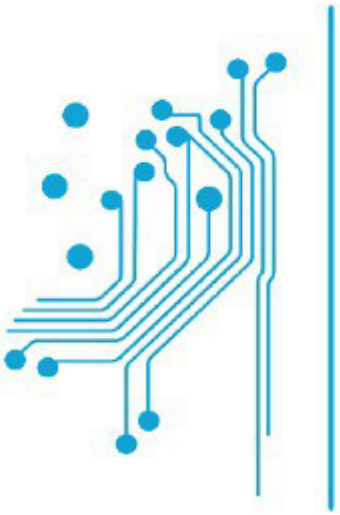


ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

"Αλγόριθμοι και μετρικές για τον ορισμό ομοιότητας εννοιών σε ετερογενή μοντέλα πληροφοριών"

Σαμαρά Βασιλική
Α.Μ. 0841

Επιβλέπων Καθηγητής
Κούγιας Ιωάννης

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Αντίρριο, / /2015

Επιτροπή αξιολόγησης:

1.

2.

3.

ΠΕΡΙΕΧΟΜΕΝΑ:

ΠΕΡΙΛΗΨΗ

Κύριος σκοπός της παρούσας πτυχιακής εργασίας είναι η καταγραφή και η ανάλυση των μοντέλων που χρησιμοποιούνται για τον ορισμό ομοιότητας εννοιών σε ετερογενή μοντέλα πληροφοριών, καθώς και η μελέτη αλγορίθμων ταιριάσματος που χρησιμοποιούνται σε αυτά.

Παρουσιάζεται επίσης η λειτουργία των συγκεκριμένων αλγορίθμων χρησιμοποιώντας κώδικα c++, και καταγράφηκαν ετερογενή μοντέλα πληροφοριών και μετρικές ομοιότητας.

ABSTRACT

The main purpose of this diploma dissertation is the recording and analysis of the models used to define similarity measures in heterogeneous information models, as well as the study of matching algorithms used in them.

The dissertation also provides the operation of these algorithms using c++ code and we recorded heterogeneous information models and similarity measures.

ΚΕΦΑΛΑΙΟ 1

ΑΝΑΠΑΡΑΣΤΑΣΗ ΓΝΩΣΗΣ

1.1) ΕΙΣΑΓΩΓΗ

Στη σύγχρονη εποχή η πληθώρα των πληροφοριών που διακινούνται καθιστά επιτακτική την ανάγκη ανάλυσης και κατανόησης τους, ώστε να μας είναι χρήσιμες. Εξαιτίας της αυξημένης πολυπλοκότητας που απαιτεί η διαχείριση ενός τόσο μεγάλου όγκου πληροφοριών απαιτούνται μέθοδοι και τεχνικές που προσεγγίζουν το πρόβλημα σημασιολογικά. Έτσι τις τελευταίες δεκαετίες, η τεχνητή νοημοσύνη, που γεννήθηκε στο χώρο της επιστήμης των υπολογιστών, έχει αναδείξει έναν νέο κλάδο, αυτόν της αναπαράστασης γνώσης και συλλογιστικής.

Ένας ορισμός που μπορεί να δοθεί για την τεχνητή νοημοσύνη, είναι ότι πραγματεύεται τη μελέτη της ευφυούς συμπεριφοράς όταν αυτή επιτυγχάνεται με υπολογιστικά μέσα. Βασιζόμενοι σε αυτό τον ορισμό, μπορούμε να οριοθετήσουμε την αναπαράσταση γνώσης και συλλογιστική, ως τον τομέα εκείνο της τεχνητής νοημοσύνης που ασχολείται με το πώς ένας (ηλεκτρονικός) πράκτορας διατυπώνει και χρησιμοποιεί τη γνώση που διαθέτει για τη λήψη αποφάσεων.

Ο κλάδος της αναπαράστασης γνώσης και συλλογιστικής επικεντρώνεται σε 3 βασικές έννοιες: τη γνώση, την αναπαράσταση της και τη συλλογιστική. Η γνώση, αποτελεί μια αρκετά αφηρημένη έννοια, η οποία είναι δύσκολο να οριστεί. Ωστόσο μία προσεγγιση είναι ότι αποτελεί ένα είδος συσχέτισης ανάμεσα σε ένα υποκείμενο, κάποιον που γνωρίζει, και μία πρόταση, η οποία αποτελεί το περιεχόμενο της γνώσης. Ένα σχετικό παράδειγμα αποτελεί η φράση «Ο Κώστας γνωρίζει ότι ο Νίκος θα συμμετέχει στη συναυλία». Η αναπαράσταση πάλι, πολυ αδρομερώς, μπορεί να περιγραφεί ως μία αντιστοίχιση ανάμεσα σε δύο κόσμους:ο ένας κόσμος αντιπροσωπεύει κομμάτια του δεύτερου, όμως με περισσότερο αναγνωρίσιμο και παραστατικό τρόπο.

Τέλος, αναφερόμενοι στη συλλογιστική, αυτή σχετίζεται με την επεξεργασία συμβόλων τα οποία αντιπροσωπεύουν ένα σύνολο από προτάσεις, προκειμένου να επιτύχουν την παραγωγή νέων συμβόλων. Η λογική επαγωγή αποτελεί τη ραχοκοκκαλιά των περισσότερων αλγορίθμων που προέρχονται από αυτόν τον τομέα.Βασική εφαρμογή του παραπάνω κλάδου είναι η ανάπτυξη συστημάτων βασισμένων σε γνώση(knowledge-based systems). Κύριο χαρακτηριστικό αυτων των συστημάτων είναι η παρουσία μιας βάσης γνώσης, δηλαδή μιας συλλογής δομών με συμβολικό χαρακτήρα που είναι αποθηκευμένα στο σύστημα και καθοδηγούν τη διαδικασία συλλογιστικής.

1.2) ΜΟΝΤΕΛΑ ΑΝΑΠΑΡΑΣΤΑΣΗΣ ΓΝΩΣΗΣ

Ο σκοπός της κατανόησης του τι σημαίνει γνώση , και ποιοι τύποι γνώσης υπάρχουν, είναι να μας επιτρέπεται να τη χρησιμοποιήσουμε σε τεχνητά συστήματα. Αυτή η μακροχρόνια φιλοδοξία έχει πυροδοτηθεί από την επιθυμία να αναπτυχθούν έξυπνες τεχνολογίες που επιτρέπουν στους υπολογιστές να εκτελούν περίπλοκες εργασίες,ώστε να βοηθήσουν τους ανθρώπους ή επειδή οι άνθρωποι δεν μπορούν να τις εκτελέσουν.

Σε αυτό το τμήμα θα εξηγήσουμε πώς η γνώση μπορεί να χρησιμοποιηθεί σε συστήματα ηλεκτρονικών υπολογιστών μέσω διαφορετικών μοντέλων αναπαράστασης γνώσης.Τα μοντέλα είναι αναπαραστάσεις των θεωριών που μας επιτρέπουν να εκτελέσουμε προσομοιώσεις και να πραγματοποιήσουμε τις δοκιμές που θα καθιστούσαν τα αποτελέσματα που προβλέπει η θεωρία.

Ως εκ τούτου, όταν μιλάμε για μοντέλα αναπαράστασης γνώσης αναφερόμαστε σε ένα συγκεκριμένο τρόπο αναπαράστασης γνώσης που θα επέτρεπε την πρόβλεψη του τι ένα σύστημα ξέρει και για τι είναι ικανό με μηχανισμούς γνώσης συλλογιστικής.

Οι τύποι των μοντέλων αναπαράστασης που χρησιμοποιούνται για συστήματα γνώσης περιλαμβάνουν κατανεμημένη,συμβολική, δηλωτική, πιθανολογική και βασισμένη σε κανόνες αναπαράσταση.

1.2.1) ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

Κατανεμημένο σύστημα είναι ένα σύνολο αυτόνομων υπολογιστών το οποίο παρουσιάζεται στους χρήστες του σαν ένας ενιαίος υπολογιστής.Είναι ένα σύστημα του οποίου τα συστατικά (υλικό και λογισμικό) βρίσκονται σε δικτυωμένους υπολογιστές και επικοινωνούν μεταξύ τους για να συντονίσουν τη δράση τους μέσω ανταλλαγής μηνυμάτων.

Το μεγαλύτερο παράδειγμα ενός κατανεμημένου συστήματος είναι το internet. Το Internet είναι ένα μεγάλο κατανεμημένο σύστημα καθώς οι χρήστες του, όπου και αν βρίσκονται, μπορούν να χρησιμοποιούν υπηρεσίες που παρέχονται από απομακρυσμένους εξυπηρετητές, όπως πρόσβαση σε ιστοσελίδες του παγκόσμιου ιστού (World Wide Web), email, μεταφορά αρχείων (ftp), πρόσβαση σε βάσεις δεδομένων, κλπ.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΚΑΤΑΝΕΜΗΜΕΝΩΝ ΣΥΣΤΗΜΑΤΩΝ

1) Μερισμός πόρων

- Κοινή χρήση πόρων υλικού και λογισμικού

2) Ανοικτή λειτουργία

-Χρήση εξοπλισμού και λογισμικού από διάφορους κατασκευαστές

3) Ταυτοχρονισμός

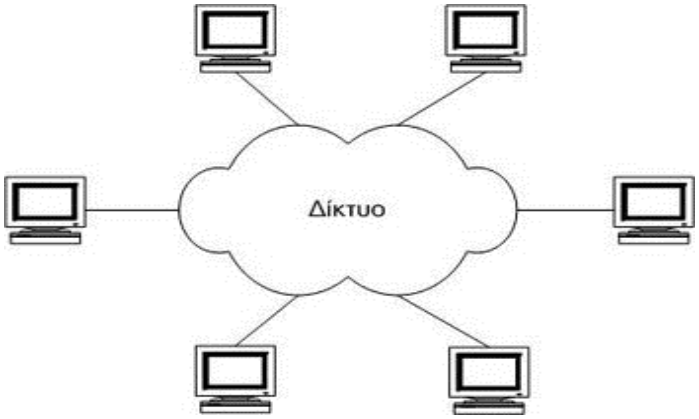
- Ταυτόχρονη επεξεργασία για τη βελτίωση της απόδοσης

4) Επεκτασιμότητα

- Αυξημένη διεκπεραιωτική ικανότητα με την προσθήκη νέων πόρων

5) Ανοχή σφαλμάτων

- Η ικανότητα συνέχειας της λειτουργίας αφού ανακύψει κάποιο σφάλμα.



Εικόνα 1:Το διαδίκτυο ως παράδειγμα κατανεμημένου συστήματος

ΔΕΝΤΡΑ ΑΠΟΦΑΣΗΣ

Τα δέντρα αποφάσεων είναι δεντρο-διαμορφωμένες δομές που αντιπροσωπεύουν τα σύνολα αποφάσεων. Αυτές οι αποφάσεις παράγουν τους κανόνες για την ταξινόμηση ενός συνόλου δεδομένων.

Μια απλή δομή όπου οι μη τερματικοί κόμβοι αντιπροσωπεύουν τα αποτελέσματα των αποφάσεων. Τα δέντρα αποφάσεων έχουν διάφορα πλεονεκτήματα,όπως το ότι είναι εύκολο να τα καταλάβουμε, μπορούν να μετασχηματιστούν σε κανόνες και πειραματικά έχει αποδειχθεί ότι λειτουργούν πολύ καλά.

Ένας αλγόριθμος κατασκευής δέντρων απόφασης

Τα δέντρα απόφασης κατασκευάζονται χρησιμοποιώντας μόνο εκείνα τα γνωρίσματα που είναι σε θέση να διακρίνουν τις έννοιες προς εκμάθηση.Για να χτίσουμε ένα δέντρο απόφασης, πρέπει αρχικά να επιλέξουμε ένα υποσύνολο περιπτώσεων από το σύνολο των δεδομένων που θα χρησιμοποιηθούν στην εκπαίδευση (**υποσύνολο δεδομένων εκπαίδευσης-training set**).

Αυτό το υποσύνολο (**δεδομένα ελέγχου-test set**) χρησιμοποιείται έπειτα από τον αλγόριθμο για να κατασκευάσει το δέντρο απόφασης. Τα υπόλοιπα δεδομένα, τα δεδομένα training set, χρησιμοποιούνται στην εξέταση της ακρίβειας του κατασκευασμένου δέντρου. Εάν το δέντρο απόφασης ταξινομεί τις περιπτώσεις σωστά,η διαδικασία ολοκληρώνεται. Εάν μια περίπτωση είναι ανακριβώς ταξινομημένη,η περίπτωση προστίθεται στο επιλεγμένο υποσύνολο των training set και ένα νέο δέντρο κατασκευάζεται.

Τα βήματα του αλγορίθμου είναι τα ακόλουθα:

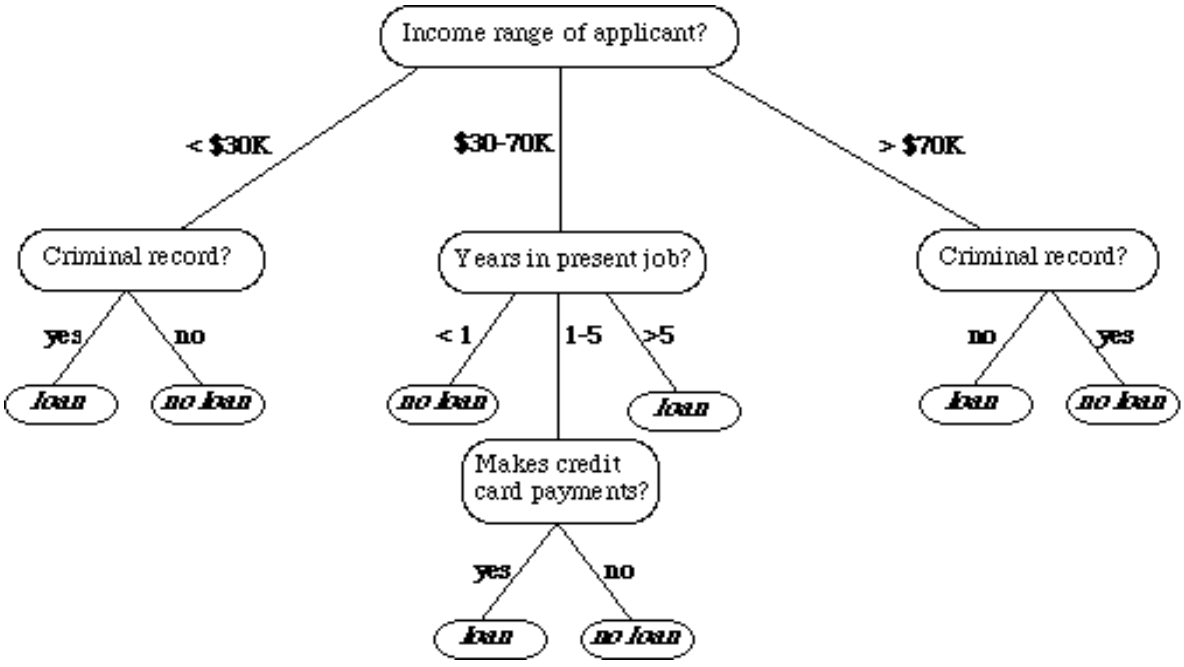
1) Έστω Τ είναι το σύνολο στιγμιότυπων εκπαίδευσης, το training set.

2) Επιλέγουμε ένα χαρακτηριστικό που διαφοροποιεί καλύτερα τις περιπτώσεις που περιλαμβάνονται στο Τ.

3) Δημιουργούμε έναν κόμβο στο δέντρο του οποίου η αξία είναι το επιλεγμένο χαρακτηριστικό. Δημιουργούμε θυγατρικούς δεσμούς από αυτόν τον κόμβο,όπου κάθε σύνδεση αντιπροσωπεύει μια μοναδική αξία για τα επιλεγμένα χαρακτηριστικά. Χρησιμοποιούμε τις τιμές των θυγατρικών δεσμών για να υποδιαιρέσουμε περαιτέρω τα στιγμιότυπα σε δευτερεύουσες κλάσεις.

4) Για κάθε δευτερεύουσα κλάση που δημιουργήθηκε στο βήμα 3: Εάν τα στιγμιότυπα στη δευτερεύουσα κλάση ικανοποιούν προκαθορισμένα κριτήρια ή εάν το σύνολο των υπολοίπων επιλογών γνωρισμάτων γι'αυτή τη διαδρομή του δέντρου είναι μηδέν, καθορίζουμε την κατηγοριοποίηση των καινούργιων στιγμιότυπων που ακολουθούν αυτή τη διαδρομή αποφάσεων. Εάν η δευτερεύουσα κλάση δεν ικανοποιεί τα προκαθορισμένα κριτήρια, και υπάρχει τουλάχιστον ένα γνώρισμα για να υποδιαιρέσει περαιτέρω τη διαδρομή του δέντρου, αφήστε το T να είναι το τρέχον σύνολο των στιγμιότυπων της δευτερεύουσας κλάσης και επιστρέφουμε στο βήμα 2.

Παράδειγμα δέντρου απόφασης



Εικόνα 2: ένα δέντρο απόφασης για να βοηθήσει ένα χρηματοπιστωτικό ίδρυμα να αποφασίσει εάν θα πρέπει να προσφερθεί σε ένα άτομο ένα δάνειο

ΑΛΓΟΡΙΘΜΟΣ ID3

Μια από της διαδεδομένες και ταυτόχρονα απλές τεχνικές που χρησιμοποιείται για την κατασκευή δένδρων αποφάσεων είναι ο αλγόριθμος ID3. Εφευρέθηκε από τον Ross Quinlan και αυτό που προσπαθεί να επιτύχει αυτός ο αλγόριθμος είναι να ελαχιστοποιήσει τον αριθμό των συγκρίσεων.

Η βασική ιδέα ενός αλγορίθμου επαγωγής είναι να κάνει ερωτήσεις των οποίων οι απαντήσεις να περιέχουν την περισσότερη πληροφορία. Λέγοντας περισσότερη πληροφορία εννοούμε ερωτήσεις που απορρίπτουν μεγάλο μέρος του χώρου αναζήτησης. Για παράδειγμα, μια ερώτηση της μορφής «is the thing alive?» είναι καλύτερη από την ερώτηση «is it my daddy?» και αυτό γιατί η πρώτη ερώτηση χωρίζει το χώρο αναζήτησης σε δύο μεγάλα πεδία σε αντίθεση με τη δεύτερη που εκτελεί μια μικρή διαίρεση του χώρου.

Η βασική ιδέα του αλγορίθμου ID3 είναι η επιλογή χαρακτηριστικών διάσπασης που περιέχουν μεγαλύτερο κέρδος πληροφορίας. Το ποσό της πληροφορίας, το οποίο σχετίζεται με την τιμή ενός χαρακτηριστικού, εξαρτάται από την πιθανότητα εμφάνισης του.

Η έννοια που χρησιμοποιείται για να μετρηθεί η πληροφορία καλείται εντροπία (Entropy). Χρησιμοποιούμε το μέτρο της εντροπίας ώστε να μετρήσουμε το πόσο ανομοιογενές είναι ένα σύνολο δεδομένων. Το μέτρο αυτό παίρνει τιμές στο διάστημα [0,1].

Ο τυπικός ορισμός της εντροπίας παρουσιάζεται στον **ορισμό**.

Ορισμός: Με δεδομένες τις πιθανότητες $p_1, p_2, \ldots, p_s$ με

$$\sum_{i=1}^{s} p_i = 1$$

η εντροπία ορίζεται ως:

$$H(p_1, p_2, \ldots, p_s) = \sum_{i=1}^{s} \left[p_i \log\left(\frac{1}{p_i}\right) \right]$$

Δεδομένης μια κατάστασης D, το H(D) , βρίσκει την ποσότητα της τάξης σε αυτή την κατάσταση. Όταν η κατάσταση D, χωρίζεται σε s νέες καταστάσεις $S = \{D_1, D_2, ..., D_s\}$, το μέτρο της εντροπίας μπορεί να εφαρμοστεί σε κάθε μια από αυτές τις νέες καταστάσεις. Κάθε βήμα του ID3 επιλέγει την κατάσταση η οποία διατάσσει περισσότερο τη διάσπαση.

Μια κατάσταση της Βάσης Δεδομένων είναι απολύτως διατεταγμένη αν όλες οι πλειάδες σε αυτήν ανήκουν στην ίδια κατηγορία. Ο ID3 επιλέγει το χαρακτηριστικό διάσπασης με το μεγαλύτερο κέρδος πληροφορίας.

Κέρδος (gain) πληροφορίας μετρά την μείωση της εντροπίας που θα προκληθεί αν χωριστεί το σύνολο δεδομένων με βάση κάποιο χαρακτηριστικό.Καταλήγοντας, Ο ID3 αλγόριθμος υπολογίζει το κέρδος μιας διάσπασης χρησιμοποιώντας των εξής τύπο:

$$Gain(D,S) = H(D) - \sum_{i=1}^{s} P(D_i) H(D_i)$$

Ο πρώτος όρος της διαφοράς είναι η εντροπία του συνόλου δεδομένων ενώ ο δεύτερος όρος είναι η εντροπία των δεδομένων μετά τη διάσπαση τους ανάλογα με την τιμή του χαρακτηριστικού S.

Ο δεύτερος όρος αποτελείται από το άθροισμα της εντροπίας για το κάθε σύνολο που προκύπτει μετά τη διάσπαση. Μια γενική περιγραφή του ID3 παρουσιάζεται παρακάτω:

1. Αρχικά πρέπει να επιλεχτεί το πιο κατάλληλο χαρακτηριστικό για έλεγχο στη ρίζα.

2. Στη συνέχεια, για κάθε δυνατή τιμή του χαρακτηριστικού δημιουργούνται οι αντίστοιχοι απόγονοι της ρίζας. Τα δεδομένα μοιράζονται στους νέους κόμβους ανάλογα με την τιμή που έχουν για το χαρακτηριστικό που ελέγχεται στη ρίζα.

3. Η όλη διαδικασία επαναλαμβάνεται για κάθε νέο κόμβο. Η επιλογή του χαρακτηριστικού θα γίνει βάσει των δεδομένων που ανήκουν στον κάθε κόμβο.

4. Ένας κόμβος γίνεται φύλλο όταν όλα τα δεδομένα που ανήκουν σε αυτόν ανήκουν στην ίδια κατηγορία (αμιγής κόμβος). Η κατηγορία αυτή γίνεται και η τιμή του φύλλου.

5. Αν σε κάποιο βάθος τελειώσουν τα χαρακτηριστικά προς έλεγχο, τότε ο κόμβος γίνεται τερματικός και σαν τιμή παίρνει εκείνη που έχει την πλειοψηφία με βάση τα δεδομένα του κόμβου αυτού.

1.2.2) ΣΥΜΒΟΛΙΚΑ ΣΥΣΤΗΜΑΤΑ

Τα συμβολικά συστήματα ονομάζονται έτσι επειδή χρησιμοποιούν κατανοητές από τον άνθρωπο παραστάσεις που βασίζονται σε σύμβολα ως τη βασική μονάδα εκπροσώπησης, καθένα από τα σύμβολα σημαίνουν κάτι όπως: μια λέξη, μια έννοια, μια ικανότητα, μια διαδικασία, μια ιδέα. Επίσης τα συμβολικά συστήματα περιλαμβάνουν δομές όπως τα σημασιολογικά δίκτυα, τα βασισμένα σε κανόνες συστήματα και τα πλαίσια.

Ο απλούστερος και πιο κατανοητός τρόπος για τη συμβολική αναπαράσταση γνώσης είναι η προτασιακή λογική (propositional logic). Η προτασιακή λογική αποτελεί την απλούστερη μορφή μαθηματικής λογικής και η παρουσίασή της κρίνεται αναγκαία, καθώς αποτελεί τη βάση για περισσότερο πολύπλοκες λογικές.

Στην προτασιακή λογική κάθε γεγονός του πραγματικού κόσμου αναπαριστάται με μια λογική πρόταση, η οποία χαρακτηρίζεται είτε ως αληθής (t-true) ή ως ψευδής (f-false), μπορεί δηλαδή να έχει δύο λογικές τιμές. Οι λογικές προτάσεις αναπαριστώνται συνήθως από λατινικούς χαρακτήρες P, Q, R, κτλ., και ονομάζονται άτομα (atoms). Τα άτομα μπορούν να συνδυαστούν με τη χρήση λογικών συμβόλων ή συνδετικών (connectives) και οι σύνθετες προτάσεις που προκύπτουν ονομάζονται ορθά δομημένοι τύποι (well formed formulae).

Παρακάτω παρουσιάζονται τα συνδετικά της προτασιακής λογικής. Θα πρέπει να σημειωθεί ότι η σύνταξη περιλαμβάνει και τρία σημεία στίξης, τα "(" και ")" (παρενθέσεις) και το "," (κόμμα).

Σύμβολο	***Ονομασία / Επεξήγηση***
∧	σύζευξη (λογικό "ΚΑΙ")
∨	διάζευξη (λογικό "Η")
¬	άρνηση
→	συνεπαγωγή ("ΕΑΝ ΤΟΤΕ")
↔	διπλή συνεπαγωγή ή ισοδυναμία ("ΑΝ ΚΑΙ ΜΟΝΟ ΑΝ").

ΠΑΡΑΔΕΙΓΜΑ:

Αναπαράσταση της ακόλουθης γνώσης με προτασιακή λογική:

- 1η πρόταση: “επιδιώκω την ειρήνη”
- 2η πρόταση: “αποφεύγω τον πόλεμο”
- 3η πρόταση: “εάν επιδιώκω την ειρήνη,τότε αποφεύγω τον πόλεμο”

Σε κάθε πρόταση αντιστοιχεί ένας λατινικός χαρακτήρας.

- **P** : “επιδιώκω την ειρήνη”
- **Q** :”αποφεύγω τον πόλεμο”

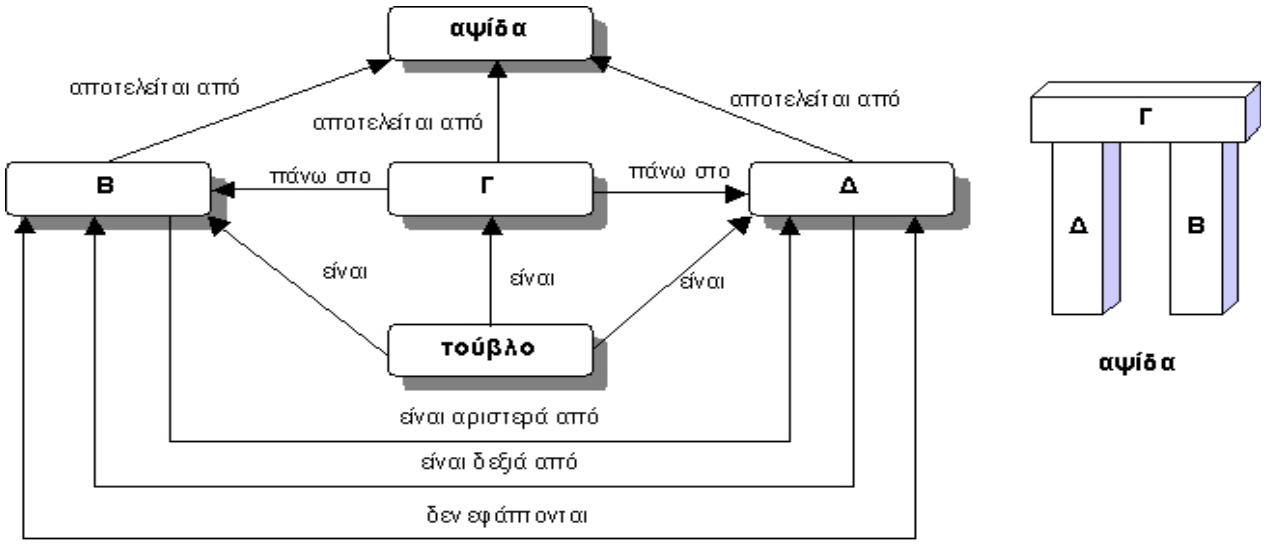
Η τρίτη πρόταση αναπαριστάται με τη χρήση του συνδετικού της συνεπαγωγής:
P→ Q “εάν επιδιώκω την ειρήνη,τότε αποφεύγω τον πόλεμο”

1.2.3) ΣΗΜΑΣΙΟΛΟΓΙΚΑ ΔΙΚΤΥΑ

Ένα σημασιολογικό δίκτυο είναι ένα δίκτυο το οποίο αντιπροσωπεύει σημασιολογικές σχέσεις μεταξύ των εννοιών και χρησιμοποιείται συχνά ως μια μορφή αναπαράστασης γνώσης. Αποτελείται από κόμβους (nodes) και δεσμούς (links) ανάμεσά τους. Οι κόμβοι υποδηλώνουν κλάσεις αντικειμένων (classes), αντικείμενα (objects), έννοιες (concepts), τιμές ιδιοτήτων (values), κλπ. και οι δεσμοί τις σχέσεις (relations) μεταξύ αυτών των αντικειμένων ή ιδιότητες που συνδέουν αντικείμενα με τιμές.

Τα σημασιολογικά δίκτυα αναπτύχθηκαν αρχικά στα πλαίσια της τεχνητής νοημοσύνης ως μέσο για την αναπαράσταση της ανθρώπινης μνήμης και της κατανόησης της γλώσσας. Ένα παράδειγμα ενός σημασιολογικού δικτύου είναι το **WordNet**, μία λεξιλογική βάση δεδομένων των αγγλικών η οποία θεωρείται ότι είναι η πιο σημαντική πηγή, που είναι διαθέσιμη στους ερευνητές της υπολογιστικής γλωσσολογίας, της ανάλυσης κειμένου, και πολλών άλλων σχετικών περιοχών. Κατασκευάστηκε στο πανεπιστήμιο του Princeton από μια ομάδα ειδικών γλωσσολόγων.

Το WordNet ομαδοποιεί αγγλικές λέξεις σε σύνολα συνωνύμων,παρέχει μικρούς, γενικούς ορισμούς, και καταγράφει τις διάφορες σημασιολογικές σχέσεις μεταξύ αυτού του συνόλου συνωνύμων.



Εικόνα 3: παράδειγμα σημασιολογικού δικτύου. Μία αψίδα που αποτελείται απο 3 τούβλα Β, Γ και Δ

Εικόνα 4: χρήση του wordnet για τη σημασιολογία της λέξης "network"(δίκτυο)

ΙΕΡΑΡΧΙΚΗ ΔΟΜΗ ΕΝΟΣ ΣΗΜΑΣΙΟΛΟΓΙΚΟΥ ΔΙΚΤΥΟΥ

Σχέσεις μεταξύ κόμβων:

1) a_kind_of (δεσμός AKO)

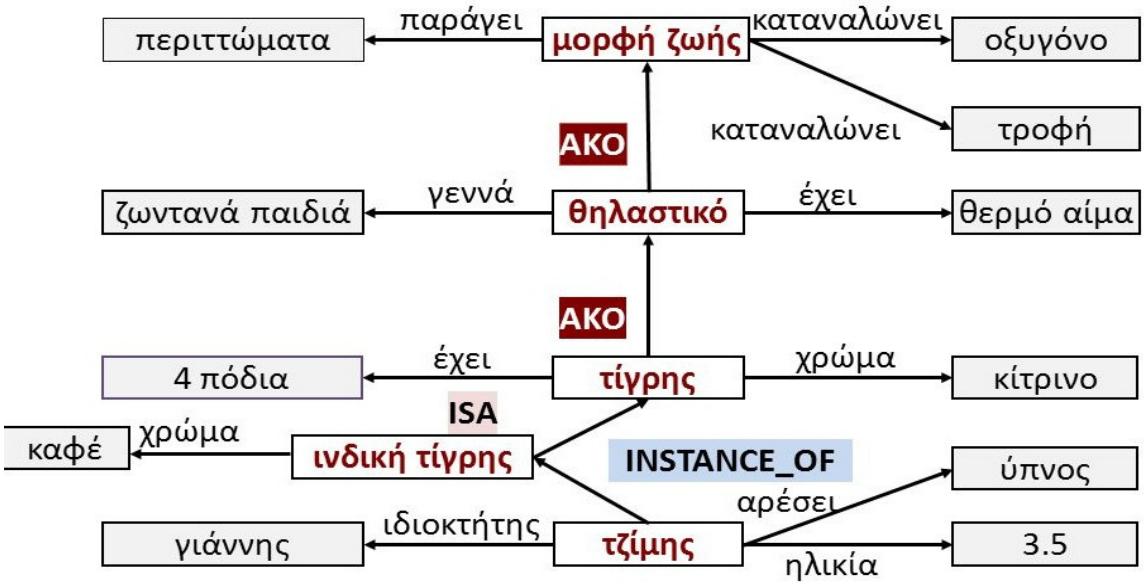
Η σχέση **AKO** υπάρχει μεταξύ κλάσεων αντικειμένων. Σε κόμβο που συνδέεται με σχέση AKO με κάποιον άλλον μπορούν να προστεθούν νέοι δεσμοί που προσδίδουν νέες ιδιότητες.

2) is_a (δεσμός ISA)

Είναι παρόμοια με τη σχέση AKO, **με τη διαφορά ότι** δε μπορεί να προστεθούν νέες ιδιότητες παρά μόνον να κληρονομηθούν οι ήδη υπάρχουσες ιδιότητες από κόμβους ψηλότερα στην ιεραρχία ή οι ιδιότητες αυτές να αλλάξουν τιμές

3) instance_of (δεσμός INSTANCE_OF)

Υπάρχει μόνο μεταξύ κόμβων αντικειμένων και κόμβων γενικότερων κλάσεων



Εικόνα 5: Παράδειγμα ιεραρχικού σημασιολογικού δικτύου

1.2.4) ΣΥΣΤΗΜΑΤΑ ΒΑΣΙΣΜΕΝΑ ΣΕ ΚΑΝΟΝΕΣ

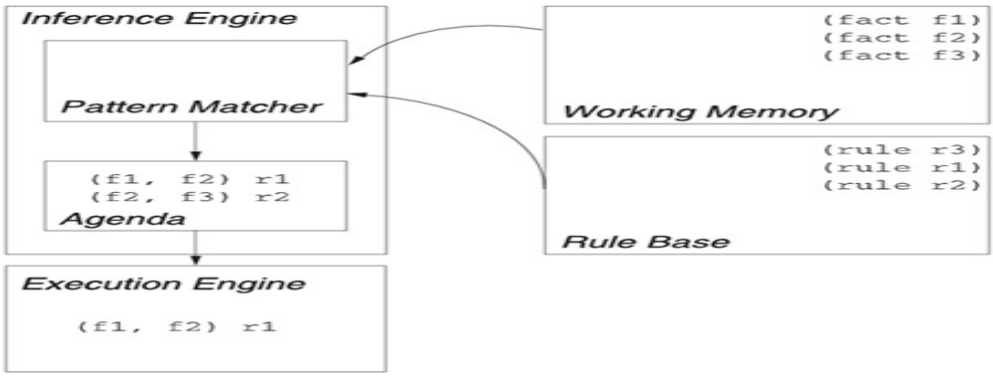
Στην επιστήμη των υπολογιστών, τα συστήματα κανόνων χρησιμοποιούνται ως ένας τρόπος για να αποθηκεύεται και να χειραγωγείται γνώση με σκοπό να ερμηνεύονται πληροφορίες με ένα χρήσιμο τρόπο. Βασίζονται σε συμβολικές αναπαραστάσεις μοντέλων που επικεντρώνονται σε διαδικαστική γνώση και είναι συνήθως οργανομένα ως μία βιβλιοθήκη κανόνων, με τη μορφή της κατάστασης – δράσης. Έχουν δηλαδή τη μορφή δομών IF-THEN.

Τα συστήματα κανόνων αποδείχθηκε ότι είναι ένας ισχυρός τρόπος αναπαράστασης δεξιοτήτων, μάθησης και επίλυσης προβλημάτων. Χρησιμοποιούνται συχνά σε εφαρμογές της τεχνητής νοημοσύνης και της έρευνας. Ένα κλασικό παράδειγμα ενός συστήματος βασισμένου σε κανόνες είναι το έμπειρο σύστημα σε συγκεκριμένους τομείς που χρησιμοποιεί κανόνες για να κάνει κρατήσεις ή επιλογές. Για παράδειγμα, ένα έμπειρο σύστημα μπορεί να βοηθήσει ένα γιατρό να επιλέξει τη σωστή διάγνωση βασιζόμενο σε μια συστάδα συμπτωμάτων, ή να επιλέξει τακτικές κινήσεις ώστε να παιχτεί ένα παιχνίδι.

Η ΒΑΣΗ ΚΑΝΟΝΩΝ

Η μηχανή κανόνων χρειάζεται ένα μέρος για την αποθήκευση των κανόνων. Η βάση κανόνων περιέχει όλους τους κανόνες που γνωρίζει το σύστημα.Οι κανόνες μπορούν να αποθηκεύονται ως απλές συμβολοσειρές (strings), αλλά συνήθως ένας μεταγλωτιστής κανόνων τους μετατρέπει σε μορφή που επεξεργάζονται αποδοτικότερα από τη μηχανή διεξαγωγής συμπερασμάτων. Στις περισσότερες μηχανές κανόνων, ο μεταγλωτιστής κανόνων κατασκευάζει μία σύνθετη, δεικτοδοτημένη δομή δεδομένων που ονομάζεται δίκτυο Rete.

Το δίκτυο Rete είναι μία δομή δεδομένων που επιταχύνει την επεξεργασία κανόνων. Επιπλέον, ένας μεταγλωτιστής κανόνων μπορεί να πραγματοποιεί αλλαγές βελτιστοποίησης στο δεξί και στο αριστερό μέρος των κανόνων για αποδοτικότερη εκτέλεση. Αναλόγως της μηχανής κανόνων, αυτές οι αλλαγές μπορεί να αποκρύπτονται από τον προγραμματιστή. Μερικές μηχανές κανόνων παρέχουν τη δυνατότητα (ή την απαίτηση) αποθήκευσης της βάσης κανόνων σε κάποια εξωτερική σχεσιακή βάση δεδομένων,ενώ κάποιες άλλες μηχανές διατηρούν τη βάση κανόνων ενσωματωμένη. Η αποθήκευση των κανόνων σε μια σχεσιακή βάση δεδομένων επιτρέπει την επιλογή των κανόνων που θα συμπεριληφθούν στο σύστημα βάσει ορισμένων κριτηρίων όπως η ημερομηνία, η ώρα και τα δικαιώματα χρήσης.



Εικόνα 5 : Η αρχιτεκτονική ενός τυπικού συστήματος κανόνων

ΤΥΠΟΙ ΣΥΣΤΗΜΑΤΩΝ ΚΑΝΟΝΩΝ

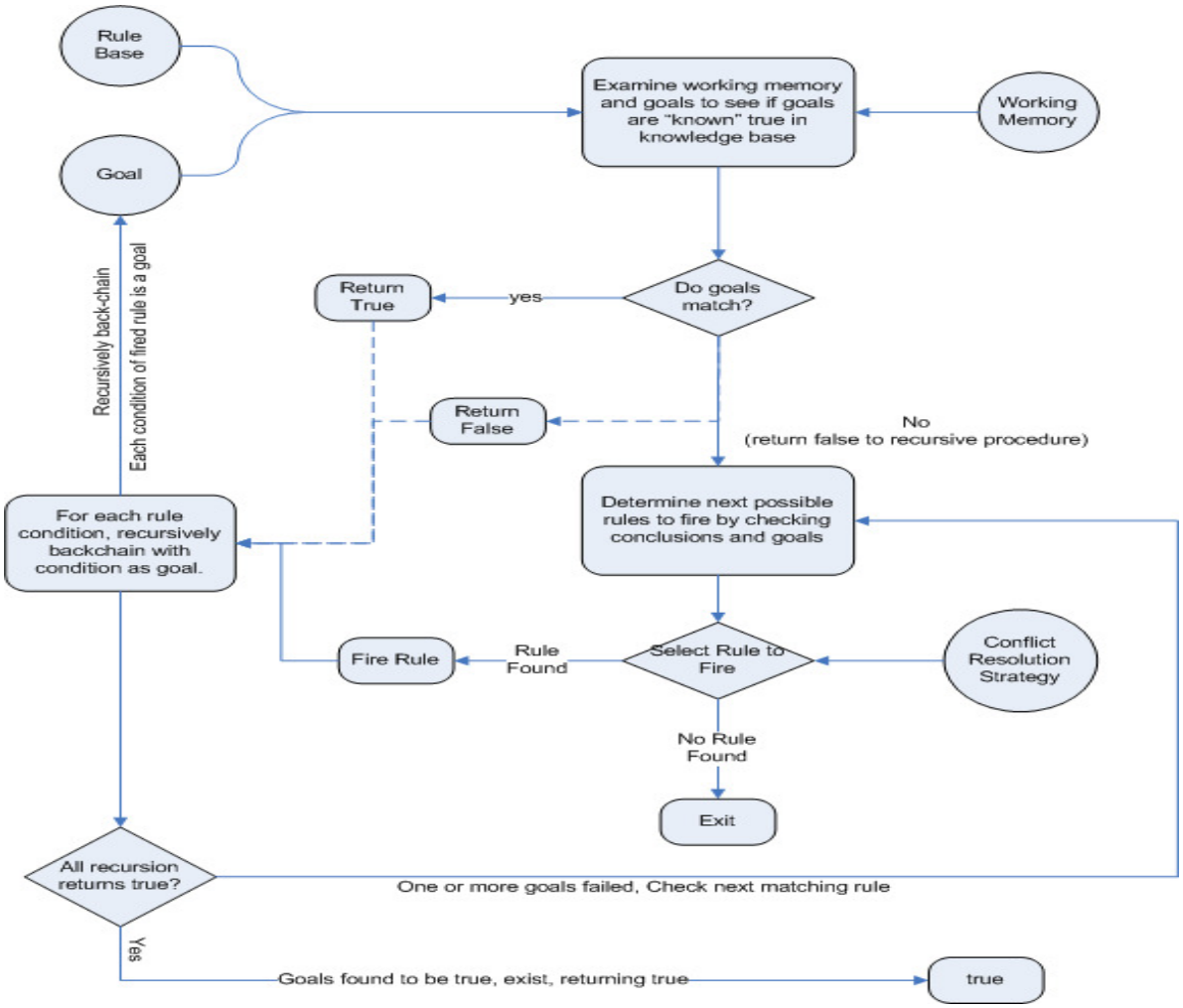
• Δίκτυα εξαγωγής συμπερασμάτων (Inference networks)
– Η βάση κανόνων μπορεί να παρασταθεί σαν ένα δίκτυο αλληλοσυνδεόμενων κανόνων και γεγονότων
– Οι σχέσεις μεταξύ κανόνων και γεγονότων είναι προκαθορισμένες
• Συστήματα ταιριάσματος προτύπων (Pattern-matching systems)
– Τα συμπεράσματα είναι πιό γενικά και μπορούν να θεωρηθούν σαν ένα σύνολο γεγονότων που μπορεί να ταιριάζουν ή όχι με διάφορα πρότυπα που αποτελούν τις υποθέσεις των κανόνων
– Τα πρότυπα περιέχουν μεταβλητές για τις οποίες αναζητούνται τιμές
– Οι σχέσεις μεταξύ κανόνων και γεγονότων σχηματοποιούνται κατά τη διαδικασία εξαγωγής συμπερασμάτων.

ΕΞΑΓΩΓΗ ΣΥΜΠΕΡΑΣΜΑΤΩΝ

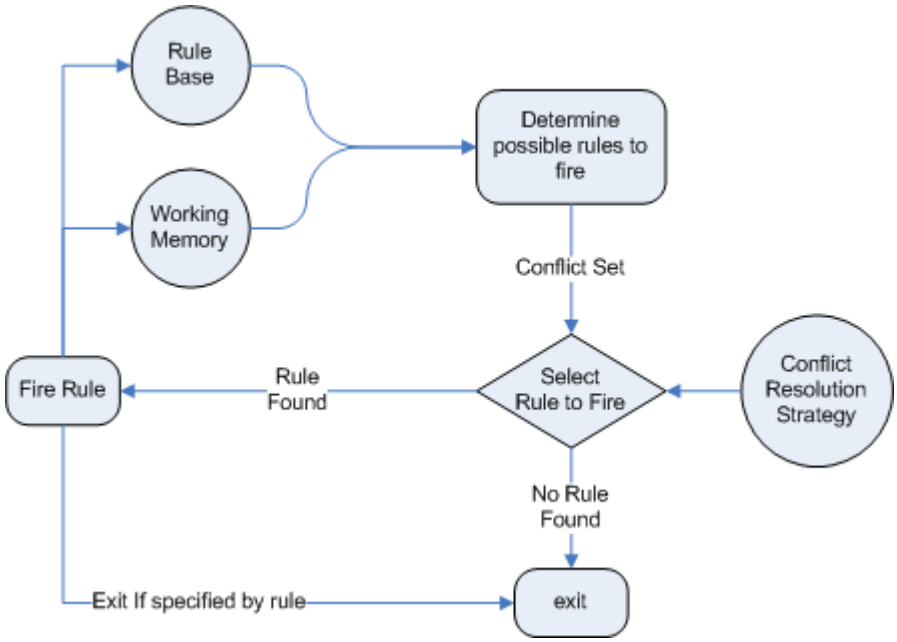
Σε ένα σύστημα βασισμένο σε κανόνες υπάρχουν δύο μέθοδοι λειτουργίας της μηχανής διεξαγωγής συμπερασμάτων. Αυτές είναι οι f**orward chaining** και **backward chaining** .

Κατά τη λειτουργία **forward chaining** μόλις εμφανιστούν τα απαραίτητα γεγονότα στη μνήμη θα ενεργοποιηθεί ο κανόνας που τα περιλαμβάνει στο αριστερό του μέρος, όμως ο προγραμματιστής είναι αυτός που πρέπει να καθορίσει τη σειρά με την οποία θα συγκεντρωθούν τα γεγονότα.

Κατά τη λειτουργία **backward chaining** η μηχανή προσπαθεί ενεργά να πυροδοτήσει έναν συγκεκριμένο κανόνα όταν αυτός είναι μερικώς ενεργοποιημένος, ζητώντας αυτόματα την προσθήκη των υπόλοιπων γεγονότων που περιλαμβάνονται στο αριστερό μέρος του κανόνα για την πλήρη ενεργοποίησή του.



Εικόνα 6: Διαδικασία Backward chaining



Εικόνα 7:Διαδικασία Forward chaining

ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΚΑΝΟΝΩΝ

• Ευελιξία-Τμηματικότητα (κάθε κανόνας είναι μια ξεχωριστή μονάδα γνώσης που μπορεί να προστεθεί, μεταβληθεί ή αφαιρεθεί ανεξάρτητα από τους άλλους κανόνες του συστήματος)

• Ομοιομορφία και απλότητα στην έκφραση της γνώσης

• Φυσικότητα έκφρασης

• Εύκολη παροχή εξηγήσεων

ΜΕΙΟΝΕΚΤΗΜΑΤΑ ΚΑΝΟΝΩΝ

• Κίνδυνος δημιουργίας ατέρμονων αλυσώσεων μεταξύ κανόνων και γεγονότων (κυρίως με την προσθήκη νέων κανόνων)

• Κίνδυνος δημιουργίας αντιφάσεων

• Μειωμένη αποδοτικότητα (σε μεγάλες ΒΚ)

• Αδιαφάνεια (στις σχέσεις μεταξύ των κανόνων)

• Δυσκολία κάλυψης πεδίων με πολλές παραμέτρους και ευαίσθητη εξάρτηση των συμπερασμάτων από αυτές

• Περιορισμένη εκφραστικότητα αναπαράστασης

Μορφές Κανόνων:	**Επεξήγηση:**
IF *συνθήκες* THEN *συμπέρασμα* **Συνεπαγωγικός κανόνας** **(Deduction)**	**Αν** οι *συνθήκες* αληθεύουν **τοτε** αληθεύει και το συμπέρασμα
IF *συνθήκες* THEN *ενέργειες διαχείρισης γνώσης* **Κανόνας Παραγωγής** **(Production)**	**Αν** οι *συνθήκες* αληθεύουν **τότε** διαχειρίσου την υπάρχουσα γνώση. (αλλαγή κατάστασης)
ON *συμβάν* IF *συνθήκες* THEN *ενέργειες* **Ενεργός (active) κανόνας**	**Όταν** συμβεί το *γεγονός (συμβάν)* **Αν** οι συνθήκες αληθεύουν **τότε** εκτέλεσε τις ενέργειες

1.2.5) ΠΛΑΙΣΙΑ

Η αναπαράσταση γνώσης με τη χρήση πλαισίων είναι ο πιο προσφιλής τρόπος αναπαράστασης γνώσης όσον αφορά δεδομένα και γεγονότα. Αναπτύχθηκε από τον Marvin Minsky για τη αναπαράσταση στερεότυπων καταστάσεων. Χρησιμοποιούνται για την αναπαράσταση των εννοιών και των αντικειμένων του προβλήματος,των χαρακτηριστικών που περιγράφουν τις έννοιες και τα αντικείμενα, των αλληλεξαρτήσεων ή σχέσεων μεταξύ τους.

Τα πλαίσια μπορούν να παρομοιαστούν με ένα γράφο εννοιών.Κάθε κόμβος (node) του γράφου εκφράζει μία έννοια και μπορεί να είναι :κλάση αντικειμένων (class) ή αντικείμενο (object). Κάθε κόμβος έχει προσκολημένο όνομα και σχισμές (slots) που εκφράζουν ιδιότητες - χαρακτηριστικά. Κάθε δεσμός (link) του γράφου μπορεί να είναι μία ιεραρχική συσχέτιση.

Παράδειγμα:

Πλαίσιο: φρούτο (κλάση)

Θέση 1: φαγώσιμο (default yes)

Θέση 2: βασική διατροφή (default yes)

Θέση 3: εύπεπτο (default no)

Πλαίσιο: μήλο (κλάση)

Είδος: φρούτο

Θέση εποχή: χειμερινό

Θέση γεύση: εύγευστο

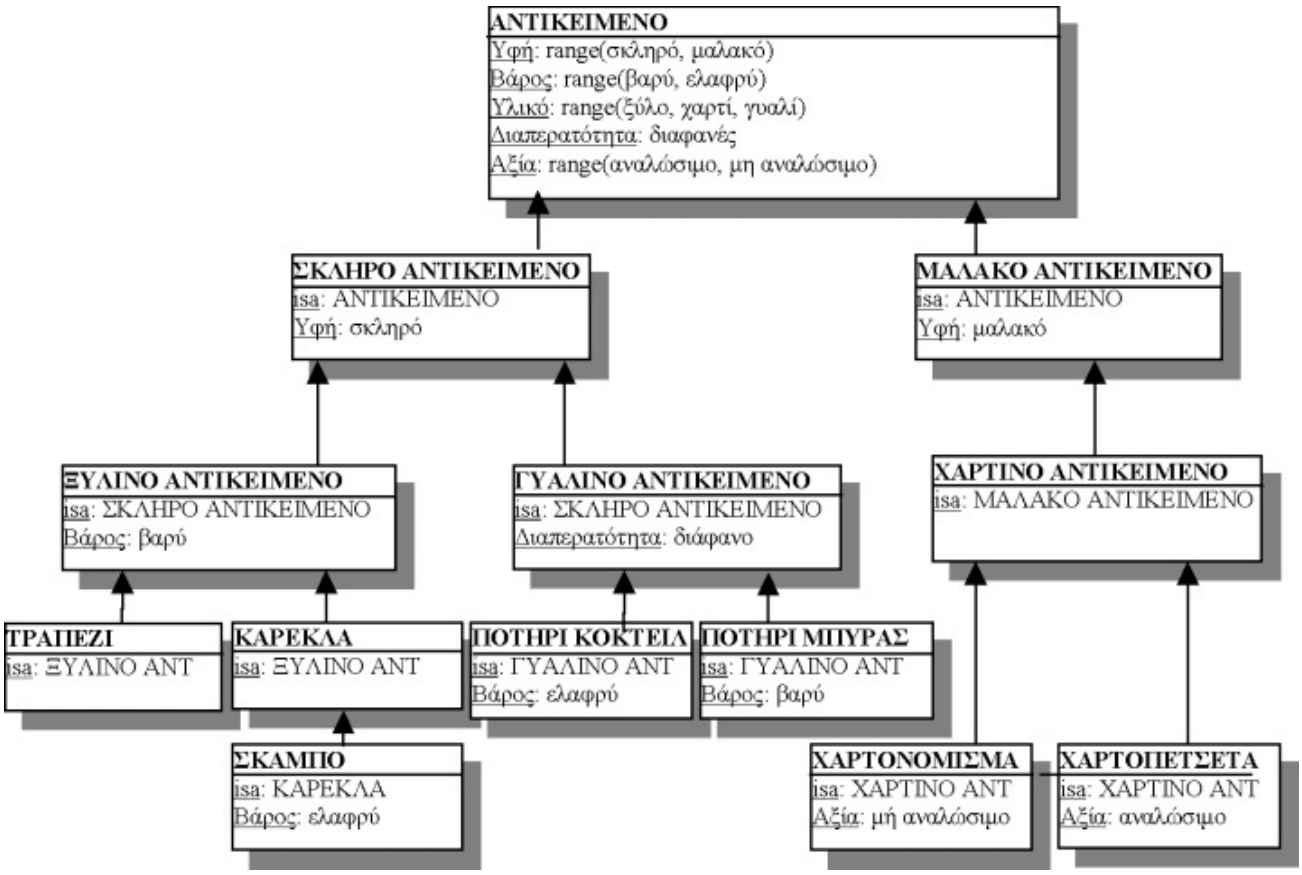
Θέση χρώμα: κόκκινο

Πλαίσιο: GOLDEN

είναι: μήλο (στιγμιότυπο)

Θέση χρώμα: κίτρινο

Θέση 3: εύπεπτο: yes



Εικόνα 8: Αναπαράσταση διάφορων αντικειμένων με τη χρήση πλαισίων

1.2.6) ΔΗΛΩΤΙΚΗ ΑΝΑΠΑΡΑΣΤΑΣΗ

Η δηλωτική γνώση ή προτασιακή γνώση, είναι ο τύπος της γνώσης που, από τη φύση της, εκφράζεται με δηλωτικές προτάσεις ή ενδεικτικές προτάσεις. Αυτό ξεχωρίζει την δηλωτική γνώση από αυτό που είναι κοινώς γνωστό ως "τεχνογνωσία", ή διαδικαστική γνώση (γνώση του πώς, και ιδιαίτερα με τον καλύτερο τρόπο, για να εκτελέσει κάποια εργασία), και "γνωρίζοντας", ή γνώση από εξοικείωση (η γνώση της ύπαρξη του κάτι).

1.2.7) ΠΙΘΑΝΟΛΟΓΙΚΗ ΑΝΑΠΑΡΑΣΤΑΣΗ

Η Θεωρία Πιθανοτήτων παρέχει ένα πλαίσιο για την αναπαράσταση και τη συλλογιστική υπό συνθήκες αβεβαιότητας. Στα πιθανολογικά συστήματα συλλογιστικής, ο κόσμος χωρίζεται σε μια σειρά τυχαίων μεταβλητών που αντιπροσωπεύουν τις διαφορετικές πτυχές του κόσμου που σχετίζονται. Κάθε τυχαία μεταβλητή έχει ένα πεδίο τιμών που μπορεί να αναλάβει. Η εκχώρηση των τιμών σε όλες τις τυχαίες μεταβλητές είναι ένα ατομικό γεγονός που αντιπροσωπεύει μία κατάσταση του κόσμου.

Το σύνολο όλων των ατομικών γεγονότων είναι αμοιβαία αποκλειόμενο και διεξοδικό. Σε κάθε ατομικό γεγονός ανατίθεται μια πιθανότητα που αντιπροσωπεύει το βαθμό των πεποιθήσεων του κόσμου που είναι σε αυτή τη συγκεκριμένη κατάσταση, και συνοψίζει την αβεβαιότητα που οφείλεται στην άγνοια. Μια κοινή πιθανότητα κατανομής αναθέτει πιθανότητες σε όλα τα ατομικά γεγονότα και πρέπει να τις συνοψίσει σε μία.Από την κοινή κατανομή πιθανότητας, διάφορες μορφές συλλογισμού και συμπερασμού μπορούν να εκτελεστούν χρησιμοποιώντας πιθανότητα λογισμού.

ΚΕΦΑΛΑΙΟ 2

ΤΑΙΡΙΑΣΜΑ (MATCHING)

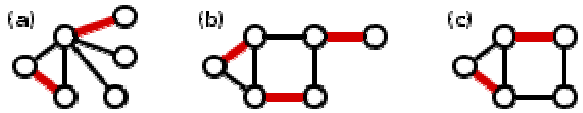
2.1) ΟΡΙΣΜΟΣ

Ταίριασμα Τ ενός γραφήματος είναι ένα υποσύνολο των πλευρών του, που δεν περιέχει προσκείμενες πλευρές. Δηλαδή, δύο οποιεσδήποτε πλευρές του αποκλείεται να συντρέχουν στην ίδια κορυφή. Αυτό σημαίνει ότι οι κορυφές εκείνες που είναι άκρα πλευρών του Τ, είναι ταιριασμένες μεταξύ τους ανά δύο μέσω πλευράς του Τ και αποκλείεται μια κορυφή να έχει πάνω από ένα ταίρι. Όσο πιο πολλές πλευρές έχει το Τ τόσο «μεγαλύτερο» λέμε ότι είναι.

Αυτό βέβαια δεν σημαίνει ότι αν το T_1 είναι μεγαλύτερο του T_2 τότε το T_2 είναι υποσύνολο του T_1. Αν δεν υπάρχει ταίριασμα πιο μεγάλο από το Τ τότε λέμε ότι το Τ είναι «μέγιστο», δηλαδή έχουν βρεθεί ταίρια για όσο το δυνατό πιο πολλές κορυφές.

Ένα **τέλειο ταίριασμα** είναι ένα ταίριασμα που αντιστοιχεί όλες τις κορυφές του γραφήματος. Δηλαδή, κάθε κορυφή του γραφήματος προσπίπτει σε ακριβώς μία ακμή της αντιστοίχισης. Κάθε τέλειο ταίριασμα είναι μέγιστο.

Ένα τέλειο ταίριασμα είναι επίσης ένα ελάχιστου μεγέθους κάλυμμα άκρης. Έτσι, ν (G) ≤ p(G), δηλαδή, το μέγεθος της μέγιστης αντιστοίχισης δεν είναι μεγαλύτερο από το μέγεθος μίας ελάχιστης κάλυψης άκρης.



Εικόνα 9: Στο σχήμα b βλέπουμε ένα τέλειο ταίριασμα

2.2) ΘΕΩΡΗΜΑ HALL

Έστω ένας διγράφος $G(V_1 \cup V_2, E)$. Το σύνολο V_1 μπορεί να αντιστοιχηθεί σε ένα υποσύνολο του V_2 αν και μόνο αν $N(S) \geq |S|$ δια πάν S υποσύνολο του V_1.

Το θεώρημα Hall (ή θεώρημα του γάμου) αποδείχθηκε από τον Philip Hall (1935), είναι ένα θεώρημα με δύο ισοδύναμα τυποποιήσεις:

-Την μαθηματική συνδυαστική διαμόρφωση που ασχολείται με μια συλλογή από πεπερασμένα σύνολα και δίνει μια ικανή και αναγκαία συνθήκη για να είμαστε σε θέση να επιλέξουμε ένα ξεχωριστό στοιχείο από κάθε σύνολο και

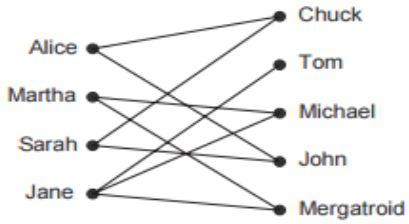
-Τη θεωρητική διατύπωση γραφημάτων που ασχολείται με ένα διμερές γράφημα και δίνει μια αναγκαία και ικανή συνθήκη για την εξεύρεση μιας αντιστοίχισης που καλύπτει τουλάχιστον μία πλευρά του γραφήματος.

Παράδειγμα

Μια τάξη περιέχει μερικά κορίτσια και τα μερικά αγόρια. Σε κάθε κορίτσι αρέσουν κάποια αγόρια και δεν του αρέσουν κάποια άλλα. Κάτω από ποιες συνθήκες μπορεί κάθε κορίτσι να συνδυαστεί με ένα αγόρι που του αρέσει;

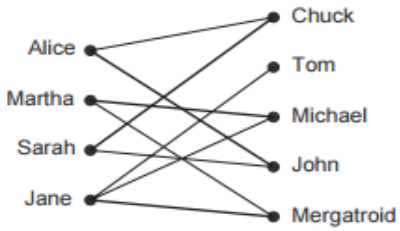
Μπορούμε να μοντελοποιήσουμε την κατάσταση με ένα διμερές γράφημα. Δημιουργούμε μια κορυφή στα αριστερά για κάθε κορίτσι και μια κορυφή στα δεξιά για κάθε αγόρι. Αν σε ένα κορίτσι αρέσει ένα αγόρι, βάζουμε μια άκρη μεταξύ τους.

Για παράδειγμα, το παρακάτω γράφημα:



Όσον αφορά το γράφημα, ο στόχος μας είναι να βρούμε μια αντίστοιχη για τα κορίτσια που να είναι, ένα υποσύνολο των ακμών έτσι ώστε ακριβώς μια άκρη να προσπίπτει σε κάθε κορίτσι και το πολύ μια ακμή να προσπίπτει σε κάθε αγόρι.

Για παράδειγμα, εδώ είναι ένας δυνατός συνδυασμός για τα κορίτσια:



Θα δηλώσουμε και θα αποδείξουμε το θεώρημα του Hall, χρησιμοποιώντας την ορολογία κορίτσι-συμπαθεί-αγόρι. Ορίζουμε το σετ των αγοριών που άρεσουν σε ένα δεδομένο σύνολο των κοριτσιών να αποτελείται από όλα τα αγόρια που αρέσουν σε τουλάχιστον ένα από αυτά τα κορίτσια.

Για παράδειγμα, το σύνολο των αγοριών που αρέσουν στη Martha και στη Jane αποτελείται από τον Tom, τον Michael, και Mergatroid. Ικανή και αναγκαία συνθήκη για λύση στο πρόβλημα των γάμων m αγοριών και w > m κοριτσιών είναι κάθε σύνολο από t αγόρια να γνωρίζουν συλλογικά

τουλάχιστον t κορίτσια (όπου $1 \leq t \leq m$)

Για παράδειγμα, δεν μπορούμε να βρούμε μια αντιστοίχιση στην περίπτωση που σε 4 κορίτσια αρέσουν μόνο 3 αγόρια.Το Θεώρημα του Hall λέει ότι αυτή η απαραίτητη προϋπόθεση είναι πράγματι επαρκής

Θεώρημα 1: Μια αντιστοίχιση για ένα σύνολο κοριτσιών G με ένα σύνολο αγοριών B μπορεί να βρεθεί αν και μόνο αν ισχύει η προϋπόθεση γάμου.

Απόδειξη:

Κατ 'αρχάς, ας υποθέσουμε ότι υπάρχει μια αντιστοίχιση που δείχνει ότι η συνθήκη του γάμου ισχύει. Θεωρήστε ένα αυθαίρετο υποσύνολο κοριτσιών. Σε κάθε κορίτσι αρέσει τουλάχιστον ένα αγόρι με το οποίο έχει αντιστοιχηθεί. Ως εκ τούτου, κάθε υποσύνολο των κοριτσιών θέλει τουλάχιστον ένα τόσο μεγάλο σύνολο των αγοριών. Έτσι, η συνθήκη γάμου ισχύει.

Στη συνέχεια, ας υποθέσουμε ότι η συνθήκη του γάμου ισχύει και θα δείξουμε ότι υπάρχει αντιστοίχιση.Χρησιμοποιούμε ισχυρή επαγωγή στο $|G|$, τον αριθμό των κοριτσιών. Αν $|G| = 1$, τότε αυτή η προϋπόθεση γάμου συνεπάγεται ότι στο μοναδικό κορίτσι αρέσει τουλάχιστον ένα αγόρι, και έτσι υπάρχει μία αντιστοίχιση. Τώρα υποθέτουμε ότι $|G| \geq 2$. Υπάρχουν δύο δυνατότητες:

1. Σε κάθε κατάλληλο υποσύνολο κοριτσιών αρέσει αυστηρά ένα μεγαλύτερο σύνολο αγοριών. Σε αυτή την περίπτωση, έχουμε κάποιο περιθώριο: Συνδυάζουμε ένα αυθαίρετο κορίτσι με ένα αγόρι που του αρέσει και στέλνουμε και τα δύο μακριά. Η κατάσταση του γάμου εξακολουθεί να ισχύει για τα υπόλοιπα αγόρια και κορίτσια, ώστε να μπορούμε να ταιριάξουμε το υπόλοιπο των κοριτσιών με επαγωγή.

2. Σε ένα κατάλληλο υποσύνολο κοριτσιών $\mathbf{X \subset G}$ αρέσει ένα σύνολο ίσου μεγέθους αγοριών $\mathbf{Y \subset B}$. Ταιριάζουμε τα κορίτσια στο X με τα αγόρια στο Y από την επαγωγή και τα στέλνουμε όλα μακριά. Θα δείξουμε ότι η συνθήκη του γάμου ισχύει και για τα υπόλοιπα αγόρια και κορίτσια, και έτσι μπορούμε να ταιριάξουμε τα υπόλοιπα κορίτσια με επαγωγή επίσης.Για το σκοπό αυτό, θα εξετάσουμε ένα αυθαίρετο υποσύνολο των υπόλοιπων κοριτσιών $\mathbf{X' \subseteq G - X}$, και αφήνουμε το Y να είναι το σύνολο των υπόλοιπων αγοριών που τους αρέσει. Πρέπει να δείξουμε ότι $|\mathbf{X'}| \leq |\mathbf{Y'}|$. Αρχικά, στο συνδυασμένο σύνολο των κοριτσιών $X \cup X'$ άρεσε το σύνολο των αγοριών $\mathbf{Y \cup Y'}$.

Έτσι, από τη προϋπόθεση γάμου, γνωρίζουμε ότι:$|\mathbf{X \cup X'}| \leq |\mathbf{Y \cup Y'}|$.
Στείλαμε μακριά $|X|$ κορίτσια από το σύνολο στα αριστερά (αφήνωντας X') και διώξαμε ίσο αριθμό αγοριών από το σύνολο στα δεξιά (αφήνωντας Y'). Ως εκ τούτου, θα πρέπει να ισχύει ότι$|\mathbf{X'}| \leq |\mathbf{Y'}|$ όπως αξιώνεται.

Και στις δύο περιπτώσεις, υπάρχει μια αντίστοιχη για τα κορίτσια. Το θεώρημα προκύπτει από την επαγωγή. Υπάρχει ένας αποδοτικός αλγόριθμος για την εύρεση μιας αντιστοίχισης ενός διμερούς γραφήματος, εαν αυτό υπάρχει. Έτσι, εάν ένα πρόβλημα μπορεί να μειωθεί με εξεύρεση αντιστοίχισης, το πρόβλημα ουσιαστικά επιλύεται από μια υπολογιστική προοπτική.

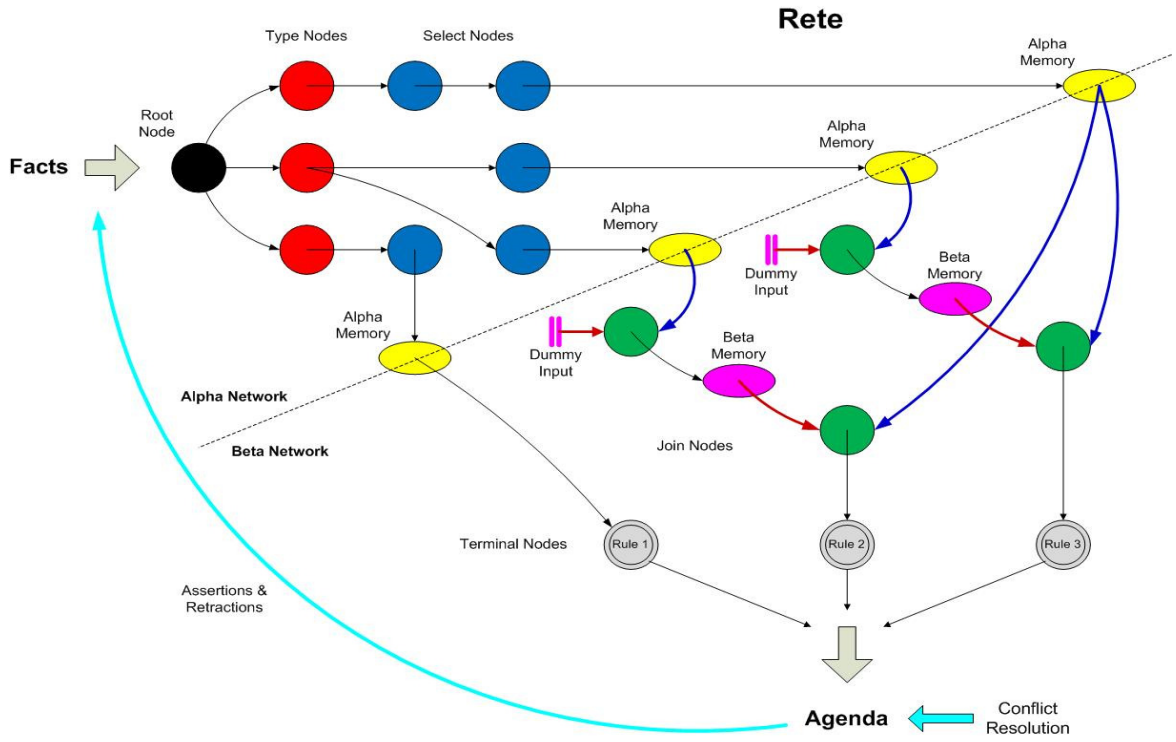
2.3) RETE ALGORITHM

Ο αλγόριθμος Rete δημιουργήθηκε και εκδόθηκε από τον καθηγητή Charles Forgy στην διδακτορική του διατριβή το 1978-79. Είναι ένας αποτελεσματικός αλγόριθμος αντιστοίχισης προτύπων για την υλοποίηση συστημάτων κανόνων παραγωγής. Είναι σχεδιασμένος να "θυσιάζει" μνήμη με σκοπό την αυξημένη ταχύτητα. Ο αλγόριθμος Rete παρουσιάζει τα εξής χαρακτηριστικά:

1) Μειώνει ή εξαλείφει ορισμένους τύπους πλεονασμού με τη δυνατότητα διαμοίρασης κόμβων.
2) Αποθηκεύει ενδιάμεσες τμηματικές αντιστοιχίες όταν εκτελεί ενώσεις μεταξύ διαφορετικών τύπων γεγονότων.
3) Παρέχει τη δυνατότητα αποτελεσματικής αφαίρεσης στοιχείων μνήμης,όταν τα γεγονότα αφαιρούνται από τη λειτουργική μνήμη.

Ο αλγόριθμος Rete μπορεί να χωριστεί σε δύο μέρη: Μεταγλώττιση και εκτέλεση. Κατά τη φάση της μεταγλώττισης ο αλγόριθμος κατανέμει τους κανόνες στη μνήμη εργασίας με τέτοιο τρόπο ώστε να δημιουργήσει ένα αποτελεσματικό δίκτυο. Η ιδέα του δικτύου είναι να φιλτράρει τα δεδομένα όσο προωθούνται στους επόμενους κόμβους του δικτύου. Οι αρχικοί κόμβοι θα έχουν πολλές αντιστοιχίσεις και όσο προχωρούν στο δίκτυο θα γίνονται λιγότερες ώσπου να καταλήξουν στο τέλος του δικτύου στους τελικούς κόμβους.

Για αυτό το λόγο ο Rete χαρακτηρίζεται ως αλγόριθμος καθοδηγούμενος από τα δεδομένα. Γιατί ξεκινά με πολλά δεδομένα τα οποία προσπαθεί μα αντιστοιχίσει σε κάποιον από τους ελάχιστους τελικούς κόμβους. Ένα Rete δίκτυο αποτυπώνεται στο σχήμα που ακολουθεί.



Εικόνα 10: Απεικονίζονται τα δίκτυα alpha και beta και οι βασικοί τύποι κόμβων που περιέχονται σε ένα δίτυο Rete. Οι πράσινοι κόμβοι είναι οι κόμβοι που θα ενταχθούν

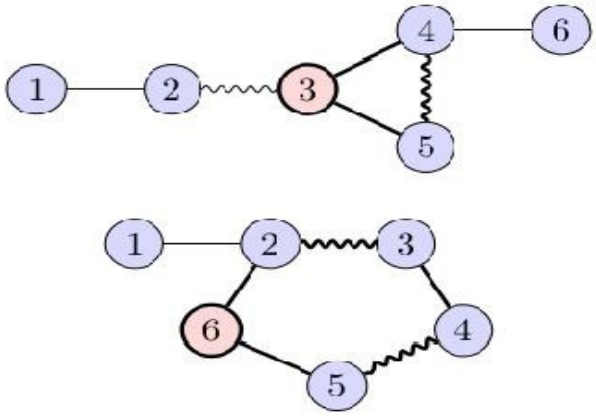
ΚΕΦΑΛΑΙΟ 3

ΑΛΓΟΡΙΘΜΟΙ ΤΑΙΡΙΑΣΜΑΤΟΣ

3.1) BLOSSOM ALGORITHM

Blossom (Άνθος)

Ορισμός:
Ένα άνθος B σε ένα (G,M) είναι ένας περίεργος κύκλος με μία μοναδική εκτεθειμένη κορυφή (τη βάση) σε ένα M∩B.



Ο blossom είναι ένας αλγόριθμος στη θεωρία γραφημάτων για την κατασκευή μέγιστου ταιριάσματος σε γραφήματα. Ο αλγόριθμος αναπτύχθηκε από τον Jack Edmond's το 1961 και δημοσιεύτηκε το 1965. Δεδομένου ενός γενικού γραφήματος G = (V, E), ο αλγόριθμος βρίσκει ένα ταίριασμα M τέτοιο ώστε κάθε κορυφή στο V να προσπίπτει με το πολύ μία ακμή στο M και το |M| να μεγιστοποιείται.

Η γενική ιδέα του αλγορίθμου:

Ένα γράφημα G και μια αντιστοίχιση M του G δίνεται.

- Ξεκινήστε από την ανάπτυξη δέντρων από κάθε εκτεθειμένη κορυφή.

- Συνεχίστε μέχρι να βρεθεί μια εκτεθειμένη κορυφή ή ένα άνθος ή μέχρι το δέντρο να μη μπορεί να αυξηθεί περαιτέρω.

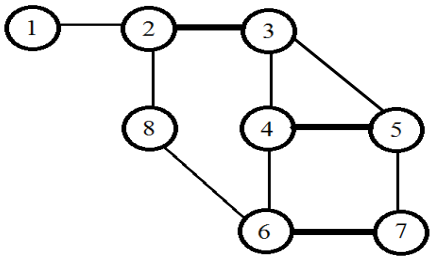
- Αν βρεθεί ένα άνθος, επισυνάψτε τις άκρες του άνθους στο G και ξεκινήστε από την αρχή με το νέο γράφημα.

- Αν μια εκτεθειμένη κορυφή βρεθεί τότε έχουμε βρεί ένα αυξητικό μονοπάτι

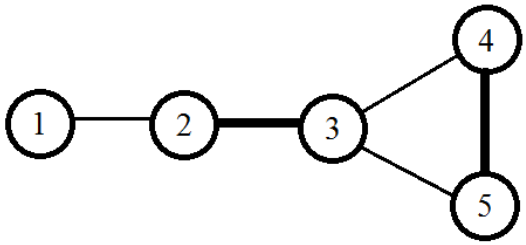
- Αντιστρέψτε τη συρρίκνωση του άνθους και αυξήστε κατά μήκος της αυξητικής διαδρομής για να λάβετε τη νέα αντιστοίχιση M .

- Επαναλάβετε με το ταίριασμα M .

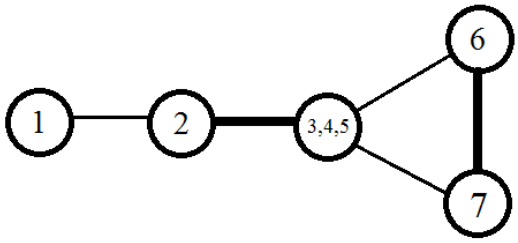
Παράδειγμα:



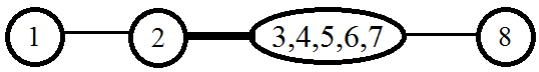
Ξεκινήστε το αναπτυσσόμενο δέντρο μέχρι το άνθος ή η εκτεθειμένη κορυφή να βρεθεί.



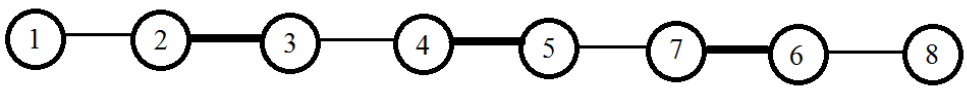
Επισυνάψτε το άνθος στο αρχικό γράφημα και ξεκινήστε πάλι το αναπτυσσόμενο δέντρο



Επισυνάψτε το άνθος στο αρχικό γράφημα και ξεκινήστε πάλι το αναπτυσσόμενο δέντρο



Το αυξητικό μονοπάτι βρέθηκε.Αντιστρέψτε τις συρρικνώσεις των άνθων και αποκτήστε το αυξητικό μονοπάτι στο G.



ΤΟ ΠΡΟΒΛΗΜΑ ΜΕΓΙΣΤΗΣ ΡΟΗΣ

Το πρόβλημα μέγιστης ροής διατυπώθηκε για πρώτη φορά το 1954 από τον TE Harris και τον FS Ross ως ένα απλοποιημένο μοντέλο της ροής της σοβιετικής σιδηροδρομικής κυκλοφορίας. Το 1955, ο Lester R. Ford, Jr. και R. Ντέλμπερτ Fulkerson δημιούργησαν τον πρώτο γνωστό αλγόριθμο, τον αλγόριθμο Ford-Fulkerson.

Τα προβλήματα μέγιστης ροής τα συναντάμε συχνά σε δίκτυα μεταφοράς (transport networks). Ένα δίκτυο μεταφοράς είναι ένα γενικό μοντέλο με διαδρομές μεταφοράς υλικών, ρευστών, αυτοκινήτων, κεφαλαίων, ηλεκτρικού φορτίου, κλπ., από ένα κόμβο-αφετηρία σε ένα κόμβο-προορισμό. Οι κλάδοι ενός δικτύου μεταφοράς αντιπροσωπεύουν τη μέγιστη χωρητικότητα της διαδρομής από ένα κόμβο σε ένα άλλο την οποία δεν μπορεί να υπερβεί η μεταφερόμενη ποσότητα.

Για παράδειγμα μια επιχείρηση προσπαθεί να μεταφέρει όλα τα προϊόντα της από το εργοστάσιο στο κεντρικό κατάστημα. Αν υπάρχουν διαφορετικοί δρόμοι και μέσα μεταφοράς πως μπορεί να τα συνδυάσει ώστε να μεταφέρει το μεγαλύτερη δυνατή ποσότητα το συντομότερο δυνατό;

3.2) FORD-FULKERSON

Ο πρώτος αλγόριθμος που δημιουργήθηκε για το πρόβλημα της μέγιστης ροής είναι ο αλγόριθμος των Ford και Fulkerson. Ο αλγόριθμος έκανε, για πρώτη φορά, χρήση της τεχνικής του αυξητικού μονοπατιού και αποτέλεσε τη βάση για πιο αποδοτικές λύσεις.

Η ιδέα πίσω από τον αλγόριθμο είναι αρκετά απλή: Όσο υπάρχει ένα μονοπάτι από την πηγή (εκκίνηση κόμβου) στον απορροφητή (τέλος κόμβου), με διαθέσιμη χωρητικότητα σε όλες τις άκρες του μονοπατιού, στέλνουμε ροή κατά μήκος ενός από αυτά τα μονοπάτια. Στη συνέχεια θα βρούμε ένα άλλο μονοπάτι, και ούτω καθεξής.
Ένα μονοπάτι με διαθέσιμη χωρητικότητα ονομάζεται αυξητικό μονοπάτι. Αν οι χωρητικότητες είναι όλες ακέραιοι αριθμοί, τότε ο χρόνος είναι O (m | f |). Αυτό συμβαίνει επειδή η εύρεση μίας αυξανόμενης διαδρομής και η ενημέρωση της ροής παίρνει O (m) χρόνο, κάθε αυξητικό μονοπάτι που θα βρεθεί πρέπει να αυξήσει τη ροή από ένα ακέραιο που θα ισούται τουλάχιστον με 1.

Παρακάτω βλέπουμε τη λειτουργία του αλγορίθμου ford-fulkerson σε ψευδοκώδικα:

$f(u,v) = 0$ για κάθε $u,v \in G$
while υπάρχει αυξητικό μονοπατι p **do**
$c = \min c\ f(u,v)$, για κάθε $(u,v) \in p$
για κάθε $(u,v) \in p$ έχουμε $f(u,v) = f(u,v) + c$ και $f(v,u) = -f(u,v)$
end while

Επίσης θα παρουσιάσουμε τη λειτουργία του αλγορίθμου και σε κώδικα c++.

```
1  c++ πρόγραμμα για την εκτέλεση του αλγορίθμου ford-fulkerson
2  #include <iostream>
3  #include <limits.h>
4  #include <string.h>
5  #include <queue>
6  using namespace std;
7
8  //ο αριθμός των κορυφών στο δεδομένο γράφημα//
9  #define V 6
10 /*Επιστρέφει true αν υπάρχει διαδρομή από την πηγή 's' στον "νεροχύτη" 't' στο υπολλειπόμενο
11 γράφημα.Επίσης γεμίζει την parent[] για να αποθηκεύσει τη διαδρομή*/
12
13 bool bfs(int rGraph[V][V], int s, int t, int parent[])
14 {
15 Δημιουργεί μία σειρά που έχουμε επισκεφτεί και επισημαίνει όλες τις κορυφές σαν να μην έχουν
16     bool visited[V];                                                    επισκεφτεί
17     memset(visited, 0, sizeof(visited));
18 //Δημιουργεί μία ουρά,τοποθετεί στην ουρά κορυφή προέλευσης και σημειώνει την κορυφή
19 προέλευσης ότι την έχουμε επισκεφτεί//
20
21     queue <int> q;
22     q.push(s);
23     visited[s] = true;
24     parent[s] = -1;
25
```

```
    //BFS(αναζήτηση πρώτα κατά πλάτος) βρόγχος//
26
27      while (!q.empty())
28      {
29          int u = q.front();
30          q.pop();
31
32          for (int v=0; v<V; v++)
33          {
34              if (visited[v]==false && rGraph[u][v] > 0)
35              {
36                  q.push(v);
37                  parent[v] = u;
38                  visited[v] = true;
39              }
40          }
41      }
42
43    //Αν φτάσαμε στο νεροχύτη BFS ξεκινώντας από την πηγή τότε επιστρέφει 'true' διαφορετικά 'false'//
44
45      return (visited[t] == true);
46  }
47
48  //Επιστρέφει τη μέγιστη ροή από το s στο t στο συγκεκριμένο γράφημα//
49  int fordFulkerson(int graph[V][V], int s, int t)
50  {
51      int u, v;
52  //Δημιουργεί ένα υπολειπόμενο γράφημα και γεμίζει το υπολειπόμενο γράφημα με τις δεδομένες
53  χωρητικότητες στον αρχικό γράφο ως χωρητικότητες στο υπολειπόμενο γράφημα//
54
55
56      int rGraph[V][V]; //Υπολειπόμενο γράφημα όπου το rGraph[i][j] υποδεικνύει την υπολειπόμενη
57                        χωρητικότητα της ακμής από το i sto j(αν υπάρχει ακμή).Αν το rGraph[i][j] είναι
58                        0 τότε δεν υπάρχει//
59      for (u = 0; u < V; u++)
60          for (v = 0; v < V; v++)
61              rGraph[u][v] = graph[u][v];
62
```

```
63      int parent[V]; //Αυτή η γραμμή είναι γεμάτη από BFS και για να αποθηκεύσει το μονοπάτι//
64
65      int max_flow = 0; //Δεν υπάρχει ροή αρχικά//
66
67     //Αυξάνει τη ροή ενώ υπάρχει μονοπάτι από την πηγή προς τον "νεροχύτη"//
68      while (bfs(rGraph, s, t, parent))
69      {
70         //Βρίσκει την ελάχιστη υπολειπόμενη χωρητικότητα των ακμών κατά μήκος της διαδρομής
71         που είναι γεμάτη με BFS//
72
73          int path_flow = INT_MAX;
74          for (v=t; v!=s; v=parent[v])
75          {
76              u = parent[v];
77              path_flow = min(path_flow, rGraph[u][v]);
78          }
79
80        //Ενημερώνει τις υπολειπόμενες χωρητικότητες των ακμών και αντιστρέφει τις ακμές κατά μήκος της
81        διαδρομής//
82          for (v=t; v != s; v=parent[v])
83          {
84              u = parent[v];
85              rGraph[u][v] -= path_flow;
86              rGraph[v][u] += path_flow;
87          }
88
89        //Προσθέτει τη ροή της διαδρομής στη συνολική ροή//
90          max_flow += path_flow;
91      }
92
93     //Επιστρέφει τη συνολική ροή//
94      return max_flow;
95  }
96
```

```
97  //πρόγραμμα οδήγησης για τη δοκιμή των παραπάνω λειτουργιών//
98  int main()
99  {
100 //Ας δημιουργήσουμε ένα γράφημα που φαίνεται στο παραπάνω παράδειγμα//
101     int graph[V][V] = { {0, 16, 13, 0, 0, 0},
102                         {0, 0, 10, 12, 0, 0},
103                         {0, 4, 0, 0, 14, 0},
104                         {0, 0, 9, 0, 0, 20},
105                         {0, 0, 0, 7, 0, 4},
106                         {0, 0, 0, 0, 0, 0}
107                       };
108
109     cout << "The maximum possible flow is " << fordFulkerson(graph, 0, 5);
110
111     return 0;
112 }
113
```

Η μέθοδος **Ford-Fulkerson** είναι βασισμένη σε 3 βασικές ιδέες:

1) Τα υπολειπόμενα δίκτυα(residual networks)

2) Τα αυξητικά μονοπάτια(augmenting paths)

3) περικοπές(cuts)

ΔΙΚΤΥΟ ΡΟΗΣ

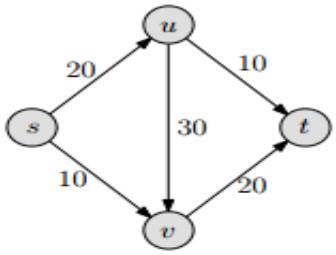
Στη θεωρία των γραφημάτων,ένα δίκτυο ροής (επίσης γνωστό ως ένα δίκτυο μεταφοράς) είναι ένα κατευθυνόμενο γράφημα όπου κάθε ακμή έχει χωρητικότητα και κάθε ακμή λαμβάνει μια ροή. Η ποσότητα της ροής σε μια ακμή δεν μπορεί να υπερβαίνει τη χωρητικότητα της ακμής. Συχνά στην επιχειρησιακή έρευνα, ένα κατευθυνόμενο γράφημα ονομάζεται δίκτυο.

Το δίκτυο ροής (**flow network**) είναι ένα κατευθυνόμενο γράφημα G = (V, E) με τα ακόλουθα χαρακτηριστικά:

• Με κάθε ακμή e ∈E είναι συσχετισμένη μια χωρητικότητα ce ≥ 0.

• Υπάρχει ένας μόνο κόμβος προέλευσης s ∈V.

• Υπάρχει ένας μόνο κόμβος απόληξης t ∈V.



Εικόνα 11: Παράδειγμα δικτύου ροής με

• προέλευση **s**

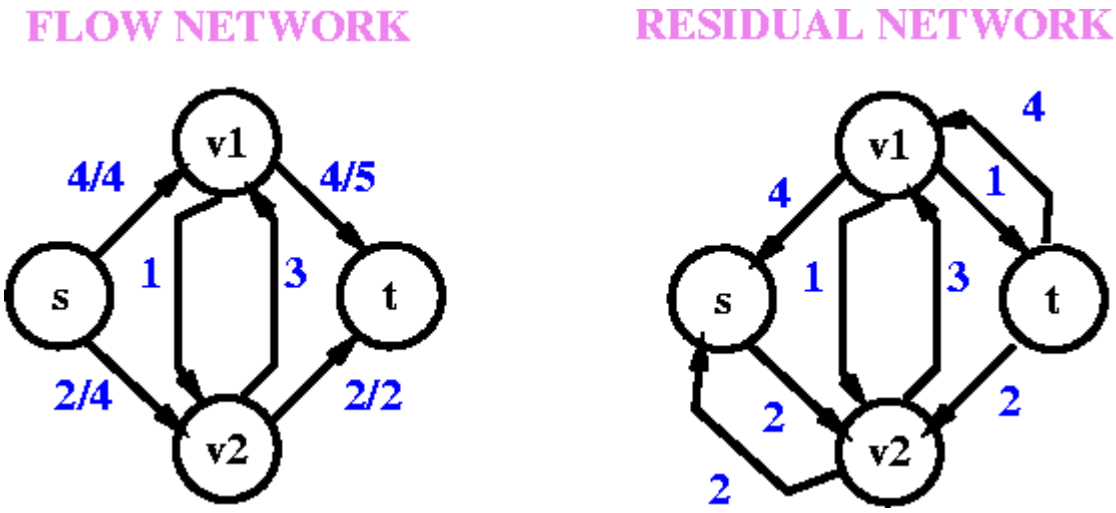
• απόληξη **t** και

• χωρητικότητες δίπλα στις ακμές.

ΥΠΟΛΕΙΜΜΑΤΙΚΌ ΔΙΚΤΥΟ

Το υπολειπόμενο δίκτυο είναι ένα γράφημα με τις ίδιες κορυφές, όπως το δίκτυο ροής, όμως οι άκρες του συλλαμβάνουν την πλεονάζουσα χωρητικότητα, που είναι το ποσό της πρόσθετης ροής που μπορούμε να στείλουμε μέσω αυτής πριν από τον κορεσμό της ακμής. Η υπολειπόμενη χωρητικότητα μίας ακμής είναι **cf (u, v) = c (u, v) - f (u, v).**

Αυτό ορίζει ένα υπολοιπόμενο δίκτυο που συμβολίζεται ως **GF (V, EF)**, δίνοντας το ποσό της διαθέσιμης χωρητικότητας. Μπορεί να υπάρχει ένα μονοπάτι από το u στο v στο υπολειπόμενο δίκτυο, έστω και αν δεν υπάρχει μονοπάτι από το u στο v στο αρχικό δίκτυο. Δεδομένου ότι οι ροές σε αντίθετες κατευθύνσεις αλληλοεξουδετερώνονται, μειώνοντας την ροή από το **v** προς το **u** είναι το ίδιο με την αύξηση της ροής από το **u** προς το **v**.



Εικόνα 12: Αριστερά βλέπουμε ένα δίκτυο ροής και δεξιά ένα υπολειπόμενο δίκτυο

ΠΕΡΙΚΟΠΕΣ (CUTS)

Στη θεωρία γράφων, μια περικοπή είναι μια διχοτόμηση των κορυφών ενός γραφήματος σε δύο ανεξάρτητα υποσύνολα. Οποιαδήποτε περικοπή καθορίζει ένα διαχωριστικό σύνολο, το σύνολο των ακμών που έχουν ένα καταληκτικό σημείο σε κάθε υποσύνολο του διαχωρισμού. Αυτές οι άκρες πρέπει να διασχίσουν την περικοπή. Σε ένα συνδεδεμένο γράφημα, κάθε διαχωριστικό σύνολο καθορίζει μια μοναδική περικοπή, και σε ορισμένες περιπτώσεις οι περικοπές προσδιορίζονται με τα διαχωριστικά σύνολά τους παρά με τα χωρίσματα της κορυφή τους.

Σε ένα δίκτυο ροής, η περικοπή s-t είναι μια περικοπή που απαιτεί την πηγή και το νεροχύτη να είναι σε διαφορετικά υποσύνολα, και το διαχωριστικό σύνολο απο μόνο του αποτελείται από ακμές που πηγαίνουν από την πλευρά της πηγής προς την πλευρά του νεροχύτη του. Η χωρητικότητα μίας περικοπής ορίζεται ως το άθροισμα της χωρητικότητα σε κάθε άκρη του διαχωριστικού συνόλου.

ΟΡΙΣΜΟΣ

Μια περικοπή **C = (S, T)** είναι ένας διαχωρισμός του **V** ενός γραφήματος **G = (V, E)** σε δύο υποσύνολα S και T. Το σύνολο διαχωρισμού της περικοπής **C = (S, T)** είναι το σύνολο
{(**u**, **v**) e **E** | **u** e **S**, **v** e **T**} των ακμών που έχουν ένα καταληκτικό σημείο στο **s** και το άλλο καταληκτικό σημείο στο **T** .Αν **s** και **t** ορίζονται κορυφές του γραφήματος **G**, τότε μία **s-t** περικοπή είναι μια περικοπή στην οποία το **s** ανήκει στο σύνολο **S** και **t** ανήκει στο σύνολο **T**.

ΔΙΑΔΡΟΜΕΣ ΕΠΑΥΞΗΣΗΣ (AUGMENTING PATHS)

Οποιαδήποτε διαδρομή στο υπολειπόμενο γράφημα ***Gf*** από τον κόμβο προέλευσης ***s*** στον κόμβο απόληξης ***t*** ονομάζεται **διαδρομή επαύξησης**.

<u>Βρίσκοντας μία αυξητική διαδρομή</u>:

-Βρίσκουμε ένα μονοπάτι από το s στο t στο υπολειπόμενο γράφημα

-Η εναπομείνουσα χωρητικότητα ενός μονοπατιού p στο Gf είναι:
CF (p) = min {CF (u, v): (u, v) είναι στο p}

• δηλαδή βρίσκω την ελάχιστη χωρητικότητα κατα μήκος του p

-Κάνω αύξηση: για όλα τα (u, v) στο p, απλά προσθέτω αυτό το cf(p) στο f (u, v) (και το αφαιρώ από το f(v, u)).

- Η προκύπτουσα ροή είναι μία έγκυρη ροή με μία μεγαλύτερη αξία.

3.3) EDMOND'S - KARP ALGORITHM

Ο αλγόριθμος εκδόθηκε για πρώτη φορά από τον Yefim (Chaim) Dinic το 1970 και ανεξάρτητα δημοσιεύθηκε από τον Jack Edmonds και τον Richard Karp το 1972.Είναι πανομοιότυπος με τον αλγόριθμο Ford-Fulkerson, εκτός του ότι η σειρά αναζήτησης κατά την εύρεση του αυξητικού μονοπατιού καθορίζεται.

Η διαδρομή που έχει βρεθεί πρέπει να είναι το συντομότερο μονοπάτι που έχει διαθέσιμη χωρητικότητα. Αυτό μπορεί να βρεθεί με μια breadth-first αναζήτηση (π.χ με τον αλγόριθμο dijkstra) θέτοντας τις τιμές της κάθε ακμής ίσες με τη μονάδα. Ο χρόνος εκτέλεσης O (V E2) βρίσκεται δείχνοντας ότι κάθε αυξανόμενη διαδρομή μπορεί να βρεθεί σε O (E) χρόνο, κάθε φορά τουλάχιστον που μία από τις ακμές E κορεστεί (μία ακμή η οποία έχει τη μέγιστη δυνατή ροή),ώστε η απόσταση από τη κορεσμένη ακμή προς την πηγή κατά μήκος της αυξητικής διαδρομής πρέπει να είναι μεγαλύτερη από την τελευταία φορά που ήταν κορεσμένη, και το μήκος να είναι το πολύ V.

Μια άλλη ιδιότητα αυτού του αλγορίθμου είναι ότι το μήκος του συντομότερου αυξανόμενου μονοπατιού αυξάνει μονοτονικά.

Αλγόριθμος Edmond's-Karp

$f(u,v) = 0$ για κάθε $u,v \in G$
while υπάρχει αυξητικό μονοπατι p **do**
Εύρεση του συντομότερου αυξητικου μονοπατιού (αναζήτηση κατά πλάτος)
$c = \min c\ \mathrm{f}(u,v)$, για κάθε $(u,v) \in p$
για κάθε$(\mathrm{u},v) \in p \Rightarrow f(u,v) = f(u,v) + c$ και $f(v,u) = -f(u,v)$
end while

Παρακάτω θα δούμε ένα δείγμα κώδικα c++ για τον αλγόριθμο edmonds-karp. Ο σκοπός του αλγορίθμου είναι να βρεθεί η μέγιστη ροή που μπορεί να περάσει μέσα από ένα δίκτυο με μία μόνο πηγή και νεροχύτη.

Ο αλγόριθμος χρησιμοποιεί αναζήτηση πρώτα κατά πλάτος σε κάθε επανάληψη για να βρει ένα μονοπάτι, από την πηγή στον νεροχύτη, μέσω του οποίου η ροή μπορεί να περάσει. Αν βρεθεί ένα μονοπάτι ενημερώνουμε την υπολειπόμενη χωρητικότητα των ακμών σε αυτό το μονοπάτι και αν δεν βρεθεί κανένα τέτοιο μονοπάτι, τότε ο αλγόριθμος τερματίζει.

```cpp
#include<cstdio>
#include<cstdio>
#include<queue>
#include<cstring>
#include<vector>
#include<iostream>

#define MAX_NODES 100 //ο μέγιστος αριθμός κόμβων σε ένα γράφο//
#define INF 2147483646  // αναπαριστά το άπειρο
#define UNINITIALIZED -1 // τιμή για κόμβους χωρίς "γονέα"
using namespace std;

//αναπαριστά τις χωρητικότητες των ακμών
int capacities[MAX_NODES][MAX_NODES];
                                    //δείχνει πόση ροή έχει περάσει από ένα κόμβο
int flowPassed[MAX_NODES][MAX_NODES];
//αναπαριστά το γράφο.Ο γράφος πρέπει να έχει και αρνητικές τιμές//
vector<int> graph[MAX_NODES];
                                    //δείχνει τους γονείς του κόμβου του μονοπατιού που δημιουργήθηκε από BFS
int parentsList[MAX_NODES];
                                    //δείχνει τη μέγιστη ροή σε ένα κόμβο του μονοπατιού που δημιουργήθηκε από BFS
int currentPathCapacity[MAX_NODES];

int bfs(int startNode, int endNode)
{
   memset(parentsList, UNINITIALIZED, sizeof(parentsList));
   memset(currentPathCapacity, 0, sizeof(currentPathCapacity));

   queue<int> q;
   q.push(startNode);

   parentsList[startNode]=-2;
   currentPathCapacity[startNode]=INF;

   while(!q.empty())
   {
      int currentNode = q.front(); q.pop();

      for(int i=0; i<graph[currentNode].size(); i++)
      {
         int to = graph[currentNode][i];
         if(parentsList[to] == UNINITIALIZED)
         {
            if(capacities[currentNode][to] - flowPassed[currentNode][to] > 0)
            {
//ενημερώνουμε το γονέα του τρέχων κόμβου να είναι ο μελλοντικός κόμβος//
               parentsList[to] = currentNode;

//ελέγχουμε ποιά είναι η ελάχιστη υπολειπόμενη χωρητικότητα ακμής μέχρι στιγμής//
               currentPathCapacity[to] = min(currentPathCapacity[currentNode],
                capacities[currentNode][to] - flowPassed[currentNode][to]);
//Αν έχουμε φτάσει στο τέλος του κόμβου οι BFS πρέπει να τερματιστούν//
               if(to == endNode) return currentPathCapacity[endNode];

//προσθέτουμε το μελλοντικό κόμβο στην ουρά//
               q.push(to);
            }
         }
      }
   }

   return 0;
}

int edmondsKarp(int startNode, int endNode)
{
   int maxFlow=0;

   while(true)
```

```
71      while(true)
72      {                  βρίσκουμε μία αυξητική διαδρομή και τη μέγιστη ροή
73                         που της αντιστοιχεί
74          int flow=bfs(startNode, endNode);
75
76  αν δε μπορούμε πια να βρούμε διαδρομές,η ροή θα είναι 0.
77          if(flow==0)
78          {
79              break;
80          }
81
82          maxFlow +=flow;
83          int currentNode=endNode;
84
85  //ενημερώνουμε τον περασμένο πίνακα ροής//
86          while(currentNode != startNode)
87          {
88              int previousNode = parentsList[currentNode];
89              flowPassed[previousNode][currentNode] += flow;
90              flowPassed[currentNode][previousNode] -= flow;
91
92              currentNode=previousNode;
93          }
94      }
95
96      return maxFlow;
97  }
98
99  int main()
100 {
101     int nodesCount, edgesCount;
102     cin>>nodesCount>>edgesCount;
103
104     int source, sink;
105     cin>>source>>sink;
106
107     for(int edge=0; edge<edgesCount; edge++)
108     {
109         int from, to, capacity;
110         cin>>from>>to>>capacity;
111
112         capacities[from][to]=capacity;
113         graph[from].push_back(to);
114
115 //προσθέτουμε τις αρνητικές ακμές//
116         graph[to].push_back(from);
117     }
```

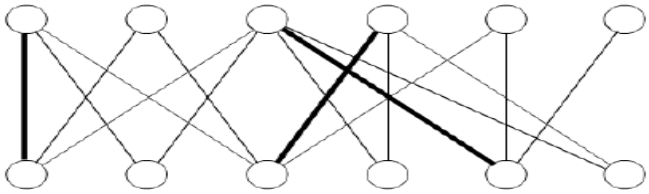
3.4) HOPCROFT-KARP ALGORITHM

```
118
119     int maxFlow = edmondsKarp(source, sink);
120
121     cout<<maxFlow<<endl;
122
123     return 0;
124 }
```

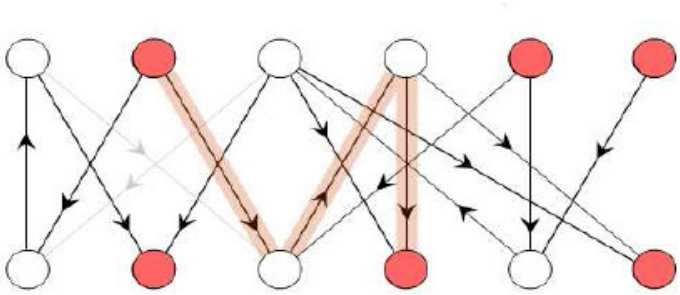
Ο αλγόριθμος βρέθηκε από τον John Hopcroft και Richard Karp (1973). Όπως και στις προηγούμενες μεθόδους ταιριάσματος, ο αλγόριθμος Hopcroft-Karp αυξάνει επανειλημμένα το μέγεθος του μερικού ταιριάσματος με την εύρεση αυξητικών μονοπατιών. Ωστόσο, προκειμένου να επιταχύνει το χρόνο εργασίας αντί να ψάχνει για μονοπάτια ένα προς ένα, αναζητά πολλά μονοπάτια στον ίδιο χρόνο.

Η βασική ιδέα είναι να εγγυηθεί ότι το μήκος της διαδρομής μεγαλώνει σε κάθε βήμα. Δεν μπορεί να υπάρχουν πολλά μεγάλα ασύνδετα μονοπάτια, έτσι ψάχνοντας για αυτά δεν θα πάρει πολύ χρόνο.

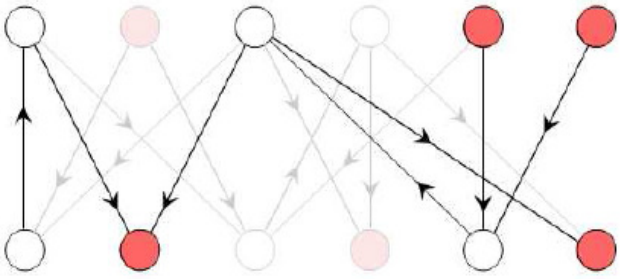
given a graph G = (X ∪Y),E)
1) Let M = { } ,
2) Find S = {P1 , P2 , … P3)
3) While S ≠ { } M = M ⊕ S Find S
4) Output M



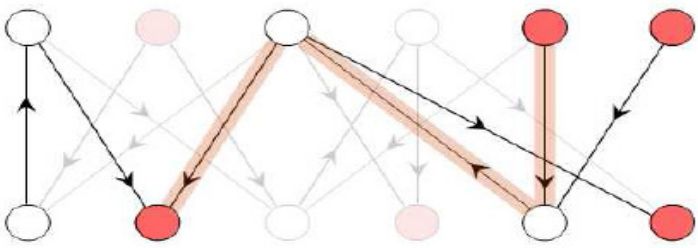
Οι σκουρόχρωμες άκρες αναπαριστούν τις άκρες σε ένα ταίριασμα M



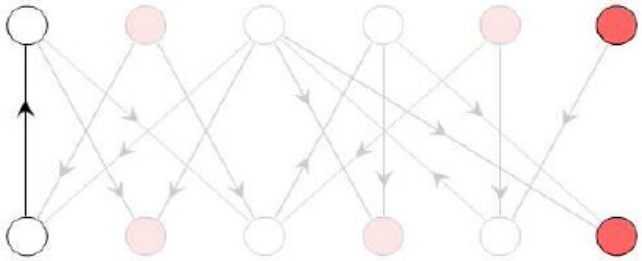
Οι ρόζ άκρες αναπαριστούν ένα αυξητικό μονοπάτι



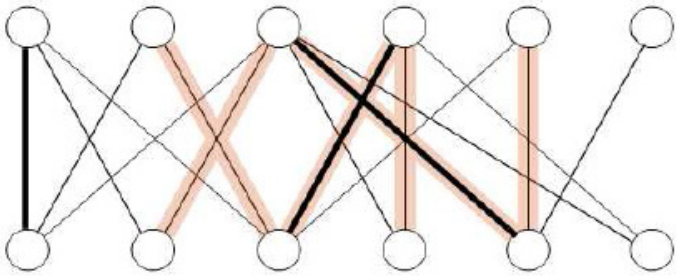
τις διαγράφουμε



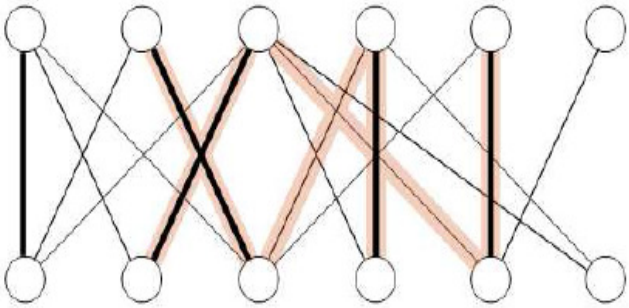
Άλλο ένα αυξητικό μονοπάτι



Δεν υπάρχουν άλλα μονοπάτια



Οι ρόζ άκρες αναπαριστούν τα μονοπάτια στο μέγιστο σύνολο S



M □ S
Παρατηρήστε το πριν και το μετά

ΚΕΦΑΛΑΙΟ 4

ΜΟΝΤΕΛΑ ΠΛΗΡΟΦΟΡΙΩΝ

4.1) ΕΙΣΑΓΩΓΗ

Ένα μοντέλο πληροφοριών δεν είναι ένας τύπος μοντέλου δεδομένων, αλλά λίγο πολύ ένα εναλλακτικό μοντέλο. Εντός του τομέα της τεχνολογίας λογισμικού και ένα μοντέλο δεδομένων αλλά και ένα μοντέλο πληροφοριών μπορεί να είναι αφηρημένες, επίσημες αναπαραστάσεις των τύπων οντότητας που περιλαμβάνουν τις ιδιότητές τους, τις σχέσεις και τις λειτουργίες που μπορούν να εκτελεστούν σε αυτά.

Οι τύποι οντότητας στο μοντέλο μπορεί να είναι είδη αντικειμένων του πραγματικού κόσμου, όπως συσκευές σε ένα δίκτυο, ή μπορεί να είναι οι ίδιοι αφηρημένοι, όπως οι οντότητες που χρησιμοποιούνται σε ένα σύστημα τιμολόγησης. Συνήθως, χρησιμοποιούνται για να διαμορφώσουν μια περιορισμένη περιοχή που μπορεί να περιγραφεί από ένα κλειστό σύνολο τύπων οντοτήτων, ιδιότητες, σχέσεις και πράξεις.

Ένα μοντέλο πληροφοριών παρέχει φορμαλισμό για την περιγραφή ενός πεδίου του προβλήματος χωρίς να περιορίζει το πώς αυτή η περιγραφή έχει αντιστοιχιστεί σε μια πραγματική εφαρμογή λογισμικού. Μπορεί να υπάρχουν πολλές απεικονίσεις του μοντέλου πληροφοριών. Αυτές οι αντιστοιχίσεις ονομάζονται μοντέλα δεδομένων, ανεξάρτητα από το αν είναι μοντέλα αντικειμένου (π.χ. με τη χρήση UML), μοντέλα σχέσης οντοτήτων ή σχήματα XML.

4.2) ΛΟΓΙΚΑ ΜΟΝΤΕΛΑ ΠΛΗΡΟΦΟΡΙΩΝ (LOGICAL INFORMATION MODELS)

Στις αρχές της δεκαετίας του '70 υπήρξαν αρκετές προτάσεις για λογικά μοντέλα δεδομένων που προσέφεραν αφηρημένες δομές μαθηματικών συμβόλων (π.χ., σύνολα, πίνακες, σχέσεις) για σκοπούς μοντελοποίησης, κρύβοντας τις λεπτομέρειες εφαρμογής από τον χρήστη. Τα σχεσιακά και τα δικτυακά μοντέλα για τις βάσεις δεδομένων είναι καλά παραδείγματα λογικών μοντέλων. Τέτοια μοντέλα ελευθερώνουν όποιον μοντελοποιεί από ανησυχίες για την εφαρμογή, ώστε να μπορεί να επικεντρώνεται στην μοντελοποίηση.

Για παράδειγμα, όταν η μοντελοποίηση έχει επιλέξει το σχεσιακό μοντέλο, μπορεί να πάει μπροστά και να χρησιμοποιήσει πίνακες για να χτίσει μια βάση πληροφοριών, χωρίς καμία σχέση με το πώς αυτοί οι πίνακες έχουν φυσικά υλοποιηθεί. Δυστυχώς, οι λογικές δομές συμβόλων είναι επίπεδες και διαισθητικές ως προς το πώς θα πρέπει να χρησιμοποιούνται για σκοπούς μοντελοποίησης.

4.3) ΕΝΝΟΙΟΛΟΓΙΚΑ ΜΟΝΤΕΛΑ ΠΛΗΡΟΦΟΡΙΩΝ (CONCEPTUAL INFORMATION MODELS)

Λίγο μετά αφού προτάθηκαν τα λογικά μοντέλα πληροφοριών, και ακόμη πριν η σχεσιακή τεχνολογία κατακτήσει τη βιομηχανία της βάσης δεδομένων, υπήρξαν νέες προτάσεις για μοντέλα πληροφοριών τα οποία προσφέρουν πιο εκφραστικές διευκολύνσεις για τις εφαρμογές μοντελοποίησης και διάρθρωση των βάσεων πληροφοριών.

Τα μοντέλα αυτά (στο εξής, εννοιολογικά μοντέλα) προσφέρουν σημασιολογικούς όρους για τη μοντελοποίηση μιας εφαρμογής, όπως : Οντότητα, Δραστηριότητα, Αντιπρόσωπος και στόχος. Τα μοντέλα αυτά έπρεπε να διαμορφώσουν μια εφαρμογή περισσότερο άμεση και φυσική.

Πρωταρχικός στόχος ενός εννοιολογικού μοντέλου είναι να μεταφέρει τις θεμελιώδεις αρχές και τις βασικές λειτουργίες του συστήματος που αντιπροσωπεύει. Επίσης, ένα εννοιολογικό μοντέλο θα πρέπει να αναπτυχθεί με τέτοιο τρόπο ώστε να παρέχει μια εύκολα κατανοητή ερμηνεία του συστήματος για τους χρήστες του μοντέλου.

4.4) FACILITY INFORMATION MODEL

Ένα μοντέλο πληροφοριών εγκατάστασης (FIM) είναι μια συλλογή (αποκεντρωμένη) σημασιολογικών βάσεων δεδομένων που ενσωματώνει κατ 'αρχήν όλα τα δεδομένα και τα έγγραφα της εγκατάστασης σε όλη τη διάρκεια του κύκλου ζωής της. Αυτό μπορεί να περιλαμβάνει φάσεις όπως ο σχεδιασμός, κατασκευή, θέση σε λειτουργία, τη λειτουργία και τη συντήρηση. Η βάση δεδομένων μπορεί να είναι κεντρική, αλλά συνήθως κατανέμεται σε διάφορες αποθήκες δεδομένων. Αυτές οι αποθήκες δεδομένων μπορεί να διατηρηθούν ανεξάρτητα, ενώ οι συμβάσεις μοντελοποίησης επιτρέπουν την ενσωμάτωση δεδομένων και τη διαλειτουργικότητα των συστημάτων.

4.5) COMMON INFORMATION MODEL

Το Common Information Model (CIM) είναι ένα βιομηχανικό πρότυπο υπολογιστή για τον καθορισμό της συσκευής και τα χαρακτηριστικά της εφαρμογής έτσι ώστε οι διαχειριστές του συστήματος και των προγραμμάτων θα είναι σε θέση να ελέγχουν τις συσκευές και τις εφαρμογές από διαφορετικούς κατασκευαστές ή πηγές με τον ίδιο τρόπο. Για παράδειγμα, μια εταιρεία που αγόρασε διάφορα είδη συσκευών αποθήκευσης από διαφορετικές εταιρείες θα είναι σε θέση να δεί το ίδιο είδος πληροφοριών (όπως: το όνομα της συσκευής και το μοντέλο, το σειριακό αριθμό, την χωρητικότητα, τη θέση του δικτύου, και τη σχέση με άλλες συσκευές ή εφαρμογές) για καθένα από αυτά ή είναι σε θέση να έχει πρόσβαση στις πληροφορίες από ένα πρόγραμμα.

Το CIM εκμεταλλεύεται την Extensible Markup Language (XML). Οι κατασκευαστές υλικού και λογισμικού επιλέγουν ένα από τα πολλά XML σχήματα που ορίζονται για την παροχή CIM πληροφοριών για το προϊόν τους.

4.6) BUILDING INFORMATION MODEL (BIM)

Η μοντελοποίηση πληροφοριών κτηρίου (BIM) είναι μια διαδικασία που περιλαμβάνει την παραγωγή και διαχείριση των ψηφιακών αναπαραστάσεων των φυσικών και λειτουργικών χαρακτηριστικών των θέσεων. Τα μοντέλα πληροφοριών κτηρίου είναι αρχεία (συχνά, αλλά όχι πάντα σε ιδιόκτητη φορμά και περιέχει δεδομένα βιομηχανικής ιδιοκτησίας), που μπορούν να ανταλλαγούν ή να είναι δικτυομένα για την υποστήριξη λήψης-αποφάσεων για έναν τόπο.

ΚΕΦΑΛΑΙΟ 5

ΜΕΤΡΙΚΕΣ ΟΜΟΙΟΤΗΤΑΣ

5.1) ΕΙΣΑΓΩΓΗ

Υπάρχουν πολλοί τρόποι για να προσδιοριστεί η ομοιότητα μεταξύ δύο πραγμάτων. Για να αναπαραστήσουμε αυτή την ομοιότητα σε μια μηχανή, πρέπει να ορίσουμε μια βαθμολογία ομοιότητας. Εάν μπορούμε να ποσοτικοποιήσουμε διαφορετικά χαρακτηριστικά των αντικειμένων δεδομένων, μπορούμε να χρησιμοποιήσουμε διαφορετικούς αλγορίθμους ομοιότητας σε όλα τα εν λόγω χαρακτηριστικά που θα αποδώσουν βαθμολογίες ομοιότητας μεταξύ των διαφορετικών αντικειμένων δεδομένων.

Για παράδειγμα, μπορούμε να αναπαραστήσουμε τους ανθρώπους ως αντικείμενα δεδομένων των οποίων τα χαρακτηριστικά είναι τα γούστα σε ταινίες. Μπορούμε να χρησιμοποιήσουμε μια μετρική ομοιότητας για να μας βοηθήσει να βρούμε ποιοι άνθρωποι είναι "παρόμοιοι" με βάση το πόσο όμοια είναι τα γούστα τους σε ταινίες. Εμείς το μόνο που χρειάζεται να ποσοτικοποιήσουμε είναι τα γνωρίσματα των γούστων των ταινιών.

Για παράδειγμα, θα μπορούσαμε να χρησιμοποιήσουμε αριθμητικές βαθμολογίες στις ταινίες αυτών των ανθρώπων.

5.2) ΕΥΚΛΕΙΔΙΑ ΑΠΟΣΤΑΣΗ

Ένας απλός αλλά ισχυρός τρόπος για να καθορίσουμε την ομοιότητα είναι να υπολογιστεί η Ευκλείδεια απόσταση μεταξύ δύο αντικειμένων δεδομένων. Για να γίνει αυτό, χρειαζόμαστε τα αντικείμενα δεδομένων να έχουν αριθμητικά γνωρίσματα..Μπορεί επίσης να χρειαστεί να ομαλοποιήσουμε τα γνωρίσματα.

Για παράδειγμα, αν έχουμε σύγκριση με βαθμολογίες των ανθρώπων για τις ταινίες, θα πρέπει να βεβαιωθούμε ότι η κλίμακα κατάταξης είναι η ίδια σε όλους τους ανθρώπους καθώς θα ήταν προβληματικό να συγκρίναμε τη βαθμολογία με 5 κάποιου σε μια κλίμακα 1-5 και την βαθμολογία με 5 ενός άλλου ατόμου σε μια κλίμακα 1-10. Το επόμενο βήμα είναι να εφαρμοστεί ο τύπος της Ευκλείδειας απόστασης:

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \sqrt{(p_1-q_1)^2+(p_2-q_2)^2+\cdots+(p_n-q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i-q_i)^2}.$$

Όσο μικρότερη είναι η απόσταση, τόσο πιο όμοια είναι τα αντικείμενα δεδομένων.

5.3) PEARSON

Ο συντελεστής συσχέτισης Pearson (Pearson's Correlation Coefficient), αναπτύχθηκε από τον Karl Pearson, βασισμένος στην ιδέα που είχε εισάγει ο Francis Galton στη δεκαετία του 1880. Ο συντελεστής Pearson είναι μία πιο σύνθετη και περίπλοκη προσέγγιση για την εξεύρεση ομοιότητας.

Ο συντελεστής υπολογίζει τη γραμμική συσχέτιση δυο διανυσμάτων και επιστρέφει τιμές από -1 έως 1. Όσο η τιμή πλησιάζει το 1 και -1 αντίστοιχα, τα διανύσματα μοιάζουν όλο και περισσότερο, ενώ όσο η τιμή πλησιάζει στο μηδέν, τα διανύσματα είναι όλο και πιο αντίθετα. Ο τύπος της μετρικής είναι ο ακόλουθος:

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y},$$

Ο συντελεστής βρίσκεται από διαίρεση της συνδιασποράς με το γινόμενο των τυπικών αποκλίσεων των ιδιοτήτων των δύο αντικειμένων δεδομένων. Το πλεονέκτημα του Συντελεστή Pearson έναντι στην Ευκλείδεια απόσταση είναι ότι είναι πιο εύρωστο απέναντι σε δεδομένα που δεν είναι κανονικοποιημένα.

5.4) ΣΥΝΤΕΛΕΣΤΗΣ JACCARD

Όταν έχουμε να κάνουμε με αντικείμενα δεδομένων τα οποία έχουν δυαδικές ιδιότητες, είναι πιο αποτελεσματικό να υπολογίσουμε την ομοιότητα χρησιμοποιώντας ένα συντελεστή Jaccard. Η εξίσωση για να βρούμε τον συντελεστή Jaccard έχει ως εξής:

$$J = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}.$$

Το M11 αντιπροσωπεύει το συνολικό αριθμό των χαρακτηριστικών, όπου τα δύο αντικείμενα δεδομένων έχουν ένα 1. Το M10 και M01 αντιπροσωπεύουν το συνολικό αριθμό των χαρακτηριστικών, όπου ένα αντικείμενο δεδομένων έχει ένα 1 και το άλλο έχει το 0. Οι συνολικές ιδιότητες ταιριάσματος στη συνέχεια διαιρούνται με τα συνολικά μη-αντιστοιχισμένα χαρακτηριστικά συν τα αντιστοιχισμένα.Ένα τέλειο σκορ ομοιότητας θα είναι τότε ένα 1.Για παράδειγμα, εάν το αντικείμενο "A" είχε ιδιότητες των 1, 0, 1, και το αντικείμενο "B" είχε ιδιότητες των 1, 1, 1, ο συντελεστής Jaccard θα είναι 2/3.

5.5) ΟΜΟΙΟΤΗΤΑ ΣΥΝΗΜΙΤΟΝΟΥ (COSINE SIMILARITY)

Η εύρεση της ομοιότητας συνημιτόνου μεταξύ δύο αντικειμένων δεδομένων προϋποθέτει ότι τα δύο αντικείμενα αντιπροσωπεύουν τις ιδιότητές τους σε ένα διάνυσμα. Η ομοιότητα στη συνέχεια μετράται ως το συνημίτονο μεταξύ των δύο διανυσμάτων. Επιστρέφει τιμές ανάμεσα στο -1 και το 1. Κυρίως όμως στην ομοιότητα χρησιμοποιεί μόνο θετικές τιμές στο διάστημα [0,1], οπότε όσο πιο κοντά στο 1 είναι η τιμή τόσο μεγαλύτερη η ομοιότητα, ενώ όσο πιο κοντά στο 0, τόσο μικρότερη.

Η ομοιότητα συνημιτόνου βρίσκεται από τον παρακάτω τύπο:

$$\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}.$$

5.6) ΣΥΝΤΕΛΕΣΤΗΣ TANIMOTO

Ο συντελεστής Tanimoto είναι μια εκτεταμένη εκδοχή του συντελεστή Jaccard και της ομοιότητας συνημιτόνου. Προϋποθέτει επίσης ότι κάθε αντικείμενο δεδομένων είναι ένα διάνυσμα χαρακτηριστικών. Τα χαρακτηριστικά μπορούν ή δεν μπορούν να είναι δυαδικά στην περίπτωση αυτή.Ο συντελεστής Tanimoto βρίσκεται από την ακόλουθη εξίσωση:

$$T(A,B) = \frac{A \cdot B}{\|A\|^2 + \|B\|^2 - A \cdot B}.$$

Στην εξίσωση, τα A και B είναι τα αντικείμενα δεδομένων που αντιπροσωπεύονται από διανύσματα. Το σκορ ομοιότητας είναι το γινόμενο των A και B διαιρούμενο με τα τετραγωνισμένα μεγέθη των A και B μείον το γινόμενο.

Χρησιμοποιώντας το παράδειγμα του μπακάλικου,ο συντελεστής Tanimoto εξασφαλίζει ότι ο πελάτης που αγοράζει πέντε μήλα και ένα πορτοκάλι θα είναι διαφορετικός από έναν πελάτη ο οποίος αγοράζει πέντε πορτοκάλια και ένα μήλο.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Στη παρούσα πτυχιακή εργασία αρχικά εξετάστηκαν αναλυτικά διάφορα μοντέλα που χρησιμοποιούνται έτσι ώστε να αναπαραστίσουμε τη γνώση που κατέχουμε και παρουσιάστηκαν κατανοητά παραδείγματα. Στη συνέχεια ορίστηκε η έννοια του ταιριάσματος σε γραφήματα και παρουσιάστηκαν αλγόριθμοι ταιριάσματος (για αντιστοίχιση προτύπων, κατασκευή μέγιστου προτύπου κλπ).

Παρουσιάστηκαν επίσης μοντέλα πληροφοριών και μετρικές που χρησιμοποιούμε για να ορίσουμε την ομοιότητα μεταξύ δύο αντικειμένων και είδαμε πως ανάλογα με το τι πρέπει να συγκρίνουμε χρησιμοποιείται διαφορετική μετρική ομοιότητας.

ΒΙΒΛΙΟΓΡΑΦΙΑ:

- http://www.cs.dartmouth.edu/~ac/Teach/CS105-Winter05/Handouts/vempala-blossom.pdf
- http://ikee.lib.auth.gr/record/114473/files/ptuxiaki.pdf
- https://en.wikipedia.org/wiki/Rete_algorithm
- http://swt-www.informatik.uni-hamburg.de/uploads/media/Diplomarbeit_Eryk_Lagun.pdf
- Predicting the Choice of Contraceptive Method using Classification,Παπαδόπουλος Χρήστος,τμήμα εφαρμοσμένης πληροφορικής (θεσσαλονίκη)
- http://www.cs.ucy.ac.cy/~mavronic/Classes/cs232/Notes/notes10-11.pdf
- https://en.wikipedia.org/wiki/Ford-Fulkerson_algorithm
- https://en.wikipedia.org/wiki/Edmonds-Karp_algorithm
- http://www.dit.hua.gr/~michail/teaching/algs/slides/070_Flows.pdf
- http://www.cs.cmu.edu/~avrim/451f11/lectures/lect1027.pdf
- http://www.cs.ucc.ie/~gprovan/CS4407/Ford-Fulkerson-lecture1.pdf
- http://www.cs.toronto.edu/~jm/2507S/Readings/Survey.pdf
- http://p-comp.di.uoa.gr/resources/Lika_MscThesis.pdf
- http://www.teilar.gr/dbData/ProfAnn/profann-96a4cd4e.pdf