά περιβάλλοντα* (frameworks). Μόνο η συνάρτηση κόστους απαιτεί ένα βαθμό "συγχώνευσης" με το δεδομένο σύστημα προκειμένου να οριστεί κατάλληλα, και κατά συνέπεια δεν χρειάζονται ιδιαίτερα σημαντικές αλλαγές σε ένα προσχεδιασμένο περιβάλλον ώστε να εφαρμοστεί ένας ΓΑ.

7. Είναι επεκτάσιμοι. Συνήθως οι ΓΑ επιδέχονται αρκετές αλλαγές ώστε να προσαρμοστούν στα διάφορα προβλήματα, χωρίς αυτό να μειώνει την αποτελεσματικότητά τους. Μάλιστα, λόγω της συχνής χρήσης τους σε πληθώρα προβλημάτων, δεκάδες παραλλαγές έχουν προκύψει τόσο ως προς την μορφή της κωδικοποίησης όσο και ως προς τη διαδικασία που ακολουθείται κατά την διάρκεια της εξέλιξης.

8. Μπορούν να συνδυαστούν χωρίς προβλήματα με άλλες τεχνικές. Σε προβλήματα για τα οποία υπάρχουν πιο αποτελεσματικές μέθοδοι επίλυσης, οι ΓΑ πολλές φορές χρησιμεύουν έμμεσα, σχηματίζοντας υβριδικά συστήματα. Μία αρκετά γνωστή υβριδική μορφή αποτελεί ο συνδυασμός γενετικών αλγορίθμων με *Τεχνητά Νευρωνικά Δίκτυα* (Artificial Neural Networks), όπου οι ΓΑ εκτελούνται για την εύρεση μίας επιθυμητής λύσης, η οποία χρησιμοποιείται στη συνέχεια για την εκπαίδευση του ΤΝΔ.

3.6 Εφαρμογές των Γενετικών Αλγορίθμων

Όπως προαναφέρθηκε, ο κυριότερος λόγος που οι ΓΑ έχουν αναπτυχθεί και έχουν γνωρίσει πληθώρα παραλλαγών τα τελευταία χρόνια, είναι το πολύ μεγάλο πεδίο εφαρμογής τους. Σε αυτό το σημείο θα εξεταστούν οι σημαντικότερες κατηγορίες προβλημάτων στις οποίες χρησιμοποιούνται.

Προβλήματα Βελτιστοποίησης Πρόκειται για προβλήματα όπου στόχος είναι η μεγιστοποίηση (ή ελαχιστοποίηση) μίας αντικειμενικής συνάρτησης, προσδιορίζοντας κατάλληλα τις τιμές συγκεκριμένων παραμέτρων. Σε αυτές τις περιπτώσεις η ανταπόκριση των ΓΑ είναι πολύ μεγάλη, καθώς η λογική στην οποία βασίζονται χρησιμεύει ακριβώς για την επίλυση τέτοιας φύσης προβλημάτων.

Σχεδιασμός μηχανικών συστημάτων Μία από τις πρώτες εφαρμογές των ΓΑ ήταν η χρησιμοποίηση τους για τη βελτιστοποίηση παραμέτρων σε αεροσκάφη στα μέσα της δεκαετίας του 1960. Μέχρι σήμερα έχουν εφαρμοστεί πολλά μοντέλα προσομοίωσης για την βελτιστοποίηση του προσανατολισμού και της δρομολόγησης μηχανικών οχημάτων που στηρίζονται σε γενετικούς αλγορίθμους.

Machine Learning Σε αυτή την κατηγορία οι ΓΑ έχουν καθοριστικό ρόλο, εμφανιζόμενοι κυρίως σε υβριδική μορφή σε συνδυασμό με άλλες τεχνικές. Ιδιαίτερα διαδεδομένη είναι η χρήση τους για την εκπαίδευση Τεχνητών Νευρωνικών Δικτύων όπως και ο συνδυασμός τους με *Ασαφή Συστήματα* (Fuzzy Systems) από τον οποίο προέκυψαν και οι *Ασαφείς Γενετικοί Αλγόριθμοι* (Fuzzy Genetic Algorithms). Ωστόσο και αυτούσιοι οι ΓΑ έχουν εφαρμογές στον κλάδο του machine learning καθώς ο βασικός μηχανισμός υλοποίησής τους προσεγγίζει πολύ το μοντέλο της εκπαίδευσης, μέσω της εξελικτικής διαδικασίας που ακολουθεί.

Ρομποτική Γενετικοί αλγόριθμοι έχουν χρησιμοποιηθεί και στον κλάδο της ρομποτικής για τον συντονισμό των κινήσεων ρομπότ. Πιο συγκεκριμένα μπορούν να κωδικοποιήσουν τις σχετικές παραμέτρους όπως γωνίες και αποστάσεις, προκειμένου να ελαχιστοποιηθεί το σφάλμα της τελικής απόστασης από τον επιθυμητό στόχο και να προσομοιωθεί κατά το καλύτερο δυνατόν η ανθρώπινη κίνηση [Mess02].

Εφαρμογές του TSP Το *Πρόβλημα του Πλανόδιου Πωλητή* (Traveling Salesman Problem) ή TSP είναι ένα θεμελιώδες πρόβλημα συνδυαστικής βελτιστοποίησης στον κλάδο της Θεωρητικής Πληροφορικής. Στη γενική μορφή του TSP αναζητείται η ακολουθία προσπέλασης όλων των κόμβων ενός μη κατευθυνόμενου γράφου με βάρη, η οποία ελαχιστοποιεί το συνολικό κόστος, δηλαδή το άθροισμα των βαρών όλων των ακμών που διασχίζονται, με την προϋπόθεση κάθε κόμβος να προσπελαστεί ακριβώς μία φορά. Το συγκεκριμένο πρόβλημα έχει επιλυθεί με τη χρήση ΓΑ [Potv96] και μπορεί να επεκταθεί σε πάρα πολλά πραγματικά προβλήματα, τα οποία κατά συνέπεια είναι δυνατόν να προσεγγιστούν επίσης από γενετικούς αλγορίθμους.

Βιολογία Οι γενετικοί αλγόριθμοι στηρίχθηκαν σε βασικές έννοιες της εξέλιξης δανεισμένες από τη βιολογία και η λειτουργία τους μιμείται το εξελικτικό μοντέλο της φύσης. Ως εκ τούτου μπορούν με πολύ μικρές αλλαγές να προσομοιώσουν το DNA και να χρησιμοποιηθούν για την επίλυση προβλημάτων στον τομέα της βιολογίας με μεγάλη ευκολία [Levi].

Προβλήματα Χρονοδρομολόγησης Οι γενετικοί αλγόριθμοι ενδείκνυνται για τέτοιου είδους προβλήματα, που έχουν μεγάλη πολυπλοκότητα και δεν μπορούν να επιλυθούν με κάποια αναλυτική μέθοδο.

Οικονομία Πολλά οικονομικά μοντέλα είναι δυνατόν να περιγραφούν και να μοντελοποιηθούν με τη χρήση ΓΑ. Ένα σημαντικό πλεονέκτημα είναι ότι υπάρχει πληθώρα μαθηματικών εργαλείων ικανών να ορίσουν αποδοτικά τη συνάρτηση καταλληλότητας του αλγορίθμου, σε αντίθεση με άλλες περιπτώσεις [Geis07]. Τα σημαντικότερα μοντέλα που έχουν προσομοιωθεί με ΓΑ είναι:

1. Μοντέλο Αράχνης (Cobweb Model)
2. Θεωρία Παιγνίων (Game Theory)
3. Μοντέλο Επικαλυπτόμενων Γενεών (Overlapping Generations Model)

Κατά κύριο λόγο, ωστόσο, οι υλοποιήσεις γενετικών αλγορίθμων στον τομέα της οικονομίας αφορούν το cobweb model.

3.7 Γενετικοί Αλγόριθμοι και Video Games

Η μεγαλύτερη πρόκληση στα ηλεκτρονικά παιχνίδια είναι η δημιουργία ενός ευφυούς συστήματος ελέγχου των χαρακτήρων, εχθρών ή και συμπαικτών, από τον υπολογιστή. Για να καταστεί αυτό εφικτό, την τελευταία δεκαετία άρχισε να εισάγεται στα video games τεχνητή νοημοσύνη, με τέτοιο ρυθμό μάλιστα ώστε πλέον θεωρείται δεδομένη σε όλα σχεδόν τα παιχνίδια. Σαν αποτέλεσμα, σήμερα υπάρχει πολύ μεγάλη ποικιλία μεθόδων εφαρμογής της, προκειμένου να είναι τα παιχνίδια πιο ελκυστικά.

Η εφαρμογή της Τεχνητής Νοημοσύνης σε ένα video game εστιάζει σε τρεις βασικούς άξονες οι οποίοι είναι η ικανότητα κίνησης των χαρακτήρων, η επιλογή της θέσης της κίνησης και η δυνατότητα στρατηγικές σκέψης. Καθώς η συμπεριφορά των χαρακτήρων εξαρτάται από ένα μεγάλο πλήθος αλληλοσυνδεόμενων παραμέτρων, ο καθορισμός των οποίων δεν μπορεί να γίνει χειροκίνητα, αυτοματοποιημένες μέθοδοι απαιτούνται για τον προσδιορισμό της [Mill09, Bull].

Μία τέτοια μέθοδος είναι η υλοποίηση ενός κατάλληλα προσαρμοσμένου γενετικού αλγορίθμου. Ο ΓΑ προσεγγίζει τις ζητούμενες παραμέτρους με την τεχνική *δοκιμής-λάθους* (trial and error), εξετάζοντας πολλούς πληθυσμούς από πιθανές λύσεις και απορρίπτοντας τις λιγότερο αποτελεσματικές μέχρι να καταλήξει στη βέλτιστη.

Το μεγάλο πλεονέκτημά του είναι ότι καταλήγει σε διαφορετική λύση όταν αντιμετωπίζει διαφορετικούς παίχτες, προσαρμόζοντας ουσιαστικά τη συμπεριφορά των ελεγχόμενων από τον υπολογιστή χαρακτήρων στον τρόπο παιχνιδιού του κάθε παίχτη-ανθρώπου. Επομένως η επιλογή των ΓΑ ενδείκνυται σε παιχνίδια που η συμπεριφορά του πραγματικού παίχτη μπορεί να αλλάζει σε μεγάλο βαθμό, απαιτώντας την αντίστοιχη ανταπόκριση από τον παίχτη-υπολογιστή [Bour14].

Άλλο ένα στοιχείο που όταν υπάρχει ενισχύει την επιλογή γενετικών αλγορίθμων είναι η δυσκολία πρόβλεψης της συμπεριφοράς των παιχτών. Για να είναι ένας χαρακτήρας (συνήθως αντίπαλος) ικανοποιητικός πρέπει να είναι σε θέση να προσχεδιάσει τις κινήσεις του ανάλογα με τις κινήσεις του παίχτη. Σε πολύπλοκα παιχνίδια, που δεν είναι δυνατόν να προβλεφθούν όλες οι πιθανές ενέργειες από τους σχεδιαστές, οι ΓΑ αποτελούν μία εναλλακτική μέθοδο.

Φυσικά οι ΓΑ δεν συνιστούν την ιδανικότερη επιλογή για την εισαγωγή τεχνητής νοημοσύνης σε όλα τα video games, καθώς ανάλογα με τη φύση του κάθε παιχνιδιού υπάρχουν και αντίστοιχες τεχνικές που είναι αποδοτικότερες. Σε κάθε περίπτωση είναι αναγκαία η ορθή κατανόηση του εκάστοτε περιβάλλοντος, προκειμένου να επιλεχθεί η καταλληλότερη μέθοδος.

Κεφάλαιο 4

Framework και Υλοποίηση

4.1 GVG-AI

4.1.1 Περιγραφή του GVG-AI

Ο *Γενικός Διαγωνισμός Ηλεκτρονικών Παιχνιδιών βασισμένος σε Τεχνητή Νοημοσύνη* (General Video Game-AI Competition) ή πιο απλά GVG-AI είναι ένας διαγωνισμός που διεξάγεται με στόχο την ανάπτυξη πρακτόρων, ικανών να ανταπεξέλθουν σε περισσότερα από ένα διαφορετικά video games. Η ιδέα που εξυπηρετεί είναι ότι ένας ευφυής πράκτορας δεν αρκεί να είναι αποτελεσματικός σε ένα συγκεκριμένο πρόβλημα (σε αυτή την περίπτωση σε ένα συγκεκριμένο παιχνίδι) αλλά πρέπει να είναι εξίσου προσαρμόσιμος σε μία ευρύτερη κατηγορία.

Ο πρώτος διαγωνισμός GVG-AI πραγματοποιήθηκε στο πλαίσιο του *Συνεδρίου για Υπολογιστική Νοημοσύνη και Παιχνίδια* (Computer Intelligence and Games – CIG) το 2014 και είχε 14 υποψηφιότητες [Lieb16]. Έκτοτε λαμβάνει χώρα μία φορά το χρόνο έχοντας αυξανόμενη συμμετοχή και προσφέροντας όλο και περισσότερες προκλήσεις όπως για παράδειγμα την ανάπτυξη πρακτόρων για *παιχνίδια περισσότερων παιχτών* (multi-player games). Μέχρι στιγμής έχουν χρησιμοποιηθεί για την ανάπτυξη των πρακτόρων κυρίως συνδυαστικές μέθοδοι που περιλαμβάνουν ευριστικές τεχνικές, εξελικτικούς αλγορίθμους, δέντρα αναζήτησης κα. Ωστόσο ακόμα δεν έχει βρεθεί κάποια μέθοδος αρκετά αποτελεσματική ώστε να θεωρηθεί ότι ο GVG-AI έχει εξαντλήσει τα περιθώρια βελτίωσης ενός γενικού πράκτορα και έχει πετύχει το στόχο του.

Ο GVG-AI προσφέρει μία πλατφόρμα στην οποία μπορούν να αναπτυχθούν και να δοκιμαστούν πράκτορες σε ένα σύνολο παιχνιδιών. Ο *προγραμματιστικός σκελετός* (framework) περιλαμβάνει παιχνίδια δύο διαστάσεων (2D), βασισμένα σε κλασσικά παιχνίδια των δεκαετιών 1980-1990 όπως Zelda, Pokemon, Bomberman κ.ά.

Στο διαγωνιστικό κομμάτι, το σύνολο των παιχνιδιών είναι χωρισμένο σε τρία υποσύνολα, που είναι απαραίτητα για τη δομή και τον τρόπο υλοποίησης του διαγωνισμού.

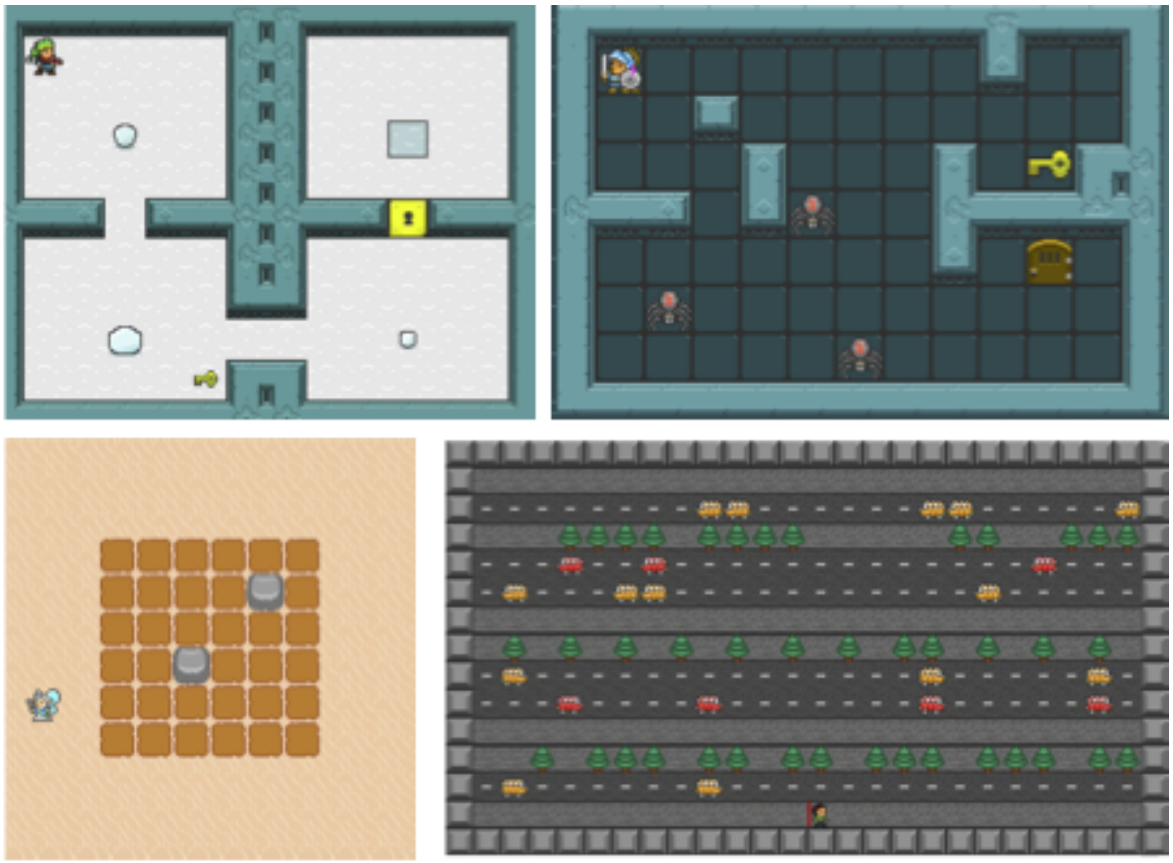
Το πρώτο σύνολο παιχνιδιών, είναι αυτό που έχουν οι διαγωνιζόμενοι στη διάθεσή τους για να εκπαιδεύσουν και να δοκιμάσουν τους πράκτορές τους. Για καθένα από αυτά τα παιχνίδια είναι διαθέσιμα πέντε διαφορετικά επίπεδα-πίστες ενώ η περιγραφή και ο σκοπός τους είναι επίσης γνωστά.

Το δεύτερο σύνολο είναι κρυφό, δηλαδή περιλαμβάνει παιχνίδια που δεν είναι εκ των προτέρων γνωστά. Οι συμμετέχοντες μπορούν να υποβάλλουν τις λύσεις τους (όσες φορές επιθυμούν) και αφού αυτές αξιολογηθούν, να βλέπουν την κατάταξη τους στον αναρτημένο πίνακα για να είναι σε θέση να ξέρουν πως ανταποκρίνονται οι πράκτορές τους σε άγνωστα περιβάλλοντα και να κάνουν περαιτέρω βελτιώσεις. Μετά τη λήξη του διαγωνισμού γνωστοποιούνται και αυτά τα παιχνίδια επαλήθευσης.

Αφού γίνουν οι οριστικές υποβολές, οι πράκτορες εφαρμόζονται σε ένα τρίτο σύνολο επίσης αγνώστων, στους συμμετέχοντες, παιχνιδιών και από την αξιολόγησή τους προκύπτει η τελική κατάταξη. Το τελευταίο αυτό σετ χρησιμοποιείται στον επόμενο διαγωνισμό ως το αρχικό σετ εκπαίδευσης [Pere16].

Ανάλογα με το είδος των παιχνιδιών υπάρχουν τρία διαφορετικά συστήματα βαθμολόγησης:

Δυαδικό (Binary Case) Είναι το πιο απλό σύστημα καθώς ελέγχει μόνο αν το παιχνίδι ολοκληρώθηκε επιτυχώς. Σε αυτή την περίπτωση η βαθμολογία είναι 1, διαφορετικά σε περίπτωση αποτυχίας



Σχήμα 4.1: Ενδεικτικά παιχνίδια του GVG-AI Framework

είναι 0.

Σταδιακό (Incremental Case) Σε αυτό το σύστημα, υπάρχουν ενέργειες ή στόχοι που όταν επιτευχθούν δίνουν μία μικρή αύξηση στη βαθμολογία. Παραδείγματα τέτοιων ενεργειών είναι η εξολόθρευση εχθρών ή η συλλογή κάποιων αντικειμένων.

Διακεκομμένο (Discontinuous Case) Το διακεκομμένο σύστημα βαθμολόγησης λειτουργεί όπως το σταδιακό έχοντας όμως μία επιπλέον διαβάθμιση στο bonus των ενεργειών. Οι στόχοι που είναι δυσκολότερο να πραγματοποιηθούν προσδίδουν μεγαλύτερη αύξηση στη βαθμολογία από άλλους υποδεέστερους, όπως για παράδειγμα η εύρεση του κλειδιού σε σχέση με ένα άλλο απλό αντικείμενο.

Ένας άλλος διαχωρισμός μπορεί να γίνει στα παιχνίδια με βάση το πλήθος των δυνατών ενεργειών. Η πλήρης λίστα των ενεργειών περιλαμβάνει κίνηση προς τις τέσσερις βασικές κατευθύνσεις (πάνω, κάτω, δεξιά, αριστερά) και μία επιπλέον ενέργεια "use", η οποία μπορεί να έχει διαφορετική συμπεριφορά ανάλογα με το παιχνίδι. Σύμφωνα με αυτή τη μέθοδο διάκρισης υπάρχουν τρεις κατηγορίες:

A0 Είναι η κατηγορία που περιλαμβάνει τα παιχνίδια στα οποία είναι διαθέσιμες όλες οι ενέργειες που υποστηρίζονται στο GVG-AI framework.

A1 Πρόκειται για παιχνίδια στα οποία υπάρχει μόνο η δυνατότητα κίνησης στο χώρο, δηλαδή δεν περιλαμβάνουν ενέργεια "use".

A2 Τα παιχνίδια αυτής της κατηγορίας υποστηρίζουν ενέργεια "use" αλλά έχουν περιορισμό ως προς την κίνησή τους καθώς κινούνται μόνο δεξιά και αριστερά.

Στα περισσότερα παιχνίδια εκτός από τον βασικό χαρακτήρα, ο χειρισμός του οποίου είναι και το αντικείμενο του διαγωνισμού, υπάρχουν και άλλοι χαρακτήρες που ελέγχονται από τον υπολογιστή. Σε αυτούς θα αναφερόμαστε με τον όρο ***N**on-**P**layer-**C**haracters* ή για συντομία NPCs. Οι NPCs είναι συνήθως εχθροί όμως σε ορισμένα παιχνίδια μπορεί να έχουν βοηθητικό ρόλο για τον παίχτη.

Το τελευταίο απαραίτητο στοιχείο για την περιγραφή ενός παιχνιδιού του GVG-AI είναι η μέθοδος τερματισμού. Οι δυνατές μέθοδοι τερματισμού (που μπορεί να οδηγούν είτε σε νίκη είτε σε ήττα) είναι οι εξής:

Μετρητές (Counters) Οι μετρητές τερματίζουν το παιχνίδι αν ικανοποιηθεί κάποια συνθήκη δημιουργίας ή καταστροφής κάποιου ή κάποιων συγκεκριμένων χαρακτήρων. Συνηθέστερος μετρητής είναι αυτός που ελέγχει την καταστροφή του παίχτη από εχθρούς.

Πόρτα Εξόδου (Exit Door) Ολοκληρώνει το παιχνίδι αν ο παίχτης φτάσει στην πόρτα εξόδου ή κάποιο άλλο σημείο της πίστας με παρόμοια λειτουργία.

Τέλος Χρόνου (Timeout) Κάποια παιχνίδια τελειώνουν όταν ολοκληρωθεί ένα συγκεκριμένο χρονικό διάστημα, όπως για παράδειγμα συμβαίνει σε ένα παιχνίδι επιβίωσης.

Προκειμένου να αποφευχθεί ο κίνδυνος δημιουργίας πρακτόρων που θα εκτελούνται ατέρμονα αν δεν μπορούν να ικανοποιήσουν κάποια από τις παραπάνω συνθήκες, έχει οριστεί ένα όριο χρόνου (σε βήματα/timesteps) το οποίο όταν ξεπεραστεί τερματίζει το παιχνίδι με ήττα για τον παίχτη. Αυτό το χρονικό όριο ισχύει για όλα τα παιχνίδια και είναι εντελώς ανεξάρτητο από το κριτήριο τερματισμού timeout.

Στον Πίνακα 4.1 παρουσιάζονται ενδεικτικά παιχνίδια που περιλαμβάνονται στο GVG-AI framework και οι κατηγοριοποιήσεις τους.

Πίνακας 4.1: Κατηγοριοποίηση παιχνιδιών του GVG-AI framework

Games	Score System	NPC Types			Resources	Terminations			Action Set
		Friendly	Enemies	>1 type		Counters	Exit Door	Timeout	
Aliens	**I**		✓			✓			A2
Boulderdash	**I**		✓	✓	✓		✓		A0
Butterflies	**I**	✓				✓			A1
Chase	**I**	✓	✓			✓			A1
Frogs	**B**						✓		A1
Missile Command	**I**		✓			✓			A0
Portals	**B**						✓		A1
Sokoban	**I**					✓			A1
Survive Zombies	**I**	✓	✓		✓			✓	A1
Zelda	**I**		✓		✓		✓		A0

Για την αξιολόγηση των υποβληθέντων πρακτόρων, ο καθένας εφαρμόζεται σε 10 παιχνίδια, από 50 φορές στο καθένα (10 για κάθε ένα από τα 5 επίπεδα). Τα κριτήρια αξιολόγησης είναι κατ' αρχάς ο αριθμός των παιχνιδιών που ολοκληρώθηκαν με επιτυχία και σε περίπτωση ισοπαλίας, κατά σειρά η συνολική βαθμολογία του πράκτορα και ο απαιτούμενος χρόνος για την ολοκλήρωση των παιχνιδιών [Pere16].

4.1.2 GVG-AI Framework

4.1.2.1 Video Game Description Language (VGDL)

Ο GVG-AI προσφέρει ένα Java Framework που αναλύει και αποκωδικοποιεί την *Video Game Description Language* (VGDL). Η VGDL είναι μία γλώσσα ειδικά σχεδιασμένη για την περιγραφή ηλεκτρονικών παιχνιδιών. Υλοποιήθηκε από τον Tom Schaul σε Python με τη χρήση της βιβλιοθήκης py-game, η οποία περιλαμβάνει γραφικά ειδικά για το σχεδιασμό παιχνιδιών τύπου *NES* (Nintendo Entertainment System).

Η βασική οντότητα στην VGDL είναι το *αντικείμενο* (object). Ως αντικείμενα θεωρούνται όλα τα στοιχεία του παιχνιδιού όπως χαρακτήρες, πόρτες, τοίχοι κλπ. Τα αντικείμενα μπορούν να αλληλεπιδρούν μεταξύ τους όταν συγκρούονται και κάποια από αυτά να κινούνται. Με βάση τη μέθοδο κίνησης, τα αντικείμενα διακρίνονται σε ενεργητικά, όταν κινούνται με βάση κάποια εσωτερική συμπεριφορά και παθητικά όταν μπορούν να κινηθούν μόνο εξαρτώμενα από τις φυσικές τους ιδιότητες (πχ ένα αντικείμενο που πέφτει προς το έδαφος λόγω βαρύτητας).

Για να οριστεί πλήρως ένα παιχνίδι σε VGDL απαιτείται ο ορισμός των αρχείων *Περιγραφής Παιχνιδιού* και *Περιγραφής Επιπέδου* [Pere16].

Αρχείο Περιγραφής Παιχνιδιού (Game Description File) Προσδιορίζονται όλα τα αντικείμενα του παιχνιδιού και οι πιθανοί τρόποι με τους οποίους που μπορούν να αλληλεπιδρούν μεταξύ τους. Επίσης στο game description file περιγράφονται οι συνθήκες τερματισμού του παιχνιδιού. Η δομή του αρχείου περιλαμβάνει τέσσερα βασικά τμήματα:

- SpriteSet: Ορίζονται τα διάφορα αντικείμενα και οι ιδιότητές τους όπως η εμφάνιση τους στο χώρο (χρώμα, σχήμα κλπ), οι συμπεριφορές τους (πχ παραγωγή άλλων αντικειμένων) και οι φυσικές τους ιδιότητες (πχ μάζα). Οι κλάσεις των αντικειμένων οργανώνονται σε δενδρική δομή, έτσι ώστε οι κλάσεις-παιδιά να κληρονομούν τον τύπο και τις ιδιότητες της κλάσης-γονέα.

```
SpriteSet
     sprite_1 >
          sprite_2 > type_1 parameters
          sprite_3 > type_2 parameters
```

Σχήμα 4.2: Γενική περιγραφή ενός SpriteSet σε VGDL

Οι σημαντικότεροι τύποι αντικειμένων είναι: *Avatar, Immovable, Passive, RandomNPC* και *Door*.

- InteractionSet: Ορίζεται η συμπεριφορά/αποτελέσματα μετά τη σύγκρουση δύο αντικειμένων. Η συμπεριφορά των αντικειμένων εξαρτάται από τον τύπο του καθενός, γι' αυτό ορίζεται ξεχωριστά για κάθε πιθανό συνδυασμό των διάφορων τύπων.

```
InteractionSet
     sprite_1 sprite_2 > interaction parameters
```

Σχήμα 4.3: Γενική περιγραφή ενός InteractionSet σε VGDL

Τα σημαντικότερα interactions είναι: *killSprite, transformTo, removeScore, reverseDirection* και *collectResource*.

- TerminationSet: Είναι το τμήμα του αρχείου στο οποίο περιγράφονται οι συνθήκες τερματισμού του παιχνιδιού καθώς και το αποτέλεσμα (νίκη-ήττα) σε κάθε περίπτωση.

  Οι συνθήκες τερματισμού έχουν εξεταστεί αναλυτικά στην Ενότητα 4.1.1

```
TerminationSet
     Condition parameters win=True
     Condition parameters win=False
```

Σχήμα 4.4: Γενική περιγραφή ενός TerminationSet σε VGDL

- LevelMapping: Γίνεται η αντιστοίχιση κάθε αντικειμένου σε ένα σύμβολο, ώστε να μπορεί να αναπαρασταθεί στο level description file.

```
LevelMapping
     symbol_1  >    sprite_1
     symbol_2  >    sprite_2 sprite_3
```

Σχήμα 4.5: Γενική περιγραφή ενός LevelMapping σε VGDL

```
BasicGame
     SpriteSet
          hole    > Immovable color=DARKBLUE img=hole
          avatar  > MovingAvatar
          box     > Passive img=box
          wall    > Immovable img=wall
     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
     LevelMapping
          0 > hole
          1 > box
          A > avatar
          w > wall
```

Σχήμα 4.6: Αρχείο Περιγραφής Παιχνιδιού σε VGDL για το παιχνίδι Sokoban

Αρχείο Περιγραφής Επιπέδου (Level Description File) Αυτό το αρχείο περιέχει έναν δισδιάστατο πίνακα από σύμβολα που αναπαριστούν τα αντικείμενα του παιχνιδιού και ορίζουν την αρχική τους διάταξη στο χώρο. Τα σύμβολα που χρησιμοποιούνται έχουν οριστεί στο game description file με την προϋπόθεση οι *χαρακτήρες των παιχτών* (avatars) να αναπαριστώνται με κεφαλαία γράμματα και τα υπόλοιπα αντικείμενα με πεζά.

```
wwwwwwwwwwww
w         w  w
w   1        w
w   A 1 w 0ww
www w1   wwww
w        w 0 w
w 1         ww
w           ww
wwwwwwwwwwww
```

Σχήμα 4.7: Παράδειγμα αρχείου Περιγραφής Επιπέδου σε VGDL

4.1.2.2 Java-VGDL API

Το Java-VGDL framework μπορεί να φορτώσει παιχνίδια και επίπεδα που είναι ορισμένα σε VGDL και προσφέρει μία *διεπαφή* (API) που δίνει στους συμμετέχοντες τη δυνατότητα να αναπτύξουν πράκτορες που ελέγχουν τις κινήσεις των avatars καθώς επίσης και level generators.

Η οργάνωση του κώδικα γίνεται σε *πακέτα* (java packages). Τα βασικότερα πακέτα είναι:

- **controllers**: Αυτό το πακέτο περιλαμβάνει δύο πακέτα για single-player και multi-player controllers, μέσα στα οποία μπορούν να δημιουργηθούν οι αντίστοιχοι νέοι agents. Τα πακέτα αυτά περιέχουν τις κλάσεις *Test* και *TestMultiPlayer* αντίστοιχα οι οποίες εκτελούν τα παιχνίδια στο VGDL Framework εφαρμόζοντας κάθε φορά τον επιθυμητό agent.

- **core**: Περιλαμβάνει βασικά πακέτα για τη δημιουργία των παιχνιδιών, τον τερματισμό τους, τις παραμέτρους του διαγωνισμού κα. Αυτά τα πακέτα είναι:

  - competition
  - content
  - game
  - generator
  - player
  - termination

- **levelGenerators**: Περιέχει την κλάση για τη δημιουργία επιπέδων και κάποιες ενδεικτικές γεννήτριες.

- **ontology**: Σε αυτό το πακέτο περιγράφονται οι τύποι των sprites και οι ιδιότητές τους για την VGDL.

- **ruleGenerators**: Περιέχει την κλάση για τη δημιουργία κανόνων και κάποιες ενδεικτικές γεννήτριες.

- **tools**: Περιλαμβάνει κάποιες επιπλέον απαραίτητες κλάσεις για τον διαγωνισμό.

Για τη δημιουργία ενός controller, απαιτείται η δημιουργία μίας νέας Java κλάσης, η οποία *κληρονομεί* (inherits) βασικές μεθόδους από μία προσχεδιασμένη διαθέσιμη κλάση (*AbstractPlayer*) και *υλοποιεί* (implements) δύο από αυτές:

Agent Πρόκειται για τον *δομητή* (constructor) της κλάσης, για τον οποίο υπάρχει η απαίτηση να ολοκληρωθεί σε λιγότερο από 1sec.

act Αυτή η μέθοδος εκτελείται σε κάθε *κύκλο του παιχνιδιού* (game cycle) και επιστρέφει την επόμενη κίνηση του avatar. Οι πιθανές κινήσεις είναι:

1. ACTION_NIL: ο agent δεν κάνει τίποτα
2. ACTION_LEFT: κίνηση μία θέση προς τα αριστερά
3. ACTION_UP: κίνηση μία θέση προς τα πάνω
4. ACTION_RIGHT: κίνηση μία θέση προς τα δεξιά
5. ACTION_DOWN: κίνηση μία θέση προς τα κάτω
6. ACTION_USE: ο agent εκτελεί μία ενέργεια που ποικίλλει ανάλογα με το παιχνίδι

Οι τέσσερις κινήσεις κατεύθυνσης, σε ορισμένα παιχνίδια μετακινούν το avatar μόνο αν έχει ήδη κατεύθυνση προς την αντίστοιχη θέση, διαφορετικά απλώς αλλάζουν την κατεύθυνσή του. Κατά συνέπεια, για να κινηθεί προς μία θέση αν έχει διαφορετική κατεύθυνση, θα πρέπει η κίνηση να εκτελεστεί δύο συνεχόμενες φορές (την πρώτη θα γυρίσει προς την επιθυμητή κατεύθυνση και τη δεύτερη θα μετακινηθεί). Επίσης δεν είναι όλες οι κινήσεις διαθέσιμες σε όλα τα παιχνίδια.

Το χρονικό όριο για κάθε κίνηση είναι 40ms. Σε περίπτωση που ο χρόνος ολοκλήρωσης είναι μεταξύ 40 msec και 50 msec τότε εκτελείται η ACTION_NIL, ενώ αν ξεπεράσει τα 50 msec τότε ο πράκτορας αποκλείεται λόγω μεγάλης καθυστέρησης.

Και οι δύο αυτές μέθοδοι (Agent, act) παίρνουν σαν παραμέτρους ένα αντικείμενο της κλάσης *ElapsedCpuTimer* το οποίο μπορεί να ενημερώνει ανά πάσα στιγμή τον agent για το πόσα ms έχουν περάσει, καθώς και ένα αντικείμενο της κλάσης *StateObservation*, το οποίο περιέχει όλη την πληροφορία για την κατάσταση του παιχνιδιού σε κάθε κύκλο.

Πιο συγκεκριμένα η κλάση StateObservation του API που προσφέρει το Java-VGDL Framework διαθέτει μεταξύ άλλων τις εξής, ιδιαίτερα χρήσιμες μεθόδους:

1. Για προσομοίωση επόμενων *καταστάσεων* (states)
   - **advance**: Παίρνει σαν παράμετρο μία ενέργεια και προσομοιώνει την κατάσταση του παιχνιδιού στον επόμενο κύκλο, αφού δηλαδή αυτή εκτελεστεί. Τα γεγονότα που είναι στοχαστικά, όμως, ενδέχεται να είναι διαφορετικά στην πραγματική επόμενη κατάσταση.
   - **copy**: Επιστρέφει ένα αντίγραφο του αντικειμένου stateObservation.
2. Σχετικά με την κατάσταση του παιχνιδιού
   - **getGameScore**: Επιστρέφει το score του παιχνιδιού.
   - **getGameTick**: Επιστρέφει τον αριθμό των *χρονικών βημάτων* (timesteps) που έχουν περάσει μέχρι τη στιγμή που καλείται.
   - **getGameWinner**: Επιστρέφει το αποτέλεσμα του παιχνιδιού (νίκη-ήττα) αν αυτό έχει ολοκληρωθεί.
   - **getWorldDimension**: Επιστρέφει τις διαστάσεις του παραθύρου του παιχνιδιού σε pixels.
   - **getBlockSize**: Επιστρέφει το μήκος (άρα και το πλάτος καθώς πρόκειται για τετράγωνα) των αντικειμένων και των χαρακτήρων του παιχνιδιού σε pixels.
   - **isGameOver**: Ενημερώνει αν έχει τελειώσει το παιχνίδι ή όχι.
3. Σχετικά με την κατάσταση του avatar
   - **getAvailableActions**: Επιστρέφει το σύνολο των κινήσεων που είναι διαθέσιμες στο παιχνίδι.

- **getAvatarPosition**: Επιστρέφει τη θέση του avatar στον κόσμο.
- **getAvatarSpeed**: Επιστρέφει την ταχύτητα του avatar.
- **getAvatarOrientation**: Επιστρέφει την κατεύθυνση του avatar.
- **getAvatarResources**: Επιστρέφει μία λίστα με τα αντικείμενα που έχει συλλέξει ο παίχτης.
- **getEventsHistory**: Επιστρέφει μία λίστα με όλα τα γεγονότα (συγκρούσεις του avatar με άλλα αντικείμενα) που έχουν συμβεί.
- **getObservationGrid**: Επιστρέφει τον κόσμο του παιχνιδιού με μορφή πλέγματος, στο οποίο κάθε κελί περιλαμβάνει μία λίστα με τα αντικείμενα και χαρακτήρες που βρίσκονται στην αντίστοιχη θέση.

4. Σχετικά με τους υπόλοιπους χαρακτήρες/αντικείμενα

- **getNPCPositions**: Επιστρέφει τις θέσεις των NPCs.
- **getImmovablePositions**: Επιστρέφει τις θέσεις των ακίνητων αντικειμένων/χαρακτήρων.
- **getMovablePositions**: Επιστρέφει τις θέσεις των κινούμενων αντικειμένων/χαρακτήρων.
- **getResourcesPositions**: Επιστρέφει τις θέσεις των αντικειμένων προς συλλογή (πόρων).
- **getPortalPositions**: Επιστρέφει τις θέσεις των πυλών.
- **getFromAvatarSprites**: Επιστρέφει μία λίστα με τα αντικείμενα/χαρακτήρες που έχει δημιουργήσει ο παίχτης (σε όσα παιχνίδια αυτό είναι δυνατό).

4.2 Υλοποίηση Πρακτόρων

4.2.1 Γενική Περιγραφή

Για την υλοποίηση των πρακτόρων στηριχθήκαμε σε Γενετικούς Αλγορίθμους χρησιμοποιώντας μία εναλλακτική προσέγγιση. Συνήθως για ένα πρόβλημα τέτοιου τύπου, όπως η επιλογή των κατάλληλων κινήσεων σε ένα παιχνίδι, οι γενετικοί αλγόριθμοι κωδικοποιούν την ακολουθία των κινήσεων μέχρι να βρεθεί αυτή που οδηγεί σε νίκη του παίχτη.

Αυτή η τεχνική όμως, έχει δύο βασικά μειονεκτήματα. Κατ' αρχάς είναι πολύ πιθανό σε ένα παιχνίδι να υπάρχουν στοχαστικά γεγονότα, δηλαδή γεγονότα που δεν συμβαίνουν πάντα με τον ίδιο τρόπο, όπως για παράδειγμα οι κινήσεις ενός εχθρού. Αυτό σημαίνει ότι μία ακολουθία κινήσεων που ήταν επιτυχής μία φορά δεν είναι σίγουρο ότι θα είναι σε κάθε περίπτωση. Το άλλο μειονέκτημα αυτής της κωδικοποίησης είναι ότι η ακολουθία των κινήσεων προσαρμόζεται κατάλληλα για ένα συγκεκριμένο επίπεδο/πίστα του παιχνιδιού. Αν δοκιμάσουμε την ίδια ακολουθία σε ένα άλλο επίπεδο, κατά πάσα πιθανότητα θα αποτύχει, εκτός αν είναι πολύ παρόμοιο με αυτό στο οποίο έγινε η αρχική εκτέλεση του ΓΑ.

Επομένως προκειμένου να αναπτύξουμε έναν πράκτορα ικανό να γενικευθεί για διαφορετικά επίπεδα και να προσαρμόζεται στα στοχαστικά γεγονότα που συμβαίνουν κατά τη διάρκεια του παιχνιδιού, ακολουθήσαμε μία διαφορετική κωδικοποίηση.

Ο στόχος είναι αντί για τη σειρά εκτέλεσης των ενεργειών να κωδικοποιηθούν οι πιθανές καταστάσεις του παιχνιδιού ανάλογα με την διάταξη των *sprites* (αντικειμένων-χαρακτήρων) στο χώρο. Για παράδειγμα ο πράκτορας να μπορεί να κωδικοποιήσει την κατάσταση στην οποία κινδυνεύει από κάποιον εχθρό και να ενεργεί ανάλογα. Γι' αυτό το σκοπό, αφού αρχικά κωδικοποιήσουμε όλες τις καταστάσεις σύμφωνα με τις απαιτήσεις ενός ΓΑ (δηλαδή σε χρωμοσώματα λαμβάνοντας υπόψιν τους απαραίτητους περιορισμούς), εκτελούμε τον αλγόριθμο ώστε να βρούμε τη βέλτιστη ενέργεια για κάθε περίπτωση.

Ιδανικά, για μία τέτοια υλοποίηση, θα έπρεπε να έχουμε σε κάθε κύκλο πρόσβαση σε όλα τα στοιχεία του κόσμου του παιχνιδιού, ώστε να μπορούμε να προσδιορίσουμε πλήρως την κάθε κατάσταση.

Πρακτικά κάτι τέτοιο είναι αδύνατο, καθώς οι πιθανοί συνδυασμοί αυξάνονται εκθετικά με κάθε επιπλέον θέση που ελέγχουμε, με αποτέλεσμα τόσο ο χρόνος για την εκτέλεση του ΓΑ όσο και ο χρόνος απόφασης της κάθε ενέργειας να είναι απαγορευτικοί.

Για την εφαρμογή του γενετικού αλγορίθμου στο GVG-AI framework, χρησιμοποιήσαμε τη βιβλιοθήκη *Jenetics* [jene], η οποία είναι μία βιβλιοθήκη ειδικά σχεδιασμένη για την υλοποίηση ΓΑ σε Java. Η Jenetics διαθέτει τα βασικά δομικά στοιχεία ενός ΓΑ όπως γονίδια και χρωμοσώματα, καθώς και το μηχανισμό για την δημιουργία του αρχικού πληθυσμού και την εκτέλεση του αλγορίθμου. Επίσης προσφέρει τις σημαντικότερες υλοποιήσεις των μεθόδων επιλογής και μετάλλαξης και κάποιες βοηθητικές λειτουργίες όπως η συλλογή στατιστικών στοιχείων.

Όπως έχει ήδη αναφερθεί, ένα από τα σημαντικότερα πλεονεκτήματα των γενετικών αλγορίθμων είναι η εύκολη προσαρμογή τους σε δεδομένα προβλήματα, καθώς το μόνο στοιχείο που λειτουργεί ως συνδετικός κρίκος είναι η συνάρτηση ποιότητας. Στην προκειμένη περίπτωση η συνάρτηση ποιότητας κάθε ατόμου του πληθυσμού, η οποία ορίζεται μέσω της βιβλιοθήκης Jenetics, δημιουργεί ένα στιγμιότυπο του παιχνιδιού και του πράκτορα ο οποίος ελέγχει το avatar (μέσω του GVG-AI framework). Ο πράκτορας χρησιμοποιεί τα γονίδια του γενότυπου και αποφασίζει ποιες ενέργειες θα εκτελέσει στο παιχνίδι. Στη συνέχεια τα απαραίτητα στοιχεία της τελικής κατάστασης του παιχνιδιού (πχ νίκη-ήττα, χρόνος ολοκλήρωσης κλπ) γίνονται προσβάσιμα από τη συνάρτηση ποιότητας και υπολογίζεται ο βαθμός καταλληλότητας του ατόμου.

Όσον αφορά την υπόλοιπη υλοποίηση οι δύο βιβλιοθήκες λειτουργούν ανεξάρτητα, χωρίς να χρειάζονται επιπλέον τροποποιήσεις. Η εξέλιξη του πληθυσμού και η δημιουργία των νέων γενιών γίνεται αποκλειστικά με τη χρήση του API που προσφέρει η Jenetics και η εκτέλεση του παιχνιδιού για κάθε ένα από τα άτομα-λύσεις, πάνω στο framework του GVG-AI.

Συνολικά υλοποιήθηκαν τρεις διαφορετικοί πράκτορες, στηριζόμενοι στη λογική της κωδικοποίησης των καταστάσεων με τη χρήση ΓΑ. Σε όλους τους πράκτορες, για την κωδικοποίηση των πληροφοριών υπάρχουν περισσότερα από ένα χρωμοσώματα, καθένα από τα οποία αντιστοιχεί σε μία θέση του κόσμου (ή σε ένα συνδυασμό θέσεων). Τα γονίδια είναι αριθμητικά και συγκεκριμένα παίρνουν ακέραιες τιμές, όμως διαφέρουν μεταξύ των τριών υλοποιήσεων, γι' αυτό θα εξεταστούν σε κάθε μία ξεχωριστά. Οι έξι πιθανές ενέργειες "βαθμολογούνται" με κάποια μέθοδο ανάλογα με την υλοποίηση και τις τιμές των γονιδίων, και τελικά επιλέγεται η ενέργεια με τη μεγαλύτερη βαθμολογία.

Επειδή και οι τρεις πράκτορες έχουν τον ίδιο πυρήνα, με κάποιες παραλλαγές, υλοποιούν αρκετές κοινές λειτουργίες, που σχετίζονται με τη λήψη πληροφοριών για την κατάσταση του κόσμου. Παρακάτω παρουσιάζονται οι βασικότερες κοινές μέθοδοι που αναπτύχθηκαν για αυτό το σκοπό.

- **isWall**: Παίρνει σαν παράμετρο μία συγκεκριμένη θέση στο παράθυρο του παιχνιδιού και ελέγχει αν περιλαμβάνει τοίχο.
- **isNpc**: Παίρνει σαν παράμετρο μία συγκεκριμένη θέση στο παράθυρο του παιχνιδιού και ελέγχει αν περιλαμβάνει NPC.
- **distance**: Επιστρέφει την *απόσταση manhattan* (άθροισμα οριζόντιας και κατακόρυφης απόστασης) μεταξύ δύο θέσεων του κόσμου του παιχνιδιού σε pixels.
- **boxes_position**: Παίρνει σαν παράμετρο τη θέση ενός αντικειμένου σε pixels και την επιστρέφει μετρημένη σε *κελιά* (cells) του πλέγματος του κόσμου (Σχήμα 4.8).
- **getDistanceFromPortal**: Επιστρέφει την απόσταση του avatar από την κοντινότερη πόρτα, μετρημένη σε pixels.
- **getDistanceFromKey**: Επιστρέφει την απόσταση του avatar από το κλειδί σε pixels.
- **getMaxDistanceFromTarget**: Επιστρέφει τη μέγιστη απόσταση που μπορεί να υπάρχει μεταξύ μίας συγκεκριμένης θέσης του κόσμου και οποιασδήποτε άλλης σε pixels. Πρακτικά η θέση που θα έχει τη μέγιστη απόσταση από τη ζητούμενη, θα είναι μία από τις τέσσερις γωνίες του κόσμου.

Σχήμα 4.8: Κελί παιχνιδιού

- **updateValues**: Ανανεώνει τη "βαθμολογία" των πιθανών ενεργειών ανάλογα με την κατάσταση του κόσμου.
- **chooseAction**: Επιλέγει την ενέργεια που τελικά θα εκτελεστεί σε κάθε κύκλο.
- **getBestAction**: Κάθε πιθανή ενέργεια κωδικοποιείται στον ΓΑ με έναν ακέραιο αριθμό [0-5]. Η getBestAction κάνει την αποκωδικοποίηση ώστε να εκτελεστεί κάθε φορά η αντίστοιχη ενέργεια.

Η συνάρτηση ποιότητας που χρησιμοποιήθηκε, "βαθμολογεί" τα άτομα-λύσεις με βάση την αποτελεσματικότητά τους σε δύο βασικούς άξονες. Ο πρώτος αφορά την δυνατότητα του avatar να επιβιώσει και ο δεύτερος την ικανότητά του να πλησιάσει τον εκάστοτε στόχο (πχ την έξοδο). Καθώς, ο άμεσος σκοπός στο παιχνίδι Zelda που μελετήθηκε δεν είναι η εξολόθρευση των αντιπάλων, οι εχθροί δεν συμπεριλήφθηκαν στη συνάρτηση ποιότητας για να μην οδηγηθεί ο πράκτορας σε αποπροσανατολισμό από τον βασικό στόχο του παιχνιδιού.

Η μαθηματική έκφραση της συνάρτησης ποιότητας είναι:

$$fitness = isAlive * (0.5 * (1 - portalDist) * gotKey + 0.5 * (1 - keyDist * not(gotKey)))$$

Ο συντελεστής *isAlive* παίρνει την τιμή 1 αν ο παίχτης έχει επιβιώσει και 0.7 αν έχει χάσει. Δεν μηδενίζεται σε περίπτωση ήττας γιατί έτσι δεν θα αξιοποιούνταν οι πληροφορίες που συλλέγουν οι πιο αδύναμοι πράκτορες σχετικά με το περιβάλλον. Όσον αφορά το υπόλοιπο σκέλος, κάθε πράκτορας αξιολογείται με βάση την εύρεση των αντικειμένων-στόχων που είναι το κλειδί και η πόρτα-έξοδος ή την απόσταση του από αυτά σε περίπτωση αποτυχίας.

Τέλος, για την υλοποίηση των γενετικών αλγορίθμων επιλέχθηκαν έπειτα από αρκετές δοκιμές οι εξής παράμετροι:

- **Τελεστής επιλογής**: Truncation
- **Τελεστής διασταύρωσης**: Single-point
- **Τελεστής μετάλλαξης**: Uniform
- **Μέγεθος πληθυσμού**: 70
- **Μέγιστος αριθμός γενιών**: 100

4.2.2 Υλοποίηση 1ου Πράκτορα

Αυτός ο πράκτορας λαμβάνει υπόψιν του δώδεκα σημεία του κόσμου του παιχνιδιού, τα οχτώ που βρίσκονται γύρω από το avatar και τα τέσσερα που βρίσκονται στις βασικές κατευθύνσεις του (πάνω, κάτω, δεξιά και αριστερά) και απέχουν από αυτό δύο θέσεις (Σχήμα 4.9). Κάθε θέση κωδικοποιείται με το αντίστοιχο χρωμόσωμα, συνεπώς κάθε γενότυπος αποτελείται από δώδεκα χρωμοσώματα.

Σχήμα 4.9: Εξεταζόμενα σημεία του κόσμου στον Πρώτο Πράκτορα

Όλες οι θέσεις εξετάζονται για το αν περιλαμβάνουν τοίχο, NPC (εχθρό) ή αν είναι κενές και αλλάζουν τη βαθμολογία των πιθανών ενεργειών ως εξής. Τα χρωμοσώματα που κωδικοποιούν τις θέσεις, αποτελούνται από δώδεκα γονίδια με ακέραιες τιμές στο διάστημα [-2, 0]. Αν η θέση είναι κενή δεν γίνεται καμία αλλαγή και ξεκινάει ο έλεγχος της επόμενης θέσης. Αν περιλαμβάνει τοίχο, τότε οι τιμές των πρώτων έξι γονιδίων, ενός για κάθε πιθανή ενέργεια, προστίθενται στη συνολική βαθμολογία των αντίστοιχων ενεργειών. Αν περιλαμβάνει NPC τότε προστίθενται οι τιμές των επόμενων έξι γονιδίων.

Συνεπώς κάθε γονίδιο σε αυτή την κωδικοποίηση είναι ένας συντελεστής για κάποια από τις πιθανές ενέργειες, και αντιστοιχεί σε ένα συγκεκριμένο sprite και μία συγκεκριμένη σχετική ως προς το avatar θέση. Για παράδειγμα το γονίδιο 4 του χρωμοσώματος 10 είναι ο συντελεστής που θα προστεθεί στην ενέργεια ACTION_RIGHT (σε αυτήν αντιστοιχεί η τιμή 4) αν στη θέση κάτω-δεξιά του

Πίνακας 4.2: Αντιστοιχία γονιδίων ενός χρωμοσώματος με τις πιθανές ενέργειες στον Πρώτο Πράκτορα

	Wall						NPC					
Index	0	1	2	3	4	5	6	7	8	9	10	11
Action	NIL	USE	LEFT	UP	RIGHT	DOWN	NIL	USE	LEFT	UP	RIGHT	DOWN

avatar (σε αυτή τη θέση αντιστοιχεί το χρωμόσωμα 10) βρίσκεται τοίχος. Για την περίπτωση των NPCs η τιμή του γονιδίου x προστίθεται στην ενέργεια που αντιστοιχεί στον ακέραιο $x - 6$, καθώς κωδικοποιούνται από τα γονίδια $6 - 11$ των χρωμοσωμάτων.

Οι τιμές των γονιδίων είναι αρνητικές με τη λογική, οι τοίχοι και οι εχθροί να ορίζουν ποιες ενέργειες είναι λιγότερο ενδεικτικές. Φυσικά το αποτέλεσμα θα ήταν ακριβώς το ίδιο αν οι τιμές ήταν θετικές, για παράδειγμα στο διάστημα [0, 2].

Αφού ολοκληρωθεί αυτή η διαδικασία, προστίθεται ένας *επιπλέον συντελεστής* (bonus) στην ενέργεια προς την κατεύθυνση που βρίσκεται ο στόχος (πχ πόρτα) και επιλέγεται η ενέργεια με τη συνολική μεγαλύτερη βαθμολογία. Για αυτή τη λειτουργία, εκτός από τις κοινές μεθόδους που ορίστηκαν προηγουμένως για όλους τους πράκτορες, υλοποιείται μία επιπλέον μέθοδος *finalizeValues*, η οποία προσδίδει αυτό το τελικό bonus στη βαθμολογία.

4.2.3 Υλοποίηση 2ου Πράκτορα

Σε αυτή την κωδικοποίηση ο πράκτορας εξετάζει τις γειτονικές του θέσεις όχι μία προς μία αλλά ανά τρεις. Σε πολλές περιπτώσεις η προσέγγιση δύο ή τριών θέσεων ανεξάρτητα, όπως στον πράκτορα που περιγράφηκε προηγουμένως, δεν έχει τα ίδια αποτελέσματα με την προσέγγισή τους ως ένα ενιαίο τμήμα του χώρου. Για παράδειγμα ας εξετάσουμε την τριάδα που αποτελείται από τις θέσεις πάνω-δεξιά, δεξιά και κάτω-δεξιά του avatar. Αν υποθέσουμε ότι οι θέσεις πάνω-δεξιά και κάτω-δεξιά περιλαμβάνουν τοίχο και η θέση ανάμεσά τους είναι κενή, στην ανεξάρτητη προσέγγιση των θέσεων ο πράκτορας θα δει δύο τοίχους και ένα κενό, και μάλλον δεν θα επιλέξει να κινηθεί προς τα δεξιά. Αντίθετα αν προσεγγίσει και τις τρεις θέσεις σαν μία ενιαία κατάσταση, είναι πιθανόν κατά την εξέλιξη του γενετικού αλγορίθμου ο πράκτορας να "μάθει" ότι η συγκεκριμένη τριάδα ευνοεί την κίνηση προς τα δεξιά.

Η λειτουργία του πράκτορα έχει ως εξής. Αρχικά γίνεται ένας απλός έλεγχος για να απορριφθούν πολύ προφανείς "λανθασμένες" κινήσεις όπως η μετακίνηση προς κατεύθυνση που υπάρχει τοίχος ή αντίπαλος. Πρόκειται για κινήσεις που θα καθυστερούσαν την εξέλιξη της εκπαίδευσης για αρκετές γενιές μέχρι να κωδικοποιηθούν κάποιες πολύ απλές καταστάσεις, οπότε γίνονται αυτόματα.

Όσον αφορά το υπόλοιπο σκέλος του αλγορίθμου, εξετάζονται συνολικά 28 τριάδες θέσεων γύρω από το avatar (7 τριάδες προς κάθε κατεύθυνση), καθεμία από τις οποίες αυξάνει κατά ένα την συνολική βαθμολογία μίας από τις πιθανές κινήσεις. Σε κάθε περίπτωση ελέγχονται μόνο οι τριάδες που βρίσκονται προς την κατεύθυνση που είναι και ο στόχος. Για παράδειγμα αν ο στόχος μία δεδομένη στιγμή είναι το κλειδί, και βρίσκεται πάνω-αριστερά σε σχέση με το avatar, θα ελεγχθούν οι 14 τριάδες που βρίσκονται πάνω και αριστερά του αντίστοιχα. Στο Σχήμα 4.10 φαίνονται οι 7 από τις 28 τριάδες που βρίσκονται δεξιά του avatar. Οι υπόλοιπες είναι εντελώς συμμετρικές προς τις άλλες 3 κατευθύνσεις.

Θεωρώντας ότι κάθε θέση μπορεί να αποτελείται από 3 διαφορετικά sprites (κενό, τοίχο ή NPC), η κάθε τριάδα μπορεί να έχει 27 διαφορετικές καταστάσεις (πχ τοίχος-τοίχος-NPC, NPC-κενό-τοίχος κλπ). Αυτός είναι και ο λόγος που δεν ομαδοποιούμε περισσότερες θέσεις μαζί, αφού για περισσότερες από τρεις αυξάνεται υπερβολικά ο αριθμός των διαφορετικών καταστάσεων και θα απαιτούνταν πάρα πολλές γενιές μέχρι ο γενετικός αλγόριθμος να βρει όλα τα επιθυμητά γονίδια. Στην ιδανική περίπτωση θα ελέγχαμε όλες τις θέσεις του κόσμου του παιχνιδιού μαζί, όμως για έναν κόσμο που αποτελείται από εκατό θέσεις για παράδειγμα, θα υπήρχαν 3^{100} πιθανές διαφορετικές καταστάσεις, κάτι που όπως γίνεται εύκολα αντιληπτό, είναι αδύνατο να κωδικοποιηθεί αποτελεσματικά από έναν ΓΑ.

Σχήμα 4.10: Ομαδοποίηση των σημείων του κόσμου στον Δεύτερο Πράκτορα

Επομένως κάθε γενότυπος περιλαμβάνει 28 χρωμοσώματα (ένα για κάθε τριάδα θέσεων), καθένα από τα οποία αποτελείται από 27 γονίδια (ένα για κάθε πιθανή κατάσταση της συγκεκριμένης τριάδας). Το κάθε γονίδιο είναι ένας ακέραιος μεταξύ 0 και 5 και αντιστοιχεί σε μία από τις δυνατές κινήσεις. Όταν διαπιστωθεί η κατάσταση της κάθε τριάδας, αυξάνεται κατά ένα η συνολική βαθμολογία της ενέργειας που ορίζει το αντίστοιχο γονίδιο. Τελικά επιλέγεται η κίνηση με την μεγαλύτερη αξία (μεταξύ αυτών που έχουν απομείνει μετά τον πρώτο αυτόματο έλεγχο). Αν υπάρχουν περισσότερες από μία κινήσεις με την ίδια αξία, τότε επιλέγεται μία από αυτές τυχαία.

Για τον συγκεκριμένο πράκτορα υλοποιήθηκαν κάποιες επιπλέον μέθοδοι, κυρίως για τη δημιουργία των τριάδων με συμμετρικό τρόπο, για τις τέσσερις κατευθύνσεις, Η σημαντικότερη όμως νέα μέθοδος είναι η *firstCheck*, η οποία πραγματοποιεί τον αρχικό έλεγχο για τις απλές, τετριμμένες κινήσεις που απορρίπτονται στο ξεκίνημα της διαδικασίας.

4.2.4 Υλοποίηση 3ου Πράκτορα

Ο τρίτος και τελευταίος πράκτορας που υλοποιήθηκε είναι παρόμοιος με τον δεύτερο, υπό την έννοια ότι ελέγχει επίσης τριάδες θέσεων, έχει όμως ορισμένες διαφορές.

Κατ' αρχάς, στον γενότυπό του, υπάρχουν δύο επιπλέον χρωμοσώματα αποκλειστικά για την κωδικοποίηση των καταστάσεων στις οποίες υπάρχει NPC σε κοντινή απόσταση από το avatar. Αυτή η προσθήκη γίνεται για δύο βασικούς λόγους. Ο ένας είναι ότι η άμεση ανταπόκριση στην ύπαρξη ενός NPC είναι καθοριστική καθώς υπάρχει ενδεχόμενος κίνδυνος, κάτι που δεν ισχύει για παράδειγμα στην περίπτωση ενός τοίχου και επομένως έχει νόημα να εξετάζεται ξεχωριστά αυτή η περίπτωση με μεγαλύτερη βαρύτητα. Ο δεύτερος λόγος είναι ότι έχοντας ελέγξει ήδη την ύπαρξη NPCs, δεν χρειάζεται να τους λάβουμε υπόψιν στην κωδικοποίηση των θέσεων. Κατά συνέπεια κάθε θέση μπορεί να έχει πλέον δύο πιθανές καταστάσεις (κενό ή τοίχος), που σημαίνει ότι οι πιθανές διαφορετικές καταστάσεις κάθε τριάδας είναι οχτώ, μειώνοντας έτσι κατά πολύ τον αριθμό των γονιδίων κάθε χρω-

μοσώματος.

Στόχος της παραπάνω αλλαγής, είναι να κωδικοποιηθούν οι περιπτώσεις που υπάρχει NPC ακριβώς δίπλα στο avatar ή σε απόσταση δύο θέσεων από αυτό, δηλαδή όταν υπάρχει άμεση απειλή. Τα αντίστοιχα χρωμοσώματα αποτελούνται από 7 γονίδια. Το πρώτο γονίδιο αντιστοιχεί στην κατάσταση που το avatar έχει κατεύθυνση προς τον NPC και τα υπόλοιπα στις καταστάσεις που έχει διαφορετική κατεύθυνση όπως φαίνεται στον Πίνακα 4.3.

Πίνακας 4.3: Κωδικοποίηση των δύο χρωμοσωμάτων ελέγχου NPCs στον Τρίτο Πράκτορα

Index	0	1	2	3	4	5	6
Direction	NPC	δεξιά του NPC	δεξιά του NPC	αντίθετα από το NPC	αντίθετα από το NPC	αριστερά του NPC	αριστερά του NPC
NPC σε αυτή την κατεύθυνση	✓	✓		✓		✓	

Άλλη μία αλλαγή που εξυπηρετεί τη μείωση της κωδικοποιημένης πληροφορίας είναι ότι σε αυτή την περίπτωση κωδικοποιούνται μόνο οι 7 διαφορετικές τριάδες που βρίσκονται στη δεξιά κατεύθυνση του avatar. Για τις υπόλοιπες κατευθύνσεις, η κάθε τριάδα αυξάνει τη βαθμολογία της ενέργειας που είναι συμμετρική με την ενέργεια που ορίζει η αντίστοιχη τριάδα θέσεων στη δεξιά κατεύθυνση. Αν για παράδειγμα μία τριάδα θέσεων δεξιά του avatar, αυξάνει όταν βρίσκεται σε κατάσταση τοίχος-τοίχος-κενό τη βαθμολογία της ενέργειας ACTION_DOWN, η αντίστοιχη τριάδα στα αριστερά του avatar θα αυξάνει τη βαθμολογία της ενέργειας ACTION_UP.

Στον δεύτερο πράκτορα που περιγράφηκε στην Υποενότητα 4.2.3, οι τριάδες δημιουργούνταν με συμμετρικό τρόπο για τις τέσσερις κατευθύνσεις, όμως η καθεμία είχε το δικό της αυτόνομο χρωμόσωμα. Αντίθετα, στον τρίτο πράκτορα οι τέσσερις συναφείς τριάδες (μία σε κάθε κατεύθυνση), "μοιράζονται" το ίδιο χρωμόσωμα και τροποποιούν κατάλληλα τα γονίδια του ώστε να τα προσαρμόσουν στην εκάστοτε κατεύθυνση.

Για την υλοποίηση του τρίτου πράκτορα προστέθηκε η μέθοδος *npcUpdateValues*, η οποία ανανεώνει τη βαθμολογία των ενεργειών με βάση την ύπαρξη NPCs και την κατεύθυνση του avatar, όπως και κάποιες μέθοδοι για τον σχηματισμό των συμμετρικών ενεργειών στις περιπτώσεις που ο στόχος δεν βρίσκεται στη δεξιά κατεύθυνση.

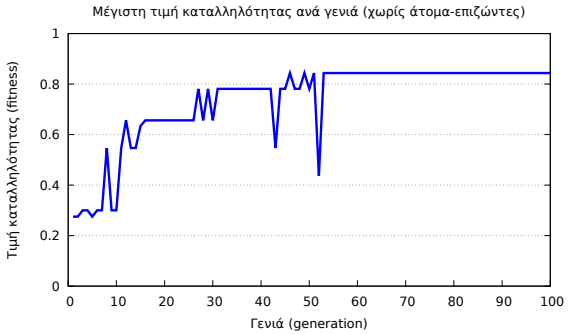
Κεφάλαιο 5

Πειραματική διαδικασία

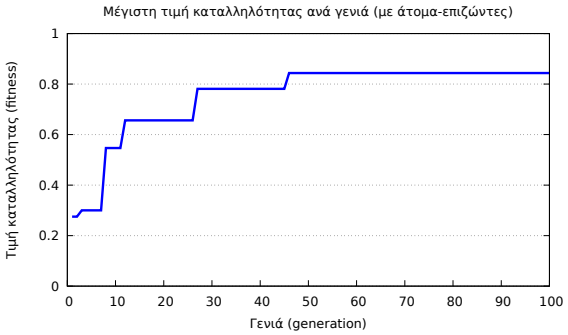
5.1 Αποτελέσματα 1ου Πράκτορα

Για την αξιολόγηση των διαφορετικών πρακτόρων που υλοποιήθηκαν, συγκεντρώθηκαν κατά την εκτέλεση των γενετικών αλγορίθμων στατιστικά σχετικά με τις τιμές καταλληλότητας, τον αριθμό βημάτων των παιχνιδιών και τον αριθμό των παιχνιδιών στα οποία οι πράκτορες κέρδισαν. Η μέγιστη τιμή καταλληλότητας που μπορεί να επιτευχθεί με τη συνάρτηση ποιότητας που χρησιμοποιείται είναι 1, ενώ ο μέγιστος επιτρεπόμενος αριθμός βημάτων σε κάθε παιχνίδι είναι 1000.

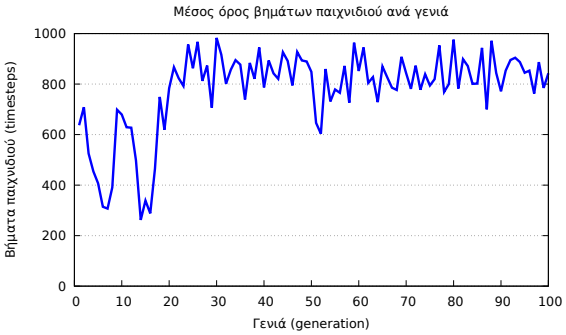
Στη συνέχεια παρουσιάζονται τα αντίστοιχα γραφήματα για τον πρώτο πράκτορα.



(a) Μέγιστη τιμή καταλληλότητας (χωρίς άτομα-επιζώντες)



(b) Μέγιστη τιμή καταλληλότητας (με άτομα-επιζώντες)



(c) Μέσος όρος βημάτων παιχνιδιού

Σχήμα 5.1: Αποτελέσματα 1ου πράκτορα ανά γενιά

Στους γενετικούς αλγορίθμους, ο πληθυσμός κάθε γενιάς αποτελείται από άτομα που έχουν επιβιώσει από προηγούμενες γενιές και από νέα άτομα που προκύπτουν από τη διαδικασία της αναπαραγωγής. Στο Σχήμα 5.1a, φαίνονται οι μέγιστες τιμές καταλληλότητας που προκύπτουν μόνο από τα νέα άτομα κάθε γενιάς. Αυτό εξηγεί και το γεγονός ότι υπάρχουν αυξομειώσεις στο διάγραμμα. Αν λάβουμε υπόψη σε κάθε γενιά και τα *άτομα-επιζώντες* (survivors) από τις προηγούμενες γενιές, τότε προφανώς η βέλτιστη τιμή καταλληλότητας για κάθε γενιά θα είναι τουλάχιστον ίση με αυτή της προηγούμενης. Στο Σχήμα 5.1b φαίνεται αυτή η προσέγγιση, ώστε να είναι πιο ξεκάθαρος ο ρυθμός

αύξησης.

Παρατηρείται ότι κατά την εκτέλεση του ΓΑ η τιμή καταλληλότητας αυξάνεται στις πρώτες 50 περίπου γενιές και έπειτα παραμένει σταθερή στην τιμή 0.84. Αυτό δείχνει ότι δεν υπάρχει άλλο περιθώριο βελτίωσης και σημαίνει ότι έχει βρεθεί η καλύτερη λύση για τη συγκεκριμένη υλοποίηση.

Στο Σχήμα 5.1c παρουσιάζεται ο χρόνος που απαιτείται κατά μέσο όρο για την ολοκλήρωση της εκτέλεσης του κάθε πράκτορα ανά γενιά και φαίνεται ότι ο μέσος όρος βημάτων παιχνιδιού αρχικά αυξάνεται και στις μεγαλύτερες γενιές εν μέρει σταθεροποιείται. Αυτό είναι λογικό με βάση την εξέλιξη της καμπύλης καταλληλότητας καθώς οι πράκτορες βελτιώνονται στις πρώτες γενιές με αποτέλεσμα να αντέχουν για περισσότερο χρόνο στο παιχνίδι, ενώ στις μεγαλύτερες γενιές που δεν υπάρχει άλλη βελτίωση εμφανίζουν παρόμοια διάρκεια στα παιχνίδια τους.

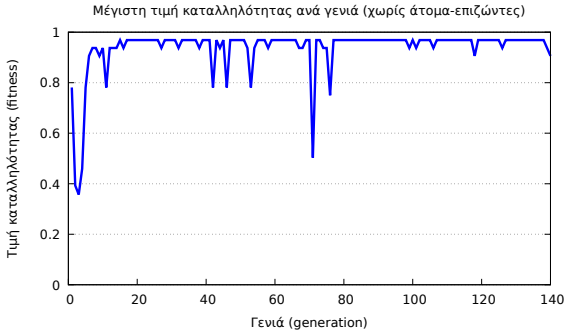
Τέλος, καθώς ο συγκεκριμένος πράκτορας δεν κατάφερε να ολοκληρώσει με επιτυχία το επίπεδο στο οποίο εφαρμόστηκε ο ΓΑ, δεν έχει νόημα η παρουσίαση του ποσοστού νίκης του.

5.2 Αποτελέσματα 2ου Πράκτορα

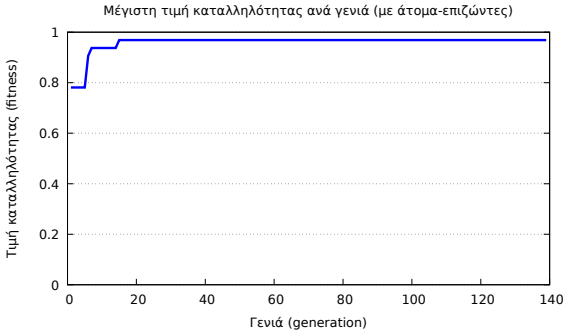
5.2.1 Εκτέλεση ΓΑ - Εκπαίδευση

Στη συνέχεια συλλέξαμε αποτελέσματα από την υλοποίηση του δεύτερου πράκτορα, ο οποίος εξετάζει τις θέσεις του κόσμου ανά τριάδες.

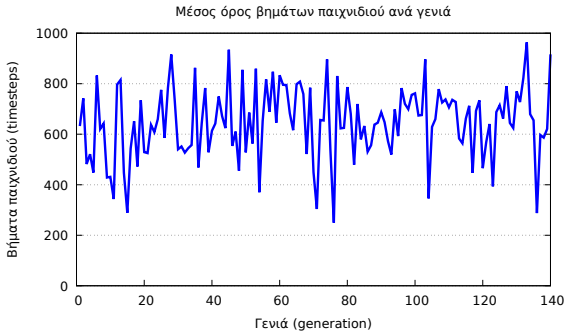
Στα διαγράμματα που ακολουθούν παρουσιάζονται οι τιμές καταλληλότητας των καλύτερων πρακτόρων που προέκυψαν (με τις δύο διαφορετικές προσεγγίσεις), ο μέσος όρος των βημάτων που χρειάστηκαν οι πράκτορες και το ποσοστό νίκης τους ανά γενιά.



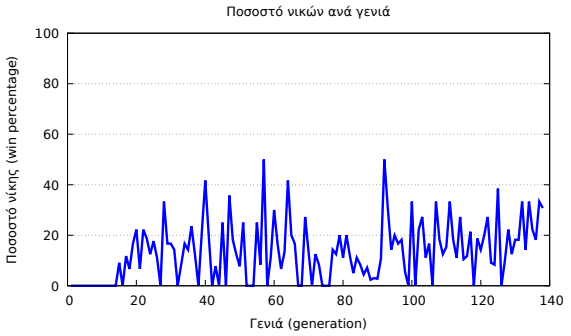
(a) Μέγιστη τιμή καταλληλότητας (χωρίς άτομα-επιζώντες)



(b) Μέγιστη τιμή καταλληλότητας (με άτομα-επιζώντες)



(c) Μέσος όρος βημάτων παιχνιδιού



(d) Ποσοστό νικών

Σχήμα 5.2: Αποτελέσματα 2ου πράκτορα ανά γενιά

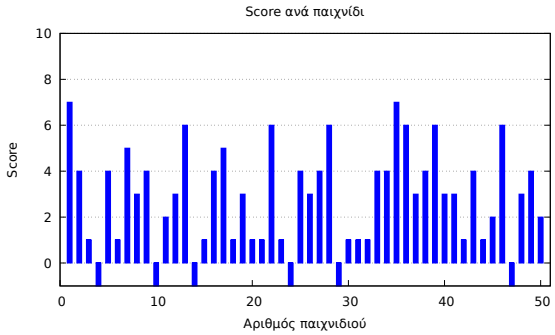
Σε αυτή την υλοποίηση οι πράκτορες πετυχαίνουν από πολύ νωρίς τιμή καταλληλότητας που πλησιάζει τη μονάδα. Αποδεικνύεται έτσι ότι η προσέγγιση των σημείων του κόσμου ανά ομάδες είναι ιδιαίτερα σημαντική για την σωστή κωδικοποίηση των καταστάσεων του παιχνιδιού. Αυτό φαίνεται

και από το ποσοστό νίκης των πρακτόρων που φτάνει μέχρι και το 50% σε αντίθεση με την προηγούμενη υλοποίηση όπου τα σημεία είχαν εξεταστεί το καθένα ξεχωριστά και δεν υπήρξαν πράκτορες ικανοί να κερδίσουν. Όσον αφορά το χρόνο εκτέλεσης, από το Σχήμα 5.2c προκύπτει ότι παρά τη μεγάλη αύξηση της αποτελεσματικότητας δεν υπήρξε ιδιαίτερη βελτίωση των πρακτόρων σε αυτό το κομμάτι.

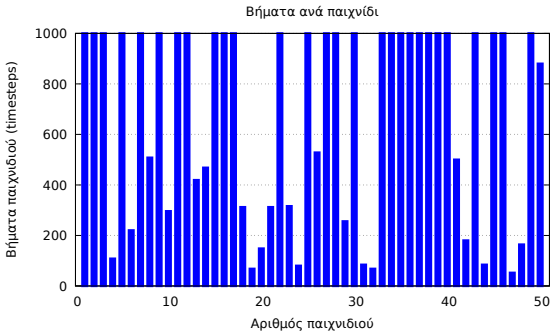
5.2.2 Δοκιμή

Αφού ολοκληρώθηκε ο γενετικός αλγόριθμος, δοκιμάσαμε τον "καλύτερο" πράκτορα που προέκυψε, στα 5 διαθέσιμα διαφορετικά επίπεδα του παιχνιδιού, από 50 φορές στο καθένα και συγκεντρώσαμε τα scores, τις νίκες και τον αριθμό των βημάτων εκτέλεσης σε κάθε περίπτωση. Κατά την *δοκιμή* (testing) προστέθηκε στον πράκτορα η δυνατότητα επιλογής τυχαίων κινήσεων σε περιπτώσεις που "εγκλωβίζεται" τοπικά. Αυτή η δυνατότητα δεν υπήρχε κατά τη διάρκεια της "εκπαίδευσης" μέσω της εκτέλεσης του ΓΑ, καθώς η τυχαιότητα θα επηρέαζε αρνητικά τον προσδιορισμό των επιθυμητών παραμέτρων.

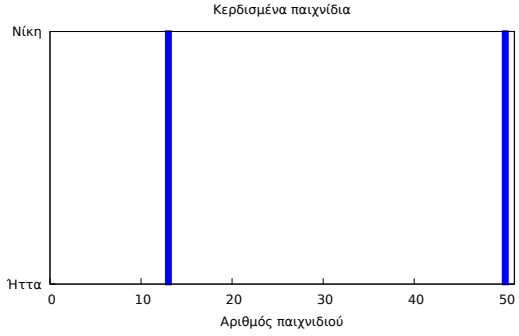
Ακολουθούν τα διαγράμματα που προέκυψαν από τα στοιχεία των δοκιμών για κάθε επίπεδο.



(a) Score

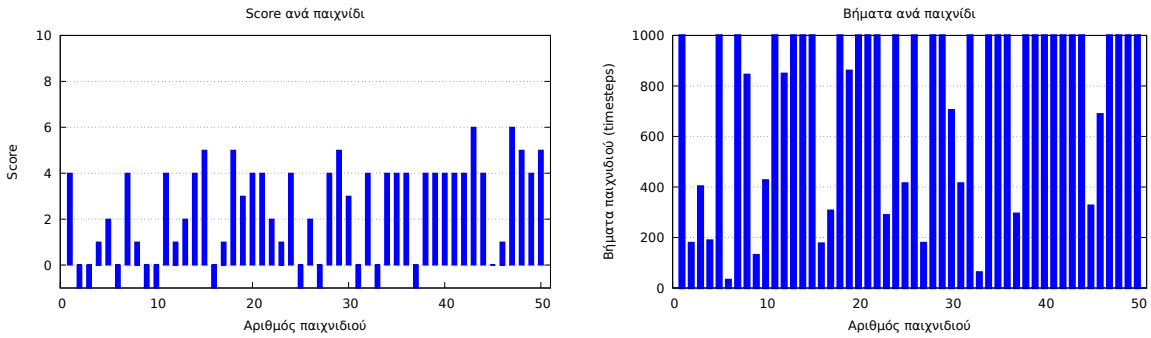


(b) Πλήθος βημάτων

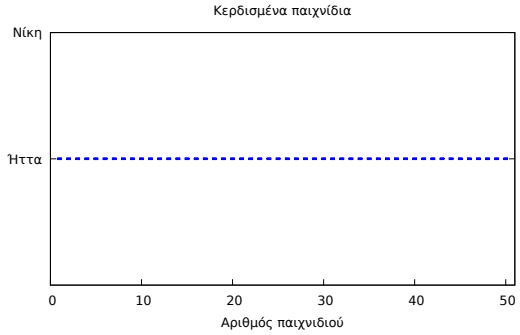


(c) Κερδισμένα παιχνίδια

Σχήμα 5.3: Αποτελέσματα του καλύτερου 2^{ου} πράκτορα ανά παιχνίδι (Επίπεδο 0)

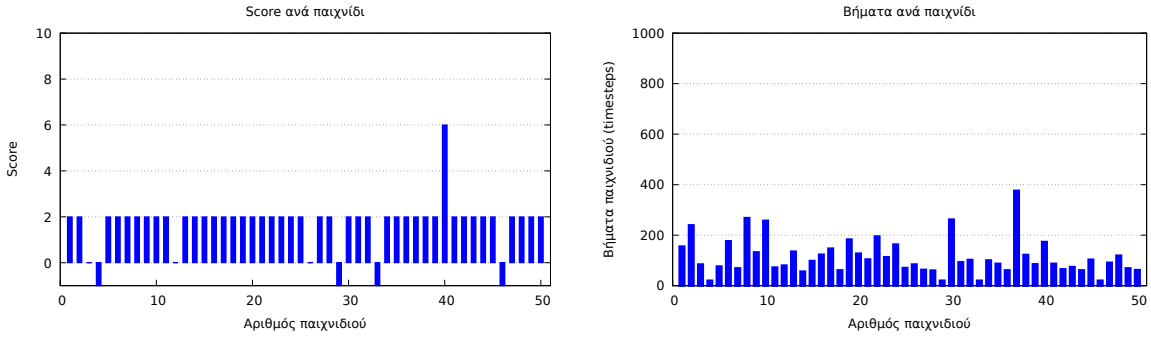


(a) Score **(b)** Πλήθος βημάτων

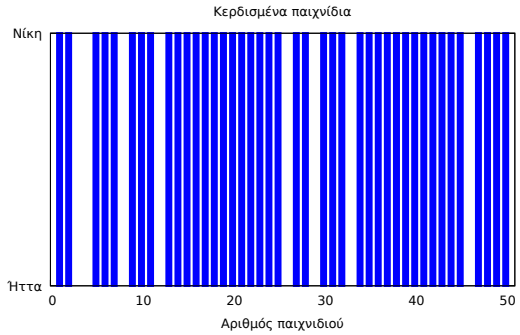


(c) Κερδισμένα παιχνίδια

Σχήμα 5.4: Αποτελέσματα του καλύτερου 2ου πράκτορα ανά παιχνίδι (Επίπεδο 1)

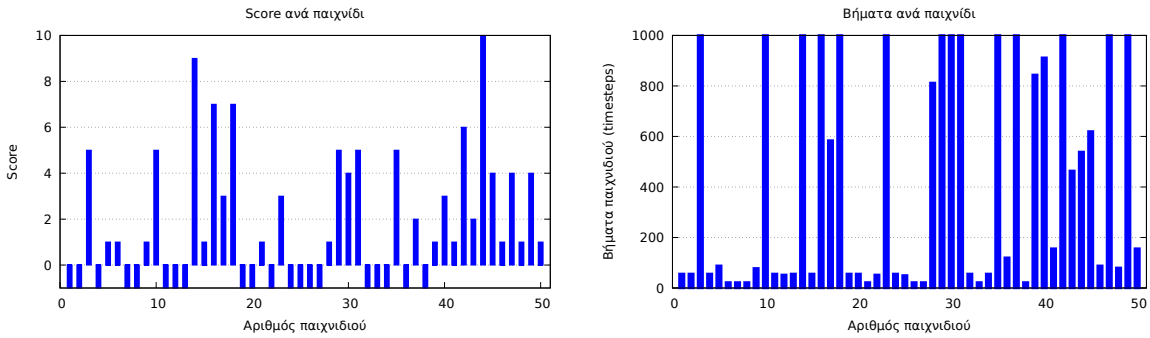


(a) Score **(b)** Πλήθος βημάτων

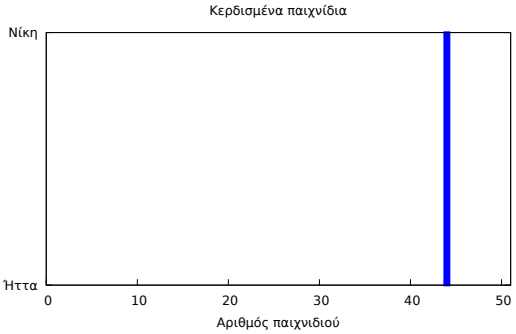


(c) Κερδισμένα παιχνίδια

Σχήμα 5.5: Αποτελέσματα του καλύτερου 2ου πράκτορα ανά παιχνίδι (Επίπεδο 2)

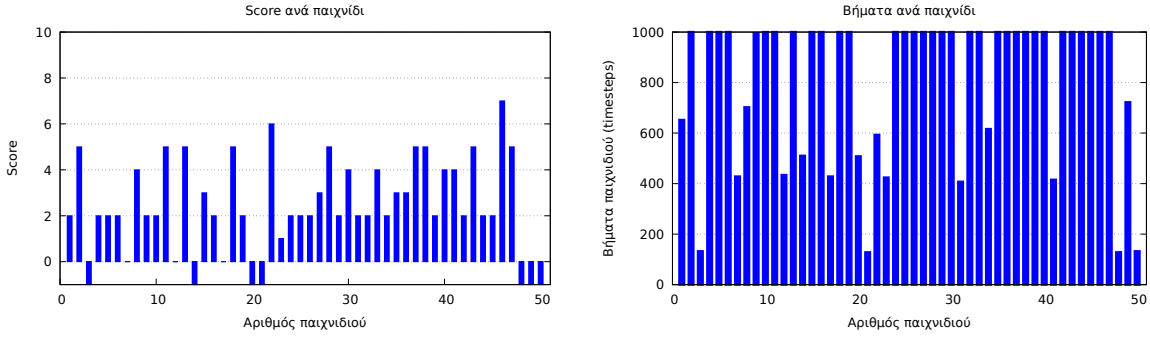


(a) Score **(b)** Πλήθος βημάτων

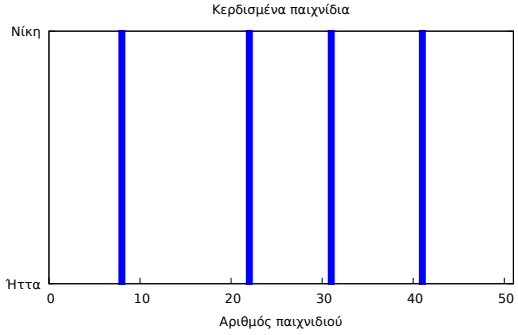


(c) Κερδισμένα παιχνίδια

Σχήμα 5.6: Αποτελέσματα του καλύτερου 2^{ου} πράκτορα ανά παιχνίδι (Επίπεδο 3)



(a) Score **(b)** Πλήθος βημάτων



(c) Κερδισμένα παιχνίδια

Σχήμα 5.7: Αποτελέσματα του καλύτερου 2^{ου} πράκτορα ανά παιχνίδι (Επίπεδο 4)

Από τις παραπάνω γραφικές γίνεται αντιληπτό ότι παρόλο που ο πράκτορας είναι ιδιαίτερα αποτελεσματικός στο Επίπεδο 2, το οποίο είναι αυτό που χρησιμοποιήθηκε κατά τη διάρκεια της "εκπαί-

δευσής" του έχοντας ποσοστό νίκης 84%, δεν ανταποκρίνεται το ίδιο καλά και στα υπόλοιπα επίπεδα, όπου το ποσοστό του δεν ξεπερνάει το 10%. Αυτό οφείλεται στο γεγονός ότι στη συγκεκριμένη τεχνική λαμβάνονται κάθε φορά υπόψιν οι θέσεις του κόσμου που βρίσκονται προς την κατεύθυνση του εκάστοτε στόχου (πχ του κλειδιού). Συνεπώς κατά την εκτέλεση του γενετικού αλγορίθμου, προσδιορίστηκαν αποτελεσματικά οι παράμετροι που κωδικοποιούν την κατάσταση των θέσεων προς την κατεύθυνση των στόχων του συγκεκριμένου επιπέδου, όμως δεν συνέβη το ίδιο και για τις υπόλοιπες θέσεις οι οποίες πιθανώς είναι απαραίτητες σε κάποιο άλλο επίπεδο.

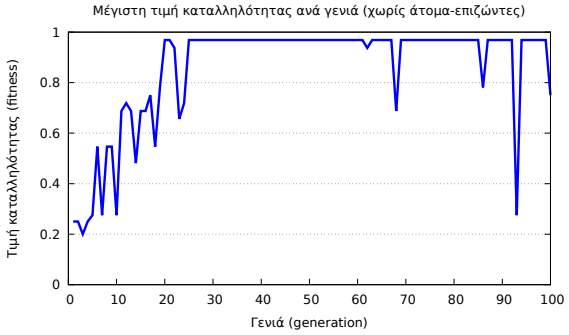
Ένα στοιχείο που αξίζει να σημειωθεί είναι ότι στο Επίπεδο 2, όπου ο πράκτορας έχει το μεγαλύτερο ποσοστό νίκης και το μικρότερο μέσο όρο βημάτων, εμφανίζει το μικρότερο score. Αυτό συμβαίνει επειδή στον υπολογισμό του score δεν προσμετράται το αποτέλεσμα του παιχνιδιού αλλά μόνο ο αριθμός των εχθρών που εξολοθρεύτηκαν και η εύρεση του κλειδιού. Συνεπώς στο Επίπεδο 2 όπου ο πράκτορας τερματίζει πιο γρήγορα, πιθανότατα συναντά λιγότερους εχθρούς απ' ότι στα υπόλοιπα επίπεδα και καταλήγει να συγκεντρώνει χαμηλότερο score.

5.3 Αποτελέσματα 3ου Πράκτορα

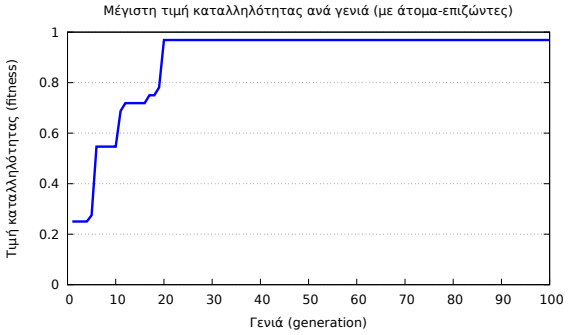
5.3.1 Εκτέλεση ΓΑ - Εκπαίδευση

Τέλος εξετάστηκε με βάση τις προηγούμενες μετρικές και η απόδοση του 3ου πράκτορα, ο οποίος σχεδιάστηκε με στόχο να αντιμετωπίσει το πρόβλημα γενίκευσης σε περισσότερα επίπεδα, που εντοπίστηκε στον 2ο πράκτορα. Προκειμένου να επιτευχθεί αυτό, ο προσδιορισμός των παραμέτρων γίνεται με συμμετρικό τρόπο και για τις τέσσερις κατευθύνσεις του χώρου (πάνω, κάτω, δεξιά, αριστερά) έτσι ώστε η εύρεση των παραμέτρων σε μία κατεύθυνση να είναι ικανή να χρησιμοποιηθεί στη συνέχεια και στις υπόλοιπες.

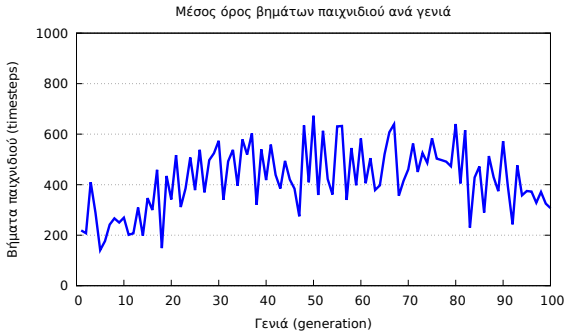
Κατ' αρχάς, παρουσιάζονται και εδώ οι τιμές καταλληλότητας των καλύτερων πρακτόρων, τα βήματα των παιχνιδιών και το ποσοστό νίκης ανά γενιά.



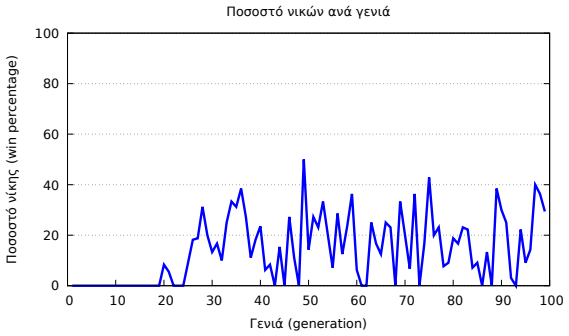
(a) Μέγιστη τιμή καταλληλότητας (χωρίς άτομα-επιζώντες)



(b) Μέγιστη τιμή καταλληλότητας (με άτομα-επιζώντες)



(c) Μέσος όρος βημάτων παιχνιδιού



(d) Ποσοστό νικών

Σχήμα 5.8: Αποτελέσματα 3ου πράκτορα ανά γενιά

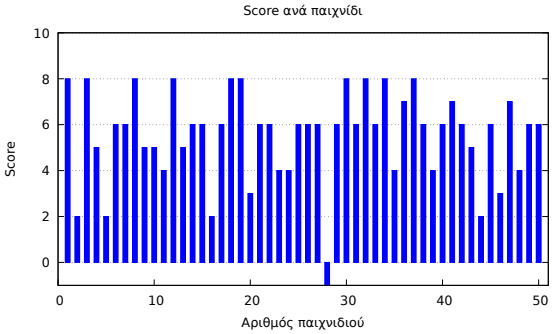
Και σε αυτή την περίπτωση ο πράκτορας βελτιώνεται αρκετά γρήγορα, και από την 20η γενιά εμφανίζει τιμή καταλληλότητας κοντά στη μονάδα. Όμως σε αυτή την υλοποίηση είναι πιο ενδιαφέροντα τα συμπεράσματα που προκύπτουν από τον μέσο αριθμό των βημάτων ανά γενιά.

Αρχικά, παρατηρείται αύξηση των βημάτων εκτέλεσης στις πρώτες 50-60 γενιές. Αυτό συμβαίνει γιατί καθώς οι πράκτορες βελτιώνονται μπορούν και "αντέχουν" για περισσότερο χρόνο στο παιχνίδι. Στις πρώτες 20 γενιές, όπου ακόμα οι πράκτορες δεν έχουν προσαρμοστεί φαίνεται ότι χάνουν σχετικά γρήγορα γι' αυτό και ο μέσος όρος των βημάτων παιχνιδιού είναι σημαντικά μικρότερος, κάτι που έρχεται σε συμφωνία και με τις τιμές καταλληλότητας για τις συγκεκριμένες γενιές. Από την 80η γενιά και έπειτα ωστόσο υπάρχει εκ νέου μείωση του χρόνου εκτέλεσης, που υποδεικνύει ότι οι πράκτορες σταδιακά βελτιώνουν την απόδοση τους όσον αφορά το χρόνο που χρειάζονται για να ολοκληρώσουν το επίπεδο.

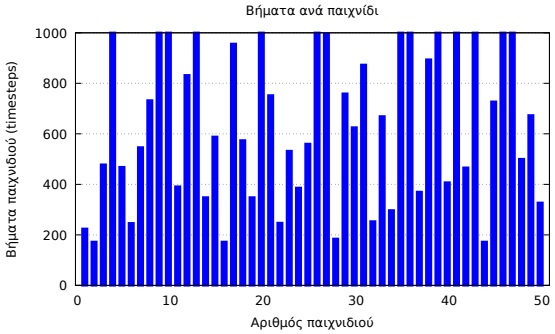
Το ποσοστό νικών κυμαίνεται κυρίως μεταξύ 20% - 40% ενώ φτάνει μέχρι και 50% στην καλύτερη περίπτωση.

5.3.2 Δοκιμή

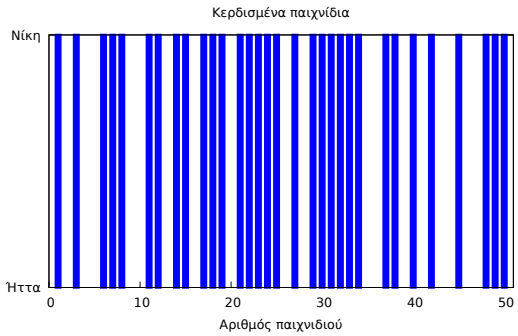
Ακριβώς όπως και στην 2η υλοποίηση, ο καλύτερος πράκτορας δοκιμάστηκε από 50 φορές σε κάθε επίπεδο και μετρήθηκαν το score, οι νίκες και ο χρόνος που χρειάστηκε για να τα ολοκληρώσει.



(a) Score

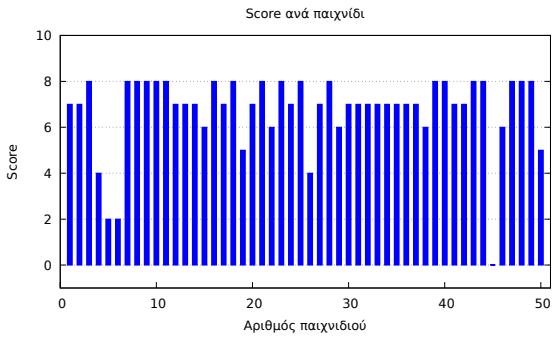


(b) Πλήθος βημάτων

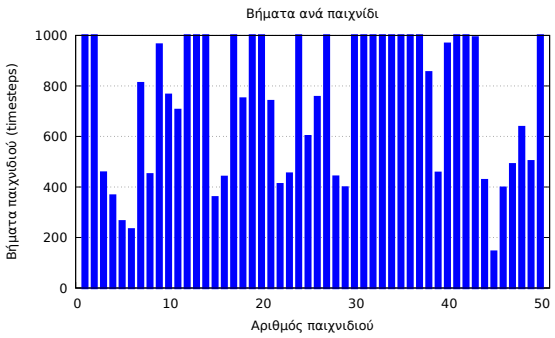


(c) Κερδισμένα παιχνίδια

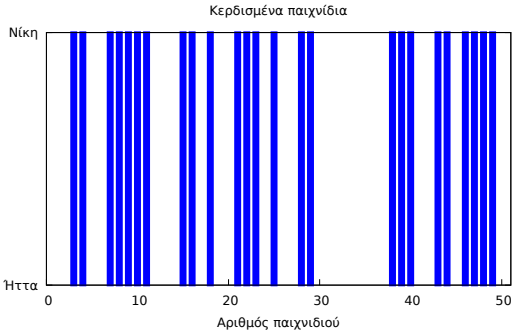
Σχήμα 5.9: Αποτελέσματα του καλύτερου 3ου πράκτορα ανά παιχνίδι (Επίπεδο 0)



(a) Score

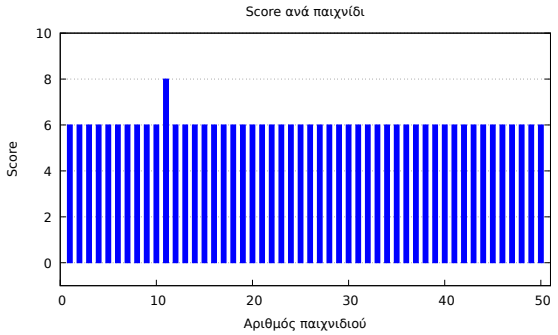


(b) Πλήθος βημάτων

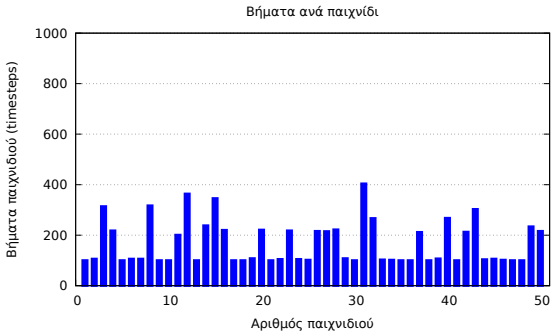


(c) Κερδισμένα παιχνίδια

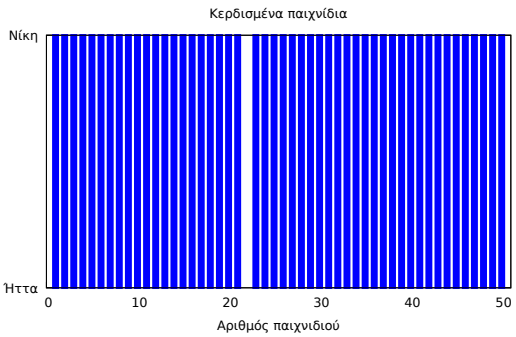
Σχήμα 5.10: Αποτελέσματα του καλύτερου 3^{ου} πράκτορα ανά παιχνίδι (Επίπεδο 1)



(a) Score

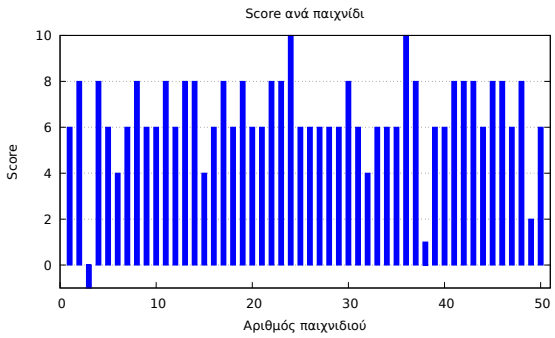


(b) Πλήθος βημάτων

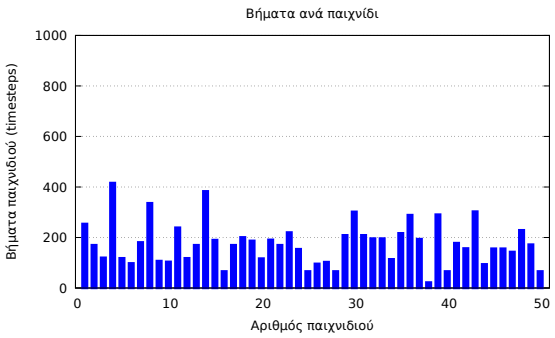


(c) Κερδισμένα παιχνίδια

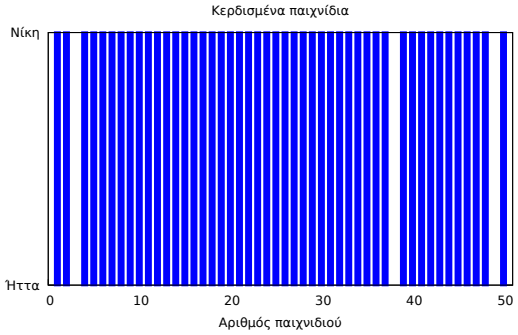
Σχήμα 5.11: Αποτελέσματα του καλύτερου 3^{ου} πράκτορα ανά παιχνίδι (Επίπεδο 2)



(a) Score

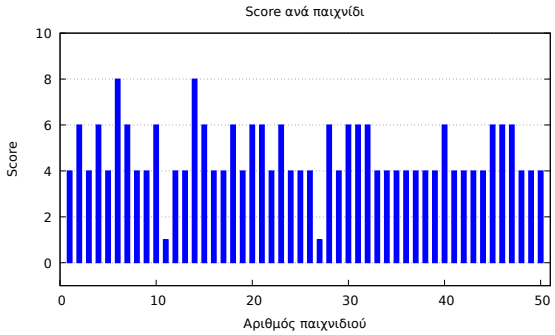


(b) Πλήθος βημάτων

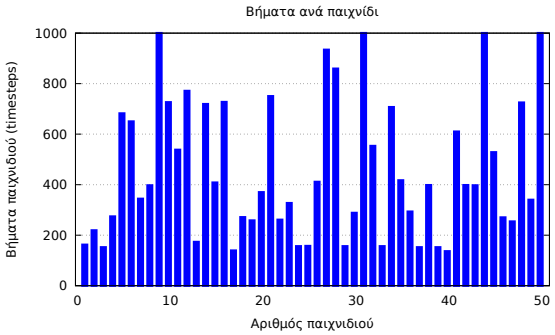


(c) Κερδισμένα παιχνίδια

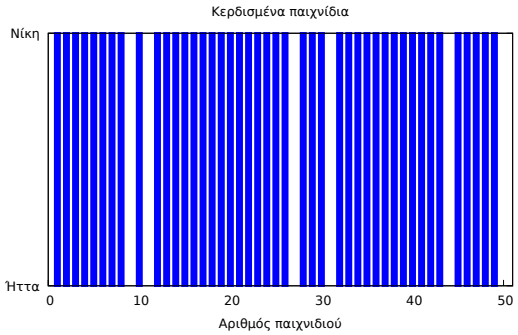
Σχήμα 5.12: Αποτελέσματα του καλύτερου 3ου πράκτορα ανά παιχνίδι (Επίπεδο 3)



(a) Score



(b) Πλήθος βημάτων

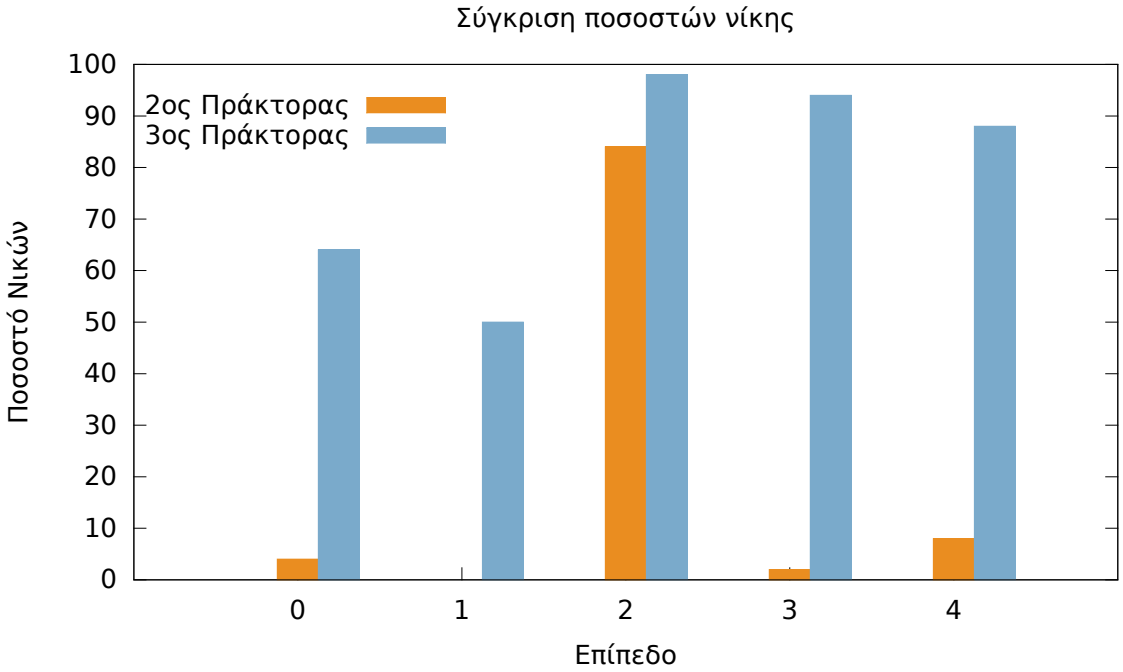


(c) Κερδισμένα παιχνίδια

Σχήμα 5.13: Αποτελέσματα του καλύτερου 3ου πράκτορα ανά παιχνίδι (Επίπεδο 4)

Είναι εμφανής η βελτίωση της ικανότητας του πράκτορα να γενικευθεί σε περισσότερα επίπεδα, σε σχέση με την προηγούμενη υλοποίηση που δοκιμάστηκε. Βέβαια και σε αυτή την περίπτωση ο πράκτορας ανταποκρίνεται καλύτερα στο επίπεδο στο οποίο εκπαιδεύτηκε (Επίπεδο 2), όπου το ποσοστό επιτυχίας του είναι 98%, όμως και στα υπόλοιπα επίπεδα τα οποία είναι "άγνωστα" έχει ποσοστό επιτυχίας τουλάχιστον 50%, το οποίο μάλιστα στα περισσότερα επίπεδα κυμαίνεται μεταξύ 80% - 90%.

Εκτός από τη δυνατότητα γενίκευσης, η τρίτη υλοποίηση του πράκτορα εμφανίζει μεγαλύτερη αποτελεσματικότητα από τη δεύτερη και στο επίπεδο που έγινε η εκπαίδευση. Στο Σχήμα 5.14 παρουσιάζονται συγκεντρωτικά τα ποσοστά επιτυχίας των πρακτόρων 2 και 3 σε κάθε επίπεδο ξεχωριστά, όπου φαίνεται η συνολική βελτίωση που παρουσιάζει ο 3ος πράκτορας.



Σχήμα 5.14: Σύγκριση ποσοστών νίκης για τον 2ο και 3ο Πράκτορα

Κεφάλαιο 6

Συμπεράσματα και Μελλοντικές Κατευθύνσεις

6.1 Συμπεράσματα

Από την πειραματική διαδικασία και τα αποτελέσματα της συγκεκριμένης εργασίας προέκυψε ότι η χρήση Γενετικών Αλγορίθμων για την κωδικοποίηση των καταστάσεων του κόσμου σε ένα ηλεκτρονικό παιχνίδι, είναι μία αποτελεσματική μέθοδος για την ανάπτυξη πρακτόρων, ικανών να γενικευθούν και να κερδίσουν σε διαφορετικά επίπεδα.

Η αποδοτικότητα των πρακτόρων που βασίζονται σε ΓΑ διαφέρει ανάλογα με την τεχνική που ακολουθείται.

Στη συνηθισμένη μέθοδο, κατά την οποία οι ΓΑ κωδικοποιούν την ακολουθία των κινήσεων του avatar, παρατηρείται σημαντικό πρόβλημα στην εφαρμογή του πράκτορα σε διαφορετικά επίπεδα. Το αποτέλεσμα είναι ο πράκτορας να "μαθαίνει" το επίπεδο στο οποίο εκτελείται ο αλγόριθμος και να προσαρμόζει τις κινήσεις του αποκλειστικά στο συγκεκριμένο μοτίβο. Στην παρούσα εργασία, για την αντιμετώπιση αυτού του προβλήματος υλοποιήθηκαν πράκτορες οι οποίοι μέσω γενετικών αλγορίθμων κωδικοποιούν τις καταστάσεις στις οποίες μπορεί να βρεθεί ο κόσμος του παιχνιδιού. Καθώς οι καταστάσεις του κόσμου δεν εξαρτώνται αποκλειστικά από την τοπολογία και τη δομή του κάθε επιπέδου, η κωδικοποίησή τους οδηγεί σε πιο ευέλικτους πράκτορες που μπορούν να χρησιμοποιηθούν και σε άλλες πίστες, εκτός από αυτή όπου "εκπαιδεύτηκαν".

Παρατηρείται ότι ακόμα και με αυτή την κωδικοποίηση ο πράκτορας, παρότι εμφανίζει ικανοποιητικά ποσοστά επιτυχίας σε όλα τα επίπεδα έχει ένα πλεονέκτημα στο επίπεδο για το οποίο εφαρμόστηκε ο γενετικός αλγόριθμος. Αυτό οφείλεται κυρίως στο γεγονός ότι σε κάθε επίπεδο, πέραν των γενικών και περισσότερο συνηθισμένων καταστάσεων του κόσμου, εμφανίζονται και ορισμένες πιο συγκεκριμένες ανάλογα με τις θέσεις των αντικειμένων και τα χαρακτηριστικά της πίστας. Ως αποτέλεσμα ο πράκτορας μαθαίνει τις βασικές λειτουργίες που χρειάζεται για να μπορεί να κερδίσει ένα παιχνίδι, αλλά τελειοποιείται μόνο για το συγκεκριμένο περιβάλλον.

Μία σημαντική παράμετρος που αποδείχθηκε πως πρέπει να λαμβάνεται υπόψη κατά την κωδικοποίηση καταστάσεων σε ένα ηλεκτρονικό παιχνίδι είναι η ομαδοποίηση των *σημείων* (blocks) του κόσμου. Όπως φάνηκε από τα αποτελέσματα των διάφορων υλοποιήσεων που πραγματοποιήθηκαν, υπάρχει μεγάλη εξάρτηση μεταξύ των διαφορετικών σημείων και γι' αυτό θα πρέπει να αντιμετωπίζονται συνδυαστικά και όχι αυτόνομα. Καθώς το ιδανικό σενάριο στο οποίο όλα τα blocks μαζί θα σχημάτιζαν μία κατάσταση είναι αδύνατο να εφαρμοστεί στην πράξη, είναι θεμιτό να δημιουργούνται πολλές μικρότερες ομάδες από blocks, οι οποίες συμβάλλουν στη συνέχεια στην συνολική κατάσταση του κόσμου για μία δεδομένη στιγμή.

Επίσης, ιδιαίτερο ρόλο έχει το πλήθος των σημείων που εξετάζονται. Αφενός όσο περισσότερα κωδικοποιούνται, τόσο περισσότερη πληροφορία παρέχεται στον πράκτορα και τον καθιστά δυνητικά αποτελεσματικότερο, αφετέρου η αύξηση του πλήθους των παραμέτρων που πρέπει να προσδιοριστούν από τον γενετικό αλγόριθμο απαιτεί περισσότερο χρόνο εκτέλεσης και εμπεριέχει τον κίνδυνο να μην εξερευνηθούν ποτέ όλοι οι πιθανοί συνδυασμοί.

Συνεπώς η επιλογή του παραθύρου του κόσμου που λαμβάνεται υπόψιν όπως και η επιλογή του τρόπου ομαδοποίησης αποτελούν δύο καθοριστικούς παράγοντες για την ανάπτυξη επιτυχημένων πρακτόρων και απαιτούν ιδιαίτερη μελέτη και αρκετές δοκιμές.

Άλλη μία προσέγγιση που φάνηκε ιδιαίτερα χρήσιμη σύμφωνα με τα αποτελέσματα των πρακτό-

ρων είναι η συμμετρική αντιμετώπιση των καταστάσεων του χώρου. Καθώς ο κόσμος του παιχνιδιού περιορίζεται σε δύο διαστάσεις, η χρήση συμμετρίας στις τέσσερις κατευθύνσεις μειώνει εντυπωσιακά τον αριθμό των παραμέτρων που καλείται να προσδιορίσει ο γενετικός αλγόριθμος και κατά συνέπεια το πλήθος των γενιών και το συνολικό χρόνο. Επιπλέον καθιστά πιο ομαλή τη συμπεριφορά του πράκτορα, καθώς αντιμετωπίζει τις αντίστοιχες καταστάσεις με την ίδια λογική για κάθε κατεύθυνση.

Τελικά, με την εφαρμογή ΓΑ και την κατάλληλη επιλογή των παραμέτρων που περιγράφηκαν παραπάνω, αναπτύχθηκαν πράκτορες που βασισμένοι στην κωδικοποίηση των καταστάσεων είναι πιο αποτελεσματικοί από έναν πράκτορα τυχαίων κινήσεων και από έναν πράκτορα ικανό να βλέπει ένα βήμα μπροστά, όπως ένας αρχάριος παίχτης-άνθρωπος. Τα ποσοστά επιτυχίας τους δείχνουν ότι η συγκεκριμένη τεχνική δέχεται βελτιώσεις, όμως μπορεί να εφαρμοστεί χωρίς πρόβλημα σε περιβάλλοντα με μικρότερες απαιτήσεις, όπως για παράδειγμα για τον έλεγχο μέτριας δυναμικότητας εχθρών από τον υπολογιστή.

Όσον αφορά το περιβάλλον στο οποίο εκτελέστηκε η πειραματική διαδικασία, επιβεβαιώθηκε η δυνατότητα προσαρμογής των γενετικών αλγορίθμων σε ένα δεδομένο πρόβλημα, χωρίς την ανάγκη ιδιαίτερης τροποποίησης της υπάρχουσας πλατφόρμας. Συγκεκριμένα για την εφαρμογή τους στο framework του GVG-AI, αφού εγκαταστάθηκε η κατάλληλη βιβλιοθήκη για την υλοποίηση ΓΑ σε Java, το μόνο τμήμα που ήταν απαραίτητο να ενσωματωθεί στο framework ήταν η συνάρτηση ποιότητας. Εφόσον καταστεί δυνατή η εισαγωγή δεδομένων από το υπάρχον περιβάλλον στη συνάρτηση ποιότητας του ΓΑ, οι δύο βιβλιοθήκες μπορούν να λειτουργούν ανεξάρτητα χωρίς επιπλέον αλλαγές.

6.2 Μελλοντικές Κατευθύνσεις

Όπως ήδη αναφέρθηκε, η τεχνική που χρησιμοποιήθηκε παρουσιάζει κάποια μειονεκτήματα ως προς τη γενίκευση του πράκτορα σε περισσότερα επίπεδα. Η αιτία είναι ότι ορισμένες καταστάσεις του κόσμου εμφανίζονται μόνο σε συγκεκριμένα επίπεδα και δεν μπορούν να κωδικοποιηθούν κατά τη διάρκεια της εκτέλεσης του γενετικού αλγορίθμου σε μία δεδομένη πίστα. Μία ιδέα που χρήζει επιπλέον διερεύνησης και δεν εξετάστηκε στα πλαίσια αυτής της διπλωματικής είναι η εφαρμογή του γενετικού αλγορίθμου σε πολλά διαφορετικά επίπεδα και στη συνέχεια ο σχηματισμός της καλύτερης λύσης συνδυαστικά από τα επιμέρους βέλτιστα χρωμοσώματα. Σε θεωρητικό πλαίσιο, τα χρωμοσώματα τα οποία κωδικοποιούν τις βασικές καταστάσεις που συναντώνται περισσότερο στον κόσμο του παιχνιδιού θα πρέπει να είναι ίδια ή παρόμοια για όλα τα επίπεδα και οι διαφορές θα εμφανίζονται μόνο σε όσα κωδικοποιούν πιο ιδιαίτερες καταστάσεις. Ουσιαστικά σε κάθε επίπεδο θα υπάρχουν χρωμοσώματα που δεν εξερευνώνται ποτέ από το γενετικό αλγόριθμο καθώς δεν εμφανίζονται στο παιχνίδι οι αντίστοιχες συνθήκες. Επομένως σε μία τέτοια προσέγγιση, το μεγαλύτερο μέρος του γενότυπου θα παραμένει σταθερό και θα αλλάζουν μόνο ορισμένα γονίδια τα οποία επιλέγονται κάθε φορά από την αντίστοιχη λύση.

Επίσης ιδιαίτερο ενδιαφέρον παρουσιάζει η προοπτική επιλογής διαφορετικών τρόπων ομαδοποίησης των εξεταζόμενων θέσεων κάθε επιπέδου. Σε αυτή την εργασία σχηματίστηκαν συγκεκριμένες ομάδες των τριών που κρίθηκε ότι περιέχουν την πιο χρήσιμη πληροφορία, ωστόσο δεν μπορεί να ειπωθεί με βεβαιότητα ότι αυτή είναι η βέλτιστη δυνατή επιλογή, δεδομένου του τεράστιου πλήθους των πιθανών συνδυασμών το οποίο εξαρτάται τόσο από τον αριθμό των θέσεων όσο και από τη διάταξή τους στον κόσμο του παιχνιδιού.

Μία άλλη πολύ ενδιαφέρουσα πρόκληση αποτελεί η δοκιμή της συγκεκριμένης μεθόδου σε έναν πράκτορα γενικού σκοπού ο οποίος θα μπορεί να εφαρμοστεί σε περισσότερα από ένα διαφορετικά παιχνίδια. Το framework του διαγωνισμού GVG-AI το οποίο χρησιμοποιήθηκε για την εφαρμογή των πρακτόρων που αναπτύχθηκαν στην παρούσα εργασία στο παιχνίδι Zelda, προσφέρει πολλά επιπλέον παιχνίδια ακριβώς για αυτό το σκοπό.

Εκ πρώτης όψεως φαίνεται ότι για να είναι δυνατή μία τέτοια προσέγγιση, είναι βασική προϋπόθεση τα παιχνίδια να είναι του ίδιου τύπου (στη συγκεκριμένη περίπτωση δοκιμάστηκε το Zelda που είναι τύπου platform) ώστε να έχουν παρόμοιους στόχους και καταστάσεις που να μπορούν να γενι-

κευθούν και να χρησιμοποιηθούν με την ίδια λογική. Μία τέτοια υλοποίηση πράκτορα θα ήταν πιθανόν ανέφικτη, για παράδειγμα, για ένα παιχνίδι τύπου racing και ένα τύπου shooting, καθώς απαιτούν τελείως διαφορετική κωδικοποίηση για την ανάπτυξη της επιθυμητής συμπεριφοράς. Στο GVG-AI framework είναι διαθέσιμο ένα μεγάλο πλήθος παιχνιδιών από τις περισσότερο δημοφιλείς κατηγορίες, συνεπώς αποτελεί ένα ιδιαίτερα προσιτό περιβάλλον τόσο για δοκιμές σε παιχνίδια ίδιου τύπου όσο ενδεχομένως και για απόπειρες σε παιχνίδια διαφορετικού χαρακτήρα.

Τέλος, ιδιαίτερα ελκυστική φαίνεται η κατεύθυνση της επέκτασης της τεχνικής που παρουσιάστηκε και σε τρισδιάστατα (3D) παιχνίδια. Φυσικά, ο κόσμος των τρισδιάστατων παιχνιδιών είναι αρκετά πιο πολύπλοκος και οι πιθανές καταστάσεις εξαρτώνται από περισσότερους παράγοντες. Επίσης καθώς το μέγεθος του κόσμου αυξάνεται σημαντικά σε αυτή την περίπτωση, είναι ίσως προτιμότερο η κωδικοποίηση των καταστάσεων να μην βασίζεται πλέον στις θέσεις γύρω από το avatar αλλά σε κάποια παράμετρο με μικρότερο αριθμό πιθανών τιμών όπως για παράδειγμα κάποια *γεγονότα* (events) του παιχνιδιού. Παρόλα αυτά με την κατάλληλη κωδικοποίηση και ορισμένες τροποποιήσεις, ενδεχομένως να μπορούν να εφαρμοστούν οι πράκτορες που υλοποιήθηκαν στο πλαίσιο αυτής της εργασίας και σε ένα τρισδιάστατο παιχνίδι κατά τον ίδιο τρόπο που συνέβη στο δισδιάστατο παιχνίδι Zelda.

Βιβλιογραφία

[Alab12] Firas Alabsi and Reyadh Naoum, "Comparison of selection methods and crossover operations using steady state genetic based intrusion detection system", *Journal of Emerging Trends in Computing and Information Sciences*, vol. 3, no. 7, pp. 1053–1058, 2012.

[AlHa09] M. T. Al-Hajri and M. A. Abido, "Assessment of Genetic Algorithm selection, crossover and mutation techniques in reactive power optimization", in *2009 IEEE Congress on Evolutionary Computation*, pp. 1005–1011, May 2009.

[Blic96] Tobias Blickle and Lothar Thiele, "A Comparison of Selection Schemes Used in Evolutionary Algorithms", *Evol. Comput.*, vol. 4, no. 4, pp. 361–394, December 1996.

[Bour14] David Bourg and Glenn Seemann, *AI for Game Developers Creating Intelligent Behavior in Games*, O'Reilly Media, 2014.

[BUKH14] Syed Basit Ali BUKHARI, Aftab AHMAD, Syed Auon RAZA and Muhammad Noman SIDDIQUE, "A ring crossover genetic algorithm for the unit commitment problem", *Turkish Journal of Electrical Engineering & Computer Sciences*, 2014.

[Bull] T Bullen and M Katchabaw, "USING GENETIC ALGORITHMS TO EVOLVE CHARACTER BEHAVIOURS IN MODERN VIDEO GAMES".

[Carr14] Jenna Carr, "An introduction to genetic algorithms", *Senior Project*, pp. 1–40, 2014.

[DaRo14] Claudio Comis Da Ronco and Ernesto Benini, *A Simplex-Crossover-Based Multi-Objective Evolutionary Algorithm*, pp. 583–598, Springer Netherlands, Dordrecht, 2014.

[Fran97] Stan Franklin and Art Graesser, *Is It an agent, or just a program?: A taxonomy for autonomous agents*, pp. 21–35, Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.

[Geis07] Sylvie Geisendorf, "Are Genetic Algorithms a good basis for economic learning models?", 01 2007.

[Han12] Jiawei Han, Micheline Kamber and Jian Pei, *Data Mining (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann, Boston, third edition edition, 2012.

[Jain01] Brijnesh J. Jain, Hartmut Pohlheim and Joachim Wegener, "On Termination Criteria of Evolutionary Algorithms", in *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, GECCO'01, pp. 768–768, San Francisco, CA, USA, 2001, Morgan Kaufmann Publishers Inc.

[Jeba13] Khalid Jebari and Mohammed Madiafi, "Selection methods for genetic algorithms", *International Journal of Emerging Sciences*, vol. 3, no. 4, pp. 333–344, 2013.

[jene] "Jenetics: Java Genetic Algorithm Library".

[Jenn98] N. R. Jennings and M. Wooldridge, "Agent Technology", chapter Applications of Intelligent Agents, pp. 3–28, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[Kram17] Oliver Kramer, *Genetic Algorithm Essentials*, Springer International Publishing, 2017.

[Kuma16] Anand Kumar, *Network Design using Genetic Algorithm*, chapter Genetic Algorithm, p. 256, LAP LAMBERT Academic Publishing, 2016.

[Levi] Michael Levin, "Application of Genetic Algorithms to Molecular Biology: Locating Putative Protein Signal Sequences".

[Lieb16] Diego Perez Liebana, Spyridon Samothrakis, Julian Togelius, Tom Schaul and Simon M. Lucas, "General Video Game AI: Competition, Challenges and Opportunities.", in Dale Schuurmans and Michael P. Wellman, editors, *AAAI*, pp. 4335–4337, AAAI Press, 2016.

[McCo04] Pamela McCorduck, *Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence*, AK Peters Ltd, 2004.

[Mess02] Chris Messom, "Genetic Algorithms for Auto-tuning Mobile Robot Motion Control", 2002.

[Mill] Frederick Mills and Robert Stufflebeam, "Introduction to Intelligent Agents".

[Mill09] Ian Millington and John Funge, *Artificial Intelligence for Games, Second Edition*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2009.

[Mitc98] Melanie Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA, USA, 1998.

[Neal02] John L Nealon and Antonio Moreno, "The Application of Agent Technology to Health Care", in *Proceedings of the AgentCities workshop: Research in large scale open agents environments, at the first international joint conference on autonomous agents and multi-agent systems (AMAS)*, pp. 169–173, Bolongna (Italy), July 2002, ACM Pres.

[Pere16] D. Perez-Liebana, S. Samothrakis, J. Togelius, T. Schaul, S. M. Lucas, A. Couëtoux, J. Lee, C. U. Lim and T. Thompson, "The 2014 General Video Game Playing Competition", *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 8, no. 3, pp. 229–243, Sept 2016.

[Pivk00] Aleksander Pivk and Matjaž Gams, "E-commerce intelligent agents", *Proceedings of ICTEC'00*, vol. 67, no. 5, pp. 251–260, 2000.

[Potv96] Jean-Yves Potvin, "Genetic algorithms for the traveling salesman problem", *Annals of Operations Research*, vol. 63, no. 3, pp. 337–370, Jun 1996.

[Russ95] Stuart J. Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.

[Russ98] Stuart Russell, "Learning Agents for Uncertain Environments (Extended Abstract)", in *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pp. 101–103, New York, NY, USA, 1998, ACM.

[Schw81] H.P. Schwefel, *Numerical Optimization of Computer Models*, Interdisciplinary systems research, John Wiley and Sons, 1981.

[Scot02] Bob Scott, *AI Game Programming Wisdom*, chapter Chapter 1.2: The illussion of intelligence, pp. 16–20, Charles River Media, 2002.

[Soni14] Nitasha Soni and Tapas Dr. Kumar, "A ring crossover genetic algorithm for the unit commitment problem", *International Journal of Computer Science and Information Technologies*, vol. 5, 2014.

[TURI50] A. M. TURING, "I.—COMPUTING MACHINERY AND INTELLIGENCE", *Mind*, vol. LIX, no. 236, pp. 433–460, 1950.

[Umba15] AJ Umbarkar and PD Sheth, "CROSSOVER OPERATORS IN GENETIC ALGORITHMS: A REVIEW.", *ICTACT journal on soft computing*, vol. 6, no. 1, 2015.

[Yann12] Geogios N. Yannakakis, "Game AI Revisited", in *Proceedings of the 9th Conference on Computing Frontiers*, CF '12, pp. 285–292, New York, NY, USA, 2012, ACM.

[Γε99] Ευστράτιος Φ. Γεωργόπουλος και Σπυρίδων Δ. Λυκοθανάσης, *Εισαγωγή στους Γενετικούς Αλγορίθμους*, Πανεπιστήμιο Πατρών, 1999.