ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Ταίριασμα υπογραφημάτων με τοπική αναζήτηση

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ladislaus Schmidt

Επιβλέπων: Κωνσταντίνος Στεργίου

Επίκουρος Καθηγητής Π.Δ.Μ.

Κοζάνη, 2014

Πανεπιστήμιο Δυτικής Μακεδονίας
Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών

Ταίριασμα υπογραφημάτων με τοπική αναζήτηση

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ladislaus Schmidt

Επιβλέπων: Κωνσταντίνος Στεργίου
Επίκουρος Καθηγητής Π.Δ.Μ.

Εγκρίθηκε από την εξεταστική επιτροπή την

(Υπογραφή) (Υπογραφή) .

....

Κ. Στεργίου

Επ. Καθηγητής Π.Δ.Μ.

Κοζάνη, 2014

Πανεπιστήμιο Δυτικής Μακεδονίας
Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών

(Υπογραφή)

.....

Ladislaus Schmidt
Διπλωματούχος Μηχανικός Πληροφορικής και Τηλεπικοινωνιών ΠΔΜ

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή μου κ.Κωνσταντίνο Στεργίου για την άμεση και πολύτιμη βοήθεια του στην εκπόνηση της παρούσας εργασίας. Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου, που με στήριξε και με στηρίζει στο μακρύ ταξίδι της επαγγελματικής μου σταδιοδρομίας.

Περίληψη

Το θέμα της συγκεκριμένης διπλωματικής εργασίας είναι η υλοποίηση του αλγόριθμου τοπικής αναζήτησης Hill-Climbing για το ταίριασμα γραφημάτων, ένα από τα σημαντικότερα προβλήματα των γραφημάτων.
Ο αλγόριθμος αντιμετωπίζεται ως ένα πρόβλημα ικανοποίησης περιορισμών και δοθέντος δύο γραφημάτων G1 και G2, αναζητά τουλάχιστον ένα ταίριασμα του G2 μέσα στο G1.
Η υλοποίηση έγινε σε γλώσσα προγραμματισμού Java.
Στα πρώτα τέσσερα κεφάλαια γίνεται μια ανάλυση από θεωρητικής πλευράς των γραφημάτων , των προβλημάτων ικανοποίησης περιορισμών , της τοπικής αναζήτησης και τέλος του αλγόριθμου hill-climbing. Έπειτα γίνεται η ανάλυση και η τεκμηρίωση του κώδικα.

Πίνακας περιεχομένων

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

1 Εισαγωγή στους γράφους

Στο τρέχον κεφάλαιο της εργασίας παρουσιάζονται συνοπτικά βασικές έννοιες της θεωρίας των γράφων οι οποίες θα χρησιμοποιηθούν στην ανάλυση της εφαρμογής που θα γίνει στα επόμενα κεφάλαια της διπλωματικής εργασίας. Επίσης γίνεται μια σύντομη αναφορά στις εφαρμογές των γράφων, καθώς και σε προβλήματα που δημιουργούνται με την χρήση αυτών, δίνοντας ιδιαίτερη έμφαση στο ταίριασμα υπογραφημάτων (subgraph matching).

1.1 Γράφος

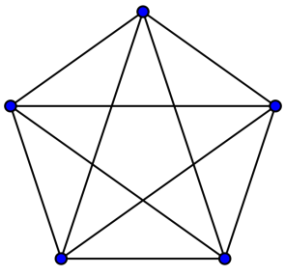
Ο γράφος στον απλούστερο ορισμό του είναι η οπτική αναπαράσταση των σχέσεων που αναπτύσσουν ορισμένες ποσότητες, σχεδιασμένες σε σχέση με ένα σύνολο αξόνων. Ένας άλλος ορισμός [1] που κινείται στο ίδιο εννοιολογικό πλαίσιο της οπτικής αναπαράστασης αναγνωρίζει τον γράφο ως απεικόνιση αποτελούμενη από ένα σύνολο σημείων (κορυφών ή κόμβων) που συνδέονται με γραμμές (ακμές). Στους κατευθυνόμενους ή προσανατολισμένους γράφους οι ακμές απεικονίζονται διανυσματικά.

Σε μία άλλη εκδοχή είναι ένα σύνολο από κόμβους (κορυφές) που ενώνονται μεταξύ τους με ακμές και ορίζεται από τον τρόπο με τον οποίο συνδέονται οι κορυφές (κόμβοι). Αν οι ακμές προσανατολίζονται οριζόμενες από διατεταγμένα ζεύγη κόμβων, τότε ο γράφος αποκαλείται κατευθυνόμενος. Αν οι ακμές δεν προσανατολίζονται, οριζόμενες απλώς από διμελή σύνολα και όχι διατεταγμένα ζεύγη, τότε αποκαλείται μη κατευθυνόμενος. Επιπλέον στοιχεία για τον ορισμό ενός γράφου είναι η σύνδεση των ακμών του με κάποια αξία, οπότε αποκαλείται σταθμισμένος.

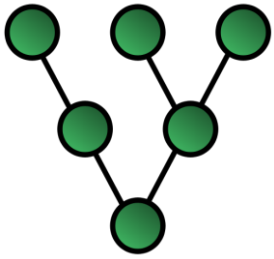
Με τη σειρά του πλήρης αποκαλείται ο γράφος που περιέχει ακμές για κάθε ζεύγος κόμβων, αραιός εκείνος που περιέχει λίγες ακμές ή αντίστροφα πυκνός

Συνοψίζοντας, ένας γράφος συμβολίζεται με $G=(V,E)$

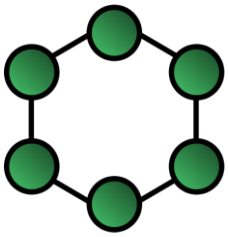
- Όπου $V=\{1,\ldots,n\}$ ένα σύνολο από κορυφές (vertices)
- $E \subseteq V \times V$ ένα σύνολο από πλευρές ή ακμές (edges)
- $e = (u,v) \in E$ συμβολίζουμε μια πλευρά
- $n = |V|$ το πλήθος των κορυφών
- $m = |E|$ το πλήθος των πλευρών



Σχήμα 1.1: Πλήρης μη κατευθυνόμενος γράφος 5 κόμβων



Σχήμα 1.2: Άκυκλος μη κατευθυνόμενος γράφος 6 κόμβων



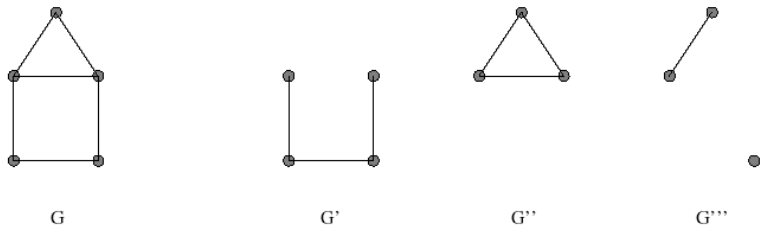
Σχήμα 1.3: Κυκλικός μη κατευθυνόμενος γράφος 6 κόμβων

1.1.1 Υπογράφος

Ο γράφος $G' = (V' , E')$ θα λέγεται **υπογράφος** ενός γράφου $G = (V , E)$ αν:

$$V' \subseteq V$$

$$E' \subseteq E$$

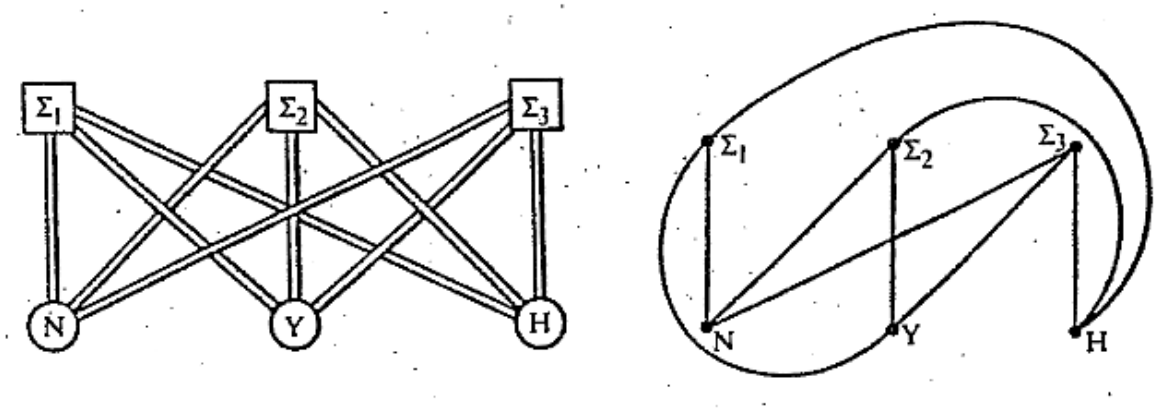


Σχήμα 1.4: Παραδείγματα υπογράφων του γράφου G

1.2 Εφαρμογές γραφημάτων

Η απλή δομή που έχουν τα γραφήματα και η άμεση εποπτεία που προσφέρουν κάνει εύκολη την αξιοποίηση τους τόσο στα μαθηματικά όσο και έξω από αυτά. Είναι χρήσιμα για την Ηλεκτρονική, την Αρχιτεκτονική, την Συγκοινωνιολογία , την Πληροφορική (Προγραμματισμό, Ανάλυση Συστημάτων, Σχεδιασμό Ηλεκτρονικών υπολογιστών, Σχεδίαση Βάσεων Δεδομένων), την Επιχειρησιακή Έρευνα, την Ιατρική κ.α. Παρακάτω αναφέρεται ένα απλό παράδειγμα εφαρμογής γράφων. Στην συνέχεια ακολουθεί ένας πίνακας (Πίνακας 1.1) με περισσότερα παραδείγματα εφαρμογής και τέλος κάποια γραφήματα από διάφορους επιστημονικούς τομείς.

Ας υποθέσουμε ότι έχουμε τρία σπίτια Σ1, Σ2, Σ3, που το καθένα είναι συνδεδεμένο με τριών ειδών παροχές – πχ νερό (Ν), υγραέριο (Υ), και ηλεκτρισμό (Η) – με σωλήνες. Μπαίνει το ερώτημα, είναι δυνατόν να κατασκευαστούν τέτοιες συνδέσεις που να μην διασταυρωθούν οι σωλήνες;
Στο σχήμα 1.5 φαίνεται πως το πρόβλημα [2] μπορεί να αναπαρασταθεί με ένα γράφημα. Οι σωλήνες εμφανίζονται ως ακμές ενώ τα σπίτια και τα κέντρα των παροχών ως κορυφές. Όπως θα δούμε παρακάτω, το γράφημα αυτό δεν μπορεί να γραφεί στο επίπεδο δίχως να διασταυρωθούν οι ακμές του. Έτσι λοιπόν η απάντηση στο πρόβλημα είναι αρνητική.



Σχήμα 1.5: Απλό παράδειγμα εφαρμογής γράφων

Εφαρμογή	Κορυφές	Ακμές	Ροή
Επικοινωνίες	Τηλεφωνικές συσκευές, υπολογιστές, δορυφόροι	Καλώδια, οπτικές ίνες, ηλεκτρομαγνητικά κύματα	ήχος, εικόνα
Κυκλώματα	λογικές πύλες, επεξεργαστές, αντιστάσεις κτλ..	σύρματα	ρεύμα
Μηχανολογία	αρθρώσεις	Βέργες, δοκοί, ελατήρια	Θερμότητα, ενέργεια
Υδραυλικές εγκαταστάσεις	Δεξαμενές, αντλιοστάσια, λίμνες	Σωληνώσεις	Νερό, πετρέλαιο

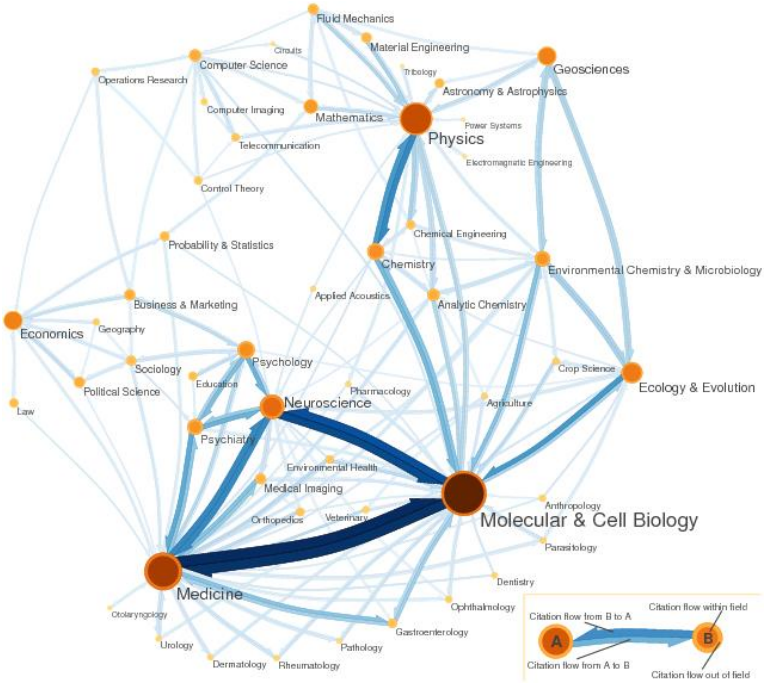
Οικονομολογικά	αποθέματα, το νόμισμα	Συναλλαγές	χρήματα
Συγκοινωνιολογία	Αεροδρόμια, Σταθμοί τραίνων, διασταυρώσεις δρόμων	Αυτοκινητόδρομοι, αεροδιάδρομοι, γραμμές τραίνων	οχήματα, επιβάτες

Πίνακας 1.1: Εφαρμογές γράφων

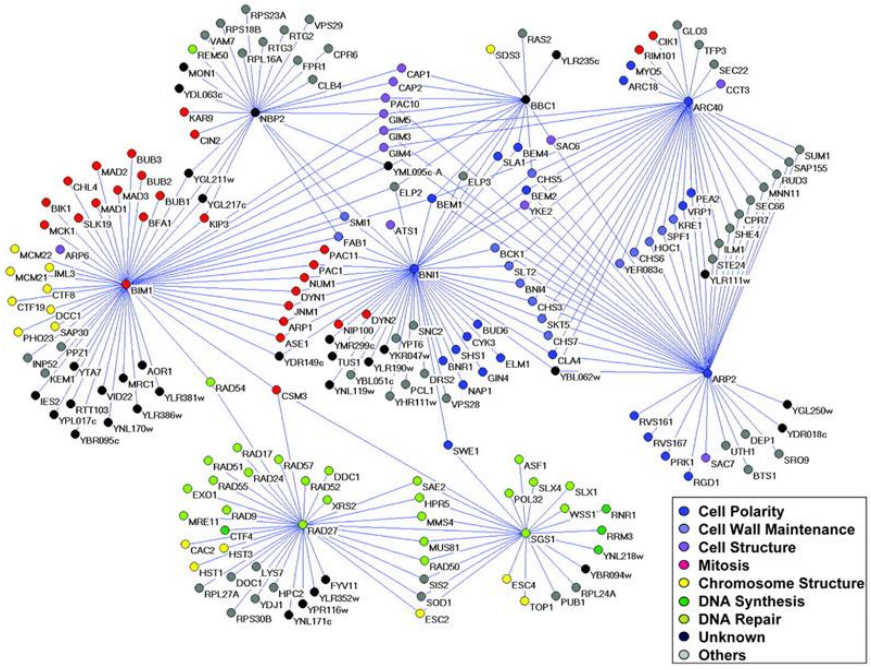
1.2.1 Παραδείγματα γράφων

Παρακάτω ακολουθούν τέσσερα παραδείγματα γράφων τόσο από την καθημερινή μας ζωή όσο και από τον επιστημονικό χώρο.

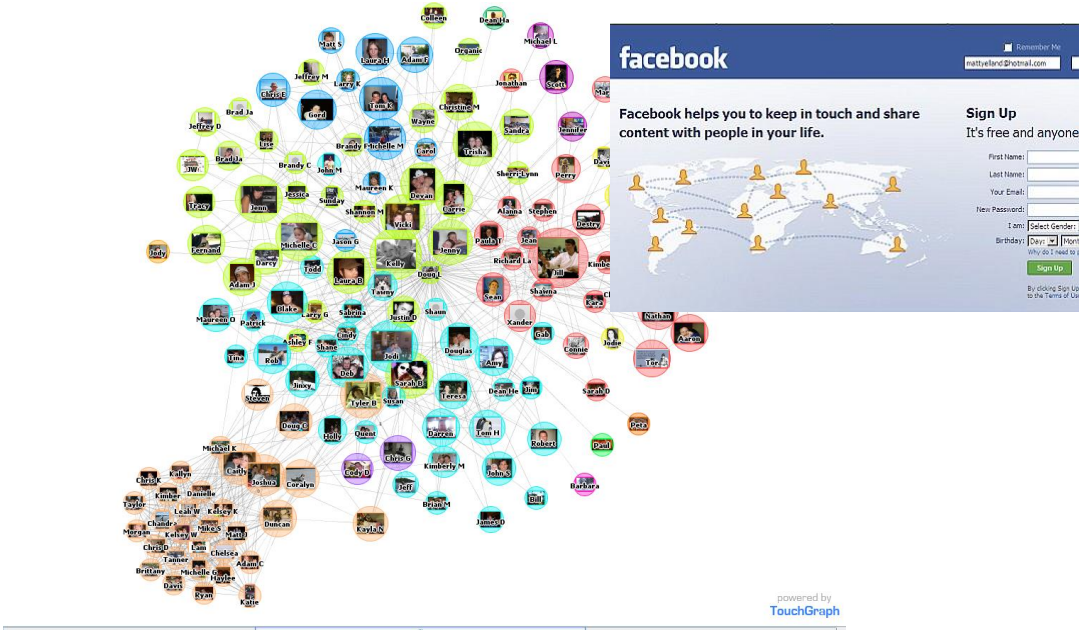
Σχήμα 1.6: GPS – Πλοήγηση (Εύρεση Ελαχίστων Διαδρομών)



Σχήμα 1.7: Χάρτης επιστήμης (science map)



Σχήμα 1.8: Δίκτυο γενετικής αλληλεπίδρασης



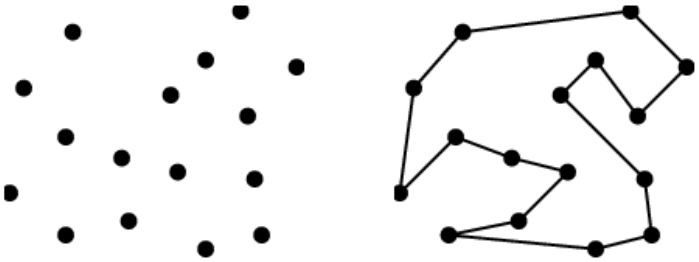
Σχήμα 1.9: Κοινωνικά δίκτυα

1.3 Διάφορα προβλήματα

Στην συνέχεια θα αναφερθούμε σε κάποια συνηθισμένα προβλήματα όσον έχουν να κάνουν με τα γραφήματα, δίνοντας περισσότερο βάρος στα ταιριάσματα που αφορούν το θέμα της διπλωματικής εργασίας.

1.3.1 Περιπλανώμενος (πλανόδιος) πωλητής (TSP)

Το πρόβλημα του πλανόδιου πωλητή περιλαμβάνει την εύρεση της (σχεδόν) μικρότερης διαδρομής που ενώνει ένα αριθμό περιοχών -πιθανόν εκατοντάδες- όπως δηλαδή κάνει ένας περιπλανώμενος πωλητής που επισκέπτεται διάφορες πόλεις με σκοπό να πουλήσει τα προϊόντα του.

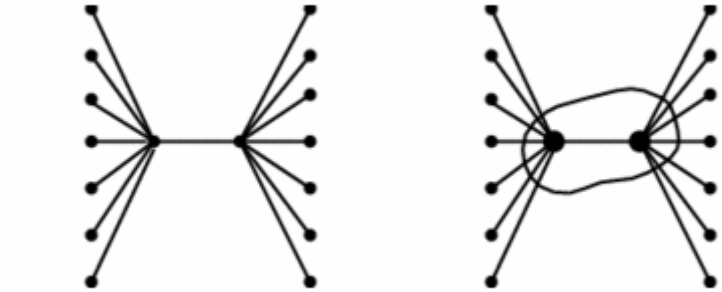


είσοδος έξοδος

Σχήμα 1.10: Το πρόβλημα του πλανόδιου πωλητή

1.3.2 Κάλυψη κορυφών (vertex cover)

Η κάλυψη κορυφών είναι μια ειδική περίπτωση του γενικότερου προβλήματος της κάλυψης συνόλου, που παίρνει ως είσοδο μια αυθαίρετη ομάδα από υποσύνολα S του σύνολο U. Επιδιώκουμε το μικρότερο υποσύνολο των υποσυνόλων από S του οποίου η ένωση είναι U. Η κάλυψη κορυφών προκύπτει σε πολλές εφαρμογές, συμπεριλαμβανομένων της ελαχιστοποίησης της Boolean λογικής.

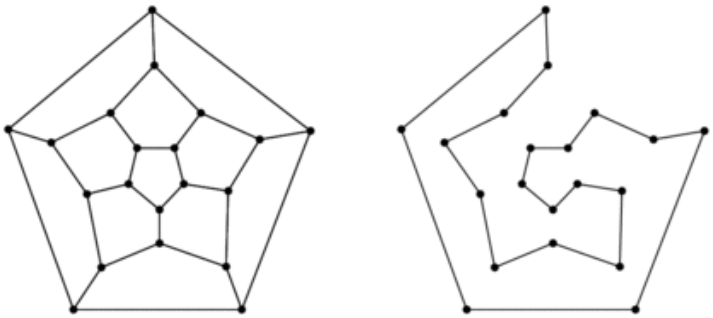


είσοδος έξοδος

Σχήμα 1.11: Το πρόβλημα της κάλυψης κορυφών

1.3.3 Κύκλος του Hamilton

Προς τιμή του Hamilton , ένας κύκλος σε ένα γράφημα, καλείται κύκλος του Hamilton (Hamilton cycle) εάν περιέχει κάθε κορυφή του γραφήματος ακριβώς μια φορά (με εξαίρεση βέβαια την αρχική και τη τελική που συμπίπτουν).



είσοδος έξοδος

Σχήμα 1.12: Κύκλος του Hamilton

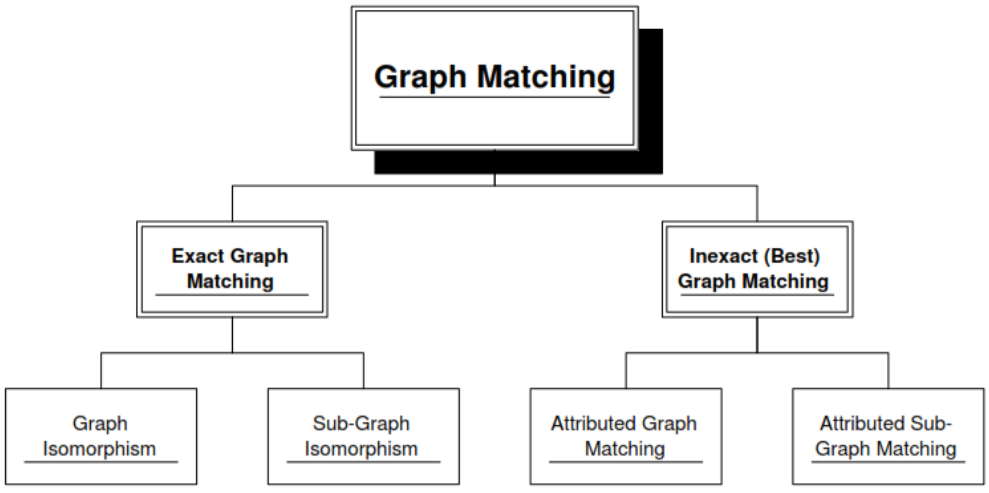
1.3.4 Ταιριάσματα (matchings)

Το ταίριασμα γραφημάτων θεωρείται ένα από τα πιο πολύπλοκα προβλήματα στην αναγνώριση αντικειμένων στην υπολογιστική όραση. Η πολυπλοκότητα του οφείλεται στην συνδυαστική του φύση. Στο σχήμα 1.13 γίνεται μια ταξινόμηση στα διαφορετικά είδη ταιριασμάτων.

Έστω γράφημα G(V,E). Ένα επικαλύπτον (spanning) υπογράφημα όπου όλες οι κορυφές έχουν βαθμό μικρότερο ή ίσο του k-παράγοντας του G(k-factor). Ένας k-παράγοντας ονομάζεται τέλειος (perfect) όταν όλες οι κορυφές έχουν βαθμό ακριβώς k. Οι πιο σημαντικοί παράγοντες ενός γραφήματος είναι οι 1-παράγοντες.

Οι 1-παράγοντες του γραφήματος G ονομάζονται ταιριάσματα. Ισοδύναμα, ένα υποσύνολο ακμών M⊆ E ονομάζεται ταίριασμα του G όταν κάθε κορυφή εφάπτεται σε μία το πολύ ακμή του M (με απλά λόγια, οι ακμές του M δεν έχουν κοινά άκρα).

Ένα ταίριασμα ονομάζεται τέλειο (perfect matching) όταν όλες οι κορυφές έχουν ταίρι στο M



Σχήμα 1.13: Η ταξινόμηση όλων των τύπων των ταιριασμάτων σε δύο κύριες κατηγορίες (ακριβής αντιστοίχιση (τέλειο ταίριασμα) γραφημάτων και το μη τέλειο ταίριασμα γραφημάτων).

Το τέλειο ταίριασμα υπογραφήματος(ισομορφισμός) έχει αποδειχθεί ότι είναι τύπου NP-complete[1]. Ωστόσο, μερικοί τύποι γραφημάτων μπορεί να έχουν χαμηλότερη πολυπλοκότητα. Για παράδειγμα, η ιδιαίτερη περίπτωση στην οποία το μεγάλο γράφημα είναι ένα δάσος και το μικρό που πρέπει να ταιριάξει είναι ένα δέντρο είναι γνωστό ότι λύνεται με αλγόριθμο πολυωνυμικού χρόνου.

Στο πρόβλημα ταιριάσματος γραφήματος που εξετάζεται στην συγκεκριμένη εργασία δίνονται δύο γραφήματα $G_1=(N_1,E_1)$ και $G_2=(N_2,E_2)$, όπου N_1 και N_2 τα σύνολα κορυφών και E_1,E_2 τα σύνολα ακμών των δύο γραφημάτων. Το ζητούμενο είναι να βρεθεί μια αντιστοίχιση f(n) της κάθε κορυφής $n \in G_1$ σε μια κορυφή $n' \in G_2$ έτσι ώστε για κάθε ακμή $e=(n_i,n_j) \in E_1$ να υπάρχει ακμή $e'=(n'_i,n'_j) \in E_2$ τέτοια ώστε $f(n_i)=n'_i$, και $f(n_j)=n'_j$.

2 Προβλήματα ικανοποίησης περιορισμών

Στο τρέχον κεφάλαιο της εργασίας παρουσιάζεται ένα βασικό θεωρητικό υπόβαθρο πάνω στα προβλήματα ικανοποίησης περιορισμών,. Στην συνέχεια περιγράφεται το παράδειγμα των N-βασιλισσών [5] για την καλύτερη κατανόηση της θεωρίας.

2.1 Θεωρητικό υπόβαθρο

Ο προγραμματισμός με περιορισμούς είναι ένας από τους βασικούς τομείς της τεχνίτης νοημοσύνης . Τα προβλήματα Ικανοποίησης Περιορισμών εμφανίζονται σε πολλές περιοχές έρευνας, όπως η τεχνητή όραση, ο καταμερισμός πόρων, ο χρονοπρογραμματισμός διαδικασιών κ.α.

Τα προβλήματα ικανοποίησης περιορισμών αποτελούνται από ένα σύνολο περιορισμένων μεταβλητών(constraint variables) , κάθε μια από τις οποίες παίρνει τιμές από ένα πεδίο (domain) τιμών. Επίσης υπάρχει και ένα σύνολο περιορισμών (constraints), που περιορίζουν τις τιμές που μπορούν να έχουν ταυτόχρονα οι μεταβλητές. Οι περιορισμοί στην ουσία είναι οι σχέσεις μεταξύ των μεταβλητών του προβλήματος. Το ζητούμενο είναι να ανατεθούν τέτοιες τιμές στις μεταβλητές, από το πεδίο τιμών καθεμίας ώστε να μην παραβιάζεται κανένας περιορισμός .
Πριν δοθεί ο ορισμός για το Πρόβλημα Ικανοποίησης Περιορισμών, ακολουθούν μερικοί βοηθητικοί ορισμοί

2.1.1 Πεδίο τιμών

[1] **NP-Complete:** NP-hard πρόβλημα που ανήκει στην κλάση NP
NP-hard: Προβλήματα τουλάχιστον τόσο «δύσκολα» όσο οποιοδήποτε πρόβλημα της κλάσης NP
NP: Πολυωνυμική επιβεβαίωση λύσης ή Μη- ντετερμινιστική πολυωνυμική πολυπλοκότητα χρόνου
P: Προβλήματα Πολυωνυμικής Πολυπλοκότητας Χρόνου

Το πεδίο τιμών μιας μεταβλητής, είναι το σύνολο όλων των δυνατών τιμών που μπορούν να ανατεθούν στη μεταβλητή αυτή. Συμβολίζουμε με Dx το πεδίο τιμών της μεταβλητής x. Το πεδίο τιμών μιας μεταβλητής μπορεί να είναι είτε πεπερασμένο είτε άπειρο.

2.1.2 Ανάθεση

Μια ανάθεση, είναι ένα σύνολο ($<x_1,u_1>,<x_2,u_2>,\ldots,<x_n,u_n>$) από ζευγάρια μεταβλητών –τιμών. Σε κάθε ζευγάρι $<x_i,u_i>$, το x_i αντιπροσωπεύει κάποια μεταβλητή του προβλήματος, και το u_i την τιμή που ανατίθεται στη μεταβλητή αυτή. Αν μια ανάθεση δεν παραβιάζει κανέναν περιορισμό που αφορά τις μεταβλητές της, τότε ονομάζεται συνεπής (consistent).
Μια ανάθεση λέγεται πλήρης, αν περιέχει όλες τις μεταβλητές του προβλήματος.

2.1.3 Περιορισμός

Ένας περιορισμός πάνω σε ένα σύνολο μεταβλητών, περιορίζει τις τιμές που μπορούν να ανατεθούν ταυτόχρονα σε αυτές τις μεταβλητές. Στην πράξη συνήθως χρησιμοποιούνται άλλες μορφές αναπαράστασης όπως συναρτήσεις, ανισότητες, πίνακες κτλ. Ένας περιορισμός που αναφέρεται στο σύνολο μεταβλητών S, συμβολίζεται ως C_S

Ανάλογα με τον αριθμό των μεταβλητών που εμπλέκονται σε έναν περιορισμό, ο περιορισμός αυτός κατατάσσεται στις εξής κατηγορίες:

- αν εμπλέκεται μόνο μια μεταβλητή, ο περιορισμός είναι μοναδιαίος (unary),
- αν εμπλέκονται δύο μεταβλητές, ο περιορισμός είναι δυαδικός (binary),
- αν εμπλέκονται περισσότερες μεταβλητές, ο περιορισμός είναι ανώτερης τάξης (higher order)

2.1.4 Ορισμός ενός Προβλήματος Ικανοποίησης Περιορισμών

Ένα πρόβλημα ικανοποίησης περιορισμών ορίζεται σαν μια τριάδα $P = (Z, D, C)$:

- Όπου Z είναι ένα πεπερασμένο σύνολο μεταβλητών $\{x_1,x_2,\ldots,x_n\}$
- D μια συνάρτηση που αντιστοιχίζει κάθε μεταβλητή x_i που ανήκει στο Z, στο πεδίο τιμών της D_{xi}
- C είναι ένα πεπερασμένο σύνολο περιορισμών πάνω σε υποσύνολα του Z.

Το ζητούμενο είναι η εύρεση μιας πλήρους ανάθεσης που ικανοποιεί ταυτόχρονα όλους τους περιορισμούς του προβλήματος. Η ανάθεση αυτή καλείται λύση του προβλήματος.

Ένα τυπικό Πρόβλημα Ικανοποίησης Περιορισμών, περιλαμβάνει πολλές μεταβλητές. Αυτό έχει σαν αποτέλεσμα, ο χώρος αναζήτησης που προκύπτει να είναι αρκετά μεγάλος και να μπορεί να φτάσει το μέγεθος του καρτεσιανού γινομένου των πεδίων τιμών όλων των μεταβλητών. Αυτό οδηγεί στο φαινόμενο της συνδυαστικής έκρηξης (combinatorial explosion), το οποίο έχει σαν

επακόλουθο τη χρονοβόρα αναζήτηση λύσης.

2.1.5 Το πρόβλημα των 8-Βασιλισσών

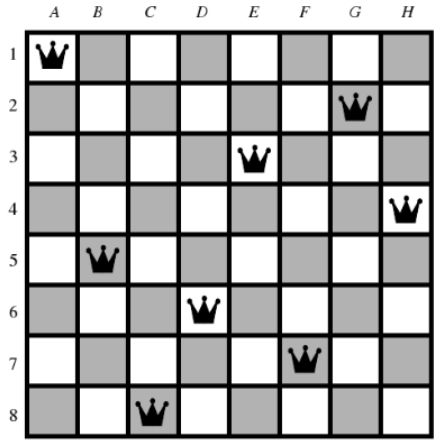
Ίσως το πιο κλασικό παράδειγμα Προβλήματος Ικανοποίησης Περιορισμών, είναι αυτό των Ν-Βασιλισσών. Το πρόβλημα αυτό χρησιμοποιείται αρκετά συχνά για την παρουσίαση αλγορίθμων επίλυσης Προβλημάτων Ικανοποίησης Περιορισμών.

Το πρόβλημα, δεδομένου ενός θετικού ακεραίου Ν, ανάγεται στην τοποθέτηση Ν Βασιλισσών σε Ν διαφορετικά τετράγωνα σε μια σκακιέρα μεγέθους Ν x Ν, με τέτοιο τρόπο ώστε καμία βασίλισσα να μην απειλεί κάποια άλλη. Δύο βασίλισσες απειλούνται αν βρίσκονται στην ίδια γραμμή, στην ίδια στήλη ή στην ίδια διαγώνιο . Το σχήμα 2.1 δείχνει μια από τις λύσεις για το πρόβλημα των 8-Βασιλισσών.
Ένας τρόπος μοντελοποίησης του προβλήματος των 8-Βασιλισσών, σαν πρόβλημα ικανοποίησης περιορισμών είναι η χρήση οκτώ μεταβλητών που αντιπροσωπεύουν τις βασίλισσες σε κάθε γραμμή:

$$Z = \{Q_1, Q_2, \ldots, Q_8\}$$

Κάθε μία από τις οκτώ μεταβλητές αυτές, έχει σαν πεδίο τιμών τις οκτώ στήλες της σκακιέρας:

$$D_{Q1} = D_{Q2} = \ldots = D_{Q8} = \{1,2,3,4,5,6,7,8\}$$



Σχήμα 2.1: Μια από τις λύσεις για το πρόβλημα των 8-Βασιλισσών.

Το γεγονός ότι κάθε μεταβλητή αντιπροσωπεύει κάποια γραμμή της σκακιέρας, εξασφαλίζει ότι δύο βασίλισσες δε θα βρίσκονται στην ίδια γραμμή. Ο περιορισμός για τις στήλες, μπορεί να εκφραστεί ως:

$$C_1 : \forall\, i, j : Q_i \neq Q_j$$

Ο περιορισμός για τις διαγώνιους, μπορεί να εκφραστεί ως:

$$C_2 : \forall\, i, j : \text{αν } Q_i = a \text{ και } Q_j = b \text{ τότε } i - j \neq a - b \text{ και } i - j \neq b - a$$

Με αυτή την μοντελοποίηση, υπάρχουν 88 πιθανοί συνδυασμοί τιμών για τις οκτώ μεταβλητές. Στο σημείο αυτό πρέπει να τονιστεί, ότι σε κάθε πρόβλημα είναι δυνατόν να υπάρχουν παραπάνω από μια δυνατή μοντελοποίηση. Για παράδειγμα για το πρόβλημα των 8-βασιλισσών, μια εναλλακτική μοντελοποίηση θα ήταν να υπάρχουν οκτώ μεταβλητές που να αντιπροσωπεύουν τις βασίλισσες και το πεδίο τιμών τους να είναι το [1…64] δηλαδή τετράγωνα της σκακιέρας. Σε αυτή τη μοντελοποίηση, υπάρχουν 648 πιθανοί συνδυασμοί τιμών για τις οκτώ μεταβλητές. Γίνεται φανερό ότι η μοντελοποίηση που θα επιλεγεί, μπορεί να επηρεάσει την απόδοση του αλγορίθμου επίλυσης

3 Τοπική Αναζήτηση

Η Τοπική Αναζήτηση είναι πολλές φορές η μόνη ρεαλιστική αντιμετώπιση, έτσι ώστε να βρεθεί λύση μέσα στα χρονικά πλαίσια που απαιτεί μια εφαρμογή. Οι αλγόριθμοι της Τοπικής Αναζήτησης θυσιάζουν την πληρότητα για να κερδίσουν από άποψη απόδοσης.

Η βασική ιδέα, πάνω στην οποία στηρίζονται όλοι αυτοί οι αλγόριθμοι, είναι η κατασκευή αρχικά μιας υποψήφιας λύσης, είτε με κάποιο τυχαίο τρόπο είτε χρησιμοποιώντας κάποια ευριστική μέθοδο, και στη συνέχεια η επαναληπτική βελτίωσή της μέσω μικρών αλλαγών σε κάθε βήμα της επανάληψης. Η κύρια διαφορά όλων αυτών των αλγορίθμων, έγκειται στο είδος των βελτιώσεων που επιχειρούν σε κάθε βήμα και επίσης στον τρόπο που αντιμετωπίζουν περιπτώσεις, όπου δεν είναι δυνατή μία άμεση βελτίωση της υποψήφιας λύσης.
Οι περισσότεροι αλγόριθμοι, χρησιμοποιούν κάποιο είδος τυχαιότητας για να αντιμετωπίσουν πιθανές περιπτώσεις μη ικανοποιητικών υποψήφιων λύσε- ων. Για τον λόγο αυτό αναφέρονται και ως στοχαστικές μέθοδοι (stochastic methods).
Οι πιο πολλοί αλγόριθμοι Τοπικής Αναζήτησης είναι εννοιολογικά απλοί – αν και μπορεί να απαιτούν προσεκτική υλοποίηση και κάπως πολύπλοκες δομές δεδομένων– και επίσης είναι αρκετά εύκολο να προσαρμοστούν σε αλλαγές στις προδιαγραφές του προβλήματος. Μπορούν να χρησιμοποιηθούν σε πολύπλοκα προβλήματα που μπορεί να μην έχουν πλήρως καθορισμένες τις προδιαγραφές τους από την αρχή. Τέλος στην γενική περίπτωση έχουν σταθερή πολυπλοκότητα χώρου.

Οι βασικοί αλγόριθμοι τοπικής αναζήτησης είναι οι εξής:

- Αναζήτηση με Αναρρίχηση Λόφων
- Προσομοιωμένη Ανόπτηση
- Τοπική ακτινική αναζήτηση
- Γενετικοί Αλγόριθμοι

Στο τρέχον κεφάλαιο θα δούμε το πώς ορίζεται ένας αλγόριθμος τοπικής αναζήτησης δεδομένου ενός προβλήματος ικανοποίησης περιορισμών, στην

συνέχεια περιγράφεται το παράδειγμα των 8-Βασιλισσών, και τέλος παρουσιάζεται ο αλγόριθμος Hill Climbing και οι παραλλαγές αυτού.

3.1 Τυπικός Ορισμός

Οι αλγόριθμοι Τοπικής Αναζήτησης ξεκινάνε από μια αρχική θέση αναζήτησης (initial search position) και σε κάθε βήμα επιλέγεται από την τοπική γειτονία (local neighbourhood) μία επόμενη θέση για μετάβαση. Αυτή η διαδικασία επαναλαμβάνεται μέχρις ότου να γίνει μετάβαση σε θέση που αντιστοιχεί σε λύση του προβλήματος. Για την αποφυγή αποτελμάτωσης στην διαδικασία της αναζήτησης, αρκετοί αλγόριθμοι χρησιμοποιούν στοιχεία τυχαιότητας, τόσο στην δημιουργία της αρχικής θέσης όσο και στα βήματα κατά την διαδικασία της αναζήτησης.

΄Ενας Αλγόριθμος Τοπικής Αναζήτησης, δεδομένου ενός Προβλήματος Ικανοποίησης Περιορισμών P , ορίζεται [5] από τις παρακάτω συνιστώσες:

• τον χώρο αναζήτησης S(P), δηλαδή το σύνολο των υποψήφιων λύσεων,

• το σύνολο των λύσεων του προβλήματος S’(P) ⊆ S(P),

• μια σχέση που δηλώνει μια γειτονιά στο S(P), N (P) ⊆ S(P) × S(P) –η γειτονία, καθορίζει τις θέσεις που είναι προσβάσιμες σε ένα βήμα από την τρέχουσα

• ένα σύνολο καταστάσεων μνήμης M (P) –στους αλγόριθμους που δεν απαιτείται κάτι πολύπλοκο από άποψη μνήμης, αυτό ,μπορεί να αποτελείται μόνο από την τρέχουσα κατάσταση, ενώ σε άλλους μπορεί να περιέχει πληροφορίες για καταστάσεις πέραν της τρέχουσας,

• μια συνάρτηση αρχικοποίησης init(P) : ∅ → D(S(P) × M (P)), η οποία καθορίζει την κατανομή των πιθανοτήτων των θέσεων του χώρου αναζήτησης σε συνδυασμό με τις καταστάσεις μνήμης· η συνάρτηση αυτή καθορίζει τι γίνεται στο στάδιο της αρχικοποίησης του αλγορίθμου,

• μια συνάρτηση επόμενου βήματος step(P) : S(P) × M (P) → D(S(P) × M (P)), η οποία δεδομένου μίας τρέχουσας θέσης και μίας κατάστασης μνήμης, καθορίζει την κατανομή των πιθανοτήτων των γειτονικών θέσεων σε συνδυασμό με τις καταστάσεις μνήμης· η συνάρτηση αυτή καθορίζει τι γίνεται σε κάθε βήμα της αναζήτησης,

• ένα κατηγόρημα τερματισμού terminate(P) : S(P)×M (P) → D({true, f alse}), το οποίο αντιστοιχίζει κάθε ζευγάρι θέσης και κατάστασης μνήμης σε μια κατανομή πιθανοτήτων αληθοτιμών, που δείχνει την πιθανότητα η αναζήτηση να τερματίσει αφού φτάσει σε αυτή την θέση με αυτή την κατάσταση μνήμης.

Στα παραπάνω, με D(S) συμβολίζεται το σύνολο των κατανομών πιθανοτήτων για

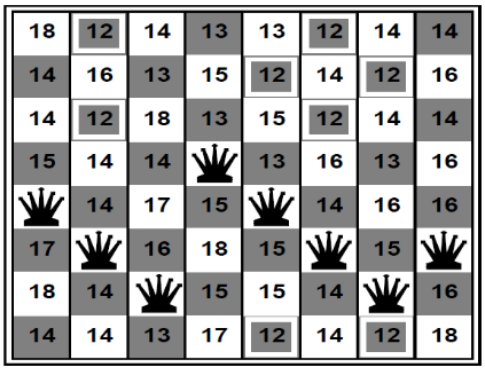
τα στοιχεία του συνόλου S. Τυπικά μία κατανομή πιθανοτήτων D ∈ D(S) είναι μια συνάρτηση D : S $\rightarrow R_0^+$ που αντιστοιχεί τα στοιχεία του S στις αντίστοιχες πιθανότητες τους.

3.1.1 Το πρόβλημα των 8-Βασιλισσών

Αν εφαρμόσουμε την τοπική αναζήτηση στο παράδειγμα των 8-Βασιλισσών, έχουμε:

- υποψήφιες λύσεις: οποιαδήποτε τοποθέτηση των 8 βασιλισσών στη σκακιέρα,
- γειτονιές: σε κάθε βήμα της αναζήτησης, υπάρχουν 8 * 7 = 56 πιθανές γειτονικές θέσεις[2],
- κριτήριο τερματισμού: 8 βασίλισσες τοποθετημένες στη σκακιέρα, έτσι ώστε να μην απειλείται καμία,
- συνάρτηση αξιολόγησης: το πλήθος των ζευγαριών βασιλισσών που απειλούνται μεταξύ τους, στην τρέχουσα θέση της αναζήτησης

Μια πιθανή θέση της αναζήτησης μαζί με τη βαθμολόγηση[3] της γειτονιάς της, δίνεται στο σχήμα 3.1



Σχήμα 3.1: Η τοπική αναζήτηση στο παράδειγμα των 8-Βασιλισσών

3.2 Αλγόριθμος Επαναληπτικής Βελτίωσης (Hill Climbing)

Ο πιο απλός αλγόριθμος Τοπικής Αναζήτησης, είναι αυτός της

[2] Σε κάθε βήμα επιλέγεται αρχικά 1 βασίλισσα από τις 8 και στη συνέχεια επιλέγεται 1 από τις 7 πιθανές θέσεις στις οποίες μπορεί να μετακινηθεί αυτή, καθώς η θέση στην οποία βρίσκεται ήδη δεν λαμβάνεται υπόψιν. Έτσι για κάθε βήμα, με δεδομένο την μετακίνηση μόνο μίας βασίλισσας, υπάρχουν 8 * 7 = 56 πιθανές διαφορετικές καταστάσεις που μπορεί να προκύψουν.

[3] Η τιμή της συνάρτησης αξιολόγησης για την παραπάνω υποψήφια λύση είναι 17. Οι αριθμοί σε κάθε τετράγωνο της σκακιέρας, δείχνουν την τιμή της συνάρτησης αξιολόγησης, αν μετακινηθεί σε αυτό η βασίλισσα της ίδιας στήλης.

Επαναληπτικής Βελτίωσης (Iterative Improvement) γνωστός και ως Ανάβαση Λόφου (Hill- Climbing). Σε κάθε βήμα της αναζήτησης, επιλέγεται μια θέση από την τρέχουσα γειτονιά, που να είναι καλύτερη από την τρέχουσα θέση. Δηλαδή επιλέγεται μια θέση $s^0 \in N(s)$ με $g(s^0) < g(s)$, όπου s είναι η τρέχουσα θέση της αναζήτησης. Υπάρχουν διάφορες ευριστικές μέθοδοι για την επιλογή της γειτονικής θέσης. Στην Επαναληπτική Καλύτερη-Βελτίωση (Iterative Best-Improvement), επιλέγεται η γειτονική θέση s^0 που έχει την ελάχιστη τιμή $g(s^0)$ στην γειτονιά N (s). Αν υπάρχουν περισσότερες από μια τέτοιες θέσεις, η επιλογή γίνεται με τυχαίο ομοιόμορφο τρόπο. Στην Επαναληπτική Πρώτη-Βελτίωση (Iterative First-Improvement), οι θέσεις της γειτονιάς ελέγχονται με βάση κάποια συγκεκριμένη σειρά και επιλέγεται η πρώτη θέση που θα είναι καλύτερη από την τρέχουσα.

Οι παραλλαγές της Επαναληπτικής Βελτίωσης, αποτελούν τη βάση για τους περισσότερους αλγορίθμους Τοπικής Αναζήτησης.

Για καλύτερη κατανόηση παραθέτουμε παρακάτω τον αλγόριθμο σε βήματα και στη συνέχεια σε ψευδοκώδικα την έκδοση, της πλέον απότομης ανάβασης, της πιο βασικής τεχνικής τοπικής αναζήτησης.

1. Η αρχική κατάσταση είναι η τρέχουσα κατάσταση.
2. Αν η κατάσταση είναι μια τελική τότε ανέφερε την λύση και σταμάτησε.
3. Βρες τις γειτονικές καταστάσεις
4. Επέλεξε την καλύτερη (κόμβο με μεγαλύτερη τιμή VALUE).
5. Επέστρεψε στο βήμα 2.

```
function Hill_Climbing (problem, Eval-fn)
returns a state that is a local maximum
inputs: problem, a search problem
        Eval-fn, an evaluation function
local variables: current, next, nodes
                 neighbors, a set of nodes
        current < MakeNode(InitialState[problem])
        loop do
                neighbors < SuccessorStates(current)
                if neighbors = 0 return current
                Eval-fn(neighbors)
                next < a highest-valued node of neighbors
                if the value of next is less or equal to the value of
                current
                        return current
        end
```

Σχήμα 3.2: Υλοποίηση του αλγορίθμου Hill Climbing

3.3 Ο άπληστος αλγόριθμος

Η αναρρίχηση λόφων λέγεται και άπληστη τοπική αναρρίχηση, επειδή αρπάζει μία γειτονική κατάσταση χωρίς να σκεφτεί παραπέρα που θα πάει στη συνέχεια. Οι άπληστοι αλγόριθμοι συχνά αποδίδουν πολύ καλά. Η

αναρρίχηση λόφων συχνά κάνει μεγάλη πρόοδο προς μια λύση, επειδή μπορεί να βελτιώσει μια κακή κατάσταση. Δυστυχώς, ο αλγόριθμος συχνά παγιδεύεται για τους παρακάτω λόγους:

3.3.1 Τοπικά Βέλτιστα (local optima)

Η αναζήτηση μπορεί να κολλήσει σε μια κατάσταση της οποίας όλες οι γειτονικές έχουν χειρότερο κόστος, αλλά δεν είναι η βέλτιστη.

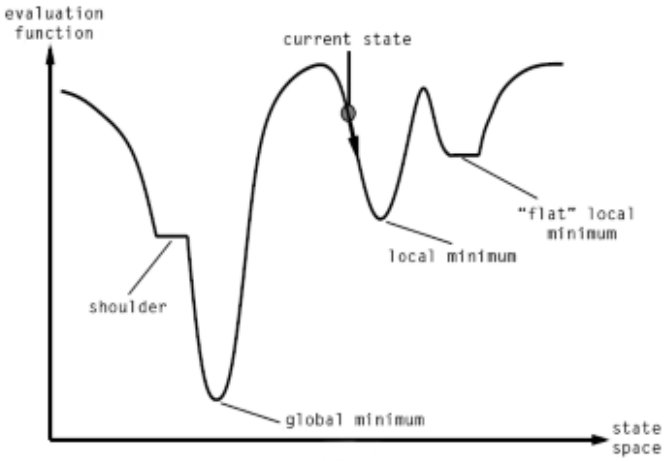
3.3.2 Οροπέδια (plateaus)

Αν όλα τα γειτονικά σημεία έχουν το ίδιο κόστος η επιλογή γίνεται τυχαία.

3.3.3 Κορυφογραμμές (ridges)

Οι κορυφογραμμές είναι ψηλά και εντοπίζονται εύκολα, όμως από εκεί και πέρα η αναζήτηση είναι πολύ αργή γιατί αποτελούνται από μια ακολουθία τοπικών βέλτιστων. Σε αυτές τις περιπτώσεις ο αλγόριθμος φτάνει σε ένα σημείο από το οποίο δεν μπορεί να κάνει καμία πρόοδο.

Ο αλγόριθμος της πλέον απότομης ανάβασης όταν φτάσει σε ένα οροπέδιο, όπου η καλύτερη διαδοχική τιμή έχει την ίδια τιμή με την τρέχουσα κατάσταση, σταματάει. Μια βελτίωση του αλγορίθμου θα ήταν να επιτρέπονται οι πλάγιες κινήσεις με την ελπίδα ότι το οροπέδιο είναι όπως είδαμε στην εικόνα 1. Όμως, στη περίπτωση που το οροπέδιο αποτελεί ένα επίπεδο τοπικό μέγιστο, ο αλγόριθμος θα εγκλωβίζεται σε έναν ατέρμονα βρόχο. Οπότε μια συνηθισμένη λύση είναι να τίθεται ένα όριο στον αριθμό των διαδοχικών πλάγιων κινήσεων που επιτρέπονται.



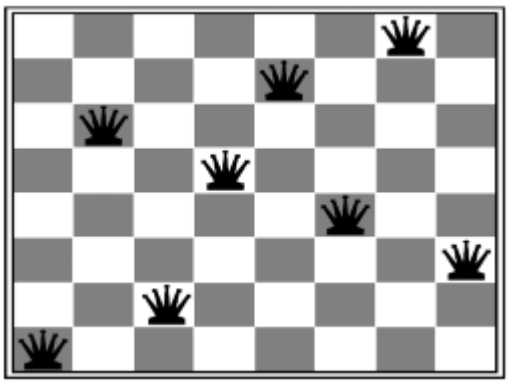
Σχήμα 3.3: Ιδιαιτερότητες της μορφολογίας του χώρου καταστάσεων

3.4 Το πρόβλημα των 8-Βασιλισσών

Η εφαρμογή [3] του γενικού αλγόριθμου του Hill Climbing στο πρόβλημα των 8-Βασιλισσών δίνει τα ακόλουθα αποτελέσματα:

- επιτυχής εύρεση λύσης στο 14% των προβλημάτων, με μέσο όρο μετά από 4 βήματα αναζήτησης,
- αποτυχία λόγω τοπικού ελαχίστου στο 86% των προβλημάτων, με μέσο όρο μετά από 3 βήματα αναζήτησης,

Σημείωση: Ο συνολικός χώρος καταστάσεων, αποτελείται από $8^8 = 17$ εκατομμύρια καταστάσεις



Σχήμα 3.4: Η τιμή της συνάρτησης αξιολόγησης για την παραπάνω υποψήφια λύση είναι 1. Όλες οι γειτονικές θέσεις έχουν τιμή > 1 και έτσι ο αλγόριθμος έχει συναντήσει τοπικό ελάχιστο.

3.5 Παραλλαγές του hill climbing

3.5.1 Στοχαστική αναρρίχηση Λόφων

Έχουν επινοηθεί πολλές παραλλαγές της αναρρίχησης λόφων. Η στοχαστική αναρρίχηση λόφων διαλέγει τυχαία από τις διαθέσιμες κινήσεις προς τα επάνω. Η πιθανότητα επιλογής μπορεί να διαφέρει ανάλογα με το πόσο απότομη κλήση έχει κάθε τέτοια κίνηση. Η παραλλαγή αυτή συνήθως συγκλίνει πιο αργά από την μέθοδο της πλέον απότομης ανάβασης, αλλά σε μερικά τοπία καταστάσεων βρίσκει καλύτερες λύσεις.

3.5.2 Αναρρίχηση λόφων με την πρώτη επιλογή

Η αναρρίχηση λόφων με την πρώτη επιλογή υλοποιεί τη στοχαστική αναρρίχηση λόφων παράγοντας διαδοχικές καταστάσεις τυχαία, μέχρι να παραχθεί κάποια που είναι καλύτερη από την τρέχουσα κατάσταση. Η στρατηγική αυτή είναι καλή όταν μια κατάσταση έχει πολλές διαδοχικές καταστάσεις.

3.5.3 Αναρρίχηση λόφων με τυχαίες επανεκκινήσεις

Οι αλγόριθμοι αναρρίχησης λόφων που περιγράψαμε μέχρι εδώ είναι μη πλήρεις, συχνά δεν καταφέρνουν να βρουν μια κατάσταση στόχου ενώ υπάρχει, επειδή μπορεί να παγιδευτούν σε τοπικά μέγιστα. Η αναρρίχηση λόφων με τυχαίες επανεκκινήσεις αποτελεί ένα πλήρη αλγόριθμο με

πιθανότητα που αγγίζει το 1. Πραγματοποιεί μια σειρά αναζητήσεων αναρρίχησης λόφων από τυχαία παραγόμενες αρχικές καταστάσεις, σταματώντας όταν βρεθεί μια κατάσταση στόχου. Αν κάθε αναρρίχηση λόφων έχει πιθανότητα επιτυχίας, ο αναμενόμενος αριθμός βημάτων είναι ίσος με 1/p.Η επιτυχία του αλγορίθμου εξαρτάται σε μεγάλο βαθμό από το σχήμα του τοπίου του χώρου καταστάσεων. Αν υπάρχουν λίγα τοπικά μέγιστα και οροπέδια ο αλγόριθμος αναρρίχησης λόφων με τυχαίες επανεκκινήσεις θα βρει μια καλή λύση πολύ γρήγορα. Πολλά όμως πραγματικά προβλήματα έχουν ένα τοπίο με εκθετικά μεγάλο αριθμό τοπικών μεγίστων στα οποία μπορεί να παγιδευτεί, αλλά παρ'όλα αυτά ένα πολύ καλό τοπικό μέγιστο συχνά μπορεί να βρεθεί μετά από μικρό αριθμό επανεκκινήσεων.

3.5.4 Ευριστικό Ελάχιστων Συγκρούσεων

Το Ευριστικό Ελάχιστων Συγκρούσεων (Min Conflict Heuristic), ίσως να είναι από τις πιο γνωστές εκδοχές της Επαναληπτικής Βελτίωσης. Η μέθοδος αυτή, αλλάζει σε κάθε βήμα την τιμή που έχει ανατεθεί σε μια μεταβλητή, με σκοπό τη μείωση του αριθμού των περιορισμών που παραβιάζονται.

Πιο συγκεκριμένα, η μέθοδος αυτή κατασκευάζει αρχικά μια υποψήφια λύση, αναθέτοντας σε κάθε μεταβλητή του προβλήματος μία τιμή από το πεδίο τιμών της, που επιλέγεται με ομοιόμορφο τυχαίο τρόπο. Στη συνέχεια, σε κάθε βήμα της επανάληψης, αρχικά επιλέγεται τυχαία μία μεταβλητή x από το σύνολο συγκρούσεων (conflict set) K (a), δηλαδή το σύνολο των μεταβλητών που εμφανίζονται σε κάποιον περιορισμό που παραβιάζεται, με βάση την τρέχουσα ανάθεση a. Έπειτα επιλέγεται μία τιμή v από το πεδίο τιμών της x, τέτοια ώστε αναθέτοντας την v στην x να ελαχιστοποιείται ο αριθμός των περιορισμών που παραβιάζονται. Αν υπάρχουν περισσότερες από μία τέτοιες τιμές v, επιλέγεται μία με τυχαίο τρόπο. Η αναζήτηση τερματίζει όταν βρεθεί μία λύση ή όταν συμπληρωθεί ένας προκαθορισμένος μέγιστος αριθμός επαναλήψεων.

Η συγκεκριμένη εκδοχή του hill climbing είναι αυτή που χρησιμοποιείται στην υλοποίηση της αναζήτησης υπογραφημάτων στην διπλωματική εργασία.

```
function MinConflictHeuristic(problem, maxSteps)
        current ← Initialise(problem)
        for  step ← 1 to maxSteps do
                if a satisfies all constrains  of problem then
                        return current                          . Βρέθηκε λύση!
                end if
                x ← randomly  selected variable
                        from conflict set K (current)
                v ← randomly  selected value
                        from the domain of x such that  assigning v to x,
                        minimises the number of unsatisfied  constraints
                current ← current with v assigned to x
        end for
        return null                                             . Αποτυχία
end function
```

Σχήμα 3.5: Ο βασικός αλγόριθμος Ελάχιστων Συγκρούσεων. Κάθε τυχαία επιλογή έχει ομοιόμορφη κατανομή πιθανοτήτων.

4 Ανάλυση εφαρμογής

Σε αυτό το κεφάλαιο γίνεται η ανάλυση του προγράμματος. Αρχικά γίνεται αναφορά στις κλάσεις, στην συνέχεια παρουσιάζονται οι Δομές δεδομένων και οι μεταβλητές του προβλήματος, αναλύονται οι Μέθοδοι του προβλήματος. Το υποκεφάλαιο 4.2 περιέχει τη Λειτουργία του Προγράμματος. Γίνεται η Περιγραφή της λειτουργίας του αλγορίθμου Ελαχίστων Συγκρούσεων, και παρατίθεται ένα παράδειγμα του αλγορίθμου. Στο τέλος του κεφαλαίου υπάρχει το διάγραμμα ροής δεδομένων του κυρίου προγράμματος.

4.1 Υλοποίηση Αλγορίθμου

Στην εργασία υλοποιήθηκε ο Hill Climbing Local Search Algorithm με την εφαρμογή του αλγόριθμου Ελάχιστων Συγκρούσεων σε πρόβλημα ικανοποίησης περιορισμών. Το πρόγραμμα υλοποιήθηκε με γλώσσα προγραμματισμού JAVA και χρησιμοποιήθηκε το Java Development Kit 1.8.

4.1.1 Μοντελοποίηση προβλήματος ως CSP

Το πρόβλημα ταιριάσματος δύο γραφημάτων $G_1=(N_1,E_1)$ και $G_2=(N_2,E_2)$ μπορεί να μοντελοποιηθεί ως πρόβλημα ικανοποίησης περιορισμών ως εξής:

- Για κάθε κορυφή $n \in G_1$ δημιουργείται μια μεταβλητή x.
- Για κάθε μεταβλητή x το πεδίο τιμών της είναι της μορφής $D(x) = \{n \mid n \in G_2\}$. Δηλαδή το D(x) περιλαμβάνει όλες τις κορυφές του G_2.
- Για κάθε ακμή $e=(n_i,n_j) \in E_1$ υπάρχει ένας δυαδικός περιορισμός ανάμεσα στις μεταβλητές που αντιστοιχούν στις κορυφές n_i και n_j. Αν x και y είναι αυτές οι μεταβλητές, τότε σύμφωνα με τον περιορισμό (x,y) πρέπει για κάθε ζεύγος αναθέσεων x=n και y=n' να ισχύει ότι $(n,n') \in E_2$.

4.1.2 Κλάσεις Προγράμματος

Στο κώδικα του προγράμματος εμπεριέχονται υλοποιημένες κλάσεις από το βιβλίο "Algorithms 4th Edition" του Robert Sedgewick και του Kevin Wayne οι οποίες είναι :

1. Graph.java
2. GraphGenerator.java
3. In.java

4. MinPQ.java
5. SET.java
6. ST.java
7. Stack.java
8. StdIn.java
9. StdOut.java
10. StdRandom.java
11. Bag.Java

Αυτές που χρησιμοποιήθηκαν άμεσα είναι η **Graph.java** η οποία αναπαριστά την δομή ενός γραφήματος με :

- ακέραιο αριθμό ακμών
- ακέραιο αριθμό κόμβων
- πίνακα τύπου bag για την αναπαράσταση των ζευγαριών κορυφών που ενώνονται.

Και η **GraphGenerator.java** η οποία χρησιμοποιήθηκε για την δημιουργία τόσο του τυχαίου γραφήματος όσο και του γραφήματος χρήστη.

Η κλάση στην οποία έγινε όλη η ανάπτυξη του δικού μας κώδικα είναι η **HillCimbingSearch.java** η οποία περιγράφεται αναλυτικά παρακάτω.

4.1.3 Δομές δεδομένων και μεταβλητές προβλήματος

Οι βοηθητικές δομές που χρησιμοποιήθηκαν είναι οι εξής:

- **Graph mainGraph:** Αναπαριστά την δομή του G1 που δημιουργείται τυχαία από τα δεδομένα που εισάγει ο χρήστης.
- **Graph subGraph:** Αναπαριστά την δομή του υπογραφήματος που δημιουργείται από τον χρήστη.
- **Map cspNodes:** Αναπαριστά την αντιστοίχηση του κάθε κόμβου G2 σε ένα κόμβο του G1
- **Map cspNodeConflicts**: Αναπαριστά την αντιστοίχηση κάθε κόμβου του G2 με παραβιάσεις με τον αριθμό των παραβιάσεων που αυτός έχει.
- **int numberOfNodes**: Μεταβλητή που αποθηκεύεται ο αριθμός των κόμβων του G2.
- **int maxSteps**: Μεταβλητή που αποθηκεύεται ο αριθμός των βημάτων που θα τρέξει ο αλγόριθμος. Πληροφορία που εισάγεται από τον χρήστη.
- **Random random :** Βοηθητική δομή για την τυχαία δημιουργία ζευγαριών ακμών.

4.1.4 Μέθοδοι προβλήματος

- **void scramble() :** Μέθοδος αρχικοποίησης του προβλήματος , όπου αντιστοιχίζονται οι κόμβοι του G2 σε τυχαία επιλεγμένους κόμβους του G1. Η αντιστοίχηση αποθηκεύεται στη δομή cspNodes.

• **int conflicts(int mainNode, int subNode):** Μέθοδος που δέχεται σαν παραμέτρους δύο κόμβους -έναν του G1 και έναν του G2 και επιστρέφει τον αριθμό των παραβιάσεων που έχουν.

• **void solve() :** Η κύρια μέθοδος όπου τρέχει ο αλγόριθμος.

1. Αρχικοποιεί τον μετρητή βημάτων ίσο με 1.
2. Μπαίνει σε λούπα ελέγχοντας αν έχουν ξεπεραστεί τα δυνατά βήματα.

2.1 Για κάθε κόμβο του G2 καλεί την μέθοδο conflicts και αποθηκεύει στην cspNodeConflicts τις τυχόν παραβιάσεις που βρίσκει.

2.2 Σε περίπτωση που δεν βρεθήκαν παραβιάσεις επιστρέφει μήνυμα επιτυχής λύσης με τον αριθμό των βημάτων που τρέξανε.

2.3 Αν υπάρχουν παραβιάσεις έστω και σε έναν κόμβο επιλέγει τυχαία έναν από αυτούς και αναζητά σε όλους τους κόμβους του κυρίως γραφήματος αν υπάρχει κάποιος με λιγότερες παραβιάσεις. Σε περίπτωση που υπάρχει κάποιος τον αντικαθιστά με αυτόν και ελέγχει αν έχει βρεθεί λύση στον αλγόριθμο. Αν υπάρχουν 2 ή περισσότεροι με λιγότερες παραβιάσεις επιλέγει τυχαία έναν από αυτούς. Αν δεν βρει κανένα με λιγότερες παραβιάσεις συνεχίζει στο επόμενο βήμα χωρίς καμία ενέργεια.

2.4 Αυξάνει τον αριθμό του μετρητή βημάτων.

3. Αν έχει βγει από την λούπα σημαίνει πως δεν έχει βρεθεί κάποια λύση μετά από το πέρας των διαθέσιμων βημάτων και βγάζει το κατάλληλο μήνυμα.

• **ArrayList<Integer> removeUnavailableNodes(ArrayList<Integer> cands):** Βοηθητική μέθοδος που αφαιρεί τους κόμβους του G1 που έχουν ήδη αντιστοίχηση με κάποιον κόμβο G2 από την δομή cands. Χρησιμοποιείται κατά την αναζήτηση γειτονικού κόμβου με λιγότερες παραβιάσεις.

4.2 Λειτουργία Προγράμματος

Στην main μέθοδο του προγράμματος δημιουργούνται δύο γραφήματα με την συμμετοχή του χρήστη, στην συνέχεια δημιουργείται ένα instance της κλάσης **HillClimbingSearch** με όνομα problem και καλείται η **scrable** για την αντιστοίχηση των κόμβων και στην συνέχεια η **solve** για την επίλυση του προβλήματος.

4.2.1 Περιγραφή λειτουργίας αλγορίθμου Ελαχίστων Συγκρούσεων

Αρχικά γίνεται η τυχαία αντιστοίχηση του κόμβου $n_i \in G_1$ με έναν κόμβο $n_j \in G_2$. Ο υπολογισμός των συγκρούσεων γίνεται ως εξής : Για κάθε γείτονα του κόμβου του G2 που επιλέχθηκε (n_{jz}) πρέπει να υπάρχει ο αντίστοιχος n_{iz} και στον G1 . Αν $n_{jz} \notin G_2$ or $n_{iz} \notin G_1$ τότε έχουμε σύγκρουση.

4.2.2 Παράδειγμα αλγορίθμου

Έστω ένα πρόβλημα περιορισμών με 2 γραφήματα:
Το κυρίως γράφημα G1(3,2) με 3 κορυφές έστω x1,x2,x3 και 2 ακμές που ενώνουν την x1 με την x2 και την x1 με την x3.

Αναλυτικά :

Ο x1 συνδέεται με {x2,x3}
Ο x2 συνδέεται με {x1}
Ο x3 συνδέεται με {x1}

Το υπογράφημα G2(2,1) με 2 κορυφές έστω y1,y2 και 1 ακμή που ενώνει την y1 με την y2 .
Ο y1 συνδέεται με {y2}
Ο y2 συνδέεται με {y1}

Βήμα 1ο : Τυχαία αντιστοίχηση των κόμβων του G2 σε κόμβους του G1:
έστω y1=x2 y2=x3

Βήμα 2ο : Υπολογισμός συγκρούσεων κάθε κόμβου :
Για y1=x2: 1 σύγκρουση: Ο x2 δεν έχει γείτονα τον x3.
Για y2=x3: 1 σύγκρουση: Ο x3 δεν έχει γείτονα τον χ2.

Βήμα3ο:Έλεγχος αν είναι λύση. Αν δεν είναι συνέχεια στο 4ο Βήμα.

Βήμα 4ο : Τυχαία επιλογή ενός κόμβου που συμμετέχει σε σύγκρουση περιορισμών:
Έστω ότι επιλέγει την μεταβλητή y1.

Βήμα 5ο : Υπολογισμός συγκρούσεων του επιλεγμένου κόμβου για κάθε κόμβο του κυρίως γραφήματος.

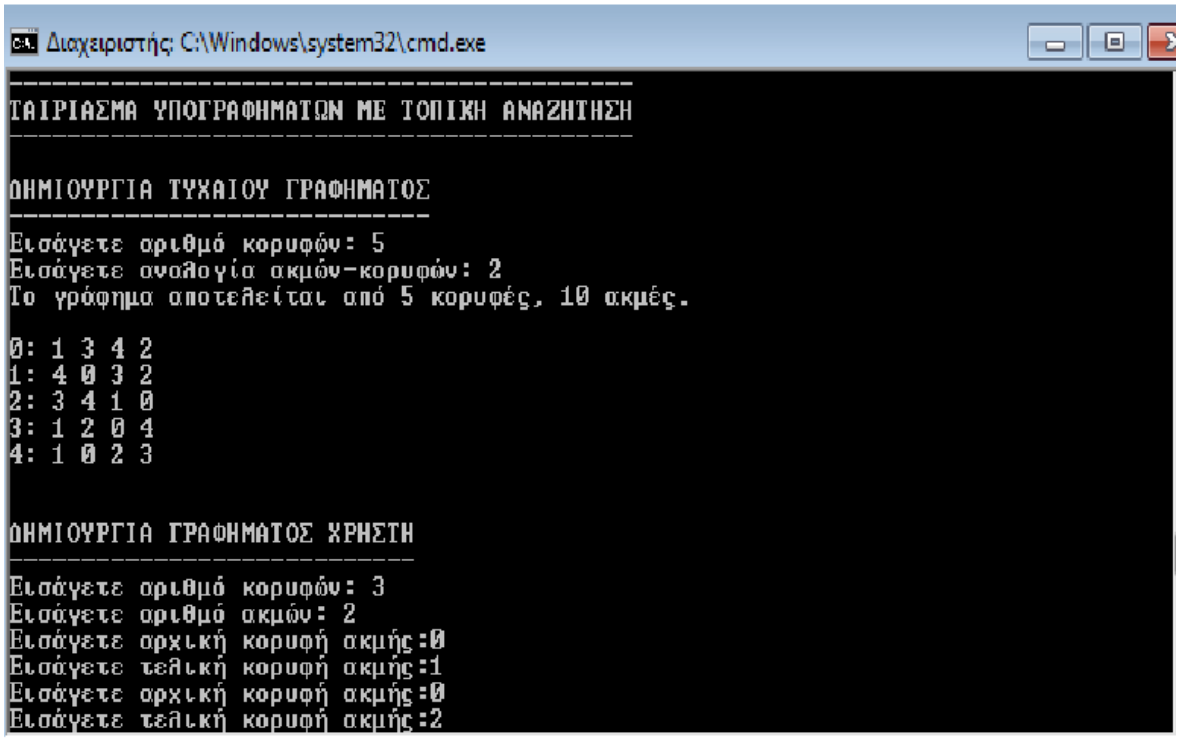
Για y1=x1 Καμία σύγκρουση. Ο x1 έχει γείτονα τον x3.
Επιλογή του κόμβου. Συνέχεια στο 2ο Βήμα.

4.3 Αποτελέσματα εφαρμογής

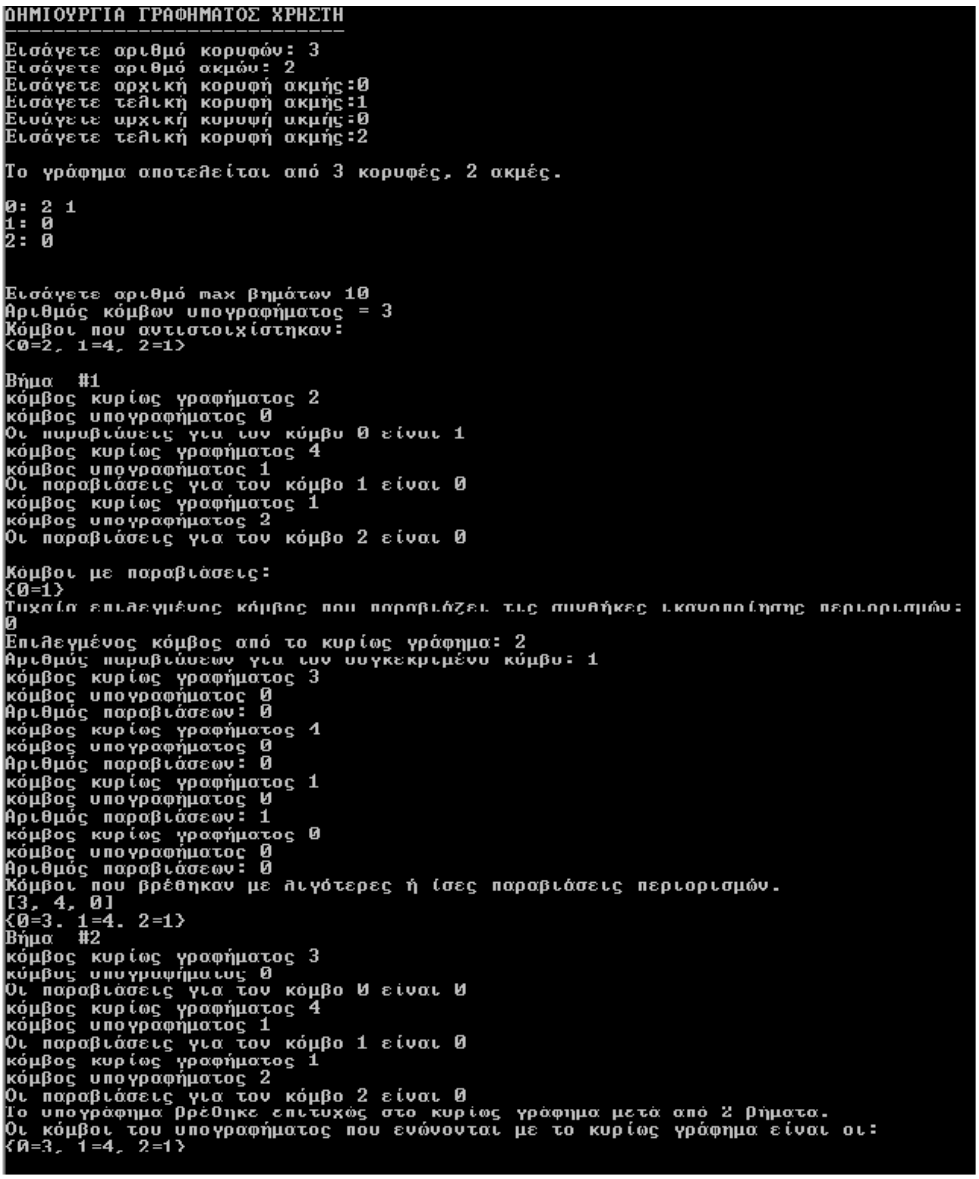
Μετά την υλοποίηση του αλγόριθμου ακολουθεί η καταγραφή των αποτελεσμάτων.
Ο αριθμός κορυφών-ακμών αυξάνεται σε κάθε περίπτωση.

Περίπτωση i)

Αριθμός κορυφών τυχαίου γραφήματος : 5
Αριθμός ακμών τυχαίου γραφήματος : 10
Αριθμός κορυφών γραφήματος χρήστη : 3
Αριθμός ακμών γραφήματος χρήστη : 2



Αριθμός βημάτων : 10



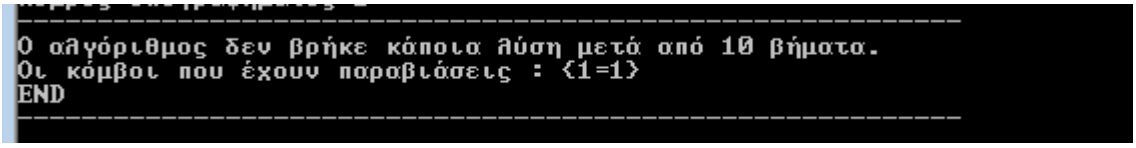
Το υπογράφημα βρέθηκε επιτυχώς στο κυρίως γράφημα μετά από 2 βήματα και η αντιστοίχηση των κορυφών είναι <0=3, 1=4 , 2=1>.

Περίπτωση ii)

Αριθμός κορυφών τυχαίου γραφήματος : 20
Αριθμός ακμών τυχαίου γραφήματος : 40
Αριθμός κορυφών γραφήματος χρήστη : 5
Αριθμός ακμών γραφήματος χρήστη : 4

Αριθμός βημάτων : 10

Σε αυτή την περίπτωση ο αλγόριθμος δεν βρήκε κάποια λύση μετά από 10 βήματα. Ο κόμβος που παρουσίασε παραβιάσεις ήταν ο κόμβος 1.



Περίπτωση iii)

Αριθμός κορυφών τυχαίου γραφήματος : 15
Αριθμός ακμών τυχαίου γραφήματος : 30
Αριθμός κορυφών γραφήματος χρήστη : 4
Αριθμός ακμών γραφήματος χρήστη : 3

Αριθμός βημάτων : 10

Το υπογράφημα βρέθηκε επιτυχώς στο κυρίως γράφημα μετά από 1 βήμα και η αντιστοίχηση των κορυφών είναι <0=2, 1=3 , 2=0 , 3=4 >.

Περίπτωση iv)

Αριθμός κορυφών τυχαίου γραφήματος : 20
Αριθμός ακμών τυχαίου γραφήματος : 100
Αριθμός κορυφών γραφήματος χρήστη : 6
Αριθμός ακμών γραφήματος χρήστη : 5

Αριθμός βημάτων : 20

Το υπογράφημα βρέθηκε επιτυχώς στο κυρίως γράφημα μετά από 4 βήματα και η αντιστοίχηση των κορυφών είναι <0=12, 1=4, 2=2 , 3=10 , 4=14 ,5=15 >.

Περίπτωση v)

Αριθμός κορυφών τυχαίου γραφήματος : 30
Αριθμός ακμών τυχαίου γραφήματος : 150
Αριθμός κορυφών γραφήματος χρήστη : 8
Αριθμός ακμών γραφήματος χρήστη : 9

Αριθμός βημάτων : 20

Σε αυτή την περίπτωση ο αλγόριθμος δεν βρήκε κάποια λύση μετά από 20 βήματα. Οι κόμβοι που παρουσίασαν παραβιάσεις ήταν οι κόμβοι <7=3>

Περίπτωση vi)

Αριθμός κορυφών τυχαίου γραφήματος : 50
Αριθμός ακμών τυχαίου γραφήματος : 50
Αριθμός κορυφών γραφήματος χρήστη : 10
Αριθμός ακμών γραφήματος χρήστη : 10

Αριθμός βημάτων : 20

Σε αυτή την περίπτωση ο αλγόριθμος δεν βρήκε κάποια λύση μετά από 20 βήματα. Οι κόμβοι που παρουσίασαν παραβιάσεις ήταν οι κόμβοι <0=3, 1=6 , 2=1, 3=2, 4=1 , 5=1>

Περίπτωση vii)

Αριθμός κορυφών τυχαίου γραφήματος : 50
Αριθμός ακμών τυχαίου γραφήματος : 100
Αριθμός κορυφών γραφήματος χρήστη : 4
Αριθμός ακμών γραφήματος χρήστη : 3

Αριθμός βημάτων : 20

Σε αυτή την περίπτωση ο αλγόριθμος δεν βρήκε κάποια λύση μετά από 20 βήματα. Οι κόμβοι που παρουσίασαν παραβιάσεις ήταν οι κόμβοι <1=1 , 2=2>

Περίπτωση viii)

Αριθμός κορυφών τυχαίου γραφήματος : 200
Αριθμός ακμών τυχαίου γραφήματος : 200
Αριθμός κορυφών γραφήματος χρήστη : 2
Αριθμός ακμών γραφήματος χρήστη : 1

Αριθμός βημάτων 5

Το υπογράφημα βρέθηκε επιτυχώς στο κυρίως γράφημα μετά από 2 βήματα και η αντιστοίχηση των κορυφών είναι <0=56, 1=127 >.

Περίπτωση ix)

Αριθμός κορυφών τυχαίου γραφήματος : 100

Αριθμός ακμών τυχαίου γραφήματος : 200
Αριθμός κορυφών γραφήματος χρήστη : 10
Αριθμός ακμών γραφήματος χρήστη : 9

Αριθμός βημάτων 50

Σε αυτή την περίπτωση ο αλγόριθμος δεν βρήκε κάποια λύση μετά από 50 βήματα. Οι κόμβοι που παρουσίασαν παραβιάσεις ήταν οι κόμβοι <0=3, 1=6 , 2=1, 3=2, 4=1 , 5=1>

Περίπτωση x)

Αριθμός κορυφών τυχαίου γραφήματος : 100
Αριθμός ακμών τυχαίου γραφήματος : 200
Αριθμός κορυφών γραφήματος χρήστη : 100
Αριθμός ακμών γραφήματος χρήστη : 5

Αριθμός βημάτων 50

Σε αυτή την περίπτωση ο αλγόριθμος δεν βρήκε κάποια λύση μετά από 50 βήματα. Οι κόμβοι που παρουσίασαν παραβιάσεις ήταν οι κόμβοι <0=1, 64=1 , 80=1, 1=1, 33=1 , 50=1 ,99=1 , 77=1>

Περίπτωση xi)

Αριθμός κορυφών τυχαίου γραφήματος : 300
Αριθμός ακμών τυχαίου γραφήματος : 300
Αριθμός κορυφών γραφήματος χρήστη : 30
Αριθμός ακμών γραφήματος χρήστη : 20

Αριθμός βημάτων 50

Σε αυτή την περίπτωση ο αλγόριθμος δεν βρήκε κάποια λύση μετά από 50 βήματα. Οι κόμβοι που παρουσίασαν παραβιάσεις ήταν οι κόμβοι <47=1, 66=4 , 83=1, 55=2, 17=3 , 50=2 ,71=1 , 77=2>

Περίπτωση xii)

Αριθμός κορυφών τυχαίου γραφήματος : 500
Αριθμός ακμών τυχαίου γραφήματος : 500
Αριθμός κορυφών γραφήματος χρήστη : 20
Αριθμός ακμών γραφήματος χρήστη : 30

Αριθμός βημάτων : 100

Σε αυτή την περίπτωση ο αλγόριθμος δεν βρήκε κάποια λύση μετά από

100 βήματα. Οι κόμβοι που παρουσίασαν παραβιάσεις ήταν οι κόμβοι
<0=1, 1=3 , 2=3, 4=1, 5=2 , 7=5 , 8=1 , 9=1 , 10=3 , 11=4 , 10=2 , 13=1 >

Περίπτωση xiii)

Αριθμός κορυφών τυχαίου γραφήματος : 500
Αριθμός ακμών τυχαίου γραφήματος : 2500
Αριθμός κορυφών γραφήματος χρήστη : 10
Αριθμός ακμών γραφήματος χρήστη : 9

Αριθμός βημάτων : 30

Σε αυτή την περίπτωση ο αλγόριθμος δεν βρήκε κάποια λύση μετά από 30 βήματα. Οι κόμβοι που παρουσίασαν παραβιάσεις ήταν οι κόμβοι
<0=1, 1=1, 3=2, 5=1, 6=1 , 8=1 , 9=1 >

5 Βιβλιογραφία

[1] Διομήδης Σπινέλλης, Αλγόριθμοι και δομές δεδομένων, 1997

[2] Παναγιώτης Σπύρου, Θεωρία Γραφημάτων, 1997

[3] Μανώλης Κουμπαράκης, Σημειώσεις Τεχνητής Νοημοσύνης, Τμήμα Πληροφορικής και Τηλεπικοινωνιων, ΕΚΠΑ, 2007

[4] Endika Bengoetxea, PhD Thesis, 2002

[5] Γεώργιος Καστρίνης, “Αλγόριθμοι τοπικής αναζήτησης στον προγραμματισμό με περιορισμούς”, Πτυχιακή εργασία, Τμήμα Πληροφορικής και Τηλεπικοινωνιών, ΕΚΠΑ, 2011

[6] Καλογιαννίδου Ευμορφία, "Τοπική Ακτινωτή Αναζήτηση για Προβλήματα Ικανοποίησης Περιορισμών", Διπλωματική εργασία, Τμήμα Μηχανικών Πληροφορικής & Τηλεπικοινωνιών, ΠΔΜ

[7] “Algorithms 4th Edition”, Robert Sedgewick and Kevin Wayne, 2011