Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Εφαρμοσμενων Μαθηματικων και Φυσικων Επιστημων

Αναδρομή και εξέλιξη συστημάτων τεχνητής νοημοσύνης

Νικήτας Σταθάτος

Επιβλέπων: Αλέξανδρος Αρβανιτάκης
Μέλος επιτροπής: Αντώνιος Χαραλαμπόπουλος
Μέλος επιτροπής: Πέτρος Στεφανέας

Περίληψη

Ένας αλγόριθμος που μπορεί να επεξεργάζεται τον κώδικά του ονομάζεται αυτοαναφορικός. Σε αυτήν τη εργασία, εξετάζουμε τη συμπεριφορά αυτοαναφορικών αλγορίθμων όταν το περιβάλλον τους επιβάλλει κάποιες συνθήκες επιλογής. Εστιάζουμε στη διαδικασία της διαγωνοποίησης, με τη χρήση της οποίας αυτοί οι αλγόριθμοι εκμεταλλεύονται την αυτοαναφορά για να εξελιχθούν με έναν απλό τρόπο. Δίνουμε παραδείγματα και κάποιες γενικεύσεις αυτής της διαδικασίας, οπού φαίνεται η αλληλεπίδραση της διαγωνοποίησης και της αυτοαναφοράς. Επίσης, συγκρίνουμε αυτή τη συμπεριφορά με άλλες τεχνικές προγραμματισμού ή υπολογισμού που αξιοποιούν κάποια μορφή επιλογής και αυτοαναφοράς.

Abstract

We call an algorithm that can edit itself self-editing or self-referential. In this thesis, we examine the behavior of self-editing algorithms in the presence of selective pressure from the environment. We focus on the process of diagonalization, through which these algorithms can take advantage of self reference to evolve in a simple and elegant way. We give examples and generalizations of this process, which help make the power of self reference clear. We also compare this behavior with other programming paradigms and computation models that make use of selection and self reference.

Πρόλογος

Η εργασία αυτή μελετάει ορισμένες ιδιότητες των αυτοαναφορικών αλγορίθμων και το πως σε τι συμπεριφορά αυτές οδηγούν υπό επιλεκτική πίεση. Στο πρώτο κεφάλαιο δίνουμε ορισμούς και παραδείγματα για δύο χρήσιμες έννοιες, τους αλγορίθμους και τους κώδικες, που αποτελούν θεμελιώδη αντικείμενα του πεδίου. Έπειτα δίνουμε κάποια παραδείγματα των βασικών διεργασιών που μπορεί να πραγματοποιήσει ένας κώδικας στον εαυτό του, κάτι που τον καθιστά αυτοαναφορικό.

Το τρίτο κεφάλαιο ασχολείται με τη διαδικασία της διαγωνοποίησης, όπως αυτή αναλύεται στο [1]. Η διαδικασία αυτή είναι που μας επιτρέπει να εκμεταλλευτούμε οποίους περιορισμούς επιβάλλει το περιβάλλον και να συνδυάσουμε τις έννοιες της εξέλιξης και της αυτοαναφοράς. Καθορίζουμε τον τρόπο με τον οποίο οι κώδικές μας θα αλληλεπιδρούν με το περιβάλλον και το πως μπορούν να πολλαπλασιάζονται, ώστε μόνο κάποιο από αυτούς να επιλέγονται για επιβίωση. Στη συνέχεια σχολιάζουμε ορισμένες πιο σύνθετες γενικεύσεις ενός συστήματος που χρησιμοποιεί διαγωνοποίηση.

Στο τελευταίο κεφάλαιο παρουσιάζουμε κάποιες άλλες υπολογιστικές μεθόδους που έχουν κάνει χρήση συναφών εννοιών και τις συγκρίνουμε με τη διαγωνοποίηση και τα αποτελέσματά της. Εστιάζουμε σε αυτοαναφορικές ιδιότητες που εμφανίζονται και στη βιολογία και στη πληροφορική, και το πως αυτά τα δύο πεδία έχουν αλληλεπιδράσει. Η σύγκριση αυτή βοηθάει να προσδιορίσουμε καλύτερα τα πλεονεκτήματα και μειονεκτήματα της διαγωνοποίησης.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Αλγόριθμοι

Οι αλγόριθμοι αποτελούσαν για αιώνες μία χρήσιμη αλλά διαισθητική έννοια. Γενικά αλγόριθμο αποκαλούμε οποιοδήποτε υπολογισμό πραγματοποιείται μέσω μίας πεπερασμένης αλληλουχίας βημάτων, με στόχο την επίλυση κάποιου προβλήματος. Αν και αυτός ο ορισμός αρχικά εξυπηρετούσε τις καθημερινές και μαθηματικές ανάγκες του όρου, στον 20ο αιώνα υπήρξε μεγάλο ενδιαφέρον για τον αυστηρό φορμαλισμό του. Αρχικός σκοπός της σχετικής έρευνας ήταν να απαντηθεί το πρόβλημα απόφασης (Entscheidungsproblem) του Hilbert, αλλά οι μετέπειτα ιδέες επηρέασαν άμεσα τον αναπτυσσόμενο κλάδο της Επιστήμης Υπολογιστών, ειδικά με την ραγδαία εξάπλωση των ηλεκτρονικών υπολογιστών και του προγραμματισμού.

Η αυστηρή τυποποίηση της έννοιας ξεκίνησε τη δεκαετία του 1930 από τους Gödel, Church, Turing και άλλους. Ένα από τα πιο σημαντικά αποτελέσματα αυτής της προσπάθειας είναι η κρίσιμη παρατήρηση ότι οι αλγόριθμοι μπορούν να ταυτιστούν με τις υπολογίσιμες συναρτήσεις, δηλαδή με τις συναρτήσεις οι τιμές των οποίων μπορούν να υπολογιστούν με μηχανιστικό τρόπο, για παράδειγμα από μία μηχανή Turing. Η γνωστή θέση των Church-Turing, την οποία δεχόμαστε ως αξίωμα, δηλώνει ότι όλες οι υπολογίσιμες συναρτήσεις μπορούν να εκφραστούν σε κάθε αρκούντως ισχυρό υπολογιστικό μοντέλο, και κανένα από αυτά τα μοντέλα δεν μπορεί να υπολογίσει κάτι παραπάνω. Μία συμβατική επιλογή μοντέλου αποτελεί το σύνολο των αναδρομικών συναρτήσεων, όπως αυτό ορίστηκε από τους Gödel, Herbrand, Kleene [5]. Σύμφωνα με τα παραπάνω, είμαστε σε θέση να χαρακτηρίσουμε έναν αλγόριθμο ή γενικότερα μία

διαδικασία ως υπολογίσιμη αν αυτή μπορεί να υλοποιηθεί με κάποιο μηχανιστικό τρόπο, για παράδειγμα σε κάποια προγραμματιστική γλώσσα.

Συχνά είναι χρήσιμο να μελετήσουμε αλγορίθμους με είσοδο δύο ή περισσότερα στοιχεία, σε αντιστοιχία με συναρτήσεις πολλών μεταβλητών. Όμως όλες αυτές οι περιπτώσεις καλύπτονται από τους αλγορίθμους που δέχονται ως είσοδο ένα string (συμβολοσειρά), από ένα κατάλληλο αλφάβητο. Αυτό προκύπτει από το γεγονός ότι υπάρχει (υπολογίσιμη) αντιστοίχηση μεταξύ του συνόλου των λέξεων ενός αριθμήσιμου αλφαβήτου και του συνόλου όλων των δυάδων λέξεων. Γενικά θα θεωρούμε ότι οι αλγόριθμοί μας έχουν ως είσοδο ένα στοιχείο που μεταφέρει όλη τη πληροφορία που χρειάζονται, εκτός και αν μας διευκολύνει μία διαφορετική αναπαράσταση. Αυτό μας δίνει το πλεονέκτημα ότι μπορούμε να μιλάμε για μία ενοποιημένη κατηγορία αλγορίθμων, ακόμη και αν αυτοί διαφέρουν στον τρόπο που διαχειρίζονται την είσοδο και την έξοδο τους.

1.2 Κώδικες

Αν και κάθε αλγόριθμος αντιστοιχεί σε μία προκαθορισμένη διαδικασία, συνήθως υπάρχουν πολλοί τρόποι με τους οποίους μπορεί να πραγματοποιηθεί αυτή η διαδικασία. Ο ακριβής τρόπος με τον οποίο υλοποιούμε τους αλγορίθμους καθορίζεται από το λεγόμενο υπολογιστικό σύστημα ή μοντέλο στο οποίο δουλεύουμε. Για να διακρίνουμε μεταξύ διαφορετικών υλοποιήσεων μέσα στο ίδιο σύστημα, χρησιμοποιούμε κωδικοποιήσεις.

Οι πιο συνηθισμένες κωδικοποιήσεις μετατρέπουν αλγορίθμους σε αριθμούς. Το πιο χαρακτηριστικό παράδειγμα τέτοιας κωδικοποίησης αποτελεί η αρίθμηση Gödel, την οποία χρησιμοποίησε ο Kurt Gödel για να αντιστοιχήσει σε κάθε λογική έκφραση έναν φυσικό αριθμό [3]. Ως ειδική περίπτωση αυτής της αντιστοίχησης, κάθε μ-αναδρομική συνάρτηση έχει ως κώδικα έναν αριθμό, τον λεγόμενο αριθμό Gödel τής. Μπορούμε να χειριστούμε αυτούς τους αριθμούς μέσω των γνωστών πράξεων, κάτι που μας προϊδεάζει για την επεξεργασία κωδίκων.

Στο εξής, ως κώδικα εννοούμε ένα (πεπερασμένο) string κάποιου αλφαβήτου. Επιπλέον υποθέτουμε ότι οι κώδικες μας είναι δομημένοι, δηλαδή αποτελούνται από τμήματα. Ένας αλγόριθμος μπορεί να έχει πρόσβαση σε αυτά με αναφορά στο όνομα του αντίστοιχου τμήματος, που αποκαλούμε διεύθυνση. Με $c \restriction \theta$ θα συμβολίζουμε το τμήμα του κώδικα c με διεύθυνση θ.

Σημειώνουμε ότι για κάθε σύστημα υπάρχουν πολλές κωδικοποιήσεις. Επιπλέον, κάθε αλγόριθμος μπορεί να έχει πολλούς διαφορετικούς πιθανούς κώδικες, εντός του ίδιου συστήματος και κωδικοποίησης. Οπότε, όταν μιλάμε για τον κώδικα c ενός αλγορίθμου C, εννοούμε απλώς ότι για τον αλγόριθμο που παίρνουμε ερμηνεύοντας τον c στο κατάλληλο υπολογιστικό σύστημα, ισχύει $alg(c) = C$. Είναι σημαντικό αυτή η ερμηνεία από κώδικα σε αλγόριθμο να γίνεται με μονοσήμαντο τρόπο, δηλαδή από κάθε (συντακτικά σωστό) κώδικα να εξάγεται ένας μοναδικός αλγόριθμος μέσω μίας συστηματικής διαδικασίας. Για να γίνει πιο κατανοητή η έννοια, παρουσιάζουμε κάποια κλασικά παραδείγματα κωδίκων.

Υπολογιστικό Σύστημα	Μορφή Κώδικα
μ-αναδρομικές συναρτήσεις	αρίθμηση Gödel
μηχανές Turing	πίνακας μεταβάσεων
λ-λογισμός	κωδικοποίηση Church
προγραμματιστικές γλώσσες	πηγαίος κώδικας

Πίνακας 1.1: Παραδείγματα υπολογιστικών συστημάτων και των αντίστοιχων κωδικοποιήσεων.

Κεφάλαιο 2

Αυτοαναφορά

2.1 Επεξεργασία κωδίκων

Κεντρικό αντικείμενο μελέτης αυτής της εργασίας είναι οι αυτοαναφορικοί αλγόριθμοι, δηλαδή οι αλγόριθμοι που μπορούν να επεξεργάζονται τον κώδικα τους. Για να μπορέσουμε να κατανοήσουμε τη λειτουργία τους, θα δώσουμε πρώτα κάποια χρήσιμα παραδείγματα αλγορίθμων οι οποίοι έχουν ως είσοδο και έξοδο κώδικες.

- Ένας τέτοιος αλγόριθμος μπορεί να διαβάσει, να εκτελέσει και να επεξεργαστεί την είσοδο του ή συγκεκριμένα τμήματα αυτής.
- Μπορεί να προσθέσει στην είσοδο ένα νέο κομμάτι κώδικα, με μία νέα διεύθυνση.
- Μπορεί να αντιγράψει την είσοδό του.
- Μπορεί να υπολογίσει παραλλαγές της εισόδου, δηλαδή να επιστρέψει το σύνολο $\{alg(e_i)(c)\}$, για διάφορους κώδικες e_i.
- Μπορεί να αποθηκεύσει σε μνήμη κάποια πληροφορία (data).

Είναι εύκολο να δούμε ότι τα παραπάνω παραδείγματα αντιστοιχούν σε υπολογίσιμες συναρτήσεις αφού μπορούμε, για παράδειγμα, να τα υλοποιήσουμε σε προγραμματιστική γλώσσα της επιλογής μας. Από εδώ και στο εξής, θα θεωρούμε ότι πράγματι οι κώδικες μας βρίσκονται σε ένα υπολογιστικό σύστημα με αρκετή περιγραφική ισχύ ώστε να μπορεί να εκφράσει όλες τις υπολογίσιμες συναρτήσεις.

2.2 Αυτο-επεξεργασία

Έστω c κάποιος αυτοαναφορικός κώδικας, δηλαδή ένας κώδικας που μπορεί να δεχτεί ως είσοδο, να επεξεργαστεί και να εκτελέσει τον εαυτό του ή τμήματά του. Έστω επίσης A κάποιος αλγόριθμος που επεξεργάζεται κώδικες, όπως στα προηγούμενα παραδείγματα. Αν ο c περιλαμβάνει μία κωδικοποίηση του A σε κάποια διεύθυνση θ_a, τότε μπορεί να πραγματοποιήσει διάφορες διεργασίες, εκτελώντας το τμήμα που βρίσκεται στη θ_a με είσοδο το c.

- Ο c μπορεί να διαβάσει, να εκτελέσει και να επεξεργαστεί τον εαυτό του ή συγκεκριμένα τμήματά του.
- Ένας τέτοιος κώδικας μπορεί να διαβάσει, να εκτελέσει και να επεξεργαστεί την είσοδο του ή συγκεκριμένα τμήματα αυτής.
- Μπορεί να αντιγράψει τον εαυτό του.
- Μπορεί να υπολογίσει παραλλαγές του εαυτού του, δηλαδή να επιστρέψει το σύνολο $\{alg(e_i)(c)\}$, για διάφορους κώδικες e_i. Παρακάτω θα αποκαλούμε αυτές τις παραλλαγές απογόνους του c και θα συμβολίζουμε ακολουθίες απογόνων με $c_1 \mapsto c_2 \mapsto \dots c_n$.
- Μπορεί να αποθηκεύσει ένα αντίγραφο του εαυτού του.

Η ικανότητα μνήμης, σε συνδυασμό με το προσθήκη νέου κώδικα, έχει ως αποτέλεσμα ότι όποτε ο c υπολογίζει μία παραλλαγή του, μπορεί να του προσθέσει σε κάποια διεύθυνση ένα αντίγραφο του c. Αν μία ακολουθία κωδίκων $c_1 \mapsto c_2 \mapsto \dots c_n$ εκμεταλλευτεί αυτήν την ιδιότητα, τότε ο c_n θα περιέχει αρκετή πληροφορία για να σχηματίσει όλους τους προγόνους του.

Η έννοια των απογόνων είναι ιδιαίτερα χρήσιμη και θα μας απασχολήσει παρακάτω. Οι κώδικες e_i που αντιστοιχούν στους απογόνους είναι και αυτοί σημαντικοι. Έστω w ένας (άμεσος) απόγονος του c, δηλαδή $w = alg(e_j)(c)$ για κάποιον κώδικα e_j από τις παραλλαγές που υπολόγισε ο c. Τότε καλούμε τον e_j αναδρομέα (recursor) του w. Ομοίως, ονομάζουμε τον κώδικα που υπολογίζει το τμήμα στη διεύθυνση θ του w, θ-αναδρομέα του w.

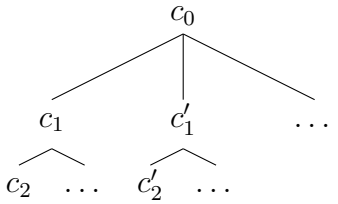
Κεφάλαιο 3

Διαγωνοποίηση

3.1 Πληθυσμοί κωδίκων

Σε αυτό το σημείο θέλουμε να μελετήσουμε πως μπορεί να χρησιμοποιηθεί η αυτοαναφορά για να επιτευχθεί η εξέλιξη αλγορίθμων, οι οποίοι θα βελτιώνονται στην επίλυση ορισμένων προβλημάτων μέσω κάποιας διαδικασίας επιλογής. Για να το κάνουμε αυτό χρειάζεται να μελετήσουμε σύνολα αυτοαναφορικών κωδίκων που ονομάζουμε πληθυσμούς. Συμβολίζουμε με $\{c_1, \ldots, c_n\}$ έναν πληθυσμό κωδίκων με τα στοιχεία $c_1, \ldots c_n$. Επειδή κάποια στοιχεία του πληθυσμού μπορεί να ταυτίζονται, τυπικά μιλάμε για ένα πολυ-σύνολο κωδίκων.

Τα στοιχεία ενός πληθυσμού μπορούν να πολλαπλασιάζονται, μέσω των διαδικασιών που περιγράψαμε στο προηγούμενο κεφάλαιο. Φυσικά, οι απόγονοι του αρχικού πληθυσμού μπορούν και αυτοί να πολλαπλασιάζονται με παρόμοιο τρόπο, όπως μπορεί και κάθε επόμενη γενιά. Έτσι, από κάθε αρχικό κώδικα σχηματίζεται ένα δέντρο $(c_t)_{t\in\{T\}}$ πιθανών απογόνων. Τώρα στους κώδικες μπορεί δράσει κάποια μορφή επιλογής, η οποία επιτρέπει μόνο σε κάποιους από τους απογόνους να επιβιώνουν σε κάθε στάδιο.



Σχήμα 3.1: Δέντρο απογόνων ενός κώδικα

Αν c_{k+1} είναι επιζών απόγονος του c_k, τότε ονομάζουμε την αντίστοιχη ακολουθία

$$c_1, c_2, \ldots, c_n, \ldots \tag{A}$$

επιζούσα. Η σημαντική παρατήρηση σε αυτό το σημείο είναι ότι, εάν οι συνθήκες του περιβάλλοντος που καθορίζουν τον τρόπο με τον οποίο γίνεται η επιλογή ακολουθούν κάποιον προβλέψιμο κανόνα, τότε αναμένουμε να αναδυθεί κάποιο μοτίβο στην Α. Θα θέλαμε ο c_n να μπορεί να χρησιμοποιήσει αυτό το μοτίβο στον υπολογισμό των απογόνων του.

Πιο αναλυτικά, θέλουμε να βρούμε έναν κώδικα r που επεξεργάζεται κώδικες, τέτοιον ώστε:

$$alg(r)(c_i) = c_{i+1}, \quad i = 1, 2, \ldots, n-1, \tag{B}$$

και στη συνέχεια να χρησιμοποιήσουμε τον r για τον υπολογισμό των απογόνων του c_n. Παρακάτω δίνουμε μία απλή εφαρμογή αυτής της ιδέας.

Έστω ότι το περιβάλλον απαιτεί από τους κώδικες μας την εμφάνιση της ακολουθίας $(1, 2, 3, \ldots)$ (για παράδειγμα, στο εσωτερικό κάποιας διεύθυνσης θ_1). Μπορούμε να παρατηρήσουμε ότι η συνάρτηση του επόμενου:

$$S(n) = n + 1,$$

πληρεί τη συνθήκη Β. Επομένως, μπορούμε να χρησιμοποιήσουμε αυτή τη συνάρτηση για τον υπολογισμό των απογόνων w του c_n:

$$w \restriction \theta_1 = S(c_n \restriction \theta_1).$$

3.2 Διαγωνοποίηση

Ο συστηματικός τρόπος εύρεσης των μοτίβων μίας επιζούσας ακολουθίας συνοψίζεται στην εξής διαδικασία:

> Υπολόγισε τον απλούστερο κώδικα r έτσι ώστε $alg(r)(c_i \restriction \theta) = (c_{i+i} \restriction \theta)$ για κάθε $i < n$. Υπολόγισε έτσι τον θ-αναδρομέα για όλους (ή σχεδόν όλους) τους απογόνους w του c_n, δηλαδή $w \restriction \theta = alg(r)(c_i \restriction \theta)$. (Γ)

Παρακάτω θα αποκαλούμε τη διαδικασία Γ *διαγωνοποίηση*, καθώς ο υπολογισμός του κάθε βήματος γίνεται με αυτοαναφορικό τρόπο, κοιτώντας τα προηγούμενα στοιχεία της ακολουθίας. Όπως είδαμε, υπό κατάλληλες συνθήκες, ο c_n περιέχει αρκετή πληροφορία για να ανακατασκευάσει τους προγόνους του. Άρα μπορεί να έχει πρόσβαση στους κώδικες $\{c_i\}_{i=1}^{n-1}$ και να εφαρμόσει τη διαγωνοποίηση σε αυτούς.

Η εύρεση του απλούστερου κώδικα που ικανοποιεί μία (υπολογίσιμη) συνθήκη μπορεί να πραγματοποιηθεί τυπώνοντας μία αρίθμηση όλων των πιθανών κωδίκων $r_1, r_2, \ldots$ (καθώς το σύνολο όλων των κωδίκων ή αλγορίθμων είναι αναδρομικά αριθμήσιμο) και διαλέγοντας το μικρότερο m για το οποίο ο r_m ικανοποιεί τη ζητούμενη συνθήκη. Εφόσον η αρίθμηση γίνεται με φυσικό τρόπο (π.χ. με αναδρομή στους κανόνες που ορίζουν τους επιτρεπτούς κώδικες), είναι αναμενόμενο οι κώδικες που αντιστοιχούν σε πιο περίπλοκους αλγορίθμους να εμφανίζονται αργότερα στην αρίθμηση. Αυτό σε πολλές περιπτώσεις μας εξασφαλίζει ότι η διαδικασία που εμείς θα θεωρούσαμε διαισθητικά 'απλούστερη' θα εμφανίζεται νωρίτερα στην αρίθμηση.

Έως τώρα, η διαγωνοποίηση φαίνεται να είναι ένας εξωτερικός περιορισμός που επιβάλλουμε στον τρόπο με τον οποίο θα πολλαπλασιάζονται οι κώδικες c_i. Μπορούμε όμως, με τον ίδιο τρόπο που περιγράψαμε στην ενότητα 2.2, να την κάνουμε εσωτερική ιδιότητα των κωδίκων. Αν d είναι κώδικας για τη διαδικασία Γ, τότε ένας αυτοαναφορικός κώδικας c που περιέχει τον d σε κάποιο τμήμα του, μπορεί να εκτελέσει τη Γ στον εαυτό του, με την προϋπόθεση ότι έχει τη δυνατότητα να αποθηκεύσει και να διαβάσει το περιεχόμενο των προγόνων του από κάποια μνήμη. Παρακάτω θα μελετήσουμε τις επιπτώσεις της διαγωνοποίησης ως περιεχόμενο αυτοαναφορικών αλγορίθμων.

3.3 Παραδείγματα

Σε αυτήν την ενότητα σκοπός μας είναι να εξετάσουμε την ισχύ της διαγωνοποίησης, ιδιαίτερα όταν αυτή αποτελεί εσωτερικό τμήμα των κωδικών ενός πληθυσμού. Τα παραδείγματα ή πειράματα που θα χρησιμοποιήσουμε έχουν τη μορφή μίας ενδεχομένως άπειρης ακολουθίας αριθμών της μορφής $(a_1, a_2, a_3, \ldots)$, την οποία θέλουμε να συμπληρώσει το σύστημα χρησιμοποιώντας διαγωνιοποίηση. Η ιδέα είναι όμοια με τα τεστ IQ που έχουν χρησιμοποιηθεί ως δοκιμές αναφοράς για την εξέταση της ανθρώπινης αλλά και της τεχνητής νοημοσύνης [6].

Επειδή προς το παρόν θεωρούμε κώδικες που έχουν ως είσοδο και έξοδο αποκλειστικά άλλους κώδικες, δεν θα χρησιμοποιήσουμε τους πρώτους όρους a_1, a_2, a_3 ως είσοδο στο σύστημα. Για απλότητα, θα υποθέσουμε ότι κάποιοι απόγονοι του αρχικού πληθυσμού μαντεύουν (στην τύχη) τους πρώτους όρους της ακολουθίας. Η αλληλεπίδραση με το περιβάλλον γίνεται μέσω μίας συγκεκριμένης διεύθυνσης, που συμβολίζουμε με θ_1. Η επιλογή και άρα η επιβίωση των απογόνων βασίζεται στο αν το περιεχόμενο της διεύθυνσης ταιριάζει με τους όρους της ζητούμενης ακολουθίας. Πιο συγκεκριμένα, θεωρούμε ότι το σύστημα περνάει επιτυχώς τη δοκιμή αν η πλειοψηφία των απογόνων έχει αποθηκεύσει τη σωστή τιμή στην κατάλληλη διεύθυνση.

Στην περίπτωση της ακολουθίας $(1, 2, 3, \dots)$, το σύστημα μπορεί να μαντέψει σωστά την υπόλοιπη ακολουθία, αρκεί η συνάρτηση του επόμενου να είναι αντιστοιχεί στον πιο απλό κώδικα που βρίσκει η Γ. Όπως εξηγήσαμε, αυτή είναι μία σχετικά φυσική υπόθεση, που όμως εξαρτάται από τον ακριβή τρόπο με τον οποίο η Γ αριθμεί το σύνολο όλων των κωδίκων.

Η χρήση της διαγωνοποίησης φυσικά θα αποτύγχανε αν η πραγματική ακολουθία που έπρεπε να βρει το σύστημα ήταν η $(1, 2, 3, 1, 1, \dots)$, ή οποιαδήποτε άλλη ξεκινούσε με τα ίδια πρώτα στοιχεία. Ένας άνθρωπος που επιχειρούσε να μαντέψει τα επόμενα στοιχεία της ίδιας ακολουθίας θα αντιμετώπιζε το ίδιο πρόβλημα, καθώς το αρχικό, πεπερασμένο τμήμα ενός τέτοιου παραδείγματος δεν μπορεί να περιέχει αρκετή πληροφορία για να περιγράψει όλα τα επόμενα στοιχεία. Το σύστημα μας έχει την ικανότητα να προσαρμοστεί σε τέτοιες περιπτώσεις αν έχει πρόσβαση σε μεγαλύτερα αρχικά τμήματα, δηλαδή χρησιμοποιώντας μεγαλύτερες τιμές του n που εμφανίζεται στη Γ, το οποίο επίσης είναι αποθηκευμένο σε κάποιο τμήμα της μνήμης των κωδίκων.

Μπορούμε να παρατηρήσουμε μία πιο περίπλοκη και ενδιαφέρουσα συμπεριφορά αν ζητήσουμε από το σύστημα να χρησιμοποιήσει τη διαγωνοποίηση σε παραδείγματα της μορφής:

$$(1, 2, 3, \dots), (1, 3, 5, \dots), (1, 4, 7, \dots) \dots$$

Αναμένουμε ότι οι αλγόριθμοι που προσθέτουν 1, 2, 3 κλπ. στην είσοδό τους περιγράφονται από αρκετά απλούς κώδικες, οπότε το σύστημα θα μπορούσε να μαντέψει κάθε μία από τις ακολουθίες σε απομόνωση. Επίσης, αναλόγως το κριτήριο τερματισμού κάθε ακολουθίας, οι κώδικες μπορούν να προσαρμοστούν και σε αυτό και να λύσουν επιτυχώς ολόκληρο το παράδειγμα (δηλαδή όλες τις ακολουθίες μαζί). Η αυτοαναφορική όμως φύση της διαδικασίας δίνει στο σύστημα επιπλέον ιδιότητες.

Καθώς έχουμε εσωτερική χρήση της διαγωνοποίησης, οι κώδικες έχουν υπολογίσει τον αναδρομέα της διεύθυνσης θ_1 και τον έχουν αποθηκεύσει σε κάποιο τμήμα τους, έστω το θ_2. Αυτός ο αναδρομέας περιέχει, σε διαφορετικά στάδια του πειράματος, κώδικες για τους αλγορίθμους που προσθέτουν 1, 2, ή 3 στην είσοδό τους. Επομένως οι αριθμοί 1, 2, 3 βρίσκονται σε μία περαιτέρω υπο-διεύθυνση του θ_2. Γνωρίζουμε ήδη ότι ένα σύστημα που χρησιμοποιεί διαγωνοποίηση μπορεί να μαντέψει επιτυχώς τους επόμενους όρους αυτής της ακολουθίας. Άρα, εκτελώντας την διαγωνοποίηση σε αυτήν την υποδιεύθυνση, το σύστημα μπορεί να κατασκευάσει έναν αναδρομέα, ο οποίος όταν εκτελεστεί θα εφαρμόσει τη συνάρτηση του επόμενου σε αυτούς τους αριθμούς. Αυτό θα έχει ως αποτέλεσμα να μαντέψει επιτυχώς και τις 3 ακολουθίες, καθώς και όλες τις επόμενες που ακολουθούν το ίδιο μοτίβο.

Σε αυτό το στάδιο το σύστημα έχει κάνει χρήση της διαγωνοποίησης για τον υπολογισμό του αναδρομέα ενός αναδρομέα. Βλέπουμε έτσι ότι πέρα από τα μοτίβα μίας επιζούσας ακολουθίας, το σύστημα μπορεί να εντοπίσει και μοτίβα μοτίβων. Η ιδέα φυσικά μπορεί να συνεχιστεί σε ακόμη μεγαλύτερο βάθος, εφόσον το επιτρέπει η διαθέσιμη μνήμη του συστήματος και των επί μέρους κωδίκων. Σημειώνουμε ότι ένας πληθυσμός κωδίκων δεν θα μπορούσε να επιτύχει σε αυτό το πείραμα αν η διαγωνοποίηση δεν βρισκόταν εσωτερικά, σε κάποιο τμήμα των κωδίκων. Σε εκείνη την περίπτωση, δεν θα υπήρχε η διεύθυνση θ_2 μέσα στους ίδιους τους κώδικες, ώστε να εφαρμόσουμε τη διαγωνοποίηση εκεί.

Για να πετύχουμε αυτό το πιο ισχυρό αποτέλεσμα, κάναμε την επιπλέον υπόθεση ότι μπορούμε να εφαρμόσουμε τη διαγωνιοποίηση σε μη διαδοχικούς κώδικες. Στην πραγματικότητα, αν κοιτάξουμε το περιεχόμενο της διεύθυνσης ως προς την οποία διαγωνοποιούμε, αυτό δεν θα είναι $(1, 2, 3, \dots)$, αλλά $(1, 1, \dots, 2, 2, \dots, 3, 3, \dots)$. Αυτή η δυσκολία μπορεί να ξεπεραστεί με διάφορους τρόπους, καθώς και η δεύτερη ακολουθία επίσης ακολουθεί σχετικά απλούς κανόνες. Για παράδειγμα, το σύστημα μπορεί να χρησιμοποιήσει έναν αλγόριθμο που προσθέτει 1 στην είσοδο όποτε συμβαίνει μία μετάβαση στο επόμενο υπο-πείραμα. Αν, όπως αναφέραμε πριν, το κριτήριο τερματισμού κάθε ακολουθίας είναι ανιχνεύσιμο από τους κώδικες (π.χ. με την εμφάνιση παρενθέσεων), τότε αυτός είναι ένας σχετικά απλός αλγόριθμος που μπορούν να εκτελέσουν και που θα επιστρέψει το σωστό αποτέλεσμα.

3.4 Γενικεύσεις

Σε αυτό το σημείο παρουσιάζουμε συνοπτικά κάποιες χρήσιμες γενικεύσεις της διαδικασίας της διαγωνοποίησης. Αυτές παρουσιάζονται πιο αναλυτικά στο [2].

Μία από τις πιο απλές μετατροπές που μπορούμε να κάνουμε είναι η αντικατάσταση της επιβίωσης ή μη ενός κλάδου με μία συνάρτηση ανταμοιβής (reward function). Αυτή η συνάρτηση απονέμει μία συγκεκριμένη αριθμητική τιμή σε κάθε κώδικα του πληθυσμού, που αντιπροσωπεύει την επιτυχία τους στο εκάστοτε περιβάλλον. Με αυτήν την αλλαγή προσφέρουμε μεγαλύτερη ευελιξία στο σύστημα, καθώς οι κώδικες τώρα έχουν την ικανότητα να εξελίσσονται με βάση τους απογόνους που πήραν μικρότερη ή μεγαλύτερη ανταμοιβή. Με λίγα λόγια, η χρήση αυτής της συνάρτησης επιτρέπει στους κώδικες να αντλούν περισσότερη πληροφορία από το περιβάλλον τους. Πρόκειται για μία απλή αλλά χρήσιμη αλλαγή, διότι γενικεύει με ουσιαστικό τρόπο το πλαίσιο της επιβίωσης που χρησιμοποιούσαμε ως τώρα. Επίσης, αυτή η μέθοδος έχει το σημαντικό πλεονέκτημα ότι μπορεί να εφαρμοστεί άμεσα σε προβλήματα με πεδίο ορισμού έναν συνεχή χώρο, καθώς η συνάρτηση ανταμοιβής εύκολα μπορεί να αποδίδει τιμές που σχηματίζουν ένα διάστημα.

Μία άλλη προσθήκη που μπορούμε να κάνουμε στη μελέτη μας είναι η δυνατότητα απενεργοποίησης τμημάτων ενός κώδικα. Όπως είδαμε στην ενότητα 2.2, ένας αυτοαναφορικός κώδικας σίγουρα μπορεί να πραγματοποιήσει τέτοιες διαδικασίες. Η χρησιμότητά τους φαίνεται αν σκεφτούμε παραδείγματα στα οποία δεν υπάρχει ένας συστηματικός τρόπος για τον πληθυσμό κωδίκων να μαντέψει ή να υπολογίσει όλες τις απαιτήσεις του περιβάλλοντος. Τέτοια παραδείγματα αποτελούν πιο ρεαλιστικά μοντέλα ενός φυσικού περιβάλλοντος, το οποίο σπάνια θα περιγραφόταν από έναν απλό, καθολικό κανόνα. Αυτό σημαίνει ότι σε πιο περίπλοκα παραδείγματα, αναμένουμε οι κώδικες να κάνουν συχνή χρήση της ικανότητας απενεργοποίησης. Είναι πάλι επιθυμητό να αποφύγουμε το δυαδικό πλαίσιο της επιβίωσης, ώστε οι κώδικες να έχουν περιθώριο για αλλεπάλληλες ενεργοποιήσεις και απενεργοποιήσεις των τμημάτων τους.

Οι παραπάνω έννοιες μπορούν να χρησιμοποιηθούν μαζί για να ορίσουμε την *αρνητική διαγωνοποίηση*. Όπως η διαδικασία Γ προσπαθεί να εντοπίσει μοτίβα που επιβίωσαν σε προηγούμενες γενιές και να τα χρησιμοποιήσει στο μέλλον, έτσι μπορούμε να ορίσουμε μία νέα διαδικασία Γ′, σκοπός της οποίας είναι να εντοπίσει μοτίβα τα οποία είχαν οδηγήσει σε μικρές τιμές της συνάρτησης ανταμοιβής και επομένως πρέπει να αποφευχθούν στο μέλλον. Πιο συγκεκριμένα, αν οι απόγονοι ενός πληθυσμού έχουν αποθηκεύσει στη μνήμη τους κώδικες με κακή απόδοση (για οποίο κριτήριο απόδοσης είναι κατάλληλο), τότε μπορούν να

εφαρμόσουν αρνητική διαγωνοποίηση σε αυτούς για να αντιληφθούν κανόνες προς αποφυγή. Μπορούμε να εκφράσουμε αυτή τη διαδικασία ως εξής:

Υπολόγισε τον απλούστερο κώδικα απόφασης r' έτσι ώστε
$alg(r')(c) = \texttt{False}$ ακριβώς για όλους τους απορριφθέντες κώδικες c. (Γ′)

Με τον όρο κώδικα απόφασης εννοούμε κάθε κώδικα που επιστρέφει μόνο `True` ή `False`, και που στην περίπτωσή μας έχει ως είσοδο άλλους κώδικες. Απορριφθέντα κώδικα μπορούμε να θεωρήσουμε όποιον δεν επιβίωσε σε προηγούμενα στάδια ή όποιον είχε τιμή ανταμοιβής κάτω από κάποιο όριο. Οι μετέπειτα απόγονοι μπορούν τώρα να κάνουν χρήση του r' για να έχουν καλύτερη πιθανότητα επιβίωσης. Το σύστημα πρέπει να υπολογίζει μόνο απογόνους c_i για τους οποίους $alg(r')(c) = \texttt{True}$. Είναι σημαντικό ο κώδικας r' να επιστρέφει `False` μόνο για τους ελαττωματικούς κώδικες και `True` για τους υπόλοιπους. Φυσικά, δεν μπορούμε να εξασφαλίσουμε ότι δεν θα παραχθεί άλλος απορριφθείς κώδικας, αλλά δίνουμε στο σύστημα έναν τρόπο να αποφεύγει σφάλματα που σύμφωνα με το ιστορικό επιβίωσης θα θεωρούσαμε προφανή.

Μία ενδιαφέρουσα παρατήρηση είναι ότι οι διαδικασίες Γ και Γ′ μπορούν να παράξουν η μία την άλλη, καθώς και οι δύο προσπαθούν με κάποιο τρόπο να βρουν χρήσιμες, γενικές συμπεριφορές για τον πληθυσμό κωδίκων. Με βάση όσα έχουμε πει, αυτές οι δύο διαδικασίες αποτελούν καλά παραδείγματα ακριβώς τέτοιων συμπεριφορών.

Κεφάλαιο 4

Αυτοαναφορά και εξέλιξη σε άλλα συστήματα

4.1 Καθολικές μηχανές Turing

Η πρώτη ρητή περιγραφή αλγορίθμων που διαβάζουν κώδικες έγινε από τον Turing στο αρχικό του άρθρο. Εκεί παρουσίασε, μεταξύ άλλων, την έννοια μίας καθολικής μηχανής υπολογισμού, που σήμερα ονομάζουμε καθολική μηχανή Turing [10].

Κάθε μηχανή Turing υπολογίζει μία (μερική) υπολογίσιμη συνάρτηση. Αποτελεί δηλαδή την υλοποίηση ενός συγκεκριμένου, στατικού προγράμματος. Όμως η λειτουργία κάθε τέτοιας μηχανής μπορεί να περιγραφεί από ένα string ενός κατάλληλου αλφαβήτου. Επομένως, είναι δυνατόν να ορίσουμε μία μηχανή U η οποία δέχεται ως είσοδο μία περιγραφή $c(M)$ (της μηχανής M), ακολουθούμενη από ένα string εισόδου s. Τότε, το αποτέλεσμα του υπολογισμού $U(c(M), s)$ θα είναι ακριβώς το ίδιο με το $M(s)$.

Φυσικά, σε κάθε υπολογιστικό σύστημα υπάρχει το αντίστοιχο της καθολικής μηχανής. Στην περίπτωση των αναδρομικών συναρτήσεων, η αντίστοιχη έννοια δίνεται από το θεώρημα κανονικής μορφής, σύμφωνα με το οποίο υπάρχουν πρωτογενώς αναδρομικές συναρτήσεις U και T τέτοιες ώστε

$$f(n) \simeq U(\mu x T(e, n, x)),$$

όπου f μία αυθαίρετη αναδρομική συνάρτηση και e ο αριθμός Gödel της f. Η ύπαρξη τέτοιων καθολικών τελεστών αποδείχτηκε από τον Kleene, με τεχνικές της αρίθμησης Gödel [9].

Αυτή η ιδέα δείχνει ξεκάθαρα ότι υπάρχουν αλγόριθμοι που μπορούν να διαβάζουν ή να επεξεργάζονται κώδικες. Στην πραγματικότητα, οι ηλεκτρονικοί υπολογιστές βασίζονται σε αυτήν την ιδιότητα. Όλοι οι επεξεργαστές, διερμηνευτές και μεταγλωττιστές που χρησιμοποιούμε σήμερα αποτελούν προσεγγίσεις καθολικών μηχανών, καθώς διαβάζουν, εκτελούν, και μερικές φορές επεξεργάζονται κώδικες άλλων αυθαίρετων προγραμμάτων.

4.2 Quines

Μία ενδιαφέρουσα εφαρμογή της αυτοαναφοράς στον προγραμματισμό είναι η ύπαρξη προγραμμάτων τα οποία έχουν ως μοναδική έξοδο τον κώδικα τους. Αυτά τα προγράμματα σήμερα ονομάζονται quines, προς τιμήν του φιλοσόφου και μαθηματικού Willard Quine, που μελέτησε εκτεταμένα την αυτοαναφορά στη λογική.

Η ύπαρξη quines προκύπτει ως άμεσο πόρισμα του εξής θεωρήματος, που αποτελεί μία μορφή του δεύτερου θεωρήματος αναδρομής του Kleene [4]:

Θεώρημα 1 *Έστω ϕ_i μία (αναδρομική) αρίθμηση των υπολογίσιμων συναρτήσεων. Τότε, για κάθε υπολογίσιμη συνάρτηση $Q(x, y)$, υπάρχει p τέτοιο ώστε:*

$$\phi_p(y) = Q(p, y).$$

Η εφαρμογή του θεωρήματος στην συνάρτηση $Q(x, y) = x$ μας εξασφαλίζει την ύπαρξη μιας συνάρτησης για την οποία ισχύει $\phi_p(y) = p$, δηλαδή μία συνάρτηση που επιστρέφει τον δείκτη της.

Το θεώρημα φυσικά ισχύει με αντίστοιχο τρόπο και σε άλλα υπολογιστικά μοντέλα. Στην περίπτωση των προγραμμάτων, οι δείκτες αντιστοιχούν στον κώδικα του αντίστοιχου προγράμματος. Αν εφαρμόσουμε το θεώρημα σε αυτό το υπολογιστικό σύστημα, παίρνουμε την ύπαρξη ενός προγράμματος που έχει ως μοναδική έξοδο τον κώδικα του, ανεξαρτήτως της εισόδου.

Επειδή η απόδειξη του θεωρήματος και όλα τα ενδιάμεσα βήματα μπορούν να γίνουν κατασκευαστικά, μας υποδεικνύουν ακριβώς πως να κατασκευάσουμε quines, κάτι που πλέον αποτελεί μία κλασική άσκηση σε διάφορες προγραμματιστικές γλώσσες.

```
c='c=%r;print(c%%c)';print(c%c)
```

Σχήμα 4.1: Παράδειγμα quine σε Python.

Όπως φαίνεται στο παράδειγμα, τα quines (όπως και όλα τα συστήματα που αντιγράφουν τον εαυτό τους) αποτελούνται από ένα τμήμα αποθηκευμένης πληροφορίας και ένα τμήμα εκτελέσιμου κώδικα, που πραγματοποιεί την αντιγραφή.

4.3 Βιολογικά συστήματα και αυτόματα

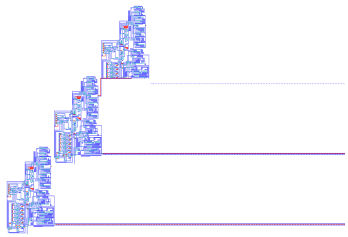
Η ιδέα της εξέλιξης μέσω (φυσικής) επιλογής προέρχεται από το γνωστό έργο του Δαρβίνου. Αν και τότε δεν υπήρχε ούτε το υπόβαθρο της Γενετικής, ούτε η Θεωρία Πληροφορίας, από το καιρό του John von Neumann η θεωρητική έρευνα πάνω στην εξέλιξη γίνεται με όλο και μεγαλύτερη έμφαση στα μαθηματικά. Τα διεπιστημονικά πεδία που έχουν αναπτυχθεί έχουν βοηθήσει και στην ανάπτυξη της βιολογίας και στη δημιουργία νέων μαθηματικών εργαλείων.

Οι κώδικες του DNA και του RNA ανήκουν στην κατηγορία των συστημάτων που ο von Neumann ονόμασε self-replicators, επειδή περιέχουν την απαραίτητη πληροφορία που περιγράφει τον τρόπο με τον οποίο θα αντιγράφουν τον εαυτό τους. Σύμφωνα με τη μοντέρνα θεωρία της φυσικής επιλογής, οποιαδήποτε άλλη λειτουργία κωδικοποιούν αυτά τα μόρια θα επιλεχθεί να περαστεί στους απογόνους τους μόνο εφόσον εξυπηρετεί την περαιτέρω αντιγραφή των γονιδίων. Επιπλέον, αρκετές από τις επιθυμητές ιδιότητες και δυνατότητες για αυτοαναφορικά συστήματα που έχουμε αναφέρει, έχουν παρατηρηθεί σε βιολογικούς κώδικες, όπως, για παράδειγμα η απενεργοποίηση ορισμένων γονιδίων ως αντίδραση σε διάφορες αλλαγές του περιβάλλοντος.

Εμπνευσμένος από την εξέλιξη μέσω φυσικής επιλογής, ο von Neumann όρισε το πρώτο κυτταρικό αυτόματο (cellular automaton), το 1949 [8]. Στη πιο συχνή του μορφή, ένα κυτταρικό αυτόματο αποτελείται από ένα άπειρο πλέγμα (συνήθως 2 διαστάσεων), συνοδευόμενο από κανόνες που καθορίζουν πότε κάθε κελί του πλέγματος βρίσκεται σε *ζωντανή* ή *νεκρή* κατάσταση. Στόχος του von Neumann ήταν να δείξει πως, όπως στην περίπτωση της βιολογικής αντιγραφής, απλοί κανόνες αναπαραγωγής και απλές αρχικές συνθήκες μπορούν να οδηγήσουν σε περίπλοκη συμπεριφορά.

Ένα από τα πιο χαρακτηριστικά παραδείγματα περίπλοκης συμπεριφοράς είναι ακριβώς η ικανότητα των αυτομάτων να υλοποιούν self-replicators. Αν και στην πράξη αυτά τα αυτόματα είναι υπερβολικά ευαίσθητα σε αλλαγές των

αρχικών συνθηκών και μικρές διαταραχές για να αποτελέσουν ένα χρήσιμο μοντέλο της εξέλιξης, άνοιξαν το δρόμο για τη μελέτη της βιολογικής εξέλιξης ως μηχανιστικό φαινόμενο.



Σχήμα 4.2: Ένα παράδειγμα self-replicator στο αυτόματο του von Neumann.

4.4 Γενετικοί αλγόριθμοι

Η εφαρμογή της εξέλιξης σε υπολογισμούς έγινε με εκτενέστερο τρόπο τις δεκαετίες του 1950 και 1960, με τη δημιουργία των γενετικών και εξελικτικών αλγορίθμων. Πέρα από τη χρήση τους ως γενικό εργαλείο βελτιστοποίησης, οι γενετικοί αλγόριθμοι αποδείχτηκαν ιδιαίτερα αποτελεσματικοί στον σχεδιασμό συστημάτων, έναν τομέα που παρουσιάζει σημαντικές δυσκολίες για άλλους τύπους αλγορίθμων [7].

Ένας γενετικός αλγόριθμος απαρτίζεται από την κωδικοποίηση του προβλήματος σε έναν χώρο αναζήτησης, έναν πληθυσμό υποψήφιων λύσεων και μία συνάρτηση προσαρμογής (fitting function). Οι λύσεις αναπαριστάνονται από strings του χώρου αναζήτησης, τα λεγόμενα *χρωμοσώματα*. Ο αρχικός πληθυσμός μπορεί να επιλεχτεί τυχαία, και κάθε επόμενη γενιά προκύπτει από τα πιο ευπροσάρμοστα στοιχεία της προηγούμενης, σε συνδυασμό με τυχαίες μεταλλάξεις των χρωμοσωμάτων. Μέσω αυτής της επαναλαμβανόμενης διαδικασίας οι λύσεις σταδιακά προσεγγίζουν ένα βέλτιστο του χώρου αναζήτησης.

Συνήθως τα χρωμοσώματα των λύσεων αντιστοιχούν σε διαφορετικές τιμές των παραμέτρων με τις οποίες θα τρέξει ο αλγόριθμος. Έτσι, η εξέλιξη του συστήματος οδηγεί σε μία βέλτιστη επιλογή παραμέτρων, τουλάχιστον τοπικά. Όμως στη γενική περίπτωση, οι υποψήφιες λύσεις, και άρα τα χρωμοσώματα, μπορούν να αντιστοιχούν σε διαφορετικούς αλγορίθμους που προσπαθούν να λύσουν το ίδιο πρόβλημα. Με αυτόν τον τρόπο, η τεχνική αυτή αποτελεί ακόμη ένα παράδειγμα του δυϊσμού μεταξύ αλγορίθμων και data.

4.5 Συγκρίσεις και συμπεράσματα

Από τα παραπάνω παραδείγματα φαίνεται ότι οι ιδέες της της εξέλιξης μέσω φυσικής επιλογής και της αυτοεπεξεργασίας και έχουν αποτελέσει από καιρό στοιχεία μελέτης στη θεωρητική πληροφορική. Η εξέλιξη υπό κατάλληλες συνθήκες έχει αποδειχθεί απροσδόκητα αποτελεσματική, ενώ και διάφορες άλλες έννοιες που προέρχονται από τη βιολογία εφαρμόζονται με μεγάλη επιτυχία σε κλάδους όπως η μηχανική μάθηση. Από την άλλη, η αυτοαναφορά ανέκαθεν αποτελούσε ένα από τα κύρια θεωρητικά υπόβαθρα της Επιστήμης Υπολογιστών.

Σε αντίθεση με άλλες υπολογιστικές εφαρμογές της εξέλιξης και της επιλογής, η διαγωνοποίηση όπως την περιγράψαμε δεν κάνει χρήση τυχαίων μεταλλάξεων. Ειδικά σε προβλήματα βελτιστοποίησης, οι τυχαίες μετακινήσεις χρησιμοποιούνται συχνά στην εξερεύνηση του χώρου αναζήτησης (search space), καθώς βοηθούν στην αποφυγή τοπικών ακροτάτων. Για τον ίδιο λόγο έχει σημασία να μελετηθούν τρόποι με τους οποίους η διαγωνοποίηση μπορεί να γίνει μη ντετερμινιστική, καθώς και αυτή έχει χαρακτηριστικά 'άπληστου' αλγορίθμου. Η τυχαιότητα μπορεί να εισαχθεί στο σύστημα με τη χρήση στοχαστικών αλγορίθμων, είτε ως μέλη του αρχικού πληθυσμού, είτε ως κομμάτι της διαγωνοποίησης. Αν, για παράδειγμα, κάποια παράμετρος των κωδίκων καθορίζει τον βαθμό με τον οποίο εξελίσσονται τυχαία οι κώδικες, τότε αυτή η παράμετρος μπορεί να πάρει μέρος στην αλληλεπίδραση της διαγωνοποίησης και του περιβάλλοντος, και να βελτιστοποιηθεί σταδιακά για τον εκάστοτε πληθυσμό.

Ένας από τους πιο συχνούς τρόπους με τον οποίο η πληροφορική χρησιμοποιεί την αυτοαναφορά, είναι η εκμετάλλευση του δυϊσμού μεταξύ code και data. Ως data συνήθως θεωρούμε όλα τα στοιχεία που εισάγονται, εξάγονται και αποθηκεύονται από κώδικες. Όπως όμως είδαμε, οι δυο έννοιες συχνά ταυτίζονται. Η ίδια ιδέα βρίσκεται και στο κέντρο του φορμαλισμού της διαγωνιοποίησης, αλλά με ιδιαίτερη έμφαση στην αναδρομική ισχύ της. Αυτό γίνεται ιδιαίτερα ξεκάθαρο όταν τη χρησιμοποιούμε για την αντιμετώπιση πιο σύνθετων προβλημάτων. Αυτά τα προβλήματα συχνά αποτελούνται από συνδυασμούς πιο απλών προβλημάτων, κάτι που καθιστά την αναδρομή κατάλληλη μεθοδολογία για την αντιμετώπισή τους. Αυτό είναι που επιτρέπει στο σύστημα που περιγράψαμε να αντιλαμβάνεται πιο πλούσια μοτίβα , κάτι που θα ήταν δύσκολο έως αδύνατό χωρίς αυτοαναφορά.

Βιβλιογραφία

[1] A. D. Arvanitakis. "Recursion and evolution: Part I". In: *arXiv e-prints*, arXiv:2001.11825 (Jan. 2020), arXiv:2001.11825.

[2] A. Arvanitakis. "Recursion and evolution: Part II". In: *arXiv e-prints*, arXiv:2007.04982 (June 2020), arXiv:2007.04982.

[3] K. Gödel. "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I.". In: *Monatsh. f. Mathematik und Physik* 38 (1931), pp. 173–198.

[4] S. C. Kleene. "On Notation for Ordinal Numbers". In: *The Journal of Symbolic Logic* 3.4 (1938), pp. 150–155.

[5] S. Kleene. *Introduction to Metamathematics*. Bibliotheca Mathematica. Wolters-Noordhoff, 1952.

[6] Y. Liu et al. "How Well Do Machines Perform on IQ tests: a Comparison Study on a Large-Scale Dataset". In: Aug. 2019, pp. 6110–6116.

[7] J. D. Lohn, G. S. Hornby, and D. S. Linden. "An Evolved Antenna for Deployment on Nasa's Space Technology 5 Mission". In: *Genetic Programming Theory and Practice II*. Ed. by U.-M. O'Reilly et al. Boston, MA: Springer US, 2005, pp. 301–315.

[8] J. V. Neumann and A. W. Burks. *Theory of Self-Reproducing Automata*. USA: University of Illinois Press, 1966.

[9] H. Rogers. *Theory of recursive functions and effective computability*. Cambridge, Mass: MIT Press, 1987.

[10] A. M. Turing. "On Computable Numbers, with an Application to the Entscheidungsproblem". In: *Proceedings of the London Mathematical Society* s2-42.1 (Jan. 1937), pp. 230–265.