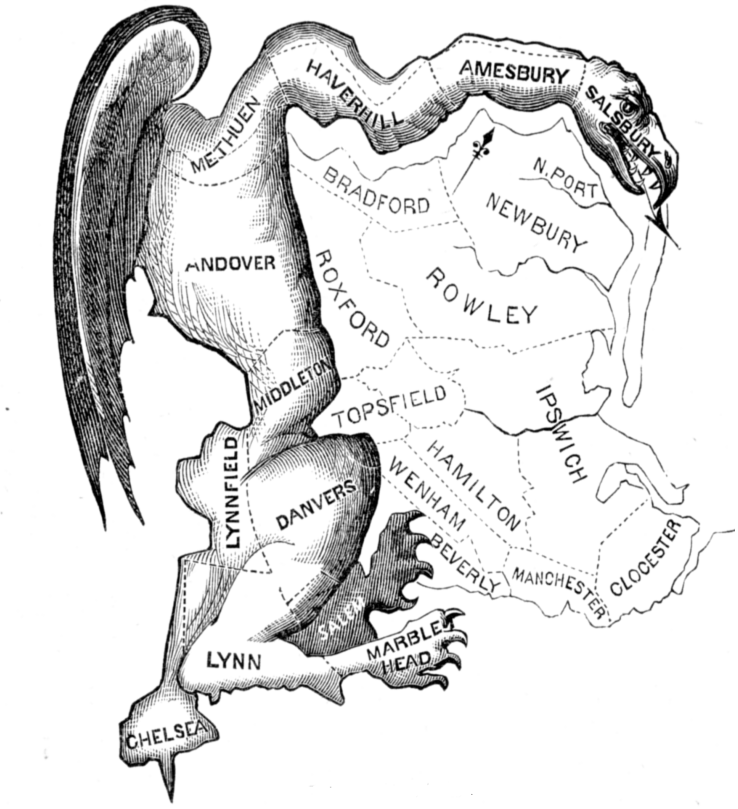
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ

ΘΕΩΡΗΤΙΚΗ ΜΕΛΕΤΗ ΑΛΓΟΡΙΘΜΩΝ ΠΕΡΙΦΕΡΕΙΟΠΟΙΗΣΗΣ ΚΑΙ ΕΦΑΡΜΟΓΗ ΤΟΥΣ ΓΙΑ ΠΑΡΑΓΩΓΗ ΕΚΛΟΓΙΚΩΝ ΠΕΡΙΦΕΡΕΙΩΝ ΣΤΟΥΣ ΔΗΜΟΥΣ ΤΟΥ ΚΑΠΟΔΙΣΤΡΙΑ



ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΠΑΝΑΓΙΩΤΙΔΗΣ ΠΕΡΙΚΛΗΣ

ΕΠΙΒΛΕΠΩΝ: Κ. ΚΟΥΤΣΟΠΟΥΛΟΣ

ΣΕΠΤΕΜΒΡΙΟΣ 2008

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον Καθηγητή κ. Κωστή Κουτσόπουλο, επιβλέποντα αυτής της εργασίας, για την καθοδήγηση και την πολύτιμη βοήθεια που μου παρείχε τόσο στην εργασία αυτή όσο και σε όλη τη διάρκεια της φοίτησής μου στο πολυτεχνείο.

Περιεχόμενα

Κεφάλαιο 1ο
Εισαγωγή

Η γέννεση των πρώτων γεωγραφικών συστημάτων στη δεκαετία του 70 δημιούργησε μεγάλες προσδοκίες στην ανάπτυξη της χωρικής ανάλυσης. Παρόλαυτα, όλα τα προβλήματα δεν λύθηκαν και ο τομέας αυτός παραμένει ακόμα και σήμερα ένας από τους δραστήριους ερευνητικά στον τομέα της γεωγραφίας. Η χρήση των ψηφιακών μέσων από την άλλη όπλισε τους ερευνητές με νέες δυνατότητες στην αντιμετώπιση πολύπλοκων προβλημάτων.

Η ομαδοποίηση μικρών μονάδων σε μεγαλύτερα σύνολα είναι ένα αρκετά συχνό πρόβλημα προς επίλυση σε τομείς όπως η χαρτογραφία, πολεοδομία, κοινωνική γεωγραφία κ.α. Σε αντίθεση όμως με τη ζήτηση, η προσφορά σε αυτοματοποιημένο λογισμικό, είτε ελεύθερο είτε στο εμπόριο, είναι σχεδόν μηδενική. Αν εξαιρέσουμε την εφαρμογή του Openshaw το 1998 *(ZDES, 1998)* δεν υπάρχει κάτι άλλο αξιόλογο στην διάθεσή μας.

Ένα μικρό κομμάτι από αυτό το κενό επιχειρεί να καλύψει αυτή η εργασία. Με αφορμή προηγούμενες μελέτες πάνω στο αντικείμενο της περιφερειοποίησης γίνεται προσπάθεια να υλοποιηθούν κάποιες από αυτές, καθώς και να προταθούν κάποιες νέες πιθανώς βελτιωμένες μέθοδοι. Το πρόβλημα που αντιμετωπίζεται μπορεί να διατυπωθεί ως εξής: Ποιος είναι ο βέλτιστος τρόπος να χωριστεί ένας αριθμός ζωνών σε έναν μικρότερο αριθμό περιφερειών έτσι ώστε οι τελευταίες να είναι ισοπληθείς. Συγκεκριμένα γίνονται τρεις προσεγγίσεις στο πρόβλημα. Η πρώτη χρησιμοποιεί μια παραλλαγή του αλγορίθμου SARA *(University of Oxford)*, η δεύτερη προσέγγιση είναι με βάση τα κεντροϊδή και η τρίτη είναι ένας συνδυασμός των δύο προηγούμενων μεθόδων με στόχο την καλύτερη προσαρμογή του λογισμικού στον ιδιόμορφο ελλαδικό χώρο. Ως χωρική μονάδα χρησιμοποιούνται οι Καποδιστριακοί δήμοι και με χρήση διαφόρων αλγορίθμων ομαδοποιούνται σε μεγαλύτερες περιφέρειες.

Η ανάπτυξη του λογισμικού έγινε σε γλώσσα προγραμματισμού PHP και για την επεξεργασία των γεωγραφικών εικόνων έγινε σε περιβάλλον ArcGIS. Άλλα βοηθητικά προγράμματα που χρησιμοποιήθηκαν είναι το OpenOffice, Apache server και το FME Suit.

Εισαγωγή στα Γεωγραφικά Συστήματα Πληροφοριών

Παρά το μεγάλο ενδιαφέρον και την τρομερή εξέλιξη που παρατηρήθηκε στη χρήση και εφαρμογή των Γεωγραφικών Συστημάτων Πληροφοριών (ΓΣΠ) στα τελευταία 30 χρόνια, εντούτοις

οι προσπάθειες για ένα σαφή και κοινά αποδεκτό ορισμό για το τι είναι ΓΣΠ και κυρίως ποιες είναι οι εφαρμογές τους δεν έχουν ακόμα ευοδωθεί. Για μια πληθώρα από λόγους, ο σαφής καθορισμός των ΓΣΠ και των εφαρμογών τους είναι περισσότερο δύσκολος απ' ό,τι οι πωλητές συστημάτων υποστηρίζουν και οι ειδικοί θα ήθελαν. Φαινομενικά έχει επικρατήσει η άποψη ότι στην περίπτωση των ΓΣΠ υπάρχει τοπικό και όχι παγκόσμιο βέλτιστο ή με άλλα λόγια η εφαρμογή ορίζει το εργαλείο.

Μια τέτοια άποψη όμως, σαφώς παραβιάζει την επιστημονική δεοντολογία και μπορεί να οδηγήσει σε επικίνδυνες ατραπούς. Αντίθετα, πιστεύεται ότι όλες αυτές οι ιδέες αποτελούν μικρά τμήματα ενός συνολικού "παζλ" που αφορά το γεωγραφικό χώρο. Πραγματικά, οι διαφορετικές ιδέες που έχουν κατά καιρούς εκφραστεί για το ΣΓΠ και τις εφαρμογές τους, μπορούν να συμπτυχθούν σε τρεις ξεχωριστές ομάδες, που είναι αλληλένδετες μεταξύ τους.

Η πρώτη ομάδα μπορεί να χαρακτηριστεί ως "Διαχειριστική προσέγγιση" και αφορά την "Χαρτογραφική προσέγγιση" η οποία εστιάζεται κυρίως στα χαρτογραφικά χαρακτηριστικά των ΓΣΠ. Η δεύτερη ομάδα αφορά την "Πληροφορική Προσέγγιση" που δίνει έμφαση στην σπουδαιότητα των ΓΣΠ ως σύγχρονων συστημάτων διαχείρισης βάσεων δεδομένων. Η ομάδα αυτή αναφέρεται σαν "Προσέγγιση Χωρικής Ανάλυσης" και βεβαίως υποστηρίζει τη σπουδαιότητα της Γεωγραφικής (Χωρικής) Ανάλυσης. Η τρίτη ομάδα αναφέρεται στη "Σχεδιαστική Προσέγγιση" και εστιάζεται στη δυνατότητα των ΓΣΠ να βοηθούν στην επίλυση χωρικών προβλημάτων, δηλαδή να συμμετέχουν ενεργά στο χωρικό σχεδιασμό.

Περιγραφή Κεφαλαίων της Εργασίας

Στο 1^{ο} κεφάλαιο γίνεται μια εισαγωγή στους αλγορίθμους περιφερειοποίησης καθώς και στον τρόπο που εφαρμόστηκαν στην παρούσα εργασία. Στη συνέχεια δίνονται κάποιοι ορισμοί για τα Γεωγραφικά Συστήματα Πληροφοριών.

Στο 2^{ο} κεφάλαιο περιγράφεται αναλυτικά ο αλγόριθμος σημειακής προσέγγισης

Στο 3^{ο} κεφάλαιο αναλύεται ο αλγόριθμος με την χρήση πολυγώνων του Openshaw. Στη συνέχεια περιγράφονται και ορισμένες βελτιωμένες εκδοχές του ίδιου αλγορίθμου.

Στο 4^{ο} κεφάλαιο περιγράφεται το πρόβλημα της συνέχειας τον περιφερειών. Κατά την περιφερειοποίηση τίθεται ένας περιορισμός σύμφωνα με τον οποίο κάθε παραγόμενη περιφέρεια οφείλει να είναι συνεχής. Το πρόβλημα αυτό είναι αρκετά πολύπλοκο και αναλύεται διεξοδικά.

Στο 5^{ο} κεφάλαιο παρουσιάζεται η εφαρμογή των αλγορίθμων που αναλύθηκαν στα προηγούμενα κεφάλαια στους δήμους του Καποδίστρια. Εδώ γίνεται τεχνική έκθεση της εφαρμογής και αναφέρονται εντυπώσεις από αυτή.

Στο 6^{ο} κεφάλαιο περιγράφονται οι πιθανοί κίνδυνοι από μία κακή περιφερειοποίηση και

γίνεται εισαγωγή στην έννοια της “πυκνότητας”.

Στο 7^{ο} κεφάλαιο παρουσιάζονται συγκεντρωμένα τα συμπεράσματα όλης της μελέτης, μια σύντομη περιγραφή της καθώς και ορισμένες προτάσεις για περαιτέρω βελτίωση της εφαρμογής στο μέλλον.

Τέλος τα παραρτήματα περιέχουν τον κώδικα της εφαρμογής και χάρτες με τα αποτελέσματα.

Κεφάλαιο 2ο
Αλγόριθμος σημειακής προσέγγισης

2.1 Εισαγωγή

Ο αλγόριθμος αυτός ανήκει στην κατηγορία των Genetic Algorithms(GA) οι οποίοι είναι παρακλάδι του Evolutionary Computing (εξελικτική υπολογιστική). Όπως φαίνεται και από το όνομα, οι αλγόριθμοι αυτοί είναι εμπνευσμένοι από τη θεωρία του Δαρβίνου περί εξέλιξης. Στην πράξη εννοούμε ότι οι λύσεις εξελίσσονται προκειμένου να προσαρμοστούν. Έτσι μπορούμε να δούμε την εξέλιξη σαν μία διαδικασία βελτιστοποίησης στην οποία οι καλύτερες λύσεις είναι αυτές που λύνουν με τον βέλτιστο τρόπο το πρόβλημα που προκύπτει από το εκάστοτε περιβάλλον. Οι GA μπορούν να ενταχθούν στις στοχαστικές ερευνητικές μεθόδους, αλλά σε αντίθεση με τις περισσότερες από αυτές, αντί να δουλεύουν με μία λύση επεξεργάζονται ένα σύνολο λύσεων.

Προκειμένου να υλοποιηθεί ο αλγόριθμος πρέπει πρώτα να καθοριστεί το σύνολο των παραμέτρων που περιγράφουν την κάθε πιθανή λύση το οποίο αποτελεί και την πρώτη αρχή του αλγορίθμου. Η δεύτερη αρχή του GA είναι ο καθορισμός της εξίσωσης βελτιστοποίησης η οποία χαρακτηρίζει την ποιότητα της λύσης. Κατ' αυτόν τον τρόπο η κάθε πιθανή λύση θα μπορεί να εξεταστεί ως προς την "ικανότητά" της να επιλύει το πρόβλημα. Για να βελτιώσουμε τον χρόνο επεξεργασίας χωρίζουμε το αρχικό σύνολο σε υποομάδες της τάξεως των 200 περίπου δήμων.

Ο αλγόριθμος που εφαρμόστηκε εδώ δεν είναι βασισμένος σε πολύγωνα, επομένως δεν χρησιμοποιεί δεδομένα επιφάνειας, εντούτοις, τα στοιχειώδη αντικείμενα είναι σημεία και στις περισσότερες περιπτώσεις τα κεντροειδή των πολυγώνων. Η όλη διαδικασία αφορά την εύρεση των κέντρων των νέων περιφερειών που θα δημιουργηθούν. Η ορολογία που θα χρησιμοποιείται για ευκολότερη κατανόηση είναι η εξής:

- Σημεία **m** είναι τα κέντρα των προς αναζήτηση περιφερειών
- Σημεία **n** είναι τα κέντρα των ζωνών που θα χρησιμοποιηθούν για την κατασκευή των περιφερειών
- **Πολύγωνα** είναι η αρχική ελάχιστη μονάδα έκτασης που μελετάται (δήμος)
- **Περιφέρεια** είναι η νέα έκταση που δημιουργείται με την συνένωση δύο η περισσοτέρων πολυγώνων

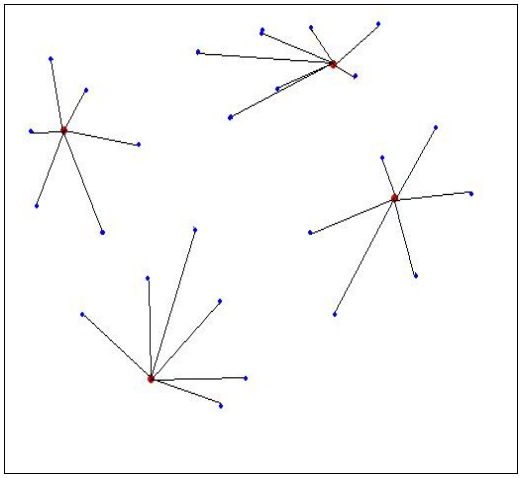
Βασιζόμενοι στο γεωγραφικό σύστημα συντεταγμένων, κάθε πολύγωνο μπορούμε να θεωρήσουμε κάθε πολύγωνο ως ένα σημείο (κεντροειδές). Αν τοποθετηθεί στο σύστημα συντεταγμένων τυχαία ένας αριθμός σημείων n, τότε το κάθε σημείο m θα βρίσκεται κοντύτερα σε ένα και μοναδικό n. Ο

αριθμός των m σημείων προφανώς εξαρτάται από τον αριθμό των περιφερειών προς κατασκευή. Όταν το κάθε m έχει συνδεθεί με ένα n τότε μία λύση έχει επιτευχθεί. Κατά αυτόν τον τρόπο με το να κατασκευάζουμε τον κατάλληλο αριθμό m σημείων καταχωρούμε και μία λύση (εικόνα 1). Προκειμένου να ξεκινήσει ο αλγόριθμος, το πρώτο βήμα είναι να παράγουμε έναν k αριθμό από n σύνολα.

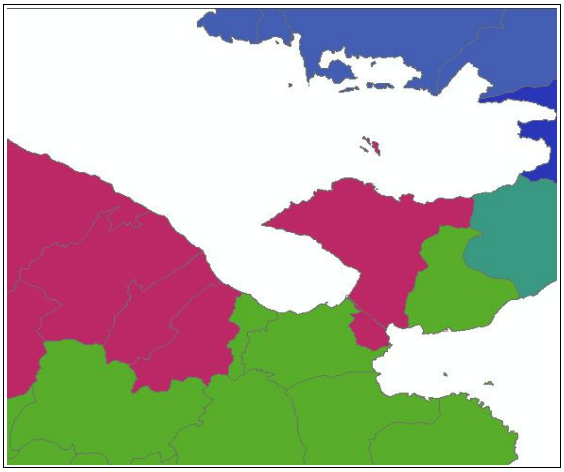
Ο αλγόριθμος εξελίσσεται ως εξής:

1. Παράγονται k σετ n σημείων. Αυτά μπορεί να βρίσκονται οπουδήποτε στο σύστημα συντεταγμένων.
2. Αφού προσδιοριστούν οι θέσεις των n, το κάθε m συνδέεται με το κοντινότερο n.
3. Υπολογίζεται η τιμή της συνάρτησης βελτιστοποίησης.
4. Όλα τα αποτελέσματα καταχωρούνται.
5. Επιστροφή στην αρχή

Παρόλα αυτά σε μερικές από τις λύσεις η συνέχεια δεν διατηρείται. Αυτό το πρόβλημα προκύπτει από το γεγονός ότι ο αλγόριθμος επιτρέπει τα n σημεία να βρίσκονται οπουδήποτε στο σύστημα συντεταγμένων. Κάτι τέτοιο φυσικά δεν είναι αποδεκτό καθώς μία από της αρχές της περιφερειοποίησης είναι η συνέχεια των περιοχών (εικόνα 2).



Εικόνα 1: Παράδειγμα σύνδεσης των κέντρων τον ζωνών με τα κέντρα τον περιφερειών



Εικόνα 2: Παράδειγμα απώλειας συνέχειας

Προκειμένου να ξεπεραστεί αυτό το πρόβλημα υπάρχουν διάφορες προτεινόμενες λύσεις στη σχετική βιβλιογραφία *(Macmillan, Openshaw)*. Ένας τρόπος είναι να τροποποιήσουμε την συνάρτηση βελτιστοποίησης θέτοντας ως περιορισμό την συνέχεια των περιφερειών. Κάτι τέτοιο

μπορεί να γίνει με τη χρήση τοπολογικού πίνακα. Αυτό βέβαια εκτός από το ότι θα αυξήσει σε γεωμετρικό βαθμό την πολυπλοκότητα και συνεπώς το συνολικό χρόνο επεξεργασίας, θα μειώσει επίσης κατά πολύ τον αριθμό των αποδεκτών λύσεων και θα ελαττώσει πιθανώς την ποιότητά τους.

Δεύτερη προσέγγιση του προβλήματος αυτού είναι μέσω μίας σειράς γεωμετρικών δεδομένων τα οποία βάζουν κάποιους περιορισμούς στο σχήμα της περιφέρειας. Ποσοτικοποιώντας το πόσο συμπαγές (compact) είναι το σχήμα της περιφέρειας μειώνουμε την πιθανότητα να εμφανιστούν ασυνέχειες. Η πιθανότητα όμως δεν μπορεί να εξαλειφθεί. Εδώ μπορούμε να σημειώσουμε ότι με την παραπάνω μέθοδο έχουμε και το θετικό της βελτίωσης του τελικού σχήματος της περιφέρειας μειώνοντας της μέγιστες αποστάσεις.

Τέλος υπάρχει και η δυνατότητα ανθρώπινης επέμβασης με στόχο την διόρθωση των ελαττωμάτων. Αυτό είναι εφικτό στην περίπτωση που ο αριθμός των δεδομένων είναι σχετικά μικρός.

2.2 Εφαρμογή του αλγόριθμου σημειακής προσέγγισης

Όπως αναφέρεται στο θεωρητικό κομμάτι του αλγορίθμου και παρατηρήθηκε και στην πράξη, για να λειτουργήσει αποτελεσματικότερα αυτή η μέθοδος απαιτείται χωρισμός των πολυγώνων σε υποομάδες. Για την περίπτωση της Ελλάδας έγιναν τρεις προσεγγίσεις. Η πρώτη είναι χωρίς καθόλου διαχωρισμό, η δεύτερη με διαχωρισμό σε έξι ζώνες με χρήση ενός μικρού καννάβου και η τρίτη με διαχωρισμό σε οχτώ ζώνες όπως και στη δεύτερη περίπτωση αλλά με δύο επιπλέον ζώνες για τις περιοχές με μεγάλη συγκέντρωση πληθυσμού (Αθήνα, Θεσσαλονίκη). Αυτό που παρατηρήθηκε είναι ότι όσο μικραίνει η περιοχή εφαρμογής τόσο βελτιώνεται το αποτέλεσμα. Αυτό είναι εμφανές στους χάρτες που παρουσιάζονται τα αποτελέσματα. Στις πρώτες δύο περιπτώσεις που τα δύο μεγάλα αστικά κέντρα δεν διαχωρίζονται έχουμε περιφέρειες με πολύ μεγάλους πληθυσμούς καθώς περιέχουν μεγάλα κομμάτια αστικών περιοχών. Στην τρίτη όμως περίπτωση η Αθήνα και η Θεσσαλονίκη έχουν περιφερειοποιηθεί με σωστότερο τρόπο, παράγοντας μία αποδεκτή λύση. Το πλάνο της εφαρμογής που αναπτύχθηκε είναι το εξής:

1. Χωρισμός των δήμων σε ομάδες και υπολογισμός του αριθμού **n** των περιφερειών της κάθε μίας
   - α)Για κάθε ομάδα παράγονται **n** ζεύγη συντεταγμένων
   - β)Τα **n** κέντρα των ζωνών ενώνονται με τα κοντινότερα **m** κέντρα των δήμων
   - γ)Για κάθε **n** υπολογίζεται η συνάρτηση βελτιστοποίησης (Τυπική απόκλιση)
   - δ)Αποθηκεύεται η καλύτερη λύση
2. Οι ομάδες επανενώνονται και οπτικοποιούνται σε χάρτη.

2.3 Πλεονεκτήματα – Μειονεκτήματα

Στα θετικά αυτής της μεθόδου μπορούμε να βάλουμε τα εξής:

- Σχετικά απλή η υλοποίηση
- Καλής ποιότητας αποτελέσματα
- Άριστη ανταπόκριση του αλγορίθμου στις ιδιομορφίες της χώρας (π.χ. Νησιά)
- Πολύ καλή διαχείριση περιοχών με μεγάλη συγκέντρωση πληθυσμού

Αρνητικά της μεθόδου:

- Αρκετά μεγάλος χρόνος επεξεργασίας
- Κάποιες απώλειες συνέχειας
- Κάποιες περιφέρειες δεν έχουν τον αναμενόμενο πληθυσμό

Κεφάλαιο 3ο

Προσέγγιση με χρήση πολυγώνων

3.1 Εισαγωγή

Σε αντίθεση με τον αλγόριθμο σημειακής προσέγγισης, ο συγκεκριμένος λαμβάνει σαν δεδομένα επεξεργασίας πολύγωνα. Η βάση αυτού του αλγορίθμου είναι η μελέτη του Openshaw πάνω στη διαδικασία της "επαναπεριφερειοποίησης" (Redistricting). Ξεκινώντας με μία τυχαία ομαδοποίηση καταλήγουμε σε μία νέα η οποία καλύπτει τις απαιτήσεις μας βάσει κάποιων κριτηρίων. Η διαδικασία περιλαμβάνει την συνένωση χωρικών δεδομένων για ***N*** ζώνες σε ***M*** περιφέρειες με προφανώς ***M<N.*** Οι τρόποι που μπορεί να γίνει αυτός ο χωρισμός είναι πολύ μεγάλος, ποιος όμως είναι ο καλύτερος μπορούμε να το υπολογίσουμε. Κατασκευάζοντας μία αντικειμενική συνάρτηση, όπως και στην σημειακή προσέγγιση, μπορούμε να ποσοτικοποιήσουμε την ποιότητα του αποτελέσματος. Σε αντίθεση όμως με την σημειακή μέθοδο εδώ δεν παράγουμε λύσεις ώστε να τις συγκρίνουμε αλλά προσπαθούμε να προσεγγίσουμε την βέλτιστη σταδιακά. Η αντικειμενική συνάρτηση μπορεί να περιλαμβάνει κριτήρια οποιασδήποτε φύσης ανάλογα με το πρόβλημα που αντιμετωπίζεται. Μπορεί για παράδειγμα να είναι οικονομικής φύσης , δημογραφικής, γεωγραφικής κλπ. Έτσι λοιπόν το πρόβλημα ανάγεται στην βελτιστοποίηση της συνάρτησης: ***F(z)*** όπου ***z*** τα κριτήρια που μας ενδιαφέρουν με τον περιορισμό βέβαια η κάθε ζώνη ***N*** να ανήκει σε μία και μόνο περιφέρεια ***M.***

Η αρχική προσέγγιση του Openshaw(1977) ήταν ο αλγόριθμος AZP (automatic zoning procedure) ο οποίος συνοψίζεται στα παρακάτω βήματα:

1. Έναρξη με παραγωγή ενός τυχαίου συστήματος περιφερειών ***M*** με ***M<N***
2. Δημιουργία λίστας με τις περιφέρειες ***M***
3. Τυχαία επιλογή και αφαίρεση οποιασδήποτε περιφέρειας ***K*** από τη λίστα.
4. Εύρεση ενός συνόλου γειτονικών στην ***K*** ζωνών οι οποίες θα μπορούσαν να προσαρτηθούν σε αυτή χωρίς να καταστραφεί η εσωτερική ενότητα της περιφέρειας δότη.
5. Τυχαία επιλογή ζωνών από την παραπάνω λίστα έως ότου να υπάρξει βελτίωση στην αντικειμενική συνάρτηση ή όταν το αποτέλεσμα παραμένει σταθερό. Στις περιπτώσεις αυτές η μετακίνηση είναι αποδεκτή και καταχωρείται. Ενημέρωση των προς μετακίνηση ζωνών και επιστροφή στο βήμα 4 ή επιστροφή στο βήμα 5 μέχρι να εξαντληθεί η λίστα.
6. Όταν η λίστα για την περιφέρεια **K** εξαντληθεί επιστροφή στο βήμα 3 ώστε να επιλεχθεί νέα περιφέρεια και να επαναληφθούν τα βήματα 4 έως 6.

7. Επανάληψη των βημάτων 2 έως 6 μέχρι να μην υπάρχουν θετικές μετακινήσεις

Ο AZP μπορεί να λειτουργήσει με οποιαδήποτε αντικειμενική συνάρτηση η οποία είναι ευαίσθητη στη συνένωση δεδομένων για ***N*** ζώνες σε ***M*** περιφέρειες και υπολογίζεται στο βήμα 5. Τέτοιες συναρτήσεις μπορεί να προέρχονται κατευθείαν από τα δεδομένα όπως για παράδειγμα το άθροισμα των ελαχίστων τετραγώνων ή άλλες με ικανότητα να προσαρμόζονται σε ένα μοντέλο που θεωρείται αποδεκτό. Ο αλγόριθμος αυτός έχει εφαρμοστεί με επιτυχία στο παρελθόν (Openshaw 1977, Openshaw & Taylor 1979) αλλά σε μικρό αριθμό δεδομένων. Οι εφαρμογές ήταν κυρίως σε χωρικά δεδομένα σε μορφή καννάβου και της τάξεως των 100 σε αριθμό.

3.2 Προβλήματα του αλγορίθμου AZP

Ο AZP δεν είναι ο πιο αποδοτικός αλγόριθμος καθώς η αναζήτηση είναι τοπική και περιορίζεται κάθε φορά σε μία μόνο περιφέρεια. Μία πιο ορθή προσέγγιση θα ήταν η δημιουργία μιας λίστας από όλες τις περιφέρειες αυξάνοντας βέβαια και τον όγκο τον υπολογισμών αλλά και την απόδοση. Ένα άλλο μειονέκτημα είναι ότι η όλη διαδικασία ενίοτε "κολλάει". Αυτό είναι γενικώς αναμενόμενο ακόμα και σε πιο βελτιωμένους αλγορίθμους και λύνεται σχετικά ικανοποιητικά με αλλαγή της αρχικής ομαδοποίησης. Άλλα προβλήματα που είχε αυτή η προσπάθεια είναι η υλοποίηση της μεθόδου διότι όταν αναπτύχθηκε αυτή η διαδικασία τα πάντα γινόντουσαν με το χέρι και σε χαρτί, έτσι η επίλυση των θεμάτων της γειτνίασης τα οποία περιλαμβάνουν πίνακες ***NxN*** (όπου N ο αριθμός των ζωνών) ήταν σχεδόν αδύνατη για μεγάλο αριθμό δεδομένων. Όλα αυτά παρακάμφθηκαν με την ανάπτυξη των υπολογιστών και των GIS.

3.3 Παραλλαγή του AZP

Το κύριο πρόβλημα του AZP είναι ότι εγκλωβίζεται σε λύσεις οι οποίες δεν είναι βέλτιστες και εξαρτάται σε μεγάλο βαθμό από τον αρχικό διαχωρισμό. Προκειμένου να αντιμετωπιστεί αυτό το πρόβλημα, μία λύση είναι να εξετάζεται κάθε φορά η πιθανότητα μετακίνησης δύο ή τριών ζωνών μαζί. Κάτι τέτοιο βέβαια αυξάνει τις απαιτήσεις σε υπολογιστική ισχύ αλλά στις μέρες μας αυτό δεν είναι πρόβλημα και συνήθως το ερώτημα είναι πώς μπορούμε να έχουμε τα βέλτιστα αποτελέσματα.

Μία άλλη διαδεδομένη μέθοδος που χρησιμοποιείται ευρέως σε προβλήματα βελτιστοποίησης είναι ένας πιθανολογικός αλγόριθμος ο "simulated annealing" (SA). Ο αλγόριθμος αυτός αναπτύχθηκε για την επίλυση προβλημάτων φυσικής και αργότερα γενικεύθηκε και εφαρμόστηκε και σε άλλους τομείς. Χαρακτηριστικό αυτού του αλγορίθμου είναι η ικανότητά του

να ξεφεύγει από λύσεις μέτριας ποιότητας αναζητώντας την πραγματικά βέλτιστη. Η εφαρμογή του SA στον AZP γίνεται αλλάζοντας το βήμα 5 ως εξής:

Όταν η μετακίνηση βελτιώνει την αντικειμενική συνάρτηση γίνεται αποδεκτή, αν δεν την βελτιώνει γίνεται αποδεκτή με πιθανότητα που δίνεται από την εξίσωση του Boltzmann:

$$R(0,1) < \exp\left(\frac{\nabla f}{T(k)}\right)$$

όπου:

∇f Είναι η αλλαγή στην αντικειμενική συνάρτηση

$T(k)$ Είναι μία μονάδα βάρους η οποία εφαρμόζεται με βήμα k

$R(0,1)$ Ένας τυχαίος αριθμός ανάμεσα στο 0 και στο 1

Το δυσκολότερο κομμάτι είναι η εύρεση ενός τρόπου να μειώνονται σταδιακά οι μετακινήσεις οι οποίες χειροτερεύουν την αντικειμενική συνάρτηση έτσι ώστε να επιτυγχάνεται τελικά η βέλτιστη λύση σε λογικό χρονικό διάστημα. Για τον λόγο αυτό χρησιμοποιείται η "θερμοκρασία" $T(k)$ η οποία "κρυώνει" σταδιακά.

$$T(k) = \frac{T(0)}{\ln(k)}$$

Αυτό είναι αρκετά πιο αργό σε σχέση με τους περισσότερους αλγόριθμους οι οποίοι χρησιμοποιούν ένα εκθετικά ελαττούμενο σύστημα του τύπου:

$$T(k) = f\,T(k-1)$$

όπου f είναι συνήθως ανάμεσα στο 0.8 και 0.95.

Η δύναμη του SA είναι το ότι επιτρέπει μετακινήσεις οι οποίες σε αντίθεση με τον αντικειμενικό σκοπό είναι λανθασμένες αλλά με πιθανότητα η οποία σιγά σιγά ελαττώνεται. Ο αρχικός λοιπόν αλγόριθμος AZP έχει εμπλουτιστεί με μία εσωτερική επαναληπτική διαδικασία η οποία μειώνει το T από μία αρχικά μεγάλη τιμή. Το $T(0)$ οφείλει να είναι τέτοιο ώστε στην αρχή να γίνονται δεκτές περίπου οι μισές μετακινήσεις. Η εμπειρία έχει δείξει ότι τα αποτελέσματα με αυτήν την τροποποίηση βελτιώνονται σε μεγάλο βαθμό και καθώς η τεχνολογία των υπολογιστών έχει προχωρήσει αρκετά η διαφορά στο χρόνο εφαρμογής είναι πλέον αμελητέα. *(Openshaw and Rao 1994)*

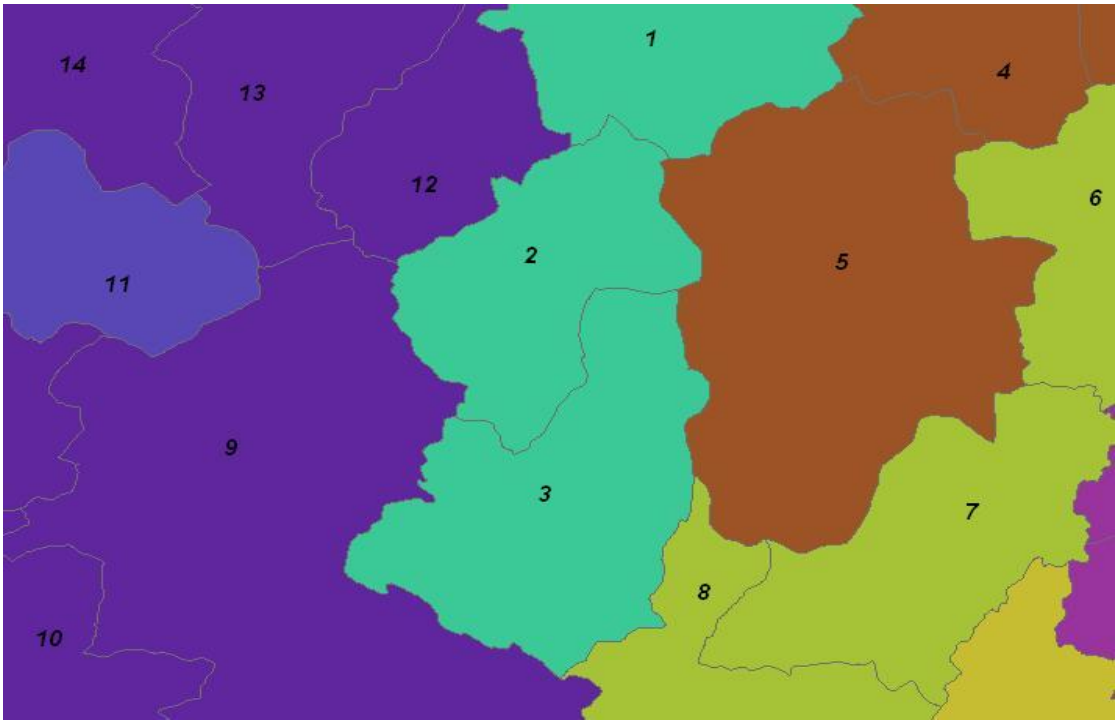
Κεφάλαιο 4ο

Το πρόβλημα της συνέχειας

4.1 Εισαγωγή

Ένας από τους περιορισμούς που τίθεται σε οποιαδήποτε περιφερειοποίηση με πρακτική αξία είναι εκείνος της συνέχειας. Η κάθε περιφέρεια οφείλει να είναι συνεχόμενη ώστε να είναι και λειτουργική. Αυτό οποίο σημαίνει ότι από κάθε σημείο της μπορεί κάποιος να κινηθεί προς ένα άλλο σημείο της ίδιας περιφέρειας χωρίς να διασχίσει μίαν άλλη. Το πρόβλημα παρόλο που φαίνεται σχετικά απλό, στην πραγματικότητα έχει πολλές δυσκολίες. Τα γεωγραφικά δεδομένα ποτέ δεν είναι ίδια και οι ιδιαιτερότητες κάθε περιοχής πρέπει να αντιμετωπίζονται διαφορετικά. Ένας τρόπος να αναπαρασταθεί η τοπολογία μίας περιοχής είναι με τη χρήση των τοπολογικών πινάκων. Είναι δυνατόν να αναπαρασταθεί η τοπολογία του χάρτη σε ένα πίνακα συνδέσεων (connectivity matrix) με στοιχεία c_{ij} όπου $c_{ij}=1$ αν οι ζώνες *i* και *j* έχουν κοινό σύνορο και $c_{ij}=0$ αν όχι.

Θεωρώντας λοιπόν ότι ξεκινάμε από μία τυχαία αρχική κατάσταση με συνεχείς περιφέρειες, ελέγχουμε σε κάθε πιθανή μετακίνηση εάν διατηρείται η συνέχειά τους. Στον παρακάτω χάρτη εμφανίζεται μία τέτοια τυχαία αρχική κατάσταση.



Εικόνα 3: Παράδειγμα χάρτη στην αρχική του κατάσταση

Από τον παραπάνω χάρτη παράγεται ο πίνακας τοπολογίας

	1	**2**	**3**	**4**	**5**	...
1	1	1	0	1	1	...
2	1	1	1	0	1	...
3	0	1	1	0	1	...
4	1	0	0	1	1	...
5	1	1	1	1	1	...
⋮	⋮	⋮	⋮	⋮	⋮	⋱

Πίνακας 1: Πίνακας τοπολογίας

Ένας προφανής και απλός τρόπος για να διατηρηθεί η συνέχεια των περιφερειών είναι να θέσουμε σε κάθε πιθανή μετακίνηση τον εξής περιορισμό: Η ζώνη η οποία αλλάζει περιφέρεια οφείλει να γειτνιάζει με την αποδέκτρια περιφέρεια. Ενώ κάτι τέτοιο ακούγεται ικανό να διατηρήσει την συνέχεια, στην πραγματικότητα δεν είναι. Το βασικό μειονέκτημα της παραπάνω προσέγγισης παράγεται από γεγονός ότι κάθε φορά λαμβάνονται υπόψη μόνο τα χαρακτηριστικά της αποδέκτριας περιφέρειας ενώ το πρόβλημα τίθεται στην περιφέρεια δότη. Αυτό δημιουργεί σε μία συγκεκριμένη περίπτωση απώλεια συνέχειας. Έστω ότι στον χάρτη της εικόνας 1 ελεγχόταν η μετακίνηση της ζώνης **12** προς την γαλάζια περιφέρεια. Θα προκύψει ο εξής πίνακας για την μοβ περιφέρεια:

	9	**10**	**13**	**14**
9	1	1	1	0
10	1	1	0	0
13	1	0	1	1
14	0	0	1	1

Πίνακας 2: Τοπολογικός πίνακας της μοβ περιφέρειας αν αφαιρεθεί η ζώνη 12

Από	Προς	λίστα ζωνών
		9,10,13,14
9	10,13	14
10	9	14
13	9,13,14	Συνέχεια αποκαταστάθηκε
14		

Πίνακας 3: Αναζήτηση συνέχειας στην μοβ περιφέρεια

Το παραπάνω παράδειγμα δεν παρουσιάζει κανένα πρόβλημα. Θα μελετήσουμε τώρα τι θα συνέβαινε αν γινόταν δοκιμή να περάσει η ζώνη 9 στην γαλάζια περιφέρεια. Στην περίπτωση αυτή η μετακίνηση θα ήταν αποδεκτή καθώς η ζώνη 9 είναι γειτονική με την γαλάζια περιφέρεια, στην πράξη όμως με την αλλαγή αυτή χάνεται η συνέχεια της μοβ περιφέρειας.

Η μέθοδος για τον έλεγχο αυτό η οποία εφαρμόστηκε από τον Openshaw και Rao (1991), εστιάζει στην περιφέρεια η οποία δημιουργείται αφού αφαιρεθεί από αυτή μία ζώνη. Η καρδιά της μεθόδου είναι η συνδεσιμότητα (connectivity) του χάρτη. Για κάθε υποσύνολο ζωνών υπάρχει και ο αντίστοιχος υποπίνακας τοπολογίας. Ο υποπίνακας για την περιφέρεια με γαλάζιο χρώμα φαίνεται στον πίνακα 2.

	1	**2**	**3**
1	1	1	0
2	1	1	1
3	0	1	1

Πίνακας 4: Υποπίνακας για περιφέρεια με γαλάζιο χρώμα

Προκειμένου να επιτευχθεί η συνέχεια της περιφέρειας αρκεί να υψώσουμε τον υποπίνακα της περιφέρειας εις την $n-1$ όπου n ο αριθμός των ζωνών της. Αν όλα τα στοιχεία του πίνακα που θα δημιουργηθεί είναι μη μηδενικά, τότε η συνέχεια διατηρείται. Αυτή η μέθοδος είναι σωστή όσο αφορά το μαθηματικό κομμάτι, αλλά απαιτεί υπερβολικό αριθμό υπολογισμών. Μέσα στις επαναληπτικές διαδικασίες εισάγονται πράξεις με πίνακες διαστάσεων κοντά στο 1000, οι οποίες ακόμα και με τους σύγχρονους υπολογιστές καθιστούν τον αλγόριθμο ιδιαίτερα χρονοβόρο.

Μία παραλλαγή της προηγούμενης μεθόδου είναι να γίνεται αναζήτηση "διαδρομών" με γειτονικά πολύγωνα τις ίδιας περιφέρειας. Αν μπορούν να συνδεθούν όλες οι ζώνες της περιφέρειας τότε θεωρείται συνεχής.

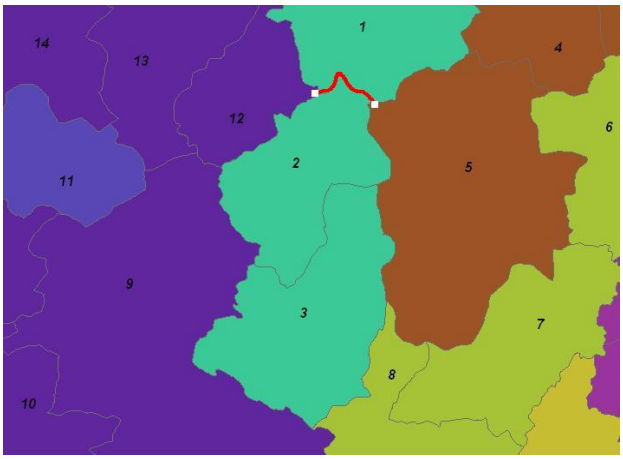
	1	**2**	**3**	**4**	**5**
1	1	1	0	1	1
2	1	1	1	0	1
3	0	1	1	0	1
4	1	0	0	1	1
5	1	1	1	1	1

Πίνακας 5: Αναζήτηση διαδρομών στον πίνακα τοπολογίας

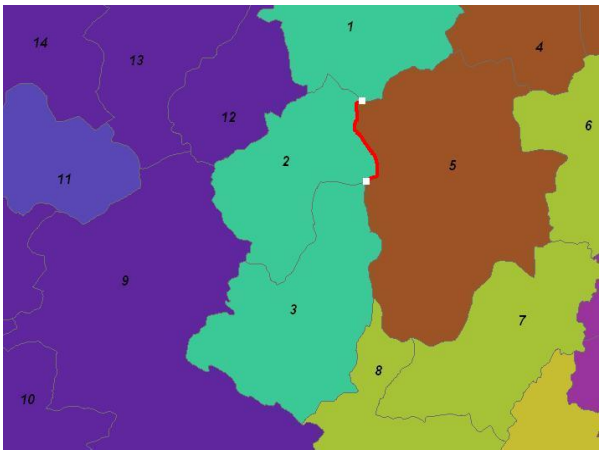
4.2Η μέθοδος των εναλλαγών (switching point)

Η μέθοδος αυτή αναπτύχθηκε από τον W. Macmillan (2000) και σε αντίθεση με τις προηγούμενες εστιάζει στην κάθε ζώνη και όχι στην περιφέρεια σαν σύνολο. Βασίζεται στην τοπολογική παρατήρηση ότι η απώλεια της συνέχειας συνεπάγεται την παραγωγή δύο ή περισσοτέρων νέων κομματιών της ίδιας περιφέρειας. Για να γίνει κατανοητή αυτή η προσέγγιση

πρέπει πρώτα να αποσαφηνιστεί η διάκριση μεταξύ εσωτερικού και εξωτερικού ορίου. Κάθε τόξο στο χάρτη είναι ένα όριο μεταξύ δύο ζωνών. Αν και οι δύο ζώνες ανήκουν στην ίδια περιφέρεια, το όριο λέγεται εσωτερικό, σε αντίθετη περίπτωση λέγεται εξωτερικό.



Εικόνα 4: Εσωτερικό Όριο

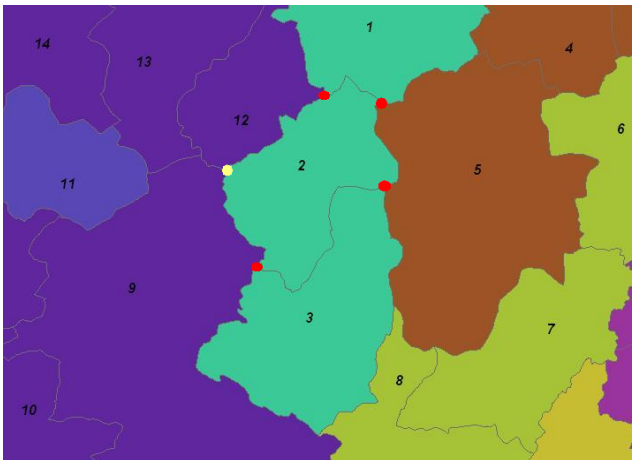


Εικόνα 5: Εξωτερικό Όριο

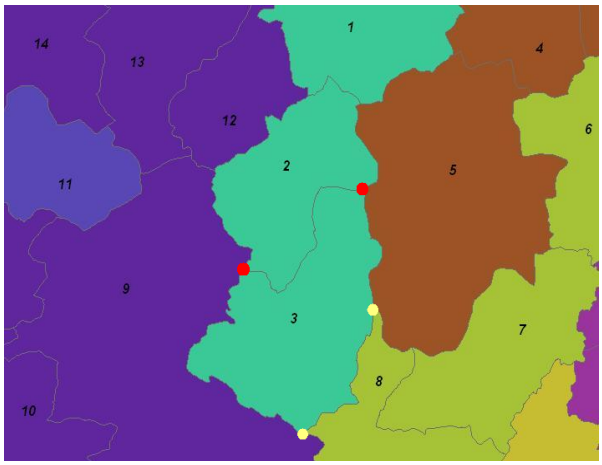
Η περίμετρος λοιπόν μιας ζώνης αποτελείται από ένα σύνολο εσωτερικών και εξωτερικών ορίων. Αν κινηθούμε περιμετρικά σε μία ζώνη, σε κάθε κόμβο το όριο μπορεί είτε να αλλάζει είτε να παραμένει σταθερό. Ο κόμβος στον οποίο γίνεται αλλαγή (από εσωτερικό σε εξωτερικό ή αντιστρόφως) ονομάζεται σημείο εναλλαγής. Ο αλγόριθμος, μετρώντας τα σημεία εναλλαγής σε κάθε πολύγωνο πριν από την πιθανή μετακίνηση, μπορεί να συμπεράνει αν η συνέχεια διατηρείται. Οι περιπτώσεις που προκύπτουν είναι οι εξής:

- 0 Σημεία εναλλαγής όταν η ζώνη δεν έχει σύνορο με άλλη περιφέρεια
- 2 Σημεία εναλλαγής όταν η ζώνη βρίσκεται στην άκρη της περιφέρειας
- 4 Σημεία εναλλαγής όταν η ζώνη αποτελεί συνδετικό κρίκο της περιφέρειας
- 6 Σημεία εναλλαγής όταν σπανίως η ζώνη συνδέει τρία τμήματα της περιφέρειας

Από τα παραπάνω γίνεται αντιληπτό ότι μόνο στην περίπτωση που έχουμε ακριβώς δύο σημεία εναλλαγής μπορούμε να δεχτούμε την μετακίνηση.



Εικόνα 6: Τέσσερα σημεία εναλλαγής



Εικόνα 7: Δύο σημεία εναλλαγής

Στην εικόνα 5 είναι εμφανές ότι αν αφαιρεθεί η ζώνη 3 από την γαλάζια περιφέρεια δεν θα υπάρξει πρόβλημα. Δεν συμβαίνει όμως το ίδιο και με την ζώνη 2, η οποία αν αφαιρεθεί θα δημιουργηθεί ασυνέχεια.

Συγκρίνοντας την μέθοδο αυτή με τις προηγούμενες του Openshaw παρατηρούμε δύο ουσιαστικές διαφορές. Η πρώτη αφορά τον όγκο δεδομένον ο οποίος στην περίπτωση των εναλλαγών είναι σαφώς μικρότερος. Η δεύτερη αφορά το πρακτικό κομμάτι της εφαρμογής. Η μέθοδος του Openshaw είναι καθαρά μαθηματική και συνεπώς δεν σφάλλει. Η προσέγγιση του Macmillan με της εναλλαγές έχει αρκετά δύσκολη υλοποίηση κυρίως στο κομμάτι που αφορά την κατασκευή της αλυσίδας των γειτονικών ζωνών. Υπάρχουν ορισμένες περιπτώσεις στις οποίες ο αλγόριθμος "μπερδεύεται" και παράγει λάθη. Ευτυχώς κάτι τέτοιο είναι αρκετά σπάνιο και μειώνει πολύ λίγο την αξία του αλγορίθμου αλλά πρέπει να αναφερθούν και αυτές οι περιπτώσεις. Η πρώτη περίπτωση όπου ενίοτε δημιουργούνται λάθη είναι όταν η ζώνη εφάπτεται με μία άλλη σε δύο διαφορετικά σημεία όπως στην εικόνα 6. Η δεύτερη περίπτωση είναι όταν τα γειτονικά πολύγωνα της ζώνης που μελετάμε γειτνιάζουν με το ίδιο πολύγωνο. Όπως για παράδειγμα στην εικόνα 7 με τη θάλασσα, στην περίπτωση αυτή ο αλγόριθμος δυσκολεύεται να κατασκευάσει την αλυσίδα των γειτονικών πολυγώνων.

Κεφάλαιο 5ο

Η εφαρμογή

5.1 Εισαγωγή

Προκειμένου να γίνει και μία πρακτική εφαρμογή των αλγορίθμων που μελετήθηκαν θέσαμε το εξής πρόβλημα προς επίλυση: Πως μπορεί να αναδιοργανωθεί η ελληνική επικράτεια σε εκλογικές περιφέρειες με βάση το γερμανικό εκλογικό σύστημα. Ως στοιχειώδης χωρική μονάδα χρησιμοποιήθηκαν οι δήμοι του Καποδίστρια στη νέα τους μορφή.

5.2 Το γερμανικό σύστημα

Η Γερμανία εκλέγει δύο βουλευτικά σώματα την **Federal Diet** (*Bundestag*) η οποία αποτελείται από 598 μέλη και το **Federal Council** (*Bundesrat*). Η *Bundestag* αποτελείται από 598 μέλη τα οποία εκλέγονται με τον εξής τρόπο: 299 εκλέγονται από μονοεδρικές περιφέρειες ενώ τα υπόλοιπα 299 από λίστες των πολιτικών παρατάξεων για όλες τις πολιτείες. Ο ψηφοφόρος καλείται να ψηφίσει δύο φορές, μία για εκπρόσωπο περιφέρειας και μία για πολιτική παράταξη. Η *Bundesrat* απαρτίζεται από 69 μέλη και περιέχει τους κυβερνήτες των πολιτειών.

Η εφαρμογή ενός παρόμοιου συστήματος στην Ελλάδα θα μπορούσε να γίνει με σχετικά λίγες αλλαγές στην διοικητική δομή της χώρας. Διατηρώντας το κοινοβούλιο ως έχει είναι δυνατόν να αλλάξουμε μόνο τον τρόπο με τον οποίο εκλέγονται οι εκπρόσωποι του λαού. Σύμφωνα λοιπόν με το γερμανικό σύστημα χωρίζουμε τον αριθμό των βουλευτών σε δύο ομάδες. Αυτοί που εκλέγονται κατευθείαν από μονοεδρικές περιφέρειες ανά την επικράτεια και αυτοί που εκλέγονται έμμεσα από τις πολιτικές παρατάξεις. Αυτό δίνει την δυνατότητα στον πολίτη να ενισχύσει τοπικό υποψήφιο διαφορετικής παράταξης από εκείνη που θα υποστηρίξει (ο πολίτης) στις εκλογές παράταξης. Αν χωρίσουμε λοιπόν τους βουλευτές στην μέση θα έχουμε 150 οι οποίοι εκλέγονται από μονοεδρικές περιφέρειες, συνεπώς το πρόβλημα ανάγεται γεωγραφικά στο να χωρίσουμε την χώρα σε 150 περιφέρειες.

Όπως έχει αναφερθεί και στα προηγούμενα κεφάλαια για να λυθεί ένα πρόβλημα περιφερειοποίησης πρέπει να οριστούν δύο βασικά πράγματα. Το πρώτο είναι η χωρική μονάδα που εδώ θα θεωρήσουμε τους καποδιστριακούς δήμους και το δεύτερο είναι τα κριτήρια με τα οποία θα γίνει ο χωρισμός. Για λόγους απλότητας στη συγκεκριμένη εφαρμογή σαν μοναδικό κριτήριο έχει ληφθεί ο πληθυσμός. Ο πληθυσμός των περιφερειών οι οποίες θα δημιουργηθούν πρέπει να είναι όσο το δυνατό πιο κοντά στο μέσο όρο. Η αντικειμενική συνάρτηση λοιπόν ο οποία καλείται να

βελτιστοποιηθεί είναι η απόκλιση του πληθυσμού των πιθανών περιφερειών από το θεωρητικό μέσο όρο.

Εδώ πρέπει να σημειωθεί ότι αντικειμενική συνάρτηση δεν είναι παρά ένας μαθηματικός τύπος ο οποίος μπορεί να αλλάξει πολύ απλά μέσα στην εφαρμογή. Στόχος της εργασίας δεν είναι να παράγει αποτελέσματα με πρακτική αξία η υλοποιήσιμα, αλλά να δείξει τον τρόπο με τον οποίο μπορεί να γίνει αυτό. Πιθανώς αν περικλείαμε στην αντικειμενική συνάρτηση ορισμένα κριτήρια κοινωνικά και οικονομικά να λαμβάναμε πιο ρεαλιστικά αποτελέσματα.

Η εφαρμογή κινήθηκε προς τρεις διαφορετικές κατευθύνσεις αλλά με τον ίδιο προορισμό. Όπως περιγράφτηκε στο θεωρητικό κομμάτι της εργασίας, οι προσεγγίσεις οι οποίες μπορούν να γίνουν είναι δύο: εκείνη κατά την οποία τα πολύγωνα χρησιμοποιούνται ως έχουν και εκείνη της σημειακής, κατά την οποία τα πολύγωνα θεωρούνται σημεία. Η τρίτη προσέγγιση έγινε κάνοντας μια προσέγγιση συνδυάζοντας τις δύο παραπάνω μεθόδους.

5.3 Προετοιμασία

Τα δεδομένα που χρησιμοποιήθηκαν περιέχονταν σε ένα αρχείο μορφής *shapefile* με τα όρια των δήμων και κοινοτήτων του Καποδίστρια. Προκειμένου να παραχθούν περισσότερες πληροφορίες έγιναν οι εξής διαδικασίες.

Μετατροπή του Shapefile σε Coverage

Οι δήμοι του Καποδίστρια είναι 1033 αλλά δεν καλύπτουν ένα πολύγωνο ο καθένας. Έτσι έχουμε 3025 πολύγωνα για 1033 δήμους. Τα θεματικά επίπεδα τα οποία παράγονται έχουν πληροφορίες που αφορούν τους δήμους, τα πολύγωνα των δήμων, την τοπολογία των πολυγώνων και τους πληθυσμούς των δήμων.

Συνένωση πολυγώνων-Δήμων

Το γεγονός ότι οι δήμοι έχουν πάνω από ένα πολύγωνο ο καθένας, τους καθιστά δύσκολους στην επεξεργασία. Συνεπώς έπρεπε να γίνει αντιστοίχηση δήμων με τα πολύγωνά τους. Αυτό γίνεται σχετικά εύκολα με χρήση του αντίστοιχου εργαλείου του ArcGIS. Με την χωρική συνένωση παράγεται ένα θεματικό επίπεδο που περιέχει την πληροφορία για το πού ανήκει κάθε πολύγωνο.

Υπολογισμός κεντροειδών

Όπως αναφέρθηκε και σε προηγούμενο κεφάλαιο, προκειμένου να γίνει η σημειακή

προσέγγιση απαιτείται ένα σημείο αναφοράς για κάθε ζώνη (δήμο). Στη συγκεκριμένη εφαρμογή το σημείο αυτό είναι το κεντροειδές του κάθε πολυγώνου. Αυτό υπολογίζεται και πάλι μέσω ενός εργαλείου του ArcGIS.

Τελική επεξεργασία

Όλα τα παραπάνω δεδομένα εξάγονται από το gis σε αρχεία κειμένου με σκοπό να χρησιμοποιηθούν από την εφαρμογή. Ορισμένα από αυτά ήταν απαραίτητο πρώτα να περάσουν από ένα ενδιάμεσο στάδιο τροποποίησης προκειμένου να έρθουν σε μορφή που να είναι εύκολα αναγνώσιμη. Τέλος δύο ζευγάρια δήμων έπρεπε να μετονομαστούν καθώς είχαν πανομοιότυπο όνομα και προκαλούσαν σύγχυση.

Αρχικά στάδια της Εφαρμογής

Η εφαρμογή όπως αναφέρθηκε έχει τρία κομμάτια, εντούτοις κάποια από τα στάδια επεξεργασίας είναι κοινά και για τα τρία. Τα στάδια αυτά περιλαμβάνουν κυρίως ρουτίνες επεξεργασίας και μορφοποίησης δεδομένων για εισαγωγή και εξαγωγή από την εφαρμογή (I/O), συναρτήσεις για υπολογισμούς κ.α. Η όλη εφαρμογή είναι χωρισμένη σε επιμέρους αρχεία ώστε ο χρόνος επεξεργασίας να μοιράζεται.

Δεδομένα εισόδου

Τα δεδομένα που διαβάζει το πρόγραμμα περιέχονται σε αρχεία κειμένου και είναι τα παρακάτω:

polyexp2.csv Περιέχει την λίστα των πολυγώνων μαζί με τα ονόματα των δήμων στους οποίους ανήκουν.

Ids2.csv Περιέχει μία απλή λίστα με τα ονόματα των δήμων.

LeftRight.csv Περιέχει όλη την πληροφορία της τοπολογίας. Για κάθε τόξο καταγράφονται τα δύο πολύγωνα τα οποία ενώνει.

Centroid2.csv Περιέχει λίστα των συντεταγμένων των κεντροειδών του κάθε δήμου.

Extra.txt Μερικές επιπλέον τοπολογικές πληροφορίες.

Starting.txt Πληροφορίες για την αρχική κατάσταση των περιφερειών.

Εισαγωγή των δεδομένων

Η βάση της εφαρμογής είναι ένας πίνακας ο οποίος περιέχει όλες τις πληροφορίες που αφορούν τους δήμους. Ο πίνακας αυτός έχει τα εξής πεδία:

[x] => Συντεταγμένη x του κεντροειδούς του δήμου

[y] => Συντεταγμένη y του κεντροειδούς του δήμου

[id] => Κωδικός αναγνώρισης του δήμου εντός της εφαρμογής

[fid] => Κωδικός αναγνώρισης του δήμου στο χάρτη

[sea] => Λαμβάνει τιμή 1 αν ο δήμος είναι νησί και 0 σε άλλη περίπτωση

[pop] => Ο πληθυσμός του δήμου

[names] => Λίστα με τα ονόματα του δήμου

[polys] => Λίστα με τα πολύγωνα που αποτελούν τον δήμο

[git] => Λίστα με τα *[id]* των δήμων που γειτνιάζουν με τον τρέχοντα

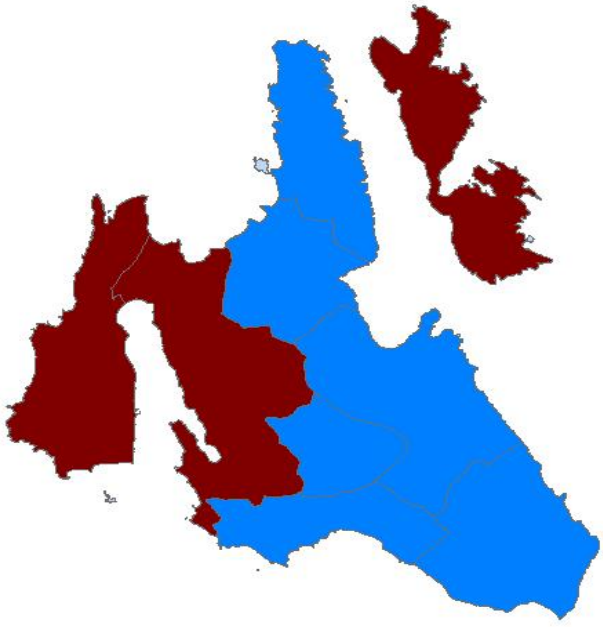
[perif] => Κωδικός της περιφέρειας στην οποία ανήκει ο δήμος

Και στις τρεις προσεγγίσεις ο παραπάνω πίνακας είναι κοινός. Η κάθε μία βέβαια τον τροποποιεί ανάλογα με την εξέλιξη του αντίστοιχου αλγορίθμου. Το πεδίο το οποίο είναι κυρίως μεταβλητό είναι αυτό του κωδικού της περιφέρειας *[perif]*. Το πρώτο λοιπόν στάδιο του προγράμματος είναι η κατασκευή αυτού του πίνακα με όλα τα σταθερά στοιχεία και με την τιμή 0 στο πεδίο *[perif]*. Ξεκινώντας διαβάζεται το αρχείο *polyexp.csv* και *ids.csv* και εισάγονται σε προσωρινό πίνακα. Στη συνέχεια διαβάζεται και το αρχείο με της πληροφορίες της τοπολογίας *LeftRight.csv* και συνδυάζονται σε νέο πίνακα. Τέλος διαβάζεται και το αρχείο με τις συντεταγμένες των κεντροειδών *Centroid2.csv* και ο πίνακας παίρνει την τελική του μορφή.

5.4 Το πρόβλημα των νησιών

Ένα από τα πιο δύσκολα προβλήματα που απαιτούσε προσπάθεια για να αντιμετωπιστεί είναι αυτό που προκαλεί η ιδιαιτερότητα του ελλαδικού χώρου με τον μεγάλο αριθμό νησιών. Η πρώτη και η τρίτη προσέγγιση που έγιναν βασίζονται στην παραδοχή ότι όλες οι χωρικές μονάδες έχουν έστω και ένα εφαπτόμενο όριο με κάποια άλλη ώστε να μπορούν να αλληλεπιδράσουν. Στην περίπτωση όμως των νησιών η τοπολογία μεταξύ των πολυγώνων είναι ιδιαίτερη καθώς δεν υπάρχει σημείο επαφής. Περαιτέρω δυσκολία προσδίδει το γεγονός ότι πολλά από τα νησιά αποτελούνται από περισσότερους από έναν δήμους, αυτό τα κάνει σχεδόν αδύνατο να διακριθούν από τους παραθαλάσσιους ηπειρωτικούς δήμους. Το παραπάνω ζήτημα είναι και ένας από τους

λόγους που αναπτύχθηκε η συνδυαστική μέθοδος που θα περιγραφεί αναλυτικά παρακάτω. Προκειμένου όμως να εφαρμοστεί η μέθοδος του Openshaw έγιναν δύο διαφορετικές κινήσεις. Η πρώτη ήταν να συνδεθούν τα νησιά με την ακτογραμμή και μεταξύ τους βάσει της απόστασής τους. Κάτι τέτοιο προχώρησε τον αλγόριθμο λίγο παρακάτω αλλά εκτός του ότι δεν έλυσε τελείως το πρόβλημα καθώς όπως προαναφέρθηκε τα μεγάλα νησιά δεν τα αντιλαμβάνεται το πρόγραμμα σαν νησιά, δημιουργήθηκε και ένα νέο σφάλμα που φαίνεται καθαρά στο παρακάτω σχήμα. Το πρόβλημα είναι ότι αν θέσουμε πολύ μικρή απόσταση προκειμένου να γίνονται σωστές συνδέσεις τότε πολλά νησιά παραμένουν ασύνδετα, αν όμως αυξήσουμε την απόσταση τότε γίνονται ορισμένες συνδέσεις οι οποίες είναι αθέμιτες.



Εικόνα 8: Παράδειγμα αθέμιτης σύνδεσης

Ο δεύτερος τρόπος είναι λιγότερο μαθηματικός αλλά περισσότερο αποτελεσματικός. Λόγω του σχετικά μικρού αριθμού είναι δυνατόν να δημιουργήσουμε μερικές συνδέσεις με το χέρι προκειμένου να βοηθήσουμε τον αλγόριθμο να εξελιχθεί. Το αρχείο *extra.txt* περιέχει ορισμένες τέτοιες συνδέσεις. Τελικά παράγεται ο πίνακας με όλα τα δεδομένα που προαναφέρθηκαν και είναι έτοιμος να χρησιμοποιηθεί από τους τρεις αλγορίθμους.

5.5 Το πρόβλημα των πληθυσμών

Ένα δεύτερο πρόβλημα που παρατηρήθηκε κατά την προετοιμασία των δεδομένων είναι η διασπορά του πληθυσμού στην επικράτεια. Στην Ελλάδα λόγω της αστυφιλίας και του σχεδιασμού των δήμων του Καποδίστρια παρατηρείται πολύ μεγάλη διακύμανση στους πληθυσμούς των δήμων. Υπάρχει για παράδειγμα ο δήμος των Αθηναίων με πληθυσμό περίπου 750.000 και ο δήμος

των Αντικυθήρων με 44. Αν λοιπόν στόχος είναι να έχουμε περιφέρειες με όσο το δυνατόν παρόμοιο πληθυσμό τότε ο μέσος όρος αυτός για 150 περιφέρειες θα είναι κοντά στις 70.000. Είναι προφανές ότι από μόνος του ο δήμος Αθηναίων, καθώς και αρκετοί άλλοι δήμοι, περιέχει πληθυσμό πολύ πάνω από τον επιθυμητό μέσο όρο. Κανένας από τους αλγορίθμους δεν είναι σε θέση να διασπάσει την χωρική μονάδα η οποία είναι ο δήμος ούτε κάτι τέτοιο είναι στους στόχους της εργασίας. Επιπλέον η αντικειμενική συνάρτηση χρησιμοποιεί πληθυσμούς και το γεγονός ότι υπάρχουν δήμοι με ακραίες τιμές καθιστά την πορεία τον αλγορίθμων προβληματική. Στο θέμα αυτό θα γίνει εμβάθυνση παρακάτω.

5.6 Η σημειακή προσέγγιση

Όπως αναφέρθηκε και στο θεωρητικό κομμάτι της εργασίας, κατά την σημειακή προσέγγιση ως χωρική μονάδα έχουμε ένα σημείο. Συνεπώς στους δήμους του Καποδίστρια θεωρούμε το κεντροειδές του κάθε δήμου ως βάση για την περιφερειοποίηση. Η λογική του αλγορίθμου είναι να δημιουργήσουμε κάποια κέντρα έλξης για τις περιφέρειες ώστε να συσχετιστούν με αυτά. Παράγοντας έναν αριθμό τυχαίων σημείων στο χάρτη, αντιστοιχούμε σε κάθε ένα από αυτά τις ζώνες οι οποίες είναι πιο κοντά σε αυτά. Η λύση αποθηκεύεται και υπολογίζεται η αντικειμενική συνάρτηση. Μετά από ένα αριθμό λύσεων επιλέγουμε τη βέλτιστη. Προκειμένου να βελτιωθεί το αποτέλεσμα αυτής της μεθόδου έγιναν και μερικές τροποποιήσεις σε σχέση με τον αλγόριθμο που περιγράφεται στην βιβλιογραφία.

Η απλή σημειακή προσέγγιση

Όπως παρατηρήθηκε και παραπάνω, η ύπαρξη πολύ μεγάλων σε πληθυσμό ζωνών δημιουργεί πρόβλημα. Ένας απλός και πρακτικός τρόπος να λυθεί αυτό είναι να θέσουμε ένα ανώτατο όριο πληθυσμού πάνω από το οποίο κάθε ζώνη θα θεωρείται περιφέρεια. Το όριο αυτό είναι κάτι λιγότερο από τον επιθυμητό μέσο όρο. Θεωρητικά ο αλγόριθμος θα έπρεπε να καταλήξει σε ένα τέτοιο συμπέρασμα από μόνος του, αλλά κάτι τέτοιο γίνεται δύσκολα και απαιτείται πολύς χρόνος επεξεργασίας χωρίς λόγο. Έτσι λοιπόν ο αλγόριθμος ξεκινάει δίνοντας ένα κωδικό περιφέρειας στις παρακάτω ζώνες:

Δήμος	Πληθυσμός	Δήμος	Πληθυσμός
Δ. ΧΑΝΙΩΝ	53373	Δ. ΙΛΙΟΥ	80859
Δ. ΡΟΔΟΥ	53709	Δ. ΑΜΑΡΟΥΣΙΟΥ	69470
Δ. ΚΑΛΑΜΑΤΑΣ	57620	Δ. ΑΧΑΡΝΩΝ	75341
Δ. ΓΛΥΦΑΔΑΣ	80409	Δ. ΠΑΤΡΕΩΝ	163446
Δ. ΠΕΙΡΑΙΩΣ	175697	Δ. ΧΑΛΚΙΔΕΩΝ	53584
Δ. ΠΑΛΑΙΟΥ ΦΑΛΗΡΟΥ	64759	Δ. ΑΓΡΙΝΙΟΥ	54253
Δ. ΑΓΙΟΥ ΔΗΜΗΤΡΙΟΥ	65173	Δ. ΛΑΜΙΕΩΝ	58601
Δ. ΗΛΙΟΥΠΟΛΕΩΣ	75904	Δ. ΒΟΛΟΥ	82439
Δ. ΝΕΑΣ ΣΜΥΡΝΗΣ	73986	Δ. ΤΡΙΚΚΑΙΩΝ	51862
Δ. ΚΑΛΛΙΘΕΑΣ	109609	Δ. ΛΑΡΙΣΑΣ	126076
Δ. ΒΥΡΩΝΟΣ	61102	Δ. ΙΩΑΝΝΙΤΩΝ	72896
Δ. ΖΩΓΡΑΦΟΥ	76115	Δ. ΚΑΤΕΡΙΝΗΣ	56434
Δ. ΚΕΡΑΤΣΙΝΙΟΥ	76102	Δ. ΚΑΛΑΜΑΡΙΑΣ	87255
Δ. ΝΙΚΑΙΑΣ	93086	Δ. ΘΕΣΣΑΛΟΝΙΚΗΣ	363987
Δ. ΚΟΡΥΔΑΛΛΟΥ	67456	Δ. ΕΥΟΣΜΟΥ	52624
Δ. ΑΙΓΑΛΕΩ	74046	Δ. ΚΑΒΑΛΑΣ	63293
Δ. ΑΓΙΑΣ ΠΑΡΑΣΚΕΥΗΣ	56836	Δ. ΑΛΕΞΑΝΔΡΟΥΠΟΛΗΣ	52720
Δ. ΑΘΗΝΑΙΩΝ	745514	Δ. ΞΑΝΘΗΣ	52270
Δ. ΓΑΛΑΤΣΙΟΥ	58042	Δ. ΣΕΡΡΩΝ	56145
Δ. ΠΕΡΙΣΤΕΡΙΟΥ	137918	Δ. ΚΟΜΟΤΗΝΗΣ	56453
Δ. ΧΑΛΑΝΔΡΙΟΥ	71684	Δ. ΔΡΑΜΑΣ	56387
Δ. ΝΕΑΣ ΙΩΝΙΑΣ	66017		

Στη συνέχεια υπολογίζονται τα όρια μεταξύ των οποίων θα βρίσκονται τα κέντρα έλξης των περιφερειών. Δεν είναι απαραίτητο να είναι ανάμεσα σε όρια αλλά έτσι τα αποτελέσματα τείνουν να είναι καλύτερα. Τα όρια είναι το μέγιστο και ελάχιστο x και y του χάρτη:

$$X_{min} = 106098\,m$$
$$X_{max} = 1001673\,m$$
$$Y_{min} = 3855755\,m$$
$$Y_{max} = 4614673\,m$$

Στη συνέχεια παράγονται *n* ζεύγη από *k* φορές και αποθηκεύονται σε ένα προσωρινό πίνακα. Κάθε σετ από *n* ζεύγη αποτελεί και μία λύση του προβλήματος. Για κάθε λύση από 1 έως *k* υπολογίζεται η αντικειμενική συνάρτηση και το βέλτιστο αποτέλεσμα είναι αυτό που τελικά εξάγεται σε ένα αρχείο. Το αρχείο αυτό περιέχει ένα κλειδί προκειμένου να γίνει η αντιστοίχηση της κάθε ζώνης με την αντίστοιχη ζώνη στο χάρτη, καθώς και τον κωδικό της περιφέρειας.

Εντυπώσεις από την εφαρμογή

Η πρώτη θετική εντύπωση από την πρώτη αυτή προσέγγιση είναι η ευκολία στην ανάπτυξή της. Αφαιρώντας τα βοηθητικά κομμάτια που έχουν να κάνουν με την είσοδο και έξοδο των δεδομένων δεν μένουν παρά μια με δύο σελίδες κώδικα. Ο χρόνος επεξεργασίας για έναν αριθμό της τάξεως των *2000* επαναλήψεων είναι περίπου *15-20* λεπτά το οποίο είναι αρκετά

ικανοποιητικό. Ένα τελευταίο ιδιαίτερα σπουδαίο θετικό στοιχείο είναι το ότι ο αλγόριθμος αυτός δεν επηρεάζεται καθόλου από την τοπολογία των πολυγώνων. Είτε αυτά είναι νησιά, είτε ανήκουν στην ηπειρωτική χώρα, τα μεταχειρίζεται με τον ίδιο τρόπο. Τα αρνητικά όμως της μεθόδου αυτής είναι αρκετά και ουσιώδη. Καταρχάς όπως αναφέρθηκε και στο θεωρητικό κομμάτι, σε ορισμένες περιπτώσεις υπάρχει απώλεια της συνέχειας των περιφερειών. Αυτό ορισμένες φορές είναι αρκετά έντονο ώστε η λύση παρόλο που είναι σχετικά καλή από άποψη πληθυσμών είναι δυσλειτουργική λόγω της ασυνέχειας. Δεύτερο μειονέκτημα είναι ότι ο αριθμός τον περιφερειών δεν είναι σταθερός και προφανώς δεν είναι πάντα ο επιθυμητός. Αυτό συμβαίνει διότι πολλές φορές κάποια σημεία έλξης είναι σε τέτοιες θέσεις ώστε καμία ζώνη να μην έλκεται από αυτά λόγω απόστασης. Τρίτο μειονέκτημα είναι το ότι παράγονται περιφέρειες με πληθυσμούς οι οποίοι αποκλίνουν πολύ από τον επιθυμητό. Αυτό συμβαίνει και προς τα πάνω και προς τα κάτω, δηλαδή έχουμε περιφέρειες και με πολύ μεγάλο πληθυσμό και με πολύ μικρό. Ένα τυπικό αποτέλεσμα μετά από μελέτη *2000* πιθανών λύσεων και επιλογή της βέλτιστης είναι ο πίνακας 6 πληθυσμών. Παρατηρείται μία ελαφριά σύγκλιση προς τον μέσο όρο αλλά κάποιες περιφέρειες αποκλείουν υπερβολικά.

Κωδικός περιφέρειας	Πληθυσμός	Κωδικός περιφέρειας	Πληθυσμός
44	6669	67	36108
46	6740	69	430
47	14645	70	18583
48	150802	73	82615
49	83416	74	65953
50	405837	75	242132
51	217668	76	117712
52	91151	78	677582
53	296831	79	90477
54	448980	80	2511
55	63298	81	1086167
56	27933	84	203313
57	322483	85	25827
59	373550	87	19440
62	130054	88	28390
63	798856	89	12119
64	252564	90	47360
66	8184	92	8612

Πίνακας 6: Τυπικοί πληθυσμοί περιφερειών με την απλή σημειακή προσέγγιση

5.7 σημειακή προσέγγιση με διαχωρισμό

Μελετώντας τα παραπάνω αποτελέσματα βγήκε το συμπέρασμα ότι ο αλγόριθμος αδυνατεί να συγκλίνει διότι είναι σχεδόν αδύνατο τα κέντρα έλξης να τοποθετηθούν σε σωστά σημεία. Προκειμένου να παρακαμφθεί το παραπάνω πρόβλημα χωρίστηκε ο χάρτης της Ελλάδας σε

μικρότερες περιοχές με τη μορφή ενός καννάβου ώστε ο αλγόριθμος να είναι πιο ακριβής. Ο χωρισμός αυτός έγινε με βάση την πυκνότητα των πληθυσμών στην ελληνική επικράτεια συνεπώς ο κάνναβος δεν έχει σταθερές διαστάσεις. Ο κάνναβος κατασκευάστηκε σταδιακά και τροποποιήθηκε αρκετές φορές καθώς τα αποτελέσματα επηρεάζονται πολύ από της διαστάσεις του. Τυπικά αποτελέσματα με την χρήση καννάβου φαίνονται στον παρακάτω πίνακα:

Κωδικός περιφέρειας	Πληθυσμός	Κωδικός περιφέρειας	Πληθυσμός
45	82547	89	87054
46	89941	90	54513
47	58383	91	112878
48	62175	92	95311
49	95377	94	80931
50	58110	95	56762
51	60503	96	78058
52	100723	97	78737
53	96254	98	83416
54	69966	99	98539
55	87916	100	74273
56	7166	101	72640
57	77442	102	71099
58	90598	103	88782
59	95917	104	70399
60	92247	105	49781
62	38022	106	85312
63	35675	107	52837
64	4106	108	91773
65	114452	109	82214
66	18128	110	70970
67	3152	111	90387
68	401206	112	84122
69	311542	113	103769
70	73458	114	75808
72	8147	115	56802
74	67354	116	69085
75	41941	117	19241
76	23525	118	87320
77	68145	119	105907
79	210562	120	165749
80	90867	122	18663
81	3022	124	241243
82	25715	125	27172
83	71048	126	9543
84	98823	127	77948
85	75578	128	84833
86	67949	129	94860
87	64402	130	74859
88	95116		

Πίνακας 7: Τυπικοί πληθυσμοί περιφερειών με σημειακή προσέγγιση και χρήση καννάβου

Εντυπώσεις από την εφαρμογή της σημειακής προσέγγισης με κάνναβο

Η πρώτη βασική παρατήρηση για αυτήν την υλοποίηση σε σχέση με την απλή μέθοδο είναι η ποιότητα το αποτελεσμάτων. Παρατηρούμε ότι υπάρχει σαφής βελτίωση στους πληθυσμούς των περιφερειών και πολύ μεγαλύτερη σύγκλιση προς τη μέση τιμή. Οι ακραίες τιμές έχουν μειωθεί κατά πολύ και είναι πλέον ρεαλιστικές. Επίσης είναι εμφανές ότι όσο μικραίνει το μέγεθος του καννάβου τόσο βελτιώνονται τα αποτελέσματα. Στους παρακάτω πίνακες φαίνεται συγκριτικά ο χωρισμός δύο διαφορετικών τετραγώνων του καννάβου. Ο πρώτος πίνακας περιγράφει μια περιοχή με πληθυσμό *293.000* κατοίκων ενώ ο δεύτερος μια περιοχή *1.060.000* κατοίκων.

Κωδικός περιφέρειας	Πληθυσμός
46	89941
48	62175
45	82547
47	58383

Πίνακας 9: Συνολικός πληθυσμός 293.000

Κωδικός περιφέρειας	Πληθυσμός
127	77948
120	165749
116	69085
124	241243
119	105907
115	56802
114	75808
118	87320
113	103769
126	9543
125	27172
122	18663
117	19241

Πίνακας 8: Συνολικός πληθυσμός 1.060.000

Στο παράρτημα ο χάρτης 2 παρουσιάζει οπτικοποιημένα τα αποτελέσματα αυτής της μεθόδου. Εκτός από ορισμένες περιπτώσεις ασυνέχειας τις οποίες αναμέναμε άλλωστε, γενικά οι περιφέρειες είναι αρκετά καλής ποιότητας και γεωμετρικά. Η δυνατότητα του αλγορίθμου να διαχειρίζεται τα νησιά όσο καλά διαχειρίζεται και τους χερσαίους δήμους φαίνεται καθαρά στο χάρτη και σε σχέση με την μέθοδο του Openshaw η οποία θα περιγραφεί παρακάτω τα αποτελέσματα είναι σαφώς καλύτερα. Ένα ακόμα πλεονέκτημα είναι ότι όλοι οι δήμοι ανήκουν πάντα σε μία περιφέρεια, κάτι που δεν είναι τόσο προφανές, ειδικά για τα νησιά, στον αλγόριθμο του openshaw.

5.8 Ο αλγόριθμος του Openshaw

Ο αλγόριθμος αυτός περιγράφηκε διεξοδικά στο θεωρητικό κομμάτι της εργασίας, όμως στην πράξη συναντήθηκαν πολλές δυσκολίες. Αυτό οφείλεται κυρίως στην ευαισθησία του

αλγορίθμου στην μορφή των δεδομένων καθώς και στην πολυπλοκότητά του. Η πορεία της εφαρμογής είναι η εξής. Καταρχάς διαβάζεται ο πίνακας με τα δεδομένα ο οποίος έχει κατασκευαστεί κατά τη διαδικασία της προετοιμασίας. Στη συνέχεια γίνονται κάποιες κινήσεις για να δημιουργηθεί μια αρχική κατάσταση διαχωρισμού. Ο αλγόριθμος θα λάβει αυτό τον αρχικό διαχωρισμό και με σταδιακές μεταθέσεις δήμων θα προσπαθήσει να προσεγγίσει την βέλτιστη λύση. Για τη δημιουργία της αρχικής κατάστασης όπως και στη σημειακή προσέγγιση πρώτα εντοπίζονται οι περιφέρειες με πληθυσμούς κοντά στο μέσο όρο και δίνεται σε αυτές κωδικός περιφέρειας. Στη συνέχεια υπάρχουν δύο δυνατότητες, η μία είναι να επιλεγούν στην τύχη τόσες ζώνες όσες είναι και οι περιφέρειες που απομένουν και η άλλη είναι να επιλεγούν "με το χέρι" οι ζώνες αυτές. Η πορεία του αλγορίθμου δεν επηρεάζεται από αυτή την επιλογή, επηρεάζονται όμως τα αποτελέσματα. Η σύγκριση αυτή θα γίνει στο τέλος. Στη συνέχεια ανεξάρτητα με ποιο τρόπο επιλεγούν αυτές οι πρώτες ζώνες, οι περιφέρειες αρχίζουν να μεγαλώνουν ενσωματώνοντας γειτονικές ζώνες. Η διαδικασία θεωρητικά ολοκληρώνεται όταν δεν υπάρχει ζώνη που να μην ανήκει σε καμία περιφέρεια. Εδώ όμως συναντήθηκε η πρώτη δυσκολία, καθώς μετά από αρκετές επαναλήψεις ορισμένες ζώνες παρέμεναν ανεξάρτητες. Αυτό συμβαίνει διότι όλες οι ζώνες δεν είναι συνδεδεμένες μεταξύ τους. Το πρόβλημα προφανώς προκαλείται από την ιδιομορφία των νησιών. Η τοπολογία ανάμεσα στα νησιά και στην ηπειρωτική χώρα είναι τέτοια ώστε δεν επιτρέπει τη σύνδεσή τους. Για να λυθεί αυτό το πρόβλημα έπρεπε είτε να δημιουργηθούν δεσμοί και πάλι "με το χέρι" είτε να αγνοηθεί αυτή η ανωμαλία και να διευθετηθεί στο τέλος πάλι με ανθρώπινη παρέμβαση. Είναι βέβαια κατανοητό ότι όσο αυξάνονται οι εξωτερικές παρεμβάσεις τόσο μειώνεται ο αυτοματισμός του αλγορίθμου και συνεπώς η ποιότητά του. Τελικά επιλέχτηκε μία μέση οδός, δηλαδή έγιναν ορισμένες τεχνητές συνδέσεις από πριν αλλά αφέθηκαν και μερικές ζώνες ασύνδετες. Οι ζώνες αυτές είναι κυρίως εκείνες οι οποίες δεν είναι σαφές με ποια άλλη ζώνη πρέπει να συνδεθούν. Αφού γίνουν όλα τα παραπάνω, ο πίνακας εξάγεται σε αρχείο κειμένου και το πρόγραμμα μπαίνει στην κυρίως διαδικασία του αλγορίθμου:

- Πρώτο βήμα είναι να επιλεγούν ποια πολύγωνα είναι πιθανά να μετακινηθούν. Θα ήταν δυνατό να δοκιμαστούν όλα αλλά κάτι τέτοιο θα απαιτούσε πολύ χρόνο. Συνεπώς επιλέγονται μόνο οι ζώνες οι οποίες έχουν κοινό όριο με κάποια άλλη περιφέρεια και εισάγονται σε μία λίστα έστω Α.
- Για κάθε στοιχείο της λίστας αυτής βρίσκονται οι περιφέρειες με τις οποίες γειτνιάζουν και εισάγονται σε νέα λίστα έστω Β.
- Κατασκευάζεται νέος πίνακας προσωρινός ίδιος με τον αρχικό και για κάθε στοιχείο της λίστας Β υπολογίζεται η αντικειμενική συνάρτηση.
- Υπολογίζεται και η αντικειμενική συνάρτηση για τον αρχικό πίνακα και η μετακίνηση

γίνεται αποδεκτή αν ισχύουν και οι τρεις παρακάτω συνθήκες.

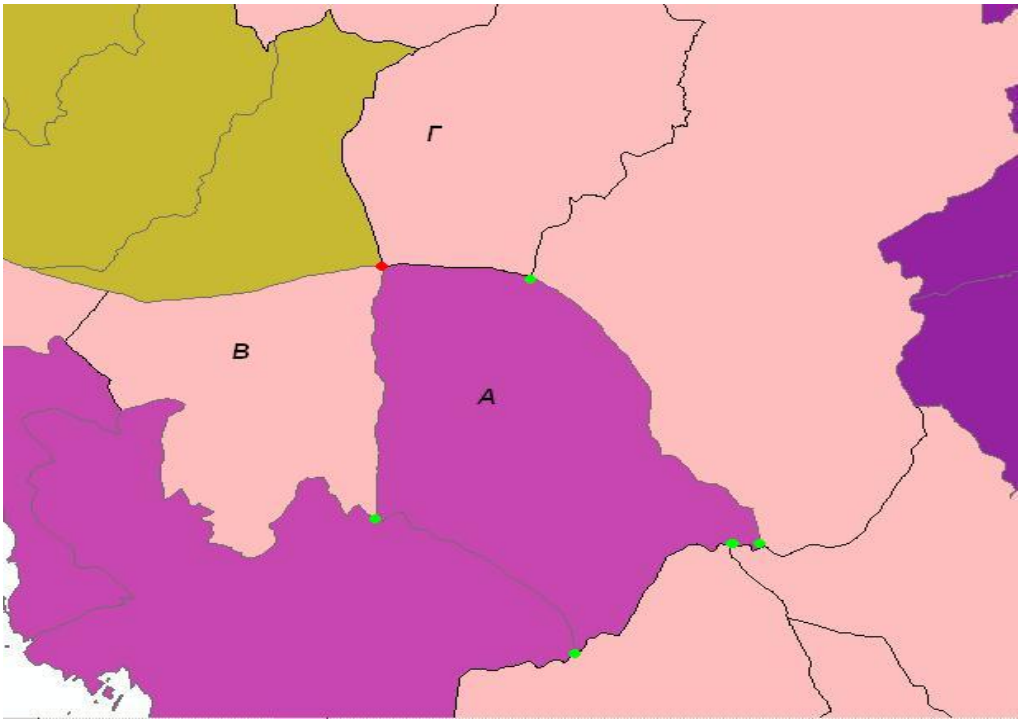
- Η αντικειμενική συνάρτηση βελτιώνεται
- Τα σημεία εναλλαγής είναι ακριβώς δύο
- Η περιφέρεια δότης έχει πάνω από ένα πολύγωνα

- Η διαδικασία επαναλαμβάνεται για έναν ορισμένο αριθμό επαναλήψεων

Εδώ πρέπει να σημειωθεί ότι κατά την εφαρμογή της μεθόδου των σημείων εναλλαγής (switching points) δεν χρησιμοποιήθηκαν οι κόμβοι όπως αναφέρεται στη βιβλιογραφία αλλά ολόκληρες οι περιφέρειες. Η διαδικασία είναι αρκετά πολύπλοκη και σχετικά χρονοβόρα και δεν περιγράφεται πολύ αναλυτικά στη βιβλιογραφία. Τα βήματα είναι τα εξής:

1. Για το πολύγωνο που μελετάται βρίσκονται τα γειτονικά πολύγωνα και εισάγονται σε μία λίστα ***A***
2. Επιλέγεται ένα τυχαίο πολύγωνο από τη λίστα ***A***, αφαιρείται από αυτήν και εισάγεται σε μία νέα λίστα ***B*** ως πρώτο στοιχείο
3. Για το τελευταίο στοιχείο της λίστας ***B*** βρίσκεται το γειτονικό της πολύγωνο το οποίο όμως ανήκει στην λίστα ***A***. Το πολύγωνο αυτό αφαιρείται από τη λίστα ***A*** και εισάγεται στο τέλος της ***B***.
4. Τα δύο προηγούμενα βήματα επαναλαμβάνονται έως ότου η λίστα ***A*** αδειάσει. Αν αυτό δεν γίνει μετά από έναν λογικό αριθμό επαναλήψεων τότε αναζητείται τρόπος να κλείσει το δαχτυλίδι των πολυγώνων με ένα ενδιάμεσο κενό.
5. Τέλος αντιγράφεται το πρώτο στοιχείο της λίστας ***B*** και εισάγεται και ως τελευταίο. Για τα ζεύγη ***n*** και ***n+1*** καταγράφεται αν υπάρχει σημείο εναλλαγής πράγμα το οποίο επαναλαμβάνεται για όλα τα στοιχεία της λίστας.

Το δεύτερο σκέλος του 4ου βήματος ήταν απαραίτητο καθώς ο αλγόριθμος κολλούσε σε περιπτώσεις σαν αυτή του παρακάτω σχήματος:



Εικόνα 9: Ειδική περίπτωση στη μέθοδο σημείων εναλλαγής

Παρατηρούμε ότι η τοπολογία ανάμεσα στα πολύγωνα ***Α*** και ***Γ*** είναι μόνο ένα σημείο επαφής. Συνεπώς δεν θεωρούνται γειτονικά. Για να λυθεί αυτό το πρόβλημα τροποποιήθηκε το βήμα αυτό ώστε να δανείζεται ένα πολύγωνο ενδιάμεσα παρόλο που δεν είναι γειτονικό με την βασική ζώνη.

Μια άλλη δυσκολία που συναντήθηκε και έγινε προσπάθεια να αντιμετωπιστεί είναι οι περιπτώσεις στις οποίες ένα σύνολο ζωνών περικλείεται ολικά από μία άλλη ζώνη όπως φαίνεται στην παρακάτω εικόνα:

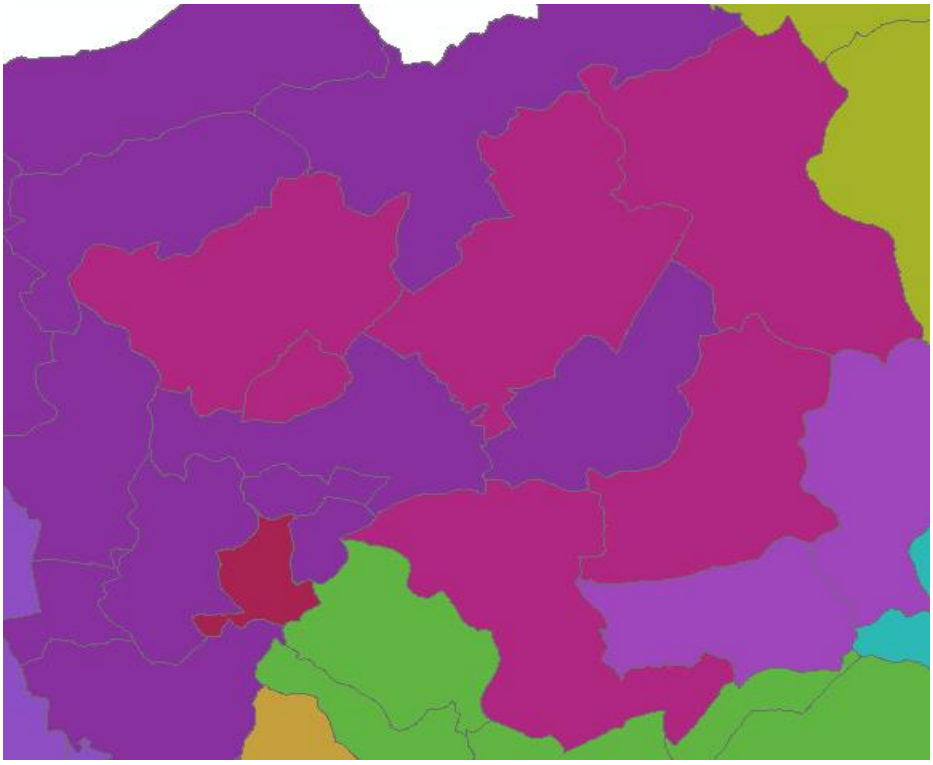

Εικόνα 10: Περικυκλωμένη περιφέρεια

Στην περίπτωση αυτή ο αλγόριθμος δεν προχωράει παρακάτω καθώς δεν μπορεί να δώσει ζώνες

στην εσωτερική περιφέρεια διότι παραβιάζεται ο κανόνας των σημείων εναλλαγής. Για να απεγκλωβιστεί ο αλγόριθμος υπάρχει μία υπορουτίνα που εκτελείται μία φορά στην αρχή του προγράμματος και μία στο τέλος η οποία αναλαμβάνει να βρει όλες τις περιφέρειες οι οποίες περιβάλλονται ολοκληρωτικά από κάποια άλλη και να σπάσει την αλυσίδα σε κάποιο σημείο.

Εντυπώσεις από την εφαρμογή της μεθόδου του Openshaw

Το πρώτο θετικό που παρατηρούμε είναι ο χρόνος επεξεργασίας ο οποίος είναι αρκετά μικρός. Συγκρίνοντας τη μέθοδο αυτή με τις προηγούμενες η σύγκλιση επέρχεται περίπου στο ένα τρίτο του χρόνου. Αυτό συμβαίνει διότι ο αλγόριθμος αυτός επικεντρώνεται κατευθείαν στις προβληματικές περιφέρειες και προσπαθεί να τις διορθώσει. Ένα ακόμα θετικό είναι ότι τα αποτελέσματα είναι μοναδικά και αφού επέλθει η σύγκλιση είμαστε σίγουροι ότι δεν υπάρχει περαιτέρω βελτίωση. Επίσης στα δυνατά σημεία του αλγορίθμου πρέπει να σημειωθεί η καλή διαχείριση του θέματος της συνέχειας των περιφερειών, κάτι στο οποίο πάσχει η σημειακή μέθοδος. Βασικό αρνητικό του αλγορίθμου είναι η πολυπλοκότητά του η οποία την κάνει επιρρεπή σε σφάλματα και δύσκολη στην εφαρμογή. Η ύπαρξη πολλών περιπτώσεων κυρίως στις τοπολογίες απαιτεί και μεγάλο αριθμό συναρτήσεων για να τις καλύπτουν, με αποτέλεσμα να μειώνεται η ακρίβεια. Σε αυτό βέβαια συμβάλλει και η ιδιομορφία των δεδομένων όπως έχει αναφερθεί και νωρίτερα καθώς καθιστούν την ύπαρξη πολλών ρουτίνων επεξεργασίας απαραίτητη. Άλλο αρνητικό είναι η αδυναμία του αλγορίθμου να χειριστεί τοπολογίες διαφορετικές από τις εφαπτομενικές. Έτσι τα νησιά μένουν εκτός της διαδικασίας. Ένα άλλο αρνητικό που θα αναλυθεί και σε ξεχωριστό κεφάλαιο είναι η παραγωγή από τον αλγόριθμο μεγάλης ποσότητας “Gerrymandering”, το οποίο σημαίνει ότι πολλές περιφέρειες έχουν σχήματα αλλόκοτα καθιστώντας τες άσχημες αισθητικά αλλά και κακές λειτουργικά. Ένα τέτοιο παράδειγμα φαίνεται στην παρακάτω εικόνα:



Εικόνα 11: Περίπτωση έντονου Gerrymandering

Τέλος πρέπει να σημειωθεί ότι ούτε και σε αυτή την περίπτωση τα αποτελέσματα είναι ιδανικά. Ο αλγόριθμος αυτός φαίνεται να είναι πιο δυνατός στην ηπειρωτική χώρα και μακριά από την ακτογραμμή. Επίσης παρατηρείται μία σχετικά μεγάλη ευαισθησία του αλγορίθμου στις αλλαγές στην αρχική κατάσταση. Με διαφορετική αρχική περιφερειοποίηση τα αποτελέσματα μεταβάλλονται ριζικά. Αυτό βέβαια δικαιολογείται από την φύση του αλγορίθμου αλλά θα περιμέναμε να επηρεάζεται μόνο η μορφή των αποτελεσμάτων και όχι η ποιότητά τους.

5.9 Μια συνδυαστική προσέγγιση

Μελετώντας τα προτερήματα και μειονεκτήματα των δύο προηγούμενων αλγορίθμων καταλήγουμε στο συμπέρασμα ότι κανένας από τους δύο δεν καλύπτει αποδοτικά όλες τις περιπτώσεις. Πιθανώς όμως ένας συνδυασμός τους θα μπορούσε να αποφέρει βελτιωμένα αποτελέσματα. Έτσι για παράδειγμα το πρόβλημα της ελλιπούς σύνδεσης των νησιών αντιμετωπίζεται με επιτυχία στη σημειακή προσέγγιση αλλά όχι στον αλγόριθμο του Openshaw. Επίσης παρατηρήθηκε ότι η αρχική κατάσταση επηρεάζει σε μεγάλο βαθμό την πορεία του αλγορίθμου του Openshaw. Το σκεπτικό είναι να λάβουμε τα αποτελέσματα του αλγορίθμου της σημειακής προσέγγισης και να τα εισάγουμε ως δεδομένα αρχικής κατάστασης στον αλγόριθμο του Openshaw. Ο κίνδυνος σε αυτό το εγχείρημα είναι να διογκωθεί το πρόβλημα της ασυνέχειας του σημειακού αλγορίθμου διότι στις πιθανές ασυνέχειες που θα δημιουργηθούν από τη σημειακή

προσέγγιση θα προσαρτηθούν περισσότεροι δήμοι.

Τα βήματα που ακολουθούνται στα δύο επιμέρους κομμάτια είναι περίπου ίδια με αυτά που περιγράφτηκαν ξεχωριστά στα δύο προηγούμενα κεφάλαια οπότε εδώ θα επικεντρωθούμε στα σημεία που διαφοροποιούνται. Στο πρώτο βήμα λοιπόν ο σημειακός αλγόριθμος αναλαμβάνει να περιφερειοποιήσει τόσο τα νησιά όσο και την ηπειρωτική χώρα ενώ στο δεύτερο ο αλγόριθμος του Openshaw τελειοποιεί το αποτέλεσμα κάνοντας πιθανές μετακινήσεις. Εδώ πρέπει να σημειωθεί ότι στην προσπάθεια αυτή δοκιμάστηκε η βελτιωμένη προσέγγιση "Simulated annealing" η οποία περιγράφηκε στο θεωρητικό κομμάτι της εργασίας. Συνοπτικά, η μέθοδος αυτή προσπαθεί να προσεγγίσει την βέλτιστη λύση κάνοντας αποδεκτές στα αρχικά στάδια της διαδικασίας μετακινήσεις οι οποίες φαίνεται να είναι μη αποδοτικές.

Τα αποτελέσματα, τα οποία φαίνονται στους παρακάτω πίνακες παρουσιάζουν ιδιαίτερο ενδιαφέρον. Παρατηρείται εντυπωσιακά καλή σύγκλιση προς τον επιθυμητό μέσο όρο εκτός από μία με δύο περιπτώσεις οι οποίες πιθανώς να εξαλειφθούν με μερικές επαναλήψεις. Τα αποτελέσματα των πληθυσμών παρουσιάζονται στον παρακάτω πίνακα μαζί με τους κωδικούς της κάθε περιφέρειας.

Κωδικός περιφέρειας	Πληθυσμός	Κωδικός περιφέρειας	Πληθυσμός
1	71534	31	70623
2	70985	32	126076
3	57620	33	72896
4	54787	34	69494
6	64759	35	87255
7	65173	36	363987
8	75904	37	94277
9	73986	38	63293
10	109609	39	56055
11	72241	40	56779
12	76115	41	57417
13	76102	42	56453
14	160542	43	56387
15	94603	45	48774
16	104608	46	70528
17	76539	47	48628
18	745514	48	70591
19	124059	49	72370
20	137918	50	74286
21	97266	51	64356
22	92261	52	73401
23	80859	53	70940
24	88996	54	57815
25	80121	55	51777
26	163446	57	52765
27	69538	58	53447
28	67596	59	49661
29	70125	60	48274
30	82439	31	70623

Πίνακας 10: Αποτελέσματα Συνδυαστικής προσέγγισης

Κωδικός περιφέρειας	Πληθυσμός	Κωδικός περιφέρειας	Πληθυσμός
61	66432	91	67777
62	49273	92	68838
63	46377	93	67980
64	89415	94	67414
65	59035	95	69154
67	74432	96	74446
68	49347	97	5302
69	77939	98	69835
70	70478	99	69174
71	46453	100	71896
72	46090	101	72202
73	47725	102	71403
74	76850	103	71817
75	48079	104	70475
76	69626	105	61568
78	48689	106	61486
79	88887	108	64289
80	70685	109	69095
81	72454	110	69022
82	72674	111	69197
83	48156	112	68707
84	94575	113	68797
85	72669	114	69909
86	48938	115	69449
87	42972	116	69451
88	48767	117	69051
89	66387	118	69131
90	67298	119	72680
61	66432	120	68342

Πίνακας 11: Αποτελέσματα Συνδυαστικής προσέγγισης

Κωδικός περιφέρειας	Πληθυσμός
121	69459
122	72689
124	65013
125	66219
126	66642
127	61059
128	64538
129	71238
130	71463
131	84306
133	58709
134	71363
135	67559
136	57416
138	65731
139	58120
140	72360
141	67427
142	54262
143	53759
144	54678
145	53398

Πίνακας 12: Αποτελέσματα Συνδυαστικής προσέγγισης

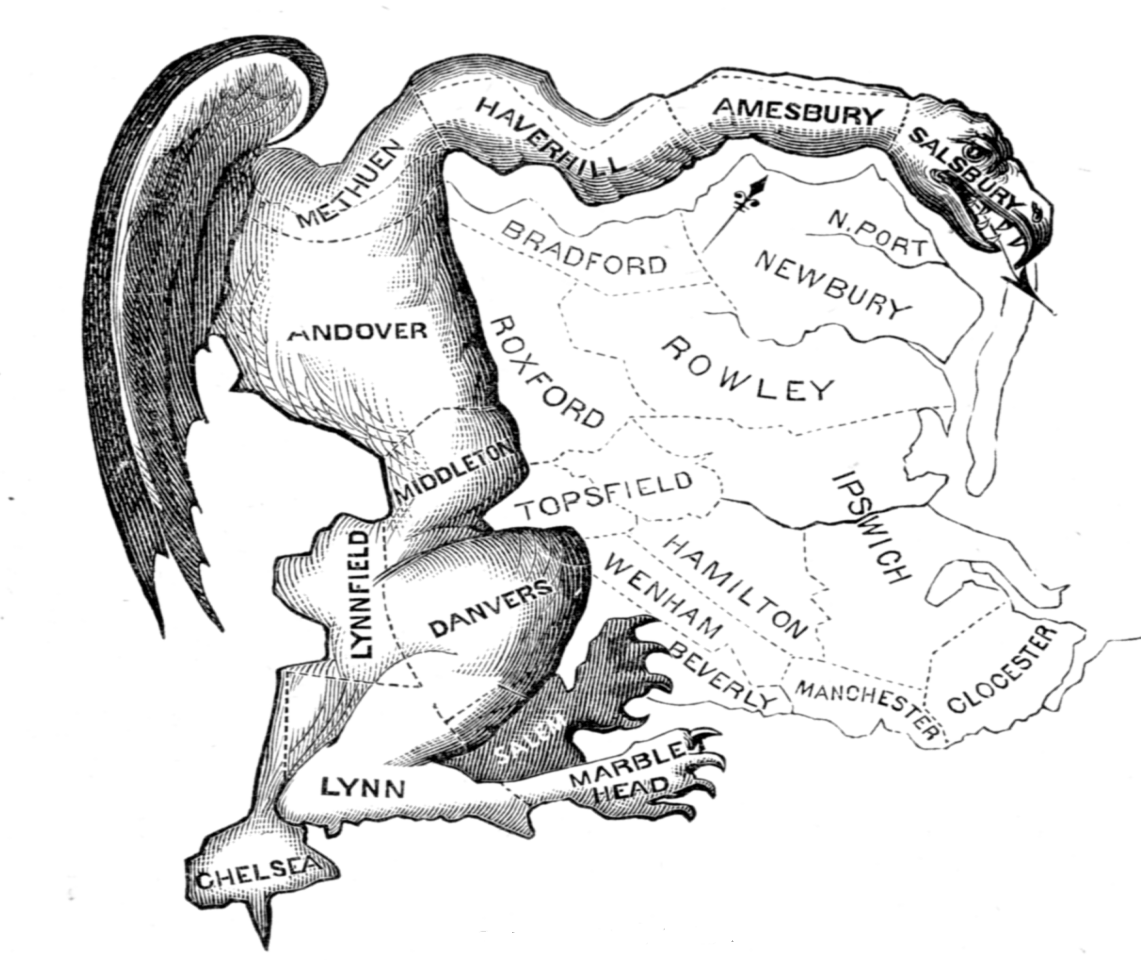
Εντυπώσεις από την συνδυαστική προσέγγιση

Η πρώτη παρατήρηση που πρέπει να γίνει είναι η ποιότητα του αποτελέσματος ως προς τον πληθυσμό. Χαρακτηριστικό είναι ότι η μία περιφέρεια που έχει μεγάλη απόκλιση είναι ένα νησί το οποίο ο σημειακός αλγόριθμος δεν κατάφερε να το ομαδοποιήσει σωστά και στη συνέχεια η δεύτερη προσέγγιση δεν διόρθωσε το λάθος λόγω έλλειψης συνδεσιμότητας. Δυστυχώς όμως ο συνδυασμός των θετικών στοιχείων των δύο μεθόδων προκάλεσε και συνδυασμό των αρνητικών τους πλευρών. Έτσι όπως αναμέναμε οι ασυνέχειες αυξήθηκαν σε σχέση και με τις δύο προηγούμενες προσπάθειες και διατηρήθηκε επίσης το πρόβλημα της σημειακής προσέγγισης που αφορά την αδυναμία της να ορίσει συγκεκριμένο αριθμό περιφερειών. Βέβαια παρατηρώντας το χάρτη και μελετώντας τους πίνακες πληθυσμών είναι αρκετά εύκολο να διορθώσει κανείς αυτές τις ατέλειες με σχετικά μικρό κόπο κάτι που είναι αδύνατο να γίνει σε περίπτωση απόκλισης των πληθυσμών.

Κεφάλαιο 6ο

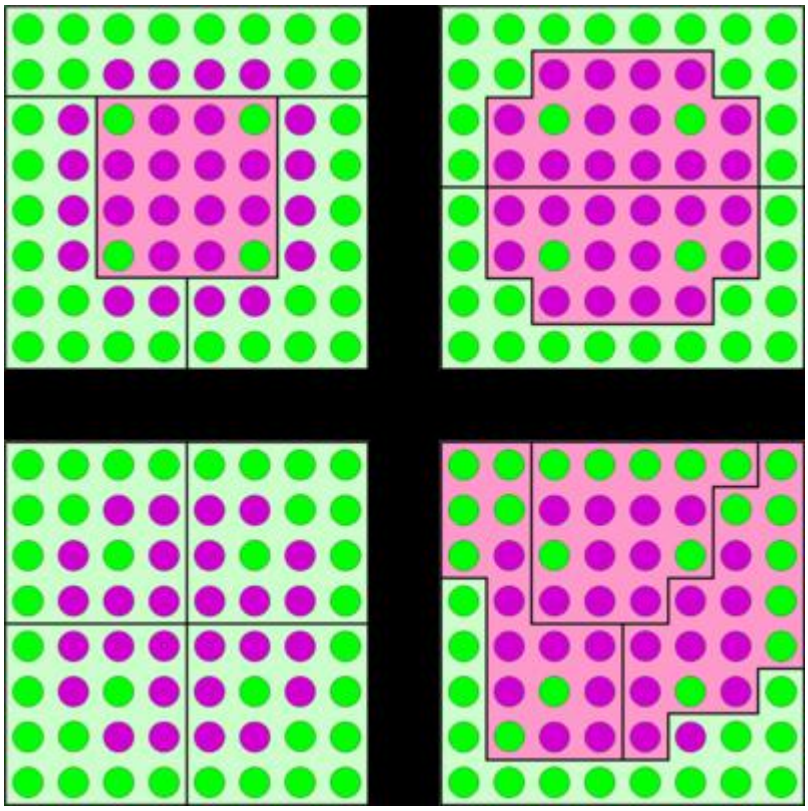
Η Συνοχή των Περιφερειών και το Φαινόμενο “Gerrymandering”

Νωρίτερα στο κεφάλαιο της εφαρμογής έγινε αναφορά στις έννοιες συνοχή και “Gerrymandering”. Λόγω του ενδιαφέροντος που παρουσιάζουν γίνεται μια σύντομη περιγραφή τους στο κεφάλαιο αυτό. Η έννοια του “Gerrymandering” γίνεται εύκολα κατανοητή αν ανατρέξουμε στην ιστορία και ετυμολογία της λέξης. Το 1810 στην πολιτεία της Μασαχουσέτης ο τότε κυβερνήτης Elbridge Gerry αποφάσισε να αναδιοργανώσει τις εκλογικές περιφέρειες με στόχο να ωφεληθεί η Ρεπουμπλικανική του παράταξη. Το νομοσχέδιο αυτό πέρασε το 1812 και σχολιάστηκε αρνητικά από τον τύπο της εποχής. Στο έντυπο Boston Gazette δημοσιεύθηκε η παρακάτω γελοιογραφία που αναπαριστά μία εκλογική περιφέρεια:



Εικόνα 12: Γελοιογραφική αναπαράσταση εκλογικής περιφέρειας στην κομητεία Essex της Μασαχουσέτης

Έτσι το όνομα του Elbridge Gerry συμπτύχθηκε με την λέξη “Salamander” για να γεννηθεί ο όρος “Gerrymander”. Ο τρόπος με τον οποίο μπορεί να επηρεάσει το φαινόμενο αυτό τα εκλογικά αποτελέσματα εξαρτάται άμεσα από το εκλογικό σύστημα της χώρας. Τα συστήματα τα οποία μπορούν να επηρεαστούν πιο πολύ από το “Gerrymandering” είναι τα μη αναλογικά. Τα παρακάτω σχεδιαγράμματα δίνουν ένα παράδειγμα σε ένα σύστημα λίγων μονοεδρικών περιφερειών.



Εικόνα 13: Παράδειγμα "Gerrymandering"

Ο κάθε κύκλος αναπαριστά έναν ψηφοφόρο ο οποίος ψηφίζει μία από τις δύο παρατάξεις. Σε κάθε εικόνα δηλαδή υπάρχουν *64* ψήφοι οι οποίες κατανέμονται με πανομοιότυπο τρόπο. Αυτό που αλλάζει σε κάθε περίπτωση είναι ο χωρισμός των περιφερειών. Αν για παράδειγμα το σύστημα είναι το λεγόμενο *“first pass the post”* (όποιος έχει μεγαλύτερο ποσοστό κερδίζει την περιφέρεια) τότε στην πρώτη εικόνα το πράσινο κόμμα κερδίζει τρεις έδρες, στην δεξιά εικόνα κερδίζει δύο έδρες, στην κάτω αριστερά τέσσερις, και στην τελευταία μόνο μία. Έτσι λοιπόν ενώ η μοβ παράταξη έχει και στις τέσσερις περιπτώσεις σαφώς λιγότερες ψήφους (*28 στις 64)*, με αλλαγή στην περιφερειοποίηση μπορεί να λάβει ακόμα και τρεις στις τέσσερις έδρες. Ο πολιτικός κίνδυνος είναι μεγάλος και η ανάγκη για αντικειμενική περιφερειοποίηση υπαρκτή.

Το σχήμα μιας περιφέρειας επηρεάζει επίσης και την λειτουργικότητά της. Στην παρούσα εργασία που μελετώνται εκλογικές περιφέρειες είναι λογικό ο ψηφοφόρος να διανύει όσο το δυνατόν λιγότερη απόσταση προκειμένου να ασκήσει το δικαίωμά του. Ένας δεύτερος λόγος ίσως

και ουσιαστικότερος είναι το ποσοστό κοινωνικής ομοιογένειας εντός της περιφέρειας το οποίο μειώνεται όσο αυξάνονται οι εσωτερικές αποστάσεις της περιφέρειας. Προκειμένου να αντιμετωπιστούν τα παραπάνω προβλήματα είναι δυνατόν να τεθούν κατά την περιφερειοποίηση και ορισμένα κριτήρια γεωμετρικά ώστε να αποφευχθεί η παραγωγή παράξενων και δυσλειτουργικών σχημάτων. Προκειμένου να γίνει κάτι τέτοιο πρέπει πρώτα να ποσοτικοποιήσουμε την ποιότητα του σχήματος των περιφερειών σε ένα μέγεθος. Η ποσότητα αυτή ονομάζεται "πυκνότητα" (Compactness). Για την μέτρηση της πυκνότητας έχουν προταθεί πάνω από τριάντα κριτήρια *(Altman 1998, Niemi 1991, Young 1988)*. Τα περισσότερα από αυτά ανήκουν σε τρεις κατηγορίες: εμβαδού, περιμέτρου, πληθυσμού. Μερικά παραδείγματα κριτηρίων πυκνότητας παραθέτονται παρακάτω:

- $\frac{W}{L}$ Όπου L η μεγαλύτερη διάμετρος και W η μέγιστη κάθετη στην L διάμετρος (Harris 1964)
- $\frac{W}{L}$ Όπου L,W οι πλευρές του περιγεγραμμένου ορθογωνίου με ελάχιστη περίμετρο (Niemi 1991)
- $W - L$ Όπου W και L οι μεγαλύτεροι κατακόρυφοι και οριζόντιοι άξονες αντίστοιχα (Eig and Seitzinger 1981)
- Διάμετρος του εγγεγραμμένου κύκλου / Διάμετρος περιγεγραμμένου κύκλου (Flaherty and Crumplin 1992)
- Το εμβαδόν του εγγεγραμμένου κύκλου / Το εμβαδόν της περιφέρειας (Ehrenburg 1892, Frolov 1974)
- Ο λόγος του εμβαδού της περιφέρειας προς το εμβαδόν του κύκλου με ίση περίμετρο (Gibbs 1961)

Τα ερωτήματα βέβαια που τίθενται τώρα είναι αρκετά. Τι ακριβώς μετράνε τα κριτήρια πυκνότητας και κατά πόσο καταπολεμούν το "Gerrymandering"; Ποιο από την πληθώρα που παρέχεται πρέπει να επιλέξουμε και τι όριο πρέπει να θέσουμε σε αυτό; Η κάθε ερώτηση από αυτές επιδέχεται μεγάλη ανάλυση και επειδή είναι εκτός των στόχων αυτής της εργασίας δεν θα γίνει εμβάθυνση. Ωστόσο είναι ένα θέμα το οποίο ο μελετητής οφείλει να το λάβει υπόψη του.

Κατά την εφαρμογή των διαφόρων αλγορίθμων στους δήμους του Καποδίστρια έγινε απόπειρα να εισαχθούν και κάποια κριτήρια πυκνότητας. Δυστυχώς όμως τα αποτελέσματα ήταν μη αποδεκτά καθώς ο αλγόριθμος λόγω των επιπλέον περιορισμών εγκλωβιζόταν σε λύσεις πολύ υποδεέστερες από αυτές που παρουσιάστηκαν. Μιας και τα οφέλη από αυτό το εγχείρημα ήταν

ελάχιστα σε σχέση με τις αρνητικές επιδράσεις, δεν συμπεριλήφθηκε στις εφαρμογές που αναπτύχθηκαν στα προηγούμενα κεφάλαια. Ο λόγος που ο αλγόριθμος επηρεάστηκε αρνητικά από την εισαγωγή κριτηρίων πυκνότητας είναι ότι όπως έχει αναφερθεί και στο κεφάλαιο 5 της εφαρμογής, όσο αυξάνονται οι περιορισμοί τόσο μειώνεται η ευελιξία του και στην περίπτωση της εφαρμογής στον Καποδίστρια ήδη τα προβλήματα που συμβάλλουν αρνητικά είναι πολλά.

Κεφάλαιο 7ο

Συμπεράσματα

7.1 Συνοπτική Παρουσίαση

Ορισμένες φορές παρουσιάζεται η ανάγκη να γίνει ομαδοποίηση κάποιων χωρικών μονάδων σε μεγαλύτερα σύνολα. Η διαδικασία αυτή ονομάζεται περιφερειοποίηση και προκειμένου να υλοποιηθεί μπορούν να ακολουθηθούν διάφορες οδοί. Σε κάποιες περιπτώσεις, συνήθως όταν πρόκειται για μικρό αριθμό δεδομένων, η εμπειρία του μελετητή είναι αρκετή για να φέρει σε πέρας αυτή τη διαδικασία, σε άλλες όμως όπου οι παράμετροι είναι πολλές και ο όγκος των δεδομένων ξεπερνάει τις ανθρώπινες δυνατότητες επεξεργασίας απαιτείται κάτι πιο αποτελεσματικό. Με την πρόοδο της τεχνολογίας των υπολογιστών οι χωρικοί αναλυτές εξοπλίστηκαν με υπολογιστές ικανούς να λύσουν αυτό το πρόβλημα. Συγκεκριμένα, τα ψηφιακά εργαλεία που αναπτύχθηκαν ονομάζονται "Γεωγραφικά Συστήματα Πληροφοριών" (GIS) και αναλαμβάνουν με τη βοήθεια του χρήστη να ολοκληρώσουν τέτοιου είδους πολύπλοκες εργασίες. Επανερχόμενοι στο αρχικό θέμα, η περιφερειοποίηση με τη βοήθεια των GIS μπορεί να γίνει σαφώς πιο εύκολα και με περισσότερη ακρίβεια.

Πριν όμως αναζητήσουμε τον τρόπο ανάπτυξης της εφαρμογής θα πρέπει να προσεγγίσουμε το πρόβλημα της περιφερειοποίησης θεωρητικά καθώς υπάρχουν αρκετοί διαφορετικοί αλγόριθμοι. Η επιλογή ενός από τους αλγορίθμους περιφερειοποίησης που αναφέρονται στη διεθνή βιβλιογραφία (*Openshaw, Altman, Alvanidis, Macmillan)* δεν είναι εύκολη καθώς ο καθένας έχει διαφορετικά χαρακτηριστικά. Μετά από μελέτη των θετικών και αρνητικών του καθενός επιλέχτηκαν δύο από αυτούς, οι οποίοι είναι ο AZP (Openshaw 1977) και ο "Point approach algorithm" (Fernando Bacao, Marco Painho 2002). Ο πρώτος ξεκινάει από μία ήδη υπάρχουσα τυχαία περιφερειοποίηση και προσπαθεί με διαδοχικές αλλαγές να καταλήξει σε βέλτιστο αποτέλεσμα. Ο δεύτερος είναι σημειακός αλγόριθμος το οποίο σημαίνει ότι αντιλαμβάνεται τα πολύγονα σαν σημεία. Η λογική του είναι ότι παράγει τυχαία "κέντρα" περιφερειών τα οποία προσελκύουν τα κοντινά σημεία-πολύγωνα. Στη συνέχεια μελετάται μέσω μιας συνάρτησης η ποιότητα του αποτελέσματος και η διαδικασία επαναλαμβάνεται μέχρι να βρεθεί μία αποδεκτή λύση. Κοινό χαρακτηριστικό των αλγορίθμων περιφερειοποίησης είναι η αντικειμενική συνάρτηση, η οποία έχει τον ρόλο να ποσοτικοποιεί την ποιότητα του αποτελέσματος. Σε κάθε περίπτωση η συνάρτηση αυτή πρέπει να βελτιστοποιηθεί προκειμένου η λύση να προσεγγίζει την ιδανική. Η συνάρτηση αυτή μπορεί να περιγράφει διάφορα μεγέθη ανάλογα με τους στόχους της μελέτης. Για

παράδειγμα μπορεί να αφορά τον πληθυσμό της κάθε περιφέρειας, κάποια γεωγραφικά χαρακτηριστικά, χρήσεις γης, κ.α.

Προκειμένου να μελετηθούν οι παραπάνω αλγόριθμοι έγινε προσπάθεια υλοποίησής τους με βάση τους δήμους του Καποδίστρια και με στόχο την ομαδοποίησή τους σε εκλογικές περιφέρειες. Το εκλογικό πρότυπο που επιλέχτηκε είναι το γερμανικό σύμφωνα με το οποίο η χώρα χωρίζεται σε μονοεδρικές περιφέρειες από τις οποίες εκλέγεται το μισό βουλευτικό σώμα. Το υπόλοιπο μισό εκλέγεται μέσω δεύτερης ψηφοφορίας από λίστες των παρατάξεων. Εφαρμόζοντας το σύστημα αυτό στην Ελλάδα πρέπει να χωρίσουμε την χώρα σε 150 περιφέρειες. Το κριτήριο που τέθηκε σαν αντικειμενική συνάρτηση είναι ο πληθυσμός. Ιδανική λύση είναι αυτή στην οποία όλες οι περιφέρειες έχουν πληθυσμό τον πληθυσμό τής χώρας διαιρεμένο με τον αριθμό των περιφερειών. Η όλη εφαρμογή υλοποιήθηκε σε γλώσσα προγραμματισμού PHP και τα δεδομένα εξήχθησαν από το λογισμικό ArcGIS.

Ο αλγόριθμος AZP για να εφαρμοστεί απαιτεί μια αρχική τυχαία περιφερειοποίηση. Το πρώτο βήμα λοιπόν αφού εισαχθούν τα δεδομένα είναι να γίνει αυτό. Εδώ όμως πρέπει να επισημάνουμε ότι τόσο η αρχική περιφερειοποίηση όσο και η τελική οφείλουν να υπακούν στους κανόνες της συνέχειας που σημαίνει ότι οι περιφέρειες πρέπει να είναι συνεχείς και να μην διακόπτονται από τις γειτονικές. Αυτό κατά την τυχαία περιφερειοποίηση επιτυγχάνεται προσαρτώντας στην κάθε περιφέρεια μόνο τα γειτονικά πολύγωνα και κατά την πορεία του κύριου αλγορίθμου με την μέθοδο των σημείων εναλλαγής *(Macmillan 2000)* που περιγράφεται αναλυτικά στο θεωρητικό κομμάτι της εργασίας. Τα μειονεκτήματα του αλγορίθμου είναι η αδυναμία του να διαχειριστεί τα πολύγωνα που δεν εφάπτονται με κανένα γειτονικό όπως τα νησιά και η εξάρτησή του από την τυχαία αρχική κατάσταση καθώς ορισμένες φορές εγκλωβίζεται σε αποτελέσματα που απέχουν από τη βέλτιστη λύση. Στα πλεονεκτήματα πρέπει να αναφέρουμε την καλή ικανότητα του αλγορίθμου να διατηρεί τις περιφέρειες συνεχείς.

Ο σημειακός αλγόριθμος είναι αρκετά πιο απλός στην ανάπτυξη καθώς το μόνο που απαιτεί είναι η παραγωγή τόσων τυχαίων σημείων όσες και οι περιφέρειες και η σύνδεση αυτών των σημείων με τα κεντροϊδή των κοντινότερων δήμων. Η αποδοχή ότι ο κάθε δήμος είναι ένα σημείο έχει το καλό ότι οι ιδιομορφίες της τοπολογίας δεν επηρεάζουν τον αλγόριθμο όπως στην προηγούμενη εφαρμογή. Το αρνητικό όμως είναι ότι συχνά παράγονται ασυνέχειες οι οποίες ενίοτε καθιστούν το αποτέλεσμα μη αποδεκτό. Επίσης ο αλγόριθμος αυτός παράγει συχνά περιφέρειες με πληθυσμό που αποκλίνει πολύ από τον επιθυμητό.

Μελετώντας τα θετικά και αρνητικά των δύο παραπάνω αλγορίθμων γεννήθηκε η ιδέα να

προσεγγιστεί το θέμα με μία συνδυαστική μέθοδο που να επωφελείται από τα δυνατά στοιχεία της κάθε εφαρμογής. Το σκεπτικό είναι να χρησιμοποιήσουμε την ικανότητα του αλγορίθμου του Openshaw να διαχειρίζεται μικρές περιφέρειες και ταυτόχρονα με τη βοήθεια του σημειακού αλγορίθμου να ρυθμίσουμε τους δήμους που παρουσιάζουν προβλήματα λόγω της τοπολογίας τους. Στην πράξη έγινε πρώτα μία προσωρινή περιφερειοποίηση με τον σημειακό αλγόριθμο και στη συνέχεια διορθώθηκε με τη βοήθεια του αλγορίθμου του Openshaw. Εδώ πρέπει να σημειωθεί ότι και στους δύο επιμέρους αλγορίθμους χρησιμοποιήθηκαν οι βελτιωμένες τους εκδόσεις. Στο μεν αλγόριθμο του Openshaw τέθηκε και η παράμετρος του "Simulation Annealing" ενώ ο σημειακός αλγόριθμος εφαρμόστηκε σε κάνναβο. Οι παραλλαγές αυτές των αλγορίθμων περιγράφονται αναλυτικά στο θεωρητικό κομμάτι της εργασίας. Τα αποτελέσματα αυτής της προσπάθειας ήταν αρκετά καλά καθώς οι πληθυσμοί συνέκλιναν σε μεγάλο βαθμό στον επιθυμητό μέσο όρο. Το αρνητικό ήταν ότι σε ορισμένες περιπτώσεις αυξήθηκαν οι ασυνέχειες στις περιφέρειες.

7.2 Αποτελέσματα-Συμπεράσματα

Μελετώντας τα αποτελέσματα των εφαρμογών καθώς και τη θεωρία πίσω από κάθε μία, παρατηρούμε μία ευαισθησία των αλγορίθμων στη μορφή των δεδομένων που δέχονται ως είσοδο. Φαίνεται ότι ανάλογα με την περιοχή που πρέπει να περιφερειοποιηθεί η επιλογή του αλγορίθμου είναι διαφορετική. Επίσης όπως συνέβη στην παρούσα εργασία, ορισμένες φορές απαιτείται τροποποίηση των αλγορίθμων προκειμένου να προσαρμοστούν καλύτερα στα δεδομένα της περιοχής μελέτης.

Δυστυχώς παρόλο που στην παγκόσμια βιβλιογραφία το θεωρητικό κομμάτι της περιφερειοποίησης καλύπτεται διεξοδικά, πρακτικές εφαρμογές υπάρχουν πολύ λίγες. Αξιοσημείωτη είναι η προσπάθεια του εργαστηρίου Cogit που στεγάζεται στο IGN *(Institut Géographique National)* στο Παρίσι. Το εργαστήριο αυτό έχει αναπτύξει σε ανοιχτό κώδικα *Java* μία πλατφόρμα η οποία συγκεντρώνει πληθώρα βοηθημάτων για τη διαχείριση χωρικών δεδομένων (γεωμετρία, τοπολογία, μεταδεδομένα, γεωγραφικά χαρακτηριστικά κ.α). Ορισμένα από αυτά θα ήταν ιδιαίτερα χρήσιμα σε εργασίες όπως αυτή της περιφερειοποίησης.

Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο η περιφερειοποίηση ειδικά στην περίπτωση των εκλογών εγκυμονεί πολλούς κινδύνους για την αντικειμενικότητα όλου του θεσμού και ως εκ τούτου και της δημοκρατίας. Συνεπώς ακόμα και και με κάποια μειονεκτήματα η βάση της μελέτης για την περιφερειοποίηση θα πρέπει να είναι αυτοματοποιημένη με όσο το δυνατόν λιγότερη ανθρώπινη παρέμβαση. Στην εφαρμογή που παρουσιάστηκε σε αυτή την εργασία ο ανθρώπινος παράγοντας έχει μειωθεί στο ελάχιστο καθώς στην συνδυαστική προσέγγιση το μόνο που χρειάζεται είναι ορισμένες διορθώσεις στο τέλος ώστε να γίνουν συνεχείς οι περιφέρειες. Οι

αλλαγές αυτές λόγω του μικρού τους αριθμού έχουν αμελητέα επίδραση στο συνολικό αποτέλεσμα.

7.3 Προτάσεις για το μέλλον

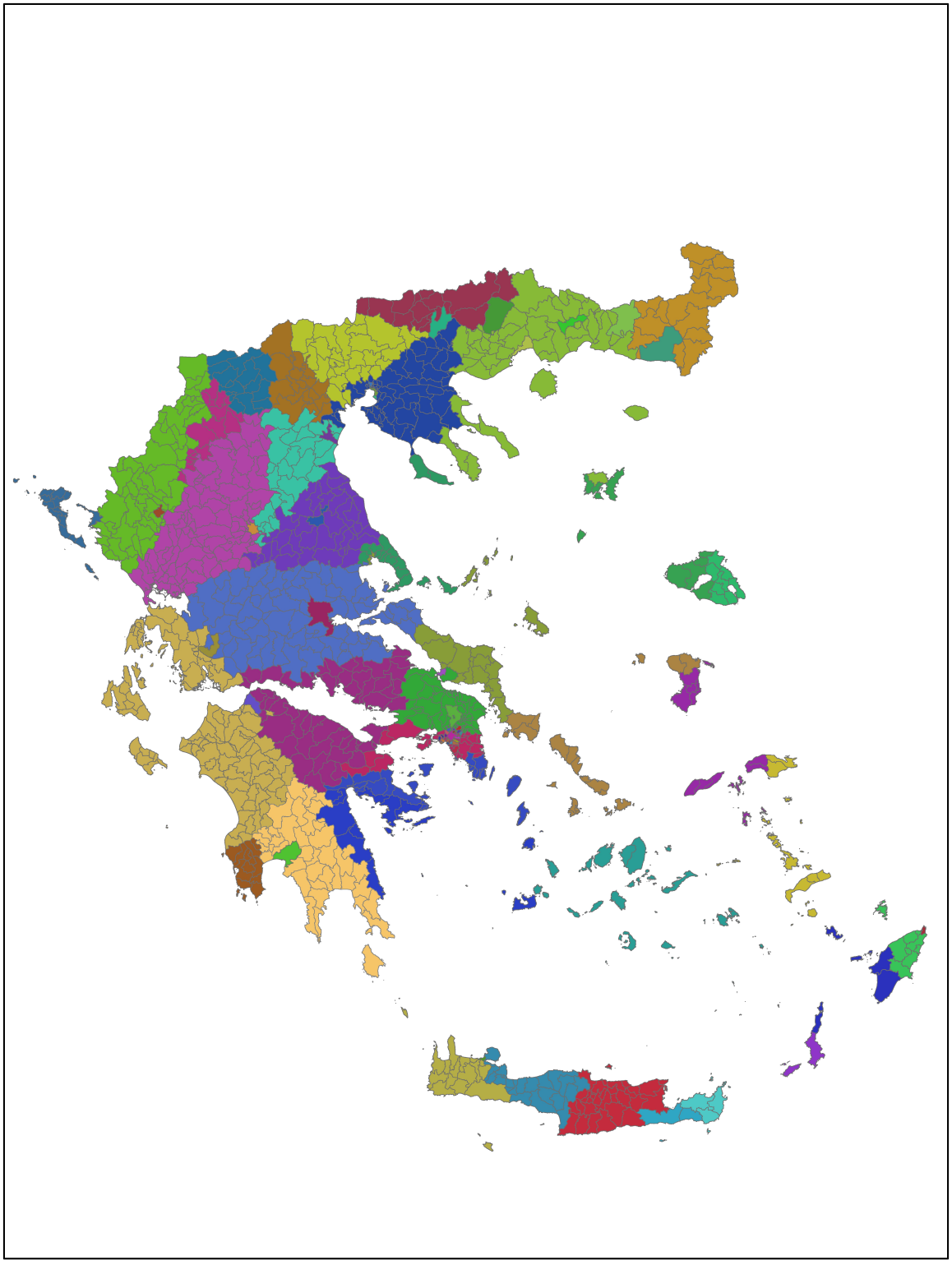
Κατά την εφαρμογή παρατηρήθηκε δυσκολία στη διαχείριση των δήμων του Καποδίστρια λόγω της μεγάλης διακύμανσης των πληθυσμών. Υπάρχουν δήμοι με πληθυσμό κάτω από *1000* και άλλοι με πάνω από *500.000* . Προκειμένου να γίνουν οι αλγόριθμοι πιο ευέλικτοι ώστε να μπορούν να δεχτούν περισσότερες παραμέτρους θα πρέπει να αλλάξει η χωρική μονάδα. Κάτι τέτοιο βέβαια δεν είναι εφικτό άμεσα αλλά σε βάθος χρόνου θα έλυνε αρκετά προβλήματα ομοιογένειας, καθώς μια μονάδα είναι πιο λειτουργική όταν απαρτίζεται από άτομα με κοινά χαρακτηριστικά και κάτι τέτοιο είναι αδιανόητο για δήμους με υπέρογκο πληθυσμό. Αντίστοιχα οι πολύ μικροί δήμοι είναι επίσης προβληματικοί καθώς είναι ασύμφορο να υπάρχουν υπηρεσίες για τόσο μικρούς πληθυσμούς. Ξεπερνώντας αυτό το πρόβλημα θα μπορούσαμε να εισάγουμε στην αντικειμενική συνάρτηση των αλγορίθμων και παραμέτρους κοινωνικού και οικονομικού χαρακτήρα για να επιτύχουμε και ομοιογένεια στις εκλογικές περιφέρειες ώστε άτομα με κοινά συμφέροντα να έρχονται πιο κοντά. Άλλες παράμετροι που μπορούν να εισαχθούν είναι εκείνες της πυκνότητας. Εισάγοντάς τες θα βελτιωθεί το σχήμα της κάθε περιφέρειας μειώνοντας τις μέγιστες αποστάσεις εντός της περιφέρειας. Έτσι αν για παράδειγμα πρόκειται για διοικητικές περιφέρειες ο πολίτης θα διανύει την ελάχιστη απόσταση για να επισκέπτεται τις δημόσιες υπηρεσίες.

Βιβλιογραφία

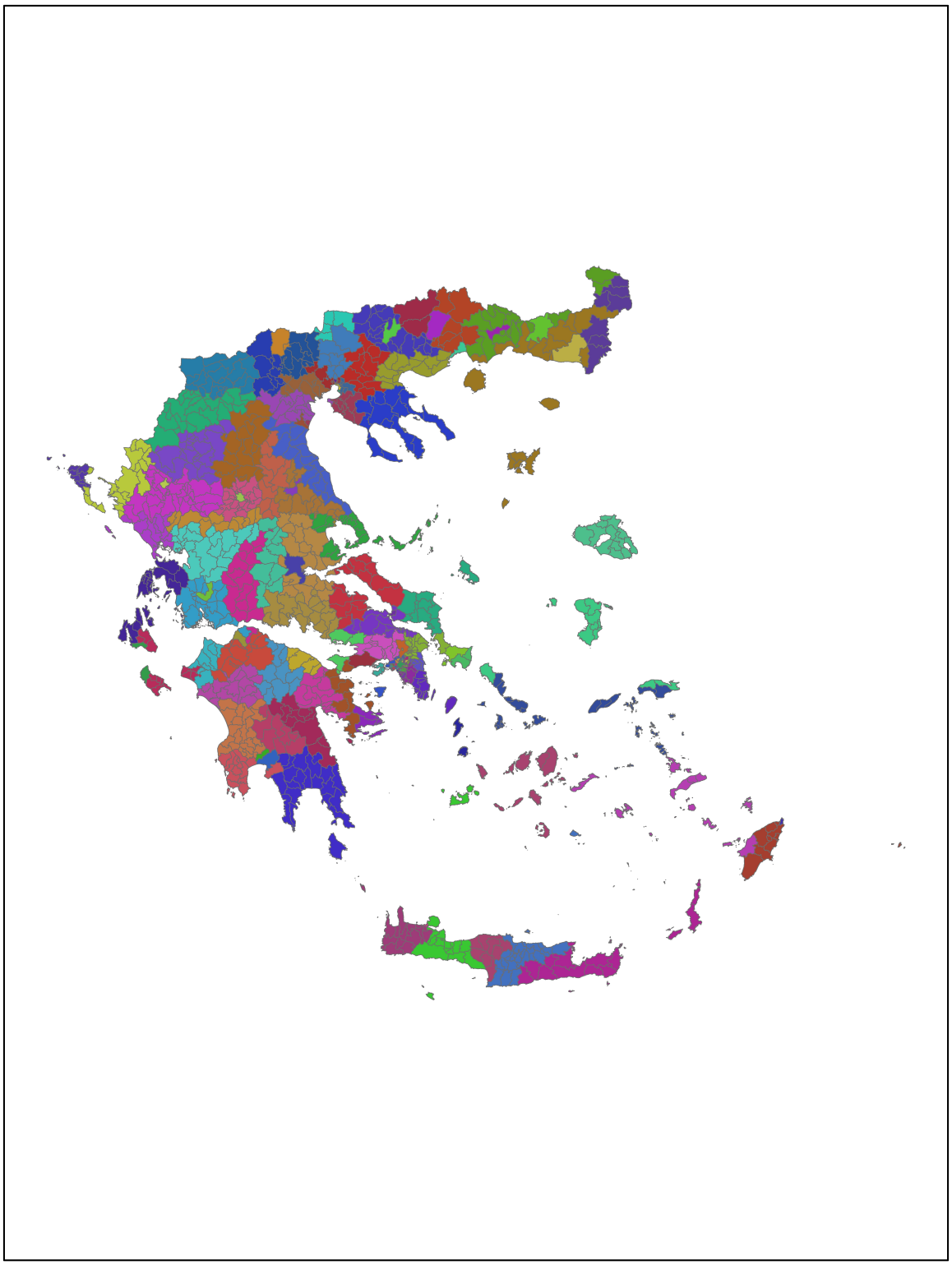
1. Κ. Κουτσόπουλος, 2002. ***Γεωγραφικά Συστήματα Πληροφοριών και Ανάλυση Χώρου.*** Εθνικό Μετσόβιο Πολυτεχνείο. Εκδόσεις Παπασωτηρίου
2. Micah Altman 1998. ***Modeling the effect of mandatory district compactness on partisan gerrymanders.*** Department of Government, Littauer M-39, Harvard University, Cambridge
3. Daniel D. Polsby, Robert D. Popper. ***Partisan Gerrymandering: Harms and a new Solution*** A Heartland Policy Study No.34, 1991
4. Fernando Bação, Marco Painho (2002). ***A Point approach to Zone Design***. Instituto Superior de Estatística e Gestão de Informação – Universidade Nova de Lisboa
5. W. Macmillian, Journal of Geographical Systems, 2001. ***Redistricting in a GIS Environment: An Optimisation Algorithm Using Switching-Points***
6. Openshaw S.Rao L (1995) ***Algorithms for reengineering 1991 Census geography***. Environment and Planning A 27:425-446
7. Stan Openshaw, Seraphim Alvanides and Simon Whalley. ***Some further experiments with designing output areas for the 2001 UK census.*** Centre for Computational Geography School of Geography University of Leeds
8. http://en.wikipedia.org/wiki/Simulated_annealing
9. http://oxygene-project.sourceforge.net/
10. http://en.wikipedia.org/wiki/German_elections
11. http://www.php.net

Παράρτημα Α: Χάρτες

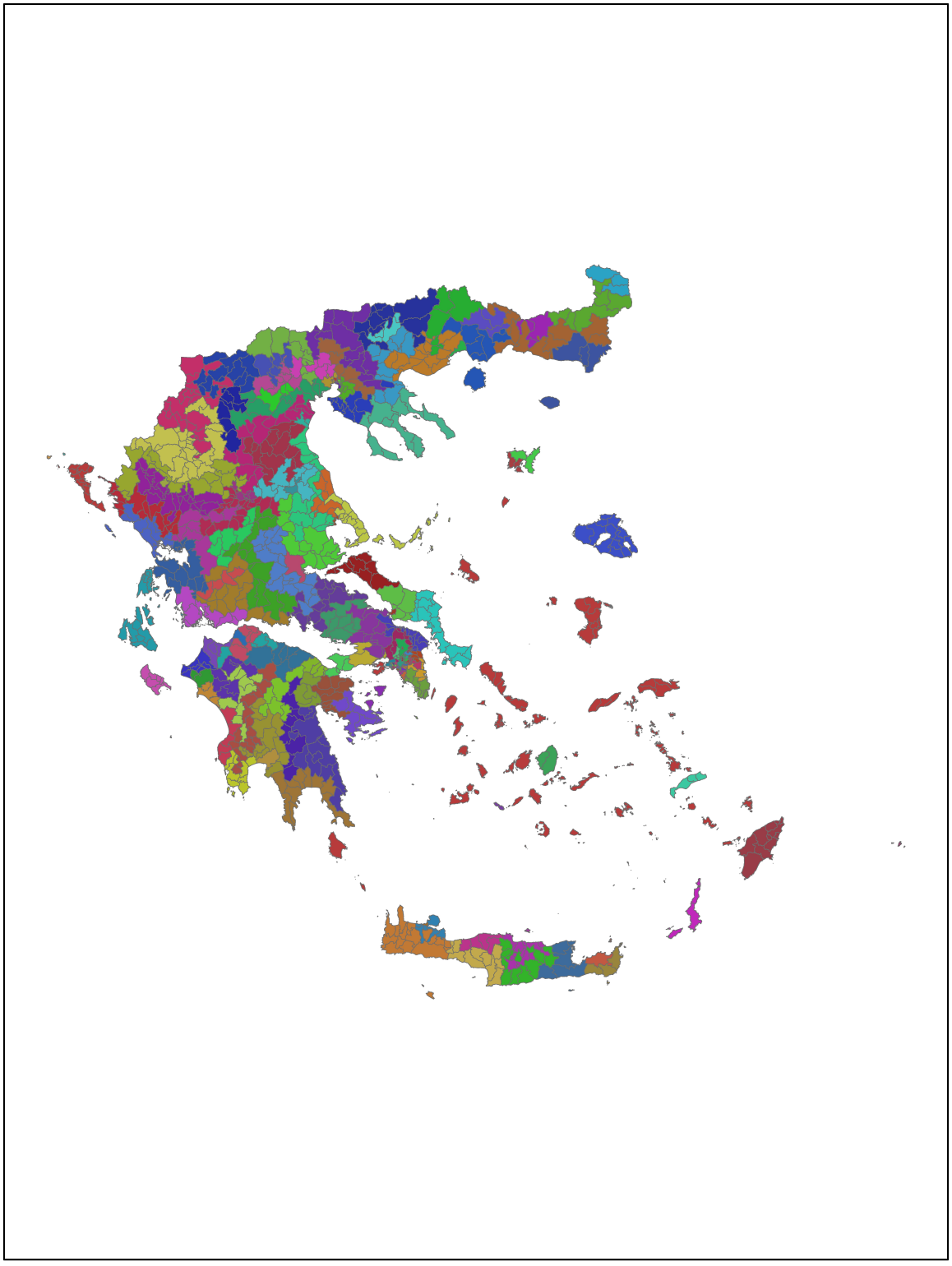
Χάρτης 1. Απλή σημειακή προσέγγιση



Χάρτης 2. Σημειακή προσέγγιση με χρήση καννάβου



Χάρτης 3. Ο αλγόριθμος του Openshaw



Χάρτης 4. Η συνδυαστική προσέγγιση

Παράρτημα Β: Κώδικας

Κώδικας για την προετοιμασία

```
<?php
$lines = file('polyexp2.csv');
$i=0;

foreach ($lines as $line) {
        $pieces=explode("%",$line);
        $i++;
        $kapo[$i]['id']=$pieces[0];
        $kapo[$i]['po']=$pieces[1];
        $kapo[$i]['git']=array();
        $pieces[4] = preg_replace("/[\n\r]/","",$pieces[4]);
        $kapo[$i]['names']=array($pieces[2],$pieces[3],$pieces[4]);
        $names[$i]['names']=array($pieces[2],$pieces[3],$pieces[4]);
        $names[$i]['set']=True;
        $kapo[$i]['set']=True;
}
$newnames=array();
$counter=0;

$lines=file("ids2.csv");
foreach ($lines as $one){
$pieces=explode("#",$one);
$pieces[3] = preg_replace("/[\n\r]/","",$pieces[3]);
 for($i=1;$i<=count($names);$i++){
 if($names[$i]['names'][0]==$pieces[1] and $names[$i]['names'][1]==$pieces[2] and
$names[$i]['names'][2]==$pieces[3]){
 $names[$i]['fid']=$pieces[0];
 }
 }
}

$names[1]['fid']=9999999;
$names=remove_dups($names,'fid');


echo count($names);
for($i=1;$i<=count($names);$i++){
$names[$i]['polys']=array();
$names[$i]['id']=$i;
$names[$i]['git']=array();
$names[$i]['sea']=0;
$names[$i]['perif']=0;
        for($j=1;$j<=count($kapo);$j++){
                if($names[$i]['names']==$kapo[$j]['names']){
    $names[$i]['pop']=$kapo[$j]['po'];
                array_push($names[$i]['polys'],$kapo[$j]['id']);
                }
```

```
        }
}

$lines = file('LeftRight.csv');
foreach ($lines as $line) {
        $pieces=explode(" ",$line);
  $pieces[4] = preg_replace("/[\n\r]/","",$pieces[4]);
  $ena=$pieces[4];
  $dio=$pieces[3];
          for($i=1;$i<=count($names);$i++){
        if(in_array($ena,$names[$i]['polys'])){
        $to=$i;
        }

    }
    for($i=1;$i<=count($names);$i++){
        if(in_array($dio,$names[$i]['polys'])){
        $me=$i;
        }

    }
  if(in_array($me,$names[$to]['git'])==False) array_push($names[$to]['git'],$me);
  if(in_array($to,$names[$me]['git'])==False) array_push($names[$me]['git'],$to);

}


$flag=0;
foreach($names as $one){
  $flag++;
  $sum=0;
  $inner_flag=0;
  foreach($one['git'] as $current){
    $inner_flag++;
    $sum=$sum+$current;
  }
  $ave=$sum/$inner_flag;
  if($ave==1)$names[$flag]['sea']=1;
}

$handle = fopen("out.txt", "w");
 foreach($names as $one){
 fwrite($handle,$one['id']);
 fwrite($handle,'#');
 fwrite($handle,$one['sea']);
 fwrite($handle,'#');
 fwrite($handle,$one['pop']);
 fwrite($handle,'#');
  foreach($one['names'] as $name){
    fwrite($handle,$name);
```

```
   fwrite($handle,'&');
  }
  fwrite($handle,'#');
  foreach($one['git'] as $git){
    fwrite($handle,$git);
    fwrite($handle,'&');
  }
  fwrite($handle,'#');
  foreach($one['polys'] as $poly){
    fwrite($handle,$poly);
    fwrite($handle,'&');
  }
 fwrite($handle,"\n");
  }


fclose($handle);
 exit;

function remove_dups($array, $row_element) {
    $new_array[1] = $array[1];
    foreach ($array as $current) {
        $add_flag = 1;
        foreach ($new_array as $tmp) {
            if ($current[$row_element]==$tmp[$row_element]) {
                $add_flag = 0; break;
            }
        }
        if ($add_flag) $new_array[] = $current;
    }
    return $new_array;
}

 ?>
```

Η σημειακή προσέγγιση

```
<?php
 $lines = file('meta_centr.txt');
$i=0;
$pop=0;
$perif=150;
$kapodistrias=array();
foreach ($lines as $line) {
  $i++;
  $pieces=explode("#",$line);
  $kapodistrias[$i]['id']=$pieces[0];
  $kapodistrias[$i]['fid']=$pieces[1];
  $kapodistrias[$i]['sea']=$pieces[2];
  $kapodistrias[$i]['pop']=$pieces[3];
  $kapodistrias[$i]['names']=explode("&",$pieces[4]);
  array_pop($kapodistrias[$i]['names']);
  $kapodistrias[$i]['polys']=explode("&",$pieces[6]);
  array_pop($kapodistrias[$i]['polys']);
  $kapodistrias[$i]['git']=explode("&",$pieces[5]);
  array_pop($kapodistrias[$i]['git']);
  $kapodistrias[$i]['perif']=0;
  $kapodistrias[$i]['x']=$pieces[8];
  $pieces[9] = preg_replace("/[\n\r]/","",$pieces[9]);
  $pieces[9] = preg_replace("/,/",".",$pieces[9]);
  settype($pieces[9],"double");
  $kapodistrias[$i]['y']=$pieces[9];
  $pop=$pop+$pieces[3];
}
$ave=$pop/$perif;

$badlist1=array();
$flag=0;
$bigpop=0;

for($i=1;$i<count($kapodistrias);$i++){
  if($kapodistrias[$i]['pop']>50000){
  $flag++;
  $bigpop=$bigpop+$kapodistrias[$i]['pop'];
  $kapodistrias[$i]['perif']=$flag;
  array_push($badlist1,$kapodistrias[$i]['id']);
  }

}

$perif=150-variant_fix($bigpop/$ave);

$start=$flag+1;
```

```
$xmin=9999999999999999;
$ymin=9999999999999999;
$xmax=0;
$ymax=0;

foreach ($kapodistrias as $one){
  if($one['id']==1)continue;
  if($one['x']>$xmax)$xmax=$one['x'];
  if($one['x']<$xmin)$xmin=$one['x'];
  if($one['y']>$ymax)$ymax=$one['y'];
  if($one['y']<$ymin)$ymin=$one['y'];
}
settype($ymin,'integer');
settype($ymax,'integer');
$ymin=variant_fix($ymin);
$ymax=variant_fix($ymax);
settype($ymin,'integer');
settype($ymax,'integer');

$lastop=999999999999999999;

$backup=$kapodistrias;
$flag=2000;
while($flag>0){
  $kapodistrias=$backup;
if($flag/100-variant_fix($flag/100)==0)echo $flag."\n";

for($i=$start;$i<=$perif;$i++){
  $x=mt_rand($xmin,$xmax);
  $y=mt_rand($ymin,$ymax);
  $n[$i]['x']=$x;
  $n[$i]['y']=$y;
}

for($i=2;$i<=count($kapodistrias);$i++){
  $id=$kapodistrias[$i]['id'];
  if(in_array($id,$badlist1))continue 1;
  $min=99999999999999999999;
  for($j=$start;$j<=$perif;$j++){
    $dx=$kapodistrias[$i]['x']-$n[$j]['x'];
    $dy=$kapodistrias[$i]['y']-$n[$j]['y'];
  $iporizo=pow($dx,2)+pow($dy,2);
  $dist=pow($iporizo,0.5);
  if($dist<$min){
    $min=$dist;
    $kapodistrias[$id]['perif']=$j;
  }
 }
```

```
}
$flag--;
$good=0;
$op=optimize($kapodistrias,$badlist1);

$periflist=array();
$poplist=array();
foreach ($kapodistrias as $one){
array_push($periflist,$one['perif']);
}
$periflist=array_unique($periflist);
foreach ($periflist as $one){
  $temppop=0;
  foreach ($kapodistrias as $two){
    if($one==$two['perif']){
      $temppop=$temppop+$two['pop'];
    }
  }
  array_push($poplist,$temppop);
}
asort($poplist);
array_shift($poplist);
$last=count($poplist)-1;
if($poplist[0]>20000 and $poplist[$last]<200000)$good=1;

 if($op<$lastop){
 $lastop=$op;
 $best=$kapodistrias;
 echo $op;
 echo "\n";
 }
}
$periflist=array();
foreach ($kapodistrias as $one){
array_push($periflist,$one['perif']);
}
$periflist=array_unique($periflist);
foreach ($periflist as $one){
  $temppop=0;
  foreach ($kapodistrias as $two){
    if($one==$two['perif']){
      $temppop=$temppop+$two['pop'];
    }
  }

  if(in_array($one,$badlist1)){
    }else{
  }
}
```

```
visual_html($best);
export_centroids($best);
exit;




function visual_html($arr){
$handle=fopen('centr_out.htm','w');
fwrite($handle,'<head><meta http-equiv="content-type" content="text/html;
charset=iso-8859-7></head>');

fwrite($handle,'<table>');
foreach ($arr as $one){
        if ($one['id']<>1){
        fwrite($handle, '<tr>');
        fwrite($handle,'<td>'.$one['id'].'</td>');
        fwrite($handle,'<td>'.$one['fid'].'</td>');
        foreach ($one['names'] as $name){
        fwrite($handle,'<td>'.$name.'</td>');
        }
        }
        fwrite($handle,'<td>'.$one['perif'].'</td></tr>');
}


fwrite($handle,'</table>');


}







function optimize($arr,$blist){
global $ave,$perif,$start;
$periflist=array();
foreach ($arr as $one){
  if (in_array($one['id'],$blist))continue 1;
  array_push($periflist,$one['perif']);
}
$periflist=array_unique($periflist);
```

```
asort($periflist);
array_shift($periflist);
$poplist=array();

foreach($periflist as $one){
  $temp=0;
  foreach ($arr as $two){
    if($two['perif']==$one){
    $temp=$temp+$two['pop'];
    }
  }
array_push($poplist,$temp);
}

$apoklisi=standard_deviation_population($poplist);

return $apoklisi;

}

function standard_deviation_population ($a)
{
  //variable and initializations
  $the_standard_deviation = 0.0;
  $the_variance = 0.0;
  $the_mean = 70000;
  $the_array_sum = array_sum($a); //sum the elements
  $number_elements = count($a); //count the number of elements
  //calculate the mean
//  $the_mean = $the_array_sum / $number_elements;

  //calculate the variance
  for ($i = 0; $i < $number_elements; $i++)
  {
    //sum the array
    $the_variance = $the_variance + ($a[$i] - $the_mean) * ($a[$i] - $the_mean);
  }

//  $the_variance = $the_variance / $number_elements;

  //calculate the standard deviation
  $the_standard_deviation = pow( $the_variance, 0.5);

  //return the variance
  return $the_standard_deviation;
}


function export_centroids($arr){
$handle = fopen("test.txt", "w");
```

```
        foreach ($arr as $one){
          fwrite($handle,$one['id']);
          fwrite($handle,'#');
          fwrite($handle,$one['fid']);
          fwrite($handle,'#');
          fwrite($handle,$one['sea']);
          fwrite($handle,'#');
          fwrite($handle,$one['pop']);
          fwrite($handle,'#');
            foreach($one['names'] as $name){
    fwrite($handle,$name);
    fwrite($handle,'&');
   }
   fwrite($handle,'#');
   foreach($one['git'] as $git){
     fwrite($handle,$git);
     fwrite($handle,'&');
   }
   fwrite($handle,'#');
   foreach($one['polys'] as $poly){
     fwrite($handle,$poly);
     fwrite($handle,'&');
   }
   fwrite($handle,'#');
    fwrite($handle,$one['perif']);
    fwrite($handle,'#');
    fwrite($handle,$one['x']);
    fwrite($handle,'#');
    fwrite($handle,$one['y']);

  fwrite($handle,"\n");
 }
fclose($handle);
}
?>
```

Η σημειακή προσέγγιση με κάνναβο

```
<?php
 $lines = file('meta_centr.txt');
$i=0;
$pop=0;
$perif=150;
$kapodistrias=array();
foreach ($lines as $line) {
  $i++;
  $pieces=explode("#",$line);
  $kapodistrias[$i]['id']=$pieces[0];
  $kapodistrias[$i]['fid']=$pieces[1];
  $kapodistrias[$i]['sea']=$pieces[2];
  $kapodistrias[$i]['pop']=$pieces[3];
  $kapodistrias[$i]['names']=explode("&",$pieces[4]);
  array_pop($kapodistrias[$i]['names']);
  $kapodistrias[$i]['polys']=explode("&",$pieces[6]);
  array_pop($kapodistrias[$i]['polys']);
  $kapodistrias[$i]['git']=explode("&",$pieces[5]);
  array_pop($kapodistrias[$i]['git']);
  $kapodistrias[$i]['perif']=0;
  $kapodistrias[$i]['x']=$pieces[8];
  $pieces[9] = preg_replace("/[\n\r]/","",$pieces[9]);
  $pieces[9] = preg_replace("/,/",".",$pieces[9]);
  settype($pieces[9],"double");
  $kapodistrias[$i]['y']=$pieces[9];
  $pop=$pop+$pieces[3];
}
$ave=$pop/$perif;
$thal=$kapodistrias[1];
$badlist1=array();
$flag=0;
$bigpop=0;
$badarray=array();
for($i=1;$i<count($kapodistrias);$i++){
  if($kapodistrias[$i]['pop']>50000){
  $flag++;
  $bigpop=$bigpop+$kapodistrias[$i]['pop'];
  $kapodistrias[$i]['perif']=$flag;
  array_push($badlist1,$kapodistrias[$i]['id']);
  array_push($badarray,$kapodistrias[$i]);
  }

}

$perif=150-variant_fix($bigpop/$ave);

$start=$flag+1;
```

```
$xmin=9999999999999999;
$ymin=9999999999999999;
$xmax=0;
$ymax=0;

foreach ($kapodistrias as $one){
  if($one['id']==1)continue;
  if($one['x']>$xmax)$xmax=$one['x'];
  if($one['x']<$xmin)$xmin=$one['x'];
  if($one['y']>$ymax)$ymax=$one['y'];
  if($one['y']<$ymin)$ymin=$one['y'];
}
settype($ymin,'integer');
settype($ymax,'integer');
$ymin=variant_fix($ymin);
$ymax=variant_fix($ymax);
settype($ymin,'integer');
settype($ymax,'integer');

$lastop=999999999999999999;

$lastop=999999999999999999;
$kapodistrias2=array();
$groups=array();

$ymin=$ymin-1;
$xmin=$xmin-1;
$ymax=$ymax+1;
$xmax=$xmax+1;
$vimay=($ymax-$ymin)/3;
$vimax=($xmax-$xmin)/2;

$groups[1]['xmin']=$xmin;
$groups[1]['xmax']=$xmin+$vimax;
$groups[1]['ymin']=$ymin;
$groups[1]['ymax']=$ymin+$vimay;

$groups[2]['xmin']=$xmin+$vimax;
$groups[2]['xmax']=$xmin+2*$vimax;
$groups[2]['ymin']=$ymin;
$groups[2]['ymax']=$ymin+$vimay;

$groups[3]['xmin']=$xmin;
$groups[3]['xmax']=$xmin+($vimax)/2;
$groups[3]['ymin']=$ymin+$vimay;
$groups[3]['ymax']=$ymin+2*($vimay)*3/4;
```

```
$kannav=kannavos($xmin,$xmin+($vimax)/2,$ymin+$vimay,$ymin+2*($vimay)*3/4);

$groups[7]['xmin']=$xmin+($vimax)/2;
$groups[7]['xmax']=$xmin+($vimax);
$groups[7]['ymin']=$ymin+$vimay;
$groups[7]['ymax']=$ymin+2*($vimay)*3/4;

$groups[8]['xmin']=$xmin;
$groups[8]['xmax']=$xmin+($vimax)/2;
$groups[8]['ymin']=$ymin+2*($vimay)*3/4;
$groups[8]['ymax']=$ymin+2*($vimay);

$groups[9]['xmin']=$xmin+($vimax)/2;
$groups[9]['xmax']=$xmin+($vimax);
$groups[9]['ymin']=$ymin+2*($vimay)*3/4;
$groups[9]['ymax']=$ymin+2*($vimay);

$groups[4]['xmin']=$xmin+$vimax;
$groups[4]['xmax']=$xmin+2*$vimax;
$groups[4]['ymin']=$ymin+$vimay;
$groups[4]['ymax']=$ymin+2*$vimay;

$groups[5]['xmin']=$xmin;
$groups[5]['xmax']=$xmin+$vimax/2;
$groups[5]['ymin']=$ymin+2*$vimay;
$groups[5]['ymax']=$ymin+$vimay*5/2;

$groups[10]['xmin']=$xmin;
$groups[10]['xmax']=$xmin+$vimax/2;
$groups[10]['ymin']=$ymin+$vimay*5/2;
$groups[10]['ymax']=$ymin+3*$vimay;

$groups[11]['xmin']=$xmin+$vimax/2;
$groups[11]['xmax']=$xmin+$vimax;
$groups[11]['ymin']=$ymin+2*$vimay;
$groups[11]['ymax']=$ymin+$vimay*5/2;

$groups[12]['xmin']=$xmin+$vimax/2;
$groups[12]['xmax']=$xmin+$vimax;
$groups[12]['ymin']=$ymin+$vimay*5/2;
$groups[12]['ymax']=$ymin+3*$vimay;

$groups[6]['xmin']=$xmin+$vimax;
$groups[6]['xmax']=$xmin+2*$vimax;
$groups[6]['ymin']=$ymin+2*$vimay;
$groups[6]['ymax']=$ymin+3*$vimay;

$num=0;
$globperif=$start;
foreach ($groups as $one){
```

```
   $num++;
  unset($kapotemp);
  $kapotemp=array();
  $temppop=0;
  foreach($kapodistrias as $two){
    if($two['x']>=$one['xmin'] and $two['x']<=$one['xmax'] and $two['y']>=$one['ymin'] and
$two['y']<=$one['ymax']){
      $id=$two['id'];
      if(in_array($id,$badlist1))continue 1;
      array_push($kapotemp,$two);
      $temppop+=$two['pop'];
    }
  }

$perif=variant_fix($temppop/60000);
echo "arithmos periferion pou tha ginoun einai $perif\n";
$ave=$temppop/$perif;
$flag=3000;
$lastop=9999999999999999999;
echo " to count einai".count($kapotemp)."\n";
$backup=$kapotemp;
while($flag>0){
$kapotemp=$backup;
for($i=1;$i<=$perif;$i++){

  $one['ymin']=variant_fix($one['ymin']);
  $one['ymax']=variant_fix($one['ymax']);
  settype($one['ymin'],'integer');
  settype($one['ymax'],'integer');
  $x=mt_rand($one['xmin'],$one['xmax']);
  $y=mt_rand($one['ymin'],$one['ymax']);
  $n[$i]['x']=$x;
  $n[$i]['y']=$y;
}


for($i=0;$i<count($kapotemp);$i++){

  $min=99999999999999999999;
  for($j=1;$j<=$perif;$j++){
  $iporizo=pow($kapotemp[$i]['x']-$n[$j]['x'],2)+pow($kapotemp[$i]['y']-$n[$j]['y'],2);
  $dist=sqrt($iporizo);
  if($dist<$min){
    $min=$dist;
    $kapotemp[$i]['perif']=$j;
  }
  }
}
```

```
$flag--;
$op=optimize($kapotemp,$badlist1);
 if($op<$lastop){
 $lastop=$op;
 $best=$kapotemp;
 }

}

for($i=0;$i<count($best);$i++){
$best[$i]['perif']=$best[$i]['perif']+$globperif;
}
$globperif=$globperif+$perif;
$kapodistrias2=array_merge($kapodistrias2,$best);

$periflist=array();
foreach ($best as $one){
array_push($periflist,$one['perif']);
}
$periflist=array_unique($periflist);
foreach ($periflist as $one){
  $temppop=0;
  foreach ($best as $two){
    if($one==$two['perif']){
      $temppop=$temppop+$two['pop'];
    }
  }

  if(in_array($one,$badlist1)){
    }else{
   echo "$one"."#"."$temppop\n";
  }
}
unset($periflist);


}
echo "teliko=";
echo optimize($kapodistrias2,$badlist1);
foreach ($kapodistrias as $one){
  $id=$one['id'];
  if (in_array($id,$badlist1)){
    array_push($kapodistrias2,$one);
  }
}

$kapodistrias3=array();
foreach ($kapodistrias2 as $one){
  $id=$one['id'];
  $kapodistrias3[$id]=$one;
```

```
}

echo "\n";

$periflist=array();
foreach ($kapodistrias3 as $one){
array_push($periflist,$one['perif']);
}
$periflist=array_unique($periflist);
foreach ($periflist as $one){
  $temppop=0;
  foreach ($kapodistrias3 as $two){
    if($one==$two['perif']){
      $temppop=$temppop+$two['pop'];
    }
  }

  if(in_array($one,$badlist1)){
    }else{

  }
}


$kapodistrias3[1]=$thal;
export_centroids($kapodistrias3);
visual_html($kapodistrias3);
exit;




function visual_html($arr){
$handle=fopen('centr_out.htm','w');
fwrite($handle,'<head><meta http-equiv="content-type" content="text/html;
charset=iso-8859-7></head>');

fwrite($handle,'<table>');
foreach ($arr as $one){
        if ($one['id']<>1){
        fwrite($handle, '<tr>');
        fwrite($handle,'<td>'.$one['id'].'</td>');
        fwrite($handle,'<td>'.$one['fid'].'</td>');
        foreach ($one['names'] as $name){
        fwrite($handle,'<td>'.$name.'</td>');
        }
        }
```

```
        fwrite($handle,'<td>'.$one['perif'].'</td></tr>');
}


fwrite($handle,'</table>');


}

function optimize($arr,$blist){
global $perif;
$periflist=array();
foreach ($arr as $one){
  if (in_array($one['id'],$blist))continue 1;
  array_push($periflist,$one['perif']);
}

$poplist=array();

foreach($periflist as $one){
  $temp=0;
  foreach ($arr as $two){
    if($two['perif']==$one){
    $temp=$temp+$two['pop'];
    }
  }
array_push($poplist,$temp);
}

$apoklisi=standard_deviation_population($poplist);

return $apoklisi;

}

function standard_deviation_population ($a)
{
  //variable and initializations
  $the_standard_deviation = 0.0;
  $the_variance = 0.0;
  $the_mean = 70000;
  $the_array_sum = array_sum($a); //sum the elements
  $number_elements = count($a); //count the number of elements
  //calculate the mean
//  $the_mean = $the_array_sum / $number_elements;

  //calculate the variance
  for ($i = 0; $i < $number_elements; $i++)
  {
    //sum the array
```

```
    $the_variance = $the_variance + ($a[$i] - $the_mean) * ($a[$i] - $the_mean);
  }
//  $the_variance = $the_variance / $number_elements;

  //calculate the standard deviation
  $the_standard_deviation = pow( $the_variance, 0.5);

  //return the variance
  return $the_standard_deviation;
}

function export_centroids($arr){
 $handle = fopen("test.txt", "w");
        foreach ($arr as $one){
          fwrite($handle,$one['id']);
          fwrite($handle,'#');
          fwrite($handle,$one['fid']);
          fwrite($handle,'#');
          fwrite($handle,$one['sea']);
          fwrite($handle,'#');
          fwrite($handle,$one['pop']);
          fwrite($handle,'#');
            foreach($one['names'] as $name){
    fwrite($handle,$name);
    fwrite($handle,'&');
  }
  fwrite($handle,'#');
  foreach($one['git'] as $git){
    fwrite($handle,$git);
    fwrite($handle,'&');
  }
  fwrite($handle,'#');
  foreach($one['polys'] as $poly){
    fwrite($handle,$poly);
    fwrite($handle,'&');
  }
  fwrite($handle,'#');
   fwrite($handle,$one['perif']);
   fwrite($handle,'#');
   fwrite($handle,$one['x']);
   fwrite($handle,'#');
   fwrite($handle,$one['y']);

 fwrite($handle,"\n");
}
fclose($handle);
}
?>
```

Ο αλγόριθμος του Openshaw

```
<?php
$lines = file('out.txt');
$i=0;
$kapodistrias=array();

$pop=0;
$perif=50;
foreach ($lines as $line) {
  $i++;
  $pieces=explode("#",$line);
  $kapodistrias[$i]['id']=$pieces[0];
  $kapodistrias[$i]['pop']=$pieces[2];
  $kapodistrias[$i]['sea']=$pieces[1];
  $pop=$pop+$kapodistrias[$i]['pop'];
  $kapodistrias[$i]['names']=explode("&",$pieces[3]);
  array_pop($kapodistrias[$i]['names']);
  $kapodistrias[$i]['polys']=explode("&",$pieces[5]);
  array_pop($kapodistrias[$i]['polys']);
  $kapodistrias[$i]['git']=explode("&",$pieces[4]);
  array_pop($kapodistrias[$i]['git']);
  $kapodistrias[$i]['perif']=0;
}


$ave=$pop/$perif;
$centroids=array();
$flag=1;
$lines=file('Centroid2.csv');
foreach ($lines as $one){
        $pieces=explode("#",$one);
        $pieces[7] = preg_replace("/[\n\r]/","",$pieces[7]);
        $centroids[$flag]['x']=$pieces[1];
        $centroids[$flag]['y']=$pieces[2];
        $centroids[$flag]['id']=$flag;
        $centroids[$flag]['names']=array();
        array_push($centroids[$flag]['names'],$pieces[4]);
        array_push($centroids[$flag]['names'],$pieces[5]);
        array_push($centroids[$flag]['names'],$pieces[6]);
        $flag++;
}

for($i=1;$i<=count($kapodistrias);$i++){
        foreach ($centroids as $cen){
        if($cen['names']==$kapodistrias[$i]['names']){
        $kapodistrias[$i]['x']=$cen['x'];
        $kapodistrias[$i]['y']=$cen['y'];
        }
    }
}
```

```
$limit=5;

for($i=1;$i<=count($kapodistrias);$i++){
       $fl=0;
       $sea=0;
       $coast=0;
       foreach ($kapodistrias[$i]['git'] as $git){
       if ($git==1)$sea=1;
       $fl++;
      }
      if ($sea==1 and $fl>1) $coast=1;

      if($coast==1){
            for($j=1;$j<=count($kapodistrias);$j++){
                  if($kapodistrias[$j]['sea']==1){
                  $dist=distance($kapodistrias[$i],$kapodistrias[$j]);
                  if($dist<=$limit){
                  array_push($kapodistrias[$i]['git'],$kapodistrias[$j]['id']);
                  array_push($kapodistrias[$j]['git'],$kapodistrias[$i]['id']);
                  }
                  }
                }
          }

        for($j=1;$j<=count($kapodistrias);$j++){
        if($kapodistrias[$i]['sea']==1 and $kapodistrias[$j]['sea']==1 and $i<>$j){
       $dist=distance($kapodistrias[$i],$kapodistrias[$j]);
       if($dist<=$limit){
       array_push($kapodistrias[$i]['git'],$kapodistrias[$j]['id']);
       array_push($kapodistrias[$j]['git'],$kapodistrias[$i]['id']);
       }
      }
    }
}


$lines=file("ids2.csv");
foreach ($lines as $one){
$pieces=explode("#",$one);
$pieces[3] = preg_replace("/[\n\r]/","",$pieces[3]);
 for($i=1;$i<=count($kapodistrias);$i++){
 if($kapodistrias[$i]['names'][0]==$pieces[1] and $kapodistrias[$i]['names'][1]==$pieces[2]
and $kapodistrias[$i]['names'][2]==$pieces[3]){
 $kapodistrias[$i]['fid']=$pieces[0];
 }
 }
}
```

```
$kapodistrias[1]['fid']=999999;
$kapodistrias[1]['x']=0;
$kapodistrias[1]['y']=0;



$lines=file('extra.txt');
foreach($lines as $line){
  $pieces=explode("#",$line);
  $pieces[1] = preg_replace("/[\n\r]/","",$pieces[1]);
  $one=$pieces[0];
  $two=$pieces[1];
  foreach ($kapodistrias as $dimos){
  if($dimos['fid']==$one){
     $oneid=$dimos['id'];
     }elseif($dimos['fid']==$two){
     $twoid=$dimos['id'];

  }
  }
  array_push($kapodistrias[$oneid]['git'],$twoid);
  array_push($kapodistrias[$twoid]['git'],$oneid);
  echo "to $oneid me to $twoid \n";

}

export_centroids($kapodistrias);

exit;


function export_centroids($arr){
$handle = fopen("init.txt", "w");
        foreach ($arr as $one){
          fwrite($handle,$one['id']);
          fwrite($handle,'#');
          fwrite($handle,$one['fid']);
          fwrite($handle,'#');
          fwrite($handle,$one['sea']);
          fwrite($handle,'#');
          fwrite($handle,$one['pop']);
          fwrite($handle,'#');
            foreach($one['names'] as $name){
    fwrite($handle,$name);
    fwrite($handle,'&');
  }
  fwrite($handle,'#');
  foreach($one['git'] as $git){
```

```
   fwrite($handle,$git);
   fwrite($handle,'&');
  }
  fwrite($handle,'#');
  foreach($one['polys'] as $poly){
   fwrite($handle,$poly);
   fwrite($handle,'&');
  }
  fwrite($handle,'#');
   fwrite($handle,$one['perif']);
   fwrite($handle,'#');
   fwrite($handle,$one['x']);
   fwrite($handle,'#');
   fwrite($handle,$one['y']);

 fwrite($handle,"\n");
}
fclose($handle);
}

function distance($one,$dio){
 $dist=0;
        $dx=pow(($one['x']-$dio['x']),2);
        $dy=pow(($one['y']-$dio['y']),2);
        $dist=sqrt($dx+$dy);

 return $dist;
}


function veltisto($arr){
 global $ave,$perif;
   $sums=array();
   $vel=0;

    foreach($arr as $one){
    if(isset($sums[$one['perif']])==False) $sums[$one['perif']]=0;
    $sums[$one['perif']]=$sums[$one['perif']]+$one['pop'];
          }
foreach($sums as $one){
 $vel=$vel+abs($one-$ave);
}

 return $vel/$perif;
}


function move_the_from_to($the,$to,$array){
```

```
$output=$array;
$output[$the]['perif']=$to;

return $output;
}


function str_replace_array ($search, $replace, $subject) {
   if (is_array($subject)) {
      foreach ($subject as $id=>$inner_subject) {
         $subject[$id]=str_replace_array($search, $replace, $inner_subject);
      }
   } else {
      $subject=str_replace($search, $replace, $subject);
   }
   return $subject;
}


function remove_dups($array, $row_element) {
   $new_array[1] = $array[1];
   foreach ($array as $current) {
      $add_flag = 1;
      foreach ($new_array as $tmp) {
         if ($current[$row_element]==$tmp[$row_element]) {
            $add_flag = 0; break;
         }
      }
      if ($add_flag) $new_array[] = $current;
   }
   return $new_array;
}



?>

<?php

$lines = file('init.txt');
$groups=array();

$i=0;
$pop=0;
$perif=150;
$kapodistrias=array();
foreach ($lines as $line) {
  $i++;
  $pieces=explode("#",$line);
  $kapodistrias[$i]['x']=$pieces[8];
```

```
  $pieces[9] = preg_replace("/[\n\r]/","",$pieces[9]);
  $kapodistrias[$i]['y']=$pieces[9];
  $pieces=explode("#",$line);
  $kapodistrias[$i]['id']=$pieces[0];
  $kapodistrias[$i]['fid']=$pieces[1];
  $kapodistrias[$i]['sea']=$pieces[2];
  $kapodistrias[$i]['pop']=$pieces[3];
  $kapodistrias[$i]['names']=explode("&",$pieces[4]);
  array_pop($kapodistrias[$i]['names']);
  $kapodistrias[$i]['polys']=explode("&",$pieces[6]);
  array_pop($kapodistrias[$i]['polys']);
  $kapodistrias[$i]['git']=explode("&",$pieces[5]);
  array_pop($kapodistrias[$i]['git']);
  $kapodistrias[$i]['perif']=$pieces[7];
  $pop=$pop+$pieces[3];
}
echo $pop;
echo "\n";
$citypop=0;
$badlist1=array();
$flag=0;
for($i=1;$i<count($kapodistrias);$i++){
  if($kapodistrias[$i]['pop']>50000){
  $flag++;
  $kapodistrias[$i]['perif']=$flag;
  $citypop=$citypop+$kapodistrias[$i]['pop'];
  array_push($badlist1,$kapodistrias[$i]['id']);
  }

}

$lines=file('starting.txt');
$init=array();
foreach ($lines as $line){
  $line = preg_replace("/[\n\r]/","",$line);
  array_push($init,$line);
}

$goodlist=array();
for($i=1;$i<=count($kapodistrias);$i++){
  if($kapodistrias[$i]['perif']==0)array_push($goodlist,$kapodistrias[$i]['id']);
$fid=$kapodistrias[$i]['fid'];
foreach ($init as $one){
  if($fid==$one){
    $flag++;
    $kapodistrias[$i]['perif']=$flag;
    $citypop=$citypop+$kapodistrias[$i]['pop'];
  }
```

```
}
}
$dif=$pop-$citypop;
$remain=variant_fix($dif/70000);

$extra=$flag;
$perif=150;
for($i=$flag;$i<=($perif);$i++){
  $sux=0;
  while($sux==0){
  $ran=rand(2,count($kapodistrias));
  if ($kapodistrias[$ran]['perif']==0){
  $kapodistrias[$ran]['perif']=$i;
  $sux=1;
  }
  }
 }


$flag=count($kapodistrias);

$counter=0;

while($counter<250){
  $counter++;
  for($i=2;$i<=count($kapodistrias);$i++){
    if($kapodistrias[$i]['perif']<>0) continue;
     foreach ($kapodistrias[$i]['git'] as $one){
      if($kapodistrias[$one]['perif']<>0){
       $kapodistrias[$i]['perif']=$kapodistrias[$one]['perif'];
       $flag--;
       echo ($flag-$extra);
       echo "\n";
       continue 2;
      }

     }

  }

}

export_centroids($kapodistrias);
exit;


function apoklisi($arr){
global $ave;

$list=array();
```

```
$list=array_fill(0,51,0);
foreach($arr as $one){
        $id=$one['perif'];
        $list[$id]=$list[$id]+$one['pop'];
}
$sum=0;
foreach ($list as $one){
$sum=$sum+pow(($one-$ave),2);
}
return sqrt($sum);
}



function choose2($arr){
$pithana=array();
foreach($arr as $one){
 $templist=get_git_perif($arr,$one['id']);
 $templist=array_remove($templist,$one['perif']);
 $templist=array_unique($templist);
 if(count($templist)>0 and $one['id']>1){
 array_push($pithana,$one['id']);
 }
 }
return $pithana;
}



function get_git_perif($arr,$poly){
$gitperif=array();
foreach ($arr[$poly]['git'] as $one){
array_push($gitperif,$arr[$one]['perif']);
}
$gitperif=array_unique($gitperif);
return $gitperif;
}


function veltisto($arr,$poly,$to){
global $ave;
$from=$arr[$poly]['perif'];
$sumfrom=0;
$sumto=0;
foreach ($arr as $one){
if($one['perif']==$from)$sumfrom=$sumfrom+$one['pop'];
if($one['perif']==$to)$sumto=$sumto+$one['pop'];
}
```

```
$prinfrom=$sumfrom;
$printo=$sumto;
//echo "\n";

//echo "plithismos $sumfrom tou $from \n";
//echo "plithismos $sumto tou $to \n";
$metafrom=$prinfrom-$arr[$poly]['pop'];
$metato=$printo+$arr[$poly]['pop'];
$prin=abs($prinfrom-$ave)+abs($printo-$ave);
$meta=abs($metafrom-$ave)+abs($metato-$ave);
$vlaka=0;
if($sumto<$sumfrom) $vlaka=1;

return $vlaka;
}


function visual($arr){
echo '<table>';
foreach ($arr as $one){
        if ($one['id']<>1){
        echo '<tr>';
        echo '<td>'.$one['id'].'</td>';
        echo '<td>'.$one['fid'].'</td>';
        foreach ($one['names'] as $name){
        echo '<td>'.$name.'</td>';
        }
        }
        echo '<td>'.$one['perif'].'</td></tr>';
}


echo '</table>';


}




function ring($poly,$arr){
$ring=array();
$temp=$arr[$poly]['git'];
if($temp[0]==1){
  array_push($temp,$temp[0]);
  array_shift($temp);
}
array_push($ring,$temp[0]);
```

```
array_remove($temp,$temp[0]);
array_remove($temp,$poly);
$flag=count($temp);
$prev=0;
$counter=0;
while($prev<$flag){
  $counter++;
  if ($counter>30)break;
            foreach($temp as $one){
              $dum=$ring[$prev];
              if(in_array($one,$arr[$dum]['git'])){
              $prev++;
              $ring[$prev]=$one;
              array_remove($temp,$one);
              continue 2;
             }

            }
            $middle=array();
            foreach($temp as $one){
            $middle=array_intersect($arr[$one]['git'],$arr[$ring[$prev]]['git']);
            if (count($middle)>1){
              $prev++;
              $ring[$prev]=$one;
              array_remove($temp,$one);
              continue 2;
            }

          }

 }
$ring=array_unique($ring);
return $ring;
}


function count_perif_polys($arr,$poly){
$flag=0;
$perif=$arr[$poly]['perif'];
foreach ($arr as $one){
  if($one['perif']==$perif)$flag++;
  }
return $flag;
}

function switching($poly,$arr){
  $ring=array();
  $ring=ring($poly,$arr);
```

```
  $current=$arr[$poly]['perif'];
  $previous=$ring[count($ring)-1];
  $switch=0;
  for($i=0;$i<count($ring);$i++){
    $id=$ring[$i];
    $next=$arr[$id]['perif'];
    if($previous==$current and $next<>$current)$switch++;
    if($previous<>$current and $next==$current)$switch++;
    $previous=$next;
  }

 return $switch;
}



function array_remove(array &$a_Input, $m_SearchValue, $b_Strict = False) {
    $a_Keys = array_keys($a_Input, $m_SearchValue, $b_Strict);
    foreach($a_Keys as $s_Key) {
        unset($a_Input[$s_Key]);
    }
    $i=0;
    $na_Input=array();
    foreach($a_Input as $one){
    $na_Input[$i]=$one;
    $i++;
    }

    return $na_Input;
}



function export_centroids($arr){
 $handle = fopen("final.txt", "w");
        foreach ($arr as $one){
          fwrite($handle,$one['id']);
          fwrite($handle,'#');
          fwrite($handle,$one['fid']);
          fwrite($handle,'#');
          fwrite($handle,$one['sea']);
          fwrite($handle,'#');
          fwrite($handle,$one['pop']);
          fwrite($handle,'#');
            foreach($one['names'] as $name){
    fwrite($handle,$name);
    fwrite($handle,'&');
  }
  fwrite($handle,'#');
```

```
 foreach($one['git'] as $git){
   fwrite($handle,$git);
   fwrite($handle,'&');
  }
  fwrite($handle,'#');
  foreach($one['polys'] as $poly){
   fwrite($handle,$poly);
   fwrite($handle,'&');
  }
  fwrite($handle,'#');
   fwrite($handle,$one['perif']);
//   if ($one['id']==1)continue 1;
   fwrite($handle,'#');
   fwrite($handle,$one['x']);
   fwrite($handle,'#');
   fwrite($handle,$one['y']);

 fwrite($handle,"\n");
}
fclose($handle);
}
?>
```

Η συνδυαστική μέθοδος

```
<?php
$lines = file('init.txt');
$groups=array();

$i=0;
$pop=0;
$perif=50;
$kapodistrias=array();
foreach ($lines as $line) {
  $i++;
  $pieces=explode("#",$line);
  $kapodistrias[$i]['x']=$pieces[8];
  $pieces[9] = preg_replace("/[\n\r]/","",$pieces[9]);
  $kapodistrias[$i]['y']=$pieces[9];
  $pieces=explode("#",$line);
  $kapodistrias[$i]['id']=$pieces[0];
  $kapodistrias[$i]['fid']=$pieces[1];
  $kapodistrias[$i]['sea']=$pieces[2];
  $kapodistrias[$i]['pop']=$pieces[3];
  $kapodistrias[$i]['names']=explode("&",$pieces[4]);
  array_pop($kapodistrias[$i]['names']);
  $kapodistrias[$i]['polys']=explode("&",$pieces[6]);
  array_pop($kapodistrias[$i]['polys']);
  $kapodistrias[$i]['git']=explode("&",$pieces[5]);
  array_pop($kapodistrias[$i]['git']);
  $kapodistrias[$i]['perif']=$pieces[7];
  $pop=$pop+$pieces[3];
}

$xmin=9999999999999999;
$ymin=9999999999999999;
$xmax=0;
$ymax=0;

$badlist1=array();
$flag=0;


for($i=1;$i<count($kapodistrias);$i++){
  if($kapodistrias[$i]['pop']>50000){
  $flag++;
  $kapodistrias[$i]['perif']=$flag;
  array_push($badlist1,$kapodistrias[$i]['id']);
  }

}

$single=$flag;
```

```
foreach ($kapodistrias as $one){
  if($one['id']==1)continue;
  if($one['x']>$xmax)$xmax=$one['x'];
  if($one['x']<$xmin)$xmin=$one['x'];
  if($one['y']>$ymax)$ymax=$one['y'];
  if($one['y']<$ymin)$ymin=$one['y'];
}
settype($ymin,'integer');
settype($ymax,'integer');


$kapodistrias2=array();
$groups=array();
$lastop=999999999999999999;
$ymin=$ymin-1;
$xmin=$xmin-1;
$ymax=$ymax+1;
$xmax=$xmax+1;
$vimay=($ymax-$ymin)/3;
$vimax=($xmax-$xmin)/2;

//Athina
$groups[1]['xmin']=454182;
$groups[1]['xmax']=504160;
$groups[1]['ymin']=4179735;
$groups[1]['ymax']=4231541;
//Thessaloniki
$groups[2]['xmin']=386966;
$groups[2]['xmax']=428399;
$groups[2]['ymin']=4480487;
$groups[2]['ymax']=4519988;
//Nisia

$groups[3]['xmin']=517884;
$groups[3]['xmax']=911827;
$groups[3]['ymin']=3935489;
$groups[3]['ymax']=4432328;

$initperif=0;
$periflist=array();
$mpee=0;
foreach ($groups as $one){
  $periflist=array();
  $mpee++;
  echo "Current Group is $mpee\n";
$currentgroup=array();
$currentpop=0;
```

```
  $badlist=array();
   foreach($kapodistrias as $two){

    if($two['x']>$one['xmin'] and $two['x']<$one['xmax'] and $two['y']>$one['ymin'] and
$two['y']<$one['ymax']){
      array_push($badlist,$two['id']);
      if(in_array($two['id'],$badlist1)){

      }else{

      array_push($currentgroup,$two);
      $currentpop+=$two['pop'];
      }
     }
  }

//Epilogi ton kentron gia ti lista $currentgroup
$currentperif=(variant_int($currentpop/70000));

$ave=$currentpop/$currentperif;

//Poso na tin psaksoume
$polapl=500;

$flag=$currentperif*$polapl;
$lastop=9999999999999999999;
while($flag>0){
for($i=1;$i<=$currentperif;$i++){
  $one['ymin']=variant_fix($one['ymin']);
  $one['ymax']=variant_fix($one['ymax']);
  settype($one['ymin'],'integer');
  settype($one['ymax'],'integer');
  $x=mt_rand($one['xmin'],$one['xmax']);
  $y=mt_rand($one['ymin'],$one['ymax']);
  $n[$i]['x']=$x;
  $n[$i]['y']=$y;
}

for($i=0;$i<count($currentgroup);$i++){
  $min=99999999999999999999;
  for($j=1;$j<=$currentperif;$j++){
  $iporizo=pow($currentgroup[$i]['x']-$n[$j]['x'],2)+pow($currentgroup[$i]['y']-$n[$j]['y'],2);
  $dist=sqrt($iporizo);
  if($dist<$min){
    $min=$dist;
    $currentgroup[$i]['perif']=$j;
  }
 }
}
$flag--;
```

```
$op=optimize($currentgroup);
 if($op<$lastop){
 $lastop=$op;
 $best=$currentgroup;
 echo $op;
 echo "\n";
 }



}

$single=$single+$currentperif;
for($i=1;$i<=count($kapodistrias);$i++){
  foreach ($best as $one){
    if($kapodistrias[$i]['id']==$one['id']){
    $kapodistrias[$i]['perif']=$one['perif']+$single;
    array_push($periflist,$one['perif']);
    }
  }
}
$periflist=array_unique($periflist);


foreach ($periflist as $one){
  $pops=0;
  foreach ($best as $two){

    if($two['perif']==$one){
      if ($one==0) echo "yoooo".$two['fid']."\n";
      $pops=$pops+$two['pop'];
    }
  }
  echo "pop einai : $pops \n";
}

}


foreach ($kapodistrias as $one){
  array_push($periflist,$one['perif']);
}

$periflist=array_unique($periflist);
$periflist=fixkeys($periflist);

$kapodistrias[365]['perif']=0;
$kapodistrias[321]['perif']=0;
```

```
$kapodistrias[285]['perif']=0;
$kapodistrias[388]['perif']=0;

export_centroids($kapodistrias);

exit;




function visual_html($arr){
$handle=fopen('centr_out.htm','w');
fwrite($handle,'<head><meta http-equiv="content-type" content="text/html;
charset=iso-8859-7></head>');

fwrite($handle,'<table>');
foreach ($arr as $one){
        if ($one['id']<>1){
        fwrite($handle, '<tr>');
        fwrite($handle,'<td>'.$one['id'].'</td>');
        fwrite($handle,'<td>'.$one['fid'].'</td>');
        foreach ($one['names'] as $name){
        fwrite($handle,'<td>'.$name.'</td>');
        }
        }
        fwrite($handle,'<td>'.$one['perif'].'</td></tr>');
}


fwrite($handle,'</table>');


}
function optimize($arr){
global $ave,$currentperif;
$list=array();
$list=array_fill(1,$currentperif,0);
for($j=1;$j<=$currentperif;$j++){
  foreach($arr as $one){
    if($one['perif']==$j){
      $list[$j]=$list[$j]+$one['pop'];
    }
  }
}
$apoklisi=0;
foreach($list as $one){
  $apoklisi=$apoklisi+pow(($one-$ave),2);
}
```

```
return $apoklisi;

}



function fixkeys($arr){
  $result=array();
  asort($arr);
  foreach ($arr as $one){
    array_push($result,$one);

  }
return $result;
}

function export_centroids($arr){
$handle = fopen("final.txt", "w");
        foreach ($arr as $one){
          fwrite($handle,$one['id']);
          fwrite($handle,'#');
          fwrite($handle,$one['fid']);
          fwrite($handle,'#');
          fwrite($handle,$one['sea']);
          fwrite($handle,'#');
          fwrite($handle,$one['pop']);
          fwrite($handle,'#');
            foreach($one['names'] as $name){
   fwrite($handle,$name);
   fwrite($handle,'&');
  }
  fwrite($handle,'#');
  foreach($one['git'] as $git){
    fwrite($handle,$git);
   fwrite($handle,'&');
  }
  fwrite($handle,'#');
  foreach($one['polys'] as $poly){
    fwrite($handle,$poly);
   fwrite($handle,'&');
  }
  fwrite($handle,'#');
   fwrite($handle,$one['perif']);
//   if ($one['id']==1)continue 1;
   fwrite($handle,'#');
```

```
  fwrite($handle,$one['x']);
  fwrite($handle,'#');
  fwrite($handle,$one['y']);

 fwrite($handle,"\n");
}
fclose($handle);
}
?>

<?php


$lines = file('final.txt');
$groups=array();

$i=0;
$pop=0;
$perif=150;
$kapodistrias=array();
foreach ($lines as $line) {
  $i++;
  $pieces=explode("#",$line);
  $kapodistrias[$i]['x']=$pieces[8];
  $pieces[9] = preg_replace("/[\n\r]/","",$pieces[9]);
  $kapodistrias[$i]['y']=$pieces[9];
  $pieces=explode("#",$line);
  $kapodistrias[$i]['id']=$pieces[0];
  $kapodistrias[$i]['fid']=$pieces[1];
  $kapodistrias[$i]['sea']=$pieces[2];
  $kapodistrias[$i]['pop']=$pieces[3];
  $kapodistrias[$i]['names']=explode("&",$pieces[4]);
  array_pop($kapodistrias[$i]['names']);
  $kapodistrias[$i]['polys']=explode("&",$pieces[6]);
  array_pop($kapodistrias[$i]['polys']);
  $kapodistrias[$i]['git']=explode("&",$pieces[5]);
  array_pop($kapodistrias[$i]['git']);
  $kapodistrias[$i]['perif']=$pieces[7];
  $pop=$pop+$pieces[3];
}

$max=0;
foreach ($kapodistrias as $one){
  if($one['perif']>$max)$max=$one['perif'];
}


$lines=file('starting.txt');
$init=array();
foreach ($lines as $line){
```

```
  $line = preg_replace("/[\n\r]/","",$line);
  array_push($init,$line);
}

$flag=$max;
$goodlist=array();
for($i=1;$i<=count($kapodistrias);$i++){
  if($kapodistrias[$i]['perif']==0)array_push($goodlist,$kapodistrias[$i]['id']);
$fid=$kapodistrias[$i]['fid'];
foreach ($init as $one){
  if($fid==$one){
    $flag++;
    $kapodistrias[$i]['perif']=$flag;
  }
}
}

//Random periferies
$max=0;
foreach ($kapodistrias as $one){
  if($one['perif']>$max)$max=$one['perif'];

}
$backup=$kapodistrias;
$inner=99;

$kapodistrias=$backup;
$flag=$max;

$flag++;
$counter=$perif-$flag;

$counter=0;
while($counter<350){
  for($i=2;$i<=count($kapodistrias);$i++){
    if($kapodistrias[$i]['perif']<>0) continue;
     foreach ($kapodistrias[$i]['git'] as $one){
      if($kapodistrias[$one]['perif']<>0){
       $kapodistrias[$i]['perif']=$kapodistrias[$one]['perif'];
       continue 2;
      }

     }

  }
$counter++;
}
$inner=0;
```

```
foreach ($kapodistrias as $one){
 if ($one['perif']==0)$inner++;

}

//}
visual_html($kapodistrias);
export_centroids($kapodistrias);
exit;



function export_centroids($arr){
$handle = fopen("filled2.txt", "w");
        foreach ($arr as $one){
          fwrite($handle,$one['id']);
          fwrite($handle,'#');
          fwrite($handle,$one['fid']);
          fwrite($handle,'#');
          fwrite($handle,$one['sea']);
          fwrite($handle,'#');
          fwrite($handle,$one['pop']);
          fwrite($handle,'#');
            foreach($one['names'] as $name){
   fwrite($handle,$name);
   fwrite($handle,'&');
  }
  fwrite($handle,'#');
  foreach($one['git'] as $git){
    fwrite($handle,$git);
    fwrite($handle,'&');
  }
  fwrite($handle,'#');
  foreach($one['polys'] as $poly){
    fwrite($handle,$poly);
    fwrite($handle,'&');
  }
  fwrite($handle,'#');
   fwrite($handle,$one['perif']);
   fwrite($handle,'#');
   fwrite($handle,$one['x']);
   fwrite($handle,'#');
   fwrite($handle,$one['y']);

 fwrite($handle,"\n");
}
fclose($handle);
}
```

```
function visual_html($arr){
 $handle=fopen('centr_out2.htm','w');
fwrite($handle,'<head><meta http-equiv="content-type" content="text/html;
 charset=iso-8859-7></head>');

fwrite($handle,'<table>');
foreach ($arr as $one){
        if ($one['id']<>1){
        fwrite($handle, '<tr>');
        fwrite($handle,'<td>'.$one['id'].'</td>');
        fwrite($handle,'<td>'.$one['fid'].'</td>');
        foreach ($one['names'] as $name){
        fwrite($handle,'<td>'.$name.'</td>');
        }
        }
        fwrite($handle,'<td>'.$one['perif'].'</td></tr>');
}


fwrite($handle,'</table>');


}
?>

<?php
$lines = file('filled2.txt');
$i=0;
$pop=0;
$kapodistrias=array();
foreach ($lines as $line) {
  $i++;
  $pieces=explode("#",$line);
  $kapodistrias[$i]['x']=$pieces[8];
  $pieces[9] = preg_replace("/[\n\r]/","",$pieces[9]);
  $kapodistrias[$i]['y']=$pieces[9];
  $pieces=explode("#",$line);
  $kapodistrias[$i]['id']=$pieces[0];
  $kapodistrias[$i]['fid']=$pieces[1];
  $kapodistrias[$i]['sea']=$pieces[2];
  $kapodistrias[$i]['pop']=$pieces[3];
  $kapodistrias[$i]['names']=explode("&",$pieces[4]);
  array_pop($kapodistrias[$i]['names']);
  $kapodistrias[$i]['polys']=explode("&",$pieces[6]);
  array_pop($kapodistrias[$i]['polys']);
  $kapodistrias[$i]['git']=explode("&",$pieces[5]);
  array_pop($kapodistrias[$i]['git']);
  $kapodistrias[$i]['perif']=$pieces[7];
```

```
  $pop=$pop+$pieces[3];
}
$max=0;
$periflist=array();
foreach($kapodistrias as $one){
  if($one['perif']>$max)$max=$one['perif'];
array_push($periflist,$one['perif']);
}
$periflist=array_unique($periflist);
$perif=$max;
$kapofix=$kapodistrias;
unset($kapodistrias);
$kapodistrias=array();
foreach ($kapofix as $one){
  $id=$one['id'];
  $kapodistrias[$id]=$one;
}
ksort($kapodistrias);
$kapodistrias[1]['perif']=999;

$ave=$pop/$max;

$temper=1;
$loopes=0;
$loopesxi=1;
$lastchance=1;

while($loopesxi>0){
  $loopesxi--;
  $loopes++;
echo "eimaste stin loumpa $loopes\n";
$polylist=choose2($kapodistrias);

foreach ($polylist as $onepoly){

$pithana=get_git_perif($kapodistrias,$onepoly);

        foreach($pithana as $try){
   if($try==999)continue 1;
  $kapotemp=$kapodistrias;
  $kapotemp[$onepoly]['perif']=$try;
  $meta=apoklisi2($kapotemp);
  $prin=apoklisi2($kapodistrias);
  if($meta<$prin and switching($onepoly,$kapodistrias)<=2 and count_perif_polys($kapodistrias,
$onepoly)>1){
  $kapodistrias[$onepoly]['perif']=$try;
   $loopesxi=1;
                continue 2;

  }elseif($meta>$prin and switching($onepoly,$kapodistrias)<=2 and
```

```
count_perif_polys($kapodistrias,$onepoly)>1){
  $prob=mt_rand(0,10);
  $yo=$meta-$prin;
  $p=exp(-($meta-$prin)/$temper);
   if($p>0.5){
   $kapodistrias[$onepoly]['perif']=$try;
   $loopesxi=1;
                    continue 2;
   }
  }
}
}

 echo $meta."\n";
if($lastchance>0 and $loopesxi==0){
$kapodistrias=lastchance($kapodistrias);
$kapodistrias=lastchance($kapodistrias);
  $lastchance=0;
  $loopesxi=1;
}
$temper=$temper*0.50;
}



visual_html($kapodistrias);
export_centroids($kapodistrias);
exit;



//Functions


function is_compact($arr,$try){
  $compact=1;
  $x=array();
  $y=array();
  foreach ($arr as $one){
    if($one['perif']==$try){
      array_push($x,$one['x']);
      array_push($y,$one['y']);
    }
   }
   asort($x);
   asort($y);
   $maxelementx=count($x)-1;
```

```
   $maxelementy=count($y)-1;
   $dx=$x[$maxelementx]-$x[0];
   $dy=$y[$maxelementy]-$y[0];
   if($dx>2*$dy or $dy>2*$dx) $compact=0;

return $compact;
}



function visual_html($arr){
$handle=fopen('centr_out.htm','w');
fwrite($handle,'<head><meta http-equiv="content-type" content="text/html;
charset=iso-8859-7></head>');

fwrite($handle,'<table>');
foreach ($arr as $one){
       if ($one['id']<>1){
       fwrite($handle, '<tr>');
       fwrite($handle,'<td>'.$one['id'].'</td>');
       fwrite($handle,'<td>'.$one['fid'].'</td>');
       foreach ($one['names'] as $name){
       fwrite($handle,'<td>'.$name.'</td>');
       }
       }
       fwrite($handle,'<td>'.$one['perif'].'</td></tr>');
}


fwrite($handle,'</table>');


}


function choose2($arr){
  unset($pithana);
$pithana=array();
foreach($arr as $one){

  if ($one['id']==1 or $one['perif']==0)continue 1;
 $templist=get_git_perif($arr,$one['id']);
 $templist=array_remove($templist,$one['perif']);
 $templist=array_remove($templist,0);
 $templist=array_unique($templist);
 if(count($templist)>=1 and $one['id']>1){
 array_push($pithana,$one['id']);
 }
 }
```

```
return $pithana;
}


function get_git_perif($arr,$poly){
$gitperif=array();
foreach ($arr[$poly]['git'] as $one){
  if ($one==1034)echo "\n $poly";
array_push($gitperif,$arr[$one]['perif']);
}
$gitperif=array_unique($gitperif);
return $gitperif;
}


function array_remove(array &$a_Input, $m_SearchValue, $b_Strict = False) {
    $a_Keys = array_keys($a_Input, $m_SearchValue, $b_Strict);
    foreach($a_Keys as $s_Key) {
        unset($a_Input[$s_Key]);
    }
    $i=0;
    $na_Input=array();
    foreach($a_Input as $one){
    $na_Input[$i]=$one;
    $i++;
    }

    return $na_Input;
}




function standard_deviation_population ($a)
{
  //variable and initializations
  $the_standard_deviation = 0.0;
  $the_variance = 0.0;
  $the_mean = 0.0;
  $the_array_sum = array_sum($a); //sum the elements
  $number_elements = count($a); //count the number of elements

  //calculate the mean
  $the_mean = $the_array_sum / $number_elements;

  //calculate the variance
  for ($i = 0; $i < $number_elements; $i++)
  {
```

```
    //sum the array
    $the_variance = $the_variance + ($a[$i] - $the_mean) * ($a[$i] - $the_mean);
  }

  $the_variance = $the_variance / $number_elements;

  //calculate the standard deviation
  $the_standard_deviation = pow( $the_variance, 0.5);

  //return the variance
  return $the_standard_deviation;
}


function deviation($arr){
$perif=array();
  foreach ($arr as $one){
  array_push($perif,$one['perif']);
}
$perif=array_unique($perif);
$pops=array();
foreach ($perif as $one){
  if ($perif==0)continue 1;
  $temppop=0;
  foreach ($arr as $two){
    if($one==$two['perif']){
      $temppop=$temppop+$two['pop'];
    }
  }
array_push($pops,$temppop);
}

$std=standard_deviation_population($pops);

return $std;

}



function apoklisi2($arr){
global $ave,$perif;
$perif2=$perif+1;
$list=array();
$list=array_fill(0,$perif2,0);
foreach($arr as $one){
  $id=$one['perif'];
  if ($id==999)continue 1;
        $list[$id]+=$one['pop'];
}
```

```
$result=standard_deviation_population($list);
return $result;
}

function switching($poly,$arr){
  $ring=array();
  $ring=ring($poly,$arr);
  $current=$arr[$poly]['perif'];
  $previous=$ring[count($ring)-1];
  $previous=$arr[$previous]['perif'];
  $switch=0;
  for($i=0;$i<count($ring);$i++){
    $id=$ring[$i];
    $next=$arr[$id]['perif'];
    if($previous==$current and $next<>$current)$switch++;
    if($previous<>$current and $next==$current)$switch++;
    $previous=$next;
  }

return $switch;
}


function ring($poly,$arr){
  unset($ring);
$ring=array();
$temp=$arr[$poly]['git'];
if(count($temp)==0){
  echo "\n $poly";
  exit;
}
if($temp[0]==1){
  array_push($temp,$temp[0]);
  array_shift($temp);
}
array_push($ring,$temp[0]);
array_remove($temp,$temp[0]);
array_remove($temp,$poly);
$flag=count($temp);
$prev=0;
$counter=0;
while($prev<$flag){
  $counter++;
  if ($counter>30)break;
            foreach($temp as $one){
              $dum=$ring[$prev];
              if(in_array($one,$arr[$dum]['git'])){
              $prev++;
```

```
            $ring[$prev]=$one;
            array_remove($temp,$one);
            continue 2;
          }

        }
        $middle=array();
        foreach($temp as $one){
        $middle=array_intersect($arr[$one]['git'],$arr[$ring[$prev]]['git']);
        if (count($middle)>1){
            $prev++;
            $ring[$prev]=$one;
            array_remove($temp,$one);
            continue 2;
        }

       }

 }
$ring=array_unique($ring);
return $ring;
}


function count_perif_polys($arr,$poly){
$flag=0;
$perif=$arr[$poly]['perif'];
foreach ($arr as $one){
  if($one['perif']==$perif)$flag++;
  }
return $flag;
}

function export_centroids($arr){
$handle = fopen("test.txt", "w");
       foreach ($arr as $one){
         fwrite($handle,$one['id']);
         fwrite($handle,'#');
         fwrite($handle,$one['fid']);
         fwrite($handle,'#');
         fwrite($handle,$one['sea']);
         fwrite($handle,'#');
         fwrite($handle,$one['pop']);
         fwrite($handle,'#');
           foreach($one['names'] as $name){
   fwrite($handle,$name);
   fwrite($handle,'&');
  }
  fwrite($handle,'#');
  foreach($one['git'] as $git){
```

```
   fwrite($handle,$git);
   fwrite($handle,'&');
  }
  fwrite($handle,'#');
  foreach($one['polys'] as $poly){
    fwrite($handle,$poly);
    fwrite($handle,'&');
  }
  fwrite($handle,'#');
   fwrite($handle,$one['perif']);
   fwrite($handle,'#');
   fwrite($handle,$one['x']);
   fwrite($handle,'#');
   fwrite($handle,$one['y']);

 fwrite($handle,"\n");
}
fclose($handle);
}


function lastchance($arr){
  $badlist=array();
$periflist=array();
        foreach ($arr as $one){
        array_push($periflist,$one['perif']);
        }
$periflist=array_unique($periflist);
foreach ($periflist as $one){
  $sux=0;
  $gitlist=array();
  foreach ($arr as $two){
    if($one==$two['perif']){
    foreach ($two['git'] as $three){
      if($arr[$three]['perif']<>$one)array_push($gitlist,$arr[$three]['perif']);
    }
    }
  }

$gitlist=array_unique($gitlist);
if(count($gitlist)==1){
  $needsbrake=$gitlist[0];
  if($needsbrake==999 or in_array($needsbrake,$badlist))continue 1;
  echo " \n Tha spasei to ".$needsbrake."\n";
  foreach ($arr as $four){
    if($four['perif']==$needsbrake and $sux==0){
      $giperif=get_git_perif($arr,$four['id']);
```

```
    if(switching($four['id'],$arr)==4 and in_array($one,$giperif)){
      $id=$four['id'];
      $arr[$id]['perif']=$one;
      $moufa=$arr[$id]['fid'];
      echo "\n To $moufa ginete $one \n";
      array_push($badlist,$needsbrake);

  //    echo $id;
      $sux=1;
     }
   }
  }
}



}
return $arr;
}
?>
```