ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΕΦΑΡΜΟΣΜΕΝΑ ΗΛΕΚΤΡΟΝΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Δημιουργία λογισμικού υπολογισμού βέλτιστης διαδρομής εκσκαφής και τοποθέτησης οπτικών ινών»

Του φοιτητή
Γκέκα Δημήτριου
Αρ. Μητρώου: 51902Μ

Επιβλέπων
Τζέκης Παναγιώτης
Αναπληρωτής
Καθηγητής

Ιούνιος 2021

Δημιουργία λογισμικού υπολογισμού βέλτιστης διαδρομής εκσκαφής και τοποθέτησης οπτικών ινών
Κωδικός Δ.Ε. 21154
Γκέκας Δημήτριος
Τζέκης Παναγιώτης
Ημερομηνία ανάληψης Δ.Ε. 30/01/2021
Ημερομηνία περάτωσης Δ.Ε. 15/06/2021

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της μεταπτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Μεταπτυχιακό Πρόγραμμα Σπουδών «Εφαρμοσμένα Ηλεκτρονικά Συστήματα» στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Αφιέρωση»

Στην γυναίκα μου Αλεξάνδρα για την πολύτιμη στήριξη της και στον γιό μου Ορφέα, για όλες τις ώρες παιχνιδιού που στερήθηκε κατά την διάρκεια αυτού του ταξιδιού.

Πρόλογος

Η εργασία αυτή έχει ως στόχο τη δημιουργία ενός αυτοματοποιημένου συστήματος σχεδίασης οπτικού δικτύου, το οποίο είναι σε θέση να εντοπίζει τη βέλτιστη περιοχή στην οποία θα αναπτυχθεί, αλλά και την αυτόματη σχεδίαση του. Με πολυετή εμπειρία στον χώρο και έχοντας γνώση ότι τα συστήματα σχεδίασης/καταγραφής είναι απαραίτητα εργαλεία τα οποία κοστίζουν αρκετές χιλιάδες ευρώ, τόσο στην αγορά τους, όσο και στις συνδρομές τους και λαμβάνοντας υπόψιν ότι αυτή τη στιγμή η κατασκευή των οπτικών δικτύων βρίσκεται σε πλήρη ανάπτυξη. Έγινε μοντελοποίηση της διαδικασίας αυτής με τη χρήση του λογισμικού Qgis. Το μεγαλύτερο όφελος χρήσης εντοπίζεται στις εργατοώρες που μπορεί να εξοικονομήσει ένας μηχανικός και στην πρόληψη του ανθρωπίνου λάθους, το οποίο μπορεί να δημιουργήσει ζημίες μεγάλου κόστους. Τα εξαγόμενα αποτελέσματα είναι η οπτική σχεδίαση του δικτύου επάνω στον χάρτη αλλά και η εξαγωγή μιας σχέσης κόστους/κέρδους που θα δώσει στον μηχανικό την επιλογή υλοποίησης ή μη του έργου. Το σύστημα συνεχίζει να είναι σε πλήρη ανάπτυξη και πέρα από την σχεδίαση εντάσσονται και λειτουργίες καταγραφής του δικτύου που είναι εξίσου σημαντικές.

Περίληψη

Πριν προχωρήσει η υλοποίηση ενός δικτύου απαιτείται να γίνει έρευνα και να συνυπολογιστούν πολλαπλοί παράγοντες. Κάποιοι από αυτούς είναι, η περιοχή στην οποία θα αναπτυχθεί το δίκτυο, η ζήτηση, οι δυνητικοί χρήστες, τα δομικά υλικά που είναι κατασκευασμένο το οδόστρωμα, αλλά και τα κριτήρια/άρθρα του δήμου που απαιτούνται για την έκδοση αδειών. Όλα τα παραπάνω αποτελούν παράγοντες οι οποίοι επηρεάζουν άμεσα τον προϋπολογισμό του έργου, κρίνοντας το, βιώσιμο ή μη. Με την χρήση των μοντέλων που δημιουργήθηκαν, όλοι αυτοί οι παράγοντες υπολογίζονται αυτοματοποιημένα και δίνουν τη δυνατότητα στον μηχανικό, να μπορεί να αποφασίσει εύκολα και γρήγορα εάν είναι προς όφελος του να προχωρήσει το έργο ή όχι. Το λογισμικό μπορεί να εντοπίζει την περιοχή που είναι πιο ωφέλιμη, να σχεδιάζει το δίκτυο σε λίγα βήματα, να παρέχει πληροφορίες καταγραφής δικτύου και τέλος να εξάγει αποτελέσματα όπως τα υλικά που θα χρειαστούν. Ακόμα μπορεί να συγκρίνει το κόστος των αδειών και το κόστος εκσκαφής σε σχέση με το προσδοκώμενο κέρδος της επένδυσης. Βασικό πλεονέκτημα χρήσης του λογισμικού είναι ότι η διαδικασία που απαιτείται για να σχεδιαστεί ένα δίκτυο και να αδειοδοτηθεί μειώνεται στο ελάχιστο. Η σχεδίαση ενός δικτύου της τάξεως των πέντε χιλιομέτρων, απαιτεί περίπου δέκα ημέρες εργασίας. Συμπεριλαμβανομένης και της διαδικασίας των διορθώσεων, που προκύπτουν από την επιτόπια αυτοψία, μπορεί να φτάσει και στον ένα μήνα. Με την χρήση του μοντέλου η σχεδίαση γίνεται σε μια ώρα. Επιπλέον, μεγάλο όφελος αποτελεί το γεγονός ότι παρέχει κάλυψη δικτύου 100%, κάτι που με την χειροκίνητη σχεδίαση δεν είναι πάντα δεδομένο, καθώς από λάθος του μηχανικού μπορεί να γίνει παράληψη σύνδεσης κάποιου κτιρίου.

«Develop a software that calculates the shortest paths of excavation and placement of optical cables »

«Dimitris Gkekas»

Abstract

In order to create an optical network, it is important first to analyze the area it will be developed and take in consideration multiple factors before. Some of which are, the area in which the network will be developed, the demand, the potential users, the building materials of the roads, the criteria of the municipality required, in order to have the permission to make contractions in the area and many others. All the above are factors that directly affect the budget of the project that make it viable or not. By using the software, all these factors are calculated automatically and give the engineer the advantage to decide easily and quickly, if it is in his interest to proceed with the project or not. The software can locate the area that is most officiant to plan the network in a few steps, provide network logging information, and extract results such as the materials that will be needed, the cost of permits, the cost of excavation relative to the expected return on investment. A key advantage of using the software is that the process required to design a network and license it is kept to a minimum. Designing a five-kilometer network takes about ten days. Adding the on-site autopsy, corrections and reschedule it can be up to a month. Using the software, the design time drop to an hour. In addition, a great benefit is the fact that it provides 100% network coverage, something that with manual design is not always a given, as due to a mistake of an engineer, a connection of a building can be omitted.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά και να εκφράσω την εκτίμησή μου στον επιβλέποντα καθηγητή μου κ.. Παναγιώτη Τζέκη, για την ευκαιρία που μου έδωσε να ασχοληθώ και να μελετήσω αυτό το ιδιαίτερα ενδιαφέρον θέμα. Για τη βοήθεια, και τις πολύτιμες γνώσεις που μου προσέφερε, όπως και για τον χρόνο που αφιέρωσε καθ' όλη τη διάρκεια της διπλωματικής αλλά και κατά την διάρκεια του μεταπτυχιακού προγράμματος.

Περιεχόμενα

Κατάλογος εικόνων

Κατάλογος Σχημάτων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
QGIS	Quantum geographic information system
QuickOSM	Quick Open Street Maps
CRS	Coordinate reference system
EPSG	European Petroleum Survey Group

Ετοιμολογίες

Επίπεδο	Ένας τύπος δομής που αποθηκεύονται δεδομένα ώστε να απεικονίζουν πραγματικές γεωγραφίες σε ψηφιακή μορφή.
Πίνακας Χαρακτηριστικών	Ο πίνακας εμφανίζει πληροφορίες σχετικά με τα χαρακτηριστικά ενός επιλεγμένου επιπέδου.
Πεδίο	Η κάθε στήλη του πίνακα των χαρακτηριστικών.

Κεφάλαιο 1ο: Σκοπός της εργασίας

1.1 Εισαγωγή

Τα τελευταία χρόνια τα δίκτυα και κυρίως αυτά των οπτικών ινών βρίσκονται σε πλήρη ανάπτυξη. Μόνο στην Ελλάδα εκτελούνται έργα χιλιάδων χιλιομέτρων, κόστους εκατοντάδων εκατομμύριων [1] τα οποία θα συνεχιστούν για πολλά χρόνια ακόμα. Ενδεικτικό είναι ότι η ευρωπαϊκή ένωση, στην προσπάθειά της να αποκτήσουν όλοι πρόσβαση σε δίκτυα υπερύψηλων ταχυτήτων, έχει επιδοτήσει πολλούς παρόχους για την κατασκευή των δικτύων αυτών αλλά και νοικοκυριά/επιχειρήσεις με ευρωπαϊκά προγράμματα όπως το Superfast broadband [2]. Μεγάλο μειονέκτημα είναι ότι τα προγράμματα σχεδίασης είναι ελάχιστα και κοστίζουν ακριβά, τόσο στην αγορά τους όσο και στις συνδρομές τους.

1.2 Σκοπός

Η εργασία αυτή έχει εφαρμοσμένο χαρακτήρα και σκοπός της είναι η δημιουργία ενός μοντέλου το οποίο θα μπορεί να εκτελεί όλα τα βήματα της μελέτης και σχεδίασης ενός οπτικού δικτύου αυτόματα. Η διαδικασία της μελέτης και σχεδίασης, είναι χρονοβόρα και απαιτεί τουλάχιστον μια ομάδα μηχανικών να εργάζεται αποκλειστικά για αυτόν τον σκοπό. Συνήθως όταν η σχεδίαση γίνεται χειροκίνητα, στο ίδιο έργο μπορεί να εργάζονται από δυο έως και πέντε μηχανικοί, σχεδιάζοντας ο καθένας την δική του μελέτη. Όταν ολοκληρωθούν οι μελέτες γίνεται η σύγκριση τους, ώστε στο τέλος να γίνει η επιλογή αυτής που θεωρείται πιο κατάλληλη προς υλοποίηση. Αυτό από μόνο του είναι μειονέκτημα τόσο στους πόρους ανθρωπίνου δυναμικού που χρησιμοποιούνται όσο και στον χρόνο που δαπανάται. Αποτέλεσμα αυτού είναι να υπάρχουν μεγάλες καθυστερήσεις στην σχεδίαση και κατ' επέκταση στην κατασκευή του δικτύου. Με την χρήση των μοντέλων που έχουν δημιουργηθεί, γίνεται όλη η διαδικασία αυτόματα, με τον μηχανικό να πρέπει να πραγματοποιήσει μόνο οπτικό έλεγχο και ενδεχομένως κάποιες διορθώσεις στο τελικό σχέδιο. Τα δίκτυα εκτείνονται σε όλη την χώρα περιέχοντας μεγάλο αριθμό δεδομένων, γι' αυτό και ένα αξιόπιστο σύστημα σχεδίασης και καταγραφής είναι απαραίτητο εργαλείο.

1.3 Παρόμοια συστήματα

Κάποια από τα ευρέως χρησιμοποιούμενα λογισμικά για την χρήση δημιουργίας και καταγραφής δικτύων είναι τα Sunvision, Comsof, biarri networks και το 3gis [3-6]. Αυτήν την στιγμή δεν υπάρχει κανένα αντίστοιχο ελληνικό λογισμικό που να παρέχει λειτουργίες τόσο στην σχεδίαση όσο και στην καταγραφή δικτύων. Όλα τα προαναφερθέντα λογισμικά παρέχουν δυνατότητες σχεδίασης και καταγραφής δικτύου, θεωρώντας τις δυο λειτουργίες ως δυο διαφορετικές ¨οντότητες¨ και συνεπώς με διαφορετικές συνδρομές. Οι συνδρομές αυτές συνήθως παρέχονται ανά χρήστη και αυτό πολλαπλασιάζει το κόστος κατά πολύ. Συνοψίζοντας, στόχος της εργασίας είναι η δημιουργία ενός αυτοματοποιημένου μοντέλου, το οποίο θα εκτελεί όλα τα βήματα της διαδικασίας της σχεδίασης, συγκριτικά με τις ήδη υπάρχουσες λύσεις που υπάρχουν αυτή τη στιγμή στην αγορά.

Κεφάλαιο 2ο: Δημιουργία βάσης δεδομένων

2.1 Εισαγωγή στο λειτουργικό

Για την εργασία έχει επιλεχθεί το λογισμικό QGIS 3.18.2 [7] το οποίο είναι ένα λογισμικό γεωγραφικών πληροφοριακών συστημάτων που δημιουργήθηκε τον Μάιο του 2002. Είναι ένα εργαλείο με αμέτρητες εφαρμογές το οποίο πλέον διδάσκεται και σε πανεπιστήμια [8][9]. Μεγάλο του προτέρημα είναι ότι είναι ανοιχτού κώδικα και αναπτύσσεται συνεχώς από μία ευρεία κοινότητα επιστημόνων, με συνεχείς αναβαθμίσεις και δυνατότητα προγραμματισμού τόσο σε python όσο και σε διαγράμματα ροής.[10] Ακόμα, έχουν χρησιμοποιηθεί τα εργαλεία Grass 7.85 τα οποία ανήκουν στο βασικό τμήμα του λογισμικού και επιπρόσθετα τα plugins QuickOSM[11] και Networks. [12] Ακόμη, τα διαγράμματα ροής έχουν γίνει με την χρήση του lucidapp [13].

2.2 Σύστημα αναφοράς συντεταγμένων (CRS)

Το σύστημα αναφοράς συντεταγμένων είναι ο τρόπος με τον οποίο μεταφέρεται στον χάρτη ένα τρισδιάστατο τμήμα της γης σε προβολή δυο διαστάσεων. Κάθε περιοχή έχει και το δικό της σύστημα αναφοράς συντεταγμένων ανάλογα με το τεταρτημόριο του χάρτη στο οποίο βρίσκεται. *¨Για να αποφευχθεί η σύγχυση μεταξύ των διαφορετικών συστημάτων συντεταγμένων που χρησιμοποιούνται, κάθε σύστημα έχει συσχετιστεί με έναν μοναδικό κωδικό από τον επιστημονικό οργανισμό European Petroleum Survey Group (EPSG)¨* [14].

Ένας παράγοντας που πρέπει να ληφθεί υπόψη είναι ότι το σύστημα αναφοράς συντεταγμένων πρέπει να είναι το ίδιο για όλα τα επίπεδα, καθώς αν υπάρχουν επίπεδα με διαφορετικά συστήματα αναφοράς συντεταγμένων θα έχει ως αποτέλεσμα να μην γίνει απεικόνιση σε κάποια από τα επίπεδα. Αν και το σύστημα αναφοράς στην Ελλάδα είναι το EPSG2100, στην παρούσα εργασία έχει επιλεχθεί EPSG3857 WGS 84 / Pseudo-Mercator το οποίο είναι πλήρως συμβατό με το QuickOsm, σε αντίθεση με το EPSG2100 το οποίο πολλές φορές δεν μπορούσε να εξάγει όλα τα δεδομένα.

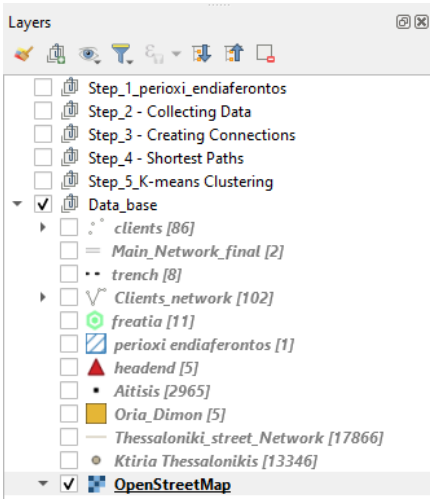
2.3 Δημιουργία βάσης δεδομένων

Στο QGIS υπάρχουν δυο ειδών επίπεδα, τα οποία είναι τα ¨μόνιμα¨, όπου αποθηκεύουν όλες τις αλλαγές του μηχανικού και βρίσκονται εκεί την επόμενη φορά που θα δουλέψει στο ¨project¨, και τα ¨προσωρινά¨ επίπεδα, τα οποία αποθηκεύονται στην προσωρινή μνήμη του συστήματος και εφόσον κλείσει το πρόγραμμα τα δεδομένα τους δεν ανακτώνται. Τα επίπεδα αυτά αποτελούνται από τρεις βασικούς τύπους γεωμετρίας, οι οποίοι είναι οι εξής: σημείων, γραμμών ή πολυγώνων. Σαν πρώτο βήμα για τη σχεδίαση του συστήματος έχει δημιουργηθεί μία βάση δεδομένων της μορφής.qgz, η οποία αποτελείται από έντεκα βασικά μόνιμα επίπεδα όπως περιγράφονται παρακάτω.

- Ένα επίπεδο της μορφής «σημείων» το οποίο έχει ονομαστεί ¨clients¨ και εμπεριέχει όλα τα σημεία τελικής σύνδεσης.
- Ένα επίπεδο της μορφής «σημείων» το οποίο έχει ονομαστεί ¨freatia¨ και εμπεριέχει όλα τα σημεία των υφιστάμενων φρεατίων.
- Ένα επίπεδο της μορφής «γραμμής» το οποίο έχει ονομαστεί ¨Main_Network_final¨ και εμπεριέχει το βασικό κορμό του δικτύου.
- Ένα επίπεδο της μορφής «γραμμής» το οποίο έχει ονομαστεί ¨trench¨ και συνδέει τα φρεάτια μεταξύ τους.

- Ένα επίπεδο της μορφής «γραμμής» το οποίο έχει ονομαστεί ¨Clients_network¨ και εμπεριέχει τις συνδέσεις από τα φρεάτια στις απολήξεις των κτιρίων.
- Ένα επίπεδο της μορφής «πολυγώνου» το οποίο έχει ονομαστεί ¨perioxi_endiaferontos¨ το οποίο χρησιμοποιείται για να δημιουργηθεί το δίκτυο εντός του πολυγώνου.
- Ένα επίπεδο της μορφής «σημείων» το οποίο έχει ονομαστεί ¨Headends¨ και εμπεριέχει όλα τα ¨Κέντρα¨ σύνδεσης με το υφιστάμενο δίκτυο.
- Ένα επίπεδο της μορφής «γραμμής» το οποίο έχει ονομαστεί ¨Thessaloniki_street_Network¨ το οποίο περιέχει όλο το οδικό δίκτυο της Θεσσαλονίκης.
- Ένα επίπεδο της μορφής «σημείων» το οποίο έχει ονομαστεί ¨Aitisis¨ το οποίο εμπεριέχει όλες τις αιτήσεις ενδιαφέροντος.
- Ένα επίπεδο της μορφής «πολυγώνου» το οποίο έχει ονομαστεί ¨Oria_dimon¨ το οποίο εμπεριέχει τα όρια των δήμων.
- Τέλος, ένα επίπεδο της μορφής «σημείων» το οποίο έχει ονομαστεί ¨ktiria_thessalonikis¨ και εμπεριέχει όλα τα κτίρια του δήμου Θεσσαλονίκης.

Σε όλα τα επίπεδα έχουν δημιουργηθεί ¨styles layer¨ (βλ. κεφάλαιο 5), τα οποία βοηθούν στην απεικόνιση των γεωμετριών στον χάρτη με τέτοιο τρόπο ώστε να παρέχουν μια εύκολη ¨ανάγνωση¨ του δικτύου βλέποντας το. Σε αρκετά από αυτά έχουν δημιουργηθεί εικονικά πεδία (virtual fields) (βλ. κεφάλαιο 5) τα οποία δίνουν τη δυνατότητα δυναμικής αναπαράστασης των αποτελεσμάτων, με τέτοιον τρόπο ώστε όταν λαμβάνει χώρα μια αλλαγή σε ένα πεδίο, να αλλάζει και η άμεσα συνδεδεμένη πληροφορία με αυτό.
Η σχεδίαση των μοντέλων έχει πραγματοποιηθεί με την βοήθεια του εργαλείου modeler και την χρήση αλγορίθμων. Το εργαλείο modeler δίνει την δυνατότητα προγραμματισμού σε python αλλά και σε διαγράμματα ροής. Έχουν χρησιμοποιηθεί και οι δυο τρόποι κατά περίπτωση.
Στόχος αυτού του κεφαλαίου είναι η γνωριμία με το λειτουργικό σύστημα Qgis στο οποίο θα αναπτυχθούν τα μοντέλα όπως και η γνωριμία με κάποιες από τις βασικές έννοιες του λογισμικού που θα αναφερθούν παρακάτω. Επίσης, γίνεται επεξήγηση των βασικών επιπέδων που έχουν δημιουργηθεί, τόσο στις λειτουργίες τους όσο και στις γεωμετρίες τους.



Εικόνα 2.3-1 Βάση δεδομένων

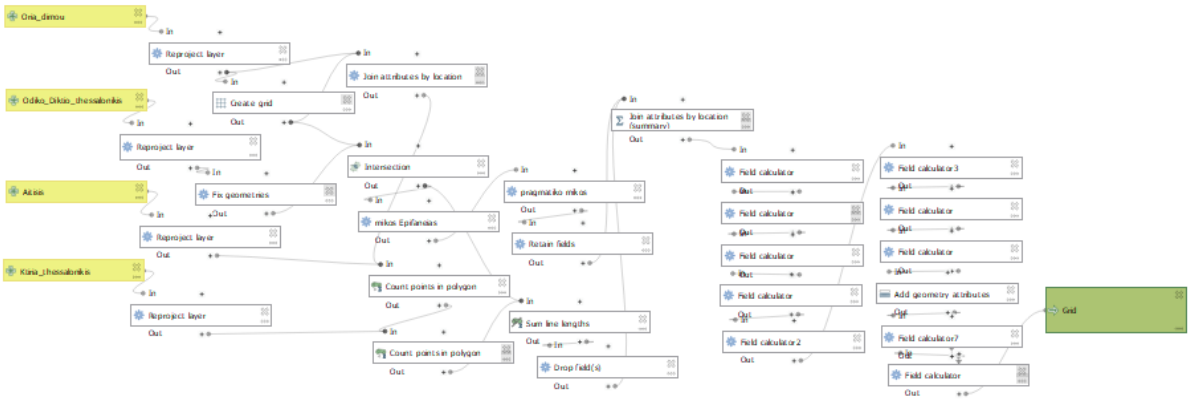
Κεφάλαιο 3ο: Επιλογή περιοχής σχεδίασης

3.1 Περιοχή ενδιαφέροντος

Πριν ξεκινήσει η σχεδίαση ενός δικτύου, το πρώτο βήμα είναι να γίνει μελέτη της περιοχής η οποία είναι πιο συμφέρουσα για να αναπτυχθεί το δίκτυο αυτό. Θα πρέπει να εξεταστούν παράγοντες όπως η ζήτηση, οι δυνητικοί χρήστες που μπορεί να προκύψουν, το κόστος του έργου και το κέρδος που θα αποφέρει. Για την επιλογή της περιοχής ανάπτυξης του δικτύου, έχει δημιουργηθεί ένα μοντέλο το οποίο εξάγει ένα πλέγμα της πόλης σε επιμέρους ¨τετράγωνα¨, (εφεξής θα ονομάζονται κυψέλες), τα οποία περιέχουν πληροφορίες, όπως το κόστος, το πιθανό κέρδος, την ζήτηση εντός της κάθε περιοχής κ.α. (Πλήρης κώδικας – Παράρτημα Α)

3.2 Πως λειτουργεί

Ο χρήστης θα πρέπει να δώσει τέσσερις εισόδους οι οποίες είναι, τα όρια των δήμων, το οδικό δίκτυο της πόλης, οι αιτήσεις εκδήλωσης ενδιαφέροντος που έχουν πραγματοποιηθεί και τέλος όλα τα κτίρια της πόλης.

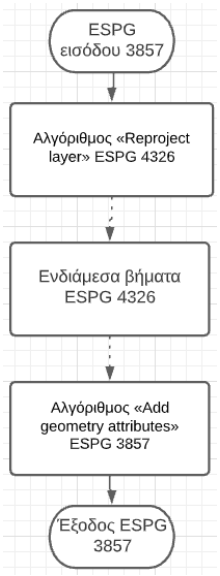


Εικόνα 3.2-1 Μοντέλο περιοχής ενδιαφέροντος

Αφού ο χρήστης δώσει τις εισόδους και ¨τρέξει¨ ο κώδικας, πραγματοποιούνται οι παρακάτω ενέργειες. Αρχικά για κάθε είσοδο καλείται ο αλγόριθμος «Reproject layer» ο οποίος θα αλλάξει το EPSG (βλ. ενότητα 2.2) όλων των επιπέδων, ο αλγόριθμος δημιουργεί ένα νέο διανυσματικό επίπεδο, ίδιων χαρακτηριστικών με αυτά του επιπέδου εισαγωγής, αλλά με γεωμετρίες που αναπαρίστανται σε νέο προβολικό σύστημα. Όπως φαίνεται και στο σχήμα (3.2-1) η είσοδος του επιπέδου έχει προβολικό σύστημα EPSG3857 και μετά την χρήση του αλγορίθμου γίνεται η αλλαγή σε EPSG4326. Έπειτα, διενεργούνται όλα τα βήματα των υπολοίπων αλγορίθμων και πριν την εξαγωγή του νέου επιπέδου, ξανά αλλάζει το προβολικό σύστημα σε EPSG 3857. Αυτή την φορά αλλάζει από τον αλγόριθμο «Add geometry attributes» του οποίου η χρησιμότητα θα αναλυθεί παρακάτω. Ο λόγος που δεν ξανά καλείται ο αλγόριθμος «Reproject layer» είναι γιατί ο αλγόριθμος «Add geometry attributes» δίνει την

δυνατότητα αυτής της αλλαγής, καθώς μπορεί να εξάγει τα αποτελέσματα στο EPSG του βασικού επιπέδου ή όποιου άλλου συστήματος του ζητηθεί, συνεπώς θα ήταν περιττό να καλεστεί ξανά ο αλγόριθμος «Reproject layer». Ο λόγος της μετατροπής αυτής προέρχεται από την παρατήρηση ότι πολλές τιμές στο πεδίο που περιέχει το συνολικό μήκος του οδικού δικτύου (βλ. σελ. 7), με το πέρας του αλγορίθμου «Sum line length», επέστρεφαν κάποια πεδία ως κενές τιμές ¨null¨, δηλαδή δεν περιείχαν το συνολικό μήκος του δικτύου της κάθε κυψέλης με αποτέλεσμα να μην μπορεί να υπολογιστεί το κόστος κατασκευής. Με την μετατροπή του συστήματος συντεταγμένων σε EPSG 4326 αυτό διορθώνεται.

Καθώς όμως όλα τα βασικά επίπεδα είναι δημιουργημένα στο σύστημα συντεταγμένων EPSG 3857, πριν το τέλος του μοντέλου και αφού έχουν γίνει όλοι οι απαραίτητοι υπολογισμοί, πρέπει να αναστραφεί στο αρχικό σύστημα συντεταγμένων.

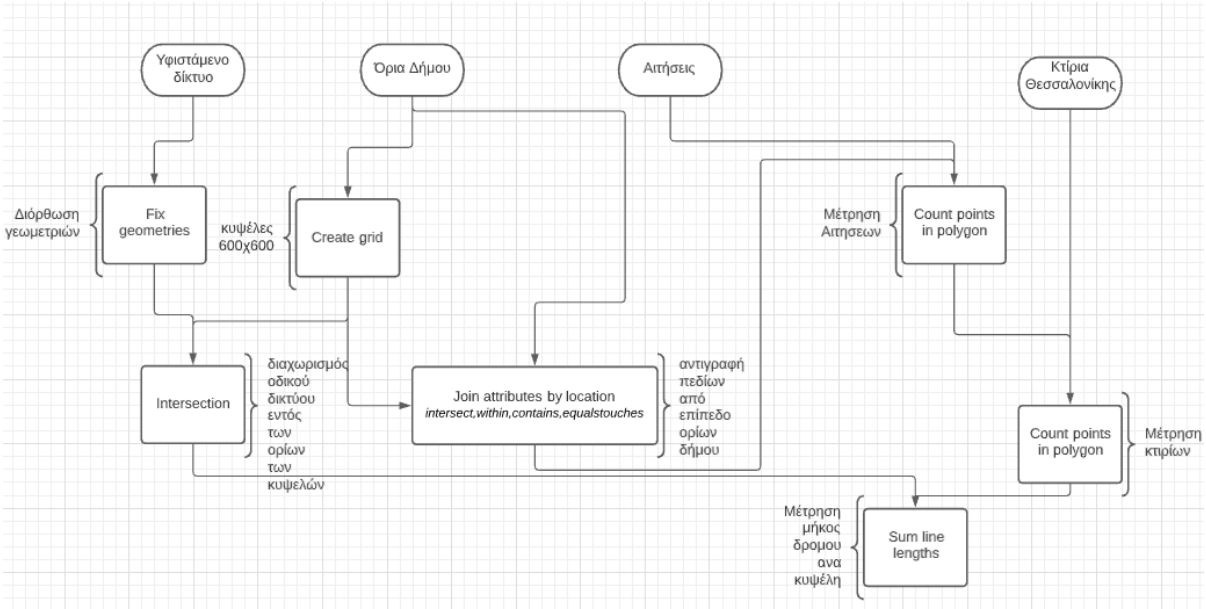


Σχήμα 3.2-1 Διάγραμμα ροής αλλαγής EPSG

Ο επόμενος αλγόριθμος που καλείται είναι ο «Create grid» ο οποίος θα δημιουργήσει ένα πλέγμα στον χάρτη. Σκοπός είναι η τμηματοποίηση του έτσι ώστε να μπορεί να γίνει η σύγκριση των υποπεριοχών. Ο αλγόριθμος κατά την έξοδο του δημιουργεί ένα διανυσματικό επίπεδο με ένα πλέγμα που καλύπτει μια δεδομένη έκταση όπου στην περίπτωση του μοντέλου, είναι εντός του επιπέδου που περιέχει τα όρια του κάθε δήμου. Δίνεται η δυνατότητα να εισαχθούν επίπεδα διαφορετικών τύπων. Στην περίπτωση του μοντέλου είναι τύπου πολυγώνων. Η έκταση του πλέγματος και οι τιμές απόστασης καθορίζονται δηλώνοντας τις οριζόντιες και κάθετες αποστάσεις στον κώδικα. Για το πλέγμα αυτό, έχουν δοθεί αποστάσεις 600x600 μέτρα το οποίο είναι ένα τυπικό μέγεθος κυψελών. Το μέγεθος των κυψελών μπορεί να παραμετροποιηθεί από τον προγραμματιστή ανάλογα με τις ανάγκες του χρήστη.

Αρκετοί αλγόριθμοι κατά την εκτέλεση τους διαγράφουν ή αντικαθιστούν κρίσιμα πεδία που προέρχονται από το βασικό επίπεδο. Αποτέλεσμα αυτού, να χάνονται πληροφορίες που θα χρειαστούν μετέπειτα. Σε αυτό το στάδιο έχει γίνει η τμηματοποίηση των κυψελών αλλά δεν υπάρχει η πληροφορία σε ποιον δήμο ανήκει η κάθε μια. Αυτήν η πληροφορία θα πρέπει να εισαχθεί από το βασικό επίπεδο των ορίων του δήμου εκ νέου. Για τον λόγο αυτό καλείται ο αλγόριθμος «Join attributes by location». Ο αλγόριθμος παίρνει ένα επίπεδο εισόδου και δημιουργεί ένα πιστό αντίγραφο του, προσθέτοντας

πεδία που λαμβάνονται από ένα άλλο επίπεδο συνένωσης. Αυτό δίνει την δυνατότητα να περάσουν στο νέο επίπεδο πληροφορίες που έχουν χαθεί στην πορεία, αντλώντας τις από το βασικό επίπεδο. Ο αλγόριθμος θα δεχθεί ως βασικό επίπεδο εισόδου την έξοδο του αλγορίθμου «Create grid» και σαν επίπεδο συνένωσης αυτό των ορίων του δήμου. Τα κριτήρια για την προσθήκη πεδίων στο νέο επίπεδο καθορίζονται με βάση την γεωγραφική τοποθεσία των στοιχείων και τις εντολές που έχουν δοθεί στον κώδικα, intersect, within, contains, equals touches, μεταφέροντας σε κάθε κυψέλη την πληροφορία του δήμου στον οποίο ανήκει το κάθε στοιχείο. Ο λόγος που επιλέγονται όλες οι εντολές αντιγραφής των στοιχείων στον κώδικα είναι για να μην χαθεί κανένα στοιχείο. Για παράδειγμα, μπορεί ένα στοιχείο να μην είναι μέσα σε μια περιοχή (within) αλλά επάνω στα όρια της (intersect). Αν δεν επιλεγεί η εντολή ¨intersect¨, το στοιχείο δεν θα υπάρχει στο επίπεδο εξόδου. Με το πέρας του αλγορίθμου εξάγεται ένα νέο επίπεδο το οποίο περιέχει όλες τις πληροφορίες σε νέα πεδία.



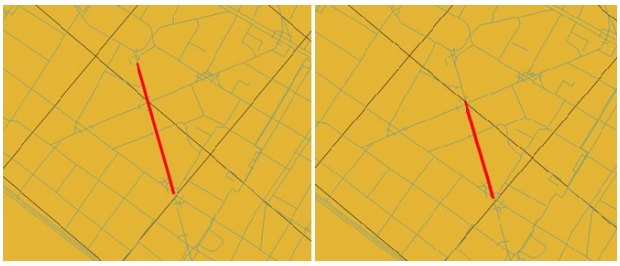
Σχήμα 3.2-2 Διάγραμμα ροής επιλογής περιοχής ενδιαφέροντος

Στη συνέχεια καλείται ο αλγόριθμος «count point in polygon», ο οποίος παίρνει ως εισόδους ένα επίπεδο σημείων και ένα επίπεδο πολυγώνων, με σκοπό να μετρήσει τον συνολικό αριθμό των σημείων μέσα σε κάθε πολύγωνο. Κατά την έξοδο δημιουργείται ένα νέο επίπεδο πολυγώνων, ίδιο ακριβώς με το επίπεδο εισόδου, αλλά περιέχει ένα επιπλέον πεδίο με τον αριθμό των σημείων που έχει μετρήσει. Ο αλγόριθμος αυτός καλείται δυο φορές στο μοντέλο. Την πρώτη αναλαμβάνει να μετρήσει τις αιτήσεις ενδιαφέροντος, έχοντας ως βασική είσοδο πολυγώνων την έξοδο του αλγορίθμου «Join attributes by location» και συγκρίσιμο επίπεδο αυτό που περιέχει τις αιτήσεις ενδιαφέροντος. Την δεύτερη φορά που καλείται είναι για να μετρήσει όλα τα κτίρια που υπάρχουν εντός της κάθε κυψέλης έχοντας ως βασική είσοδο την έξοδο του πρώτου αλγορίθμου «count point in polygon» και σαν συγκρίσιμο επίπεδο αυτό που περιέχει όλα τα κτίρια της περιοχής.

Επόμενο βήμα του μοντέλου είναι να γίνει ο προληπτικός έλεγχος του επιπέδου του οδικού δικτύου μέσω του αλγορίθμου «Fix geometries». Τα δεδομένα του οδικού δικτύου κατά την λήψη τους (βλ ενότητα 4.1), ενδέχεται να έχουν εσφαλμένες κορυφές στις γεωμετρίες τους, με συνηθέστερα λάθη τα σημεία εκκίνησης ή τερματισμού της γραμμής. Ο αλγόριθμος αυτός αναλαμβάνει να διορθώσει αυτά

τα πιθανά λάθη και την ίδια στιγμή να αφήσει ανεπηρέαστες τις γεωμετρίες που είναι σωστές. Σε αυτό το σημείο το επίπεδο του οδικού δικτύου δεν έχει υποστεί ακόμα κάποια επεξεργασία. Έτσι, ένας δρόμος μπορεί να διασχίζει διαφορετικές κυψέλες του πλέγματος, με αποτέλεσμα να δίνει εσφαλμένα δεδομένα σχετικά με το συνολικό μήκος οδικού δικτύου εντός της κάθε κυψέλης.

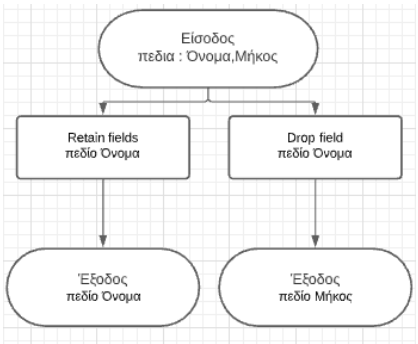
Για να μπορέσει να γίνει ο διαχωρισμός του οδικού δικτύου εντός των κυψελών, καλείται ο αλγόριθμος «Intersection», ο οποίος, συγκρίνει μεταξύ δυο επιπέδων και εξάγει τις γεωμετρίες όπου υπάρχει αλληλοκάλυψη. Έτσι δηλώνοντας ως βασικό επίπεδο, το επίπεδο των δρόμων και σαν επίπεδο σύγκρισης αυτό που περιέχει τις κυψέλες, ο αλγόριθμος ελέγχει τα σημεία του οδικού δικτύου και τα χωρίζει με τέτοιον τρόπο, ώστε κάθε γραμμή να φτάνει μέχρι τα όρια των κυψελών. Με αυτόν τον τρόπο έχει επιτευχθεί ο διαχωρισμός του οδικού δικτύου εντός της κάθε κυψέλης και τα μετρούμενα μεγέθη να είναι σωστά.



Εικόνα 3.2-2 Αριστερή εικόνα ο δρόμος διασχίζει διαφορετικές κυψέλες του πλέγματος. Δεξιά εικόνα ο δρόμος χωρισμένος ανά κυψέλη

Συνεχίζοντας, και εφόσον πλέον το οδικό δίκτυο έχει τμηματοποιηθεί σωστά εντός της κάθε κυψέλης, θα πρέπει να υπολογιστεί το συνολικό μήκος του οδικού δικτύου εντός των κυψελών. Αυτό επιτυγχάνεται με τον αλγόριθμο «Sum line lengths» ο οποίος δέχεται ως εισόδους ένα επίπεδο γραμμής, το οποίο θα είναι η έξοδος του αλγορίθμου «Intersection» και ένα επίπεδο πολυγώνου το οποίο είναι η έξοδος του δεύτερου κατά σειρά αλγορίθμου «count point in polygon». Με αυτόν τον τρόπο θα γίνει η μέτρηση του συνολικού μήκους των γραμμών εντός της κάθε κυψέλης. Πλέον υπάρχει ένα πεδίο κατά την έξοδο του νέου επιπέδου, το οποίο έχει την ετικέτα ¨mikos¨ και περιέχει το πραγματικό μήκος του οδικού δικτύου εντός της κάθε κυψέλης.

Έπειτα καλούνται οι αλγόριθμοι «Retain fields» και «Drop Fields» με σκοπό την διατήρηση ή την απόρριψη πεδίων ενδιαφέροντος.



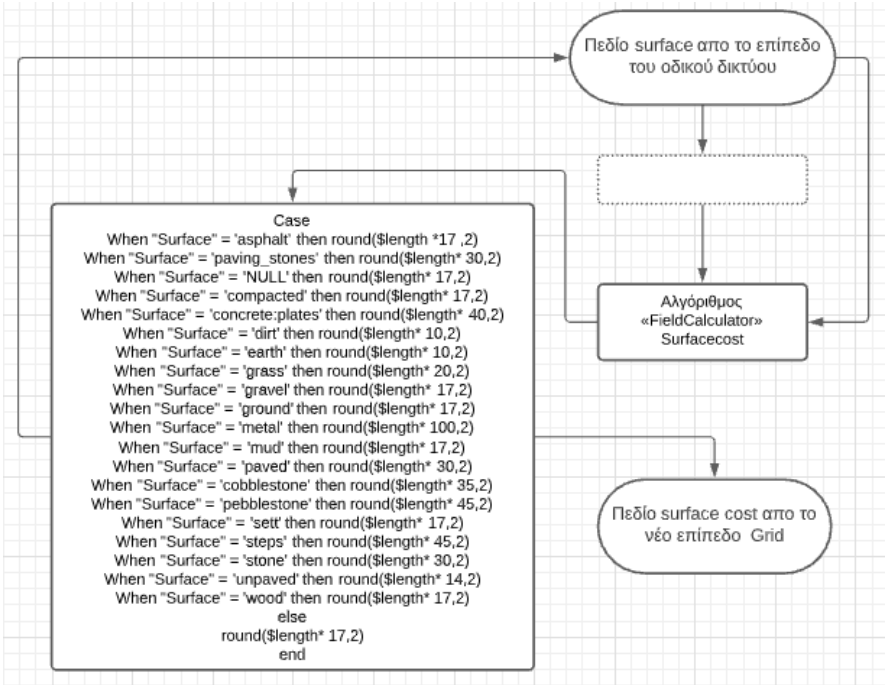
Σχήμα 3.2-3 Αλγόριθμοι Retain fields & Drop fields

Στην συνέχεια, με την πολλαπλή χρήση του αλγορίθμου «Field Calculator» θα υπολογισθούν και θα εισαχθούν νέα πεδία, τα οποία είναι σημαντικά για την αξιολόγηση της περιοχής πριν την κατασκευή του δικτύου. Ο αλγόριθμος δημιουργεί ένα νέο διανυσματικό επίπεδο με τα ίδια χαρακτηριστικά του επιπέδου εισαγωγής. Οι τιμές του πεδίου υπολογίζονται χρησιμοποιώντας μια έκφραση (if, case, sum κτλ.) και αντικαθιστούν ή προσθέτουν ένα νέο πεδίο. Η ανάλυση τους γίνεται στις ενότητες (3.2.1 - 3.2.7).

Τέλος, ο αλγόριθμος «Add geometry attributes» δημιουργεί ένα νέο διανυσματικό επίπεδο διατηρώντας όλα τα στοιχεία του εισερχομένου επιπέδου, αλλά με πρόσθετα πεδία που περιέχουν χαρακτηριστικά γεωμετρικών μετρήσεων, όπως η περίμετρος της κάθε κυψέλης. Ανάλογα με τον τύπο γεωμετρίας του διανυσματικού επιπέδου (σημείων, γραμμών ή πολυγώνων), τα πεδία που προστίθενται στον πίνακα θα είναι διαφορετικά. Ο αλγόριθμος καλείται ώστε να εισάγει την πληροφορία της περιμέτρου των κυψελών, σε ένα νέο πεδίο στο τέλος του μοντέλου και παράλληλα να κάνει την αλλαγή του EPSG.

3.2.1 Κόστος βάσει της επιφάνειας

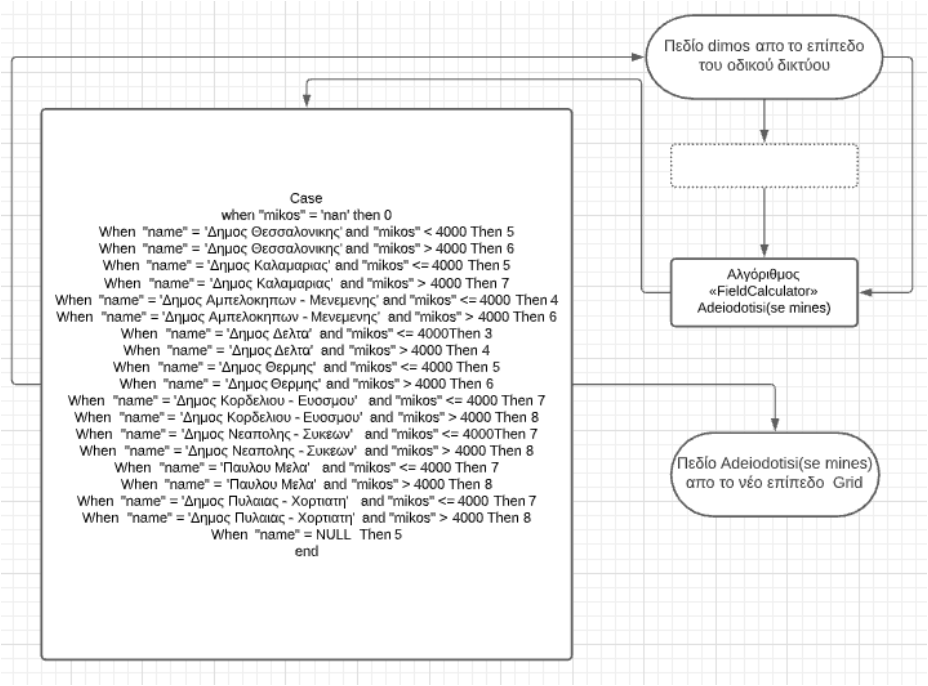
Κατά την κατασκευή ενός δικτύου, μείζων σημασία αποτελούν τα υλικά από τα οποία είναι κατασκευασμένος ο δρόμος (άσφαλτος, πέτρα, πλακόστρωτο κτλ.), καθώς είναι ένας παράγοντας που μπορεί να αυξήσει μέχρι και τέσσερις φορές το κόστος. Ανάλογα πάντα με τον εκάστοτε εργολάβο. Για τον λόγο αυτό δημιουργείται ένα νέο πεδίο, το οποίο ονομάζεται ¨Surface cost¨, μέσω του οποίου υπολογίζεται το συνολικό κόστος εκσκαφής όλων των δρόμων εντός της κυψέλης, βάσει του μήκους και της επιφάνειας του δρόμου. Το πεδίο αυτό προκύπτει από το μήκος του δρόμου σε σύγκριση με το πεδίο ¨surface¨ του επιπέδου του υφιστάμενου δικτύου, όπου εμπεριέχει τις πληροφορίες σχετικά με τα υλικά από τα οποία είναι κατασκευασμένος ο δρόμος. Από την σχέση που έχει δοθεί στο κώδικα, υπολογίζεται το κόστος εκσκαφής του κάθε δρόμου και εξάγεται το σύνολο σε ένα νέο πεδίο.



Σχήμα 3.2-4 Κόστος βάσει της επιφάνειας του δρόμου

3.2.2 Πιθανός χρόνος αδειοδότησης

Κάθε δήμος έχει τον δικό του χρόνο στο να αποδίδει την νόμιμη άδεια εκσκαφής βάσει του προσωπικού που διαθέτει και το μέγεθος του έργου. Στο παρακάτω πεδίο δημιουργείται μια σχέση μεταξύ των δυο προαναφερθέντων δεδομένων. Γνωρίζοντας εμπειρικά τους χρόνους που απαιτούνται και θέτοντας ένα όριο τεσσάρων χιλιομέτρων μεταξύ των μελετών κατηγοριοποιώντας, σε ¨μικρή¨ μελέτη και ¨μεγάλη¨, θα υπολογισθούν οι μήνες αναμονής για την έκδοση αδείας. Αποτελεί ένα σημαντικό πεδίο, καθώς οποιαδήποτε καθυστέρηση έχει ως επακόλουθο την αργοπορία της ολοκλήρωσης του έργου και κατ' επέκταση δημιουργεί εκ νέου καθυστερήσεις στην παραγωγή εσόδων.



Σχήμα 3.2-5 Χρόνος αναμονής για αδειοδότηση

3.2.3 Δυνητικοί χρήστες

Με τον ίδιο τρόπο όπως και προηγουμένως, υπολογίζονται οι δυνητικοί χρήστες σε σχέση με την περιοχή στην οποία ανήκει κάθε κτίριο. Για παράδειγμα ο δήμος Θεσσαλονίκης συγκριτικά με τον δήμο καλαμαριάς αποτελείται από μεγαλύτερα κτίρια, με περισσότερα διαμερίσματα και είναι πιο πυκνοκατοικημένος. Συνεπώς, υπάρχουν περισσότεροι δυνητικοί χρήστες. Έτσι, αν το στοιχείο ανήκει στον δήμο Θεσσαλονίκης επιστρέφει την τιμή των δώδεκα πιθανών διαμερισμάτων ενώ αν ανήκει στον δήμο Καλαμαριάς επιστρέφει την τιμή των έξι πιθανών διαμερισμάτων. Θα μπορούσε να εισαχθεί και η παράμετρος των υποπεριοχών (κέντρο, τούμπα, Χαριλάου κτλπ.), αλλά για να γίνει αυτό θα πρέπει να δημιουργηθεί μία ακόμα βάση δεδομένων με τις υποπεριοχές και να εισαχθούν τα νέα δεδομένα χειροκίνητα.

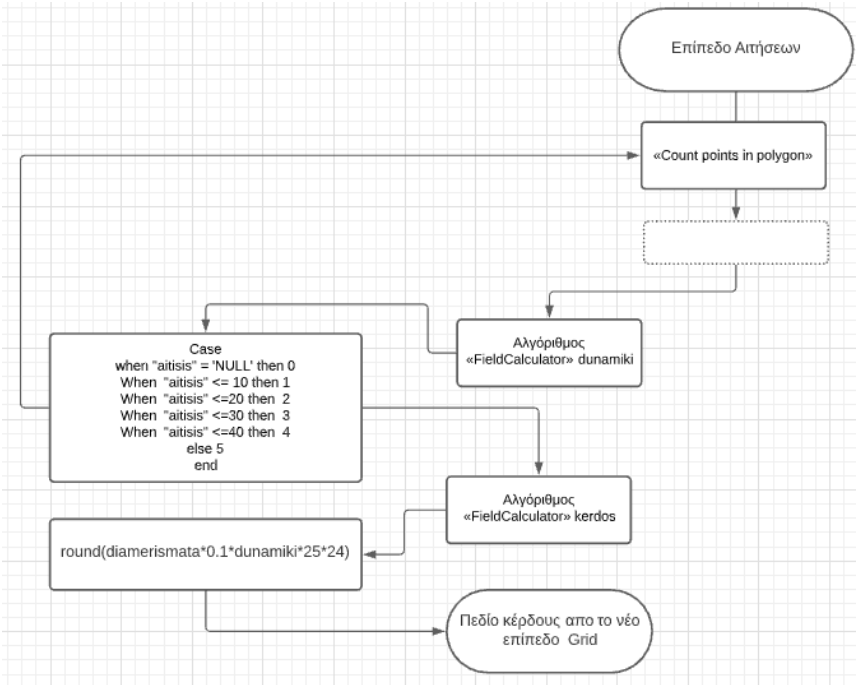
3.2.4 Τέλη διέλευσης και εγγυητικές τραπέζης

Εκτός από τα βασικά άρθρα υπολογισμού των τελών χρήσης και εγγυητικών επιστολών, ο κάθε δήμος θέτει και προαιρετικά κριτήρια για την έκδοση αδειών. Για αυτόν τον σκοπό έχουν δημιουργηθεί δυο νέα πεδία, τα οποία υπολογίζουν τις εγγυητικές τραπέζης καλής εκτέλεσης εργασιών και τα τέλη διέλευσης χρήσης του οδικού δικτύου. Αυτά αντιστοιχούν στο κόστος το οποίο πρέπει να καταβληθεί από τον πάροχο στον εκάστοτε δήμο, ώστε να του επιτραπεί η χρήση του οδικού δικτύου για έργα υποδομής. Πρακτικά αυτό σημαίνει πως για το ίδιο μήκος έργων μεταξύ δυο δήμων το κόστος αυτό μπορεί να διαφέρει.

3.2.5 Κέρδος

Για τον υπολογισμό του μέγιστου κέρδους έχει δημιουργηθεί ακόμα μια μεταβλητή η οποία δείχνει την δυναμική ενδιαφέροντος που παρουσιάζει μια περιοχή. Η μεταβλητή αυτή υπολογίζεται από τις ήδη υπάρχουσες αιτήσεις οι οποίες βρίσκονται εντός της κάθε κυψέλης, με κλίμακα βαθμονόμησης από το 1 μέχρι το 5. Έτσι, για παράδειγμα, αν σε μια περιοχή υπάρχουν πάνω από 30 αιτήσεις βαθμονομείται με τον αριθμό 3, αν αυξηθεί η ζήτηση στην περιοχή η δυναμική της θα μεγαλώνει, συνεπώς αυξάνει τον συντελεστή και κατ' επέκταση το μέγιστο κέρδος.

Για το μέγιστο πιθανό κέρδος έχει δημιουργηθεί μια σχέση η οποία υπολογίζει σαν δυνητικούς χρήστες το 10% του συνόλου των διαμερισμάτων επι τον συντελεστή δυναμικής, πολλαπλασιαζόμενα με μια μεσαίου κόστους σύνδεση της τάξεως των 25 ευρώ επί του συμβολαίου των 24 μηνών



Σχήμα 3.2-6 Διάγραμμα ροής σχέση κέρδους

3.2.6 Κόστος κατασκευής

Το κόστος κατασκευής αντιστοιχεί στο άθροισμα όλων των παραπάνω, τα οποία είναι:

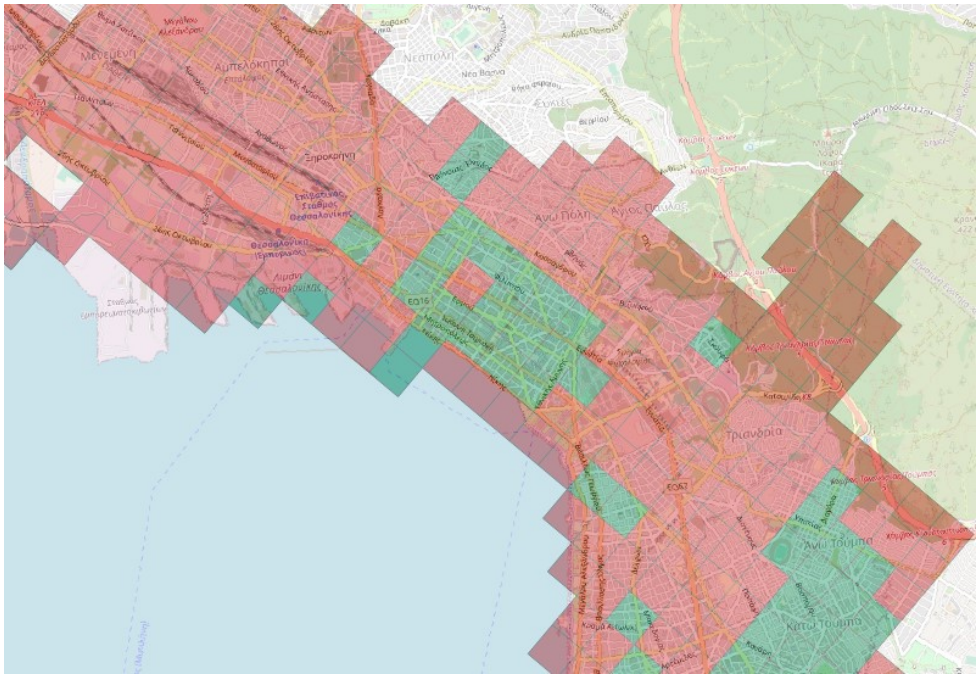
το κόστος εμφύσησης καλωδίων (που υπολογίζεται σε ένα ευρώ το μέτρο), στις χωματουργικές εργασίες βασιζόμενες στην επιφάνεια του δρόμου, στα τέλη διέλευσης που καταβάλλονται στον δήμο και στις εγγυητικές τραπέζης. Οι εγγυητικές τραπέζης είναι ένα ποσό το οποίο μετά από ένα χρονικό διάστημα (συνήθως 1 έτος) και εφόσον οι αποκαταστάσεις των δρόμων έχουν γίνει σωστά επιστρέφεται στον πάροχο, παρόλα αυτά είναι ένα κόστος που πρέπει να υπολογιστεί εξ αρχής στον προϋπολογισμό γιατί θα πρέπει να καταβληθεί εκ των προτέρων.

3.2.7 Σχέση κόστους /κέρδους

Αφού έχει υπολογιστεί το συνολικό κόστος της κατασκευής και το πιθανό κέρδος που θα αποφέρει η επένδυση, δημιουργείται μια σχέση κόστους/κέρδους. Σε αυτό το πεδίο αφαιρείται από το μέγιστο κέρδος το κόστος κατασκευής. Σκοπός είναι να μπορεί να ελεγχθεί αν μετά το πέρας των δυο ετών υπάρχει αρνητικό ισοζύγιο ή αν έχει γίνει απόσβεση του έργου.

3.3 Απεικόνιση

Ο χρήστης μπορεί να αναπαριστά τα αποτελέσματα βάσει του κριτηρίου που θεωρεί πιο σημαντικό (κόστος, κέρδος, μήκος δρόμου κτλ.). Παρακάτω γίνεται η απεικόνιση βάσει του πεδίου κόστους/ κέρδους με τον κανόνα όλες οι αρνητικές τιμές να απεικονίζονται με κόκκινο και όλες οι θετικές με πράσινο χρώμα.



Εικόνα 3.3-1 Χάρτης απεικόνισης κόστους κέρδους

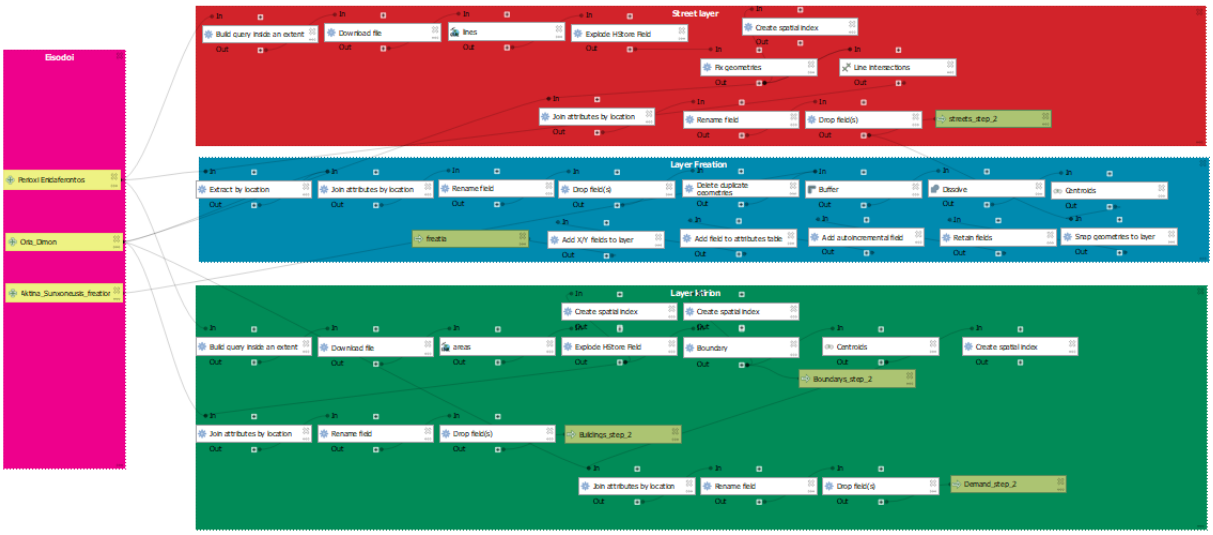
Μπορεί ο χρήστης να αλλάξει τις κλίμακες και να βάλει περισσότερες συνθήκες. Για παράδειγμα να απεικονίσει τις τιμές μεταξύ -10.000 με 10.000 ευρώ με το χρώμα κίτρινο θεωρώντας τες σαν ουδέτερες τιμές, δηλαδή ότι είναι ένα μικρό ποσό το οποίο δεν είναι αρκετό για να απορρίψει μια περιοχή ή αντίστοιχα δεν είναι αρκετό για να την επιλέξει.

Σε αυτό το κεφάλαιο δημιουργήθηκε ένα μοντέλο το οποίο επιτρέπει στον χρήστη να εξάγει σημαντικές πληροφορίες για την επιλογή της περιοχής στην οποία θα αναπτυχθεί το δίκτυο. Έχει την δυνατότητα να επιλέξει την περιοχή βάσει του προϋπολογισμού που διαθέτει ή ακόμα να συγκρίνει περιοχές οι οποίες φαίνεται να έχουν το ίδιο κόστος κατασκευής, αλλά λόγω δυναμικής και ζήτησης να παρουσιάζουν μεγαλύτερο κέρδος. Αποτελεί ένα σημαντικό εργαλείο καθώς όταν τα δεδομένα σύγκρισης είναι πάρα πολλά τότε η πιθανότητα λάθους και ο χρόνος που απαιτείται είναι πάρα πολύ μεγάλος. Με την χρήση του μοντέλου, ανεξαρτήτως μεγέθους των δεδομένων, τα αποτελέσματα εξάγονται σε λίγα λεπτά και δίνεται η δυνατότητα στον χρήστη να κάνει την σύγκριση πολύ γρήγορα.

Κεφάλαιο 4ο: Σχεδίαση δικτύου

4.1 Συλλογή δεδομένων

Τα βασικά βήματα που ακολουθούνται κατά τη σχεδίαση ενός δικτύου είναι, η σχεδίαση από τον μηχανικό στον υπολογιστή, ο επιτόπιος έλεγχος από τον μηχανικό δικτύου, ο οποίος και θα υποδείξει τις απαραίτητες αλλαγές/διορθώσεις και τέλος η επανασχεδίαση και υλοποίηση του δικτύου. Στο κεφάλαιο αυτό θα γίνει η ανάλυση των βημάτων που ακολουθούνται κατά το πρώτο σκέλος, το οποίο αφορά στη σχεδίαση του δικτύου από τον μηχανικό στον υπολογιστή.
Πρέπει να αναφερθεί ότι παρ΄όλο που όλα τα βήματα εκτελούνται αυτόματα, η μέθοδος αυτή θεωρείται ως ¨ημιαυτόματη¨, καθώς στο τέλος απαιτείται η επέμβαση του μηχανικού χειροκίνητα, ώστε να εντοπιστούν και να διορθωθούν πιθανά λάθη. Το πρώτο πράγμα που πρέπει να γίνει πριν ξεκινήσει η σχεδίαση, είναι να δημιουργηθούν τα δεδομένα εκείνα που θα χρησιμοποιηθούν ως είσοδοι για να παραχθεί το τελικό σχέδιο του δικτύου.
Κάποια από τα βασικά δεδομένα είναι το οδικό δίκτυο και οι κτιριακές υποδομές της κάθε πόλης. Η δημιουργία αυτών των δεδομένων μπορεί να γίνει με δυο τρόπους. Είτε με τη δημιουργία της βάσης δεδομένων με χειροκίνητη εισαγωγή από τον μηχανικό, είτε με τη χρήση έτοιμων βάσεων δεδομένων, όπως τους ανοιχτούς χάρτες της Google. Με τη βοήθεια του OSM plug-in και την χρήση ενός API Key, δίνεται η δυνατότητα να γίνει λήψη όλων αυτών των δεδομένων αυτόματα. Ένα μειονέκτημα στην αυτόματη μέθοδο είναι ότι η βάση δεδομένων παρουσιάζει ελλείψεις όσον αφορά τις κτιριακές υποδομές της Θεσσαλονίκης, κάτι που δεν ισχύει για παράδειγμα στην πόλη της Αθήνας. Το παρακάτω μοντέλο έχει δημιουργηθεί με σκοπό την αυτόματη λήψη των δεδομένων και ονομάζεται «Collecting_data». (Πλήρης κώδικας – Παράρτημα Β).



Εικόνα 4.1-1 Μοντέλο συλλογής δεδομένων

Εκτελώντας το μοντέλο, θα ζητηθεί από τον χρήστη να ορίσει αν θέλει να υπάρξει συγχώνευση φρεατίων (βλ. ενότητα 5.1.4), η περιοχή ενδιαφέροντος σχεδίασης του δικτύου, η οποία ορίζεται εντός του πολυγώνου που έχει δημιουργηθεί στο βασικό επίπεδο «Perioxi_endiaferontos» (βλ. ενότητα 2.3) και τα όρια των δήμων, τα οποία καθώς δεν υπάρχουν σαν πληροφορία στη βάση δεδομένων της Google και θα εισαχθούν από το επίπεδο που έχει ήδη δημιουργηθεί για αυτόν τον σκοπό. Το μοντέλο που θα ¨τρέξει¨ στο προσκήνιο θα δεχθεί τις τρεις εισόδους και αφού περατωθούν όλα τα βήματα που ακολουθούν, θα εξάγει πέντε νέες εξόδους, που περιέχουν πληροφορίες για τα φρεάτια, τους δρόμους και τα κτίρια εντός της περιοχής ενδιαφέροντος.

4.1.1 Ανάλυση μοντέλου - Οδικό δίκτυο

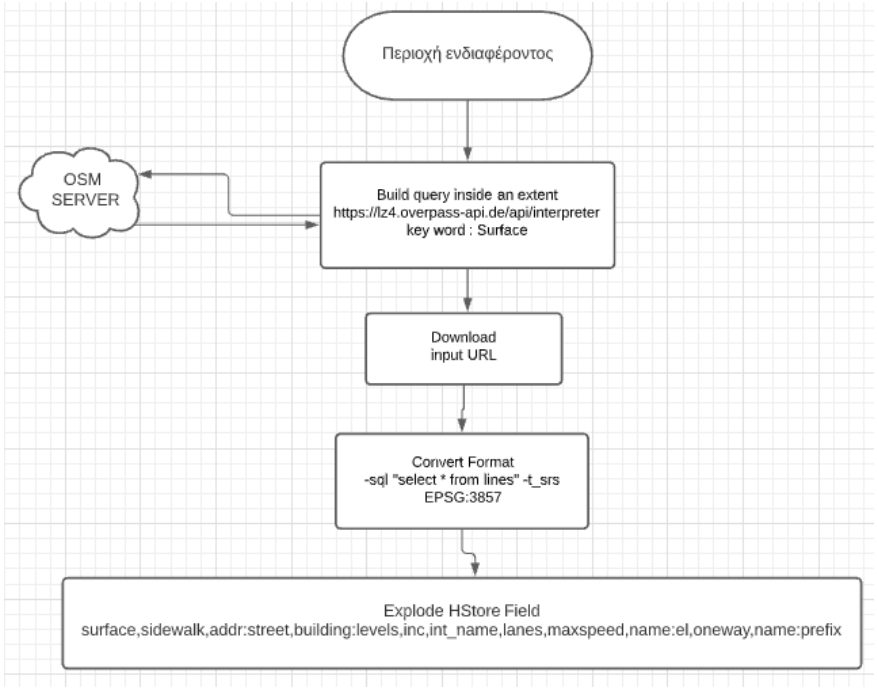
Ως είσοδοι θα χρησιμοποιηθούν δυο από τα βασικά επίπεδα, με τύπο γεωμετρίας πολυγώνου, τα οποία είναι η περιοχή ενδιαφέροντος και τα όρια των δήμων. Το πολύγωνο που αφορά την περιοχή ενδιαφέροντος, δηλώνεται ως βασική είσοδος του αλγορίθμου «Build query inside an extent». Έτσι, το αίτημα που θα γίνει προς τον server για την λήψη των δεδομένων, αφορά μόνο το πλαίσιο εντός του πολυγώνου αυτού.

Ενώ τα όρια των δήμων καλούνται και πάλι μόνο για να εισαχθεί η πληροφορία του δήμου στον οποίο ανήκει το κάθε στοιχείο και θα προστεθεί σαν ένα νέο πεδίο κατά την έξοδο του επιπέδου. Αρχικά, δημιουργείται αίτημα μέσω του αλγορίθμου «Build query inside an extent» για σύνδεση με τον server του OSM, βάσει του οδηγού του OSM [14]. Με την λέξη κλειδί ¨Surface¨, ζητείται από τον server να αποστείλει όλα τα δεδομένα σχετικά με το οδικό δίκτυο αλλά και τις πληροφορίες δομής/υλικών κατασκευής της επιφάνειας των δρόμων. Έπειτα, στην έξοδο του αιτήματος αυτού, προστίθεται ο αλγόριθμος «download», ο οποίος είναι υπεύθυνος για την λήψη των δεδομένων από τον server.

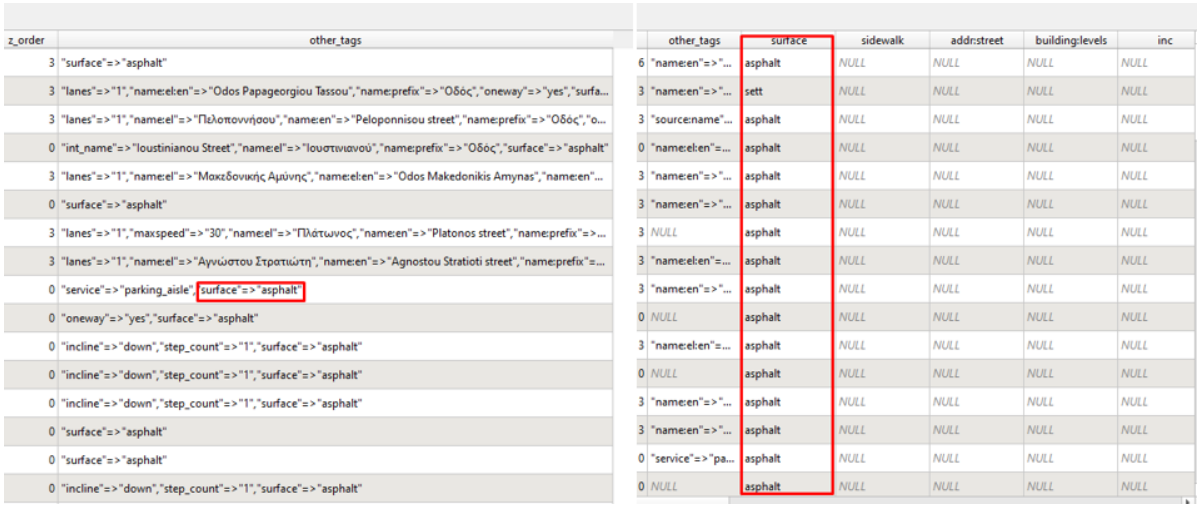
Τα δεδομένα είναι της μορφής sql και θα πρέπει να γίνει μεταφορά τους στον χάρτη βάσει του προβολικού συστήματος που χρησιμοποιείται ώστε να μπορεί να γίνει η απεικόνιση τους. Αυτό επιτυγχάνεται καλώντας τον αλγόριθμο «Convert format» στον οποίο δηλώνεται ότι τα δεδομένα που περιμένει να λάβει είναι της μορφής γραμμής στο βασικό EPSG του επιπέδου.

Λόγω του ότι κατά την λήψη των δεδομένων υπάρχουν συγχωνεύσεις πεδίων σε ένα κοινό που ονομάζεται ¨other_tags¨, πρέπει να ακολουθήσει ο διαχωρισμός τους σε νέα μοναδικά πεδία, χωρίζοντάς τα ανάλογα με την πληροφορία που περιέχουν.

Για την εργασία είναι απαραίτητη η πληροφορία των υλικών της επιφάνειας του δρόμου, η οποία είναι μέσα στα πεδία που έχουν συγχωνευτεί. Με την χρήση του αλγόριθμου «Expode HStore Field» επιτυγχάνεται ο διαχωρισμός αυτός. Ο αλγόριθμος δημιουργεί ένα αντίγραφο της υπάρχουσας εισόδου και χωρίζει τις επιθυμητές τιμές σε νέα μοναδικά πεδία. Δηλώνοντας στο πεδίο Hstore field του αλγόριθμου «Expode HStore Field» την είσοδο του πεδίου ¨other_tags¨ και στο πεδίο διαχωρισμού την πληροφορία που πρέπει να διαχωρίσει (surface), δημιουργεί ένα νέο πεδίο με την πληροφορία αυτή.



Σχήμα 4.1-1 Διάγραμμα ροής λήψης δεδομένων οδικού δικτύου

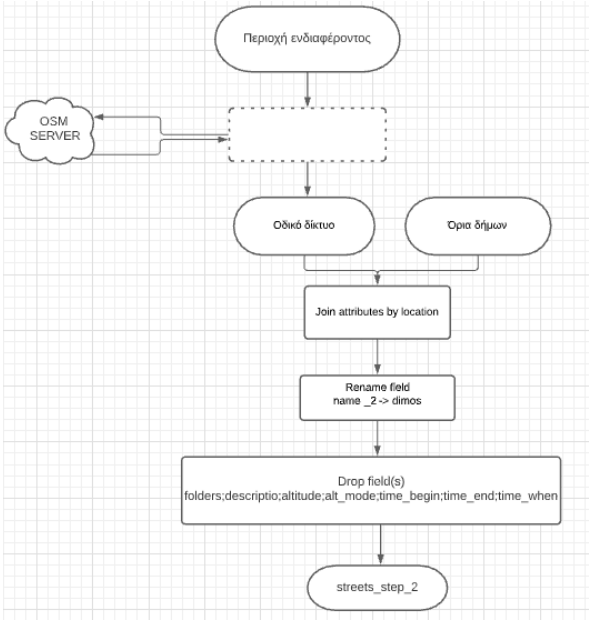


z_order	other_tags
3	"surface"=>"asphalt"
3	"lanes"=>"1","name:el:en"=>"Odos Papageorgiou Tassou","name:prefix"=>"Οδός","oneway"=>"yes","surfa...
3	"lanes"=>"1","name:el"=>"Πελοποννήσου","name:en"=>"Peloponnisou street","name:prefix"=>"Οδός","o...
0	"int_name"=>"Ioustinianou Street","name:el"=>"Ιουστινιανού","name:prefix"=>"Οδός","surface"=>"asphalt"
3	"lanes"=>"1","name:el"=>"Μακεδονικής Αμύνης","name:el:en"=>"Odos Makedonikis Amynas","name:en"...
0	"surface"=>"asphalt"
3	"lanes"=>"1","maxspeed"=>"30","name:el"=>"Πλάτωνος","name:en"=>"Platonos street","name:prefix"=>...
3	"lanes"=>"1","name:el"=>"Αγνώστου Στρατιώτη","name:en"=>"Agnostou Stratioti street","name:prefix"=...
0	"service"=>"parking_aisle","surface"=>"asphalt"
0	"oneway"=>"yes","surface"=>"asphalt"
0	"incline"=>"down","step_count"=>"1","surface"=>"asphalt"
0	"incline"=>"down","step_count"=>"1","surface"=>"asphalt"
0	"incline"=>"down","step_count"=>"1","surface"=>"asphalt"
0	"surface"=>"asphalt"
0	"surface"=>"asphalt"
0	"incline"=>"down","step_count"=>"1","surface"=>"asphalt"

	other_tags	surface	sidewalk	addr:street	building:levels	inc
6	"name:en"=>"...	asphalt	NULL	NULL	NULL	NULL
3	"name:en"=>"...	sett	NULL	NULL	NULL	NULL
3	"source:name"...	asphalt	NULL	NULL	NULL	NULL
0	"name:el:en"=...	asphalt	NULL	NULL	NULL	NULL
3	"name:en"=>"...	asphalt	NULL	NULL	NULL	NULL
3	"name:en"=>"...	asphalt	NULL	NULL	NULL	NULL
3	NULL	asphalt	NULL	NULL	NULL	NULL
3	"name:el:en"=...	asphalt	NULL	NULL	NULL	NULL
3	"name:en"=>"...	asphalt	NULL	NULL	NULL	NULL
0	NULL	asphalt	NULL	NULL	NULL	NULL
3	"name:el:en"=...	asphalt	NULL	NULL	NULL	NULL
0	NULL	asphalt	NULL	NULL	NULL	NULL
3	"name:en"=>"...	asphalt	NULL	NULL	NULL	NULL
3	"name:en"=>"...	asphalt	NULL	NULL	NULL	NULL
0	"service"=>"pa...	asphalt	NULL	NULL	NULL	NULL
0	NULL	asphalt	NULL	NULL	NULL	NULL

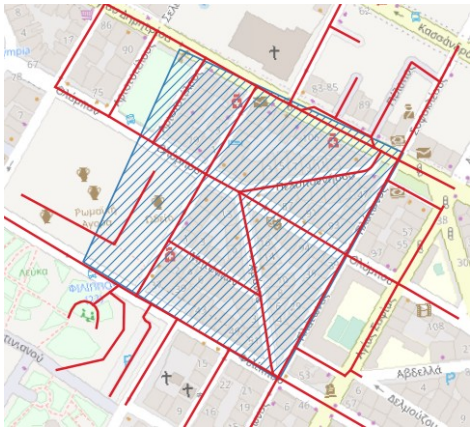
Εικόνα 4.1-2 Αριστερή εικόνα συγχωνευμένα πεδία - δεξιά εικόνα διαχωρισμένα πεδία

Όπως και προηγουμένως, καλείται ο αλγόριθμος «fix geometries» (βλ ενότητα 3.2) προληπτικά για να διορθώσει πιθανά λάθη στις γεωμετρίες. Σε αυτό το σημείο θα πρέπει να γίνει η συγχώνευση των δυο επιπέδων, δηλαδή των δεδομένων που έχουν ¨κατέβει¨ στο σύστημα από το OSM (base_layer) με αυτό το οποίο περιέχει τα όρια των δήμων, με σκοπό να εισαχθεί η πληροφορία του δήμου όπου ανήκει το κάθε στοιχείο. Ζητείται από τον αλγόριθμο «Join attributes by location» (βλ ενοτητα 3.2) να κάνει τις συγκρίσεις βάσει της τοποθεσίας των στοιχείων και να εισάγει σε νέα πεδία τις πληροφορίες αυτές. Κατά την ένωση των επιπέδων οι πληροφορίες εισάγονται σε ένα νέο πεδίο με όνομα που θα δώσει ο αλγόριθμος από μόνος του, στην προκειμένη περίπτωση είναι το ‘name_2’. Για αυτόν το λόγο καλείται ο αλγόριθμος «Rename Field(s)» για να μετονομάσει αυτό το πεδίο σε πεδίο με το όνομα ¨Dimos¨. Με τη χρήση του αλγορίθμου «Drop Filed(s)» (βλ ενότητα 3.2) αφαιρούνται και πάλι πεδία τα οποία έχουν ¨κατέβει¨ από τον server του OSM αλλά δεν περιέχουν χρήσιμες πληροφορίες για την εφαρμογή αυτή (παράδειγμα το όριο ταχύτητας του κάθε δρόμου).



Σχήμα 4.1-2 Διάγραμμα ροής δημιουργίας πεδίου Δήμου

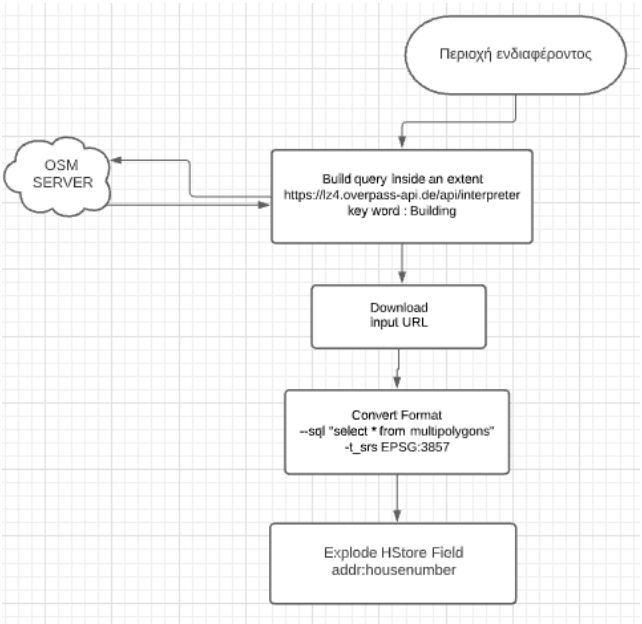
Από τον αλγόριθμο γίνεται εξαγωγή των αποτελεσμάτων με το όνομα «streets_step_2» το οποίο αποτελεί το νέο προσωρινό επίπεδο του οδικού δικτύου και έχει την παρακάτω μορφή (βλ. εικόνα 4.1-3).



Εικόνα 4.1-3 Έξοδος επιπέδου οδικού δικτύου – Streets_step_2

4.1.2 Ανάλυση μοντέλου - Κτίρια περιοχής

Ομοίως, όπως δημιουργήθηκε και το επίπεδο του οδικού δικτύου, έτσι και με το επίπεδο των κτιρίων θα πραγματοποιηθεί αίτημα προς τον server μέσω του αλγορίθμου «Build query inside an extent» (βλ. ενότητα 4.1.1). Το αίτημα αυτή τη φορά, θα αφορά δεδομένα για τις κτιριακές υποδομές εντός της περιοχής ενδιαφέροντος. Η διαφορά σε σχέση με το πρώτο βήμα είναι ότι αυτή την φορά ζητείται να σταλούν τα δεδομένα που έχουν σαν λέξη κλειδί την ετικέτα ¨bulding¨ και όχι ¨surface¨. Οι αλγόριθμοι «download» και «convert format» παραμένουν ίδιοι με το προηγούμενο βήμα, με μόνη διαφορά ότι στον αλγόριθμο «convert format» έχει δηλωθεί ότι θα περιμένει στοιχεία πολυγώνου και όχι γραμμής.



Σχήμα 4.1-3 Διάγραμμα ροής λήψης δεδομένων κτιρίων

Ο αλγόριθμος «Expode HStore Field» παραμένει και αυτός ίδιος, με την διαφορά ότι στο πεδίο Expode HStore Field ζητείται να δημιουργηθεί ένα νέο πεδίο για την πληροφορία του αριθμού της διεύθυνσης του κάθε κτιρίου και όχι την επιφάνεια του δρόμου.

Με τον ίδιο τρόπο μέσω του αλγορίθμου «Join attributes by location» (βλ. ενότητα 3.2) γίνεται συγχώνευση των επιπέδων που έχουν ¨κατέβει¨ μέχρι εκείνη την στιγμή, με το επίπεδο που περιέχει τα όρια των δήμων. Με τον αλγόριθμο «Drop Filed(s)» (βλ. ενότητα 3.2) διατηρούνται μόνο τα πεδία ενδιαφέροντος. Σε αυτό το σημείο εξάγεται το πρώτο προσωρινό επίπεδο πολυγώνου όπου περιέχει τις πληροφορίες του κάθε κτιρίου. Πλέον, υπάρχουν όλα τα δεδομένα που αντιπροσωπεύουν τα κτίρια εντός της περιοχής.

Για να πραγματοποιηθεί μετέπειτα η σύνδεση των κτιρίων με το οδικό δίκτυο, θα πρέπει το επίπεδο των κτιρίων να είναι σε μορφή σημείων, με κάθε σημείο να αντιπροσωπεύει το κάθε κτίριο. Με την χρήση του αλγορίθμου «Centroids» δημιουργείται στο κέντρο του κάθε πολυγώνου ένα νέο σημείο το οποίο έχει αντιγράψει όλα τα πεδία και τις πληροφορίες από το εισαγόμενο επίπεδο. Στον αλγόριθμο θα δοθεί ως είσοδος το επίπεδο που περιέχει τα πολύγωνα των κτιρίων και θα εξάγει τα σημεία

Σε επόμενα βήματα θα χρειαστεί ένα επίπεδο το οποίο περιέχει τις περιμέτρους των κτιρίων, με σκοπό τις μετακινήσεις των σημείων σύνδεσης από το κέντρο του κτιρίου στην πρόσοψη (βλ. ενότητα 4.2). Για τον λόγο αυτό καλείται ο αλγόριθμος «boundary» ο οποίος παίρνει ως είσοδο μια γεωμετρία πολυγώνου και από αυτήν εξάγει την περίμετρο του πολυγώνου αυτού σε επίπεδο γραμμής. Σε αυτό το επίπεδο διατηρούνται όλα τα πεδία και οι πληροφορίες που υπάρχουν στο επίπεδο εισαγωγής.

Εικόνα 4.1-4 Έξοδος τριών επιπέδων, πολυγώνων, κέντρων, περιμέτρου

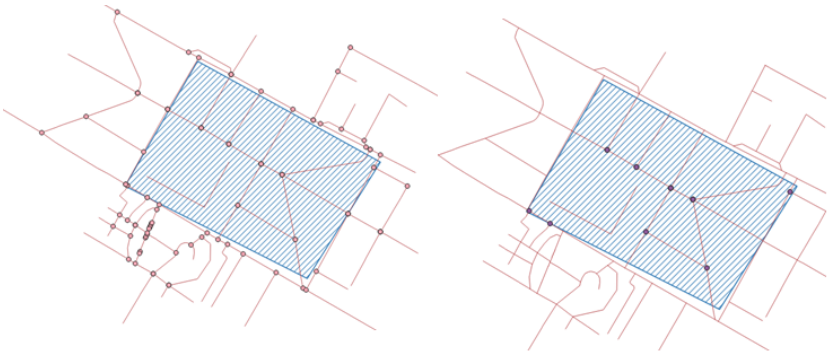
4.1.3 Ανάλυση μοντέλου - Φρεάτια

Τα φρεάτια χρησιμοποιούνται για να γίνουν οι συγκολλήσεις των οπτικών ινών. Κυρίως κατασκευάζονται στα σημεία όπου τέμνονται οι δρόμοι, με σκοπό τον διαμοιρασμό των καλωδίων προς όλες τις κατευθύνσεις. Ο σωστός αριθμός φρεατίων εξυπηρετεί στο να μην ¨φράζουν¨ οι σωλήνες του δικτύου με καλώδια που ¨πηγαίνουν¨ και ‘έρχονται’ προς την ίδια κατεύθυνση. Επίσης. με την χρήση πολλαπλών φρεατίων δίνεται η δυνατότητα σχεδίασης ενός δικτύου με τοπολογία αστέρα και όχι σειριακή. Έτσι, σε μια ενδεχόμενη βλάβη θα επηρεαστούν λιγότεροι χρήστες.

Η δημιουργία των φρεατίων πραγματοποιείται σε αυτό το βήμα και προκύπτει από τα σημεία τομής του οδικού δικτύου. Ο αλγόριθμος «line intersections» δημιουργεί ένα νέο επίπεδο σημείων όπου

διασταυρώνεται το εισαγόμενο επίπεδο με το βασικό. Στην προκειμένη περίπτωση είναι και στις δυο περιπτώσεις το οδικό δίκτυο που παράχθηκε στην ενότητα 4.1.1, καθώς στόχος είναι να βρεθούν τα σημεία τομής όλων των δρόμων στο ίδιο επίπεδο. Όταν γίνεται η λήψη των δεδομένων υπάρχουν στοιχεία τα οποία είναι ¨συνδεδεμένα¨ με άλλα εκτός της περιοχής ενδιαφέροντος, με αποτέλεσμα να γίνεται λήψη και αυτών. Για τον λόγο αυτό το επόμενο βήμα είναι να διαχωριστούν τα δεδομένα αυτά και να παραμείνουν μόνο τα σημεία εντός της περιοχής ενδιαφέροντος.

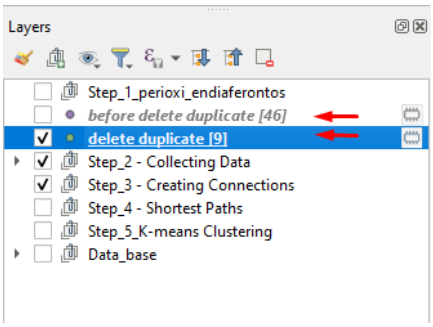
Αυτό επιτυγχάνεται με τον αλγόριθμο «extract by location» ο οποίος, δημιουργεί ένα νέο διανυσματικό επίπεδο που περιέχει μόνο πεδία από το βασικό επίπεδο εισόδου. Το βασικό επίπεδο είναι η έξοδος από τον αλγόριθμο «line intersections» ο οποίος περιέχει όλα τα φρεάτια και το συγκρίσιμο επίπεδο είναι η περιοχή ενδιαφέροντος. Κατά την εξαγωγή θα παραμείνουν μόνο αυτά που βρίσκονται εντός της ¨περιοχής ενδιαφέροντος¨.



Εικόνα 4.1-5 Σημεία τομής οδικού δικτύου - Σημεία εντός της περιοχής ενδιαφέροντος

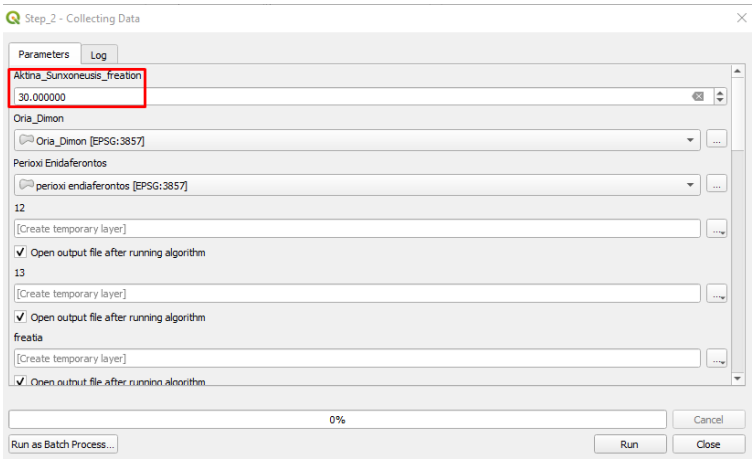
Με την δημιουργία του επιπέδου, μπορεί να δημιουργηθούν παραπάνω από ένα σημεία για την ίδια διασταύρωση, με αποτέλεσμα να καταμετρούνται παραπάνω φρεάτια από αυτά που θα χρειαστεί να κατασκευαστούν στην πραγματικότητα. Αυτό θα μπορούσε να δημιουργήσει πρόβλημα στις συνδέσεις αλλά και στον προϋπολογισμό. Ο αλγόριθμος «delete duplicate» βρίσκει διπλές γεωμετρίες και τις αφαιρεί, ελέγχοντας αν κάποια γεωμετρία είναι ακριβώς επάνω στην άλλη. Τα χαρακτηριστικά δεν ελέγχονται, οπότε σε περίπτωση που δυο στοιχεία έχουν πανομοιότυπες γεωμετρίες αλλά διαφορετικά πεδία, μόνο ένα από αυτά θα διατηρηθεί στο επίπεδο αποτελεσμάτων.

Όπως φαίνεται και στην παρακάτω εικόνα (4.1-6) για την ίδια περιοχή εντός του πλαισίου, πριν τον αλγόριθμο τα καταμετρούμενα στοιχεία ήταν σαράντα έξι ενώ μετά τον αλγόριθμο εννέα.

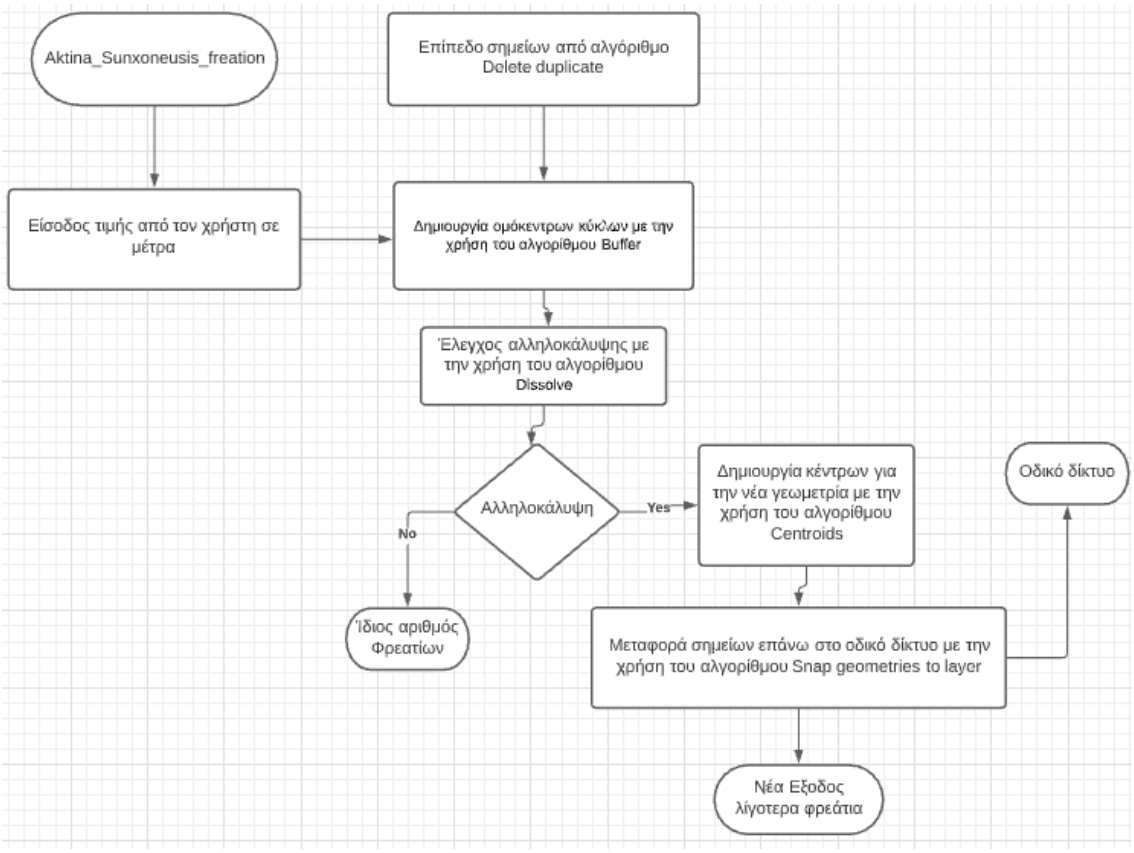


Εικόνα 4.1-6 Πριν και μετά τον αλγόριθμο delete duplicate

Ένα επιπλέον εργαλείο το οποίο μπορεί να μειώσει το κόστος κατασκευής, είναι η δυνατότητα του χρήστη να μπορεί να επιλέξει αν θέλει δυο ή και περισσότερα φρεάτια που βρίσκονται πολύ κοντά μεταξύ τους, να συγχωνευτούν σε ένα. Αυτό επιτυγχάνεται με την εισαγωγή μιας νέας αριθμητικής εισόδου η οποία ορίζει την ακτίνα συγχώνευσης (σε μέτρα) και των αλγορίθμων που αναλύονται παρακάτω.

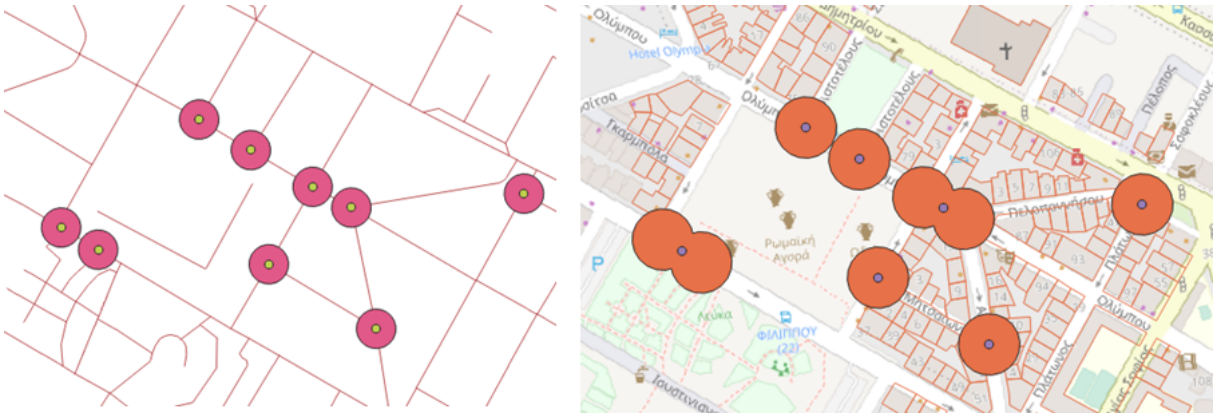


Εικόνα 4.1-7 Ορισμός απόστασης συγχώνευσης από τον χρήστη (σε μέτρα)



Σχήμα 4.1-4 Διάγραμμα ροής μείωσης φρεατίων

Η τιμή που θα δώσει ο χρήστης θα χρησιμοποιηθεί ως τιμή εισόδου στον αλγόριθμο «buffer» και είναι υπεύθυνη για την δημιουργία ενός ομόκεντρου κύκλου με ακτίνα ίση της δηλωθείσας τιμής για όλα τα στοιχεία του επιπέδου. Αφού δημιουργηθούν οι ομόκεντροι κύκλοι θα γίνει έλεγχος με την χρήση του αλγορίθμου «dissolve» και αν υπάρχει αλληλοκάλυψη σε κάποιο σημείο θα δημιουργηθεί μια νέα κοινή γεωμετρία. Έπειτα, καλείται και πάλι ο αλγόριθμος «centroids» (βλ. ενότητα 4.1.1) να δημιουργήσει νέα ¨κέντρα¨ για τις γεωμετρίες που έχουν συγχωνευθεί.



Εικόνα 4.1-8 Δημιουργία ομόκεντρων κύκλων ακτίνας 20 και 30 μέτρων

Παρατηρείται ότι στην περίπτωση δημιουργίας ομόκεντρου κύκλου ακτίνας 30μ (βλ. εικόνα 4.1.8) που έχει υπάρξει αλληλοκάλυψη σημείων, τα συνολικά φρεάτια είναι επτά σε αντίθεση με την περίπτωση ομόκεντρου κύκλου ακτίνας 20μ όπου δεν υπάρχει αλληλοκάλυψη και είναι εννέα. Στην περίπτωση που υπάρχει συγχώνευση σημείων ενδέχεται το νέο κέντρο να μην βρίσκεται επάνω στην γραμμή του οδικού δικτύου κάτι που μπορεί να δημιουργήσει προβλήματα σε επόμενα βήματα. Για αυτόν τον λόγο καλείται ο αλγόριθμος «snap geometry to layer» να μετακινήσει τα νέα παραγόμενα ¨κέντρα¨ επάνω στο επίπεδο του οδικού δικτύου.

Για τον αλγορίθμου «snap geometry to layer» έχει δηλωθεί η μέγιστη τιμή ανοχής που μπορεί να μετακινήσει ένα σημείο, η οποία είναι 500μ. Δεν έχει ιδιαίτερη σημασία αν η τιμή είναι πολύ μεγάλη, καθώς θα βρει το πλησιέστερο σημείο του επιπέδου εισόδου. Αντιθέτως, αν η τιμή ήταν μικρότερη από την απόσταση του οδικού δικτύου θα μπορούσε να παρουσιαστεί πρόβλημα στο να μην γίνει η μεταφορά του σημείου. Όπως και στα προηγούμενα βήματα ο αλγόριθμος «Retain field» (βλ. ενότητα 3.2) καλείται να κρατήσει τα πεδία ενδιαφέροντος και να απορρίψει τα υπόλοιπα.

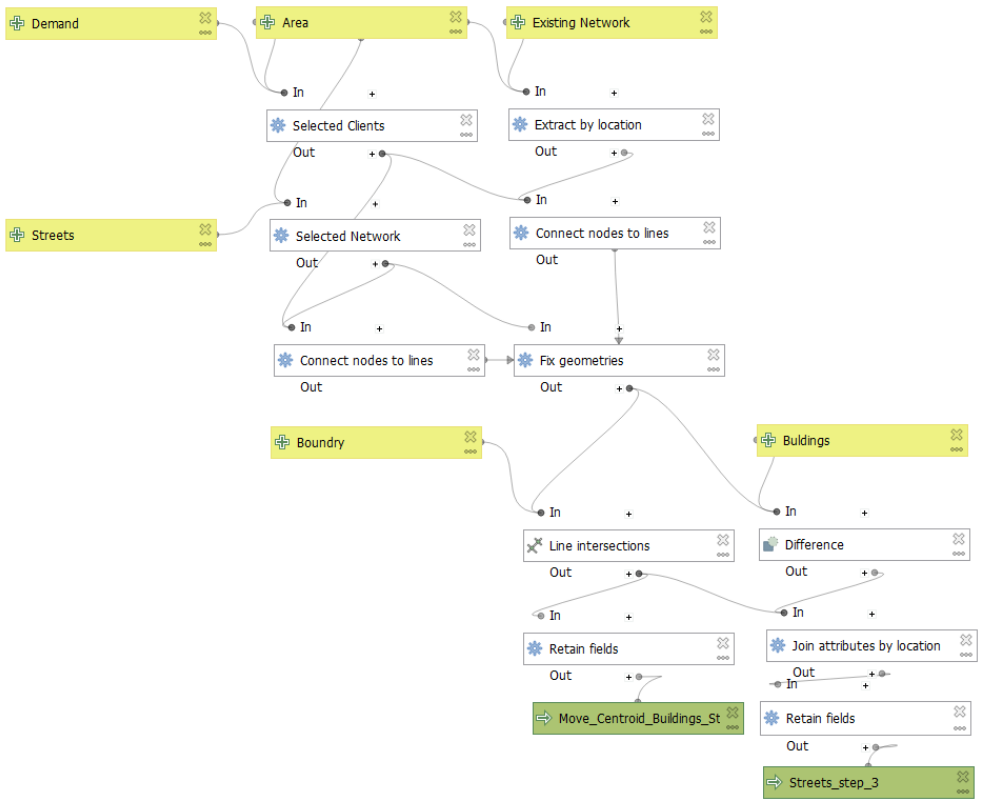
Έπειτα, θα δημιουργηθούν τρία νέα πεδία, το ένα από τον αλγόριθμο «Add autoincremental field» ο οποίος προσθέτει ένα νέο ακέραιο πεδίο στο επίπεδο με μια διαδοχική αρίθμηση για κάθε στοιχείο. Το πεδίο αυτό χρησιμοποιείται ως μοναδικό αναγνωριστικό (id) για κάθε φρεάτιο.

Το δεύτερο πεδίο δημιουργείται από τον αλγόριθμο «add field to attributes table» το οποίο είναι ένα νέο κενό πεδίο στο οποίο θα μπορεί ο χρήστης να δηλώσει χειροκίνητα τις διαστάσεις του φρεατίου όταν αυτό κατασκευαστεί.

Τέλος, με την χρήση του αλγορίθμου «add X/Y fields to layer» θα προστεθούν τα πεδία X και Y (γεωγραφικό πλάτος και μήκος). Τα πεδία X / Y θα δημιουργηθούν βάσει του EPSG του βασικού επιπέδου .

4.2 Σύνδεση κτιρίων με το δίκτυο

Σε αυτό το βήμα θα γίνει η σύνδεση των σημείων μεταξύ των επιπέδων που περιέχουν τα κτίρια (βλ. ενότητα 4.1.2) και του οδικού δικτύου (βλ. ενότητα 4.1.1). Παράλληλα θα γίνει σύνδεση του υφιστάμενου δικτύου από τη βάση δεδομένων (βλ. ενότητα 2.1.1) και η μεταφορά των σημείων σύνδεσης από το κέντρο των κτιρίων στην πρόσοψη. Στο μοντέλο έχουν οριστεί έξι είσοδοι. Κατά την περάτωση του μοντέλου δημιουργούνται δυο νέα επίπεδα. Ένα όπου περιέχει το οδικό δίκτυο με τις κάθετες τομές προς το σημείο σύνδεσης και ένα το οποίο περιέχει τα σημεία σύνδεσης. (Πλήρης κώδικας – Παράρτημα Γ).



Εικόνα 4.2-1 Μοντέλο σύνδεσης κτιρίων με το δίκτυο

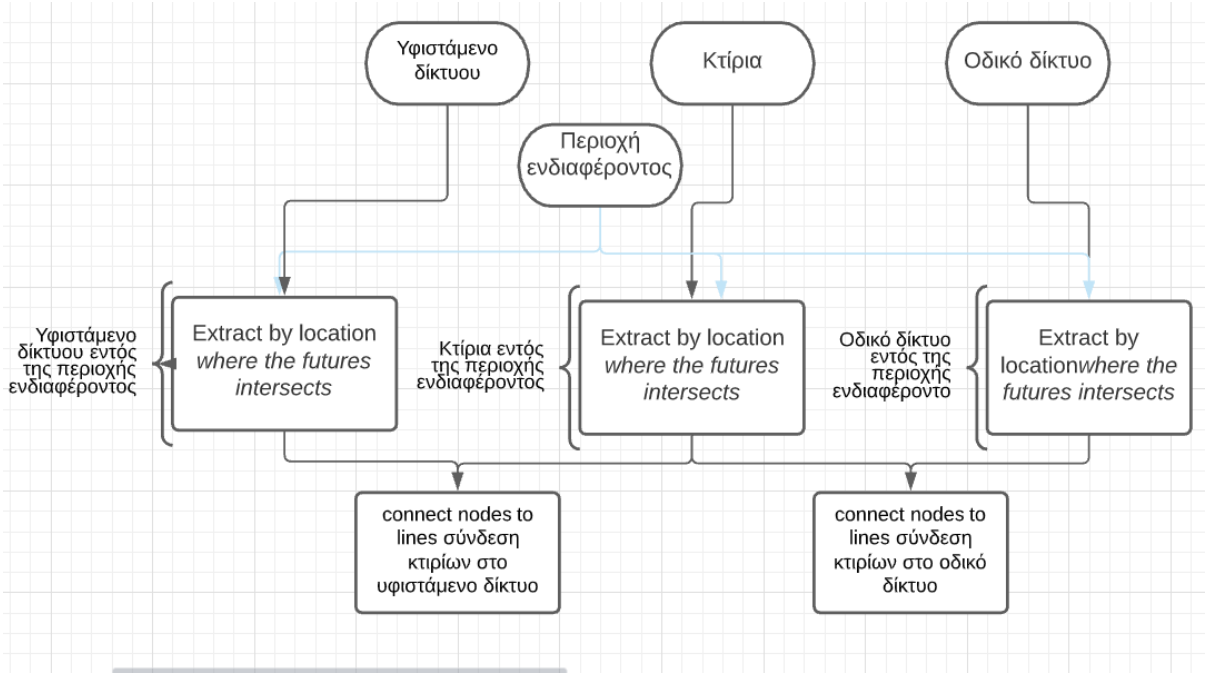
4.2.1 Ανάλυση μοντέλου - Σύνεση κτιρίων με το δίκτυο

Σε αυτό το μοντέλο θα χρειαστεί ο χρήστης να δώσει έξι εισόδους. Τέσσερις από αυτές είναι, τα προσωρινά επίπεδα που έχουν δημιουργηθεί στο προηγούμενο βήμα (βλ. ενότητες 4.1.1 – 4.1.2), οι οποίες είναι το οδικό δίκτυο και τα επίπεδα που περιέχουν τις περιμέτρους, τα πολύγωνα και τα σημεία των κτιρίων. Ακόμα δυο από αυτές προέρχονται από τη βάση δεδομένων οι οποίες είναι, η περιοχή ενδιαφέροντος και το υφιστάμενο δίκτυο (βλ. ενότητα 2.3).

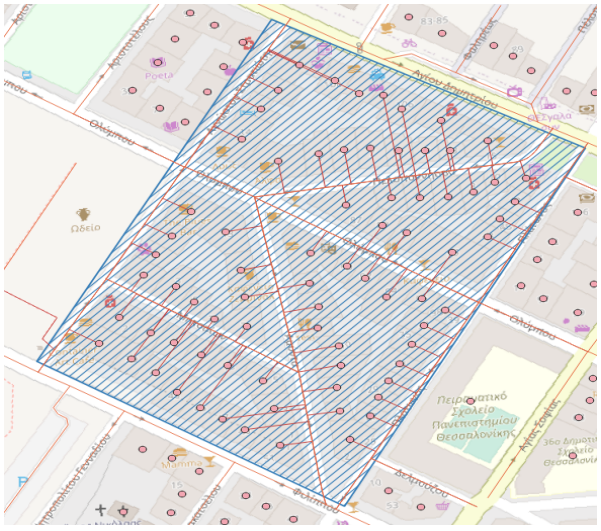
Αρχικά, καλείται ο αλγόριθμος «extract by location» (βλ. ενότητα 4.1.3) να κάνει σύγκριση της περιοχής ενδιαφέροντος με το επίπεδο των κτιρίων (σημείων) ώστε να εξάγει μόνο τα στοιχεία εντός της περιοχής αυτής, όπου και θα πραγματοποιηθούν οι συνδέσεις.

Ο ίδιος αλγόριθμος καλείται ξανά με σκοπό να πραγματοποιήσει αυτή τη φορά σύγκριση του υφιστάμενου δικτύου σε σχέση με την περιοχή ενδιαφέροντος. Σκοπός είναι να γίνουν οι συνδέσεις και στο υφιστάμενο επίπεδο του δικτύου, καθώς θα χρειαστούν σε μελλοντικά βήματα.

Τέλος, χρησιμοποιείται άλλη μία φορά για να καθοριστεί το πεδίο ενδιαφέροντος στο προσωρινό επίπεδο του οδικού δικτύου, ώστε να γίνει εξαγωγή των αποτελεσμάτων που βρίσκονται εντός της περιοχής. Έπειτα, καλείται ο αλγόριθμος «connect nodes to lines» ο οποίος θα πραγματοποιήσει τις συνδέσεις των σημείων του επιπέδου κτιρίων με το οδικό δίκτυο. Ο αλγόριθμος θα εντοπίσει την πλησιέστερη απόσταση μεταξύ των σημείων και των γραμμών και θα δημιουργήσει συνδέσεις βασιζόμενος στην απόσταση ανοχής που έχει οριστεί στον κώδικα (1000 μέτρα).

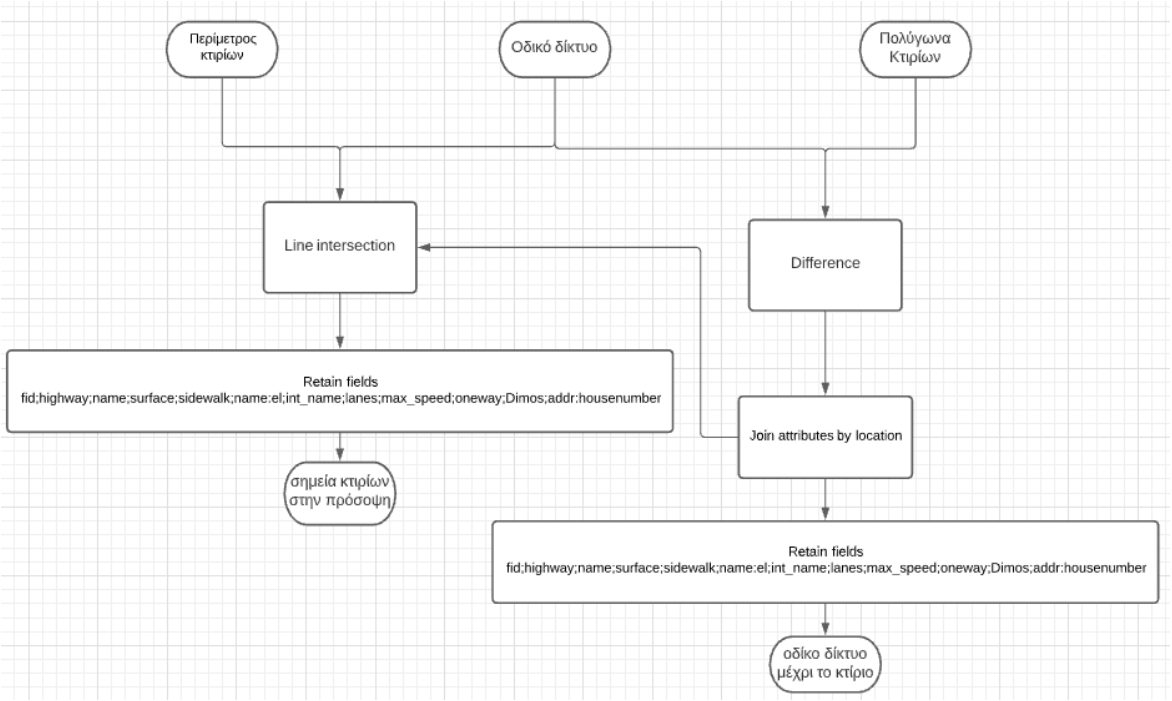


Σχήμα 4.2-1 Διάγραμμα ροής σύνδεσης στοιχείων εντός της περιοχής ενδιαφέροντος



Εικόνα 4.2-2 Σύνδεση σημείων με το δίκτυο στον χάρτη

Σε αυτό το σημείο έχουν δημιουργηθεί οι συνδέσεις, αλλά όπως φαίνεται και στην εικόνα (4.2-2), οι γραμμές των συνδέσεων φτάνουν στο κέντρο των κτιρίων, κάτι που δεν μπορεί να γίνει στην πραγματικότητά, καθώς τα έργα θα σταματήσουν στην πρόσοψη του κτιρίου. Για τον λόγο αυτό θα πρέπει να μεταφερθούν τα σημεία σύνδεσης από το κέντρο στην είσοδο του κτιρίου.



Σχήμα 4.2-2 Διάγραμμα ροής μεταφοράς κέντρων

Για να επιτευχθεί αυτό, καλείται ο αλγόριθμος «line intersections» (βλ. ενότητα 4.1.3) ο οποίος θα δεχθεί ως είσοδο το επίπεδο των περιμέτρων των κτιρίων και θα κάνει σύγκριση με το επίπεδο του οδικού δικτύου. Ο αλγόριθμος θα βρει τα σημεία που τέμνονται τα δυο επίπεδα και θα δημιουργήσει νέα σημεία. Μεγάλη προσοχή πρέπει να δοθεί στο πλαίσιο της περιοχής ενδιαφέροντος, καθώς αν για τα κτίρια ενδιαφέροντος δεν εμπεριέχεται και ο δρόμος στον οποίο πρόκειται να γίνει η σύνδεση, ο αλγόριθμος θα βρει τον επόμενο πλησιέστερο δρόμο με αποτέλεσμα οι συνδέσεις να περνούν μέσα από τα κτίρια.

Στην παρακάτω αριστερή εικόνα (4.2-3), φαίνεται ότι τα κτίρια στο κάτω μέρος εμπεριέχονται στην περιοχή ενδιαφέροντος, αλλά ο δρόμος που θεωρητικά πρέπει να γίνουν τα έργα δεν είναι εντός του πλαισίου με αποτέλεσμα τα σημεία σύνδεσης να περνούν μέσα από τα κτίρια. Στη δεξιά εικόνα αφού γίνει η αλλαγή στην περιοχή ενδιαφέροντος, με τον δρόμο σύνδεσης να περιλαμβάνεται, οι συνδέσεις έχουν πραγματοποιηθεί σωστά.

Εικόνα 4.2-3 Αριστερή εικόνα : δρόμος εκτός περιοχής. Δεξιά εικόνα : δρόμος εντός περιοχής

Θα μπορούσε επίσης να οριστεί μικρότερη ανοχή σύνδεσης μεταξύ των σημείων, ώστε να αποφευχθεί η εσφαλμένη σύνδεση από γειτονικό δρόμο. Για παράδειγμα, αν αλλάξει το πεδίο ανοχής από χίλια μέτρα σε δέκα. Αν τα δυο σημεία απείχαν πάνω από το όριο δεν θα πραγματοποιούνταν οι συνδέσεις.

Ωστόσο, επειδή δεν υπάρχει ένα συγκεκριμένο πλάτος δρόμου, αυτό θα μπορούσε να δημιουργήσει προβλήματα σε μεγάλους δρόμους, όπως για παράδειγμα σε λεωφόρους που τα πλάτη των δρόμων είναι μεγαλύτερα, με αποτέλεσμα να μην πραγματοποιούνται καθόλου οι συνδέσεις. Αν δεν γίνει η μετατόπιση της περιοχής ενδιαφέροντος θα πρέπει οι εσφαλμένες γεωμετρίες να διαγραφούν χειροκίνητα.

Σε αυτό το σημείο έχει επιτευχθεί η μεταφορά των σημείων σύνδεσης από το κέντρο στην πρόσοψη του κτιρίου και απομένει η δημιουργία της νέας γραμμής σύνδεσης, η οποία θα πρέπει να καταλήγει και αυτή στην πρόσοψη του κτιρίου και όχι στο κέντρο του. Το ζητούμενο είναι να αφαιρεθεί το μέρος αυτό που φτάνει από την πρόσοψη του κτιρίου μέχρι το κέντρο και να παραμείνει το υπόλοιπο μέρος της. Αυτό θα επιτευχθεί με τον αλγόριθμο «difference» στο οποίο θα δοθεί σαν επίπεδο εισόδου το οδικό δίκτυο και σαν επίπεδο επικάλυψης το επίπεδο πολυγώνων των κτιρίων. Κατά την έξοδο διατηρούνται μόνο τα τμήματα του βασικού επιπέδου εκτός αυτών που ταυτίζονται με το επίπεδο επικάλυψης.

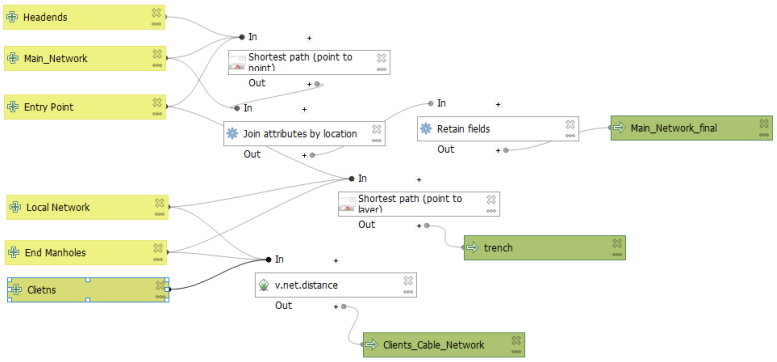
Εικόνα 4.2-4 Αριστερά: Γραμμή ως το κέντρο του κτιρίου. Δεξιά: Γραμμή ως την πρόσοψη του κτιρίου

Τα εξαγόμενα αποτελέσματα είναι δυο επίπεδα, ένα που ονομάζεται ¨move centroids step 3¨, το οποίο αποτελεί τα νέα σημεία σύνδεσης και ένα που ονομάζεται ¨street steps 3¨, το οποίο αποτελεί το οδικό δίκτυο με τις συνδέσεις μέχρι τα κτίρια.

4.3 Εύρεση σύντομων διαδρομών

Σε αυτό το βήμα θα γίνει η σύνδεση του τοπικού δικτύου με το κέντρο (Headend) και κατ' επέκταση η σύνδεση με το υφιστάμενο δίκτυο, κρατώντας μόνο εκείνες τις διαδρομές που θα πρέπει να πραγματοποιηθούν τα έργα για να συνδεθούν όλα τα σημεία. Σε αυτό το μοντέλο θα χρειαστεί ο χρήστης να δώσει έξι εισόδους οι οποίες είναι, τα σημεία σύνδεσης του προηγούμενου βήματος, τα φρεάτια, το σημείο που θα οριστεί ως σημείο εισόδου ¨Entry Point¨, το κέντρο(Headend) με το οποίο θα γίνει η σύνδεση, το τοπικό δίκτυο που έχει δημιουργηθεί και τέλος το υφιστάμενο οδικό δίκτυο.

Το προσδοκώμενο αποτέλεσμα είναι τρείς έξοδοι οι οποίες είναι, το επίπεδο που αποτελεί το δίκτυο κορμού, όπου είναι η σύνδεση του κέντρου με το φρεάτιο που έχει οριστεί ως σημείο εισόδου, το επίπεδο το οποίο αποτελεί τις συνδέσεις μεταξύ των φρεατίων και τέλος το επίπεδο το οποίο αποτελείται από τις συνδέσεις των φρεατίων έως τα κτίρια . (Πλήρης κώδικας – Παράρτημα Δ).



Εικόνα 4.3-1 Μοντέλο Βέλτιστων διαδρομών

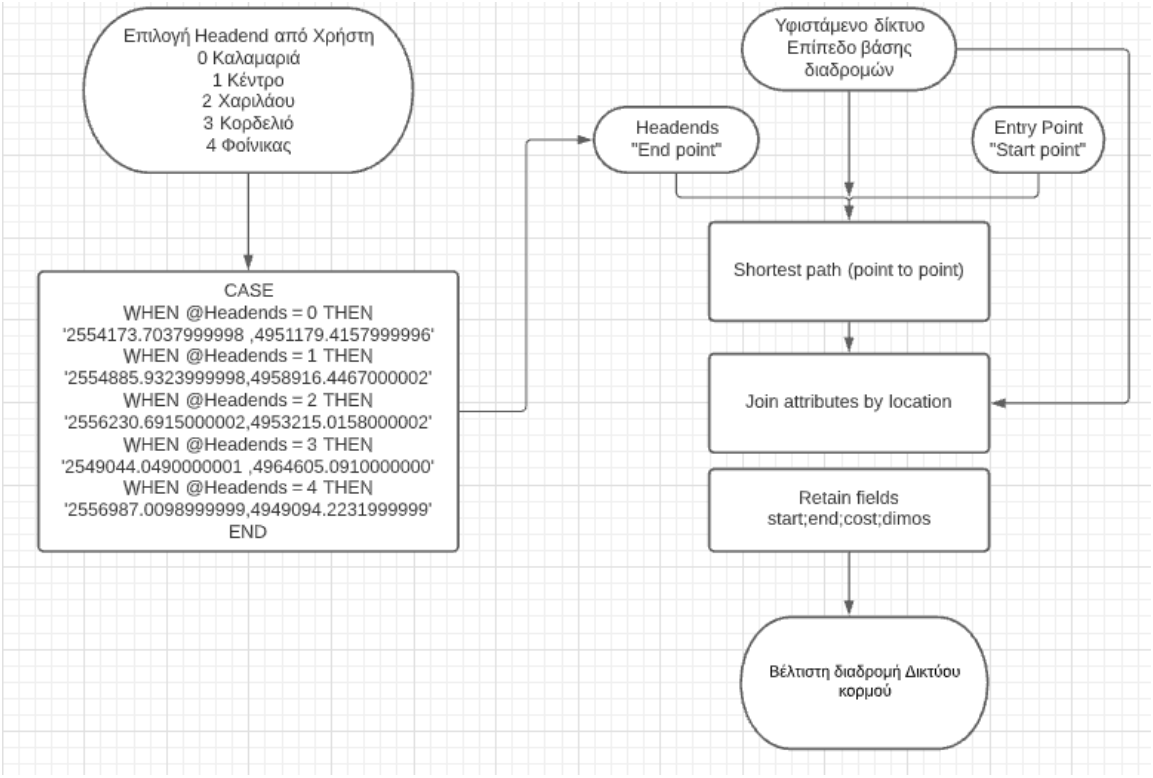
4.3.1 Δίκτυο κορμού

Σε αυτό το τμήμα του μοντέλου τα δεδομένα εισόδου είναι τα Headend, Main_Network και Entry point. Στο επίπεδο headned υπάρχουν όλα τα κέντρα σύνδεσης τα οποία διαθέτει ο πάροχος.

Για τον σκοπό της εργασίας έχουν δηλωθεί πέντε τυχαία σημεία σε διαφορετικές περιοχές της πόλης (Καλαμαριά, Κέντρο, Χαριλάου, Κορδελιό και Φοίνικας). Δίνεται η δυνατότητα στον χρήστη να μπορεί να επιλέξει σε ποιο κέντρο επιθυμεί να γίνει η σύνδεση του υφιστάμενου δικτύου με αυτό που κατασκευάζεται. Για τον σκοπό αυτό δημιουργήθηκε μια είσοδος πολλαπλών επιλογών στην οποία έχουν δοθεί ονομαστικά τα κέντρα σύνδεσης που υπάρχουν, με χαρακτηριστικό να μην επιτρέπει στον χρήστη την επιλογή παραπάνω του ενός κέντρου.

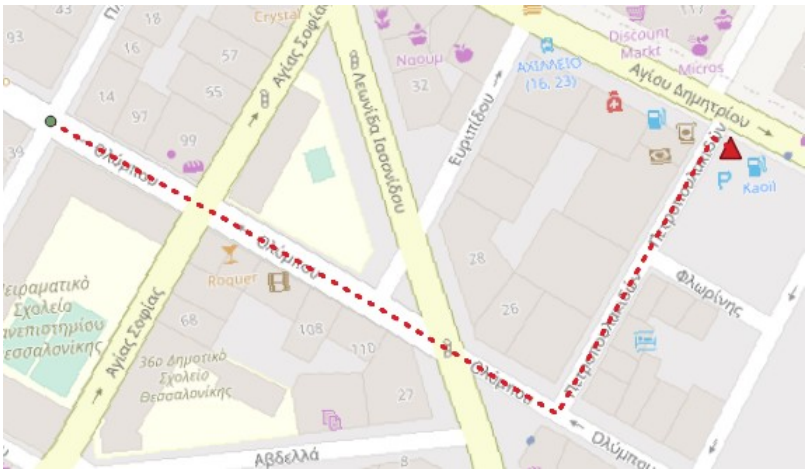
Καλείται ο αλγόριθμος «shortest paths» point to point ο οποίος βρίσκει την βέλτιστη διαδρομή μεταξύ δυο σημείων βασιζόμενος σε ένα επίπεδο γραμμών που έχει δοθεί ως είσοδος. Ως βάση διαδρομών στον

αλγόριθμο έχει δοθεί το επίπεδο που περιέχει το υφιστάμενο δίκτυο της πόλης. Στην συγκεκριμένη περίπτωση καλείται να βρει την βέλτιστη διαδρομή μεταξύ του ¨entry point¨ (start point) και του ¨Headend¨ (End point). Στο πεδίο ¨end point¨ του αλγορίθμου έχουν δηλωθεί οι συντεταγμένες των κέντρων σύνδεσης μέσω του κώδικα. Το σημείο ¨entry point¨ δίνεται χειροκίνητα από τον χρήστη, επιλέγοντας ένα σημείο επάνω στον χάρτη. Στόχος είναι να μπορεί να επιλέξει σε ποιο από όλα τα φρεάτια θέλει να καταλήξει η οπτική ίνα του κεντρικού κορμού και από εκεί να γίνει ο διαμοιρασμός μετέπειτα.



Σχήμα 4.3-1 Διάγραμμα ροής δικτύου κορμού

Έπειτα, αφού βρεθεί η βέλτιστη διαδρομή και καθώς στο επίπεδο εξόδου που έχει παραχθεί από τον αλγόριθμο δεν περιέχονται οι πληροφορίες των άλλων επιπέδων, καλείται ο αλγόριθμος «join attributes by location» (βλ. ενότητα 3.2) να κάνει συγχώνευση πεδίων ώστε να περάσουν οι πληροφορίες στο νέο επίπεδο. Τέλος καλείται ο αλγόριθμος «Retain Field(s)» (βλ. ενότητα 3.2) για να κρατήσει μόνο τα πεδία ενδιαφέροντος. Η έξοδος του επιπέδου είναι της μορφής γραμμών και έχει την παρακάτω μορφή.



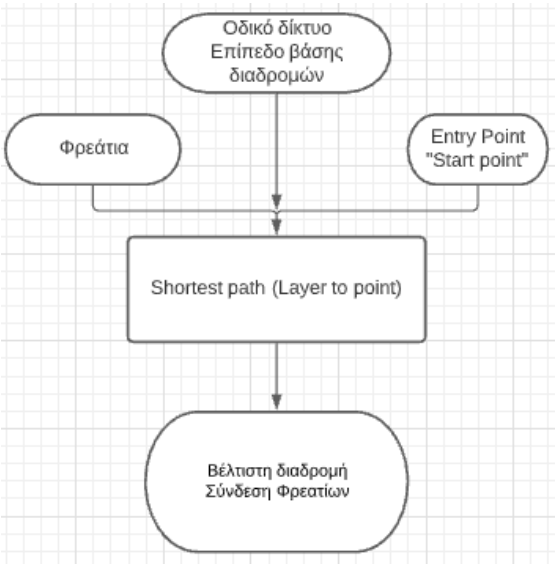
Εικόνα 4.3-2 Βέλτιστη διαδρομή δικτύου κορμού

4.3.2 Σύνδεση φρεατίων

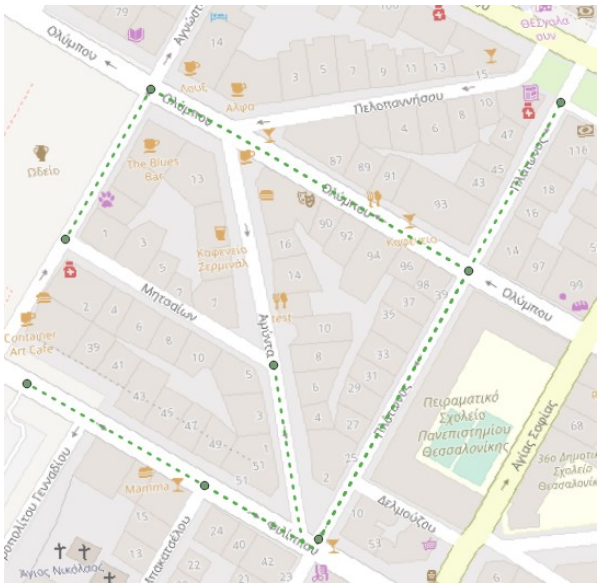
Αφού έχει ολοκληρωθεί η δημιουργία του δικτύου κορμού θα πρέπει να γίνει η σύνδεση όλων των φρεατίων μεταξύ τους, βρίσκοντας και πάλι τις βέλτιστες διαδρομές.

Καλώντας τον αλγόριθμο «shortest paths» point to layer αυτή τη φορά ο οποίος δέχεται σαν αφετηρία (Start point) ένα επίπεδο πολλαπλών σημείων και ως σημείο τερματισμού (end point) ένα μοναδικό σημείο. Ο αλγόριθμος. θα βρει τις βέλτιστες διαδρομές βασιζόμενος στο επίπεδο γραμμών που έχει δοθεί ως είσοδος. Ως σημείο αφετηρίας, έχει δοθεί το επίπεδο των φρεατίων και σαν σημείο τερματισμού το σημείο ¨entry point¨ το οποίο παίρνει αυτόματα τις συντεταγμένες από το προηγούμενο βήμα. Με αυτόν τον τρόπο θα υπολογισθούν όλες οι σύντομες διαδρομές από το φρεάτιο ¨entry point¨, σε σχέση με όλα τα υπόλοιπα.

Το δίκτυο οδεύσεων που θα χρησιμοποιήσει ο αλγόριθμος ως βάση οδεύσεων είναι το τοπικό δίκτυο που έχει δημιουργηθεί από το προηγούμενο μοντέλο (βλ. ενότητα 4.2.1). Το εξαγόμενο αποτέλεσμα απεικονίζεται στην εικόνα (4.3-3).

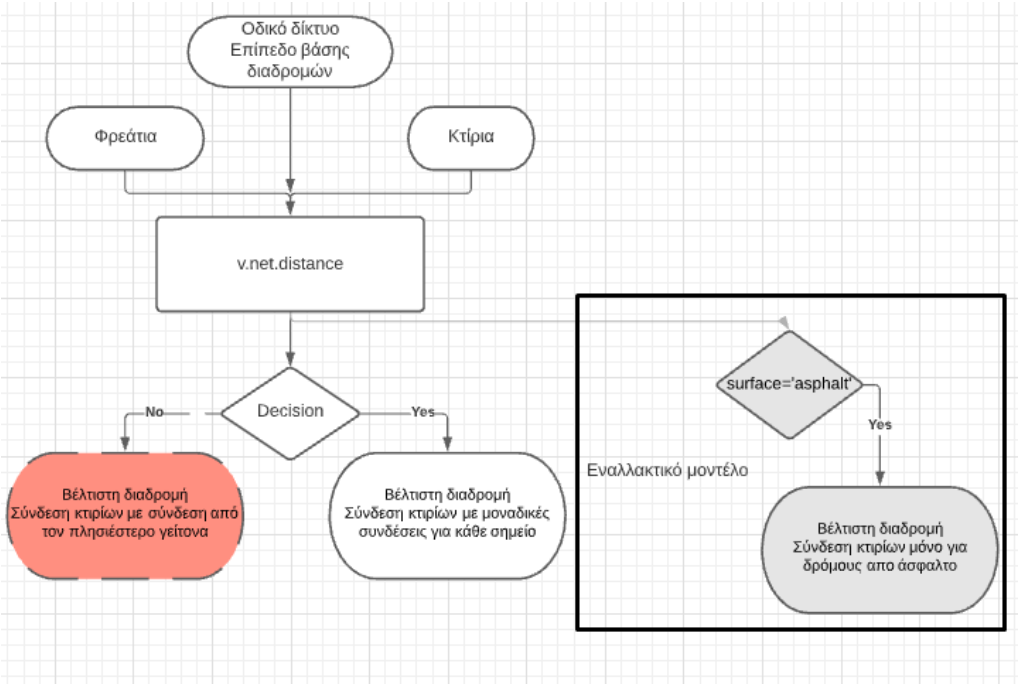


Σχήμα 4.3-2 Διάγραμμα ροής σύνδεσης φρεατίων



Εικόνα 4.3-3 Σύνδεση φρεατίων

4.3.3 Σύνδεση κτιρίων



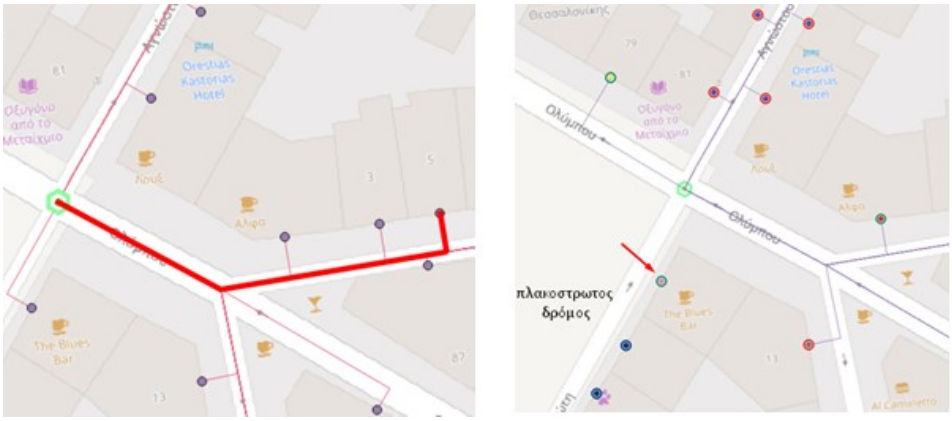
Σχήμα 4.3-3 Διάγραμμα ροής σύνδεσης κτιρίων

Αφού έχει δημιουργηθεί το επίπεδο του βασικού κορμού και της σύνδεσης των φρεατίων μεταξύ τους, αυτό που απομένει είναι να γίνουν οι συνδέσεις των κτιρίων με τα φρεάτια.

Ο αλγόριθμος «v.net.distance» έχοντας ως επίπεδο δικτύου το τοπικό δίκτυο θα βρει όλες τις σύντομες διαδρομές μεταξύ των κτιρίων και των φρεατίων. Ένα επιπλέον χαρακτηριστικό είναι ότι έχει

προγραμματιστεί έτσι ώστε να βρίσκει για κάθε σημείο την διαδρομή σαν αυτοτελή όδευση και όχι σαν συνέχεια του προηγούμενου σημείου, καθώς όπως και στην πραγματικότητα το κάθε καλώδιο θα καταλήξει στο φρεάτιο και όχι στο διπλανό κοντινότερο κτίριο.

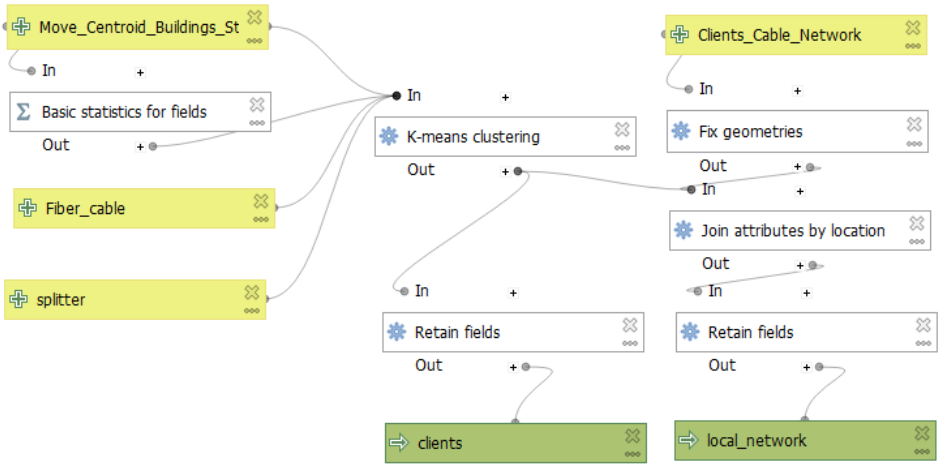
Το ίδιο μοντέλο έχει δημιουργηθεί και με την επιλογή να αγνοεί όλες τις διαδρομές που δεν αποτελούνται από το υλικό της ασφάλτου. Όταν υπάρχει εναλλακτική όδευση εντός του πλαισίου της περιοχής ενδιαφέροντος η όδευση αλλάζει περνώντας μόνο από δρόμους που έχουν την σήμανση ¨άσφαλτος¨. Το πρόβλημα του μοντέλου αυτού είναι ότι αν δεν υπάρχει ενναλακτική όδευση θα μείνουν τα σημεία ασύνδετα χωρίς να υπάρχει κάλυψη 100% όλων των κτιρίων.



Εικόνα 4.3-4 Αριστερή εικόνα : Έξοδος μοντέλου Δεξιά εικόνα : ενναλακτικό μοντέλο σχεδίαση σε άσφαλτο

4.4 Ομαδοποίηση

Τελευταίο βήμα στο σχεδιαστικό τμήμα αποτελεί η ομαδοποίηση των δεδομένων που έχουν δημιουργηθεί έως τώρα. Πριν ξεκινήσει αυτό το βήμα πρέπει να γίνει έλεγχος και να διαγραφούν πιθανές εσφαλμένες γεωμετρίες. Σε αυτό το μοντέλο ο χρήστης θα δώσει τέσσερις εισόδους και αναμένεται να δημιουργηθούν δυο νέες έξοδοι. (Πλήρης κώδικας – Παράρτημα Z)



Εικόνα 4.4-1 Μοντέλο ομαδοποιήσεις

Ο χρήστης πρέπει να δώσει ως είσοδο το μέγεθος καλωδίου (για παράδειγμα καλώδιο 48 ινών), το επίπεδο τοπικού δικτύου όπου θα γίνει η ομαδοποίηση, τα κτίρια που θα ομαδοποιηθούν και το μεγαλύτερο βαθμό διαχωρισμού που μπορεί να υποστεί μια οπτική ίνα (splitter).

4.4.1 Ομαδοποίηση κτιρίων

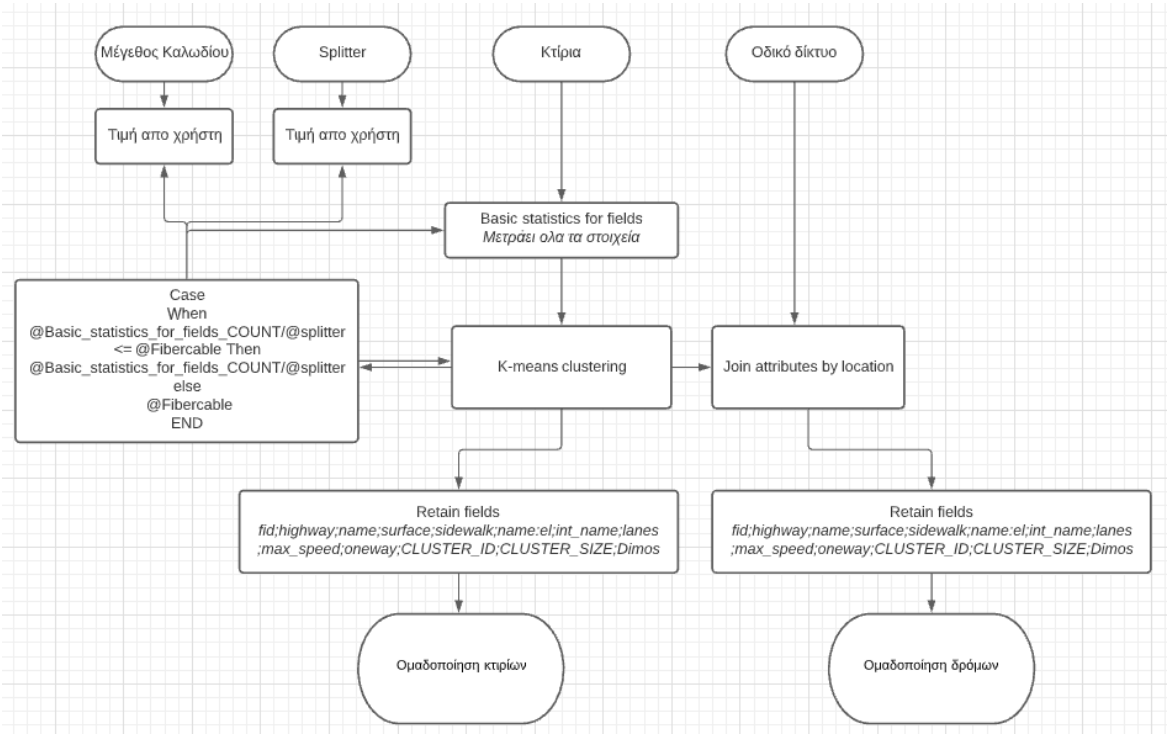
Με την ομαδοποίηση των κτιρίων θα γίνει διαμοιρασμός των οπτικών ινών ανά ομάδες. Το μέγεθος των ομάδων έχει να κάνει καθαρά με την επιλογή του μηχανικού και τον τρόπο που έχει αποφασίσει να σχεδιάσει το δίκτυο.

Στην περίπτωση της εργασίας τα καλώδια οπτικών ινών που χρησιμοποιούνται για το δίκτυο κορμού είναι εκατόν σαράντα τεσσάρων ινών, δηλαδή δώδεκα σωληνίσκων που ο καθένας περιέχει δώδεκα ίνες και αυτά που χρησιμοποιούνται για το τοπικό δίκτυο, τα οποία είναι καλώδια σαράντα οκτώ ινών δηλαδή τεσσάρων σωληνίσκων των δώδεκα ινών.

Με τον τρόπο αυτό, το καλώδιο των εκατόν σαράντα τεσσάρων οπτικών ινών θα φτάσει, από το κέντρο (headend) στο κεντρικό φρεάτιο ¨Entry point¨ και από εκεί με χρήση καλωδίων σαράντα οκτώ ινών, θα γίνει ο διαμοιρασμός προς όλα τα κτίρια και κατευθύνσεις.

Αυτό επιτυγχάνεται με την χρήση του αλγορίθμου «K-means». Η ομαδοποίηση αυτή δεν γίνεται σε τυχαίες ομάδες, αλλά την ορίζει ο μηχανικός επιλέγοντας το μέγεθος των καλωδίων και κατά πόσο θέλει να γίνει χρήση παθητικών στοιχείων (splitters) στο δίκτυο. Ο μηχανικός θα δηλώσει την τιμή των καλωδίων και των διαχωριστών ως αριθμητικές ακέραιες εισόδους πριν από την εκτέλεση του μοντέλου. Οι τιμές που μπορεί να δώσει είναι, για το καλώδιο ελάχιστη τιμή αυτή της μιας ίνας και μέγιστη τιμή αυτή των εκατόν σαράντα τεσσάρων ινών με προκαθορισμένη τιμή καλωδίου σαράντα οκτώ ινών. Ενώ αντίστοιχα για τους διαχωριστές ο μηχανικός μπορεί να δώσει ελάχιστη τιμή, την τιμή ένα, δηλαδή να μην υπάρξει διαχωρισμός της ίνας και μέγιστη τιμή, την τιμή των εξήντα τεσσάρων εξόδων, με προκαθορισμένη τιμή διαχωριστή αυτή των τεσσάρων εξόδων.

Αρχικά, καλείται ο αλγόριθμος «Basic statistics for fields», ο οποίος με βάση το επίπεδο εισαγωγής μπορεί και εξάγει βασικά στατιστικά στοιχεία από την ανάλυση τιμών σε κάθε πεδίο. Έχει δοθεί ως είσοδος στον αλγόριθμο το επίπεδο των κτιρίων και του έχει ζητηθεί να μετρήσει όλα τα στοιχεία βασιζόμενος στο πεδίο ¨fid¨. Με αυτόν τον τρόπο μετράει πόσες φορές εμφανίζεται στον πίνακα το πεδίο αυτό και εξάγει έναν αριθμό ο οποίος είναι το σύνολο των στοιχείων. Γνωρίζοντας τον συνολικό αριθμό των στοιχείων και με βάση τις τιμές που έχει δηλώσει ο μηχανικός για το καλώδιο που θα χρησιμοποιηθεί αλλά και για τους διαχωριστές, θα πραγματοποιηθεί η ομαδοποίηση.



Σχήμα 4.4-1 Διάγραμμα ροής ομαδοποίησης

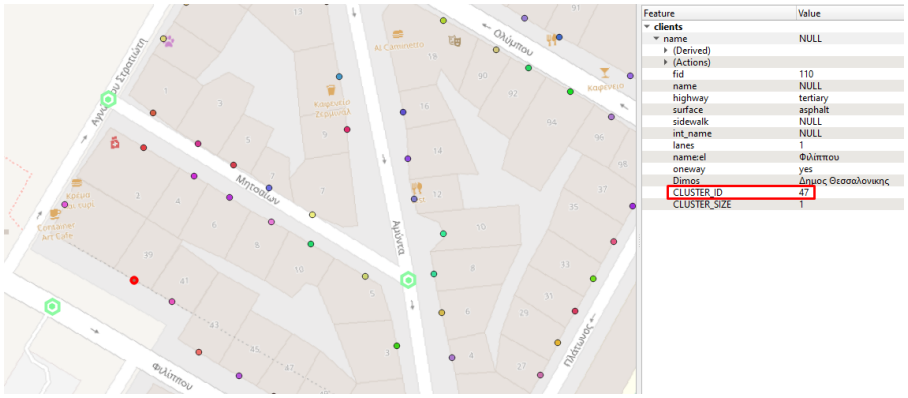
Επόμενο βήμα είναι η χρήση του αλγορίθμου «K-means» ο οποίος έχει ως είσοδο τα κτίρια της περιοχής και σαν αριθμό δημιουργίας ομάδων (Clusters) την παρακάτω σχέση της εικόνας (4.4-2).Συνεπώς, όταν τα κτίρια είναι λιγότερα από τις διαθέσιμες ίνες του καλωδίου, η ομαδοποίηση θα γίνει βάσει των διαθέσιμων ινών δια το μέγεθος του μέγιστου διαχωριστή που έχει δηλωθεί, ενώ αν τα κτίρια είναι περισσότερα από τις διαθέσιμες οπτικές ίνες η ομαδοποίηση θα γίνει βάσει του καλωδίου ανεξαρτήτως άλλων στοιχείων.

Με αυτόν τον τρόπο αποτρέπεται το γεγονός να δημιουργηθούν περισσότερες ομάδες από τις διαθέσιμες οπτικές ίνες. Αν για παράδειγμα τα σημεία σύνδεσης είναι 100 και το καλώδιο που έχει οριστεί είναι 144 ινών και στην είσοδο διαχωριστή (splitter) δηλωθεί η τιμή 1, τότε ο αλγόριθμος k-means θα δημιουργήσει ομάδες που περιέχουν μόνο ένα στοιχείο η κάθε μια και θα δημιουργηθεί ένα δίκτυο point to point, δηλαδή κάθε κτίριο θα έχει μια ίνα δεσμευμένη για αυτό.

Αν στην τιμή του διαχωριστή (splitter) δοθεί η τιμή 4 δίνεται στον αλγόριθμο η δυνατότητα να δημιουργήσει ομάδες με μέγεθος έως και τεσσάρων κτιρίων βασιζόμενος στις διαθέσιμες ίνες που υπάρχουν στο καλώδιο. Σε περίπτωση που τα κτίρια είναι περισσότερα από τις διαθέσιμες ίνες και ο χρήστης δηλώσει ότι θέλει να δημιουργηθεί ένα δίκτυο point to point ο αλγόριθμος δεν θα το επιτρέψει και θα πραγματοποιήσει αναγκαστική ομαδοποίηση. Τα σημεία ομαδοποιούνται και όσα είναι στην ίδια ομάδα έχουν το ίδιο ¨cluster id¨.

```
Case
When
@Basic_statistics_for_fields_COUNT/@
splitter <= @Fibercable Then @
Basic_statistics_for_fields_COUNT/@
splitter
else
 @Fibercable
END
```

Εικόνα 4.4-2 Σχέση ¨αριθμός δημιουργίας ομάδων¨



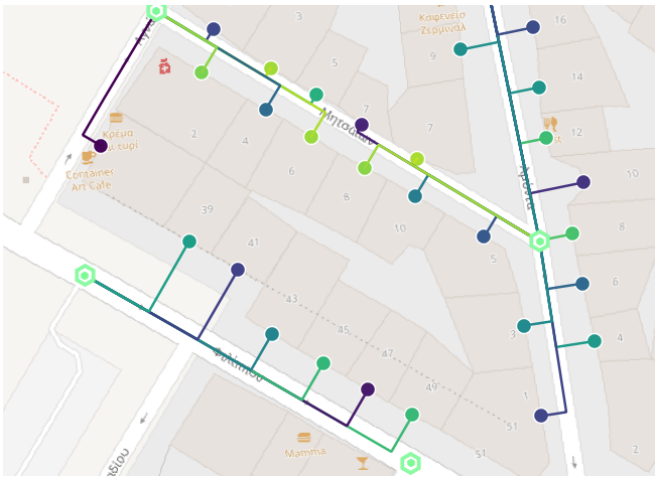
Εικόνα 4.4-3 Ομαδοποίηση στοιχείων – Cluster id

4.4.2 Ομαδοποίηση οδικού δικτύου

Σε αυτό το βήμα γίνεται η ομαδοποίηση των καλωδίων αντλώντας τις πληροφορίες από τις ήδη υπάρχουσες ομάδες των κτιρίων έτσι ώστε να έχουν το ίδιο ¨cluster id¨.

Αρχικά, πραγματοποιείται προληπτικός έλεγχος για σφάλματα στις γεωμετρίες με την χρήση του αλγορίθμου «Fix geometries» (βλ. ενότητα 3.2). Στόχος είναι κάθε όδευση που ομαδοποιείται να έχει το ίδιο ¨cluster id¨ με το σημείο σύνδεσης και για αυτόν τον λόγο καλείται ο αλγόριθμος «join attributes by location»(βλ. ενότητα 3.2). Ο αλγόριθμος θα δεχθεί ως βασική είσοδο το επίπεδο του οδικού δικτύου και ως πεδίο σύγκρισης τον αλγόριθμο «K-means». Με αυτόν τον τρόπο θα διατηρήσει όλα τα πεδία που περιέχουν τις πληροφορίες για το οδικό δίκτυο και για κάθε γραμμή που εφάπτεται με ένα κτίριο θα αντιγράψει όλες τις πληροφορίες από το συγκεκριμένο σημείο, συνεπώς και την πληροφορία του cluster id.

Τέλος, με την χρήση του αλγορίθμου «Retain fields» (βλ. ενότητα 3.2) θα διατηρηθούν μόνο τα πεδία ενδιαφέροντος και θα διαγραφούν τα υπόλοιπα.



Εικόνα 4.4-4 Κοινό Cluster id καλωδίων και σημείων

Συνοψίζοντας, σε αυτό το κεφάλαιο πραγματοποιούνται τα βήματα της δημιουργίας του δικτύου, ξεκινώντας με την δημιουργία της βάσης δεδομένων του οδικού δικτύου και των κτιριακών υποδομών.

Για την δημιουργία της βάσης γίνεται λήψη των δεδομένων από το OSM. Έπειτα, γίνεται η σύνδεση των κτιρίων με το οδικό δίκτυο. Ακολουθεί η μετατόπιση των σημείων από το κέντρο στην πρόσοψη των κτιρίων με σκοπό να ανταποκρίνεται στην πραγματικότητα. Πραγματοποιείται η εύρεση της βέλτιστης διαδρομής σε όλα τα επίπεδα (δίκτυο κορμού, δίκτυο μεταξύ φρεατίων και δίκτυο κτιρίων) αφαιρώντας τις περιττές οδεύσεις, αυτοί οι δρόμοι δηλαδή που δεν θα χρειαστεί να εκτελεστούν έργα. Τέλος, ακολουθεί η ομαδοποίηση των στοιχείων και ο διαμοιρασμός των οπτικών ινών ανάλογα με την τοπολογία δικτύου που επιθυμεί ο μηχανικός.

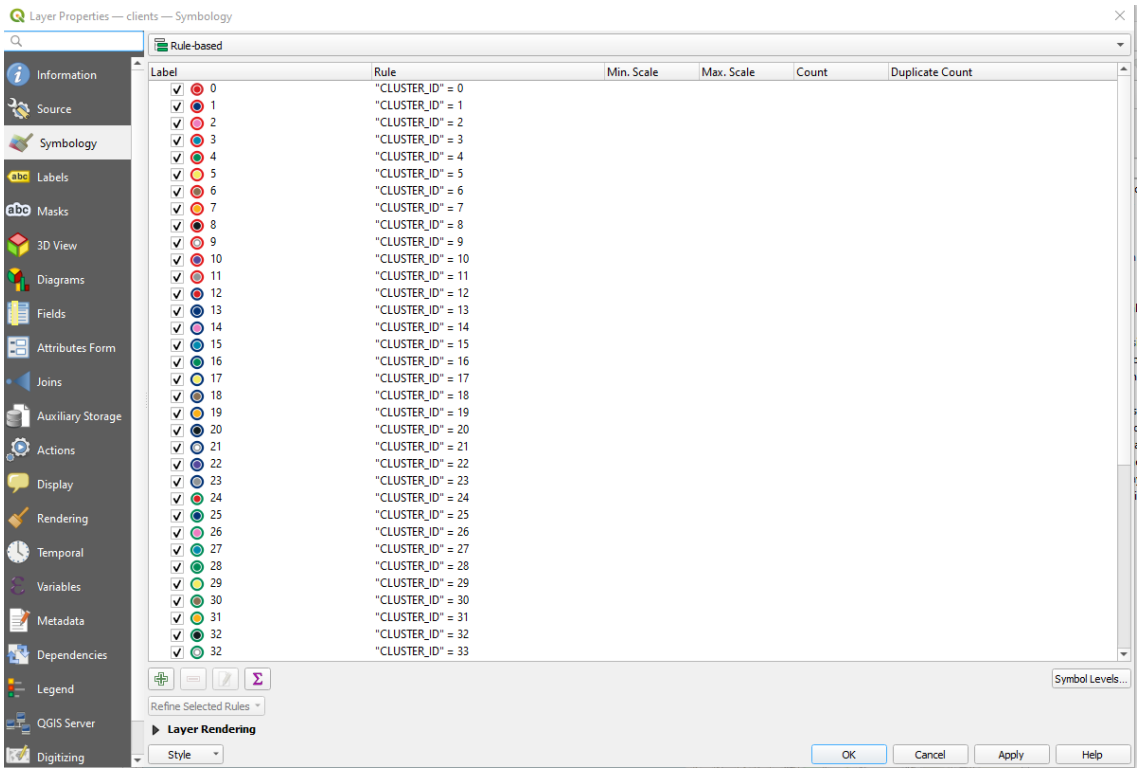
Κεφάλαιο 5ο: Εικονικά πεδία

5.1 Επίπεδο κτιρίων

Για την ευκολότερη και καλύτερη απεικόνιση του δικτύου έχουν δημιουργηθεί κανόνες στα βασικά επίπεδα με την χρήση εικονικών πεδίων (virtual layers). Σκοπός είναι τα επίπεδα να λειτουργούν δυναμικά έτσι ώστε όταν υπάρχουν αλλαγές σε κάποια πεδία από τον χρήστη να αλλάζουν και τα άμεσα σχετιζόμενα πεδία αυτόματα.

Σε αυτό το επίπεδο έχει δημιουργηθεί ένα style layer το οποίο έχει σκοπό την απεικόνιση των σημείων που αντιπροσωπεύουν τα κτίρια με βάση την ίνα που έχουν συνδεθεί. Έχει οριστεί για κάθε ομάδα ¨cluster id¨ ο αντίστοιχος συνδυασμός χρωμάτων βάσει του χρωματικού κώδικα που χρησιμοποιεί ο πάροχος.

Ο εξωτερικός κύκλος αντιπροσωπεύει το χρώμα του σωληνίσκου του καλωδίου και ο εσωτερικός το χρώμα της οπτικής ίνας. Επειδή το style layer βασίζεται στο ¨cluster id¨ του επιπέδου δίνει το πλεονέκτημα όταν αλλάξει ο χρήστης χειροκίνητα ένα στοιχείο και το μεταφέρει σε άλλη ομάδα, αυτόματα να αλλάξει και το χρώμα του στοιχείου βάσει της ομάδας που έχει μεταφερθεί.



Εικόνα 5.1-1 Δημιουργία Style layer για τα σημεία

Από τα προηγούμενα βήματα (βλ. ενότητα 4.4.2), κατά την ομαδοποίηση των στοιχείων δημιουργείται το πεδίο ¨cluster size¨ το οποίο περιέχει τον αριθμό των στοιχείων που περιλαμβάνονται στην κάθε ομάδα. Αυτό το πεδίο είναι στατικό, δηλαδή αν γίνει διαγραφή ενός στοιχείου ή μεταφερθεί σε άλλη ομάδα ο αριθμός των στοιχείων παραμένει εσφαλμένα ο ίδιος. Αυτό μπορεί να δημιουργήσει

προβλήματα στον διαμοιρασμό των οπτικών ινών αλλά και στον μετέπειτα υπολογισμό των παθητικών στοιχείων. Για τον λόγο αυτό δημιουργήθηκε το εικονικό πεδίο με όνομα ¨cluster_s¨, το οποίο μετράει όλα τα στοιχεία που εμπεριέχει το κάθε cluster id συνεχώς και ανανεώνει το σύνολο των στοιχείων κάθε φορά που αλλάζει.

```
count("CLUSTER_ID",group_by:="CLUSTER_ID")
```

Εικόνα 5.1-2 Εικονικό πεδίο – Cluster_s

Εκτός από την απεικόνιση του χρώματος του σημείου, δημιουργήθηκε και ένα πεδίο το οποίο δίνει λεκτικά το χρώμα της ίνας με σκοπό την αποφυγή λαθών. Με αυτόν τον τρόπο μπορεί ο χρήστης να διαβάσει το χρώμα επιλέγοντάς το στοιχείο, αλλά και να εξάγει τα δεδομένα σε excel. Και σε αυτό το πεδίο σε περίπτωση που αλλάξει η ομάδα του στοιχείου αλλάζει και το λεκτικό κομμάτι του πεδίου αυτόματα.

```
CASE
WHEN CLUSTER_ID = 0 THEN
'Red_Tube_Red_Fiber'
WHEN CLUSTER_ID = 1 THEN
'Red_Tube_Blue_Fiber'
WHEN CLUSTER_ID = 2 THEN
'Red_Tube_Pink_Fiber'
WHEN CLUSTER_ID = 3 THEN
'Red_Tube_Aqua_Fiber'
WHEN CLUSTER_ID = 4 THEN
'Red_Tube_Green_Fiber'
WHEN CLUSTER_ID = 5 THEN
'Red Tube Yellow Fiber'
```

Εικόνα 5.1-3 Κώδικας λεκτικής απεικόνισης του χρώματος της ίνας

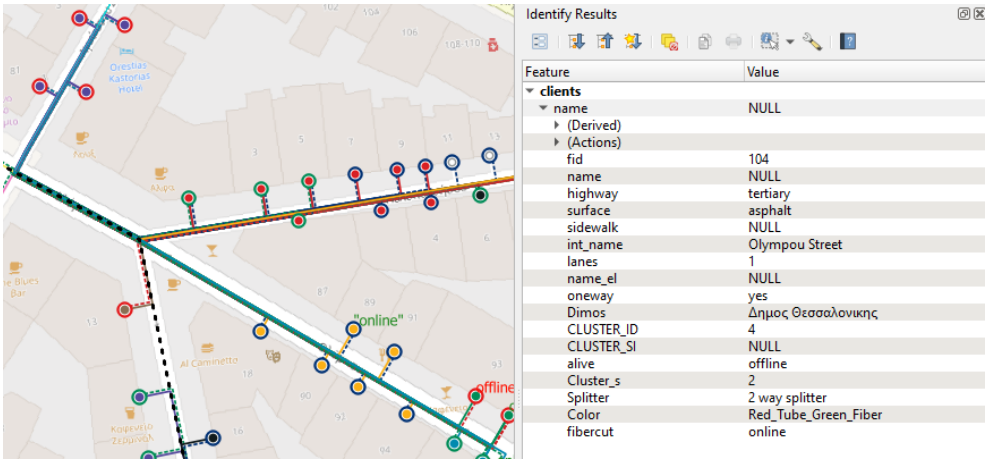
Βασιζόμενοι στο προηγούμενο εικονικό πεδίο και με βάση το πλήθος των στοιχείων που περιέχει κάθε ομάδα υπολογίζεται το μέγεθος του διαχωριστή που θα πρέπει να μπει για να εξυπηρετήσει την ομάδα.

```
CASE
WHEN Cluster_S <= 2 THEN '2 way splitter'
WHEN Cluster_S = 3  THEN '3 way splitter'
WHEN Cluster_S = 4  THEN '4 way splitter'
WHEN Cluster_S = 5  THEN '6 way splitter'
WHEN Cluster_S = 6  THEN '6 way splitter'
WHEN Cluster_S = 7  THEN '8 way splitter'
WHEN Cluster_S = 8  THEN '8 way splitter '
WHEN Cluster_S > 8  THEN '32 way splitter'
END
```

Εικόνα 5.1-4 Κώδικας υπολογισμού διακλαδωτή

Πολύ σημαντικό ακόμα είναι ότι με την χρήση των εικονικών πεδίων είναι πολύ εύκολη η αναζήτηση συγκεκριμένων δεδομένων στον χάρτη. Για παράδειγμα, δίνεται η δυνατότητα με την χρήση φίλτρων

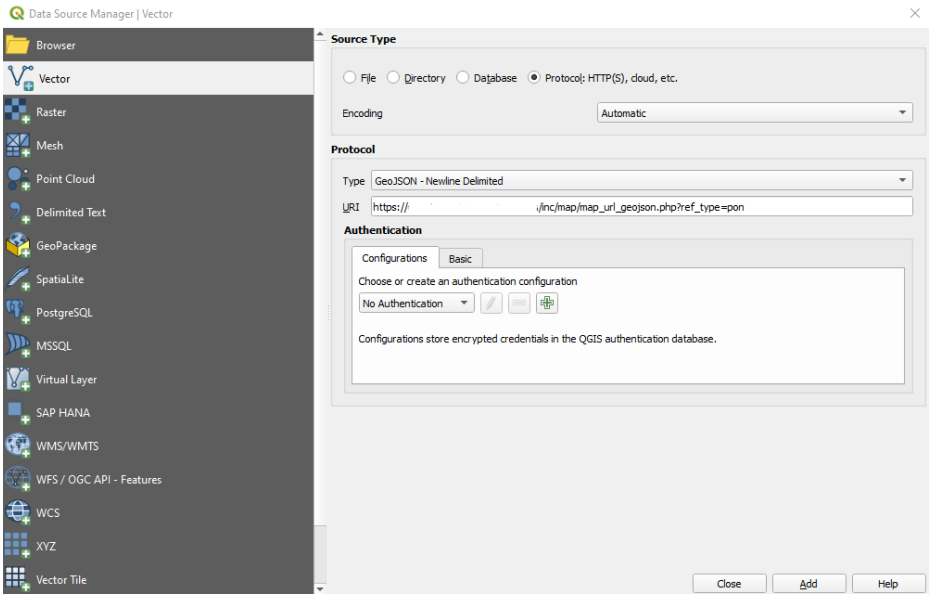
στο επίπεδο ενδιαφέροντος να απεικονιστούν μόνο τα κτίρια τα οποία είναι συνδεδεμένα με διαχωριστές τεσσάρων εξόδων.



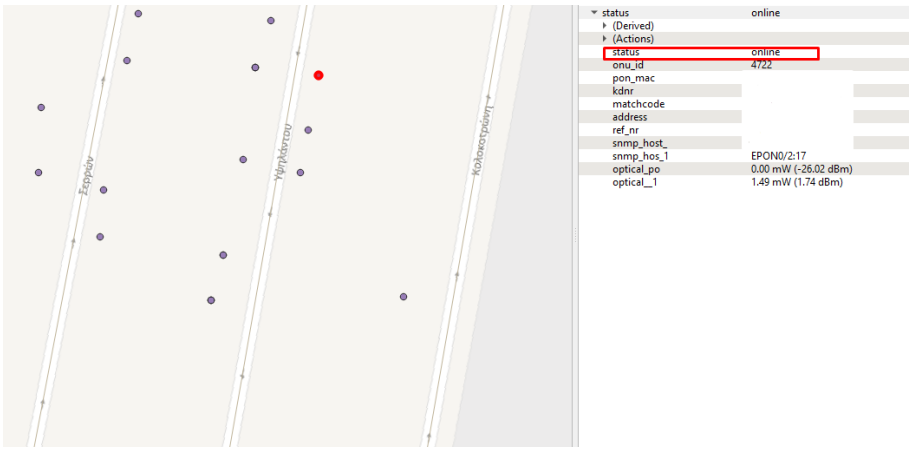
Εικόνα 5.1-5 Απεικόνιση κτιρίων και πληροφορίες εικονικών πεδίων

5.2 Πρόβλεψη βλάβης

Ένα από τα πιο χρήσιμα εργαλεία είναι ο έγκαιρος εντοπισμός μιας πιθανής βλάβης σε κάποια ίνα του δικτύου. Για τον λόγο αυτό έχει δημιουργηθεί το παρακάτω εργαλείο. Έχει δοκιμαστεί σε πραγματικές συνθήκες χρησιμοποιώντας ένα geojson link το οποίο προέρχεται από το διαχειριστικό πρόγραμμα του παρόχου. Το geojson link περιέχει τις συντεταγμένες των σημείων σύνδεσης τα οποία απεικονίζονται επάνω στον χάρτη και επιπλέον τις πληροφορίες του κάθε στοιχείου όπως την κατάσταση του onu (online/offline), το οπτικό σήμα, την διεύθυνση, το όνομα, τον αριθμό πελάτη κτλ.



Εικόνα 5.2-1 Σύνδεση με πραγματικό διαχειριστικό πρόγραμμα



Εικόνα 5.2-2 Δεδομένα από το πραγματικό διαχειριστικό πρόγραμμα

Στην περίπτωση του λογισμικού που έχει δημιουργηθεί με σκοπό την εργασία, επειδή δεν επιτρέπεται να χρησιμοποιηθούν πραγματικά στοιχεία καθώς εμπεριέχουν ευαίσθητα προσωπικά δεδομένα πελατών, έχουν δημιουργηθεί δυο νέα πεδία. Ένα που ονομάζεται ¨alive¨, στο οποίο μπορεί ο χρήστης να δηλώνει την κατάσταση του modem (online/offline) χειροκίνητα και ένα που ονομάζεται ¨fibercut¨, το οποίο ελέγχει όλο το πλήθος των στοιχείων για κάθε ομάδα και μόνο όταν όλα τα στοιχεία έχουν στο πεδίο ¨alive¨, την τιμή ¨offline¨, δίνει στο εικονικό πεδίο ¨fibercut¨ την τιμή ¨offline¨. Όταν συμβεί αυτό τότε τα στοιχεία που έχουν πάρει την τιμή ¨offline¨ στο πεδίο ¨fibercut¨ θα αρχίσουν να αναβοσβήνουν στον χάρτη.

```
case
when
sum("alive"='offline', group_by:=
"CLUSTER_ID") = Cluster_s
then 'offline'
else
'online'
end
```

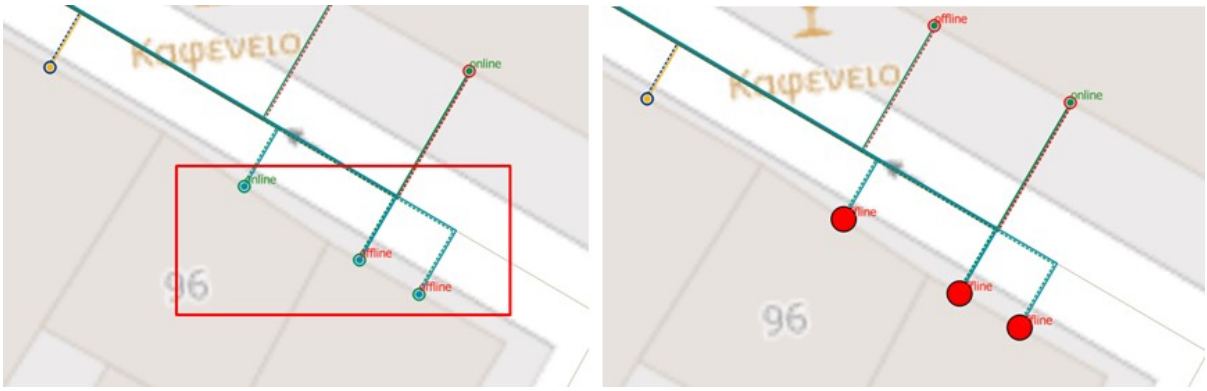
Εικόνα 5.2-3 Κατάσταση offline όταν όλα τα στοιχεία της ομάδας είναι offline

Για να μπορέσει να πραγματοποιηθεί το εφέ που κάνει τα στοιχεία να αναβοσβήνουν, στο επίπεδο styler layer έχει δημιουργηθεί ένα νέο στοιχείο με όνομα ¨Blink¨ το οποίο όποτε πάρει το πεδίο ¨fibercut¨ την τιμή ¨offline¨ αλλάζει την μορφή απεικόνισης του στοιχείου στον χάρτη.

Με τον κώδικα που του έχει δοθεί πραγματοποιείται το εφέ (βλ. εικόνα 5.2-5). Τέλος, για να μπορεί να γίνει ορατό το εφέ, ο χρήστης πρέπει να δηλώσει και το rendering του επιπέδου έτσι ώστε να κάνει ανανέωση κάθε 0.10s.

```
if(second(now()) % 5,'#00FF0000',
'#FFFF0000')
```

Εικόνα 5.2-4 Κώδικας εφέ παλμού



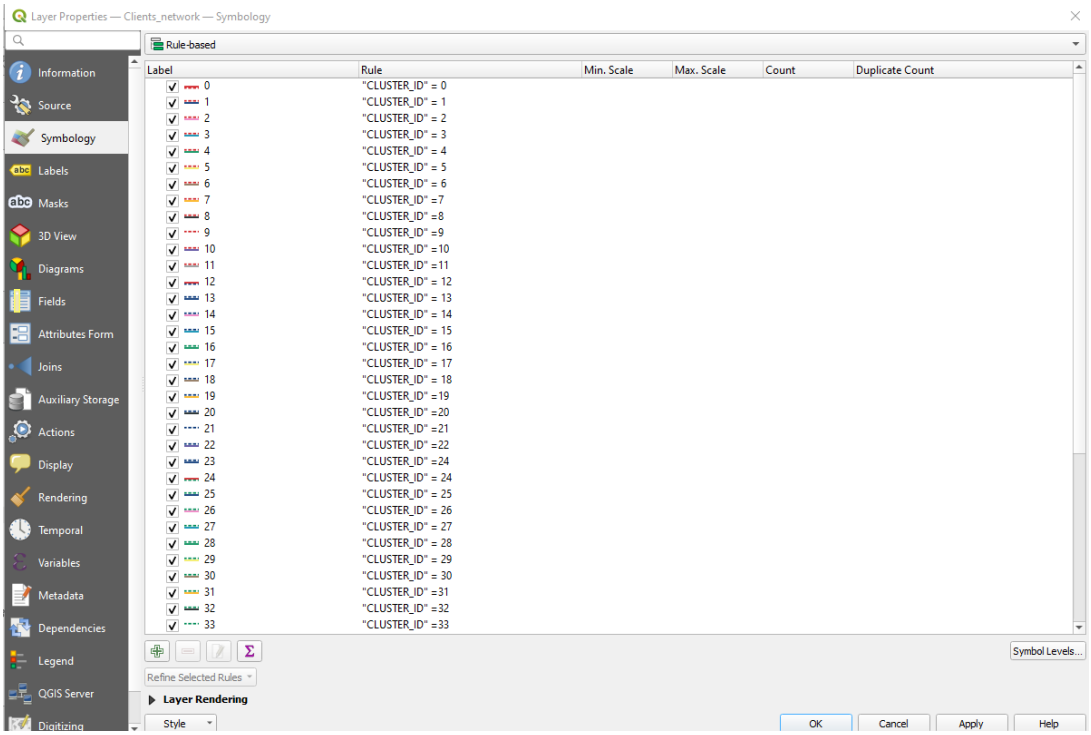
Εικόνα 5.2-5 Απεικόνιση εφέ στον χάρτη

Με αυτόν τον τρόπο σε πραγματικές συνθήκες, όταν μια ομάδα τερματικών που ανήκουν στην ίδια οπτική ίνα έχουν χάσει την επικοινωνία με το κέντρο, τα σημεία που αντιπροσωπεύουν τις συνδέσεις στον χάρτη αναβοσβήνουν. Αυτό δίνει την δυνατότητα στο τμήμα υποστήριξης, να μπορεί να ελέγξει άμεσα, αν έχει δημιουργηθεί κάποια βλάβη από συνεργείο που δουλεύει στην περιοχή ή αν πρέπει να μεταβεί κάποιος στο σημείο. Επίσης, ο κανόνας αυτός μπορεί να γίνει και με επιπλέον συνθήκες, όπως για παράδειγμα αν αλλάξει το οπτικό σήμα κατά 5db σε όλα τα τερματικά της ομάδας, να υπάρχει η ίδια ειδοποίηση, καθώς μπορεί να μην έχει χαθεί η επικοινωνία με το κέντρο, αλλά να έχει λυγίσει κάποια ίνα με αποτέλεσμα να υπολειτουργούν. Στην περίπτωση της εργασίας, αυτός ο τρόπος δεν πραγματοποιήθηκε, καθώς δεν υπάρχουν δεδομένα εισόδου που να δίνουν τις τιμές του οπτικού σήματος και δεν υπάρχει βάση δεδομένων που να αποθηκεύει την προηγούμενη κατάσταση του οπτικού σήματος για να μπορεί να γίνει η σύγκριση.

5.3 Επίπεδο δικτύου

Και σε αυτό το επίπεδο έχει δημιουργηθεί ένα style layer το οποίο έχει σκοπό την απεικόνιση των δρόμων όπως και στο επίπεδο των κτιρίων. Έχει οριστεί για κάθε ¨cluster id¨ ο αντίστοιχος συνδυασμός χρωμάτων με βάση τον χρωματικό κώδικα που χρησιμοποιεί ο πάροχος, με την διακεκομμένη γραμμή να αντιπροσωπεύει το χρώμα του σωληνίσκου και με την συνεχόμενη το χρώμα της ίνας.

Ομοίως και σε αυτό το επίπεδο επειδή το style layer βασίζεται στο ¨cluster id¨ του επιπέδου, αν αλλάξει ο χρήστης χειροκίνητα ένα στοιχείο, αυτόματα θα αλλάξει και το χρώμα του στοιχείου βάσει του ¨cluster id¨ που μεταφέρθηκε.



Εικόνα 5.3-1 Δημιουργία Style layer για τους δρόμους

Σε αυτό το επίπεδο έχει δημιουργηθεί ένα πεδίο που σκοπό έχει να μετράει από πόσα φρεάτια περνάει ένα καλώδιο. Γνωρίζοντας αυτή την πληροφορία, επιλέγοντας ένα καλώδιο μπορεί να προσδιοριστεί ο χρόνος που θα χρειαστεί για να ολοκληρωθούν οι εργασίες συγκόλλησης από το κέντρο μέχρι το επιθυμητό σημείο. Είναι μια πολύ χρήσιμη πληροφορία για το τμήμα που ασχολείται με τον προγραμματισμό των εργασιών των τεχνικών πεδίου, καθώς όταν υπάρχουν πολλές ομάδες αυτή η διαδικασία απαιτεί πολύ χρόνο. Με αυτόν τον τρόπο επιλέγοντας ένα καλώδιο μπορούν να υπολογισθούν οι μέρες εργασίας που θα απαιτηθούν για την ολοκλήρωση του δικτύου.

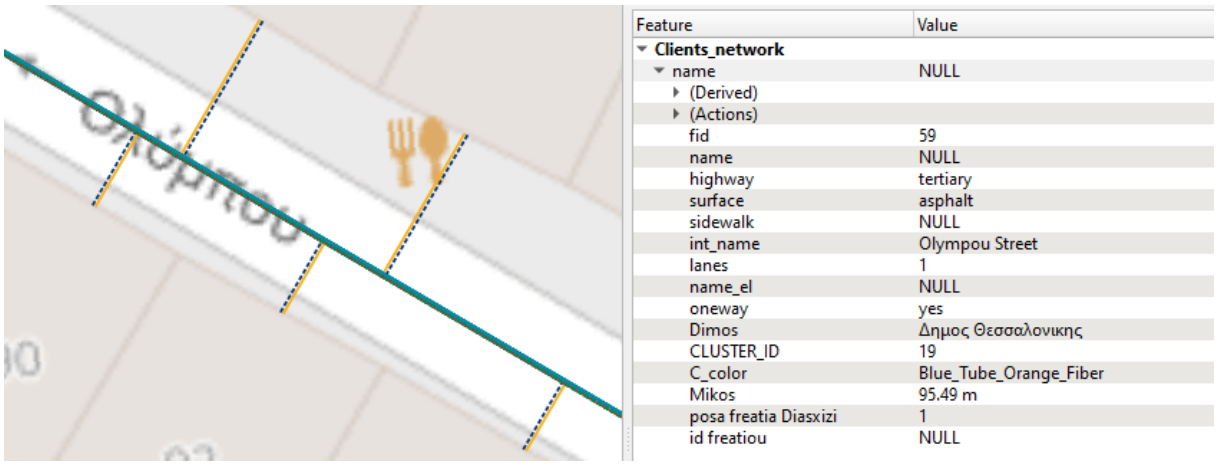
```
aggregate(layer:='freatia', aggregate:=
'count
', expression:="number",  filter:=
intersects
( buffer($geometry, 0.5), geometry(@parent
)))
```

Εικόνα 5.3-2 Κώδικας μέτρησης φρεατίων επάνω στο καλώδιο

Δημιουργήθηκε ακόμα ένα εικονικό πεδίο το οποίο δίνει την τιμή του μήκους του κάθε καλωδίου. Είναι μια πολύ χρήσιμη πληροφορία για καλώδια μεγάλων αποστάσεων ώστε να μπορούν να υπολογισθούν οι απώλειες του οπτικού σήματος που θα υπάρξουν. Επίσης, είναι μια πολύ χρήσιμη πληροφορία σε περίπτωση βλάβης. Όταν γίνει η μέτρηση με εργαλείο OTDR (Optical time-domain reflectometer) με την χρήση φίλτρου στον πίνακα στοιχείων μπορεί να γίνει απεικόνιση μόνο των στοιχείων που έχουν το μήκος ενδιαφέροντος.

```
Round($length,2)  || ' m'
```

Εικόνα 5.3-3 Κώδικας μήκους καλωδίου



Εικόνα 5.3-4 Απεικόνιση δρόμων και πληροφορίες εικονικών πεδίων

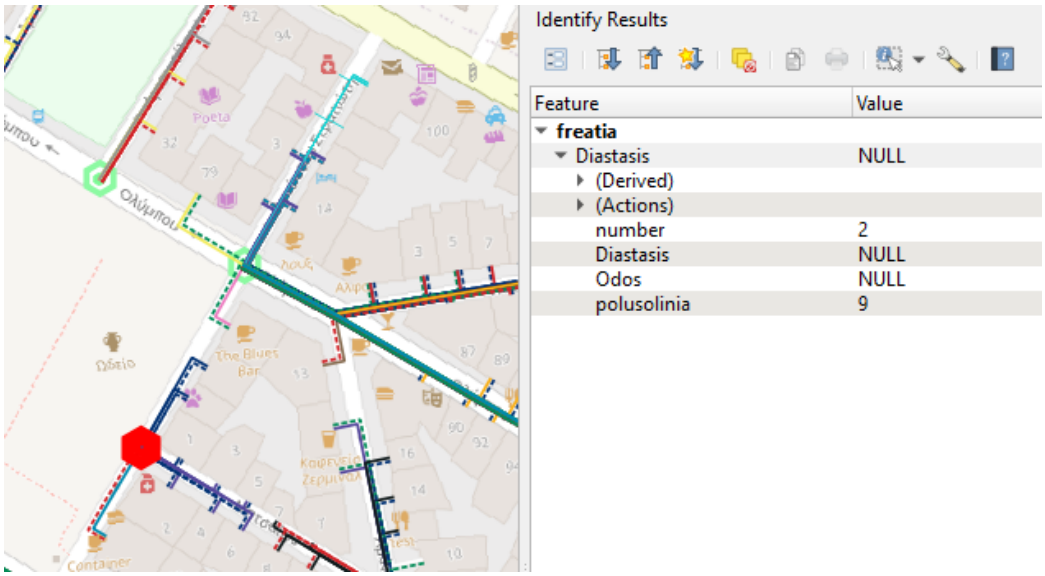
5.4 Φρεάτια

Για το επίπεδο των φρεατίων έχουν δημιουργηθεί δυο εικονικά πεδία. Το πρώτο καλεί την συνάρτηση geom_to_wkt($geometry) η οποία επιστρέφει τις συντεταγμένες κάθε γεωμετρίας. Το δεύτερο μετράει πόσες γραμμές από το επίπεδο του οδικού δικτύου καταλήγουν σε κάθε φρεάτιο. Με αυτόν τον τρόπο μπορεί να γίνει μέτρηση των καλωδίων που καταλήγουν στα φρεάτια. Στόχος είναι να γνωρίζει ο χρήστης τι είδους τερματική μούφα θα χρειαστεί κατά τις εργασίες συγκόλλησης.

```
aggregate(layer:='Clients_network',
aggregate:='count
', expression:="fid",  filter:= intersects
( buffer($geometry,5), geometry(@parent)))
```

Εικόνα 5.4-1 Κώδικας καλωδίων που καταλήγουν σε κάθε φρεάτιο

Είναι ένα πεδίο που βοηθάει τόσο τον μηχανικό στο να γνωρίζει τα υλικά που θα πρέπει να έχει μαζί του στο δίκτυο, όσο και το τμήμα που διαχειρίζεται την αποθήκη στο να γνωρίζει εκ των προτέρων τα υλικά που θα πρέπει να προμηθευτεί για ένα καινούργιο έργο.



Εικόνα 5.4-2 Απεικόνιση φρεατίων και πληροφορίες εικονικών πεδίων

Η δημιουργία εργαλείων είναι από τα πιο χρήσιμα στοιχεία ενός λογισμικού καταγραφής δικτύων, καθώς βοηθούν τόσο στη γρήγορη αποκατάσταση βλαβών, όσο και στον σωστό προγραμματισμό των εργασιών με τον οποίο επιτυγχάνεται η μείωση του χρόνου απασχόλησης του ανθρωπίνου δυναμικού.

Στο κεφάλαιο αυτό έχει γίνει η επεξήγηση της λειτουργίας και της χρησιμότητας των εικονικών πεδίων, μετατρέποντας τα σε εργαλεία απεικόνισης και πρόβλεψης βλαβών. Έχουν δημιουργηθεί styles layer για την εύκολη ¨ανάγνωση¨ του δικτύου και παρέχεται η λειτουργία δυναμικών πεδίων.

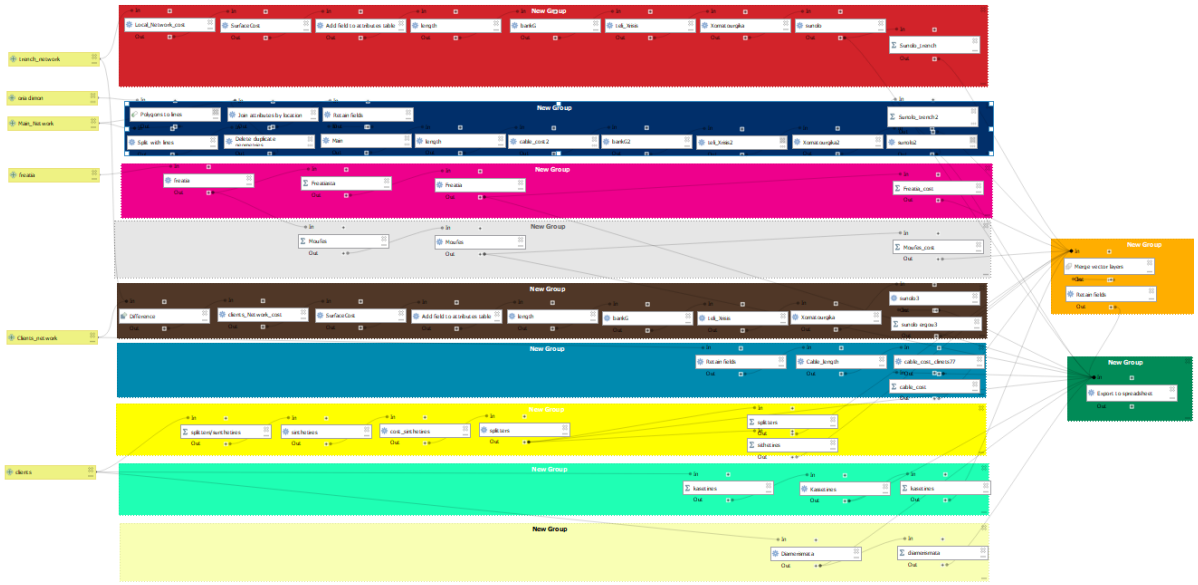
Κεφάλαιο 6ο: Εξαγωγή δεδομένων σε Excel

6.1 Μοντέλο εξαγωγής δεδομένων

Σε αυτό το επίπεδο, με την χρήση πολλαπλών «field calculator» (βλ. ενότητα 3.2.1 – 3.2.7) εξάγονται όλα τα αποτελέσματα για τα υλικά που θα χρειαστούν στην κατασκευή του δικτύου, αλλά και για τα προσδοκόμενα έσοδα του έργου. Ο χρήστης θα πρέπει να δώσει τις παρακάτω έξι εισόδους οι οποίες αφορούν το δίκτυο κορμού,

- το δίκτυο ένωσης μεταξύ φρεατίων
- το δίκτυο πελατών
- το επίπεδο φρεατίων
- το επίπεδο που περιλαμβάνει τους πελάτες
- τα όρια των δήμων.

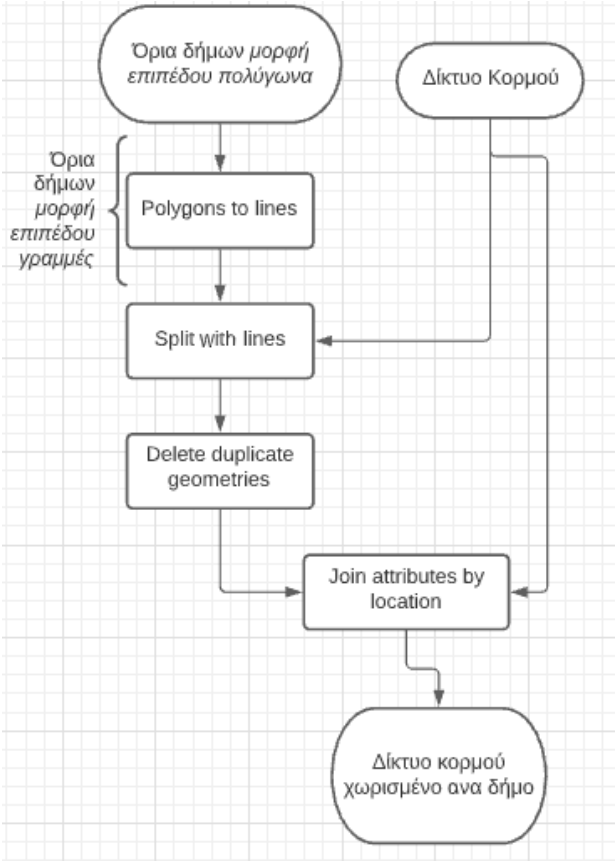
Οι τιμές που θα δοθούν για να υπολογισθεί το κόστος και το κέρδος είναι πλασματικές και δεν ανταποκρίνονται στην πραγματικότητα. (Πλήρης κώδικας – Παράρτημα ΣΤ)



Εικόνα 6.1-1 Μοντέλο εξαγωγής δεδομένων σε excel

6.2 Δίκτυο κορμού

Για το δίκτυο κορμού δηλώνεται ως είσοδος το επίπεδο του δικτύου κορμού και τα όρια των δήμων. Ο λόγος που χρησιμοποιείται σε αυτό το μοντέλο το επίπεδο με τα όρια των δήμων είναι επειδή ενδέχεται το δίκτυο κορμού να περνάει μεταξύ διαφορετικών δήμων, κατά συνέπεια πρέπει όταν αλλάζει ο δήμος να αλλάζουν και τα κριτήρια υπολογισμού του κόστους κατασκευής.

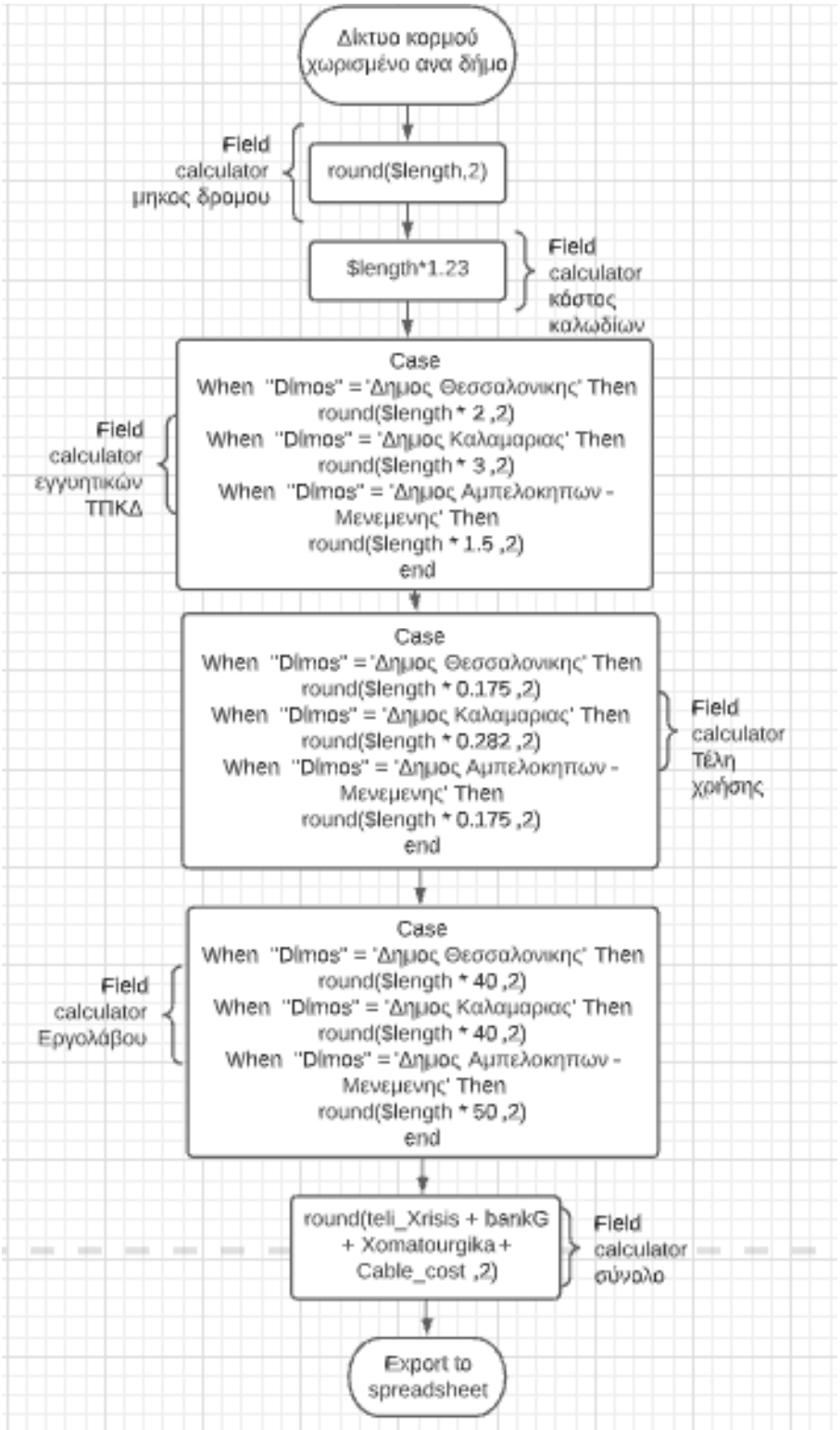


Σχήμα 6.2-1 Διάγραμμα ροής τμηματοποίησης δικτύου κορμού ανά δήμο

Καλείται ο αλγόριθμος «Polygons to lines» ο οποίος παίρνει σαν είσοδο ένα επίπεδο πολυγώνων και από αυτό εξάγει τις περιμέτρους των πολυγώνων ως επίπεδο γραμμών. Ο λόγος αυτής της μετατροπής είναι επειδή ο αλγόριθμος «Split with lines» μπορεί να δεχθεί ως εισόδους μόνο επίπεδα γραμμών.

Αφού έχει γίνει η αλλαγή του επιπέδου σε γραμμές καλείται ο αλγόριθμος «Split with lines». Ο αλγόριθμος θα δεχθεί ως βασική είσοδο το επίπεδο του δικτύου κορμού και ως επίπεδο διαχωρισμού το επίπεδο ορίων του δήμου σε γραμμές. Ο αλγόριθμος χωρίζει τις γραμμές όπου υπάρχουν σημεία τομής μεταξύ των δυο επιπέδων, εξάγοντας τα σε ένα νέο επίπεδο.

Έπειτα καλείται ο αλγόριθμος «Delete duplicate geometries» (βλ. ενότητα 4.1.3) να διαγράψει εσφαλμένες γεωμετρίες που μπορεί να έχουν δημιουργηθεί κατά την έξοδο του αλγορίθμου «Split with lines». Κατά την έξοδο του αλγορίθμου «Delete duplicate geometries» τα πεδία που περιέχουν τις πληροφορίες ενδιαφέροντος έχουν διαγραφεί, για αυτόν τον λόγο καλείται ο αλγόριθμος «Join attributes by location» (βλ. ενότητα 3.2) ώστε να αντληθούν και πάλι οι πληροφορίες ενδιαφέροντος από το βασικό επίπεδο του δικτύου κορμού. Σε αυτό το σημείο έχει επιτευχθεί ο διαχωρισμός του δικτύου κορμού στα σύνορα του κάθε δήμου.



Σχήμα 6.2-2 Διάγραμμα ροής υπολογισμού κόστος

Έπειτα θα γίνει χρήση αριθμομηχανών μέσω του αλγορίθμου «Field calculator» για να υπολογιστούν τα κόστη που θα προκύψουν από την κατασκευή του έργου. Αυτά είναι, το κόστος των καλωδίων όπου υπολογίζεται από το μήκος των καλωδίων πολλαπλασιαζόμενο με την τιμή κόστους ανά μέτρο, οι

εγγυητικές τραπέζης που κατατίθενται στο ταμείο παρακαταθηκών και δανείων, τα τέλη διέλευσης χρήσης του οδικού δικτύου και το κόστος του εργολάβου για τις χωματουργικές εργασίες.

Τέλος, υπολογίζεται το συνολικό κόστος κατασκευής για το δίκτυο κορμού και γίνεται η εξαγωγή του σε ένα φύλλο excel με την χρήση του αλγορίθμου «Export to spreadsheet» ο οποίος έχει την δυνατότητα να εξάγει κάθε επίπεδο σε ξεχωριστό φύλλο.

	A	B	C	D	E	F	G	H
1	fid	Dimos	length	cable_cost	bankG	teli_Xrisis	Xomatourgika	sunolo ergou
2	1	Δημος Θεσσαλονικης	74.18	91.2382781	222.53	20.92	2967.1	3301.79
3	2	Δημος Καλαμαριας	468.86	576.6938555	937.71	82.05	18754.27	20350.72
4	3	Δημος Θεσσαλονικης	55.53	68.30068094	166.59	15.66	2221.16	2471.71
5	4	Δημος Καλαμαριας	4270.88	5253.188397	8541.77	747.4	170835.4	185377.76
6	5	Δημος Θεσσαλονικης	1294.07	1591.700857	3882.2	364.93	51762.63	57601.46
7	6	Δημος Καλαμαριας	357.32	439.5059701	714.64	62.53	14292.88	15509.56
8								**284,613.00 €**
9								
10								

Εικόνα 6.2-1 Φύλλο excel κεντρικού κορμού

Να σημειωθεί ότι το σύνολο δεν μπορεί να παραχθεί στην ίδια σελίδα από τον αλγόριθμο και ότι έχει υπολογιστεί χειροκίνητα από τα εργαλεία του excel. Επίσης, να διευκρινιστεί ότι το δίκτυο κορμού συνήθως ενώνεται με το υφιστάμενο σε κοντινή απόσταση. Έτσι, το κόστος εκσκαφής είναι πολύ μικρότερο. Στην προκειμένη περίπτωση γίνεται υπολογισμός του δικτύου χωρίς να υπάρχει υφιστάμενο και υπολογίζεται η κατασκευή του μέχρι το κέντρο (Headend) το οποίο βρίσκεται σε μεγάλη απόσταση.

6.3 Δίκτυο μεταξύ φρεατίων

Ομοίως, όπως και στο δίκτυο κορμού, με τον ίδιο τρόπο υπολογίζεται και σε αυτό το επίπεδο το κόστος κατασκευής. Σε αυτήν την περίπτωση δεν χρησιμοποιείται το μοντέλο που ελέγχει αν κάποιος δρόμος αλλάζει δήμο, γιατί συνηθίζεται τα έργα να εκτελούνται εντός των ορίων τους. Ακόμα, αν κάποιος δρόμος τύχει και περνάει μεταξύ διαφορετικών δήμων, τα μέτρα θα είναι λίγα και η αλλαγή του κόστους αμελητέα. Παρόλα αυτά μπορεί να γίνει αντιγραφή του μοντέλου και να προστεθεί με τον ίδιο ακριβώς τρόπο.

Το πεδίο ¨Surface_cost¨ που αναφέρεται στο κόστος βάσει της επιφάνειας του δρόμου δεν προσμετράται στο τελικό σύνολο. Υπάρχει ενδεικτικά με σκοπό την σύγκριση του μήκους του δρόμου σε σχέση με το κόστος κατασκευής.

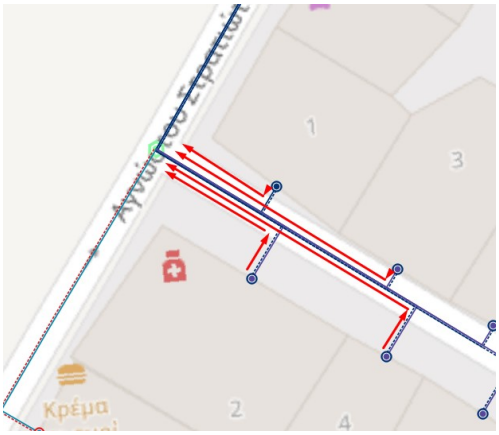
A	B	C	D	E	F	G	H	I
fid	Dimos	SurfaceCost	length	cable_cost	bankG	teli_Xrisis	Xomatourgika	sunolo ergou
1	Δημος Θεσσαλονικης	3900.74	229.46	282.2298319	458.91	40.15	4130.2	4629.26
2	Δημος Θεσσαλονικης	6016.76	353.93	435.3303341	707.85	61.94	6370.69	7140.48
3	Δημος Θεσσαλονικης	2181.53	128.33	157.8401496	256.65	22.46	2309.86	2588.97
4	Δημος Θεσσαλονικης	3117.79	183.4	225.5815554	366.8	32.09	3301.19	3700.08
5	Δημος Θεσσαλονικης	4827.97	284	349.3177232	568	49.7	5111.97	5729.67
7	Δημος Θεσσαλονικης	4615.97	271.53	333.9792009	543.06	47.52	4887.5	5478.08
8	Δημος Θεσσαλονικης	1631.32	95.96	118.0305188	191.92	16.79	1727.28	1935.99
								31,202.53 €

Εικόνα 6.3-1 Φύλλο excel δικτύου ένωσης φρεατίων

6.4 Δίκτυο σύνδεσης κτιρίων

Το δίκτυο σύνδεσης κτιρίων θα υπολογιστεί σε δυο διαφορετικά φύλλα. Ο λόγος είναι ότι τα μέτρα των χωματουργικών εργασιών δεν είναι ίδια με τα μέτρα των καλωδίων που θα χρειαστούν.

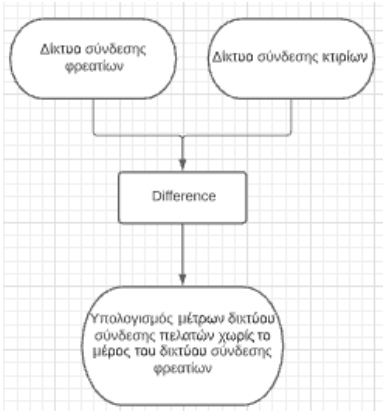
Για παράδειγμα, για την παρακάτω εικόνα τα μέτρα εκσκαφής είναι εκατόν τριάντα, καθώς ο εργολάβος θα κατασκευάσει έναν κοινό χάνδακα όπου θα μπουν όλες οι σωληνώσεις. Ενώ τα μέτρα των καλωδίων που θα χρειαστούν είναι τριακόσια, καθώς κάθε σημείο σύνδεσης καταλήγει στο φρεάτιο.



Εικόνα 6.4-1 Σκάμμα μπλε γραμμή - Καλώδια κόκκινη

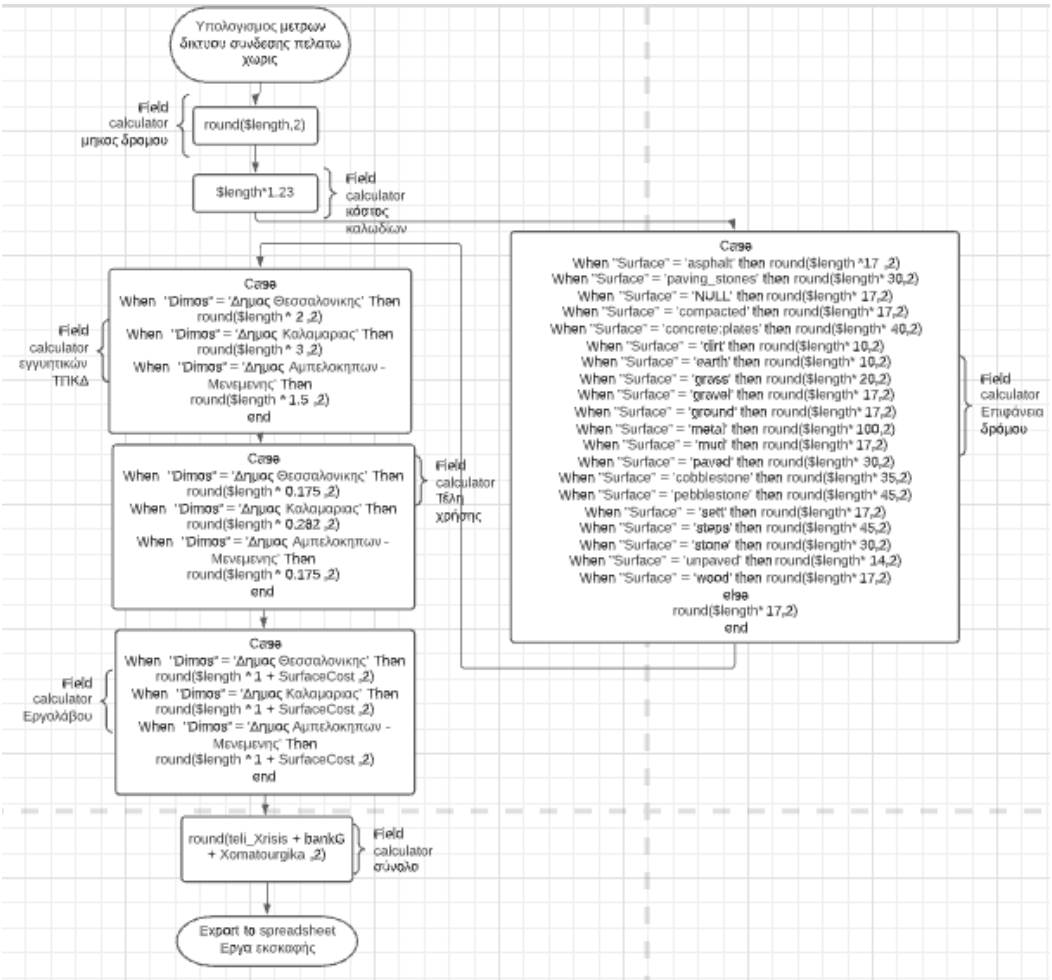
Αν υπολογιστούν τα μέτρα εκσκαφής και από τα δυο επίπεδα, λόγω του ότι το επίπεδο δικτύου σύνδεσης φρεατίων και το επίπεδο σύνδεσης κτιρίων έχουν κοινές οδεύσεις, θα μετρηθούν παραπάνω μέτρα εκσκαφής από αυτά που θα εκτελεστούν στην πραγματικότητα. Έτσι, τα μέτρα καλωδίου θα υπολογισθούν μόνο από το επίπεδο σύνδεσης πελατών, ενώ τα μέτρα εκσκαφής θα υπολογισθούν από τα κοινά σημεία των δυο επιπέδων μαζί με τα υπολειπόμενα μέτρα εκσκαφής των καθέτων τομών προς τα κτίρια.

Για τον λόγο αυτό καλείται ο αλγόριθμος «Difference» ο οποίος θα δεχθεί ως βασική είσοδο το επίπεδο των συνδέσεων των κτιρίων και ως συγκρίσιμο επίπεδο αυτό του επιπέδου των συνδέσεων των φρεατίων. Με αυτό τον τρόπο θα υπολογισθούν οι κοινές οδεύσεις μόνο μια φορά.



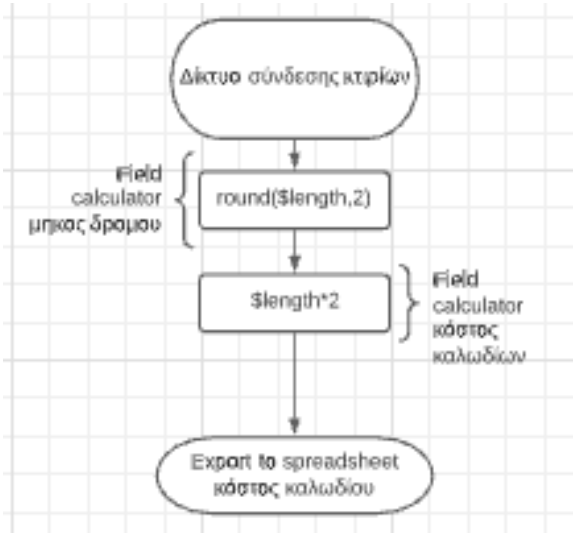
Σχήμα 6.4-1 Διάγραμμα ροής υπολογισμός μέτρων εκσκαφής στο τοπικό δίκτυο

Αφού έχει γίνει ο υπολογισμός των μέτρων εκσκαφής και των καλωδίων που θα χρειαστούν για την κατασκευή του δικτύου, με την χρήση του αλγορίθμου «Field calculator» θα υπολογιστούν τα έξοδα σχετικά με το κόστος εκσκαφής βάσει της επιφάνειας του δρόμου, το κόστος των τελών χρήσης διέλευσης για την χρήση του οδικού δικτύου, οι εγγυητικές εκτέλεσης έργων στο ταμείο παρακαταθηκών και δανείων και το κόστος κατασκευής από τον εργολάβο του έργου. Όλα αυτά θα αθροιστούν σε μια νέα στήλη όπου θα βγει το συνολικό κόστος σε ένα νέο φύλλο excel.



Σχήμα 6.4-2 Διάγραμμα ροής κόστος εκσκαφών χωρίς καλώδια

Σε αυτό το σημείο έχουν υπολογιστεί όλα τα έξοδα κατασκευής δικτύου χωρίς το κόστος των καλωδίων. Το κόστος θα υπολογιστεί από το μήκος του καλωδίου επι την τιμή του ανά μέτρο.



Σχήμα 6.4-3 Διάγραμμα ροής κόστος καλωδίων

Τα αποτελέσματα θα εξαχθούν και πάλι σε ένα νέο φύλλο excel του ιδίου αρχείου το οποίο θα περιέχει το id του κάθε καλωδίου, το μήκος του και το κόστος.

A	B	C	D
fid	name	length	cable_cost
78		17.93	36
79		22.37	45
80		31.32	63
81		35.92	72
82		17.87	36
83		26.4	53
84		47.49	95
85		134.68	269
86		138.6	277
87		21.45	43
88		37.03	74
89		52.97	106
90		35.42	71
91		33.23	66
92		15.97	32
93		48.35	97
94		46.05	92
95		338.39	677
96		343.36	687
97		23.31	47
98		326.35	653
99		378.16	756
100		42.48	85
101		355.92	712
102		367.01	734
			17,609.00 €

Εικόνα 6.4-2 Φύλλο excel δικτύου ένωσης κτιρίων κόστος καλωδίων

6.5 Φρεάτια

Όσον αφορά το κόστος των φρεατίων επειδή όλοι οι δήμοι έχουν τα ίδια κριτήρια και το ίδιο κόστος κατασκευής από τον εργολάβο, δεν υπάρχει λόγος να γίνει η εξαγωγή τους ανά τοποθεσία ή κάποιο άλλο χαρακτηριστικό τους. Συνεπώς για να βγει το συνολικό κόστος φρεατίων καλείται ο αλγόριθμος «Basic statistics for fields» ο οποίος θα μετρήσει το σύνολο των φρεατίων εντός της περιοχής ενδιαφέροντος και έπειτα με την χρήση του αλγορίθμου «field calculator», θα υπολογισθεί το συνολικό

κόστος το οποίο είναι το σύνολο των φρεατίων πολλαπλασιαζόμενο με το κόστος κατασκευής των 450 ευρώ έκαστο (ενδεικτική τιμή).

A	B	C	D
fid	count	Freatia_cost	
1	11	4,950.00 €	

Εικόνα 6.5-1 Κόστος φρεατίων

6.6 Αναλώσιμα

Σε αυτό το φύλλο υπολογίζονται όλοι οι διαχωριστές που θα χρειαστούν για την υλοποίηση του δικτύου όπως και οι συνδετήρες σύνδεσης των οπτικών ινών. Επίσης, σε ξεχωριστά φύλλα υπολογίζονται οι μούφες και οι κασετίνες που θα χρειαστούν. Για τους διαχωριστές καλείται ο αλγόριθμος «Statistics by categories» ο οποίος μετράει σε ένα επίπεδο τα στοιχεία βάσει του δηλωθέντος πεδίου. Σαν πεδίο καταμέτρησης έχει δηλωθεί το ¨Cluster_ID¨ με σκοπό να καταμετρηθούν οι διαχωριστές που θα χρειαστούν ανά ομάδα.

Συνδετήρες είναι τα παθητικά στοιχεία με τα οποία συνδέονται δυο απολήξεις οπτικών ινών. Αφού γίνει η καταμέτρηση των διαχωριστών μπορεί από την έξοδο να καταμετρηθούν και οι συνδετήρες που θα χρειαστούν στο δίκτυο. Ο αριθμός τους υπολογίζεται από τον αριθμό των εξόδων του διαχωριστή με επιπλέον έναν συνδετήρα για την είσοδο του διαχωριστή. (κώδικας του «filed calculator» στην εικόνα (6.6-1).)

```
Case
When "splitter" = '2 way splitter' then 3
When "splitter" = '3 way splitter' then 4
When "splitter" = '4 way splitter' then 5
When "splitter" = '6 way splitter' then 7
When "splitter" = '8 way splitter' then 9
When "splitter" = '32 way splitter' then 33
end
```

Εικόνα 6.6-1 Κώδικας υπολογισμού συνδετήρων

Έπειτα καλούνται ακόμα δυο «field calculator» από τους οποίους ένας υπολογίζει το κόστος του κάθε διαχωριστή και ένας το κόστος του κάθε συνδετήρα.

Εξαγωγή δεδομένων σε excel

A	B	C	D	E	F	G
fid	CLUSTER_ID	Splitter	count	sinthetires	cost_sinthetires	splitters_cost
1	7	4 way splitter	4	5	2.5	6
2	13	3 way splitter	3	4	2	15
3	26	2 way splitter	1	3	1.5	4
4	9	2 way splitter	1	3	1.5	4
5	22	6 way splitter	5	7	3.5	8
6	23	4 way splitter	4	5	2.5	6
7	20	3 way splitter	3	4	2	15
8	3	2 way splitter	1	3	1.5	4
9	15	2 way splitter	2	3	1.5	4
10	8	4 way splitter	4	5	2.5	6
11	11	2 way splitter	2	3	1.5	4
12	33	3 way splitter	3	4	2	15
13	18	3 way splitter	3	4	2	15
14	10	4 way splitter	4	5	2.5	6
15	24	4 way splitter	4	5	2.5	6
16	25	4 way splitter	4	5	2.5	6
17	29	2 way splitter	1	3	1.5	4
18	5	2 way splitter	2	3	1.5	4
19	34	2 way splitter	2	3	1.5	4
20	6	2 way splitter	1	3	1.5	4
21	12	6 way splitter	5	7	3.5	8
22	21	2 way splitter	2	3	1.5	4
23	32	2 way splitter	2	3	1.5	4
24	17	2 way splitter	2	3	1.5	4
25	19	6 way splitter	6	7	3.5	8
26	1	2 way splitter	2	3	1.5	4
27	31	2 way splitter	1	3	1.5	4
28	27	3 way splitter	3	4	2	15
29	4	2 way splitter	2	3	1.5	4
30	0	4 way splitter	4	5	2.5	6
31	16	3 way splitter	3	4	2	15
					62.50 €	**216.00 €**

Εικόνα 6.6-2 Φύλλο excel κόστους διακλαδωτών και συνδετήρων

Οι μούφες τερματισμού των καλωδίων είναι ίδιες με τον αριθμό των φρεατίων, οπότε στην έξοδο του αλγορίθμου «Basic statistics for fields» που έχει μετρήσει ήδη τα φρεάτια, θα μπει ένας «field calculator» ο οποίος θα είναι το άθροισμα των φρεατίων πολλαπλασιαζόμενο με το κόστος της μιας μούφας.

A	B	C	D
fid	count	Moufes_cost	
1	11	**3,300.00 €**	

Εικόνα 6.6-3 Φύλλο Excel κόστους για μούφες

Και το κόστος των κασετινών (τερματικών κιτίων) δεν επηρεάζεται από άλλους παράγοντες, συνεπώς υπολογίζεται από το σύνολο των κτιρίων πολλαπλασιαζόμενο με το κόστος της κάθε κασετίνας.

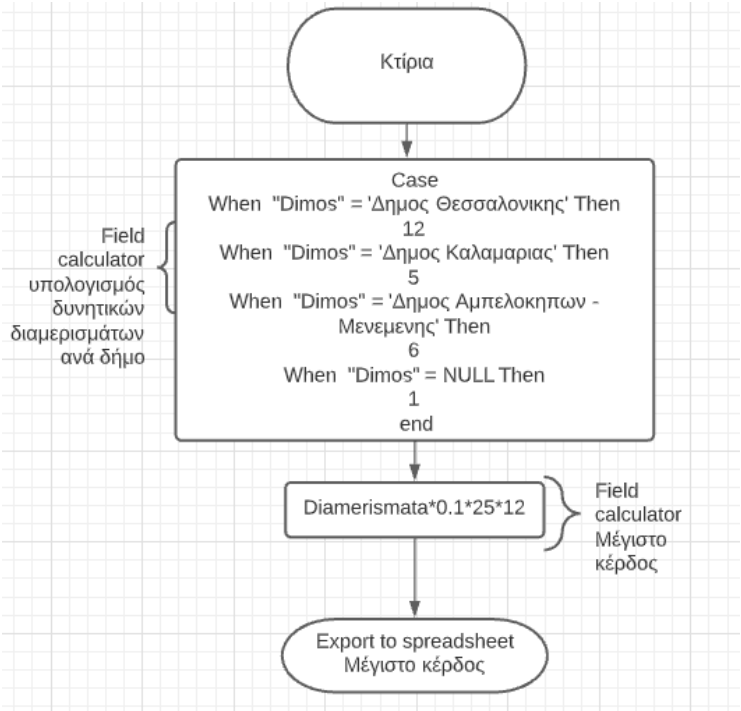
A	B	C	D
fid	count	kasetines	
1	86	1,290.00 €	

Εικόνα 6.6-4 Φύλλο excel κόστους κασετινών

6.7 Υπολογισμός μέγιστου κέρδους

Για το μέγιστο κέρδος υπολογίζεται ένα 10% από το σύνολο των διαμερισμάτων εντός της περιοχής ενδιαφέροντος πολλαπλασιαζόμενο με το κόστος μιας μέσης σύνδεσης των εικοσιπέντε ευρώ, επι 12 μήνες ώστε να βγει το ετήσιο κέρδος.

Αρχικά καλείται ο αλγόριθμος «field calculator» ο οποίος έχει ως είσοδο το επίπεδο των κτιρίων , ελέγχει σε ποιο δήμο ανήκει το κτίριο και υπολογίζει τα πιθανά διαμερίσματα που υπάρχουν. Με την χρήση ενός δεύτερου «field calculator» θα γίνει ο υπολογισμός του μέγιστου κέρδους και η έξοδος αυτού θα δοθεί στον αλγόριθμο «Export to spreadsheet» ώστε να εξάγει τα αποτελέσματα σε ένα νέο φύλλο.



Σχήμα 6.7-1 Διάγραμμα ροής κέρδους

Σε αυτό το βήμα, σε αντίθεση με το μοντέλο επιλογής ενδιαφέροντος, υπολογίζεται το κέρδος ανά έτος και όχι ανά διετία. Σκοπός είναι στο τέλος να είναι ξεκάθαρο σε ποιο έτος γίνεται η απόσβεση του έργου.

Εξαγωγή δεδομένων σε excel

1	fid	Dimos	Diamerismata	kerdos
61	89	Δημος Θεσσαλονικης	12	1080
62	92	Δημος Θεσσαλονικης	12	1080
63	91	Δημος Θεσσαλονικης	12	1080
64	120	Δημος Θεσσαλονικης	12	1080
65	119	Δημος Θεσσαλονικης	12	1080
66	121	Δημος Θεσσαλονικης	12	1080
67	116	Δημος Θεσσαλονικης	12	1080
68	115	Δημος Θεσσαλονικης	12	1080
69	118	Δημος Θεσσαλονικης	12	1080
70	117	Δημος Θεσσαλονικης	12	1080
71	104	Δημος Θεσσαλονικης	12	1080
72	114	Δημος Θεσσαλονικης	12	1080
73	105	Δημος Θεσσαλονικης	12	1080
74	104	Δημος Θεσσαλονικης	12	1080
75	128	Δημος Θεσσαλονικης	12	1080
76	121	Δημος Θεσσαλονικης	12	1080
77	138	Δημος Θεσσαλονικης	12	1080
78	137	Δημος Θεσσαλονικης	12	1080
79	140	Δημος Θεσσαλονικης	12	1080
80	139	Δημος Θεσσαλονικης	12	1080
81	131	Δημος Θεσσαλονικης	12	1080
82	132	Δημος Θεσσαλονικης	12	1080
83	130	Δημος Θεσσαλονικης	12	1080
84	130	Δημος Θεσσαλονικης	12	1080
85	129	Δημος Θεσσαλονικης	12	1080
86	129	Δημος Θεσσαλονικης	12	1080
87	129	Δημος Θεσσαλονικης	12	1080
88			1004	**92,880.00 €**

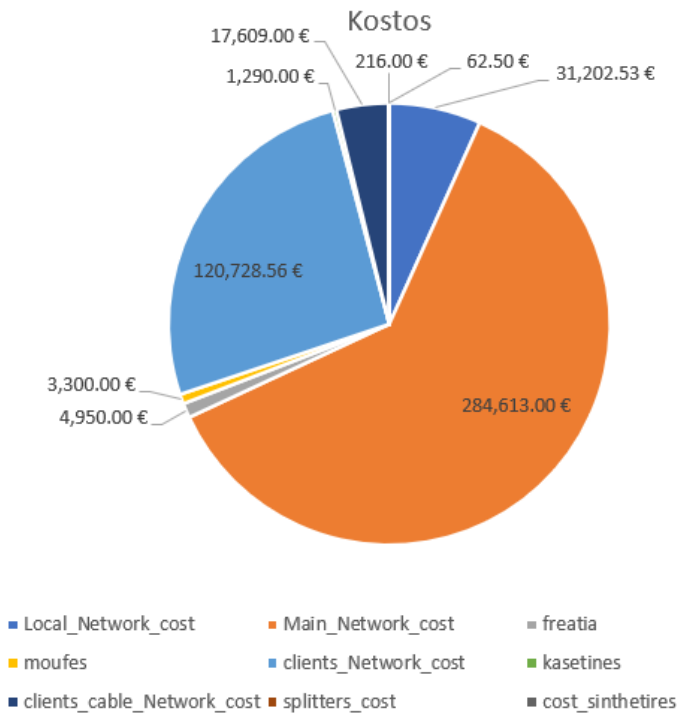
Εικόνα 6.7-1 Ετήσιο δυνητικό κέρδος επένδυσης

6.8 Σχέση κόστους/κέρδους

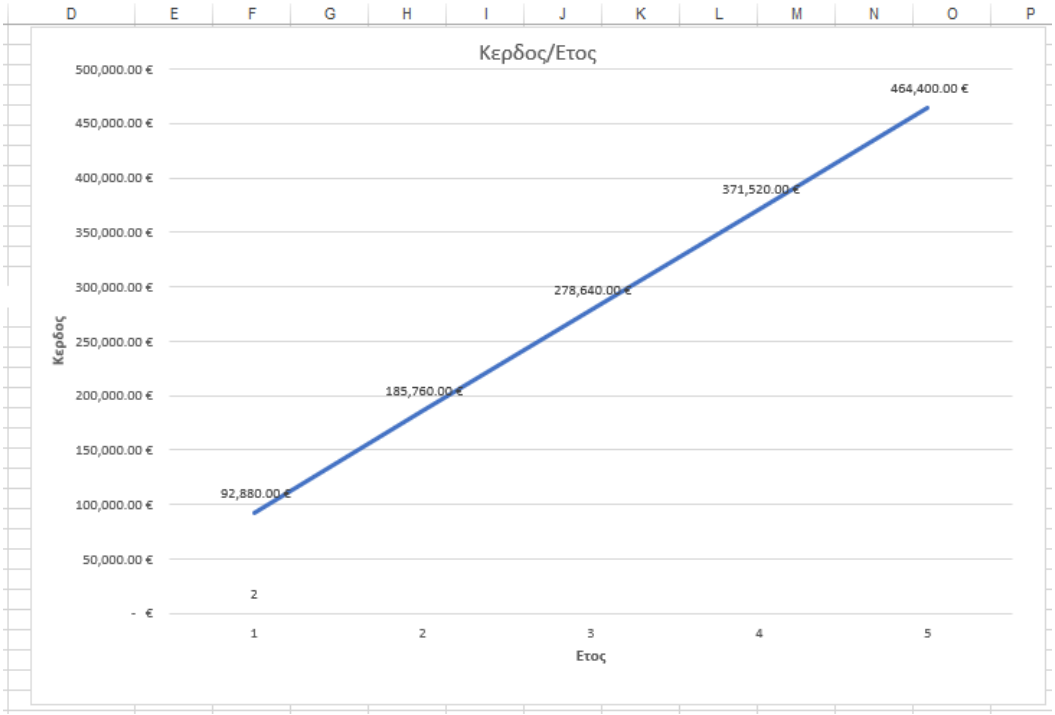
Τέλος, από τα αποτελέσματα που έχουν προκύψει, σε όλα τα επίπεδα, μπορεί ο μηχανικός να δημιουργήσει μια σχέση κόστους/κέρδους και να ελέγξει ενδεχομένως σε ποια κατηγορία το κόστος είναι πολύ μεγάλο ώστε να προβεί σε αλλαγές, αν αυτό είναι εφικτό. Ακόμα, μπορεί να υπολογίσει σε ποιο έτος το έργο θα κάνει απόσβεση και θα ξεκινήσει να παράγει έσοδα.

Katigoria	**Kostos**
Local_Network_cost	31,202.53 €
Main_Network_cost	284,613.00 €
freatia	4,950.00 €
moufes	3,300.00 €
clients_Network_cost	120,728.56 €
kasetines	1,290.00 €
clients_cable_Network_cost	17,609.00 €
splitters_cost	216.00 €
cost_sinthetires	62.50 €
	463,971.59 €
Etos	**kerdos**
1	**92,880.00 €**
2	**185,760.00 €**
3	**278,640.00 €**
4	**371,520.00 €**
5	**464,400.00 €**

Εικόνα 6.8-1 Συνολικό κόστος / κέρδος ανά έτος



Εικόνα 6.8-2 Κόστος ανά κατηγορία



Εικόνα 6.8-3 Κέρδος ανά έτος

Σε αυτό το κεφάλαιο εξάγονται όλα τα δεδομένα κόστους ανά κατηγορία σε excel. Είναι ένα πολύ σημαντικό εργαλείο υπολογισμού όλου του κόστους ενός έργου, από τα έξοδα εκσκαφής μέχρι και τα αναλώσιμα που θα χρειαστούν για την υλοποίηση. Με αυτόν τον τρόπο μπορούν να γίνουν αλλαγές

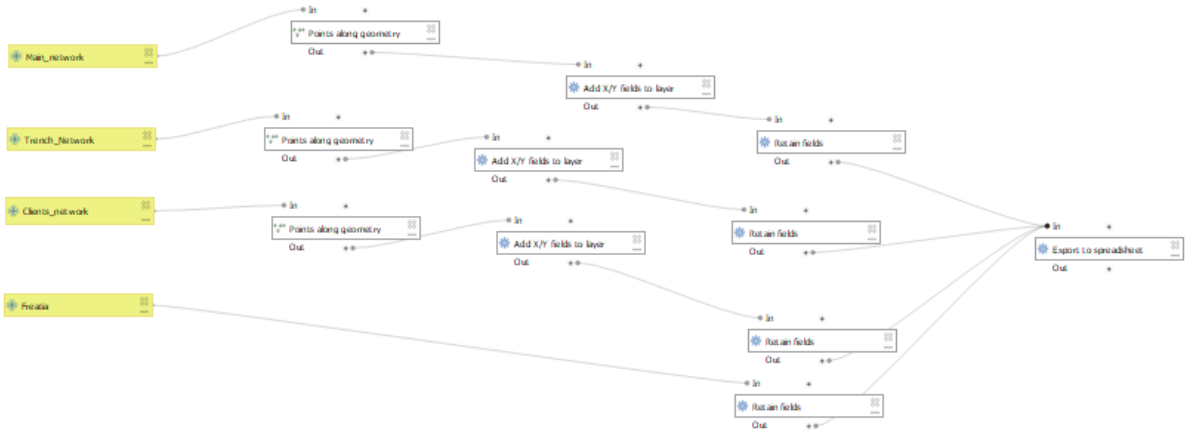
όπου αυτό είναι εφικτό και να βοηθήσουν στην μείωση του κατασκευαστικού κόστους. Χωρίς την χρήση του μοντέλου αυτή η διαδικασία πρέπει να γίνει χειροκίνητα. Είναι μια διαδικασία που απαιτεί πολύ χρόνο καθώς ο όγκος των δεδομένων είναι πολύ μεγάλος και λόγω αυτού υπάρχει μεγάλη πιθανότητα λάθος υπολογισμού.

Κεφάλαιο 7ο: Εξαγωγή συντεταγμένων σε excel

7.1 Σκοπός μοντέλου

Άλλη μια παράμετρος που επηρεάζει την έκδοση αδειών είναι ότι, πέρα από τους δήμους πρέπει να ενημερωθούν και άλλες διάφορες υπηρεσίες ή και πάροχοι όπως, η αρχαιολογία, η τροχαία, οι εταιρίες φυσικού αερίου, οι εταιρίες ηλεκτρικής ενέργειας και οι πάροχοι τηλεπικοινωνιακών δικτύων. Η ενημέρωση αυτή γίνεται είτε για να δοθεί η έγκριση στο να προχωρήσουν τα έργα, είτε για να υπάρξει ενημέρωση για υφιστάμενα δίκτυα. Η διαδικασία απαιτεί μαζί με το αίτημα και ένα αρχείο excel που περιέχει τις συντεταγμένες των δρόμων που θα εκτελεστούν τα έργα.

Το μοντέλο αυτό έχει δημιουργηθεί με σκοπό να εξάγει τις συντεταγμένες των οδεύσεων και των φρεατίων αυτόματα. Από τον χρήστη θα ζητηθεί να δώσει τέσσερις εισόδους οι οποίες είναι, το δίκτυου κορμού, το δίκτυο σύνδεσης των φρεατίων, το δίκτυο από τα φρεάτια στα κτίρια και το επίπεδο των φρεατίων (Πλήρης κώδικας – Παράρτημα Z)



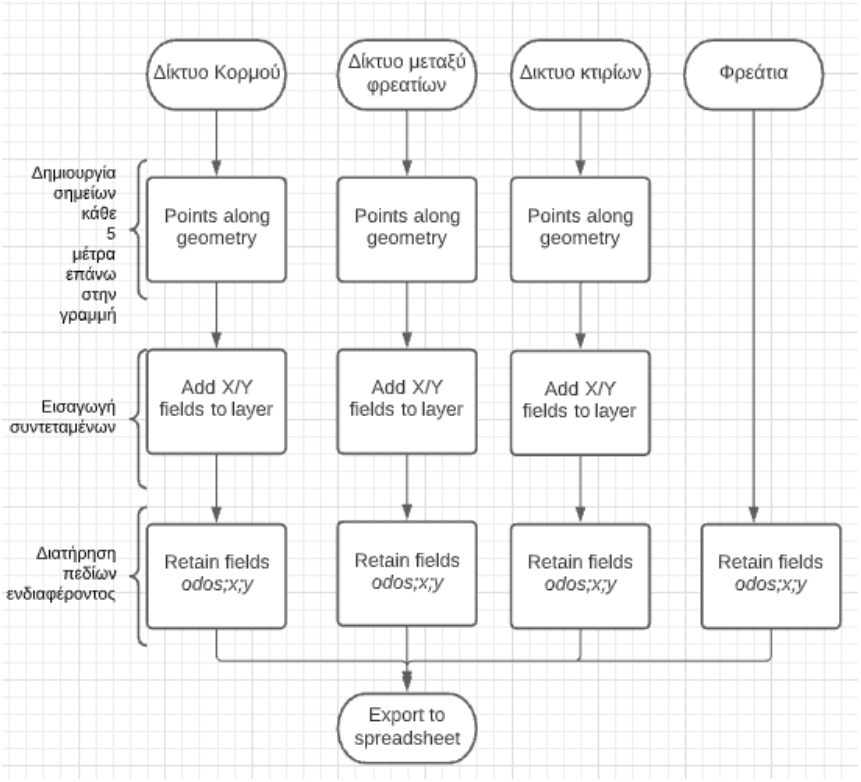
Εικόνα 7.1-1 Μοντέλο εξαγωγής συντεταγμένων

7.2 Λειτουργία μοντέλου

Για τις εισόδους που αφορούν επίπεδα γραμμών (κεντρικό δίκτυο, δίκτυο μεταξύ φρεατίων και δίκτυο σύνδεσης κτιρίων) καλείται ο αλγόριθμος «Points along geometry». Ο αλγόριθμος δημιουργεί ένα επίπεδο σημείων, τα οποία είναι κατανεμημένα επάνω στις γραμμές του επιπέδου εισόδου και η απόσταση μεταξύ τους ορίζεται ως παράμετρος στον κώδικα. Στην παρούσα περίπτωση είναι η απόσταση των 5 μέτρων.

Αφού δημιουργηθούν τα σημεία επάνω στις γραμμές καλείται ο αλγόριθμος «Add X/Y fields to layer» (βλ. ενότητα 4.1.3) ο οποίος θα εισάγει τις συντεταγμένες επάνω σε κάθε σημείο από αυτά που δημιουργήθηκαν. Για το επίπεδο των φρεατίων οι συντεταγμένες υπάρχουν από προηγούμενα βήματα (βλ. ενότητα 5.4) οπότε δεν θα χρειαστεί να ξανά εισαχθούν.

Εφόσον έχουν δημιουργηθεί πλέον τα πεδία ενδιαφέροντος, καλείται ο αλγόριθμος «Retain fields» (βλ. ενότητα 3.2) ώστε να κρατήσει μόνο τα χρήσιμα πεδία για κάθε επίπεδο. Τέλος καλείται ο αλγόριθμος «Export to spreadsheet» ο οποίος θα εξάγει τα αποτελέσματα σε ένα αρχείο excel, το οποίο θα περιέχει τέσσερα φύλλα ένα για κάθε επίπεδο εισόδου.



Σχήμα 7.2-1 Διάγραμμα ροής εξαγωγής συντεταγμένων

fid	name	int_name	x	y
1	Orestias Kastorias		2554446.842	4959034.988
2	Orestias Kastorias		2554442.72	4959037.818
3	Orestias Kastorias		2554442.696	4959041.409
4	Orestias Kastorias		2554445.526	4959045.531
5	Orestias Kastorias		2554448.356	4959049.653
6	Orestias Kastorias		2554451.186	4959053.775
7	Orestias Kastorias		2554454.016	4959057.897
8	Orestias Kastorias		2554643.786	4959045.566
9	Orestias Kastorias		2554640.485	4959047.061
10	Olympou Street		2554635.501	4959046.672
11	Olympou Street		2554630.516	4959046.283
12	Olympou Street		2554625.531	4959045.894
13	Olympou Street		2554620.546	4959045.505
14	Olympou Street		2554615.561	4959045.116
15	Olympou Street		2554610.576	4959044.727
16	Olympou Street		2554605.591	4959044.338
17	Olympou Street		2554600.607	4959043.949
18	Olympou Street		2554595.622	4959043.56
19	Olympou Street		2554590.637	4959043.171
20	Poeta		2554585.652	4959042.782
21	Poeta		2554580.667	4959042.393
22	Poeta		2554575.682	4959042.004
23	Poeta		2554570.697	4959041.616
24	Poeta		2554565.713	4959041.227
25	Poeta		2554560.728	4959040.838
26	Poeta		2554555.743	4959040.449
27	Poeta		2554550.758	4959040.06
28	Poeta		2554545.773	4959039.671
29	Poeta		2554540.788	4959039.282
30	Poeta		2554535.804	4959038.893
31	Poeta		2554530.819	4959038.504

Εικόνα 7.2-1 Έξοδος μοντέλου εξαγωγής συντεταγμένων

Ανακεφαλαιώνοντας, έχει δημιουργηθεί ένα μοντέλο το οποίο δέχεται τα επίπεδα του δικτύου και των φρεατίων ως είσοδο. Έπειτα εισάγει συντεταγμένες στα σημεία και τα εξάγει σε ένα νέο αρχείο excel.

Κεφάλαιο 8ο: Μελλοντικά σχέδια

Το λογισμικό βρίσκεται σε πλήρη ανάπτυξη καθώς ήδη χρησιμοποιείται σε πραγματικές συνθήκες.

Στόχος είναι να υπάρξει συμβατότητα σε όλα τα επίπεδα με το σύστημα διαχείρισης πελατών (CRM) και να μπορεί να ενταχθεί σαν βασικό εργαλείο του συστήματος. Κάποια από τα εργαλεία τα οποία δημιουργούνται είναι ο εμπλουτισμός σε επίπεδο καταγραφής δικτύου. Σκοπός είναι τα στοιχεία να περιέχουν εσωτερικές ¨οντότητες¨, δίνοντας την δυνατότητα περισσότερων συσχετισμών μεταξύ των επιπέδων, ώστε να παρέχει στον χρήστη περισσότερες πληροφορίες.

Έτσι, επιλέγοντας ο χρήστης μια όδευση, να μπορεί να δει όλες τις χρήσιμες πληροφορίες όπως, πόσα και ποια καλώδια περνούν μέσα από κάθε σωλήνα, πόσα παθητικά στοιχεία υπάρχουν στην διαδρομή, ποιο είναι το επίπεδο του σήματος που ξεκινάει από το κέντρο, ποιο είναι το θεωρητικό σήμα που θα έπρεπε να φτάσει και ποιο το πραγματικό. Να αναγράφονται όλες οι συγκολλήσεις και τα φρεάτια/καμπίνες που διασχίζει το καλώδιο κλπ..

Επίσης, η δημιουργία ενός κοινού server που θα φιλοξενεί το λογισμικό για να χρησιμοποιείται από όλα τα τμήματα του παρόχου, (γραμματειακή υποστήριξη, τεχνικό τμήμα, προωθητικό τμήμα κτλ.) με δυνατότητα διαβάθμισης χρηστών διαφορετικών δυνατοτήτων. Δηλαδή, χρήστες μόνο για ανάγνωση ή χρήστες που θα μπορούν να κάνουν τροποποιήσεις στο δίκτυο. Παράλληλα, ο κάθε χρήστης να βλέπει μόνο τις πληροφορίες που είναι σημαντικές για εκείνον. Να παρέχονται επιπλέον καταγραφές (logs) όπως, το όνομα του μηχανικού που έκανε τις αλλαγές και την ημερομηνία που έγιναν.

Ακόμα αυτή την στιγμή δημιουργείται μια βάση δεδομένων, με σκοπό να υπολογίζει με μεγαλύτερη ακρίβεια την δυναμική των δυνητικών πελατών, βασιζόμενη στην ημερομηνία που έχει γίνει η αίτηση, σε σχέση με τις προωθητικές ενέργειες που έχουν προηγηθεί και την ημερομηνία εκτέλεσης έργων στην περιοχή.

Τέλος θα γίνει προσπάθεια ενοποίησης όλων των μοντέλων σε ένα, με σκοπό όλη η διαδικασία της σχεδίασης να γίνεται μόνο σε ένα βήμα.

Είναι ένα εργαλείο το οποίο μπορεί να εμπλουτίζεται συνεχώς και να αναπτύσσεται τόσο σε συνηθισμένες λειτουργίες που είναι απαραίτητες για κάθε πάροχο, όσο και επάνω στις προσωπικές ανάγκες του κάθε μηχανικού.

Κεφάλαιο 9ο: Συμπεράσματα

Για την εξαγωγή αποτελεσμάτων έχει γίνει σύγκριση των δυο τρόπων σχεδίασης, οι οποίοι είναι ο χειροκίνητος σχεδιασμός ενός δικτύου και ο σχεδιασμός με την χρήση του λογισμικού.

Για την χειροκίνητη διαδικασία έχει γίνει η σύγκριση με παλαιότερες μελέτες που έχουν γίνει στο Google earth, το οποίο είναι ένα ελεύθερο λογισμικό που δεν απαιτεί κάποια συνδρομή. Στα πλεονεκτήματά του συγκαταλέγεται το γεγονός ότι γενικά είναι ένα εύχρηστο εργαλείο στο οποίο μπορεί να γίνει η σχεδίαση ενός δικτύου, χωρίς να απαιτεί ιδιαίτερες γνώσεις ως προς την χρήση του από τον μηχανικό.

Μεγάλο μειονέκτημα αποτελεί το γεγονός ότι το πρόγραμμα από μόνο του είναι ¨βαρύ¨ και εφόσον υπάρχουν πολλά δεδομένα η απόκριση του είναι πολύ αργή. Επίσης, δεν μπορεί να υπάρχει αλληλεπίδραση μεταξύ δεδομένων, συνεπώς οποιαδήποτε αλλαγή, πρέπει να γίνεται χειροκίνητα σε όλα τα στοιχεία. Δεν συνιστάται για την χρήση σε σχεδίαση δικτύου σε επαγγελματικό επίπεδο.

Το qgis δίνει πάρα πολλές δυνατότητες σύγκρισης και εξαγωγής αποτελεσμάτων. Δεν θεωρείται ένα εύκολο κατά την χρήση του λογισμικό όπως και τα περισσότερα της κατηγορίας του, καθώς για κάθε ενέργεια που πρέπει να πραγματοποιηθεί πρέπει να επιλεχθεί το αντίστοιχο εργαλείο αλλά και το αντίστοιχο επίπεδο και αυτό το καθιστά δύσχρηστο.

Όσον αφορά την χειροκίνητη διαδικασία και το Qgis απαιτεί πολύ χρόνο κατά την σχεδίαση του δικτύου αλλά παρουσιάζει πολλά οφέλη στις δυνατότητες του. Σχετικά με το μοντέλο της «επιλογής περιοχής ενδιαφέροντος», δεν μπορεί να καθοριστεί ο χρόνος της χειροκίνητης διαδικασίας που απαιτείται. Καθώς όσο αυξάνονται τα δεδομένα, όπως για παράδειγμα, οι αιτήσεις, τόσο αυξάνεται και ο χρόνος σύγκρισης στο να επιλεχθεί η κατάλληλη περιοχή που θα είναι πιο κερδοφόρα. Είναι ένα από τα πιο εύχρηστα εργαλεία με την αυτόματη διαδικασία να υπερέχει στο 100%, καθώς ο χρόνος εξαγωγής δεδομένων είναι της τάξεως μερικών λεπτών. Πρακτικά χωρίς την χρήση του μοντέλου με τον χειροκίνητο τρόπο μοιάζει πιο πολύ με τυχαία επιλογή.

Για την χειροκίνητη σχεδίαση συνήθως ένα έργο πέντε χιλιομέτρων απαιτεί, σχεδόν δέκα μέρες αποκλειστικής εργασίας στο κομμάτι της σχεδίασης, μια βδομάδα στον τοπικό έλεγχο ώστε να γίνουν υποδείξεις πιθανών αλλαγών και μια βδομάδα επανασχεδίασης των αλλαγών.

Με την χρήση του λογισμικού υπάρχει μείωση του χρόνου από δέκα μέρες σε λίγα λεπτά. Για τον επιτόπιο έλεγχο και τις διορθώσεις δεν μπορεί να βγει ασφαλές συμπέρασμα, καθώς αφενός το δειγματοληπτικό υλικό της σύγκρισης είναι μικρό και αφετέρου οι συνθήκες είναι διαφορετικές, καθώς οι μηχανικοί που πραγματοποιούν τους επιτόπιους ελέγχους έχουν και επιπλέον εργασίες, με αποτέλεσμα μια μελέτη να καθυστερεί και από εξωγενείς παράγοντες. Οπότε θα θεωρηθεί ότι ο χρόνος είναι ίδιος.

Το σίγουρο είναι ότι στο κομμάτι της αρχικής σχεδίασης, υπάρχει μεγάλη μείωση του χρόνου σε σχέση με την χειροκίνητη διαδικασία. Θα μπορούσε να είναι ακόμα μεγαλύτερη η διαφορά, αφού σε μια χειροκίνητη σχεδίαση ενός μεγαλύτερου δικτύου, της τάξεως των είκοσι χιλιομέτρων, ο χρόνος θα ανέβαινε στο τετραπλάσιο, αντιθέτως στην αυτόματη σχεδίαση θα ήταν ο ίδιος.

Στο επίπεδο του κόστους κατασκευής δεν υπάρχουν μεγάλες διαφορές καθώς και στις δυο περιπτώσεις τα κόστη που ελέγχθηκαν είναι αναλογικά τα ίδια.

Ούτε στον τομέα της κάλυψης υπήρχαν διαφορές, καθώς όποια πιθανά λάθη υπήρξαν κατά την αρχική σχεδίαση, εντοπίστηκαν και διορθώθηκαν από τον επιτόπιο έλεγχο, με αποτέλεσμα να υπάρχει κάλυψη 100% και στις δυο περιπτώσεις. Εδώ μπορεί να παρατηρηθεί μόνο ότι κατά τον σχεδιασμό με το λογισμικό υπάρχει το πλεονέκτημα της αποφυγής του ανθρωπίνου λάθους.

Για την χρήση των εικονικών πεδίων δεν υπάρχει συγκρίσιμο μέγεθος καθώς κανένα ελεύθερο πρόγραμμα δεν παρέχει αυτήν την δυνατότητα.

Η εξαγωγή σε excel είναι άλλη μια λειτουργία που δεν υπήρχε καθόλου σε αυτοματοποιημένη μορφή παρά την χειροκίνητη και χρονοβόρα διαδικασία.

Το μοντέλο εξαγωγής συντεταγμένων, είναι και αυτό ένα από τα πιο χρήσιμα εργαλεία, καθώς η διαδικασία για να περαστούν οι συντεταγμένες μιας μελέτης (πέντε χιλιομέτρων) σε ένα αρχείο excel χειροκίνητα, απαιτεί τουλάχιστον δέκα μέρες εργασίας. Με αυτό το μοντέλο, τα αποτελέσματα εξάγονται αυτόματα μέσα σε λίγα δευτερόλεπτα και χωρίς την πιθανότητα λάθους. Επίσης, μεγάλο πλεονέκτημα είναι ότι μπορεί να αλλάξει το EPSG σε περίπτωση που χρειαστεί.

Συνεπώς το μεγαλύτερο όφελος όλων είναι ο χρόνος που κερδίζει το ανθρώπινο δυναμικό του παρόχου, τόσο κατά την επιλογή της περιοχής, την σχεδίαση του δικτύου και την εξαγωγή αποτελεσμάτων σε excel όσο και στην πρόβλεψη και την αποφυγή λαθών με την χρήση των εργαλείων αυτοδιάγνωσης και των εικονικών πεδίων.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Internet Site

[1] https://www.tanea.gr/2019/01/21/science-technology/ksekina-to-megalytero-ergo-sdit-700-ekat-eyro-stis-tilepikoinonies/

[2] https://www.sfbb.gr/

[3] https://www.sunvizion.com/products/network-inventory#network-planning

[4] https://comsof.com/fiber/

[5] https://biarrinetworks.com/

[6] https://www.3-gis.com/fiber-network-planning-management-software-3-gis

[7] https://www.qgis.org/en/site/

[8]https://www.qgis.org/en/site/about/case_studies/australia_distance_learning.html

[9] https://www.uleth.ca/artsci/introductory-gis-qgis

[10] https://www.geomeletitiki.gr/qgis-%CF%84%CE%BF-%CF%80%CE%B9%CE%BF-%CE%B9%CF%83%CF%87%CF%85%CF%81%CF%8C-%CE%BB%CE%BF%CE%B3%CE%B9%CF%83%CE%BC%CE%B9%CE%BA%CF%8C-%CE%B1%CE%BD%CE%BF%CE%B9%CF%87%CF%84%CE%BF%CF%8D-%CE%BA%CF%8E%CE%B4/

[11] https://wiki.openstreetmap.org/wiki/Map_features

[12] https://plugins.qgis.org/plugins/networks/

[13] https://lucid.app/

[14] https://www.aspexit.com/en/coordinate-reference-systems/

[15] https://gis.stackexchange.com

[16] https://docs.qgis.org/3.16/en/docs/training_manual/index.html

Webinars

[1] Fiber to any Home - Faster FTTH Planning with Comsof Fiber and Design Gateway https://www.youtube.com/watch?v=8Q5zugVfNsM&ab_channel=RealworldSystems

[2] Qgis webinar https://www.youtube.com/channel/UCGS162t4hkOA0b35ucf1yng

[3] Biarri Networks Webinars https://www.youtube.com/watch?v=A3W8hPR-IQM&ab_channel=BiarriNetworks

[4] [Webinar] Generating An FTTH Design and BOM in 5 minutes https://www.youtube.com/watch?v=--vYN_ZZWko&ab_channel=BiarriNetworks

ΠΑΡΑΡΤΗΜΑ Α : Επιλογή περιοχής Ενδιαφέροντος

"""

Model exported as python.

Name : Step_1_perioxi_endiaferontos

Group : Gekas_Dimitris

With QGIS : 31802

"""

```
from qgis.core import QgsProcessing
from qgis.core import QgsProcessingAlgorithm
from qgis.core import QgsProcessingMultiStepFeedback
from qgis.core import QgsProcessingParameterVectorLayer
from qgis.core import QgsProcessingParameterFeatureSource
from qgis.core import QgsProcessingParameterFeatureSink
from qgis.core import QgsProcessingParameterBoolean
from qgis.core import QgsCoordinateReferenceSystem
from qgis.core import QgsExpression
import processing


class Step_1_perioxi_endiaferontos(QgsProcessingAlgorithm):


 def initAlgorithm(self, config=None):
  self.addParameter(QgsProcessingParameterVectorLayer('City', 'Oria_dimou', types=[QgsProcessing.TypeVectorPolygon], defaultValue=None))
  self.addParameter(QgsProcessingParameterFeatureSource('12321', 'Odiko_Diktio_thessalonikis', types=[QgsProcessing.TypeVectorAnyGeometry], defaultValue=None))
  self.addParameter(QgsProcessingParameterVectorLayer('Demandpoints', 'Aitisis', types=[QgsProcessing.TypeVectorPoint], defaultValue=None))
  self.addParameter(QgsProcessingParameterVectorLayer('olataktiria', 'Ktiria_thessalonikis', types=[QgsProcessing.TypeVectorPoint], defaultValue=None))
  self.addParameter(QgsProcessingParameterFeatureSink('Grid', 'Grid', type=QgsProcessing.TypeVectorAnyGeometry, createByDefault=True, supportsAppend=True, defaultValue=None))
```

```
 self.addParameter(QgsProcessingParameterFeatureSink('', '1', type=QgsProcessing.TypeVectorAnyGeometry, createByDefault=True, supportsAppend=True, defaultValue=None))
 self.addParameter(QgsProcessingParameterFeatureSink('', '2', type=QgsProcessing.TypeVectorAnyGeometry, createByDefault=True, supportsAppend=True, defaultValue=None))
 self.addParameter(QgsProcessingParameterBoolean('VERBOSE_LOG', 'Verbose logging', optional=True, defaultValue=False))

 def processAlgorithm(self, parameters, context, model_feedback):
 # Use a multi-step feedback, so that individual child algorithm progress reports are adjusted for the
 # overall progress through the model
 feedback = QgsProcessingMultiStepFeedback(28, model_feedback)
 results = {}
 outputs = {}

 # Reproject layer
 alg_params = {
  'INPUT': parameters['olataktiria'],
  'OPERATION': '',
  'TARGET_CRS': QgsCoordinateReferenceSystem('EPSG:4326'),
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['ReprojectLayer'] = processing.run('native:reprojectlayer', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(1)
 if feedback.isCanceled():
  return {}

 # Reproject layer
 alg_params = {
  'INPUT': parameters['City'],
  'OPERATION': '',
  'TARGET_CRS': QgsCoordinateReferenceSystem('EPSG:4326'),
```

```
 'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['ReprojectLayer'] = processing.run('native:reprojectlayer', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(2)
 if feedback.isCanceled():
  return {}

 # Reproject layer
 alg_params = {
  'INPUT': parameters['12321'],
  'OPERATION': '',
  'TARGET_CRS': QgsCoordinateReferenceSystem('EPSG:4326'),
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['ReprojectLayer'] = processing.run('native:reprojectlayer', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(3)
 if feedback.isCanceled():
  return {}

 # Fix geometries
 alg_params = {
  'INPUT': outputs['ReprojectLayer']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['FixGeometries'] = processing.run('native:fixgeometries', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(4)
 if feedback.isCanceled():
  return {}
```

```
 # Create grid
 alg_params = {
  'CRS': 'ProjectCrs',
  'EXTENT': outputs['ReprojectLayer']['OUTPUT'],
  'HOVERLAY': 0,
  'HSPACING': 600,
  'TYPE': 3,
  'VOVERLAY': 0,
  'VSPACING': 600,
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['CreateGrid'] = processing.run('native:creategrid', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(5)
 if feedback.isCanceled():
  return {}

 # Reproject layer
 alg_params = {
  'INPUT': parameters['Demandpoints'],
  'OPERATION': '',
  'TARGET_CRS': QgsCoordinateReferenceSystem('EPSG:4326'),
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['ReprojectLayer'] = processing.run('native:reprojectlayer', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(6)
 if feedback.isCanceled():
  return {}

 # Extract by location
```

```
 alg_params = {
  'INPUT': outputs['CreateGrid']['OUTPUT'],
  'INTERSECT': outputs['ReprojectLayer']['OUTPUT'],
  'PREDICATE': [6,0,4,1,3],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['ExtractByLocation'] = processing.run('native:extractbylocation', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(7)
 if feedback.isCanceled():
  return {}

 # Intersection
 alg_params = {
  'INPUT': outputs['FixGeometries']['OUTPUT'],
  'INPUT_FIELDS': [''],
  'OVERLAY': outputs['CreateGrid']['OUTPUT'],
  'OVERLAY_FIELDS': [''],
  'OVERLAY_FIELDS_PREFIX': '',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Intersection'] = processing.run('native:intersection', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(8)
 if feedback.isCanceled():
  return {}

 # Join attributes by location
 alg_params = {
  'DISCARD_NONMATCHING': False,
  'INPUT': outputs['ExtractByLocation']['OUTPUT'],
  'JOIN': outputs['ReprojectLayer']['OUTPUT'],
```

```
  'JOIN_FIELDS': QgsExpression('\'name\'').evaluate(),
  'METHOD': 0,
  'PREDICATE': [5,0,1,2,3],
  'PREFIX': '',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['JoinAttributesByLocation'] = processing.run('native:joinattributesbylocation', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

  feedback.setCurrentStep(9)
  if feedback.isCanceled():
  return {}

  # mikos Epifaneias
  alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'SurfaceCost',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 1,
  'FORMULA': 'Case\r\nWhen \"Surface\" = \'asphalt\' then round($length *17 ,2)\r\nWhen \"Surface\" = \'paving_stones\' then round($length* 30,2)\r\nWhen \"Surface\" = \'NULL\' then round($length* 17,2)\r\nWhen \"Surface\" = \'compacted\' then round($length* 17,2)\r\nWhen \"Surface\" = \'concrete:plates\' then round($length* 40,2)\r\nWhen \"Surface\" = \'dirt\' then round($length* 10,2)\r\nWhen \"Surface\" = \'earth\' then round($length* 10,2)\r\nWhen \"Surface\" = \'grass\' then round($length* 20,2)\r\nWhen \"Surface\" = \'gravel\' then round($length* 17,2)\r\nWhen \"Surface\" = \'ground\' then round($length* 17,2)\r\nWhen \"Surface\" = \'metal\' then round($length* 100,2)\r\nWhen \"Surface\" = \'mud\' then round($length* 17,2)\r\nWhen \"Surface\" = \'paved\' then round($length* 30,2)\r\nWhen \"Surface\" = \'cobblestone\' then round($length* 35,2)\r\nWhen \"Surface\" = \'pebblestone\' then round($length* 45,2)\r\nWhen \"Surface\" = \'sett\' then round($length* 17,2)\r\nWhen \"Surface\" = \'steps\' then round($length* 45,2)\r\nWhen \"Surface\" = \'stone\' then round($length* 30,2)\r\nWhen \"Surface\" = \'unpaved\' then round($length* 14,2)\r\nWhen \"Surface\" = \'wood\' then round($length* 17,2)\r\nelse\r\nround($length* 17,2)\r\nend',
  'INPUT': outputs['Intersection']['OUTPUT'],
  'OUTPUT': parameters['']
  }
  outputs['MikosEpifaneias'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
```

```
  results['] = outputs['MikosEpifaneias']['OUTPUT']


  feedback.setCurrentStep(10)
  if feedback.isCanceled():
   return {}


  # Count points in polygon
  alg_params = {
   'CLASSFIELD': '',
   'FIELD': 'Aitisis',
   'POINTS': outputs['ReprojectLayer']['OUTPUT'],
   'POLYGONS': outputs['JoinAttributesByLocation']['OUTPUT'],
   'WEIGHT': '',
   'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['CountPointsInPolygon'] = processing.run('native:countpointsinpolygon', alg_params, context=context, feedback=feedback, is_child_algorithm=True)


  feedback.setCurrentStep(11)
  if feedback.isCanceled():
   return {}


  # pragmatiko mikos
  alg_params = {
   'FIELD_LENGTH': 0,
   'FIELD_NAME': 'pragmatiko mikos',
   'FIELD_PRECISION': 0,
   'FIELD_TYPE': 0,
   'FORMULA': '$length',
   'INPUT': outputs['MikosEpifaneias']['OUTPUT'],
   'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['PragmatikoMikos'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
```

```
 feedback.setCurrentStep(12)
 if feedback.isCanceled():
  return {}

 # Count points in polygon
 alg_params = {
  'CLASSFIELD': '',
  'FIELD': 'ktiria',
  'POINTS': outputs['ReprojectLayer']['OUTPUT'],
  'POLYGONS': outputs['CountPointsInPolygon']['OUTPUT'],
  'WEIGHT': '',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['CountPointsInPolygon'] = processing.run('native:countpointsinpolygon', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(13)
 if feedback.isCanceled():
  return {}

 # Sum line lengths
 alg_params = {
  'COUNT_FIELD': 'COUNT',
  'LEN_FIELD': 'mikos',
  'LINES': outputs['Intersection']['OUTPUT'],
  'POLYGONS': outputs['CountPointsInPolygon']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['SumLineLengths'] = processing.run('native:sumlinelengths', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(14)
 if feedback.isCanceled():
```

```
 return {}

 # Retain fields
 alg_params = {
  'FIELDS': ['surfaceCost','pragmatiko mikos'],
  'INPUT': outputs['PragmatikoMikos']['OUTPUT'],
  'OUTPUT': parameters['']
 }
 outputs['RetainFields'] = processing.run('native:retainfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
 results[''] = outputs['RetainFields']['OUTPUT']

 feedback.setCurrentStep(15)
 if feedback.isCanceled():
  return {}

 # Drop field(s)
 alg_params = {
  'COLUMN': ['left','top','right','bottom','count'],
  'INPUT': outputs['SumLineLengths']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['DropFields'] = processing.run('native:deletecolumn', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(16)
 if feedback.isCanceled():
  return {}

 # Join attributes by location (summary)
 alg_params = {
  'DISCARD_NONMATCHING': False,
  'INPUT': outputs['DropFields']['OUTPUT'],
  'JOIN': outputs['RetainFields']['OUTPUT'],
```

```
 'JOIN_FIELDS': ['SurfaceCost'],
 'PREDICATE': [1],
 'SUMMARIES': [5],
 'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['JoinAttributesByLocationSummary'] = processing.run('qgis:joinbylocationsummary', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(17)
 if feedback.isCanceled():
 return {}

 # Field calculator
 alg_params = {
 'FIELD_LENGTH': 0,
 'FIELD_NAME': 'Adeiodotisi(se mines)',
 'FIELD_PRECISION': 0,
 'FIELD_TYPE': 2,
 'FORMULA': 'Case \r\nwhen \"mikos\" = \'nan\' then 0\r\n\r\nWhen \"name\" = \'Δημος Θεσσαλονικης\' and \"mikos\" < 4000 Then 5 \r\nWhen \"name\" = \'Δημος Θεσσαλονικης\' and \"mikos\" > 4000 Then 6 \r\n\r\nWhen \"name\" = \'Δημος Καλαμαριας\' and \"mikos\" <= 4000 Then 5 \r\nWhen \"name\" = \'Δημος Καλαμαριας\' and \"mikos\" > 4000 Then 7 \r\n\r\nWhen \"name\" = \'Δημος Αμπελοκηπων - Μενεμενης\' and \"mikos\" <= 4000 Then 4\r\nWhen \"name\" = \'Δημος Αμπελοκηπων - Μενεμενης\' and \"mikos\" > 4000 Then 6 \r\n\r\nWhen \"name\" = \'Δημος Δελτα\' and \"mikos\" <= 4000Then 3\r\nWhen \"name\" = \'Δημος Δελτα\' and \"mikos\" > 4000 Then 4 \r\n\r\nWhen \"name\" = \'Δημος Θερμης\' and \"mikos\" <= 4000 Then 5\r\nWhen \"name\" = \'Δημος Θερμης\' and \"mikos\" > 4000 Then 6\r\n\r\nWhen \"name\" = \'Δημος Κορδελιου - Ευοσμου\' and \"mikos\" <= 4000 Then 7\r\nWhen \"name\" = \'Δημος Κορδελιου - Ευοσμου\' and \"mikos\" > 4000 Then 8\r\n\r\nWhen \"name\" = \'Δημος Νεαπολης - Συκεων\' and \"mikos\" <= 4000Then 7\r\nWhen \"name\" = \'Δημος Νεαπολης - Συκεων\' and \"mikos\" > 4000 Then 8\r\n\r\nWhen \"name\" = \'Παυλου Μελα\' and \"mikos\" <= 4000 Then 7\r\nWhen \"name\" = \'Παυλου Μελα\' and \"mikos\" > 4000 Then 8\r\n\r\nWhen \"name\" = \'Δημος Πυλαιας - Χορτιατη\' and \"mikos\" <= 4000 Then 7\r\nWhen \"name\" = \'Δημος Πυλαιας - Χορτιατη\' and \"mikos\" > 4000 Then 8\r\n\r\n\r\nWhen \"name\" = NULL Then 5 \r\n\r\nend\r\n ',
 'INPUT': outputs['JoinAttributesByLocationSummary']['OUTPUT'],
 'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
```

```
 outputs['FieldCalculator'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)


 feedback.setCurrentStep(18)
 if feedback.isCanceled():
  return {}


 # Field calculator
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'Diamerismata',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 1,
  'FORMULA': 'Case \r\nWhen \"name\" = \'Δημος Θεσσαλονικης\' Then\r\nround(ktiria * 12 ,0)\r\n\r\nWhen \"name\" = \'Δημος Καλαμαριας\' Then\r\nround(ktiria * 6 ,0)\r\nWhen \"name\" = \'Δημος Αμπελοκηπων - Μενεμενης\' Then\r\nround(ktiria * 6 ,0)\r\nWhen \"name\" = NULL Then\r\nround(ktiria * 1 ,0)\r\n end\r\n ',
  'INPUT': outputs['FieldCalculator']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['FieldCalculator'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)


 feedback.setCurrentStep(19)
 if feedback.isCanceled():
  return {}


 # Field calculator
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'bank',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'Case \r\nWhen \"mikos\" = \'nan\' Then 0\r\nWhen \"name\" = \'Δημος Θεσσαλονικης\' Then\r\nround(mikos * 2 ,2)\r\nWhen \"name\" = \'Δημος Καλαμαριας\' Then\r\nround(mikos * 3
```

```
,2)\r\nWhen \"name\" = \'Δημος Αμπελοκηπων - Μενεμενης\' Then\r\nround(mikos * 1.5 ,2)\r\nend\r\n
',
  'INPUT': outputs['FieldCalculator']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['FieldCalculator'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

  feedback.setCurrentStep(20)
  if feedback.isCanceled():
   return {}

  # Field calculator
  alg_params = {
   'FIELD_LENGTH': 0,
   'FIELD_NAME': 'kerdos apo aitisis(2 xronia)',
   'FIELD_PRECISION': 0,
   'FIELD_TYPE': 0,
   'FORMULA': 'round(aitisis*26*24 ,0)\r\n',
   'INPUT': outputs['FieldCalculator']['OUTPUT'],
   'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['FieldCalculator'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

  feedback.setCurrentStep(21)
  if feedback.isCanceled():
   return {}

  # Field calculator2
  alg_params = {
   'FIELD_LENGTH': 0,
   'FIELD_NAME': 'dunamiki',
   'FIELD_PRECISION': 0,
```

```
  'FIELD_TYPE': 1,
  'FORMULA': 'Case \r\nwhen \"aitisis\" = \'NULL\' then 0\r\nWhen \"aitisis\" <= 10 then 1 \r\nWhen \"aitisis\" <=20 then 2\r\nWhen \"aitisis\" <=30 then 3\r\nWhen \"aitisis\" <=40 then 4\r\nelse 5\r\nend\r\n ',
  'INPUT': outputs['FieldCalculator']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['FieldCalculator2'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(22)
 if feedback.isCanceled():
  return {}

 # Field calculator3
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'megisto kerdos(2 xronia)',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'round(diamerismata*0.1*dunamiki*30*24)',
  'INPUT': outputs['FieldCalculator2']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['FieldCalculator3'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(23)
 if feedback.isCanceled():
  return {}

 # Field calculator
 alg_params = {
  'FIELD_LENGTH': 0,
```

```
  'FIELD_NAME': 'teli_Xrisis',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'Case \r\nWhen \"mikos\" = \'nan\' Then 0\r\nWhen \"name\" = \'Δημος Θεσσαλονικης\' Then\r\nround(mikos * 0.175 ,2)\r\nWhen \"name\" = \'Δημος Καλαμαριας\' Then\r\nround(mikos * 0.282 ,2)\r\nWhen \"name\" = \'Δημος Αμπελοκηπων - Μενεμενης\' Then\r\nround(mikos * 0.175 ,2)\r\nend\r\n ',
  'INPUT': outputs['FieldCalculator3']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['FieldCalculator'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(24)
 if feedback.isCanceled():
  return {}

 # Field calculator
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'Xomatourgika',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'Case \r\nWhen \"mikos\" = \'nan\' Then 0\r\nWhen \"name\" = \'Δημος Θεσσαλονικης\' Then\r\nround(mikos * 1 + SurfaceCost_sum ,2)\r\nWhen \"name\" = \'Δημος Καλαμαριας\' Then\r\nround(mikos * 1 + SurfaceCost_sum ,2)\r\nWhen \"name\" = \'Δημος Αμπελοκηπων - Μενεμενης\' Then\r\nround(mikos * 1 + SurfaceCost_sum ,2)\r\nend\r\n ',
  'INPUT': outputs['FieldCalculator']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['FieldCalculator'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(25)
 if feedback.isCanceled():
```

```
 return {}

 # Add geometry attributes
 alg_params = {
  'CALC_METHOD': 0,
  'INPUT': outputs['FieldCalculator']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['AddGeometryAttributes'] = processing.run('qgis:exportaddgeometrycolumns', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(26)
 if feedback.isCanceled():
  return {}

 # Field calculator7
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'sunolo ergou',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'round(teli_Xrisis + bank + Xomatourgika ,2)\r\n',
  'INPUT': outputs['AddGeometryAttributes']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['FieldCalculator7'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(27)
 if feedback.isCanceled():
  return {}

 # Field calculator
 alg_params = {
```

```
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'kostos\\kerdos',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'round((diamerismata*0.1*dunamiki*30*24) - (teli_Xrisis + bank + Xomatourgika))',
  'INPUT': outputs['FieldCalculator7']['OUTPUT'],
  'OUTPUT': parameters['Grid']
  }
  outputs['FieldCalculator'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
  results['Grid'] = outputs['FieldCalculator']['OUTPUT']
  return results

 def name(self):
  return 'Step_1_perioxi_endiaferontos'

 def displayName(self):
  return 'Step_1_perioxi_endiaferontos'

 def group(self):
  return 'Gekas_Dimitris'

 def groupId(self):
  return 'Gekas_Dimitris'

 def createInstance(self):
  return Step_1_perioxi_endiaferontos(
```

ΠΑΡΑΡΤΗΜΑ Β : Συλλογή Δεδομένων

"""

Model exported as python.

Name : Step_2 - Collecting Data

Group : Gekas_Dimitris

With QGIS : 31802

"""

```
from qgis.core import QgsProcessing
from qgis.core import QgsProcessingAlgorithm
from qgis.core import QgsProcessingMultiStepFeedback
from qgis.core import QgsProcessingParameterNumber
from qgis.core import QgsProcessingParameterVectorLayer
from qgis.core import QgsProcessingParameterFeatureSink
from qgis.core import QgsProcessingParameterBoolean
from qgis.core import QgsExpression
import processing


class Step_2CollectingData(QgsProcessingAlgorithm):

 def initAlgorithm(self, config=None):
  self.addParameter(QgsProcessingParameterNumber('Apostasifreation', 'Aktina_Sunxoneusis_freation', type=QgsProcessingParameterNumber.Double, minValue=10, maxValue=50, defaultValue=25))
  self.addParameter(QgsProcessingParameterVectorLayer('OriaDimon', 'Oria_Dimon', defaultValue=None))
  self.addParameter(QgsProcessingParameterVectorLayer('area', 'Perioxi Enidaferontos', defaultValue=None))
  self.addParameter(QgsProcessingParameterFeatureSink('Freatia', 'freatia', type=QgsProcessing.TypeVectorAnyGeometry, createByDefault=True, supportsAppend=True, defaultValue=None))
  self.addParameter(QgsProcessingParameterFeatureSink('Boundarys_step_2', 'Boundarys_step_2', type=QgsProcessing.TypeVectorAnyGeometry, createByDefault=True, supportsAppend=True, defaultValue=None))
```

```
 self.addParameter(QgsProcessingParameterFeatureSink('Streets_step_2', 'streets_step_2', type=QgsProcessing.TypeVectorAnyGeometry, createByDefault=True, supportsAppend=True, defaultValue=None))
 self.addParameter(QgsProcessingParameterFeatureSink('Buildings_step_2', 'Buildings_step_2', type=QgsProcessing.TypeVectorAnyGeometry, createByDefault=True, supportsAppend=True, defaultValue=None))
 self.addParameter(QgsProcessingParameterFeatureSink('Demand_step_2', 'Demand_step_2', type=QgsProcessing.TypeVectorAnyGeometry, createByDefault=True, supportsAppend=True, defaultValue=None))
 self.addParameter(QgsProcessingParameterBoolean('VERBOSE_LOG', 'Verbose logging', optional=True, defaultValue=False))

def processAlgorithm(self, parameters, context, model_feedback):
 # Use a multi-step feedback, so that individual child algorithm progress reports are adjusted for the
 # overall progress through the model
 feedback = QgsProcessingMultiStepFeedback(38, model_feedback)
 results = {}
 outputs = {}

 # Build query inside an extent
 alg_params = {
  'EXTENT': parameters['area'],
  'KEY': 'surface',
  'SERVER': 'https://lz4.overpass-api.de/api/interpreter',
  'TIMEOUT': 25,
  'VALUE': ''
 }
 outputs['BuildQueryInsideAnExtent'] = processing.run('quickosm:buildqueryextent', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(1)
 if feedback.isCanceled():
  return {}

 # Build query inside an extent
```

```
 alg_params = {
  'EXTENT': parameters['area'],
  'KEY': 'building',
  'SERVER': 'https://lz4.overpass-api.de/api/interpreter',
  'TIMEOUT': 25,
  'VALUE': ''
 }
 outputs['BuildQueryInsideAnExtent'] = processing.run('quickosm:buildqueryextent', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(2)
 if feedback.isCanceled():
  return {}

 # Download file
 alg_params = {
  'URL': outputs['BuildQueryInsideAnExtent']['OUTPUT_URL'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['DownloadFile'] = processing.run('native:filedownloader', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(3)
 if feedback.isCanceled():
  return {}

 # Download file
 alg_params = {
  'URL': outputs['BuildQueryInsideAnExtent']['OUTPUT_URL'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['DownloadFile'] = processing.run('native:filedownloader', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
```

```
 feedback.setCurrentStep(4)
 if feedback.isCanceled():
  return {}

 # lines
 alg_params = {
  'INPUT': outputs['DownloadFile']['OUTPUT'],
  'OPTIONS': '-sql \"select * from lines\" -t_srs EPSG:3857',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Lines'] = processing.run('gdal:convertformat', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(5)
 if feedback.isCanceled():
  return {}

 # areas
 alg_params = {
  'INPUT': outputs['DownloadFile']['OUTPUT'],
  'OPTIONS': '-sql \"select * from multipolygons\" -t_srs EPSG:3857',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Areas'] = processing.run('gdal:convertformat', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(6)
 if feedback.isCanceled():
  return {}

 # Explode HStore Field
 alg_params = {
```

```
  'EXPECTED_FIELDS':
'surface,sidewalk,addr:street,building:levels,inc,int_name,lanes,maxspeed,name:el,oneway,name:prefi
x',
  'FIELD': 'other_tags',
  'INPUT': outputs['Lines']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['ExplodeHstoreField'] = processing.run('native:explodehstorefield', alg_params,
context=context, feedback=feedback, is_child_algorithm=True)

  feedback.setCurrentStep(7)
  if feedback.isCanceled():
   return {}

  # Explode HStore Field
  alg_params = {
   'EXPECTED_FIELDS': 'addr:housenumber',
   'FIELD': 'other_tags',
   'INPUT': outputs['Areas']['OUTPUT'],
   'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['ExplodeHstoreField'] = processing.run('native:explodehstorefield', alg_params,
context=context, feedback=feedback, is_child_algorithm=True)

  feedback.setCurrentStep(8)
  if feedback.isCanceled():
   return {}

  # Join attributes by location
  alg_params = {
   'DISCARD_NONMATCHING': False,
   'INPUT': outputs['ExplodeHstoreField']['OUTPUT'],
   'JOIN': parameters['OriaDimon'],
   'JOIN_FIELDS': [''],
```

```
  'METHOD': 0,
  'PREDICATE': [0,1,2,3,5],
  'PREFIX': '',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['JoinAttributesByLocation'] = processing.run('native:joinattributesbylocation', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

  feedback.setCurrentStep(9)
  if feedback.isCanceled():
   return {}

  # Boundary
  alg_params = {
   'INPUT': outputs['ExplodeHstoreField']['OUTPUT'],
   'OUTPUT': parameters['Boundarys_step_2']
  }
  outputs['Boundary'] = processing.run('native:boundary', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
  results['Boundarys_step_2'] = outputs['Boundary']['OUTPUT']

  feedback.setCurrentStep(10)
  if feedback.isCanceled():
   return {}

  # Fix geometries
  alg_params = {
   'INPUT': outputs['ExplodeHstoreField']['OUTPUT'],
   'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['FixGeometries'] = processing.run('native:fixgeometries', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

  feedback.setCurrentStep(11)
```

```
  if feedback.isCanceled():
   return {}

  # Create spatial index
  alg_params = {
   'INPUT': outputs['Boundary']['OUTPUT']
  }
  outputs['CreateSpatialIndex'] = processing.run('native:createspatialindex', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

  feedback.setCurrentStep(12)
  if feedback.isCanceled():
   return {}

  # Create spatial index
  alg_params = {
   'INPUT': outputs['FixGeometries']['OUTPUT']
  }
  outputs['CreateSpatialIndex'] = processing.run('native:createspatialindex', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

  feedback.setCurrentStep(13)
  if feedback.isCanceled():
   return {}

  # Line intersections
  alg_params = {
   'INPUT': outputs['FixGeometries']['OUTPUT'],
   'INPUT_FIELDS': [''],
   'INTERSECT': outputs['FixGeometries']['OUTPUT'],
   'INTERSECT_FIELDS': [''],
   'INTERSECT_FIELDS_PREFIX': '',
   'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
```

```
 outputs['LineIntersections'] = processing.run('native:lineintersections', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(14)
 if feedback.isCanceled():
  return {}

 # Centroids
 alg_params = {
  'ALL_PARTS': True,
  'INPUT': outputs['Boundary']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Centroids'] = processing.run('native:centroids', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(15)
 if feedback.isCanceled():
  return {}

 # Create spatial index
 alg_params = {
  'INPUT': outputs['Centroids']['OUTPUT']
 }
 outputs['CreateSpatialIndex'] = processing.run('native:createspatialindex', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(16)
 if feedback.isCanceled():
  return {}

 # Create spatial index
 alg_params = {
  'INPUT': outputs['ExplodeHstoreField']['OUTPUT']
```

```
 }
 outputs['CreateSpatialIndex'] = processing.run('native:createspatialindex', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(17)
 if feedback.isCanceled():
  return {}

 # Rename field
 alg_params = {
  'FIELD': 'name_2',
  'INPUT': outputs['JoinAttributesByLocation']['OUTPUT'],
  'NEW_NAME': 'Dimos',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['RenameField'] = processing.run('native:renametablefield', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(18)
 if feedback.isCanceled():
  return {}

 # Join attributes by location
 alg_params = {
  'DISCARD_NONMATCHING': False,
  'INPUT': outputs['Centroids']['OUTPUT'],
  'JOIN': parameters['OriaDimon'],
  'JOIN_FIELDS': [''],
  'METHOD': 0,
  'PREDICATE': [0,1,2,3,5],
  'PREFIX': '',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
```

```
 outputs['JoinAttributesByLocation'] = processing.run('native:joinattributesbylocation', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(19)
 if feedback.isCanceled():
  return {}

 # Join attributes by location
 alg_params = {
  'DISCARD_NONMATCHING': False,
  'INPUT': outputs['FixGeometries']['OUTPUT'],
  'JOIN': parameters['OriaDimon'],
  'JOIN_FIELDS': [''],
  'METHOD': 0,
  'PREDICATE': [0,1,2,3,5],
  'PREFIX': '',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['JoinAttributesByLocation'] = processing.run('native:joinattributesbylocation', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(20)
 if feedback.isCanceled():
  return {}

 # Extract by location
 alg_params = {
  'INPUT': outputs['LineIntersections']['OUTPUT'],
  'INTERSECT': parameters['area'],
  'PREDICATE': [0],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['ExtractByLocation'] = processing.run('native:extractbylocation', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
```

```
 feedback.setCurrentStep(21)
 if feedback.isCanceled():
  return {}

 # Rename field
 alg_params = {
  'FIELD': 'name_2',
  'INPUT': outputs['JoinAttributesByLocation']['OUTPUT'],
  'NEW_NAME': 'Dimos',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['RenameField'] = processing.run('native:renametablefield', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(22)
 if feedback.isCanceled():
  return {}

 # Rename field
 alg_params = {
  'FIELD': 'name_2',
  'INPUT': outputs['JoinAttributesByLocation']['OUTPUT'],
  'NEW_NAME': 'Dimos',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['RenameField'] = processing.run('native:renametablefield', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(23)
 if feedback.isCanceled():
  return {}

 # Drop field(s)
```

```
 alg_params = {
  'COLUMN': ['folders','descriptio','altitude','alt_mode','time_begin','time_end','time_when'],
  'INPUT': outputs['RenameField']['OUTPUT'],
  'OUTPUT': parameters['Buildings_step_2']
 }
 outputs['DropFields'] = processing.run('native:deletecolumn', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
 results['Buildings_step_2'] = outputs['DropFields']['OUTPUT']

 feedback.setCurrentStep(24)
 if feedback.isCanceled():
  return {}

 # Join attributes by location
 alg_params = {
  'DISCARD_NONMATCHING': False,
  'INPUT': outputs['ExtractByLocation']['OUTPUT'],
  'JOIN': parameters['OriaDimon'],
  'JOIN_FIELDS': [''],
  'METHOD': 0,
  'PREDICATE': [0,1,2,3,5],
  'PREFIX': '',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['JoinAttributesByLocation'] = processing.run('native:joinattributesbylocation', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(25)
 if feedback.isCanceled():
  return {}

 # Drop field(s)
 alg_params = {
  'COLUMN': ['folders','descriptio','altitude','alt_mode','time_begin','time_end','time_when'],
```

```
 'INPUT': outputs['RenameField']['OUTPUT'],
 'OUTPUT': parameters['Streets_step_2']
 }
 outputs['DropFields'] = processing.run('native:deletecolumn', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
 results['Streets_step_2'] = outputs['DropFields']['OUTPUT']

 feedback.setCurrentStep(26)
 if feedback.isCanceled():
  return {}

 # Rename field
 alg_params = {
 'FIELD': 'name_3',
 'INPUT': outputs['JoinAttributesByLocation']['OUTPUT'],
 'NEW_NAME': 'Dimos',
 'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['RenameField'] = processing.run('native:renametablefield', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(27)
 if feedback.isCanceled():
  return {}

 # Drop field(s)
 alg_params = {
 'COLUMN': ['folders','descriptio','altitude','alt_mode','time_begin','time_end','time_when'],
 'INPUT': outputs['RenameField']['OUTPUT'],
 'OUTPUT': parameters['Demand_step_2']
 }
 outputs['DropFields'] = processing.run('native:deletecolumn', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
 results['Demand_step_2'] = outputs['DropFields']['OUTPUT']
```

```
 feedback.setCurrentStep(28)
 if feedback.isCanceled():
  return {}

 # Drop field(s)
 alg_params = {
  'COLUMN': ['folders','descriptio','altitude','alt_mode','time_begin','time_end','time_when'],
  'INPUT': outputs['RenameField']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['DropFields'] = processing.run('native:deletecolumn', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(29)
 if feedback.isCanceled():
  return {}

 # Delete duplicate geometries
 alg_params = {
  'INPUT': outputs['DropFields']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['DeleteDuplicateGeometries'] = processing.run('native:deleteduplicategeometries', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(30)
 if feedback.isCanceled():
  return {}

 # Buffer
 alg_params = {
  'DISSOLVE': False,
  'DISTANCE': QgsExpression(' @Apostasifreation ').evaluate(),
```

```
  'END_CAP_STYLE': 0,
  'INPUT': outputs['DeleteDuplicateGeometries']['OUTPUT'],
  'JOIN_STYLE': 0,
  'MITER_LIMIT': 2,
  'SEGMENTS': 5,
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['Buffer'] = processing.run('native:buffer', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

  feedback.setCurrentStep(31)
  if feedback.isCanceled():
   return {}

  # Dissolve
  alg_params = {
   'FIELD': [''],
   'INPUT': outputs['Buffer']['OUTPUT'],
   'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['Dissolve'] = processing.run('native:dissolve', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

  feedback.setCurrentStep(32)
  if feedback.isCanceled():
   return {}

  # Centroids
  alg_params = {
   'ALL_PARTS': True,
   'INPUT': outputs['Dissolve']['OUTPUT'],
   'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
```

```
 outputs['Centroids'] = processing.run('native:centroids', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(33)
 if feedback.isCanceled():
  return {}

 # Snap geometries to layer
 alg_params = {
  'BEHAVIOR': 0,
  'INPUT': outputs['Centroids']['OUTPUT'],
  'REFERENCE_LAYER': outputs['DropFields']['OUTPUT'],
  'TOLERANCE': 500,
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['SnapGeometriesToLayer'] = processing.run('native:snapgeometries', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(34)
 if feedback.isCanceled():
  return {}

 # Retain fields
 alg_params = {
  'FIELDS': ['Dimos','name'],
  'INPUT': outputs['SnapGeometriesToLayer']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['RetainFields'] = processing.run('native:retainfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(35)
 if feedback.isCanceled():
  return {}
```

```
  # Add autoincremental field
  alg_params = {
   'FIELD_NAME': 'number',
   'GROUP_FIELDS': [''],
   'INPUT': outputs['RetainFields']['OUTPUT'],
   'SORT_ASCENDING': True,
   'SORT_EXPRESSION': '',
   'SORT_NULLS_FIRST': False,
   'START': 1,
   'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['AddAutoincrementalField'] = processing.run('native:addautoincrementalfield', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

  feedback.setCurrentStep(36)
  if feedback.isCanceled():
   return {}

  # Add field to attributes table
  alg_params = {
   'FIELD_LENGTH': 10,
   'FIELD_NAME': 'Diastasis',
   'FIELD_PRECISION': 0,
   'FIELD_TYPE': 2,
   'INPUT': outputs['AddAutoincrementalField']['OUTPUT'],
   'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['AddFieldToAttributesTable'] = processing.run('native:addfieldtoattributestable', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

  feedback.setCurrentStep(37)
  if feedback.isCanceled():
   return {}
```

```
 # Add X/Y fields to layer
 alg_params = {
  'CRS': 'ProjectCrs',
  'INPUT': outputs['AddFieldToAttributesTable']['OUTPUT'],
  'PREFIX': '',
  'OUTPUT': parameters['Freatia']
 }
 outputs['AddXyFieldsToLayer'] = processing.run('native:addxyfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
 results['Freatia'] = outputs['AddXyFieldsToLayer']['OUTPUT']
 return results

 def name(self):
  return 'Step_2 - Collecting Data'

 def displayName(self):
  return 'Step_2 - Collecting Data'

 def group(self):
  return 'Gekas_Dimitris'

 def groupId(self):
  return 'Gekas_Dimitris'

 def createInstance(self):
  return Step_2CollectingData()
```

ΠΑΡΑΡΤΗΜΑ Γ : Σύνδεση κτιρίων με το δίκτυο

```
"""
```

```
Model exported as python.
Name : Step_3 - Creating Connections
Group : Gekas_Dimitris
With QGIS : 31802
"""

from qgis.core import QgsProcessing
from qgis.core import QgsProcessingAlgorithm
from qgis.core import QgsProcessingMultiStepFeedback
from qgis.core import QgsProcessingParameterVectorLayer
from qgis.core import QgsProcessingParameterFeatureSource
from qgis.core import QgsProcessingParameterFeatureSink
from qgis.core import QgsProcessingParameterBoolean
import processing


class Step_3CreatingConnections(QgsProcessingAlgorithm):

 def initAlgorithm(self, config=None):
  self.addParameter(QgsProcessingParameterVectorLayer('Demands', 'Demand', types=[QgsProcessing.TypeVectorPoint], defaultValue=None))
  self.addParameter(QgsProcessingParameterFeatureSource('ExistingNetwork', 'Existing Network', types=[QgsProcessing.TypeVectorLine], defaultValue=None))
  self.addParameter(QgsProcessingParameterVectorLayer('Headend (2)', 'Boundry', types=[QgsProcessing.TypeVectorLine], defaultValue=None))
  self.addParameter(QgsProcessingParameterVectorLayer('Headend (2) (2) (2)', 'Buldings', types=[QgsProcessing.TypeVectorPolygon], defaultValue=None))
  self.addParameter(QgsProcessingParameterVectorLayer('Network', 'Streets', types=[QgsProcessing.TypeVectorLine], defaultValue=None))
  self.addParameter(QgsProcessingParameterFeatureSource('area', 'Area', types=[QgsProcessing.TypeVectorPolygon,QgsProcessing.TypeVectorAnyGeometry], defaultValue=None))
  self.addParameter(QgsProcessingParameterFeatureSink('Streets_step_3', 'Streets_step_3', type=QgsProcessing.TypeVectorAnyGeometry, createByDefault=True, supportsAppend=True, defaultValue=None))
```

```
 self.addParameter(QgsProcessingParameterFeatureSink('Move_centroid_buildings_step_3', 'Move_Centroid_Buildings_Step_3', type=QgsProcessing.TypeVectorAnyGeometry, createByDefault=True, supportsAppend=True, defaultValue=None))
 self.addParameter(QgsProcessingParameterBoolean('VERBOSE_LOG', 'Verbose logging', optional=True, defaultValue=False))

 def processAlgorithm(self, parameters, context, model_feedback):
 # Use a multi-step feedback, so that individual child algorithm progress reports are adjusted for the
 # overall progress through the model
 feedback = QgsProcessingMultiStepFeedback(11, model_feedback)
 results = {}
 outputs = {}

 # Selected Clients
 alg_params = {
  'INPUT': parameters['Demands'],
  'INTERSECT': parameters['area'],
  'PREDICATE': [0],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['SelectedClients'] = processing.run('native:extractbylocation', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(1)
 if feedback.isCanceled():
  return {}

 # Extract by location
 alg_params = {
  'INPUT': parameters['ExistingNetwork'],
  'INTERSECT': parameters['area'],
  'PREDICATE': [0],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
```

```
 outputs['ExtractByLocation'] = processing.run('native:extractbylocation', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(2)
 if feedback.isCanceled():
  return {}

 # Selected Network
 alg_params = {
  'INPUT': parameters['Network'],
  'INTERSECT': parameters['area'],
  'PREDICATE': [0],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['SelectedNetwork'] = processing.run('native:extractbylocation', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(3)
 if feedback.isCanceled():
  return {}

 # Connect nodes to lines
 alg_params = {
  'LINES': outputs['ExtractByLocation']['OUTPUT'],
  'NODES': outputs['SelectedClients']['OUTPUT'],
  'RAYON': 1000
 }
 outputs['ConnectNodesToLines'] = processing.run('Networks:connect_nodes2lines', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(4)
 if feedback.isCanceled():
  return {}
```

```
 # Connect nodes to lines
 alg_params = {
  'LINES': outputs['SelectedNetwork']['OUTPUT'],
  'NODES': outputs['SelectedClients']['OUTPUT'],
  'RAYON': 1000
 }
 outputs['ConnectNodesToLines'] = processing.run('Networks:connect_nodes2lines', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(5)
 if feedback.isCanceled():
  return {}

 # Fix geometries
 alg_params = {
  'INPUT': outputs['SelectedNetwork']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['FixGeometries'] = processing.run('native:fixgeometries', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(6)
 if feedback.isCanceled():
  return {}

 # Line intersections
 alg_params = {
  'INPUT': parameters['Headend (2)'],
  'INPUT_FIELDS': [''],
  'INTERSECT': outputs['FixGeometries']['OUTPUT'],
  'INTERSECT_FIELDS': [''],
  'INTERSECT_FIELDS_PREFIX': '',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
```

```
 outputs['LineIntersections'] = processing.run('native:lineintersections', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(7)
 if feedback.isCanceled():
  return {}

 # Difference
 alg_params = {
  'INPUT': outputs['FixGeometries']['OUTPUT'],
  'OVERLAY': parameters['Headend (2) (2) (2)'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Difference'] = processing.run('native:difference', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(8)
 if feedback.isCanceled():
  return {}

 # Retain fields
 alg_params = {
  'FIELDS': ['fid','highway','name','surface','sidewalk','name:el','int_name','lanes','max_speed','oneway','Dimos','addr:housenumber'],
  'INPUT': outputs['LineIntersections']['OUTPUT'],
  'OUTPUT': parameters['Move_centroid_buildings_step_3']
 }
 outputs['RetainFields'] = processing.run('native:retainfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
 results['Move_centroid_buildings_step_3'] = outputs['RetainFields']['OUTPUT']

 feedback.setCurrentStep(9)
 if feedback.isCanceled():
```

```
  return {}

  # Join attributes by location
  alg_params = {
   'DISCARD_NONMATCHING': False,
   'INPUT': outputs['Difference']['OUTPUT'],
   'JOIN': outputs['LineIntersections']['OUTPUT'],
   'JOIN_FIELDS': [''],
   'METHOD': 0,
   'PREDICATE': [0],
   'PREFIX': '',
   'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['JoinAttributesByLocation'] = processing.run('native:joinattributesbylocation', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

  feedback.setCurrentStep(10)
  if feedback.isCanceled():
   return {}

  # Retain fields
  alg_params = {
   'FIELDS': ['fid','highway','name','surface','sidewalk','name:el','int_name','lanes','max_speed','oneway','Dimos','addr:housenumber'],
   'INPUT': outputs['JoinAttributesByLocation']['OUTPUT'],
   'OUTPUT': parameters['Streets_step_3']
  }
  outputs['RetainFields'] = processing.run('native:retainfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
  results['Streets_step_3'] = outputs['RetainFields']['OUTPUT']
  return results

 def name(self):
```

```
 return 'Step_3 - Creating Connections'

def displayName(self):
 return 'Step_3 - Creating Connections'

def group(self):
 return 'Gekas_Dimitris'

def groupId(self):
 return 'Gekas_Dimitris'

def createInstance(self):
 return Step_3CreatingConnections()
```

ΠΑΡΑΡΤΗΜΑ Δ : Εύρεση βέλτιστων διαδρομών

"""

Model exported as python.

Name : Step_4 - Shortest Paths

Group : Gekas_Dimitris

With QGIS : 31802

"""

```
from qgis.core import QgsProcessing
from qgis.core import QgsProcessingAlgorithm
from qgis.core import QgsProcessingMultiStepFeedback
from qgis.core import QgsProcessingParameterFeatureSource
from qgis.core import QgsProcessingParameterPoint
from qgis.core import QgsProcessingParameterEnum
from qgis.core import QgsProcessingParameterBoolean
from qgis.core import QgsProcessingParameterVectorDestination
from qgis.core import QgsProcessingParameterFeatureSink
from qgis.core import QgsProcessingParameterDefinition
from qgis.core import QgsExpression
import processing


class Step_4ShortestPaths(QgsProcessingAlgorithm):

 def initAlgorithm(self, config=None):
  self.addParameter(QgsProcessingParameterFeatureSource('Clietns', 'Clietns', types=[QgsProcessing.TypeVectorPoint], defaultValue=None))
  self.addParameter(QgsProcessingParameterFeatureSource('EndManholes', 'End Manholes', types=[QgsProcessing.TypeVectorPoint], defaultValue=None))
  self.addParameter(QgsProcessingParameterPoint('EntryPoint', 'Entry Point', defaultValue=''))
  self.addParameter(QgsProcessingParameterEnum('Headends', 'Headends', options=['Kalamaria','Kentro','Charilaou','Kordelio','Foinikas'], allowMultiple=False, usesStaticStrings=False, defaultValue=[4]))
```

```
 self.addParameter(QgsProcessingParameterFeatureSource('LocalNetwork', 'Local Network', types=[QgsProcessing.TypeVectorLine], defaultValue=None))
 self.addParameter(QgsProcessingParameterFeatureSource('Network', 'Main_Network', types=[QgsProcessing.TypeVectorLine], defaultValue=None))
 param = QgsProcessingParameterBoolean('asphalt', 'asphalt', optional=True, defaultValue=False)
 param.setFlags(param.flags() | QgsProcessingParameterDefinition.FlagAdvanced)
 self.addParameter(param)
 self.addParameter(QgsProcessingParameterVectorDestination('Clients_cable_network', 'Clients_Cable_Network', type=QgsProcessing.TypeVectorAnyGeometry, createByDefault=True, defaultValue=None))
 self.addParameter(QgsProcessingParameterFeatureSink('Main_network_final', 'Main_Network_final', type=QgsProcessing.TypeVectorAnyGeometry, createByDefault=True, supportsAppend=True, defaultValue=None))
 self.addParameter(QgsProcessingParameterFeatureSink('Trench', 'trench', type=QgsProcessing.TypeVectorLine, createByDefault=True, defaultValue=None))
 self.addParameter(QgsProcessingParameterBoolean('VERBOSE_LOG', 'Verbose logging', optional=True, defaultValue=False))

def processAlgorithm(self, parameters, context, model_feedback):
 # Use a multi-step feedback, so that individual child algorithm progress reports are adjusted for the
 # overall progress through the model
 feedback = QgsProcessingMultiStepFeedback(5, model_feedback)
 results = {}
 outputs = {}

 # v.net.distance
 alg_params = {
  '-g': False,
  '-l': True,
  'GRASS_MIN_AREA_PARAMETER': 0.0001,
  'GRASS_OUTPUT_TYPE_PARAMETER': 0,
  'GRASS_REGION_PARAMETER': None,
  'GRASS_SNAP_TOLERANCE_PARAMETER': -1,
  'GRASS_VECTOR_DSCO': '',
  'GRASS_VECTOR_EXPORT_NOCAT': False,
```

```
 'GRASS_VECTOR_LCO': '',
 'arc_backward_column': '',
 'arc_column': '',
 'arc_type': [0,1],
 'flayer': parameters['Clietns'],
 'from_cats': '',
 'from_where': '',
 'input': parameters['LocalNetwork'],
 'node_column': '',
 'threshold': 300,
 'tlayer': parameters['EndManholes'],
 'to_cats': '',
 'to_type': [0,1,2],
 'to_where': '',
 'output': parameters['Clients_cable_network']
 }
 outputs['Vnetdistance'] = processing.run('grass7:v.net.distance', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
 results['Clients_cable_network'] = outputs['Vnetdistance']['output']

 feedback.setCurrentStep(1)
 if feedback.isCanceled():
  return {}

 # Shortest path (point to layer)
 alg_params = {
  'DEFAULT_DIRECTION': 2,
  'DEFAULT_SPEED': 50,
  'DIRECTION_FIELD': '',
  'END_POINTS': parameters['EndManholes'],
  'INPUT': parameters['LocalNetwork'],
  'SPEED_FIELD': '',
  'START_POINT': parameters['EntryPoint'],
```

```
  'STRATEGY': 0,
  'TOLERANCE': 0,
  'VALUE_BACKWARD': '',
  'VALUE_BOTH': '',
  'VALUE_FORWARD': '',
  'OUTPUT': parameters['Trench']
  }
  outputs['ShortestPathPointToLayer'] = processing.run('native:shortestpathpointtolayer', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
  results['Trench'] = outputs['ShortestPathPointToLayer']['OUTPUT']

  feedback.setCurrentStep(2)
  if feedback.isCanceled():
   return {}

  # Shortest path (point to point)
  alg_params = {
   'DEFAULT_DIRECTION': 2,
   'DEFAULT_SPEED': 50,
   'DIRECTION_FIELD': '',
   'END_POINT': QgsExpression('CASE\r\nWHEN @Headends = 0 THEN \r\n\'2554173.7037999998,4951179.4157999996\'\r\nWHEN @Headends = 1 THEN \r\n\'2554885.9323999998,4958916.4467000002\'\r\nWHEN @Headends = 2 THEN\r\n\'2556230.6915000002,4953215.0158000002\'\r\nWHEN @Headends = 3 THEN\r\n\'2549044.0490000001 ,4964605.0910000000\'\r\nWHEN @Headends = 4 THEN \'2556987.0098999999,4949094.2231999999\'\r\nEND').evaluate(),
   'INPUT': parameters['Network'],
   'SPEED_FIELD': '',
   'START_POINT': parameters['EntryPoint'],
   'STRATEGY': 0,
   'TOLERANCE': 0,
   'VALUE_BACKWARD': '',
   'VALUE_BOTH': '',
   'VALUE_FORWARD': '',
   'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
```

```
 }
 outputs['ShortestPathPointToPoint'] = processing.run('native:shortestpathpointtopoint', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(3)
 if feedback.isCanceled():
  return {}

 # Join attributes by location
 alg_params = {
  'DISCARD_NONMATCHING': False,
  'INPUT': outputs['ShortestPathPointToPoint']['OUTPUT'],
  'JOIN': parameters['Network'],
  'JOIN_FIELDS': [''],
  'METHOD': 0,
  'PREDICATE': [0,1,2,3,5],
  'PREFIX': '',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['JoinAttributesByLocation'] = processing.run('native:joinattributesbylocation', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(4)
 if feedback.isCanceled():
  return {}

 # Retain fields
 alg_params = {
  'FIELDS': ['start','end','cost','dimos'],
  'INPUT': outputs['JoinAttributesByLocation']['OUTPUT'],
  'OUTPUT': parameters['Main_network_final']
 }
 outputs['RetainFields'] = processing.run('native:retainfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
```

```
 results['Main_network_final'] = outputs['RetainFields']['OUTPUT']
 return results

def name(self):
 return 'Step_4 - Shortest Paths'

def displayName(self):
 return 'Step_4 - Shortest Paths'

def group(self):
 return 'Gekas_Dimitris'

def groupId(self):
 return 'Gekas_Dimitris'

def createInstance(self):
 return Step_4ShortestPaths()
```

ΠΑΡΑΡΤΗΜΑ Ε : Ομαδοποίηση

"""

"""

Model exported as python.

Name : Step_5_K-means Clustering

Group : Gekas_Dimitris

With QGIS : 31802

"""

```
from qgis.core import QgsProcessing
from qgis.core import QgsProcessingAlgorithm
from qgis.core import QgsProcessingMultiStepFeedback
from qgis.core import QgsProcessingParameterNumber
from qgis.core import QgsProcessingParameterVectorLayer
from qgis.core import QgsProcessingParameterFeatureSink
from qgis.core import QgsProcessingParameterBoolean
from qgis.core import QgsExpression
import processing


class Step_5_kmeansClustering(QgsProcessingAlgorithm):

  def initAlgorithm(self, config=None):
    self.addParameter(QgsProcessingParameterNumber('Fibercable', 'Fiber_cable', type=QgsProcessingParameterNumber.Integer, minValue=1, maxValue=144, defaultValue=48))
    self.addParameter(QgsProcessingParameterVectorLayer('Headend (2) (2)', 'Clients_Cable_Network', types=[QgsProcessing.TypeVectorLine], defaultValue=None))
    self.addParameter(QgsProcessingParameterVectorLayer('Headend (2) (2) (2)', 'Move_Centroid_Buildings_Step_3', types=[QgsProcessing.TypeVectorPoint], defaultValue=None))
    self.addParameter(QgsProcessingParameterNumber('splitter', 'splitter', type=QgsProcessingParameterNumber.Integer, minValue=1, maxValue=64, defaultValue=4))
    self.addParameter(QgsProcessingParameterFeatureSink('Clients', 'clients', type=QgsProcessing.TypeVectorAnyGeometry, createByDefault=True, supportsAppend=True, defaultValue=None))
```

```
  self.addParameter(QgsProcessingParameterFeatureSink('Local_network', 'local_network', type=QgsProcessing.TypeVectorAnyGeometry, createByDefault=True, supportsAppend=True, defaultValue=None))
  self.addParameter(QgsProcessingParameterBoolean('VERBOSE_LOG', 'Verbose logging', optional=True, defaultValue=False))

 def processAlgorithm(self, parameters, context, model_feedback):
  # Use a multi-step feedback, so that individual child algorithm progress reports are adjusted for the
  # overall progress through the model
  feedback = QgsProcessingMultiStepFeedback(6, model_feedback)
  results = {}
  outputs = {}

  # Basic statistics for fields
  alg_params = {
   'FIELD_NAME': 'fid',
   'INPUT_LAYER': parameters['Headend (2) (2) (2)']
  }
  outputs['BasicStatisticsForFields'] = processing.run('qgis:basicstatisticsforfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

  feedback.setCurrentStep(1)
  if feedback.isCanceled():
   return {}

  # Fix geometries
  alg_params = {
   'INPUT': parameters['Headend (2) (2)'],
   'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['FixGeometries'] = processing.run('native:fixgeometries', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

  feedback.setCurrentStep(2)
```

```
  if feedback.isCanceled():
    return {}

  # K-means clustering
  alg_params = {
    'CLUSTERS': QgsExpression('Case\r\nWhen\r\n@Basic_statistics_for_fields_COUNT/@splitter <= @Fibercable Then @Basic_statistics_for_fields_COUNT/@splitter\r\nelse\r\n @Fibercable \r\nEND\r\n').evaluate(),
    'FIELD_NAME': 'CLUSTER_ID',
    'INPUT': parameters['Headend (2) (2) (2)'],
    'SIZE_FIELD_NAME': 'CLUSTER_SIZE',
    'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['KmeansClustering'] = processing.run('native:kmeansclustering', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

  feedback.setCurrentStep(3)
  if feedback.isCanceled():
    return {}

  # Retain fields
  alg_params = {
    'FIELDS': ['fid','highway','name','surface','sidewalk','name:el','int_name','lanes','max_speed','oneway','CLUSTER_ID','CLUSTER_SIZE','Dimos'],
    'INPUT': outputs['KmeansClustering']['OUTPUT'],
    'OUTPUT': parameters['Clients']
  }
  outputs['RetainFields'] = processing.run('native:retainfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
  results['Clients'] = outputs['RetainFields']['OUTPUT']

  feedback.setCurrentStep(4)
  if feedback.isCanceled():
    return {}
```

```
  # Join attributes by location
  alg_params = {
    'DISCARD_NONMATCHING': False,
    'INPUT': outputs['FixGeometries']['OUTPUT'],
    'JOIN': outputs['KmeansClustering']['OUTPUT'],
    'JOIN_FIELDS': [''],
    'METHOD': 0,
    'PREDICATE': [0],
    'PREFIX': '',
    'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['JoinAttributesByLocation'] = processing.run('native:joinattributesbylocation', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

  feedback.setCurrentStep(5)
  if feedback.isCanceled():
    return {}

  # Retain fields
  alg_params = {
    'FIELDS': ['fid','highway','name','surface','sidewalk','name:el','int_name','lanes','max_speed','oneway','CLUSTER_ID','CLUSTER_SIZE','Dimos'],
    'INPUT': outputs['JoinAttributesByLocation']['OUTPUT'],
    'OUTPUT': parameters['Local_network']
  }
  outputs['RetainFields'] = processing.run('native:retainfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
  results['Local_network'] = outputs['RetainFields']['OUTPUT']
  return results

 def name(self):
  return 'Step_5_K-means Clustering'
```

```
def displayName(self):
 return 'Step_5_K-means Clustering'

def group(self):
 return 'Gekas_Dimitris'

def groupId(self):
 return 'Gekas_Dimitris'

def createInstance(self):
 return Step_5_kmeansClustering()
```

ΠΑΡΑΡΤΗΜΑ ΣΤ : Εξαγωγή δεδομένων σε Excel

"""

Model exported as python.

Name : Step_6_excel

Group : Gekas_Dimitris

With QGIS : 31802

"""

```
from qgis.core import QgsProcessing
from qgis.core import QgsProcessingAlgorithm
from qgis.core import QgsProcessingMultiStepFeedback
from qgis.core import QgsProcessingParameterVectorLayer
from qgis.core import QgsProcessingParameterBoolean
from qgis.core import QgsExpression
import processing


class Step_6_excel(QgsProcessingAlgorithm):

 def initAlgorithm(self, config=None):
  self.addParameter(QgsProcessingParameterVectorLayer('Clientsnetwork', 'Clients_network', defaultValue=None))
  self.addParameter(QgsProcessingParameterVectorLayer('Local', 'trench_network', defaultValue=None))
  self.addParameter(QgsProcessingParameterVectorLayer('clients', 'clients', defaultValue=None))
  self.addParameter(QgsProcessingParameterVectorLayer('freatia', 'freatia', defaultValue=None))
  self.addParameter(QgsProcessingParameterVectorLayer('main', 'Main_Network', defaultValue=None))
  self.addParameter(QgsProcessingParameterVectorLayer('oriadimon', 'oria dimon', types=[QgsProcessing.TypeVectorPolygon], defaultValue=None))
  self.addParameter(QgsProcessingParameterBoolean('VERBOSE_LOG', 'Verbose logging', optional=True, defaultValue=False))

 def processAlgorithm(self, parameters, context, model_feedback):
```

```
 # Use a multi-step feedback, so that individual child algorithm progress reports are adjusted for the
 # overall progress through the model
 feedback = QgsProcessingMultiStepFeedback(58, model_feedback)
 results = {}
 outputs = {}

 # Polygons to lines
 alg_params = {
  'INPUT': parameters['oriadimon'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['PolygonsToLines'] = processing.run('native:polygonstolines', alg_params, context=context,
feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(1)
 if feedback.isCanceled():
  return {}

 # splitters/sunthetires
 alg_params = {
  'CATEGORIES_FIELD_NAME': ['cluster_id','splitter'],
  'INPUT': parameters['clients'],
  'VALUES_FIELD_NAME': QgsExpression('@splitter').evaluate(),
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Splitterssunthetires'] = processing.run('qgis:statisticsbycategories', alg_params,
context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(2)
 if feedback.isCanceled():
  return {}

 # Split with lines
 alg_params = {
```

```
 'INPUT': parameters['main'],
 'LINES': outputs['PolygonsToLines']['OUTPUT'],
 'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['SplitWithLines'] = processing.run('native:splitwithlines', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(3)
 if feedback.isCanceled():
 return {}

 # Diamerismata
 alg_params = {
 'FIELD_LENGTH': 0,
 'FIELD_NAME': 'Diamerismata',
 'FIELD_PRECISION': 0,
 'FIELD_TYPE': 1,
 'FORMULA': 'Case \r\nWhen \"Dimos\" = \'Δημος Θεσσαλονικης\' Then\r\n12\r\nWhen \"Dimos\" = \'Δημος Καλαμαριας\' Then\r\n5\r\nWhen \"Dimos\" = \'Δημος Αμπελοκηπων - Μενεμενης\' Then\r\n6\r\nWhen \"Dimos\" = NULL Then\r\n1\r\n end\r\n ',
 'INPUT': parameters['clients'],
 'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Diamerismata'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(4)
 if feedback.isCanceled():
 return {}

 # Retain fields
 alg_params = {
 'FIELDS': ['fid','dimos','Diamerismata'],
 'INPUT': outputs['Diamerismata']['OUTPUT'],
```

```
 'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['RetainFields'] = processing.run('native:retainfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(5)
 if feedback.isCanceled():
  return {}

 # kasetines
 alg_params = {
  'CATEGORIES_FIELD_NAME': [''],
  'INPUT': parameters['clients'],
  'VALUES_FIELD_NAME': '',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Kasetines'] = processing.run('qgis:statisticsbycategories', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(6)
 if feedback.isCanceled():
  return {}

 # Delete duplicate geometries
 alg_params = {
  'INPUT': outputs['SplitWithLines']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['DeleteDuplicateGeometries'] = processing.run('native:deleteduplicategeometries', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(7)
 if feedback.isCanceled():
  return {}
```

```
 # Retain fields
 alg_params = {
  'FIELDS': ['name'],
  'INPUT': parameters['Clientsnetwork'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['RetainFields'] = processing.run('native:retainfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(8)
 if feedback.isCanceled():
  return {}

 # Local_Network_cost
 alg_params = {
  'FIELDS': ['name','Surface','Dimos'],
  'INPUT': parameters['Local'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Local_network_cost'] = processing.run('native:retainfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(9)
 if feedback.isCanceled():
  return {}

 # Kassetines
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'kasetines',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 1,
  'FORMULA': '\"count\"*15',
```

```
  'INPUT': outputs['Kasetines']['OUTPUT'],
  'OUTPUT':
'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/kasetines',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Kassetines'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)


 feedback.setCurrentStep(10)
 if feedback.isCanceled():
  return {}


 # SurfaceCost
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'SurfaceCost',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'Case\r\nWhen \"Surface\" = \'asphalt\' then round($length *17 ,2)\r\nWhen \"Surface\" = \'paving_stones\' then round($length* 30,2)\r\nWhen \"Surface\" = \'NULL\' then round($length* 17,2)\r\nWhen \"Surface\" = \'compacted\' then round($length* 17,2)\r\nWhen \"Surface\" = \'concrete:plates\' then round($length* 40,2)\r\nWhen \"Surface\" = \'dirt\' then round($length* 10,2)\r\nWhen \"Surface\" = \'earth\' then round($length* 10,2)\r\nWhen \"Surface\" = \'grass\' then round($length* 20,2)\r\nWhen \"Surface\" = \'gravel\' then round($length* 17,2)\r\nWhen \"Surface\" = \'ground\' then round($length* 17,2)\r\nWhen \"Surface\" = \'metal\' then round($length* 100,2)\r\nWhen \"Surface\" = \'mud\' then round($length* 17,2)\r\nWhen \"Surface\" = \'paved\' then round($length* 30,2)\r\nWhen \"Surface\" = \'cobblestone\' then round($length* 35,2)\r\nWhen \"Surface\" = \'pebblestone\' then round($length* 45,2)\r\nWhen \"Surface\" = \'sett\' then round($length* 17,2)\r\nWhen \"Surface\" = \'steps\' then round($length* 45,2)\r\nWhen \"Surface\" = \'stone\' then round($length* 30,2)\r\nWhen \"Surface\" = \'unpaved\' then round($length* 14,2)\r\nWhen \"Surface\" = \'wood\' then round($length* 17,2)\r\nelse\r\nround($length* 17,2)\r\nend',
  'INPUT': outputs['Local_network_cost']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Surfacecost'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
```

```
 feedback.setCurrentStep(11)
 if feedback.isCanceled():
  return {}

 # Difference
 alg_params = {
  'INPUT': parameters['Clientsnetwork'],
  'OVERLAY': parameters['Local'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Difference'] = processing.run('native:difference', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(12)
 if feedback.isCanceled():
  return {}

 # kerdos
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'kerdos',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 1,
  'FORMULA': 'Diamerismata*0.3*25*12',
  'INPUT': outputs['RetainFields']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Kerdos'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(13)
 if feedback.isCanceled():
  return {}
```

```
 # freatia
 alg_params = {
  'FIELDS': ['number'],
  'INPUT': parameters['freatia'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Freatia'] = processing.run('native:retainfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(14)
 if feedback.isCanceled():
  return {}

 # Freatiasta
 alg_params = {
  'CATEGORIES_FIELD_NAME': [''],
  'INPUT': outputs['Freatia']['OUTPUT'],
  'VALUES_FIELD_NAME': '',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Freatiasta'] = processing.run('qgis:statisticsbycategories', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(15)
 if feedback.isCanceled():
  return {}

 # sinthetires
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'sinthetires',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 1,
```

```
 'FORMULA': 'Case\r\nWhen \"splitter\" = \'2 way splitter\' then 3\r\nWhen \"splitter\" = \'3 way splitter\' then 4\r\nWhen \"splitter\" = \'4 way splitter\' then 5\r\nWhen \"splitter\" = \'6 way splitter\' then 7\r\nWhen \"splitter\" = \'8 way splitter\' then 9\r\nWhen \"splitter\" = \'32 way splitter\' then 33\r\nend',
 'INPUT': outputs['Splitterssunthetires']['OUTPUT'],
 'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Sinthetires'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(16)
 if feedback.isCanceled():
  return {}

 # Cable_length
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'length',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'round($length,2)',
  'INPUT': outputs['RetainFields']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Cable_length'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(17)
 if feedback.isCanceled():
  return {}

 # Freatia
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'Freatia_cost',
```

```
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': '\"count\"*450',
  'INPUT': outputs['Freatiasta']['OUTPUT'],
  'OUTPUT':
'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/freatia',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['Freatia'] = processing.run('native:fieldcalculator', alg_params, context=context,
feedback=feedback, is_child_algorithm=True)

  feedback.setCurrentStep(18)
  if feedback.isCanceled():
   return {}

  # Join attributes by location
  alg_params = {
  'DISCARD_NONMATCHING': False,
  'INPUT': outputs['DeleteDuplicateGeometries']['OUTPUT'],
  'JOIN': parameters['main'],
  'JOIN_FIELDS': [''],
  'METHOD': 0,
  'PREDICATE': [3],
  'PREFIX': '',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['JoinAttributesByLocation'] = processing.run('native:joinattributesbylocation', alg_params,
context=context, feedback=feedback, is_child_algorithm=True)

  feedback.setCurrentStep(19)
  if feedback.isCanceled():
   return {}

  # Moufes
```

```
 alg_params = {
  'CATEGORIES_FIELD_NAME': [''],
  'INPUT': outputs['Freatia']['OUTPUT'],
  'VALUES_FIELD_NAME': '',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Moufes'] = processing.run('qgis:statisticsbycategories', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(20)
 if feedback.isCanceled():
  return {}

 # kasetines
 alg_params = {
  'CATEGORIES_FIELD_NAME': [''],
  'INPUT': outputs['Kassetines']['OUTPUT'],
  'OUTPUT': 'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/kasetines_sunolo',
  'VALUES_FIELD_NAME': 'kasetines',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Kasetines'] = processing.run('qgis:statisticsbycategories', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(21)
 if feedback.isCanceled():
  return {}

 # Add field to attributes table
 alg_params = {
  'FIELD_LENGTH': 10,
  'FIELD_NAME': 'length',
```

```
 'FIELD_PRECISION': 0,
 'FIELD_TYPE': 1,
 'INPUT': outputs['Surfacecost']['OUTPUT'],
 'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['AddFieldToAttributesTable'] = processing.run('native:addfieldtoattributestable', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(22)
 if feedback.isCanceled():
  return {}

 # cost_sinthetires
 alg_params = {
 'FIELD_LENGTH': 0,
 'FIELD_NAME': 'cost_sinthetires',
 'FIELD_PRECISION': 0,
 'FIELD_TYPE': 0,
 'FORMULA': 'sinthetires*0.5',
 'INPUT': outputs['Sinthetires']['OUTPUT'],
 'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Cost_sinthetires'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(23)
 if feedback.isCanceled():
  return {}

 # clients_Network_cost
 alg_params = {
 'FIELDS': ['name','Surface','Dimos'],
 'INPUT': outputs['Difference']['OUTPUT'],
 'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
```

```
 }
 outputs['Clients_network_cost'] = processing.run('native:retainfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(24)
 if feedback.isCanceled():
  return {}

 # cable_cost2
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'cable_cost',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': '$length*1.23',
  'INPUT': outputs['AddFieldToAttributesTable']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Cable_cost2'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(25)
 if feedback.isCanceled():
  return {}

 # length
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'length',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'round($length,2)',
  'INPUT': outputs['Cable_cost2']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
```

```
 }
 outputs['Length'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(26)
 if feedback.isCanceled():
  return {}

 # splitters
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'splitters_cost',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 1,
  'FORMULA': 'Case\r\nWhen \"splitter\" = \'2 way splitter\' then 3.5\r\nWhen \"splitter\" = \'3 way splitter\' then 15\r\nWhen \"splitter\" = \'4 way splitter\' then 6\r\nWhen \"splitter\" = \'6 way splitter\' then 8\r\nWhen \"splitter\" = \'8 way splitter\' then 12\r\nWhen \"splitter\" = \'32 way splitter\' then 35\r\nend',
  'INPUT': outputs['Cost_sinthetires']['OUTPUT'],
  'OUTPUT': 'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/splitters',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Splitters'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(27)
 if feedback.isCanceled():
  return {}

 # bankG
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'bankG',
  'FIELD_PRECISION': 0,
```

```
  'FIELD_TYPE': 0,
  'FORMULA': 'Case \r\nWhen \"Dimos\" = \'Δημος Θεσσαλονικης\' Then\r\nround($length * 2 ,2)\r\nWhen \"Dimos\" = \'Δημος Καλαμαριας\' Then\r\nround($length * 3 ,2)\r\nWhen \"Dimos\" = \'Δημος Αμπελοκηπων - Μενεμενης\' Then\r\nround($length * 1.5 ,2)\r\nend\r\n ',
  'INPUT': outputs['Length']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Bankg'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(28)
 if feedback.isCanceled():
  return {}

 # cable_cost_clinets77
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'cable_cost',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 1,
  'FORMULA': '$length*2',
  'INPUT': outputs['Cable_length']['OUTPUT'],
  'OUTPUT':
'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/clients_cable_Network_cost',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Cable_cost_clinets77'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(29)
 if feedback.isCanceled():
  return {}

 # Freatia_cost
```

```
 alg_params = {
  'CATEGORIES_FIELD_NAME': [''],
  'INPUT': outputs['Freatia']['OUTPUT'],
  'OUTPUT':
'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/freatia_sunolo',
  'VALUES_FIELD_NAME': 'Freatia_cost',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Freatia_cost'] = processing.run('qgis:statisticsbycategories', alg_params, context=context,
feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(30)
 if feedback.isCanceled():
  return {}

 # splitters
 alg_params = {
  'CATEGORIES_FIELD_NAME': [''],
  'INPUT': outputs['Splitters']['OUTPUT'],
  'OUTPUT':
'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/splitters_splitters
_cost',
  'VALUES_FIELD_NAME': 'splitters_cost',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Splitters'] = processing.run('qgis:statisticsbycategories', alg_params, context=context,
feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(31)
 if feedback.isCanceled():
  return {}

 # cable_cost
 alg_params = {
```

```
  'CATEGORIES_FIELD_NAME': [''],
  'INPUT': outputs['Cable_cost_clinets77']['OUTPUT'],
  'OUTPUT':
'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/clients_cable_Network_cost_sunolo',
  'VALUES_FIELD_NAME': 'cable_cost',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Cable_cost'] = processing.run('qgis:statisticsbycategories', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(32)
 if feedback.isCanceled():
  return {}

 # Moufes
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'Moufes_cost',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': '\"count\"*300',
  'INPUT': outputs['Moufes']['OUTPUT'],
  'OUTPUT':
'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/moufes',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Moufes'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(33)
 if feedback.isCanceled():
  return {}
```

```
 # Retain fields
 alg_params = {
  'FIELDS': ['Dimos'],
  'INPUT': outputs['JoinAttributesByLocation']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['RetainFields'] = processing.run('native:retainfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(34)
 if feedback.isCanceled():
  return {}

 # length
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'length',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'round($length,2)',
  'INPUT': outputs['RetainFields']['OUTPUT'],
  'OUTPUT': 'TEMPORARY_OUTPUT',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Length'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(35)
 if feedback.isCanceled():
  return {}

 # teli_Xrisis
 alg_params = {
  'FIELD_LENGTH': 0,
```

```
  'FIELD_NAME': 'teli_Xrisis',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'Case \r\nWhen \"Dimos\" = \'Δημος Θεσσαλονικης\' Then\r\nround($length * 0.175 ,2)\r\nWhen \"Dimos\" = \'Δημος Καλαμαριας\' Then\r\nround($length * 0.282 ,2)\r\nWhen \"Dimos\" = \'Δημος Αμπελοκηπων - Μενεμενης\' Then\r\nround($length * 0.175 ,2)\r\nend\r\n ',
  'INPUT': outputs['Bankg']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Teli_xrisis'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(36)
 if feedback.isCanceled():
  return {}

 # Moufes_cost
 alg_params = {
  'CATEGORIES_FIELD_NAME': [''],
  'INPUT': outputs['Moufes']['OUTPUT'],
  'OUTPUT': 'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/moufes_sunolo',
  'VALUES_FIELD_NAME': 'Moufes_cost',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Moufes_cost'] = processing.run('qgis:statisticsbycategories', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(37)
 if feedback.isCanceled():
  return {}

 # SurfaceCost
 alg_params = {
```

```
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'SurfaceCost',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'Case\r\nWhen \"Surface\" = \'asphalt\' then round($length *17 ,2)\r\nWhen \"Surface\" = \'paving_stones\' then round($length* 30,2)\r\nWhen \"Surface\" = \'NULL\' then round($length* 17,2)\r\nWhen \"Surface\" = \'compacted\' then round($length* 17,2)\r\nWhen \"Surface\" = \'concrete:plates\' then round($length* 40,2)\r\nWhen \"Surface\" = \'dirt\' then round($length* 10,2)\r\nWhen \"Surface\" = \'earth\' then round($length* 10,2)\r\nWhen \"Surface\" = \'grass\' then round($length* 20,2)\r\nWhen \"Surface\" = \'gravel\' then round($length* 17,2)\r\nWhen \"Surface\" = \'ground\' then round($length* 17,2)\r\nWhen \"Surface\" = \'metal\' then round($length* 100,2)\r\nWhen \"Surface\" = \'mud\' then round($length* 17,2)\r\nWhen \"Surface\" = \'paved\' then round($length* 30,2)\r\nWhen \"Surface\" = \'cobblestone\' then round($length* 35,2)\r\nWhen \"Surface\" = \'pebblestone\' then round($length* 45,2)\r\nWhen \"Surface\" = \'sett\' then round($length* 17,2)\r\nWhen \"Surface\" = \'steps\' then round($length* 45,2)\r\nWhen \"Surface\" = \'stone\' then round($length* 30,2)\r\nWhen \"Surface\" = \'unpaved\' then round($length* 14,2)\r\nWhen \"Surface\" = \'wood\' then round($length* 17,2)\r\nelse\r\nround($length* 17,2)\r\nend',
  'INPUT': outputs['Clients_network_cost']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Surfacecost'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(38)
 if feedback.isCanceled():
  return {}

 # sithetires
 alg_params = {
  'CATEGORIES_FIELD_NAME': [''],
  'INPUT': outputs['Splitters']['OUTPUT'],
  'OUTPUT':
'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/splitters_cost_sinthetires',
  'VALUES_FIELD_NAME': 'cost_sinthetires',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
```

```
 outputs['Sithetires'] = processing.run('qgis:statisticsbycategories', alg_params, context=context, feedback=feedback, is_child_algorithm=True)


 feedback.setCurrentStep(39)
 if feedback.isCanceled():
  return {}


 # cable_cost2
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'cable_cost',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': '$length*1.23',
  'INPUT': outputs['Length']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Cable_cost2'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)


 feedback.setCurrentStep(40)
 if feedback.isCanceled():
  return {}


 # bankG2
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'bankG',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'Case \r\nWhen \"Dimos\" = \'Δημος Θεσσαλονικης\' Then\r\nround($length * 2 ,2)\r\nWhen \"Dimos\" = \'Δημος Καλαμαριας\' Then\r\nround($length * 3 ,2)\r\nWhen \"Dimos\" = \'Δημος Αμπελοκηπων - Μενεμενης\' Then\r\nround($length * 1.5 ,2)\r\nend\r\n ',
  'INPUT': outputs['Cable_cost2']['OUTPUT'],
```

```
 'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Bankg2'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(41)
 if feedback.isCanceled():
  return {}

 # teli_Xrisis2
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'teli_Xrisis',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'Case \r\nWhen \"Dimos\" = \'Δημος Θεσσαλονικης\' Then\r\nround($length * 0.175 ,2)\r\nWhen \"Dimos\" = \'Δημος Καλαμαριας\' Then\r\nround($length * 0.282 ,2)\r\nWhen \"Dimos\" = \'Δημος Αμπελοκηπων - Μενεμενης\' Then\r\nround($length * 0.175 ,2)\r\nend\r\n ',
  'INPUT': outputs['Bankg2']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Teli_xrisis2'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(42)
 if feedback.isCanceled():
  return {}

 # Add field to attributes table
 alg_params = {
  'FIELD_LENGTH': 10,
  'FIELD_NAME': 'length',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 1,
```

```
 'INPUT': outputs['Surfacecost']['OUTPUT'],
 'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['AddFieldToAttributesTable'] = processing.run('native:addfieldtoattributestable', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(43)
 if feedback.isCanceled():
  return {}

 # Xomatourgika2
 alg_params = {
 'FIELD_LENGTH': 0,
 'FIELD_NAME': 'Xomatourgika',
 'FIELD_PRECISION': 0,
 'FIELD_TYPE': 0,
 'FORMULA': 'Case \r\nWhen \"Dimos\" = \'Δημος Θεσσαλονικης\' Then\r\nround($length * 40 ,2)\r\nWhen \"Dimos\" = \'Δημος Καλαμαριας\' Then\r\nround($length * 40 ,2)\r\nWhen \"Dimos\" = \'Δημος Αμπελοκηπων - Μενεμενης\' Then\r\nround($length * 50 ,2)\r\nend\r\n ',
 'INPUT': outputs['Teli_xrisis2']['OUTPUT'],
 'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Xomatourgika2'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(44)
 if feedback.isCanceled():
  return {}

 # Xomatourgika
 alg_params = {
 'FIELD_LENGTH': 0,
 'FIELD_NAME': 'Xomatourgika',
 'FIELD_PRECISION': 0,
```

```
  'FIELD_TYPE': 0,
  'FORMULA': 'Case \r\nWhen \"Dimos\" = \'Δημος Θεσσαλονικης\' Then\r\nround($length * 1 + SurfaceCost ,2)\r\nWhen \"Dimos\" = \'Δημος Καλαμαριας\' Then\r\nround($length * 1 + SurfaceCost ,2)\r\nWhen \"Dimos\" = \'Δημος Αμπελοκηπων - Μενεμενης\' Then\r\nround($length * 1 + SurfaceCost ,2)\r\nend\r\n ',
  'INPUT': outputs['Teli_xrisis']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Xomatourgika'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(45)
 if feedback.isCanceled():
  return {}

 # sunolo
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'sunolo ergou',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'round(teli_Xrisis + bankG + Xomatourgika ,2) ',
  'INPUT': outputs['Xomatourgika']['OUTPUT'],
  'OUTPUT':
'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/Local_Network_cost',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Sunolo'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(46)
 if feedback.isCanceled():
  return {}
```

```
 # Sunolo_trench
 alg_params = {
  'CATEGORIES_FIELD_NAME': [''],
  'INPUT': outputs['Sunolo']['OUTPUT'],
  'OUTPUT':
'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/Local_Network_
cost_sunolo',
  'VALUES_FIELD_NAME': 'sunolo ergou',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Sunolo_trench'] = processing.run('qgis:statisticsbycategories', alg_params, context=context,
feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(47)
 if feedback.isCanceled():
  return {}

 # length
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'length',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'round($length,2)',
  'INPUT': outputs['AddFieldToAttributesTable']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Length'] = processing.run('native:fieldcalculator', alg_params, context=context,
feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(48)
 if feedback.isCanceled():
  return {}
```

```
 # sunolo2
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'sunolo ergou',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'round(teli_Xrisis + bankG + Xomatourgika + Cable_cost ,2)\r\n',
  'INPUT': outputs['Xomatourgika2']['OUTPUT'],
  'OUTPUT':
'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/Main_Network_
cost',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Sunolo2'] = processing.run('native:fieldcalculator', alg_params, context=context,
feedback=feedback, is_child_algorithm=True)


 feedback.setCurrentStep(49)
 if feedback.isCanceled():
  return {}


 # bankG
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'bankG',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'Case \r\nWhen \"Dimos\" = \'Δημος Θεσσαλονικης\' Then\r\nround($length * 2
,2)\r\nWhen \"Dimos\" = \'Δημος Καλαμαριας\' Then\r\nround($length * 3 ,2)\r\nWhen \"Dimos\" =
\'Δημος Αμπελοκηπων - Μενεμενης\' Then\r\nround($length * 1.5 ,2)\r\nend\r\n ',
  'INPUT': outputs['Length']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Bankg'] = processing.run('native:fieldcalculator', alg_params, context=context,
feedback=feedback, is_child_algorithm=True)
```

```
 feedback.setCurrentStep(50)
 if feedback.isCanceled():
  return {}

 # teli_Xrisis
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'teli_Xrisis',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'Case \r\nWhen \"Dimos\" = \'Δημος Θεσσαλονικης\' Then\r\nround($length * 0.175 ,2)\r\nWhen \"Dimos\" = \'Δημος Καλαμαριας\' Then\r\nround($length * 0.282 ,2)\r\nWhen \"Dimos\" = \'Δημος Αμπελοκηπων - Μενεμενης\' Then\r\nround($length * 0.175 ,2)\r\nend\r\n ',
  'INPUT': outputs['Bankg']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Teli_xrisis'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(51)
 if feedback.isCanceled():
  return {}

 # Sunolo_trench2
 alg_params = {
  'CATEGORIES_FIELD_NAME': [''],
  'INPUT': outputs['Sunolo2']['OUTPUT'],
  'OUTPUT': 'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/Main_Network_cost_sunolo',
  'VALUES_FIELD_NAME': 'sunolo ergou',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Sunolo_trench2'] = processing.run('qgis:statisticsbycategories', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
```

```
 feedback.setCurrentStep(52)
 if feedback.isCanceled():
  return {}

 # Xomatourgika
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'Xomatourgika',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'Case \r\nWhen \"Dimos\" = \'Δημος Θεσσαλονικης\' Then\r\nround($length * 1 + SurfaceCost ,2)\r\nWhen \"Dimos\" = \'Δημος Καλαμαριας\' Then\r\nround($length * 1 + SurfaceCost ,2)\r\nWhen \"Dimos\" = \'Δημος Αμπελοκηπων - Μενεμενης\' Then\r\nround($length * 1 + SurfaceCost ,2)\r\nend\r\n ',
  'INPUT': outputs['Teli_xrisis']['OUTPUT'],
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Xomatourgika'] = processing.run('native:fieldcalculator', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(53)
 if feedback.isCanceled():
  return {}

 # sunolo3
 alg_params = {
  'FIELD_LENGTH': 0,
  'FIELD_NAME': 'sunolo ergou',
  'FIELD_PRECISION': 0,
  'FIELD_TYPE': 0,
  'FORMULA': 'round(teli_Xrisis + bankG + Xomatourgika ,2)\r\n',
  'INPUT': outputs['Xomatourgika']['OUTPUT'],
```

```
  'OUTPUT':
'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/clients_Network
_cost',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['Sunolo3'] = processing.run('native:fieldcalculator', alg_params, context=context,
feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(54)
 if feedback.isCanceled():
  return {}

 # sunolo ergou3
 alg_params = {
  'CATEGORIES_FIELD_NAME': [''],
  'INPUT': outputs['Sunolo3']['OUTPUT'],
  'OUTPUT':
'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/clients_Network
_cost_sunolo',
  'VALUES_FIELD_NAME': 'sunolo ergou',
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
 }
 outputs['SunoloErgou3'] = processing.run('qgis:statisticsbycategories', alg_params, context=context,
feedback=feedback, is_child_algorithm=True)

 feedback.setCurrentStep(55)
 if feedback.isCanceled():
  return {}

 # Merge vector layers
 alg_params = {
  'CRS': 'ProjectCrs',
  'LAYERS':
[outputs['Sunolo_trench']['OUTPUT'],outputs['Sunolo_trench2']['OUTPUT'],outputs['Freatia_cost']['O
UTPUT'],outputs['Moufes_cost']['OUTPUT'],outputs['SunoloErgou3']['OUTPUT'],outputs['Kasetines']
```

['OUTPUT'],outputs['Cable_cost']['OUTPUT'],outputs['Splitters']['OUTPUT'],outputs['Sithetires']['OUTPUT']],

'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT

}

outputs['MergeVectorLayers'] = processing.run('native:mergevectorlayers', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

feedback.setCurrentStep(56)

if feedback.isCanceled():

return {}

Retain fields

alg_params = {

'FIELDS': ['name','sum'],

'INPUT': outputs['MergeVectorLayers']['OUTPUT'],

'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT

}

outputs['RetainFields'] = processing.run('native:retainfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

feedback.setCurrentStep(57)

if feedback.isCanceled():

return {}

Export to spreadsheet

alg_params = {

'FORMATTED_VALUES': False,

'LAYERS': [outputs['Sunolo']['OUTPUT'],outputs['Sunolo2']['OUTPUT'],outputs['Freatia']['OUTPUT'],outputs['Splitters']['OUTPUT'],outputs['Moufes']['OUTPUT'],outputs['Sunolo3']['OUTPUT'],outputs['Cable_cost_clinets77']['OUTPUT'],outputs['Kassetines']['OUTPUT'],outputs['RetainFields']['OUTPUT'],outputs['Splitters']['OUTPUT'],outputs['Kerdos']['OUTPUT']],

'OUTPUT': 'C:/Users/gas19/OneDrive/Υπολογιστής/testme.xlsx',

'OVERWRITE': True,

'USE_ALIAS': False,

```
  'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['ExportToSpreadsheet'] = processing.run('native:exporttospreadsheet', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
  return results

 def name(self):
  return 'Step_6_excel'

 def displayName(self):
  return 'Step_6_excel'

 def group(self):
  return 'Gekas_Dimitris'

 def groupId(self):
  return 'Gekas_Dimitris'

 def createInstance(self):
  return Step_6_excel()
```

ΠΑΡΑΡΤΗΜΑ Ζ : Εξαγωγή Συντεταγμένων σε Excel

"""

"""

Model exported as python.

Name : Step_7 - suntetagmenes

Group : Gekas_Dimitris

With QGIS : 31802

"""

```
from qgis.core import QgsProcessing
from qgis.core import QgsProcessingAlgorithm
from qgis.core import QgsProcessingMultiStepFeedback
from qgis.core import QgsProcessingParameterVectorLayer
from qgis.core import QgsProcessingParameterBoolean
import processing


class Step_7Suntetagmenes(QgsProcessingAlgorithm):

  def initAlgorithm(self, config=None):
    self.addParameter(QgsProcessingParameterVectorLayer('Clientsnetwork', 'Clients_network', types=[QgsProcessing.TypeVectorLine], defaultValue=None))
    self.addParameter(QgsProcessingParameterVectorLayer('Freatia', 'Freatia', types=[QgsProcessing.TypeVectorPoint], defaultValue=None))
    self.addParameter(QgsProcessingParameterVectorLayer('Mainnetwork', 'Main_network', types=[QgsProcessing.TypeVectorLine], defaultValue=None))
    self.addParameter(QgsProcessingParameterVectorLayer('TrenchNetwork', 'Trench_Network', types=[QgsProcessing.TypeVectorLine], defaultValue=None))
    self.addParameter(QgsProcessingParameterBoolean('VERBOSE_LOG', 'Verbose logging', optional=True, defaultValue=False))

  def processAlgorithm(self, parameters, context, model_feedback):
    # Use a multi-step feedback, so that individual child algorithm progress reports are adjusted for the
    # overall progress through the model
```

```
    feedback = QgsProcessingMultiStepFeedback(12, model_feedback)
    results = {}
    outputs = {}

    # Points along geometry
    alg_params = {
     'DISTANCE': 5,
     'END_OFFSET': 0,
     'INPUT': parameters['Clientsnetwork'],
     'START_OFFSET': 0,
     'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
    }
    outputs['PointsAlongGeometry'] = processing.run('native:pointsalonglines', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

    feedback.setCurrentStep(1)
    if feedback.isCanceled():
     return {}

    # Add X/Y fields to layer
    alg_params = {
     'CRS': 'ProjectCrs',
     'INPUT': outputs['PointsAlongGeometry']['OUTPUT'],
     'PREFIX': '',
     'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
    }
    outputs['AddXyFieldsToLayer'] = processing.run('native:addxyfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

    feedback.setCurrentStep(2)
    if feedback.isCanceled():
     return {}

    # Points along geometry
```

```
    alg_params = {
     'DISTANCE': 5,
     'END_OFFSET': 0,
     'INPUT': parameters['Mainnetwork'],
     'START_OFFSET': 0,
     'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
    }
    outputs['PointsAlongGeometry'] = processing.run('native:pointsalonglines', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

    feedback.setCurrentStep(3)
    if feedback.isCanceled():
     return {}

    # Add X/Y fields to layer
    alg_params = {
     'CRS': 'ProjectCrs',
     'INPUT': parameters['Freatia'],
     'PREFIX': '',
     'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
    }
    outputs['AddXyFieldsToLayer'] = processing.run('native:addxyfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

    feedback.setCurrentStep(4)
    if feedback.isCanceled():
     return {}

    # Add X/Y fields to layer
    alg_params = {
     'CRS': 'ProjectCrs',
     'INPUT': outputs['PointsAlongGeometry']['OUTPUT'],
     'PREFIX': '',
     'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
```

```
    }
    outputs['AddXyFieldsToLayer'] = processing.run('native:addxyfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

    feedback.setCurrentStep(5)
    if feedback.isCanceled():
      return {}

    # Points along geometry
    alg_params = {
      'DISTANCE': 5,
      'END_OFFSET': 0,
      'INPUT': parameters['TrenchNetwork'],
      'START_OFFSET': 0,
      'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
    }
    outputs['PointsAlongGeometry'] = processing.run('native:pointsalonglines', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

    feedback.setCurrentStep(6)
    if feedback.isCanceled():
      return {}

    # Retain fields
    alg_params = {
      'FIELDS': ['odos','x','y'],
      'INPUT': outputs['AddXyFieldsToLayer']['OUTPUT'],
      'OUTPUT': 'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/suntetagmene_fr eation',
      'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
    }
    outputs['RetainFields'] = processing.run('native:retainfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
```

```
    feedback.setCurrentStep(7)
    if feedback.isCanceled():
        return {}

    # Retain fields
    alg_params = {
        'FIELDS': ['dimos','x','y'],
        'INPUT': outputs['AddXyFieldsToLayer']['OUTPUT'],
        'OUTPUT': 'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/suntetagmene_kentrikou_diktiou',
        'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
    }
    outputs['RetainFields'] = processing.run('native:retainfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

    feedback.setCurrentStep(8)
    if feedback.isCanceled():
        return {}

    # Retain fields
    alg_params = {
        'FIELDS': ['name','int_name','x','y'],
        'INPUT': outputs['AddXyFieldsToLayer']['OUTPUT'],
        'OUTPUT': 'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/suntetagmene_diktiou',
        'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
    }
    outputs['RetainFields'] = processing.run('native:retainfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

    feedback.setCurrentStep(9)
    if feedback.isCanceled():
```

```
        return {}

    # Add X/Y fields to layer
    alg_params = {
        'CRS': 'ProjectCrs',
        'INPUT': outputs['PointsAlongGeometry']['OUTPUT'],
        'PREFIX': '',
        'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
    }
    outputs['AddXyFieldsToLayer'] = processing.run('native:addxyfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

    feedback.setCurrentStep(10)
    if feedback.isCanceled():
        return {}

    # Retain fields
    alg_params = {
        'FIELDS': ['odos','x','y'],
        'INPUT': outputs['AddXyFieldsToLayer']['OUTPUT'],
        'OUTPUT': 'C:/Users/gas19/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/outputs/suntetagmene_trench_diktiou',
        'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
    }
    outputs['RetainFields'] = processing.run('native:retainfields', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

    feedback.setCurrentStep(11)
    if feedback.isCanceled():
        return {}

    # Export to spreadsheet
    alg_params = {
```

```
    'FORMATTED_VALUES': False,
    'LAYERS': [outputs['RetainFields']['OUTPUT'],outputs['RetainFields']['OUTPUT'],outputs['RetainFields']['OUTPUT'],outputs['RetainFields']['OUTPUT']],
    'OUTPUT': 'C:/Users/gas19/OneDrive/Υπολογιστής/suntetagmenes.xlsx',
    'OVERWRITE': True,
    'USE_ALIAS': False,
    'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
  }
  outputs['ExportToSpreadsheet'] = processing.run('native:exporttospreadsheet', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
  return results

 def name(self):
  return 'Step_7 - suntetagmenes'

 def displayName(self):
  return 'Step_7 - suntetagmenes'

 def group(self):
  return 'Gekas_Dimitris'

 def groupId(self):
  return 'Gekas_Dimitris'

 def createInstance(self):
  return Step_7Suntetagmenes()
```