ΤΕΙ ΚΕΝΤΡΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΤΕ

Ανάπτυξη εξελικτικού αλγορίθμου βελτιστοποίησης βασισμένου σε Ευφυείς Σταγόνες Νερού και εφαρμογή στην επίλυση πρακτικών προβλημάτων βελτιστοποίησης

Πτυχιακή Εργασία

Τανίδης Σπάρτακος (2640)

Επιβλέπων: Δρ. Σπυρίδων Α. Καζαρλής, Ηλεκτρολόγος Μηχανικός και Μηχανικός Ηλεκτρονικών Υπολογιστών

Σέρρες 23 Ιανουαρίου 2018

Περίληψη

Σκοπός της παρούσας πτυχιακής είναι η μελέτη του εξελικτικού αλγορίθμου βελτιστοποίησης βασισμένου σε Ευφυείς Σταγόνες Νερού (Intelligent Water Drop Algorithm - IWD) και η χρήση του στην επίλυση πρακτικών προβλημάτων βελτιστοποίησης. Αναλύονται τα βήματα του αλγορίθμου και δημιουργείται μια υλοποίησή του στην γλώσσα προγραμματισμού C++ ώστε να γίνει επίλυση του Travelling Salesman Problem (TSP). Ακολουθεί μελέτη των αποτελεσμάτων και της αποδοτικότητας του αλγορίθμου με διαφορετικές παραμέτρους και προδιαγραφές προβλημάτων.

Υπεύθυνη Δήλωση

Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής Τ.Ε. του Τ.Ε.Ι. Κεντρικής Μακεδονίας.

Ευχαριστίες

Ευχαριστώ την οικογένειά μου, τους φίλους μου, την σχολή μου και τον Tom Cruise.

Περιεχόμενα

Κατάλογος σχημάτων

1 Εισαγωγή

Τα τελευταία 40 χρόνια έγινε ραγδαία αύξηση της υπολογιστικής ισχύος των μηχανών. Πλέον είναι εμπορικά διαθέσιμοι επεξεργαστές με 12 πυρήνες των 4GHz ο καθένας. Αυτό έκανε επιτακτική την ανάγκη εφεύρεσης νέων αλγορίθμων που μπορούν να λειτουργήσουν παράλληλα, κάτι που είχε ως αποτέλεσμα την αυξανόμενη εξάρτησή μας στις μηχανές ως τρόπο επίλυσης προβλημάτων όπως αυτά της βελτιστοποίησης.

Παρ' όλα αυτά, πολλά από τα προβλήματα που καλείτε να λύση η επιστήμη των υπολογιστών είναι δύσκολο να λυθούν με συμβατικούς αλγορίθμους. Ακόμα και οι πιο ισχυροί υπέρ-υπολογιστές δυσκολεύονται να λύσουν έναν αρκετά μεγάλο αριθμό από προβλήματα που συναντούνται στην καθημερινότητα[1] χρησιμοποιώντας γραμμικό ή δυναμικό προγραμματισμό. Αυτό οδήγησε στην εξεύρεση νέων τρόπων επίλυσης προβλημάτων, ένας από τους οποίους είναι η **εξελικτική υπολογιστική**.

1.1 Εξελικτική Υπολογιστική

Η εξελικτική υπολογιστική προσπαθεί να προσομοιώσει τον τρόπο που λειτουργεί η βιολογική εξέλιξη. Παρατηρώντας των τρόπο που η φύση λύνει προβλήματα και διορθώνει ατέλειες με την πάροδο του χρόνου, μπορούμε να προσαρμόσουμε τις μεθόδους της στον τομέα των υπολογιστών.

Τα πρώτα βήματα έγιναν την δεκαετία του '80 με τους πρώτους ονομαζόμενους 'γενετικούς αλγόριθμους'[2]. Με την πάροδο του χρόνου άρχισαν να τυποποιούνται και να προσαρμόζονται κάποιες βασικές έννοιες όπως 'αναπαραγωγή', 'μετάλλαξη' και 'φυσική επιλογή' στον τομέα την εξελικτικής υπολογιστικής. Οι μέθοδοι τις οποίες χρησιμοποιεί η φύση για την βελτίωση των ατελειών κάθε οργανισμού προσαρμόστηκαν στις ανάγκες μας για την εύρεση λύσεων σε δύσκολα προβλήματα. Σε αντίθεση όμως με την φύση, που μπορεί να χρειαστεί ακόμα και αιώνες για κάθε γενιά, οι υπολογιστές μπορούν να τρέξουν προσομοιώσεις χιλιάδων γενεών σε λίγα δευτερόλεπτα.

Μερικά παραδείγματα φυσικών λειτουργιών που προσομοιώνονται με αλγορίθμους ή έχουν αποτελέσει πηγή έμπνευσης είναι τα εξής

- Ο τρόπος με τον οποίο τα μυρμήγκια επιλέγουν το μονοπάτι από την αποικία τους ως μία πηγή τροφής[3]
- Ο τρόπος με τον οποίο οι μέλισσες ανταλλάσσουν πληροφορίες μεταξύ τους κατά την περισυλλογή τροφής ώστε να βελτιώσουν την απόδοσή τους[4][5]
- Η επιρροή της μάζας ενός αντικειμένου στην βαρυτική του έλξη και το σύστημα που δημιουργούν πολλά τέτοια αντικείμενα. Στην περίπτωση αυτή, το αντικείμενο με την μεγαλύτερη μάζα αναπαριστά την βέλτιστη λύση[6]
- Ο τρόπος που οι παίκτες της Jazz μουσικής αυτοσχεδιάζουν και δημιουργούν νέους ρυθμούς [7]
- Ο τρόπος που οι πυγολαμπίδες αλλάζουν το επίπεδο φωτεινότητάς τους για να προσελκύσουν άλλες πυγολαμπίδες [8]

- Ο τρόπος κίνησης των βατράχων ώστε να βρουν περισσότερο φαγητό [9]
- Ο τρόπος που οι νυχτερίδες χρησιμοποιούν τον ηχοεντοπισμό έτσι ώστε να μοιράσουν ομοιόμορφα τον χώρο στον οποίο κινούνται, με αποτέλεσμα την μείωση των συγκρούσεων [10]

Παρατηρούμε ότι πολλοί από τους αλγόριθμους αυτούς βασίζονται στον μεγάλο πληθυσμό αντικειμένων ή οργανισμών. Αυτό ονομάζεται **νοημοσύνη σμήνους (swarm intelligence)**[11] όπου ο μεγάλος αριθμός ενεργών στοιχείων σε συνδυασμό με τις πολλές επαναλήψεις και την 'φυσική επιλογή' απαλείφουν τις ατέλειες ενός μεμονωμένου στοιχείου και οδηγούν στην βελτίωση του συνόλου. Φυσικά η ίδια λογική μπορεί να εφαρμοστεί και στους ανθρώπους, όπως όταν έγινε χρήση του διαδικτύου για την λήψη σημαντικών αποφάσεων[12].

Η βάση των περισσότερων γενετικών αλγορίθμων (ΓΑ) είναι η 'φυσική επιλογή'. Χρησιμοποιούνται ευρετικές (heuristic) μέθοδοι για την προσέγγιση λύσεων, δίνοντας έμφαση στις 'καλές' και απορρίπτοντας τις κακές, με σκοπό εύρεση της ιδεατής. Ένα πλεονέκτημα είναι ότι η πιθανή λύση μπορεί να βρεθεί σε πολύ μικρότερο χρονικό διάστημα από ότι θα έπαιρνε με δυναμικό/γραμμικό προγραμματισμό. Επίσης πολλές φορές είναι ιδιαίτερα δύσκολο έως αδύνατο να προσδιοριστεί μία τέτοια μέθοδος ή να δίνει λύση για πολύ περιορισμένο αριθμό προβλημάτων[1]. Φυσικά ένα μεγάλο μειονέκτημα είναι ότι δεν είναι βέβαιο ότι θα βρεθεί λύση, ή ότι η λύση που θα δοθεί θα πληρεί τις προδιαγραφές μας. Θα δούμε και στην παρούσα εργασία ότι, ενώ το πλησιάζουμε αρκετά κοντά , δεν βρίσκουμε το απόλυτο βέλτιστο.

1.2 Αλγόριθμος IWD

Ο εξελικτικός αλγόριθμος Intelligent Water Drop (IWD) ανήκει στην οικογένεια των αλγορίθμων **νοημοσύνης σμήνους**. Μιμείται την κίνηση ενός ποταμιού καθώς αυτό ρέει από την πηγή τους προς την θάλασσα και τις αλλαγές που φέρει η ροή αυτή του στο έδαφος. Το ποτάμι αποτελείται από έναν μεγάλο αριθμό από σταγόνες (σμήνος) και κάθε μία από αυτές έχει την δική της επιρροή στην διαδρομή. Με το να απορροφά και να εναποθέτει 'υλικό' από την επιφάνεια στην οποία κινείτε, επηρεάζει την διαδρομή των σταγόνων που την ακολουθούν. Η επιρροή κάθε σταγόνας μεμονωμένα είναι ελάχιστη αλλά, σαν σύνολο, ένα ποτάμι χαράσσει σε βάθος χρόνου μία διαδρομή που πολλές φορές είναι και η συντομότερη.

Πρώτη εφαρμογή των ιδιοτήτων αυτών σε αλγόριθμο βελτιστοποίησης έγινε από τον Hamed Shah-Hosseini το 2007[13] στην υλοποίησή του για την επίλυση του TSP. Ο Hosseini προσομοιώνει τις σταγόνες από τις οποίες αποτελείται ένα ποτάμι καθώς και το έδαφος επάνω στο οποίο κινείται και την αλληλεπίδραση μεταξύ τους. Η σταγόνα αφαιρεί ή εναποθέτει κομμάτια του εδάφους και το έδαφος φέρει αντίσταση στην κίνηση της σταγόνας. Κάθε σταγόνα αφήνει το ίχνος της και οι επόμενες σταγόνες τείνουν να ακολουθήσουν αυτές που είχαν το ευκολότερο ταξίδι. Μετά από κάθε γενιά έχουμε σημαντική αλλαγή στην τοπολογία του εδάφους.

Τα δύο σημαντικότερα χαρακτηριστικά μίας ευφυούς σταγόνας (στο εξής **IWD**) είναι η 'ταχύτητα' (στο εξής **velocity**) και το 'έδαφος' (στο εξής **soil**). Το velocity αντιπροσωπεύει την ταχύτητα με την οποία κινείται μία σταγόνα από περιοχή σε

περιοχή. Όπως ένα ποτάμι μεταφέρει σωματίδια με την κίνησή του, έτσι και οι σταγόνες νερού του IWD μεταφέρουν το 'έδαφος' με την κίνησή τους.

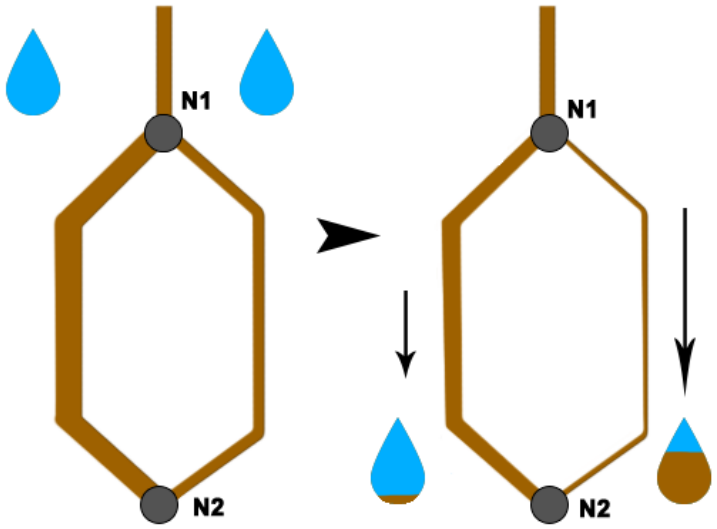
Δύο είναι οι βασικοί κανόνες που χαρακτηρίζουν το **soil** και το **velocity**.

1. Όσο πιο γρήγορη μία σταγόνα, τόσο περισσότερο soil αποφορά από το έδαφος στο οποίο κινείται

2. Όσο περισσότερο soil υπάρχει στο μονοπάτι μίας σταγόνας, τόσο πιο αργή είναι (μικρότερο velocity)

Ο συνδυασμός τους έχει ως αποτέλεσμα μία σταγόνα που είναι σε ένα μονοπάτι με πολύ soil να κινείται πιο αργά και να απορροφά λιγότερο soil από το έδαφος. Αντίθετα, μία σταγόνα που κινείται σε μονοπάτι με λίγο soil, κινείται ταχύτερα και απορροφά περισσότερο soil. Η επιλογή του μονοπατιού που θα ακολουθήσει μία σταγόνα επηρεάζεται από το πόσο soil υπάρχει στα διαθέσιμα μονοπάτια. Οι σταγόνες συνηθίζουν να επιλέγουν τα μονοπάτια με το λιγότερο soil, και άρα την λιγότερη αντίσταση[14]. Αυτό έχει ως αποτέλεσμα να 'χαράσσουν' μία πορεία που τείνει προς τη βέλτιστη δυνατή.

Κάθε σταγόνα αλλάζει το τοπίο από το οποίο περνάει σε ένα πολύ μικρό βαθμό. Με την πάροδο του χρόνου όμως, και με την χρήση πολλών σταγόνων, αυτή η 'αλλοίωση' γίνεται πιο εμφανής. Χρησιμοποιώντας κάποιες άλλες μεθόδους, όπως το να επιλέγουμε τα συντομότερα μονοπάτια κάθε γενιάς και να τα εμβαθύνουμε, μπορούμε καταλήξουμε σε μία πιθανή βέλτιστη λύση ακόμα γρηγορότερα.



Σχήμα 1: Παράδειγμα δύο σταγόνων που ακολουθούν διαφορετική διαδρομή

Ο αλγόριθμος παρουσιάζει πολλές ομοιότητες με τον αλγόριθμο Ant Colony Optimization (ACO) [15]. Σε αντίθεση όμως με τον ACO, που κάθε μυρμήγκι αποθέτει φερομόνες στην διαδρομή που ακολουθεί, οι οποίες προσελκύουν τα επόμενα, οι σταγόνες του IWD αφαιρούν έδαφος από το περιβάλλον τους ώστε να το κάνουν πιο θεμιτό για τις επόμενες. Επίσης ο αλγόριθμος προβλέπει την δυνατότητα να προστεθεί soil σε περίπτωση που είναι απαραίτητο.

Μερικά από τα προβλήματα στα οποία έχει εφαρμοστεί ο αλγόριθμος IWD είναι τα εξής

Travelling Salesman Problem (TSP) Ένα πολύ γνωστό πρόβλημα στο οποίο έγινε και η 1η υλοποίησή του αλγορίθμου. Περισσότερες πληροφορίες στην ενότητα 1.3

Multiple Knapsack Problem Προτείνεται ο IWD-MKP[16] που είναι ο αλγόριθμος IWD προσαρμοσμένος για την επίλυση του MKP, στο οποίο προσπαθούμε να βάλουμε αντικείμενα σε σακίδια με όσο το δυνατόν βέλτιστο τρόπο. Το MKP είναι μια γενικευμένη μορφή του Knapsack Problem[17] με πολλά σακίδια αντί για ένα.

n-Queen Probmel Η γενικευμένη μορφή του προβλήματος των 8 βασιλλισών[18], στο οποίο πρέπει να τοποθετηθούν 8 βασίλισσες πάνω σε μια σκακιέρα έτσι ώστε να μην αλληλοαπειλούνται. Έχει αποδειχθεί ότι ο αλγόριθμος IWD μπορεί να προσαρμοστεί ώστε να είναι δυνατή η επίλυση αυτού του προβλήματος[16].

Όπως φαίνεται και από τα παραπάνω παραδείγματα, με τις κατάλληλες μετατροπές, υπάρχει η δυνατότητα προσαρμογής του αλγορίθμου για την επίλυση ποικίλης μορφής προβλημάτων.

1.3 Πρόβλημα

Για την μελέτη του αλγορίθμου έγινε προσπάθεια επίλυσης του TSP. Στο TSP μας δίνεται ένας αριθμός από πόλεις και πρέπει να βρούμε την διαδρομή την οποία θα ακολουθήσει ένας πωλητής έτσι ώστε η απόσταση που θα διανύσει να είναι όσον το δυνατόν μικρότερη. Οι περιορισμοί είναι ότι δεν επιτρέπεται να περάσει δύο φορές από μία πόλη και ότι πρέπει να καταλήξει στην πόλη από την οποία ξεκίνησε.

Σχήμα 2: Παράδειγμα TSP και πιθανής λύσης του

Το TSP είναι ένα NP-Hard πρόβλημα και η επίλυσή του με γραμμικό προγραμματισμό είναι , ανάλογα με το μέγεθος του προβλήματος, ιδιαίτερα χρονοβόρα. Υπάρχουν κάποιες μέθοδοι και αλγόριθμοι που δίνουν λύσεις χρησιμοποιώντας δυναμικό προγραμματισμό[19] αλλά ακόμα και για μικρό αριθμό πόλεων οι λύσεις αυτές δεν είναι πάντα πρακτικές.

Στην δικιά μου εργασία μελετώ την κατηγορία των συμμετρικών TSP, στην οποία η απόσταση από μία πόλη Α προς μία πόλη Β είναι ίδια άσχετα με την κατεύθυνση κίνησης.

2 Μεθοδολογία

2.1 Μαθηματική ανάλυση προβλήματος και αλγορίθμου

Για την προσομοίωση του TSP δημιουργείται ένας γράφος του οποίοι οι **κόμβοι** αντιπροσωπεύουν τις πόλεις και οι **ακμές** τους δρόμους που τις συνδέουν. Ο βασικός αλγόριθμος IWD που χρησιμοποιούμε απαιτεί έναν πλήρη γράφο[13] έτσι κάθε πόλη είναι συνδεδεμένη με όλες τις υπόλοιπες.

Αρχικά δημιουργούμε μία σταγόνα για κάθε πόλη. Η αρχικοποίηση γίνεται βάση κάποιων παραμέτρων, οι οποίες τίθενται από τον αλγόριθμο.

Πίνακας 1: Παράμετροι μεταβολής σταγόνων και ενδεικτικές τιμές τους.

Παράμετρος	Τιμές	Επεξήγηση
InitSoil	1000	Το αρχικό soil κάθε ακμής
InitVel	200	Το αρχικό velocity κάθε σταγόνας
a_v	1	Παράμετρος μεταβολής soil
b_v	0.01	Παράμετρος μεταβολής soil
c_v	1	Παράμετρος μεταβολής soil
a_s	1	Παράμετρος μεταβολής velocity
b_s	0.01	Παράμετρος μεταβολής velocity
c_s	1	Παράμετρος μεταβολής velocity
ρ_n/ρ_{IWD}	0.9	Γενική παράμετρος μεταβολής. Πιθανές τιμές [0,1]
ϵ	0.0001	Πολύ μικρός θετικός αριθμός για αποφυγή διαίρεσης με το μηδέν

Οι μεταβλητές αυτές μπορούν να προσαρμοστούν βάση του προβλήματος που καλείται να λύση ο αλγόριθμος.

Μόλις αρχικοποιηθούν οι σταγόνες, μπορούν να ξεκινήσουν το "ταξίδι" τους. Κάθε σταγόνα διαθέτει μία λίστα V_N με τους κόμβους που έχει επισκεφθεί. Μία σταγόνα στον κόμβο i επιλέγει τον κόμβο j στον οποίο θα μετακινηθεί βάση της πιθανότητας του κόμβου $P_{i\longrightarrow j}$. Αυτή υπολογίζεται με τον τύπο

$$P_{i\longrightarrow j} = \frac{f(soil(i,j))}{\sum_{k\notin V_N} f(soil(i,k))} \tag{1}$$

με

$$f(soil(i,j)) = \frac{1}{\epsilon + g(soil(i,j))} \tag{2}$$

και

$$g(soil(i,j)) = \begin{cases} soil(i,j) & \text{Εάν } \min\limits_{l\notin V_N}(soil(i,l)) \geq 0 \\ soil(i,j) - \min\limits_{l\notin V_N}(soil(i,l)) & \text{αλλιώς} \end{cases} \tag{3}$$

Με $soil(i,j)$ ορίζουμε το soil που υπάρχει στην ακμή που ενώνει τους κόμβους (i,j).Το ϵ είναι ένας πολύ μικρός θετικός αριθμός που μας βοηθάει να αποφύγουμε την διαίρεση με το μηδέν. Αμέσως μετά ανανεώνουμε το velocity της σταγόνας σύμφωνα με τον τύπο (4) καθώς και το soil της σταγόνας βάση του(5).Βλέπουμε ότι οι κόμβοι με μικρότερο $soil$ έχουν μεγαλύτερη πιθανότητα να επιλεγούν.

$$vel^{IWD} = vel^{IWD} + \frac{a_v}{b_v + c_v \cdot soil^2(i,j)} \tag{4}$$

$$soil^{IWD} = soil^{IWD} + \Delta soil(i,j) \tag{5}$$

$$soil(i,j) = (1-\rho_n) \cdot soil(i,j) - \rho_n \cdot \Delta soil(i,j) \tag{6}$$

$$\Delta soil(i,j) = \frac{a_s}{b_s + c_s \cdot time(i,j)}$$

$$time(i,j) = \frac{HUD(i,j)}{\mathsf{max}(\epsilon, vel_{new})}$$

$$HUD(i,j) = \|C(i) - C(j)\| \tag{7}$$

Η συνάρτηση Heuristic Undesirability (HUD) $HUD(i,j)$(7) ορίζει την αντίσταση μίας σταγόνας να μετακινηθεί από τον κόμβο i στον j. Στην περίπτωση μας ορίζεται ως η Ευκλείδεια απόσταση μεταξύ τους.

Μετά το πέρας της διαδρομής κάθε σταγόνας, υπολογίζουμε την ποιότητα p της λύσης της. Στην περίπτωσή μας είναι η απόσταση της διαδρομής που ακολούθησε. Συγκρίνουμε τις αποστάσεις για κάθε σταγόνα IWD και θεωρούμε τη μικρότερη ως το *τοπικό βέλτιστο* T^{IB}. Για να κινηθεί ταχύτερα ο αλγόριθμος προς το ολικό βέλτιστο T^{IWD}, επιλέγουμε την διαδρομή που μας απέφερε το τοπικό βέλτιστο και αλλάζουμε το soil κατά μήκος της βάση του τύπου (8).

$$soil(i,j) = (1-\rho_{IWD}) \cdot soil(i,j) + \rho_{IWD} \cdot \frac{1}{N_{IB}-1} \cdot soil_{IB} \ , \ \forall (i,j) \in T^{IB} \ [16] \tag{8}$$

Με αυτόν τον τρόπο οι καλύτερες διαδρομές μίας γενιάς γίνονται ακόμα πιο θεμιτές από μεταγενέστερες σταγόνες. Στην περίπτωση που η T^{IWD} είναι χειρότερη από την T^{IB}, την αντικαθιστούμε με την νέα βέλτιστη λύση.

Μόλις ολοκληρώσουν όλες οι σταγόνες το ταξίδι τους, ελέγχουμε αν πληρούμε τις προϋποθέσεις για την λήξη του αλγορίθμου. Αυτές μπορούν να αποτελούνται από ένα συγκεκριμένο αριθμό "γενεών" ή κάποια ελάχιστη απαιτούμενη ποιότητα λύσης. Στην περίπτωσή μας ορίζουμε έναν μέγιστο αριθμό από επαναλήψεις που αντιπροσωπεύουν τις γενιές μας. Μετά το τέλος όλων των επαναλήψεων θα έχουμε πλησιάσει ή βρει την βέλτιστη λύση.

2.2 Βήματα αλγορίθμου

Συγκεκριμένα τα βήματα που ακολουθούμε είναι τα εξής :
1) Δημιουργούμε τον πλήρη γράφο $G(N,E)$ όπου N οι **κόμβοι** και E οι **ακμές**

1.1) Θέτουμε το $soil = InitSoil$ σε κάθε ακμή

2) Δημιουργούμε τις σταγόνες μας και τις αρχικοποιούμε

2.1) Για κάθε σταγόνα δημιουργούμε την άδεια λίστα V_N με τους κόμβους που έχει επισκεφθεί

2.2) Οι σταγόνες έχουν μηδενικό αρχικό $soil$

3) Θέτουμε $q(T^{IWD}) = +\infty$,

4) Ορίζουμε τις προϋποθέσεις τερματισμού του αλγορίθμου. Στην περίπτωσή μας ορίζουμε $iter_{max}$ ως τον μέγιστο αριθμό γενεών

5) Για κάθε σταγόνα...

5.1) Βάζουμε τον κόμβο στον οποίο βρίσκεται στην λίστα V_N

5.2) Επιλέγουμε τον κόμβο j στον οποίο θα μετακινηθούμε βάσει των πιθανοτήτων που υπολογίζονται με τον τύπο (1)

5.3) Ανανεώνουμε το $soil(i, j)$ της ακμής που ενώνει τους δύο κόμβους βάση του τύπου (6)

5.4) Ανανεώνουμε το $velocity^{IWD}$ της σταγόνας βάση του τύπου (4)

5.5) Ανανεώνουμε το $soil^{IWD}$ της σταγόνας βάση του τύπου (5)

6) Επαναλαμβάνουμε τα βήματα (5.1) έως και (5.5) μέχρι κάθε σταγόνα να ολοκληρώσει το ταξίδι της

7) Υπολογίζουμε το μήκος της διαδρομής T κάθε σταγόνας και βρίσκουμε το τοπικό βέλτιστο T^{IB}.

8) Ανανεώνουμε το $soil(i, j)$ κατά μήκος της διαδρομής T^{IB} σύμφωνα με τον τύπο (8)

9) Εάν $q(T^{IB}) > q(T^{TB})$ θέτουμε $T^{TB} = T^{IB}$

10) Αυξάνουμε τον αριθμό της τωρινής γενιάς $iter = iter + 1$. Εάν δεν πληρούμε τους κανόνες τερματισμού μεταφερόμαστε στο βήμα (5)

11) Ο αλγόριθμος έχει ολοκληρωθεί. Η λύση T^{TB} είναι η βέλτιστη διαδρομή

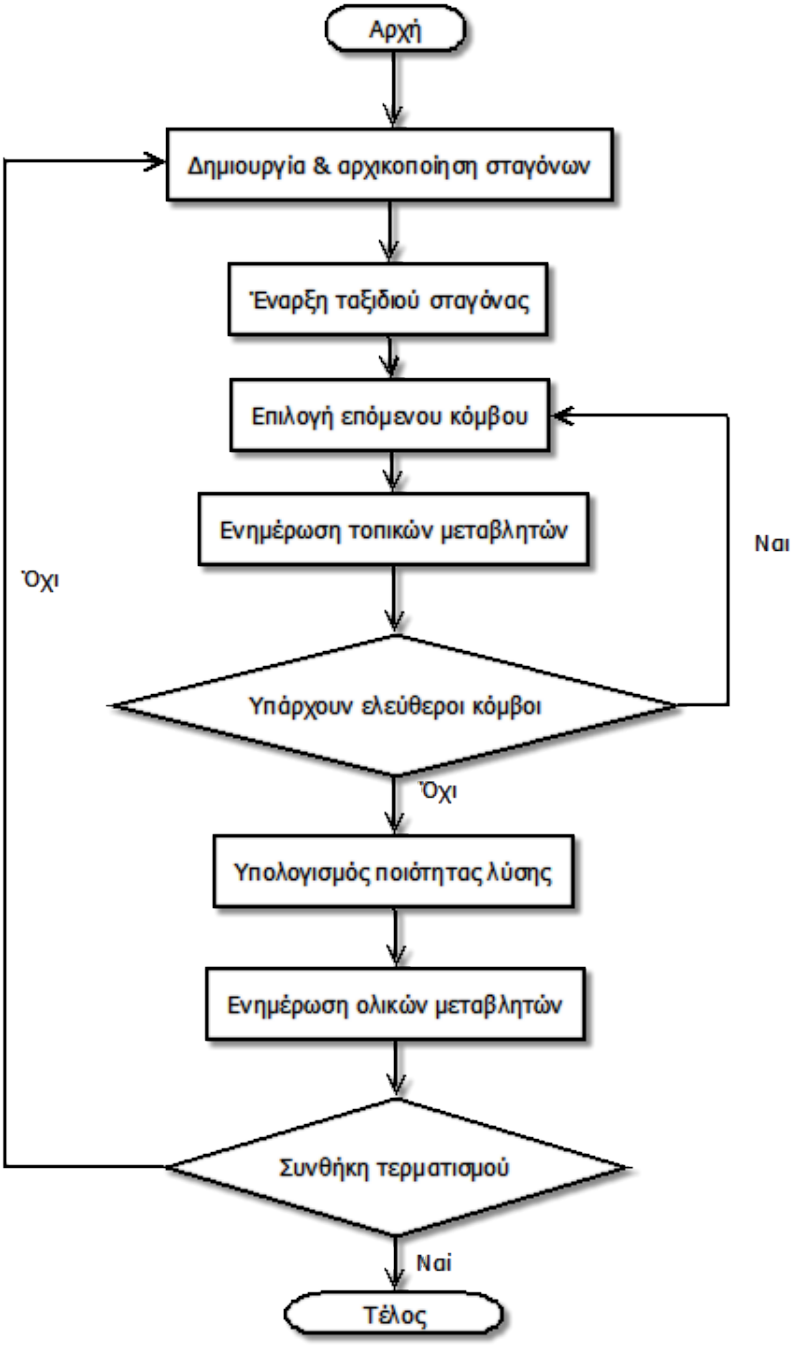
3 Υλοποίηση

Για την τελική υλοποίηση επιλέχθηκε η γλώσσα προγραμματισμού C++ σε περιβάλλον Microsoft Windows και με χρήση του IDE Visual Studio 2015 Community Edition (στο εξής VS2015). Δημιουργήθηκε μία εφαρμογή σε περιβάλλον κονσόλας με όνομα iwd-console και ένα GUI (iwd-gui) με την χρήση του .NET Framework.

3.1 Πρωτότυπο σε Javascript

Πριν την τελική υλοποίηση σε C++, δημιουργήθηκε ένα πρωτότυπο στην γλώσσα προγραμματισμού Javascript για να γίνει μελέτη του αλγορίθμου. Η υλοποίηση χρησιμοποιούσε την βιβλιοθήκη cytoscape.js[1] η οποία παρέχει όλες τις απαραίτητες λειτουργίες για την δημιουργία και επεξεργασία γράφων. Έτσι δημιουργήθηκε

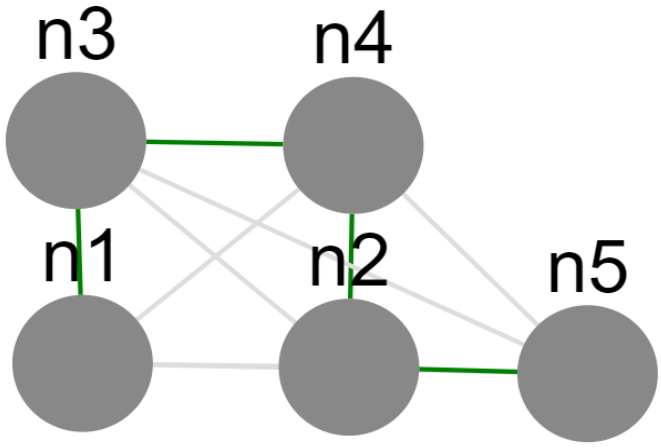
[1] `http://js.cytoscape.org`



Σχήμα 3: Διάγραμμα ροής εκτέλεσης αλγορίθμου IWD

εύκολα ένα πρωτότυπο πρόγραμμα για την μελέτη της λειτουργικότητας του αλγορίθμου.

Όπως βλέπουμε στο Σχήμα 4 ο αλγόριθμος κατέληγε στην βέλτιστη λύση σε μικρότερα προβλήματα (οι πράσινες γραμμές συμβολίζουν την διαδρομή που μας δίνεται ως βέλτιστη). Δυστυχώς οι επιδόσεις σε μεγαλύτερους γράφους δεν ήταν ικανοποιητικές και έτσι έγινε η επιλογή της C++.



Σχήμα 4: Αποτέλεσμα εκτέλεσης αλγορίθμου IWD σε περιβάλλον browser

3.2 Υλοποίηση σε C++

Για τους σκοπούς της εργασίας έγιναν δύο διαφορετικές εφαρμογές. Η `iwd-console` που λειτουργεί σε περιβάλλον κονσόλας και η `iwd-gui` που είναι παραθυρική εφαρμογή για το λειτουργικό Microsoft Windows. Επιλέχθηκε η έκδοση C++11 και η συγγραφή του κώδικα έγινε σε περιβάλλον Windows. Για compiler χρησιμοποιήθηκε ο `msvc40` που είναι ενσωματωμένος στο Visual Studio 2015/2017. Να σημειωθεί ότι η εφαρμογή κονσόλας μπορεί να γίνει compile με οποιονδήποτε compiler που υποστηρίζει C++11 μιας και χρησιμοποιήθηκαν μόνο εντολές της Standard C++ και συμβατές με αυτήν βιβλιοθήκες.

3.2.1 Βιβλιοθήκες

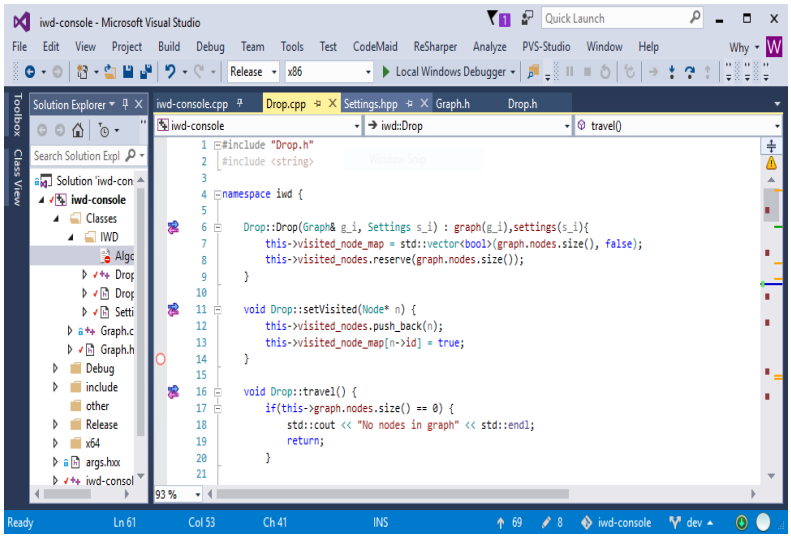
Χρησιμοποιήθηκε τρεις βοηθητικές βιβλιοθήκες, η `args`[2], `JSON`[3] και `fmt`[4]. Σκοπός της `args` είναι η ευκολότερη επεξεργασία των παραμέτρων λειτουργίας του προγράμματος. Συγκεκριμένα παρέχει εντολές για την ευκολότερη προσπέλαση των δεδομένων της `argv` και την εξαγωγή των τιμών της. Η `JSON` χρησιμοποιήθηκε για την προσπέλαση των αρχείων ρυθμίσεων, που περιγράφονται στην ενότητα 4.3. Η `fmt` βοηθάει στη ευκολότερη μορφοποίηση των κειμένων που εμφανίζει η εφαρμογή κατά την εκτέλεσή της. Και οι τρεις είναι βιβλιοθήκες ανοικτού κώδικα και είναι συμβατές με την έκδοση C++11.

3.2.2 Visual Studio 2015/2017 Community Edition

Το Visual Studio είναι ένα Integrated Development Environment (IDE) της Microsoft για περιβάλλον Windows. Επιλέχθηκε διότι παρέχει όλα τα απαραίτητα εργαλεία για ανάπτυξη εφαρμογών C++ σε περιβάλλον Windows όπως compiler, debugger, profiler κλπ, καθώς και πλήρη υποστήριξη για το .NET Framework. Ο compiler που παρέχεται από την Microsoft υποστηρίζει την έκδοση C++ που έχομε

[2] `https://github.com/Taywee/args`
[3] `https://github.com/nlohmann/json`
[4] `http://fmtlib.net`

επιλέξει και ο ενσωματωμένος profiler ήταν ιδιαίτερα χρήσιμος κατά βελτιστοποίηση της εφαρμογής. Η "Community Edition" είναι δωρεάν[5].



Σχήμα 5: Παράθυρο VS2015 σε περιβάλλον Windows 10

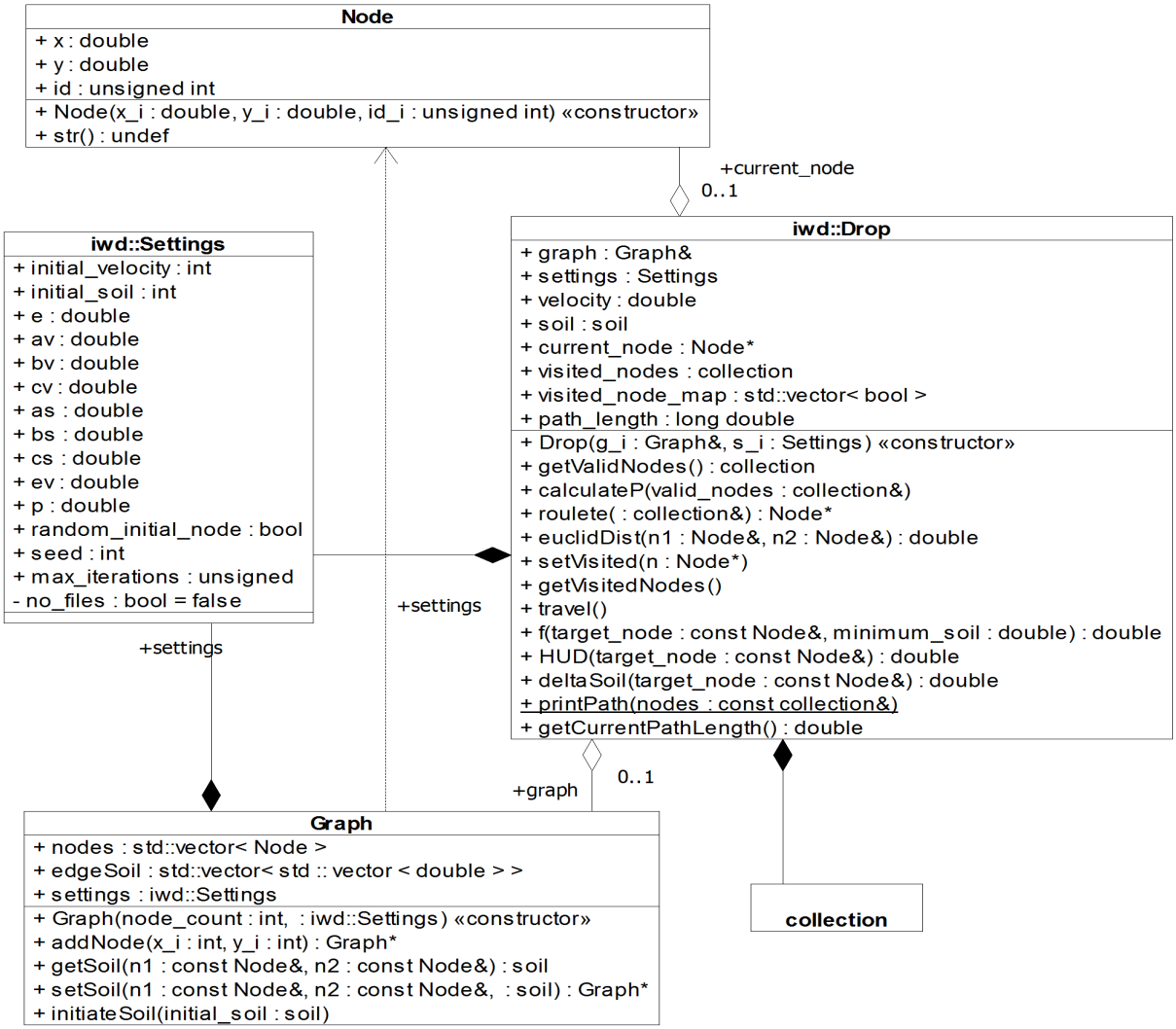
3.3 Σχεδιασμός

Το βασικό μέρος του κώδικα αποτελείται από τέσσερις κλάσεις

- **`Node`** : Αναπαριστά τους κόμβους ενός γράφου. Τα βασικά χαρακτηριστικά της είναι οι `int x,y` που ορίζουν την θέση του κόμβου στο επίπεδο και το `id` που είναι ένας μοναδικός για κάθε κόμβο αριθμός. Το `int id` το ορίζουμε εμείς αυτόματα κατά την διάρκεια δημιουργίας ενός κόμβου. Η μέθοδος `str()` είναι μία βοηθητική μέγεθος που τυπώνει τα χαρακτηριστικά του κόμβου.
- **`Graph`** : Η κλάση που αναπαριστά έναν γράφο. Περιέχει αριθμό από `nodes`. Η μέθοδος `initiateSoil()` αρχικοποιεί το *soil* σε όλους τους κόμβους. Να σημειωθεί ότι δεν υπάρχει υλοποίηση των ακμών ενός γράφου. Επειδή ο γράφος μας είναι πλήρης, δεν θα υπήρχε νόημα κρατάμε συσχετίσεις ανάμεσα σε κόμβους. Αντ' αυτού έχουμε έναν πίνακα `edgeSoil()` που κρατάει το *soil* που περιέχει κάθε ακμή. Οι μέθοδοι `setSoil()` και `getSoil()` ορίζουν και επιστρέφουν αντίστοιχα το *soil* κάθε ακμής.
- **`iwd::Settings`** : Η κλάση που περιέχει όλες τις ρυθμίσεις και επιλογές μας.
- **`iwd::Drop`** : Η βασική κλάση της εφαρμογής μας. Περιέχει την αναπαράσταση μίας σταγόνας ώς δεδομένα καθώς και τις βασικές λειτουργίες του αλγορίθμου IWD. Εκτός από το *soil* και *velocity*, κάθε σταγόνα έχει πρόσβαση στην μεταβλητή του γράφου και στις επιλογές που έχουμε ορίσει. Εσωτερικά έχει έναν δείκτη `current_node` προς έναν κόμβο του γράφου που διατηρεί την θέση της σταγόνας κατά την διάρκεια του ταξιδιού της. Η λίστα V_N αντιπροσωπεύεται από τις μεταβλητές `visited_nodes` και `visited_node_map`. Το κυρίως μέρος του αλγορίθμου βρίσκεται στην μέθοδο `travel()`. Μετά την εκτέλεση της `travel()`, η `visited_nodes` θα περιέχει μία πιθανή λύση του

[5] `https://www.visualstudio.com/vs/community/`

αλγορίθμου και η `path_length` θα είναι ίση με την απόσταση της διαδρομής που ακολούθησε η σταγόνα.



Σχήμα 6: Unified Modeling Language (UML) διάγραμμα των κλάσεων και των συσχετίσεων τους

3.3.1 `iwd::Drop`

Η κλάση αυτή αντιπροσωπεύει την δομή μίας Ευφυούς Σταγόνας.

`typedef std::vector<Node*> collection` ορίζουμε την λέξη-κλειδί ως μία λίστα `vector<>` έτσι ώστε αν αργότερα χρειαστεί να την αλλάξουμε με κάποιον άλλο container να είναι ευκολότερο. Κάθε `collection` συγκρατεί μία λίστα από δείκτες προς τους κόμβους ενός γράφου.

`Graph& graph` κάθε σταγόνα έχει μία αναφορά (στο εξής reference) στον γράφο. Στην ουσία είναι ένας γράφος ο οποίος αλλάζει όσο οι γενιές του αλγορίθμου αυξάνονται. Με αυτόν τον τρόπο μειώνουμε την μνήμη RAM που καταναλώνει το πρόγραμμα, γίνεται γρηγορότερο λόγω αποφυγής αντιγραφής δεδομένων στην μνήμη και βοηθάει στην υλοποίηση του αλγορίθμου.

`Settings settings` αντίθετα κάθε σταγόνα έχει τις δικές της ρυθμίσεις. Αυτό είναι για να μπορούμε να πειραματιστούμε ευκολότερα με διάφορες παραμέτρους εκτέλεσης

`std::vector<bool> visited_node_map` με το να έχουμε έναν πίνακα από `bool` τιμές επιταχύνουμε την αναζήτηση των κόμβων που είναι διαθέσιμοι.

`auto calculateP(collection& valid_nodes)` & **`Node* roulete(collection&)`** οι δύο αυτές μέθοδοι συνδυάζονται για να μας δώσουν τον επόμενο κόμβο κατά την διάρκεια του ταξιδιού της σταγόνας. Η `calculateP()` υπολογίζει την πιθανότητα επιλογής κάθε κόμβου από τους διαθέσιμους σύμφωνα με τον τύπο (1) και η `roulete()` μας επιστρέφει έναν δείκτη προς τον κόμβο.

`void travel()` η βασική μέθοδος κάθε σταγόνας. Με το πέρας της εκτέλεσής της, η σταγόνα θα πρέπει να είναι σε μία κατάσταση στην οποία η `visited_nodes` περιέχει την διαδρομή που ακολούθησε η σταγόνα, η `path_length` είναι το μήκος αυτής της διαδρομής και οι `soil` και `velocity` είναι οι τελικές τιμές του $soil^{IWD}$ και $velocity^{IWD}$ αντίστοιχα.

3.3.2 `iwd::Settings`

Ο σκοπός της κλάσης αυτής είναι να διατηρεί όλες μας τις επιλογές. Αυτές είναι όλες οι μεταβλητές που απαιτούνται για την λειτουργία του αλγορίθμου, καθώς και κάποιες επιλογές που επηρεάζουν τον τρόπο λειτουργίας του αλγορίθμου χωρίς να είναι μέρος των προδιαγραφών του. Στον Πίνακα 2 υπάρχει η αντιστοιχία των μεταβλητών της κλάσης με αυτές που ορίζει ο αλγόριθμος.

Πίνακας 2: Πίνακας με όλες τις ρυθμίσεις και την αντιστοίχηση τους με τον Πίνακα 1

Μεταβλητή	Αρχική Τιμή	Αντιστοίχηση
`int initial_soil`	1000	InitSoil
`int initial_velocity`	200	InitVel
`double av`	1	a_v
`double bv`	0.01	b_v
`double cv`	1	c_v
`double as`	1	a_s
`double bs`	0.01	b_s
`double cs`	1	c_s
`double p`	0.9	ρ_n/ρ_{IWD}
`double e`	0.0001	ϵ

Οι υπόλοιπες μεταβλητές είναι οι εξής :

`bool random_initial_node` (default : `true`) Εάν η επιλογή αυτή είναι `true` κάθε σταγόνα ξεκινάει το ταξίδι της από τυχαίο κόμβο. Αλλιώς ξεκινάει από τον κόμβο με εσωτερικό δείκτη `id = 0`.

`int seed` Μεταβλητή που χρησιμοποιείται στην αρχικοποίηση των μηχανών παραγωγής τυχαίων αριθμόν (Random number generators - RNG) όπως `mt19937` ή `rand()`. Αν δεν οριστεί από την χρήστη, αρχικοποιείται σε μια τυχαία τιμή. Ορίζοντάς την μπορούμε να αναπαράγουμε τα αποτελέσματα της ενότητας 6

`int max_iterations` (default : `100`) Ο μέγιστος αριθμός γενεών $iter_{max}$

`std::string out_filename` (default : `"output"`) Το όνομα των αρχείων που δημιουργούνται με την εκτέλεση του προγράμματος.

`bool no_files` (default : `false`) Ελέγχει αν θα δημιουργηθούν ή όχι αρχεία μετά την εκτέλεση.

Η κλάση δεν διαθέτει μεθόδους. Η πρόσβαση στα μέλη της γίνεται με άμεση ανάθεση/ανάγνωση. Αυτό έγινε για λόγους ευκολίας και απλότητας κατά την ανάπτυξη.

3.4 Υλοποίηση iwd-console

Χρησιμοποιώντας τις κλάσεις που περιγράφονται στην ενότητα 3.3 δημιουργήθηκε μια υλοποίηση που τρέχει σε περιβάλλον κονσόλας. Χρησιμοποιεί εντολές της έκδοσης C++11 και έτσι μπορεί να τρέξει σε οποιοδήποτε λειτουργικό έχει διαθέσιμο compiler αυτής της έκδοσης. Οι βιβλιοθήκες (ενότητα 3.2.1) που χρησιμοποιούνται είναι και αυτές cross-platform.

Η `iwd-console` δέχεται ορίσματα από την γραμμή εντολών και τα επεξεργάζεται με τις εντολές της βιβλιοθήκης `args`, η οποία παρέχει και τρόπο για να ορίσουμε ποιες παράμετροι είναι απαραίτητες καθώς και τις αποδεκτές τιμές τους.Στον ενότητα 4.1 περιγράφονται αναλυτικά τα ορίσματα που δέχεται η εφαρμογή.

Να σημειώσουμε ότι, αν και παρέχονται εντολές για αυτόματη δημιουργία μηνυμάτων βοήθειας, η βιβλιοθήκη δεν λειτουργεί σωστά με Unicode χαρακτήρες, με αποτέλεσμα να μην μπορεί να εμφανίσει μηνύματα στα ελληνικά. Για αυτόν τον λόγο δημιουργήθηκε ένα macro που δέχεται ένα κείμενο και κάνει τις απαραίτητες ενέργειες ώστε να εμφανιστεί σωστά στην γραμμή εντολών.

Λίστα 1: Macro εμφάνισης ελληνικών χαρακτήρων

```
1     #define GREEK(x) {                \
2     _setmode(_fileno(stdout), _O_U8TEXT); \
3     std::wcout << x << std::endl;     \
4     _setmode(_fileno(stdout), _O_TEXT);   \
5     }
6
```

Μετά την προσπέλαση των ορισμάτων θα πρέπει να είμαστε σε θέση να εκτελέσουμε τα βήματα που περιγράφει ο αλγόριθμος. Χρησιμοποιούνται μόνο containers της Standard Template Library (STL) όπως `std::vector`. Για να ελαττώσουμε την κατανάλωση μνήμης RAM, ο γράφος δημιουργείτε μόνο μία φορά. Για κάθε αναφορά σε κάποιο στοιχείο του χρησιμοποιούνται μόνο δείκτες ή αναφορές(references). Το ίδιο συμβαίνει και όταν περνάει σαν όρισμα σε κάποια συνάρτηση.

Όταν θέλουμε να μεταφέρουμε μια συλλογή από κόμβους, όπως για παράδειγμα όταν θέλουμε να κρατήσουμε μια λίστα με κόμβους που έχει επισκεφθεί κάθε

σταγόνα, χρησιμοποιούμε τον τύπο `collection`. Ο τύπος `collection` δεν είναι τίποτα άλλο παρά ένας vector με δείκτες προς στοιχεία του γράφου. Για την μορφοποίηση των κειμένων που εμφανίζονται τόσο πριν όσο και μετά το πέρας της εκτέλεσης χρησιμοποιείται η βιβλιοθήκη `fmt`. Λεπτομέρειες για τον τρόπο λειτουργίας και χρήσης της `iwd-console` υπάρχουν στην ενότητα 4.

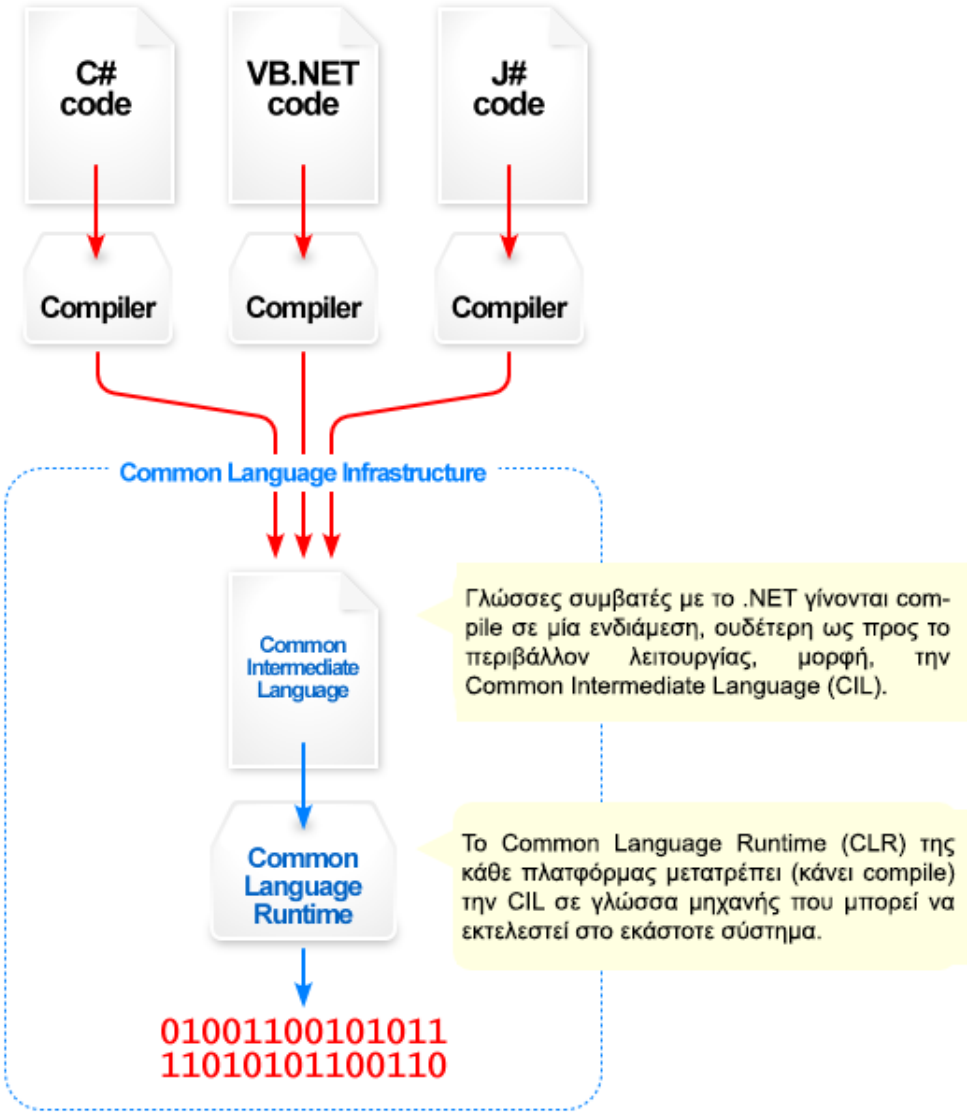
3.5 Υλοποίηση iwd-gui

Για την υλοποίηση της `iwd-gui` έγινε χρήση της C++/CLI(C++ modified for Common Language Infrastructure) σε συνδυασμό με κλάσεις του Microsoft .NET Framework (φόρμες).Γίνεται χρήση των κλάσεων που περιγράφονται στην ενότητα 3.3 μιας και οι δύο γλώσσες είναι συμβατές και ο σχεδιασμός του γραφικού περιβάλλοντος έγινε με τον Windows Forms Designer. Όπως και για την `iwd-console`, χρησιμοποιήθηκε το το IDE Visual Studio, αν και για την `iwd-gui` επιλέχθηκε η έκδοση 2017 λόγο την βελτίωσης πολλών εργαλείων. Να σημειωθεί ότι οι βασικές κλάσεις παραμένουν απαράλλαχτες.

3.5.1 .NET Framework και Common Language Infrastructure

Το Commom Language Infrastructure (CLI) είναι ένα standard που έχει δημιουργήσει η Microsoft. Περιγράφει έναν τρόπο για compilation κώδικα από διάφορες γλώσσες προγραμματισμού (C#,Visual Basic κλπ) σε μία ενδιάμεση μορφή, την Common Intermediate Language (CIL). Με αυτόν τον τρόπο μπορεί μια υλοποίηση του Commom Language Runtime (CLR) να μετατρέψει τον κώδικα αυτό σε γλώσσα μηχανής που μπορεί να εκτελεστεί στο εκάστοτε περιβάλλον. Σε αυτό το κομμάτι μοιάζει πολύ με την Java Virtual Machine (JVM). Στο σχήμα 7 υπάρχει ένα διάγραμμα με την γενική αρχιτεκτονική του CLI.

Η C++/CLI είναι ένα υπερσύνολο της C++ με κάποιες προσθήκες της Microsoft για ευκολότερη ανάπτυξη στο managed περιβάλλον της CLI. Μπορεί να θεωρηθεί μια ξεχωριστή γλώσσα μιας και έχει δικό της standard[20] και πολλές προσθήκες όσων αφορά τις λέξεις-κλειδιά. Ένα μεγάλο πλεονέκτημά της και ο κύριος λόγος που επιλέχθηκε για την υλοποίησή μας είναι η πλήρης συμβατότητά της με την C++. Με αυτόν τον τρόπο μπορούμε να χρησιμοποιήσουμε τις κλάσεις που δημιουργήσαμε χωρίς καμία τροποποίηση. Η μόνη διαφορά με την `iwd-console` είναι ότι έπρεπε να προσέξουμε κάποιες λεπτομέρειες στον τρόπο χρήσης τους μιας και το περιβάλλον εκτέλεσης του CLR χρησιμοποιεί garbage collector. Περισσότερες λεπτομέρειες καθώς και δυσκολίες που συναντήθηκαν υπάρχουν στις ενότητες 3.5.3 και 3.5.4.



Σχήμα 7: Γενική επισκόπηση λειτουργίας του CLI

Το .NET Framework είναι ένας συνδυασμός από βιβλιοθήκες και εργαλεία που τρέχουν στο περιβάλλον του CLR για το λειτουργικό Microsoft Windows. Περιέχει, μεταξύ άλλων, ένα σύνολο από βιβλιοθήκες και εργαλεία για την ευκολότερη δημιουργία παραθυρικών εφαρμογών. Τα απαραίτητα αρχεία και βιβλιοθήκες είναι ενσωματωμένα στον Visual Studio και έτσι είναι πολύ εύκολη η χρήση τους. Επίσης το Visual Studio περιέχει το εργαλείο Windows Forms Designer που δίνει την δυνατότητα σχεδιασμού φορμών με drag-and-drop των επιθυμητών αντικειμένων (κουμπιά,ετικέτες κλπ).

3.5.2 Βοηθητικές κλάσεις

Για την ευκολότερη ανάπτυξη την εφαρμογής δημιουργήθηκαν κάποιες βοηθητικές κλάσεις, οι `iwdgui::Settings` και `IwdWorkerState`, οι οποίες βρίσκονται στο αρχείο `IwdHelpClasses.h`. Και οι δύο είναι managed κλάσεις που κληρονομούν απο το `System::Object`.

Τα μέλη της `iwdgui::Settings` είναι ίδια με αυτά της `iwd::Settings`. Η μόνη διαφορά είναι οτι το μέλος `out_filename` είναι τύπου `String^` σε αντίθεση με το

`std::string` της `iwd::Settings`. Αυτό έγινε επειδή υπήρχαν προβλήματα κατά την μεταφορά απο managed κώδικα σε unmanaged κώδικα. Επίσης η `iwdgui::Settings` χρησιμοποιείται στην κλάση `IwdWorkerState`, και δεν επιτρέπεται μια managed κλάση να περιέχει unmanaged αντικείμενα.

Οι μόνες μέθοδοι που έχει η κλάση `iwdgui::Settings` είναι ο contstructor της που δέχεται σαν όρισμα ένα αντικείμενο τύπου `iwd::Settings` και αντιγράφει τα δεδομένα της και η `iwd::Settings getOriginal()` που επιστρέφει ένα αντικείμενο τύπου `iwd::Settings` με τα δεδομένα της. Το μόνο σημείο που χρησιμοποιείτε η κλάση αυτή είναι κατά την δημιουργία του thread που θα εκτελέσει τον αλγόριθμο.

Η κλάση `IwdWorkerState` περιέχει δεδομένα που είναι απαραίτητα για την λειτουργία του αλγορίθμου, καθώς και για την κατάσταση λειτουργίας του. Χρησιμοποιείται κατά την δημιουργία του thread που τρέχει τον αλγόριθμο καθώς και για την επικοινωνία του με το thread που περιέχει το user interface. Οι συναρτήσεις της `BackgroundWorker` μπορούν να δεχτούν ένα αντικείμενο τύπου `System::Object ^`, το οποίο περιέχει τα δεδομένα που χρειάζεται να λειτουργήσει. Το ίδιο αντικείμενο μπορεί να μεταφέρει δεδομένα πίσω στο αρχικό thread. Με αυτόν τον τρόπο γίνεται δυνατή η άμεση αναφορά της προόδου του αλγορίθμου κάνοντας κλήση της συνάρτησης `BackgroundWorker::ReportProgress()`.

3.5.3 Λεπτομέρειες

Ο σχεδιασμός του γραφικού περιβάλλοντος έγινε με τον Windows Forms Designer και αποτελείτε από δύο φόρμες, τις `Main` και `ProgressForm`, οι οποίες περιέχουν τις κλάσεις `Main` και `ProgressForm` αντίστοιχα. Και οι δύο αυτές κλάσεις κληρονομούν την κλάση `System::Windows::Forms::Form` που είναι η βασική φόρμα της .NET Framework Class Library.

Η φόρμα `Main` περιέχει το βασικό παράθυρο λειτουργίας. Η φόρμα `ProgressForm` περιέχει τα στοιχεία εκτέλεσης κάθε σταγόνας,δείχνει την κατάσταση της και το ποσοστό προόδου του ταξιδιού της. Δίνει επίσης την δυνατότητα ακύρωσης της εκτέλεσής της.

Μέσα από την `Main.h` μπορούμε να επιλέξουμε το αρχείο γράφου και να ορίσουμε τις απαραίτητες παραμέτρους για την λειτουργία του αλγορίθμου. Στον Πίνακα 3 υπάρχουν όλα τα στοιχεία (components) που περιέχει η φόρμα με μια σύντομη περιγραφή της λειτουργίας τους.

Πίνακας 3: Στοιχεία φόρμας `Main`

Όνομα	Τύπος	Περιγραφή
`groupBox1`	GroupBox	Ομαδοποίηση στοιχείων
`groupBox2`	GroupBox	Ομαδοποίηση στοιχείων
`groupBox3`	GroupBox	Ομαδοποίηση στοιχείων
`var_no_files`	CheckBox	Δημιουργία αρχείων εξόδου (`no_files`)
`var_random_initial_node`	CheckBox	Επιλογή τυχαίου αρχικού κόμβου (`random_initial_node`)
`graphOpenDialog`	OpenFileDialog	Διάλογος επιλογής αρχείου γράφου
`var_max_iterations`	TextBox	Ορισμός μέγιστου αριθμού γενεών (`max_iterations`)
`var_output_filename`	TextBox	Ορισμός ονόματος αρχείων εξόδου (`out_filename`)
`var_initial_velocity`	TextBox	Ορισμός παραμέτρου $InitVel$ (`initial_velocity`)
`var_initial_soil`	TextBox	Ορισμός παραμέτρου $InitSoil$ (`initial_soil`)
`var_av`	TextBox	Ορισμός παραμέτρου a_v (`av`)
`var_as`	TextBox	Ορισμός παραμέτρου a_s (`as`)
`var_bv`	TextBox	Ορισμός παραμέτρου b_v (`bv`)
`var_bs`	TextBox	Ορισμός παραμέτρου b_s (`bs`)
`var_cv`	TextBox	Ορισμός παραμέτρου c_v (`cv`)
`var_cs`	TextBox	Ορισμός παραμέτρου c_s (`cs`)
`var_e`	TextBox	Ορισμός παραμέτρου ϵ (`e`)
`var_p`	TextBox	Ορισμός παραμέτρου ρ (`p`)
`graphFileInput`	TextBox	Πεδίο εισαγωγής διαδρομής προς αρχείο γράφου
`btn_reset`	Button	Κουμπί επαναφοράς ρυθμίσεων στις αρχικές
`btn_open_file`	Button	Κουμπί εκτέλεσης `graphOpenDialog`
`btn_run`	Button	Κουμπί έναρξης ταξιδιού σταγόνας
`label1` έως `label14`	Label	Στατικό κείμενο

Πέρα από τα components που δημιουργούμε με τον WFD, η `Main` περιέχει και την λίστα `List<ProgressForm^>^ windows`, το οποίο είναι μία λίστα με όλα τα παράθυρα `ProgressForm` που δημιουργούμε. Αυτό συμβαίνει επειδή μπορούμε να έχουμε πολλά προβλήματα να τρέχουν ταυτόχρονα και χρειαζόμαστε έναν τρόπο να κρατάμε όλα τα παράθυρα που δημιουργούνται.

Τα παρακάτω κουμπιά περιέχουν τον κώδικα των βασικών λειτουργιών της φόρμας.

`btn_open_file` Πατώντας το εκτελείτε το event `btn_open_file_Click`.

`btn_run` Πατώντας το εκτελείτε το event `btn_run_Click`.

`btn_reset` Πατώντας το εκτελείτε το event `btn_reset_Click`.

Οι περισσότερες από αυτές τις λειτουργίες βρίσκονται σε events που εκτελούνται. Στην παρακάτω λίστα υπάρχουν όλες οι μέθοδοι διαθέσιμοι στην κλάση `Main`.

`void resetSettings()` Επαναφέρει όλα τα πεδία `TextBox` (εκτός του `graphFileInput`) της φόρμας στις αρχικές τιμές τους.

`iwd::Settings settingsFromInput()` Διαβάζει τις τιμές που έχουμε εισάγει στα `TextBox` και δημιουργεί ένα αντικείμενο `iwd::Settings` από αυτές.

`System::Void btn_open_file_Click` (event) Εκτελεί την μέθοδο `ShowDialog()` του `graphOpenDialog`.

`System::Void graphOpenDialog_FileOk` (event) Εκτελείτε όταν επιλέξουμε αρχείο από τον διάλογο του `graphOpenDialog`. Τοποθετεί το μονοπάτι προς το αρχείο που επιλέξαμε σαν τιμή στον `TextBox graphFileInput`.

`System::Void btn_run_Click` (event) Αρχικά ελέγχονται οι τιμές που έχουν δοθεί στην βασική φόρμα. Αν είναι όλες ορθές και μπορεί να αναγνωσθεί το αρχείο γράφου, δημιουργείται ένα παράθυρο `ProgressForm`, το τοποθετεί στην λίστα `windows` και ξεκινάει η εκτέλεση του αλγορίθμου.

`System::Void btn_reset_Click` (event) Εκτελεί την μέθοδο `resetSettings()` της `Main`.

`System::Void graphFileInput_DoubleClick` (event) Εκτελείτε όταν κάνουμε διπλό-click στο `TextBox graphFileInput`. Εάν είναι κενό, ανοίγει ο διάλογος επιλογής γράφου(`graphOpenDialog`).

Η `ProgressForm` περιέχει τα στοιχεία απαραίτητα για την έναρξη μιας εργασίας καθώς και την αναφορά της προόδου προς τον χρήστη. Παρέχει την δυνατότητα πρόωρου τερματισμού της διαδικασίας εκτέλεσης του αλγορίθμου. Η λειτουργία του αλγορίθμου καθαυτή εκτελείτε σε δικό της νήμα (thread) για την αποφυγή του 'παγώματος' του user interface. Νήμα αυτό δημιουργείτε από την κλάση `ProgressForm` μέσω του `iwdWorker`. Στον Πίνακα 4 υπάρχουν όλα τα components της φόρμας. Εκτός από αυτά, η κλάση έχει και τα παρακάτω μέλη :

`bool closeWhenFinished` Γίνεται `true` όταν θέλουμε να κλείσει το τωρινό παράθυρο προόδου με το πέρας της εκτέλεσης.

`DateTime opStart` Κρατάει την χρονική στιγμή έναρξης της εκτέλεσης ώστε να μπορούμε να υπολογίσουμε την διάρκειά της.

και τις παρακάτω μεθόδους :

`System::Void iwdWorker_DoWork` (event) Το βασικότερο event του `iwdWorker`. Όλος ο κώδικα που περιέχεται εδώ εκτελείτε σε ξεχωριστό thread. Είναι υπεύθυνο για την εκτέλεση των βημάτων του αλγορίθμου, την ενημέρωση του παραθύρου για την πρόοδο της εκτέλεσης και της διακοπής της σε περίπτωση ακύρωσης από τον χρήστη.

`System::Void iwdWorker_ProgressChanged` (event) Ανανεώνει τα αντικείμενα της φόρμας προόδου σύμφωνα με τα δεδομένα της κατάστασης εκτέλεσης του αλγορίθμου.

`System::Void iwdWorker_RunWorkerCompleted` (event) Εκτελείτε όταν ολοκληρωθεί ή ακυρωθεί η εκτέλεση του κώδικα του `iwdWorker_DoWork`.

`System::Void btnCancel_Click` (event) Καλεί την `cancelWork()`.

`System::Void ProgressForm_FormClosing` (event) Εκτελείτε όταν ο χρήστης κλείσει το παράθυρο της φόρμας προόδου. Σε περίπτωση που ο `iwdWorker` εκτελείτε ακόμα στέλνουμε σήμα να σταματήσει, ακυρώνουμε το κλείσιμο της φόρμας, και θέτει την τιμή του `closeWhenFinished` σε `true`.

`void cancelWork` Σταματά την εκτέλεση του αλγορίθμου καλώντας την `CancelAsync` του `iwdWorker`.

`System::Void iwdTimer_Tick` (event) Εκτελείτε ανά τακτά χρονικά διαστήματα που ορίζουμε εμείς (στην περίπτωσή μας ανά 100ms). Ανανεώνει το κείμενο της `lblTimer` ώστε να δείχνει την διάρκεια εκτέλεσης του αλγορίθμου.

Το component `iwdWorker` είναι υπεύθυνο για το μεγαλύτερο μέρος των εργασιών που επιτελεί η κλάση `ProgressForm`. Σκοπός του είναι να δημιουργήσει ένα νέο thread στο οποίο θα εκτελεστεί ο αλγόριθμος. Αυτό συμβαίνει επειδή το user interface 'τρέχει' στο αρχικό thread. Εάν εκτελούσαμε τον κώδικα του αλγορίθμου στο ίδιο thread, θα σταματούσε η ανανέωση του UI μέχρις ότου να ολοκληρωθεί. Αυτό θα σήμαινε ότι θα ήταν αδύνατο να χρησιμοποιηθούν τα παράθυρα, θα ήταν αδύνατη η ενημέρωση των πεδίων μας (όπως αυτά που δείχνουν την κατάσταση εκτέλεσης του αλγορίθμου) και θα ήταν αδύνατη και η ακύρωση της εκτέλεσης του αλγορίθμου. Το περιβάλλον χρήσης θα είχε 'παγώσει' μέχρι να τελειώσουν όλα τα βήματα του αλγορίθμου. Δημιουργώντας ένα ξεχωριστό thread για κάθε εργασία αποφεύγουμε όλα αυτά τα προβλήματα. Σαν επιπλέον πλεονέκτημα μας δίνεται η δυνατότητα να εκτελούμε τον αλγόριθμο για πολλά προβλήματα ταυτόχρονα, με μόνο περιορισμό τους διαθέσιμους πόρους του μηχανήματος.

3.5.4 Δυσκολίες

Η χρήση διαφορετικού thread για την εκτέλεση του αλγορίθμου έλυσε πολλά προβλήματα εισάγοντας όμως ένα επίπεδο πολυπλοκότητας. Το πρώτο πρόβλημα ήταν ότι δεν επιτρέπεται η επεξεργασία αντικειμένων ενός thread από το άλλο. Έτσι ήταν αδύνατο το thread του αλγορίθμου να 'διαβάσει' τις παραμέτρους που έχει δώσει ο χρήστης στην αρχική φόρμα ή να ενημερωθεί το παράθυρο που βλέπει ο χρήστης για την πρόοδο του αλγορίθμου. Ο μόνος τρόπος επικοινωνίας είναι ο constructor και η μέθοδος `ReportProgress` του `BackgroundWorker` που δέχονται σαν όρισμα ένα αντικείμενο τύπου `Object^`. Για αυτόν τον λόγο δημιουργήθηκε και η βοηθητική κλάση `IwdWorkerState` που περιγράφεται στην ενότητα 3.5.2.

Το δεύτερο πρόβλημα ήταν η μεταφορά δεδομένων από το ένα thread στο άλλο, ειδικά εφόσον υπάρχει ασυμφωνία στον τρόπο διαχείρισης μνήμης μεταξύ managed και unmanaged κώδικα. Έτσι unmanaged αντικείμενα και δεδομένα που χρησιμοποιούνται στο thread του `BackgroundWorker` πρέπει να δημιουργούνται εκεί. Για αυτόν τον λόγο γίνεται ανάγνωση του αρχείου γράφου μέσα στο ίδιο το thread.

Ένα άλλο πρόβλημα ήταν ότι κλείνοντας το παράθυρο προόδου που περιέχει τον `BackgroundWorker` δεν σταματάει αυτόματα η εκτέλεσή του μιας και από την στιγμή που ξεκινάει μια εργασία, είναι σαν ξεχωριστή οντότητα. Έτσι υπήρχε περίπτωση το παράθυρο προόδου να είναι κλειστό ενώ η εργασία συνεχίζει να εκτελείτε, καταναλώνοντας πόρους εν αγνοία του χρήστη. Για αυτόν τον λόγο, όταν ο χρήστης επιλέξει να κλείσει το παράθυρο προόδου, αντί να κλείσει το παράθυρο θέτουμε το μέλος `closeWhenFinished` σε `true` και στέλνουμε σήμα να ακυρωθεί η εκτέλεση του αλγορίθμου. Στην συνέχεια, μόλις εκτελεστεί το event `iwdWorker_RunWorkerCompleted` ελέγχουμε την κατάσταση της μεταβλητής `closeWhenFinished`. Εάν είναι `true` σημαίνει ότι ο χρήστης έχει ζητήσει το κλείσιμο του παραθύρου και το κλείνουμε.

Πίνακας 4: Στοιχεία φόρμας `ProgressForm`

Όνομα	Τύπος	Περιγραφή
`groupBox1`	GroupBox	Ομαδοποίηση στοιχείων
`groupBox2`	GroupBox	Ομαδοποίηση στοιχείων
`iwdWorker`	BackgroundWorker	Εκτέλεση αλγορίθμου σε ξεχωριστό νήμα (thread)
`dropPath`	RichTextBox	Δείχνει την τωρινή βέλτιστη διαδρομή
`lblStatus`	Label	Δείχνει την κατάσταση εκτέλεσης του αλγορίθμου
`lbl_initial_velocity`	Label	Τιμή παραμέτρου $InitVel$ (`initial_velocity`)
`lbl_initial_soil`	Label	Τιμή παραμέτρου $InitSoil$ (`initial_soil`)
`lbl_av`	Label	Τιμή παραμέτρου a_v (`av`)
`lbl_as`	Label	Τιμή παραμέτρου a_s (`as`)
`lbl_bv`	Label	Τιμή παραμέτρου b_v (`bv`)
`lbl_bs`	Label	Τιμή παραμέτρου b_s (`bs`)
`lbl_cv`	Label	Τιμή παραμέτρου c_v (`cv`)
`lbl_cs`	Label	Τιμή παραμέτρου c_s (`cs`)
`lbl_p`	Label	Τιμή παραμέτρου ϵ (`e`)
`lbl_e`	Label	Τιμή παραμέτρου ρ (`p`)
`lbl_no_files`	Label	Δείχνει αν θα δημιουργηθούν αρχεία μετά την εκτέλεση
`lbl_output_filename`	Label	Δείχνει το όνομα του αρχείου εξόδου
`lbl_random_initial_node`	Label	Δείχνει αν έχει οριστεί επιλογή τυχαίου αρχικού κόμβου
`lblNodes`	Label	Δείχνει τον αριθμό των κόμβων του γράφου
`lblPathLength`	Label	Δείχνει το μήκος της τωρινής βέλτιστης διαδρομής
`lblTimer`	Label	Δείχνεις την διάρκεια εκτέλεσης του αλγορίθμου
`iwdTimer`	Timer	Ανανεώνει την διάρκεια εκτέλεσης
`iwdStatusStrip`	StatusStrip	Μπάρα με στοιχεία εκτέλεσης
`iwdWorkProgressBar`	ToolStripProgressBar	Μπάρα ένδειξης ποσοστού ολοκλήρωσης
`lblCurrentIteration`	ToolStripStatusLabel	Δείχνει σε ποια γενιά/επανάληψη βρίσκεται ο αλγόριθμος
`btnCancel`	Button	Κουμπί ακύρωσης εκτέλεσης αλγορίθμου
`label1` έως `label4`	Label	Στατικό κείμενο

4 Χρήση iwd-console

Για την εκτέλεση το προγράμματος πρέπει να του περάσουμε σαν παράμετρο ένα αρχείο με την περιγραφή του γράφου που θα χρησιμοποιηθεί στο πρόβλημά μας. Μπορούμε να χρησιμοποιήσουμε οποιοδήποτε Command Line Interface (CLI). Για να εμφανιστούν όλες οι διαθέσιμες επιλογές μπορούμε να καλέσουμε την εφαρμογή με την παράμετρο `-h` . Οι περισσότερες παράμετροι είναι άμεσα συνδεδεμένοι με τις ιδιότητες της κλάσης `iwd::settings`. Για παράδειγμα θέτοντας `-i 1000` ορίζουμε `max_iterations = 1000` . Οι ρυθμίσεις που δεν μπορούν να οριστούν άμεσα από την γραμμή εντολών μπορούν να οριστούν στο αρχείο ρυθμίσεων, που περιγράφεται στην ενότητα 4.3

4.1 Command Line Arguments

Η μόνη απαραίτητη παράμετρος είναι η `-g` που ορίζει το μονοπάτι προς το αρχείο που περιέχει τα στοιχεία του γράφου.

`-h, --help` Εμφανίζει την βοήθεια καθώς και τις διαθέσιμες παραμέτρους

`-i[iterations]` Ορίζει τον μέγιστο αριθμό γενεών

`-g[filepath], --graph=[filepath]` (**απαραίτητη**) Ορίζει το αρχείο που περιέχει τους κόμβους του γράφου. Δέχεται σαν όρισμα το όνομα του αρχείο με το μονοπάτι του. Υπάρχει παράδειγμα της μορφής των αρχείων που μπορούν να προσπελαστούν στο τέλος του παρόντος εγγράφου. Εάν δεν είναι δυνατή η πρόσβαση στο αρχείο, η λειτουργία της εφαρμογής σταματά και εμφανίζεται μήνυμα σφάλματος.

`-s[number], --seed=[number]` Η τιμή του `iwd::settings::seed` που περιγράφεται στην ενότητα 3.3.2. Δέχεται θετικούς ακέραιους αριθμούς. Για οποιαδήποτε άλλη τιμή, το αποτέλεσμα είναι ακαθόριστο

`--settings=[filename]` Διαδρομή προς το αρχείο με τις παραμέτρους λειτουργίας του αλγορίθμου και της εφαρμογής. Το αρχείο πρέπει να είναι ένα ορθώς μορφοποιημένο JSON αρχείο. Περισσότερες πληροφορίες για την μορφή του αρχείου και την λειτουργία του υπάρχουν στην ενότητα 3.3.2

`-o[filename], --output=[filename]` (default : "output") Η εφαρμογή δημιουργεί κάποια αρχεία κατά την εκτέλεσή της. Με την παράμετρο αυτή ορίζουμε το όνομα των αρχείων που θα δημιουργηθούν.

`-r, --random-initial-node` Εάν οριστεί, κάθε φορά που ταξιδεύει μία σταγόνα ξεκινάει από τυχαίο αρχικό κόμβο.

`--po, --print-settings` Εάν οριστεί, εμφανίζει τις ρυθμίσεις στην γραμμή εντολών πριν την εκτέλεση του αλγορίθμου.

`--no-files` Εάν οριστεί, δεν δημιουργούνται αρχεία εξόδου (σε περίπτωση που απλά θέλετε να εκτελέσετε τον αλγόριθμο). Σε περίπτωση που οριστεί, η παράμετρος `o-` αγνοείται.

Ένα παράδειγμα χρήσης των παραμέτρων είναι `iwd-console-Win32.exe --print-settings -g eil51.tsp -o results -i 500 --settings=settings_file.json`.

4.2 Έξοδος προγράμματος

Δύο είναι οι βασικές έξοδοι της εφαρμογής

- Βασική έξοδος του προγράμματος
- Τα αρχεία που δημιουργούνται κατά την εκτέλεση του προγράμματος

Σαν βασική έξοδο εννοείται το standard output και standard error (`cout` και `cerr` αντίστοιχα). Εκεί εμφανίζονται τα όποια μηνύματα λάθους καθώς και, στο τέλος της λειτουργίας του προγράμματος, κάποια στοιχεία για την εκτέλεση του αλγορίθμου όπως η τελική διαδρομή και το μήκος της. Τα μηνύματα εξόδου εξαρτώνται και από το τι έχουμε περάσει σαν ορίσματα στην γραμμή εντολών.

Τα αρχεία εξόδου περιέχουν δεδομένα που χρησιμοποιούνται για την δημιουργία των διαγραμμάτων και εικόνων που υπάρχουν στην ενότητα 6. Το αρχείο με κατάληξη `.pdat` έχει δεδομένα για το διάγραμμα των βέλτιστων λύσεων που βρίσκει ο αλγόριθμος ανά γενεά. Το αρχείο αυτό μπορεί να χρησιμοποιηθεί από εργαλεία όπως το `gnuplot`[6].

Τα αρχεία με κατάληξη `.ndat` περιέχουν τα στοιχεία της βέλτιστης, κατά τον αλγόριθμό μας, διαδρομής του γράφου. Μπορούν να χρησιμοποιηθούν από εργαλεία όπως το `graphviz`[7].

4.3 Αρχείο Ρυθμίσεων

Υπάρχει η δυνατότητα οι ρυθμίσεις της εφαρμογής να οριστούν σε ένα αρχείο τύπου Javascript Object Notation (JSON). Όλες οι παράμετροι που μπορούν να οριστούν από την γραμμή εντολών και οι λοιπές ρυθμίσεις που υπάρχουν στην κλάση `iwd::Settings` μπορούν να αρχικοποιηθούν στο αρχείο αυτό. Εξαιρούνται οι παράμετροι `-g` και `--settings`.

Λίστα 2: Παράδειγμα αρχείου ρυθμίσεων

```
1     {
2     "av": 0.1,
3     "bv" : 0.01,
4     "cv": 1,
5     "as": 0.1,
6     "bs": 0.01,
7     "cs": 1,
8     "e": 0.0001,
9     "p": 0.9,
10    "max_iterations" : 300,
11    "out_filename":"test",
12    "initial_velocity" : 10000,
13    "initial_soil" : 200
14    }
15
```

Στο παράδειγμα της Λίστας 2 βλέπουμε ότι μπορούμε να ορίσουμε όλες τις επιλογές της κλάσης `iwd::Settings`, ακόμα και αυτές που δεν μπορούμε να αρχικοποιήσουμε από την γραμμή εντολών. Να σημειώσουμε ότι τα ορίσματα που

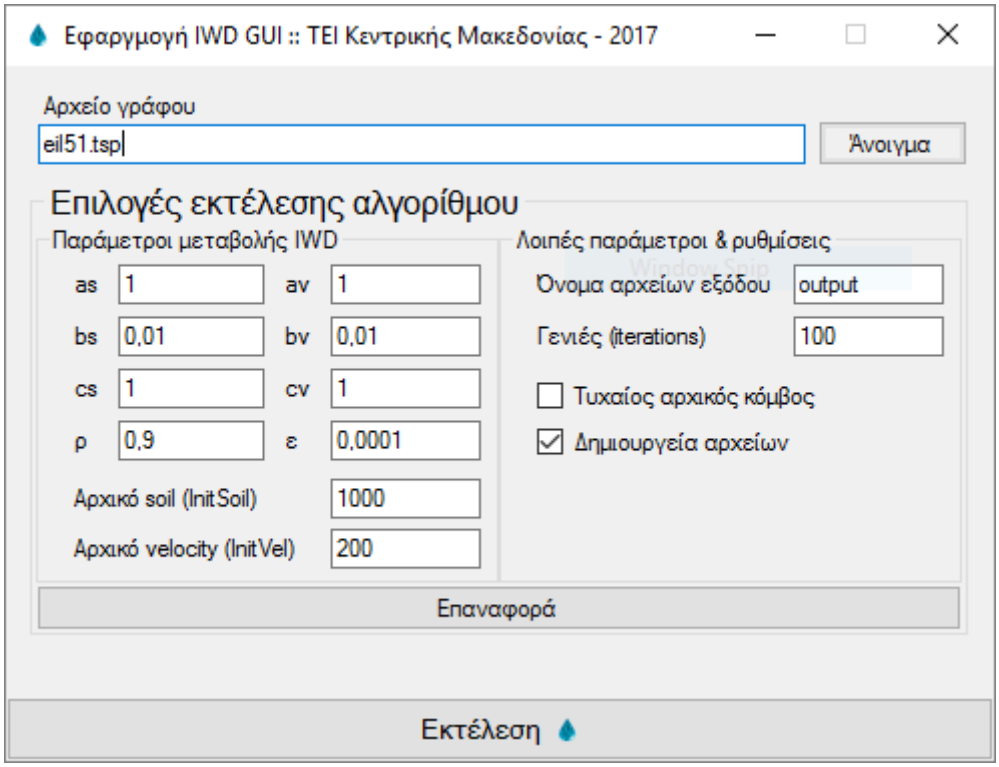
[6] `http://gnuplot.sourceforge.net/`
[7] `http://graphviz.org/`

θέτουμε από την γραμμή εντολών έχουν μεγαλύτερη βαρύτητα από το αρχείο ρυθμίσεων. Για παράδειγμα αν ορίσουμε `"max_iterations" : 300` στο αρχείο, αλλά περάσουμε σαν όρισμα `-i 1200` , η τελική τιμή του `max_iterations` θα είναι 1200.

5 Χρήση iwd-gui

5.1 Βασικό παράθυρο

Στο σχήμα 8 φαίνεται το βασικό παράθυρο της εφαρμογής. Από εδώ μπορούμε να επιλέξουμε το αρχείο με τα δεδομένα του γράφου καθώς και να ορίσουμε τις παραμέτρους της εφαρμογής.



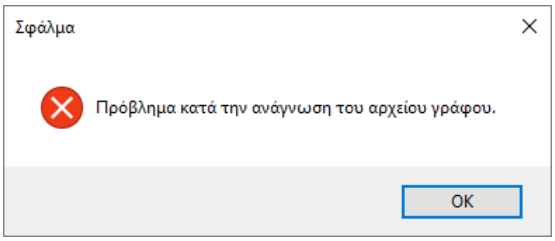
Σχήμα 8: Βασικό παράθυρο εφαρμογής iwd-gui

Πατώντας το κουμπί "Άνοιγμα" θα ανοίξει ένα παράθυρο από το οποίο μπορούμε να επιλέξουμε το αρχείο γράφου. Όπως και για την `iwd-console`, αυτό το βήμα είναι απαραίτητο μιας και ο αλγόριθμος δεν μπορεί να λειτουργήσει χωρίς αυτό. Επίσης μπορούμε να εισάγουμε χειροκίνητα το μονοπάτι προς το αρχείο στο πεδίο κειμένου που είναι στα αριστερά του κουμπιού. Σε περίπτωση που το αρχείο δεν υπάρχει ή υπάρξει κάποιο πρόβλημα κατά την ανάγνωσή του εμφανίζεται ένα μήνυμα λάθους στην οθόνη (σχήμα 9α΄).

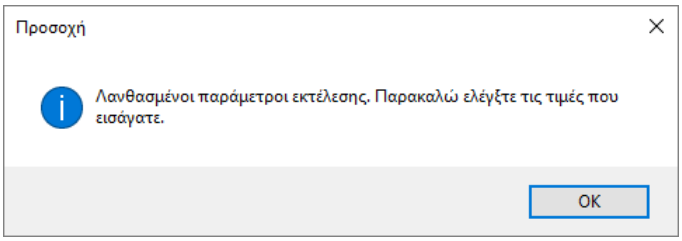
Στην συνέχεια μπορούμε να ορίσουμε τις παραμέτρους εκτέλεσης. Οι επιλογές είναι ίδιες με αυτές που περιγράφονται στην ενότητα 3.3.2, με την διαφορά δεν μπορούμε να ορίσουμε την παράμετρο **seed** λόγο τεχνικών περιορισμών της υλοποίησης.

Το κουμπί "Επαναφορά" επαναφέρει τις προκαθορισμένες επιλογές.

Σχήμα 9: Μηνύματα λάθους εφαρμογής iwd-gui



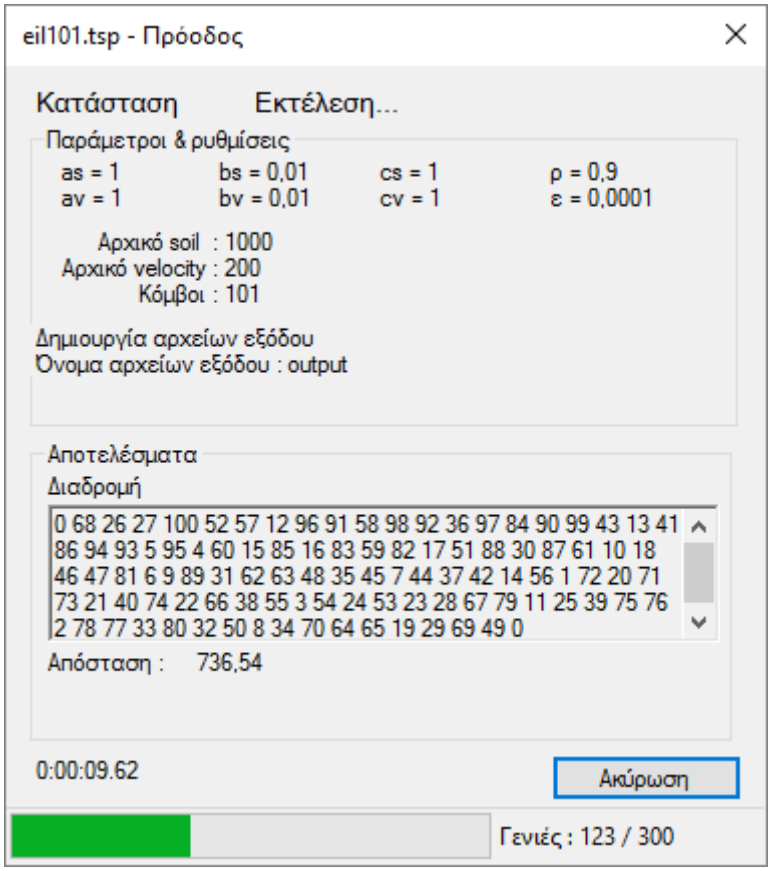
(α΄) Πρόβλημα με το αρχείο γράφου



(β΄) Μήνυμα σε περίπτωση λανθασμένων παραμέτρων εφαρμογής

5.2 Παράθυρο προόδου εκτέλεσης

Εάν δεν υπάρξει κάποιο πρόβλημα με τις επιλογές που έχουμε ορίσει ξεκινάει η εκτέλεση του αλγορίθμου και δημιουργείται ένα νέο παράθυρο (σχήμα 10) που φαίνεται η πρόοδός του.



Σχήμα 10: Ενδεικτικό παράθυρο προόδου iwd-gui για το πρόβλημα eil101.tsp

Σε αυτό εμφανίζονται οι παράμετροι και επιλογές που έχουμε ορίσει (τμήμα "Παράμετροι & Ρυθμίσεις") καθώς και τα αποτελέσματά μας, τα οποία ανανεώνονται κατά την διάρκεια της εκτέλεσης. Στο τμήμα "Αποτελέσματα" υπάρχει η βέλτιστη διαδρομή που έχει βρει ο αλγόριθμός μας καθώς και το μήκος της. Όταν ολοκληρωθεί η εκτέλεση του αλγορίθμου, τα πεδία αυτά θα περιέχουν την βέλτιστη λύση. Με την έναρξη της εκτέλεσης ξεκινάει και ένα χρονόμετρο που καταγράφει την διάρκειά της.

Στο κάτω μέρος του παραθύρου υπάρχει η μπάρα προόδου με το ποσοστό ολοκλήρωσης του αλγορίθμου καθώς και την τωρινή γενιά. Τέλος υπάρχει το κουμπί "**Ακύρωση**" που σταματάει την εκτέλεση του αλγορίθμου (το παράθυρο παραμένει ανοικτός μέχρι να το κλείσουμε χειροκίνητα). Να σημειωθεί ότι ανάλογα με τις δυνατότητες του υπολογιστή και τον φόρτο εργασίας που έχει ο επεξεργαστής, η διακοπή της λειτουργίας μπορεί να μην γίνει άμεσα.

6 Πειράματα

Για τα πειράματα μας χρησιμοποιήσαμε τα TSP προβλήματα που διατίθενται από το Universität Heidelberg [21]. Τα παραδείγματα αυτά είναι εκτενώς μελετημένα και μαζί με πολλά από αυτά παρέχεται και η βέλτιστη λύση. Με αυτόν τον τρόπο είναι ευκολότερη η μελέτη της αποδοτικότητας του αλγορίθμου. Εκτελέστηκαν με την εφαρμογή `iwd-console` που επιτρέπει την ευκολότερη αυτοματοποίηση καθώς επίσης και τον ορισμό της παραμέτρου "**seed**", που κάνει εφικτή την ακριβή αναπαραγωγή των αποτελεσμάτων μας.

Πίνακας 5: Παράμετροι εκτέλεσης πειραμάτων

(α΄) Πείραμα 1

Εκτέλεση	seed	γενιές
r1	1001	300
r2	1002	300
r3	1003	300
r4	1004	300
r5	1005	300
r6	1006	300
r7	1007	300
r8	1008	300
r9	1009	300
r10	10010	300

(β΄) Πείραμα 2

seed	γενιές	μεταβλητές
1000	[300,500,1000,10000]	$a_v, a_s = 0.01$
1000	[300,500,1000,10000]	$a_v, a_s = 0.1$
1000	[300,500,1000,10000]	$a_v, a_s = 10$
1000	[300,500,1000,10000]	$b_v, b_s = 0.1$
1000	[300,500,1000,10000]	$b_v, b_s = 1$
1000	[300,500,1000,10000]	$b_v, b_s = 10$
1000	[300,500,1000,10000]	$c_v, c_s = 0.1$
1000	[300,500,1000,10000]	$c_v, c_s = 10$
1000	[300,500,1000,10000]	$c_v, c_s = 100$

Τα χαρακτηριστικά του μηχανήματος που έγιναν τα πειράματα βρίσκονται στον Πίνακα 6

Πίνακας 6: Χαρακτηριστικά Υπολογιστή

CPU	Intel®Core2Quad Q9300 2.5GHz
RAM	8GiB
GPU	ATI Radeon HD4870
Λειτουργικό Σύστημα	Windows 10 Pro Edition x64

Για το 1ο πείραμα επιλέχθηκαν τα προβλήματα `eil51`,`eil101` και `a280` , με 51, 101 και 280 κόμβους αντίστοιχα. Εκτελέσαμε τον αλγόριθμο δέκα φορές για κάθε πρόβλημα. Ορίσαμε σαν μέγιστο τις 300 γενιές και αλλάξαμε το seed για κάθε εκτέλεση για να πάρουμε κάποιες γενικές μετρήσεις της αποδοτικότητας του αλγόριθμου και της υλοποίησής μας.

Για το 2ο πείραμα εστιάσαμε στο `eil51` για να δοκιμάσουμε την βελτίωση των αποτελεσμάτων με περισσότερες γενιές καθώς και την επιρροή των παραμέτρων μεταβολής σταγόνας (Πίνακας 1) στην ποιότητα των λύσεων. Συγκεκριμένα επιλέξαμε τις 300,500,1000 και 10000 γενιές, αλλάξαμε τις μεταβλητές a_v, a_s,b_v, b_s, c_v, c_s και συγκρίναμε τα αποτελέσματά μας.

7 Αποτελέσματα και Συμπεράσματα

7.1 Πείραμα 1

Ως απόσταση(^{optimal}) ορίζουμε την βέλτιστη δυνατή διαδρομή ενός προβλήματος, όπως μας δίνεται από τα δεδομένα του TSPLIB[21], και ως απόσταση(^{found}) ορίζουμε την βέλτιστη απόσταση που μας δίνει ο αλγόριθμός μας. Υπολογίζουμε την μέση βέλτιστη απόσταση που έχουμε μετά από 10 εκτελέσεις, καθώς και την τυπική απόκλιση (αριθμός και ποσοστό) τους. Ως ποιότητα λύσης q ορίζουμε το ποσοστό που πλησιάζει η λύση που βρήκαμε την βέλτιστη δυνατή και σαν λάθος E έχουμε ουσιαστικά το $100 - q$. Υπολογίζουμε και το ποσοστό βελτίωσης που έχουμε για τις πρώτες 100 γενιές του αλγορίθμου αλλά και για τις γενιές μετά την 100η ώστε να δούμε πόσες γενιές κατά μέσο όρο χρειάζεται ο αλγόριθμος για να πλησιάσει ή να βρει την βέλτιστη λύση και κατά πόσο η αύξηση των γενεών βελτιώνει τα αποτελέσματά μας. Τέλος κρατάμε τον μέσο χρόνο εκτέλεσης κάθε δοκιμής για να μπορέσουμε να συγκρίνουμε την επιβάρυνση που έχει η αύξηση των πόλεων στην απόδοση της εφαρμογής.

Στα σχήματα 11 έως 13 (σελίδα 37) βλέπουμε μια γενική επισκόπηση της απόδοσης του αλγορίθμου. Τα αποτελέσματα υπάρχουν αναλυτικότερα στον Πίνακα 9 (σελίδα 37).

Πίνακας 7: Στατιστικά στοιχεία εκτέλεσης πειραμάτων για eil51, eil101 και a280

Διαδρομή	eil51	eil101	a280
Βέλτιστη απόσταση(^{optimal})[8]	426	629	-[9]
Βέλτιστη απόσταση(^{found})[10]	463.54	695.70	3364.12
Μέση απόσταση	477.53	742.19	3594.27
Τυπική απόκλιση	12.21	27.7	91.45
Τυπική απόκλιση(%)	2.52	3.64	2.54
q βέλτιστης απόστασης(^{found})(%)	91.9	90.41	-
q (μέσης απόστασης) (%)	89.2	84.74	-
Λάθος Ε βέλτιστης απόστασης(^{found})(%)	8.8	10.6	-
Λάθος Ε μέσης απόστασης (%)	12.1	18	-
% βελτίωσης (πρώτες 100 γενιές)	126.78	147.15	278.51
% βελτίωσης (μετά τις 100 γενιές)	4.74	6.05	1.89
Μέσος χρόνος εκτέλεσης (sec)	1.7	12.7	225.7

Θέτοντας ως ποιότητα λύσης

$$q(tour) = \frac{solution^{optimal}(tour)}{solution^{found}(tour)} * 100 \tag{9}$$

και ποσοστό λάθους

$$E = \frac{solution^{found} - solution^{optimal}}{solution^{optimal}} * 100 \tag{10}$$

Να σημειώσουμε ότι λόγο του ότι δεν έχουμε την βέλτιστη λύση για το πρόβλημα a280, κάποια από τα αποτελέσματά του παραμένουν κενά.

7.2 Πείραμα 2

Στον Πίνακα 8 έχουμε τις διαδρομές που βρίσκει ο αλγόριθμός μας για διαφορετικές τιμές των μεταβλητών ανανέωσης γράφου. Στην πρώτη σειρά έχουμε τα αποτελέσματα με τις αρχικές τιμές, οι οποίες προτείνονται για το πρόβλημα TSP[13] ενώ στις υπόλοιπες έχουμε αλλάξει τις τιμές για a_v/a_s, b_v/b_s και c_v/c_s. Μια γραφική επισκόπηση των αποτελεσμάτων υπάρχει στο κεφάλαιο 10.

Ως 'Αρχικές' έχουμε τις προεπιλεγμένες τιμές των μεταβλητών, οι οποίες φαίνονται στους Πίνακες 1 και 2.

Βλέπουμε ο αλγόριθμος, παρότι βελτιώνει πάντοτε την λύση που βρίσκει, έχει πολύ έντονες διαφορές ανάλογα με τον συνδυασμό τιμών που θα επιλέξουμε. Η χειρότερη λύση που βρίσκουμε είναι 1170.56 μετά από 3949 γενιές, με $b_v, b_s = 1$. Αυτή είναι παραπάνω από διπλάσια της καλύτερης που βρίσκουμε (447.2 μετά από 9147 γενιές) κάτι που καθιστά έντονη την σημασία επιλογής σωστών παραμέτρων για κάθε πρόβλημα.

Μπορούμε επίσης να δούμε ότι οι μεταβλητές αυτές δεν επηρεάζουν μόνο την ποιότητα της λύσης αλλά και την αποδοτικότητα του αλγορίθμου. Για κάποιους

[8] Η βέλτιστη δυνατή διαδρομή

[9] Δεν έχουμε στοιχεία για την βέλτιστη απόσταση αυτού του προβλήματος

[10] Η βέλτιστη απόσταση που έχουμε βρει με τον αλγόριθμό μας

συνδυασμούς τιμών ο αλγόριθμος 'κολλάει' σε κάποια τοπικά βέλτιστα και δεν βελτιώνει την λύση του ακόμα και μετά από χιλιάδες γενιές. Συγκεκριμένα για $a_v, a_s = 0.01$ δεν υπάρχει βελτίωση για τις τελευταίες 8978 γενιές.

Πίνακας 8: Πίνακας με τα αποτελέσματα βέλτιστων διαδρομών για διαφορετικές τιμές των μεταβλητών ανανέωσης γράφου

Ρυθμίσεις	Βέλτιστη διαδρομή (επαναλήψεις)			
	300	500	1000	10000
Αρχικές	499.83 (140)	479.93 (327)	465.61 (764)	**447.2 (9147)**
$a_v, a_s = 0.01$	674.26 (27)	674.26 (27)	674.26 (27)	674.26 (27)
$a_v, a_s = 0.1$	477.93 (144)	470.18 (353)	467.72 (807)	467.18 (1022)
$a_v, a_s = 10$	531.66 (290)	495.45 (334)	481.61 (600)	448.49 (5201)
$b_v, b_s = 0.1$	1033.1 (114)	1027.41 (449)	1011.24 (841)	918.61 (7361)
$b_v, b_s = 1$	1295.97 (248)	1243.66 (337)	1243.66 (337)	1170.56 (3949)
$b_v, b_s = 10$	1261.2 (145)	1236.74 (383)	1236.74(383)	1211.8 (1809)
$c_v, c_s = 0.1$	638.8 (176)	638.8 (176)	638.8 (176)	621.32 (3712)
$c_v, c_s = 10$	470.4 (238)	468.20 (414)	457 (537)	451.52 (5499)
$c_v, c_s = 100$	524.56 (208)	524.56 (208)	524.56 (208)	491.78 (4986)

7.3 Συμπεράσματα

Μπορούμε να παρατηρήσουμε ότι ο αλγόριθμός μας τείνει προς μια βέλτιστη λύση σχετικά γρήγορα. Από τις βέλτιστες διαδρομές που βρίσκουμε (Πίνακας 7) βλέπουμε ότι ο αλγόριθμος πλησιάζει αρκετά κοντά στην βέλτιστη λύση και είναι εμφανές από τα διαγράμματα (Σχήματα 11 έως 13) ότι κινείται ταχύτατα προς αυτήν, ενώ η απόδοση αυτή είναι σταθερή μεταξύ των λύσεων. Δυσκολεύεται όμως να κάνει μικρές βελτιώσεις στην ποιότητα της λύσης. Οι μεγαλύτερες αλλαγές εμφανίζονται στις πρώτες 100 γενιές σε όλες τις δοκιμές που κάναμε ενώ το ποσοστό βελτίωσης μειώνεται αισθητά στις υπόλοιπες. Επίσης παρατηρούμε ότι η ποιότητα (τύπος 9) των βέλτιστων λύσεων που βρήκαμε είναι άνω του 90%, αν και η ποιότητα της μέσης απόστασης είναι χαμηλότερη.

Αν και βλέπουμε ότι πλησιάζουμε αρκετά κοντά στην βέλτιστη δυνατή λύση ακόμα και σε σχετικά μεγάλα προβλήματα, η πολυπλοκότητα του αλγορίθμου αυξάνεται ραγδαία για κάθε νέο κόμβο που προσθέτουμε, κάτι που μπορούμε να δούμε και από τον χρόνο εκτέλεσης. Ενώ για 51 πόλεις χρειαζόμαστε λιγότερο από 2s, για τον διπλάσιο αριθμό πόλεων(101) χρειαζόμαστε παραπάνω από εξαπλάσιο χρόνο, ενώ για 280 πόλεις ο μέσος χρόνος εκτέλεσης είναι σχεδόν 130 φορές μεγαλύτερος.

Επίσης ο αλγόριθμος, με την μορφή που έχει τώρα, απαιτεί πλήρη γράφο για να λειτουργήσει. Αυτό μειώνει κατά πολύ τον αριθμό των προβλημάτων που μπορεί να λύσει. Για παράδειγμα, αν υποθέσουμε ότι κάθε κόμβος είναι μια πόλη, θα έπρεπε κάθε μία από αυτές να συνδέεται με έναν δρόμο διπλής κατεύθυνσης προς όλες τις άλλες. Αυτό είναι πρακτικά αδύνατο για οποιοδήποτε σημαντικό αριθμό πόλεων.

Τέλος, η αποτελεσματικότητα του αλγορίθμου εξαρτάται άμεσα από τις παραμέτρους μεταβολής (πίνακας 1), οι οποίες είναι κυρίως εμπειρικές/ευρετικές.

Παράμετροι που οδηγούν σε βέλτιστη λύση για ένα πρόβλημα μπορεί να είναι άχρηστες σε ένα άλλο. Θα πρέπει να προσαρμόζονται ανάλογα με τις απαιτήσεις μας, και ο μόνος τρόπος να γίνει αυτό είναι μέσω δοκιμών. Αν και υπάρχουν κάποιες ενδεικτικές τιμές και για άλλα προβλήματα ([16]), αποτελεί γενικότερο πρόβλημα της χρησιμότητας τους αλγορίθμου.

Πιθανές λύσεις σε αυτά τα προβλήματα αναφέρονται στην ενότητα 8.1

8 Πιθανές βελτιώσεις

8.1 Βελτιώσεις στον αλγόριθμο

8.1.1 Multi-core/Multi-trhead αλγόριθμος

Ένας πιθανός τρόπος βελτίωσης του χρόνου εκτέλεσης του αλγορίθμου θα ήταν να γίνουν μετατροπές στον τρόπο λειτουργίας του έτσι ώστε κάθε σταγόνα να μπορεί να λειτουργεί ανεξάρτητα από τις υπόλοιπες. Ως έχει, κάθε σταγόνα "αλλοιώνει" το περιβάλλον με το ταξίδι της. Έτσι κάθε επόμενη σταγόνα βασίζεται στα αποτελέσματα της προηγούμενης, κάτι που οδηγεί σε σειριακή εκτέλεση των βημάτων.

Με τις κατάλληλες τροποποιήσεις, θα μπορούσαμε να παραλληλοποιήσουμε την διαδικασία του ταξιδιού. Έτσι, για κάθε γενιά, όλες οι σταγόνες θα ταξίδευαν ταυτόχρονα. Από άποψη υλοποίησης, αυτό θα σήμαινε ότι κάθε σταγόνα θα έτρεχε σε δικό της νήμα/πυρήνα, με αποτέλεσμα την αποδοτικότερη χρήση των διαθέσιμων πόρων.

8.1.2 Διεύρυνση δυνατοτήτων

Όπως ανεφέρθηκε ο αλγόριθμος απαιτεί πλήρη γράφο για να λειτουργήσει σωστά. Αυτό επειδή κάθε σταγόνα μπορεί να περάσει μόνο μία φορά από κάθε ακμή. Έτσι, εάν έχουμε ένα κόμβο που δεν συνδέεται με όλους τους υπόλοιπους, υπάρχει περίπτωση να πάει μια σταγόνα εκεί και να μην μπορεί να συνεχίσει το ταξίδι της.

Για να ξεπεράσουμε αυτό το ζήτημα θα μπορούσαμε να βάλουμε επιπλέον ελέγχους κάθε φορά που μια σταγόνα επιλέγει το επόμενο βήμα της. Ένας τέτοιος έλεγχος θα ήταν ότι αν μία σταγόνα έφτανε σε αδιέξοδο, θα τερμάτιζε το ταξίδι της και θα σηματοδοτούσε την διαδρομή που ακολούθησε ως "μη βέλτιστη" (αυξάνοντας δραματικά το soil) έτσι ώστε οι επόμενες σταγόνες να μην την προτιμήσουν. Άλλη εναλλακτική θα ήταν οι σταγόνες να μπορούν να οπισθοδρομήσουν σε περίπτωση που είναι σε αδιέξοδο έτσι μέχρις ότου να μπορούν να κινηθούν ξανά.

8.1.3 Παράμετροι μεταβολής

Μία πιθανή λύση για το ζήτημα των παραμέτρων μεταβολής προτείνουν οι Mohammed M. Msallam και Mohammad Hamdan στο άρθρο με τίτλο "Improved intelligent water drops algorithm using adaptive schema"[22]. Ο αλγόριθμος Adaptive Intelligent Water Drop (A-IWD) που προτείνουν ελέγχει αν έχουν περάσει αρκετές γενεές χωρίς βελτίωση της λύσης. Στην περίπτωση αυτή αλλάζουν οι τιμές των

$InitVel$ και $InitSoil$ και αρχικοποιούνται με βάση μια φόρμουλας που προτείνουν. Σύμφωνα με τα αποτελέσματά τους, αυτό οδηγεί σε βελτίωση της ποιότητας των λύσεων σε σχέση με τον αρχικό αλγόριθμο IWD.

Αν και η λύση τους αναφέρεται μόνο στις μεταβλητές $initialsoil$ και $initialvelocity$, η ίδια μέθοδος θα μπορούσε να εφαρμοστεί και σε όλες τις υπόλοιπες παραμέτρους μεταβολής.

Μία άλλη πιθανή λύση θα ήταν οι μεταβλητές αυτές να μπορούν να επιλέγονται από ένα εύρος τιμών και, ανάλογα με το ποιες από αυτές οδηγούν σε βελτίωση της λύσης, μετά από μερικές γενιές να επιλέγονται οι καλύτερες.

8.2 Βελτιώσεις στον κώδικα

8.2.1 Σχεδιασμός εφαρμογής

Αν και έγινε προσπάθεια στο να σχεδιαστεί σωστά η εφαρμογή, το τελικό αποτέλεσμα θα μπορούσε να είναι καλύτερο. Ως έχει, η οποιαδήποτε προσπάθεια αλλαγής του κώδικα είναι δύσκολη μιας και απαιτεί δραστικές αλλαγές στον πηγαίο κώδικα.

Για παράδειγμα για να γίνει υλοποίηση του A-IWD[22] θα πρέπει να ξαναγραφτεί μεγάλο μέρος του κώδικα. Το ίδιο θα γινόταν και αν, λόγου χάρη, θέλαμε να αλλάξουμε τον τρόπο μέτρησης της απόστασης δύο κόμβων. Θα έπρεπε να ξανά γραφτεί όλος ο αλγόριθμος. Εναλλακτικά, θα μπορούσαμε να έχουμε ένα interface που ορίζει την μέθοδο `distance` και κάθε φορά που θέλουμε να εισάγουμε μια νέα μέθοδο υπολογισμού απόστασης, το μόνο που θα χρειαζόταν ήταν να αλλάξει η υλοποίησή μας.

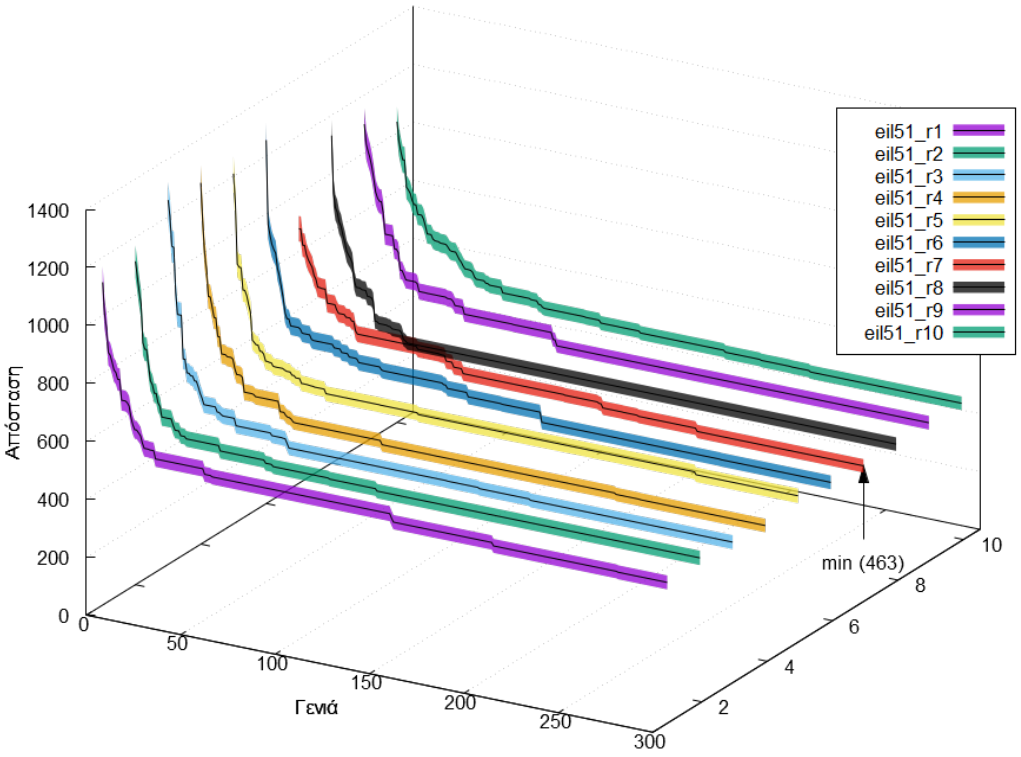
Άλλες πιθανές αλλαγές θα ήταν ο διαχωρισμός μίας σταγόνας από τον αλγόριθμο που τις χρησιμοποιεί. Τα χαρακτηριστικά κάθε σταγόνας παραμένουν σταθερά σε κάθε πρόβλημα, το μόνο που αλλάζει είναι οι λειτουργίες που επιτελεί κάθε αλγόριθμος για να βρει την λύση του. Έτσι θα ήταν ποιο εύκολος ο πειραματισμός σε νέες μεθόδους και βελτιώσεις στον αρχικό IWD αλγόριθμο.

9 Πίνακες και δεδομένα

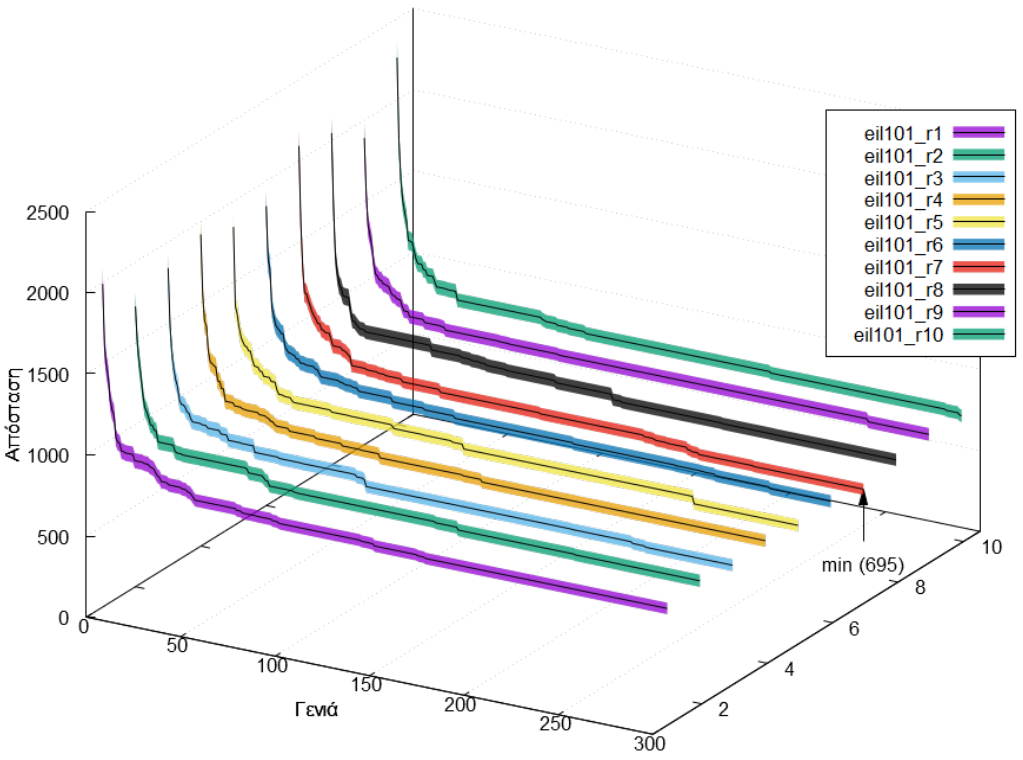
Πίνακας 9: Αποτελέσματα πειράματος 1

Εκτέλεση (seed)	Βέλτιστη λύση			Ποιότητα λύσης			Χρόνος εκτέλεσης (ms)		
	eil51	eil101	a280	eil51	eil101	a280	eil51	eil101	a280
r1 (1001)	481.353	710.902	3364.12	88.50	88.47	-	1675	12320	204821
r2 (1002)	495.928	756.230	3653.54	85.89	83.17	-	1681	11844	209732
r3 (1003)	480.306	727.181	3595.35	88.69	86.49	-	1676	11636	208805
r4 (1004)	467.499	753.409	3642.33	91.12	83.48	-	1664	11762	208868
r5 (1005)	500.067	722.793	3639.78	85.18	87.02	-	1687	11782	206426
r6 (1006)	476.531	746.190	3652.38	89.39	84.29	-	1689	11721	203214
r7 (1007)	463.546	695.706	3566.41	91.90	90.41	-	1680	11669	198237
r8 (1008)	468.476	751.788	3689.34	90.93	83.66	-	1670	11747	199352
r9 (1009)	472.217	783.445	3567.27	90.21	80.28	-	1667	11843	196998
r10 (10010)	469.448	774.289	3572.21	90.74	81.23	-	1688	11542	197386

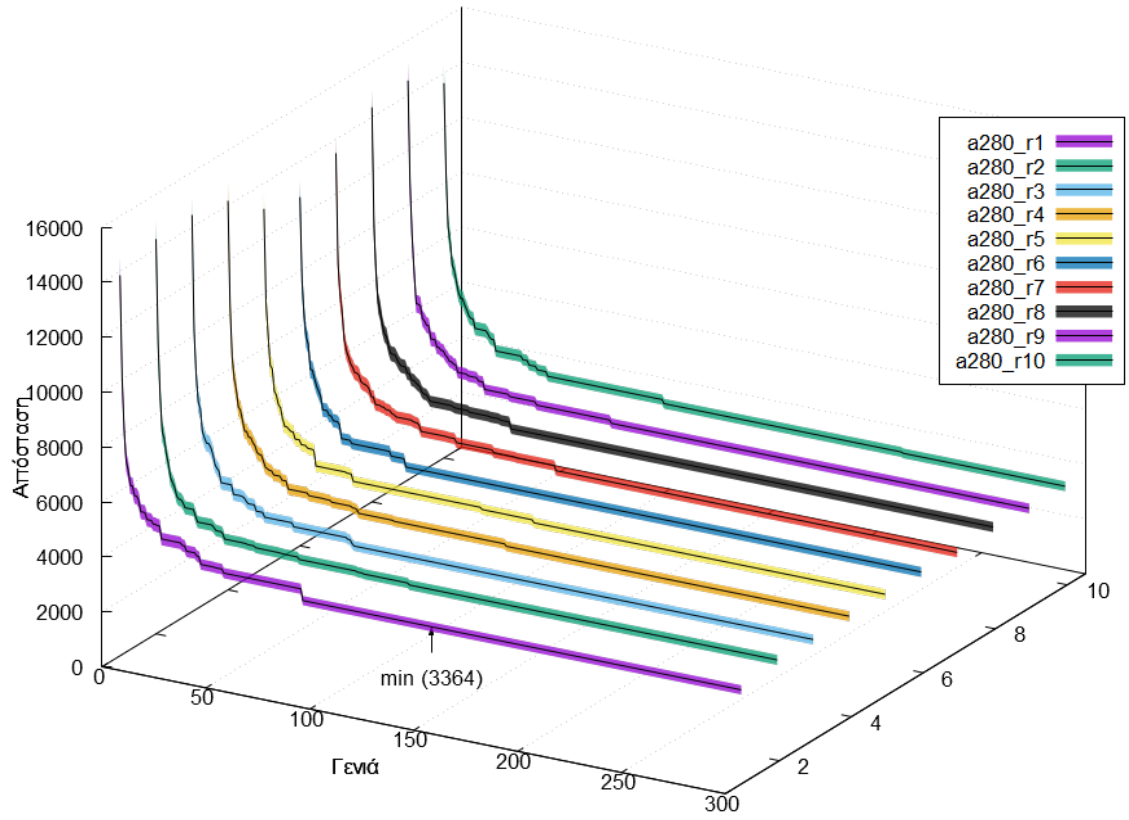
10 Διαγράμματα και σχήματα



Σχήμα 11: Αποτελέσματα 10 εκτελέσεων για eil51

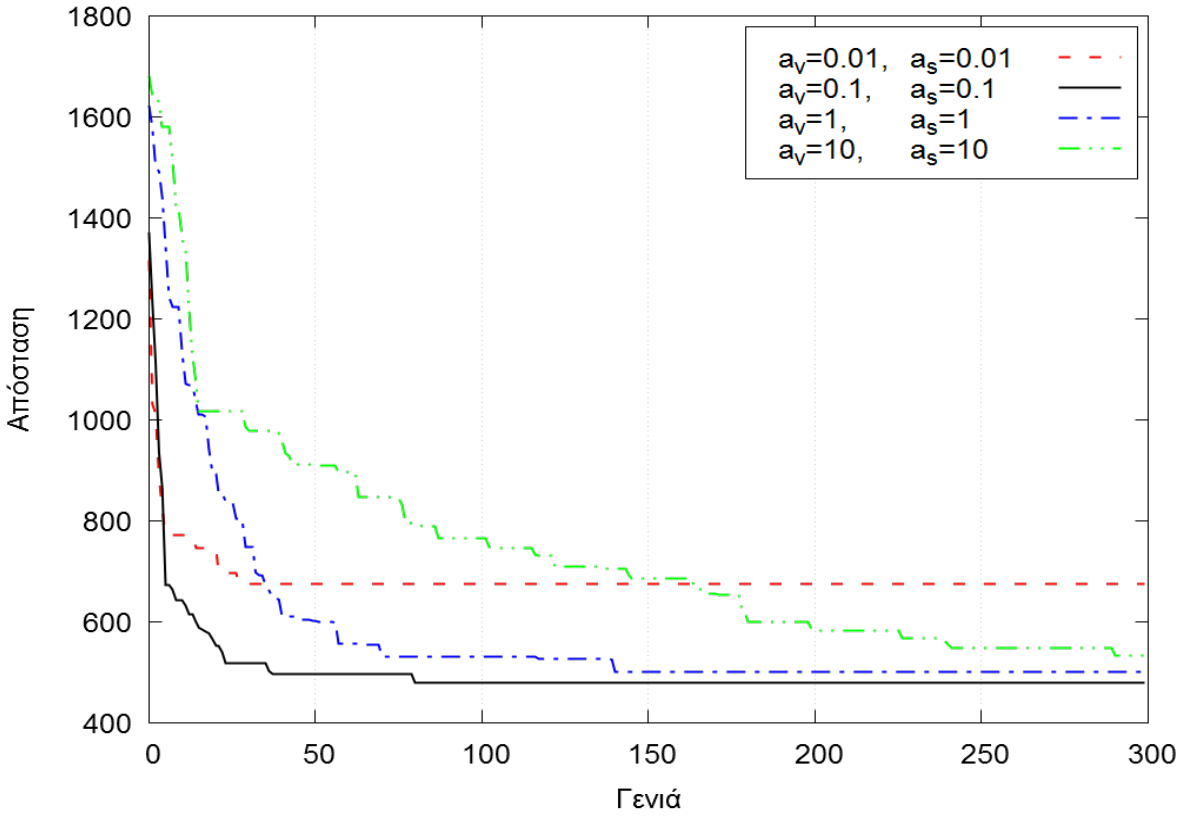


Σχήμα 12: Αποτελέσματα 10 εκτελέσεων για eil101

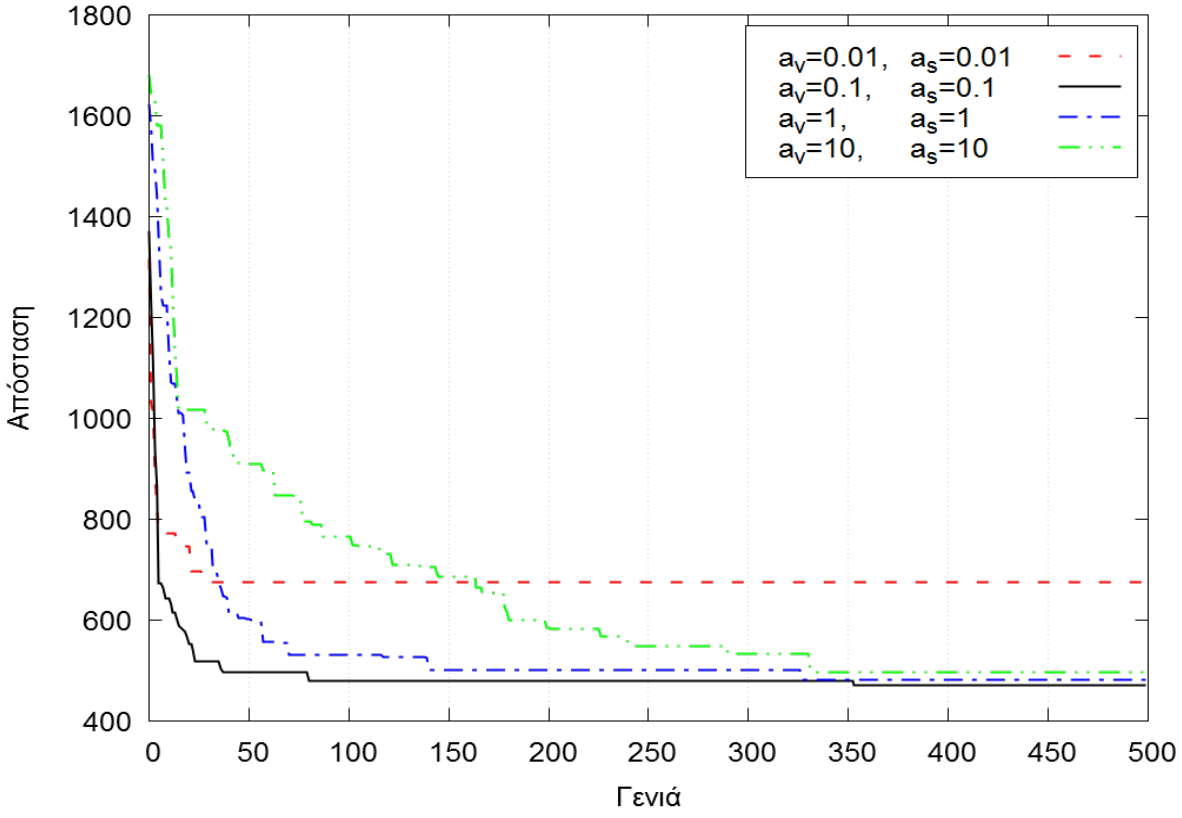


Σχήμα 13: Αποτελέσματα 10 εκτελέσεων για a280

Σχήμα 14: Αποτελέσματα Πειράματος 2 για διαφορετικά a_v/a_s

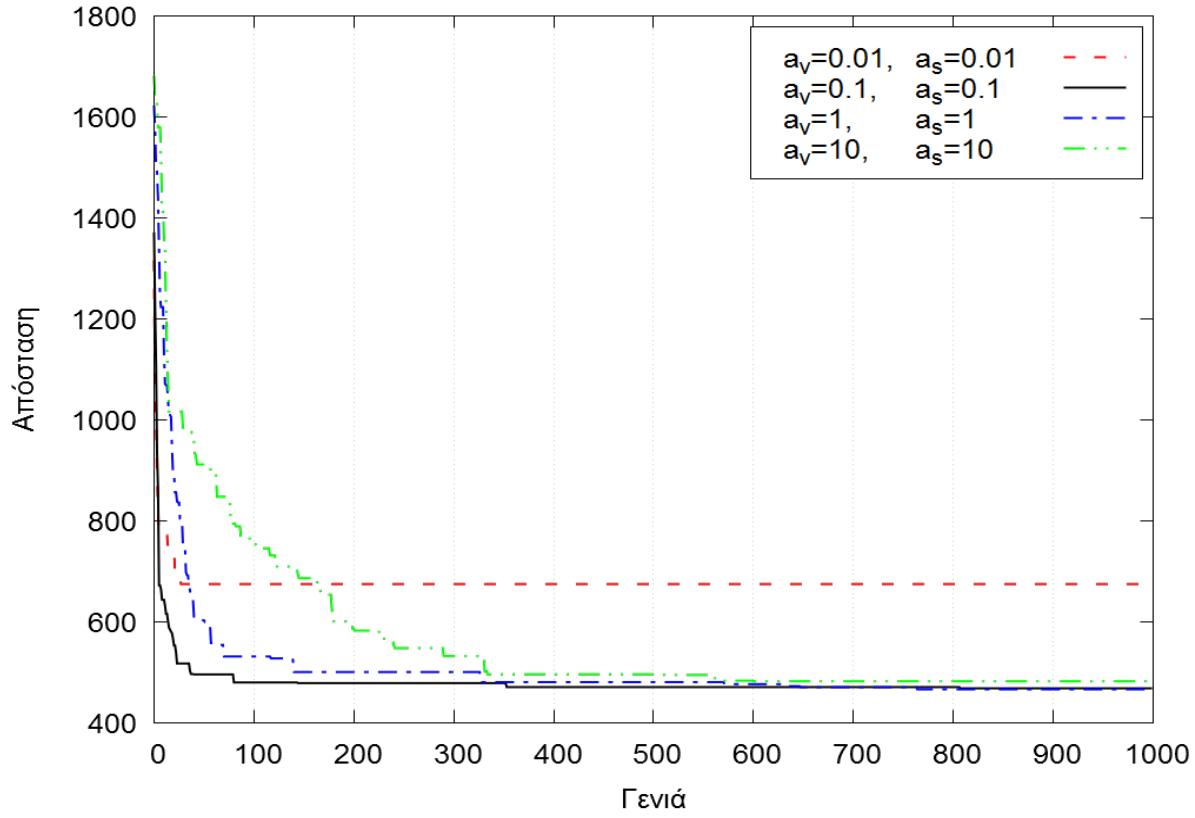


(α΄) 300 γενιές

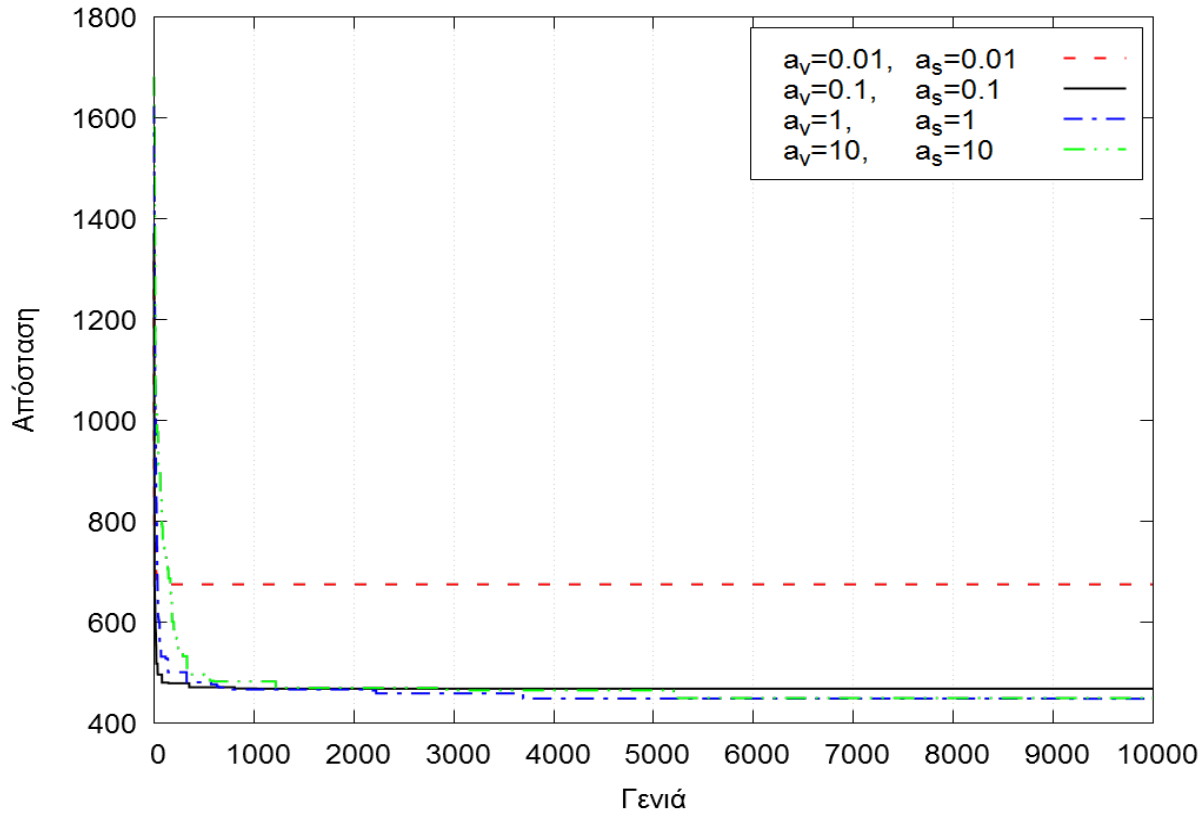


(β΄) 500 γενιές

Σχήμα 15: Αποτελέσματα Πειράματος 2 για διαφορετικά a_v/a_s

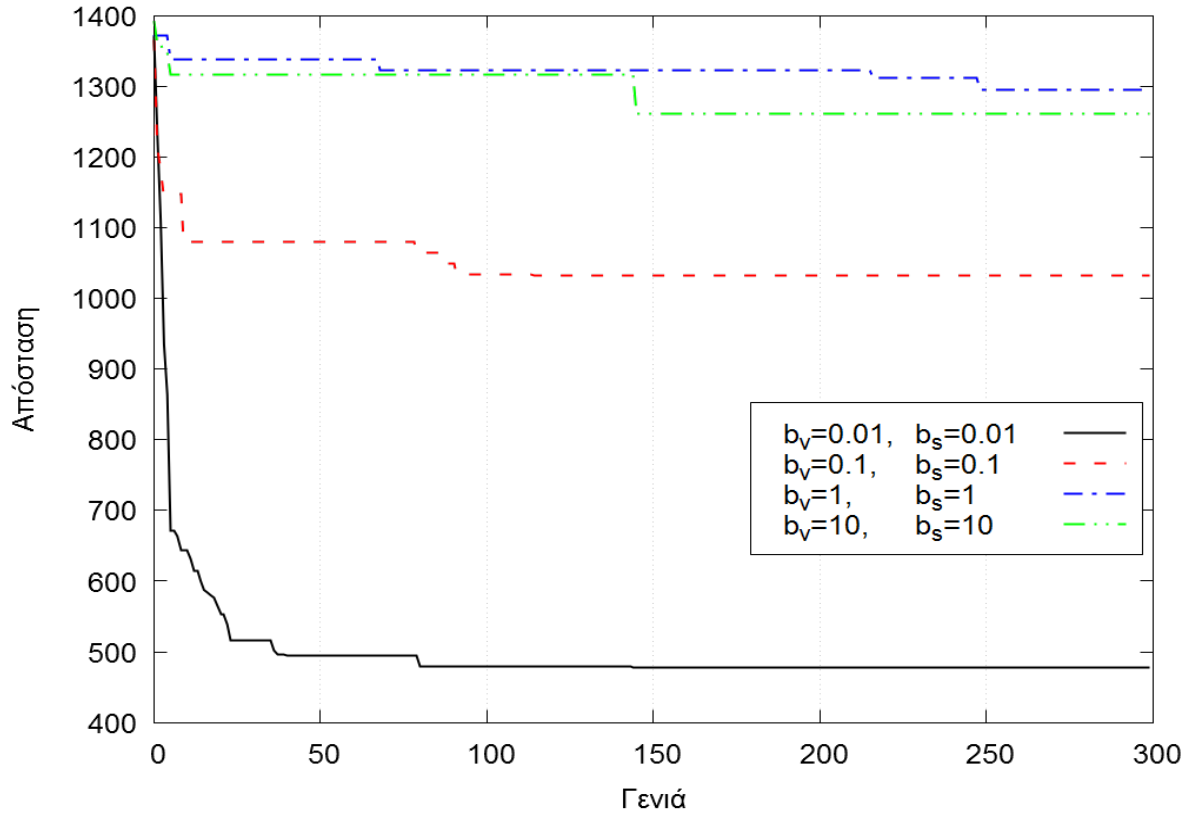


(α΄) 1000 γενιές

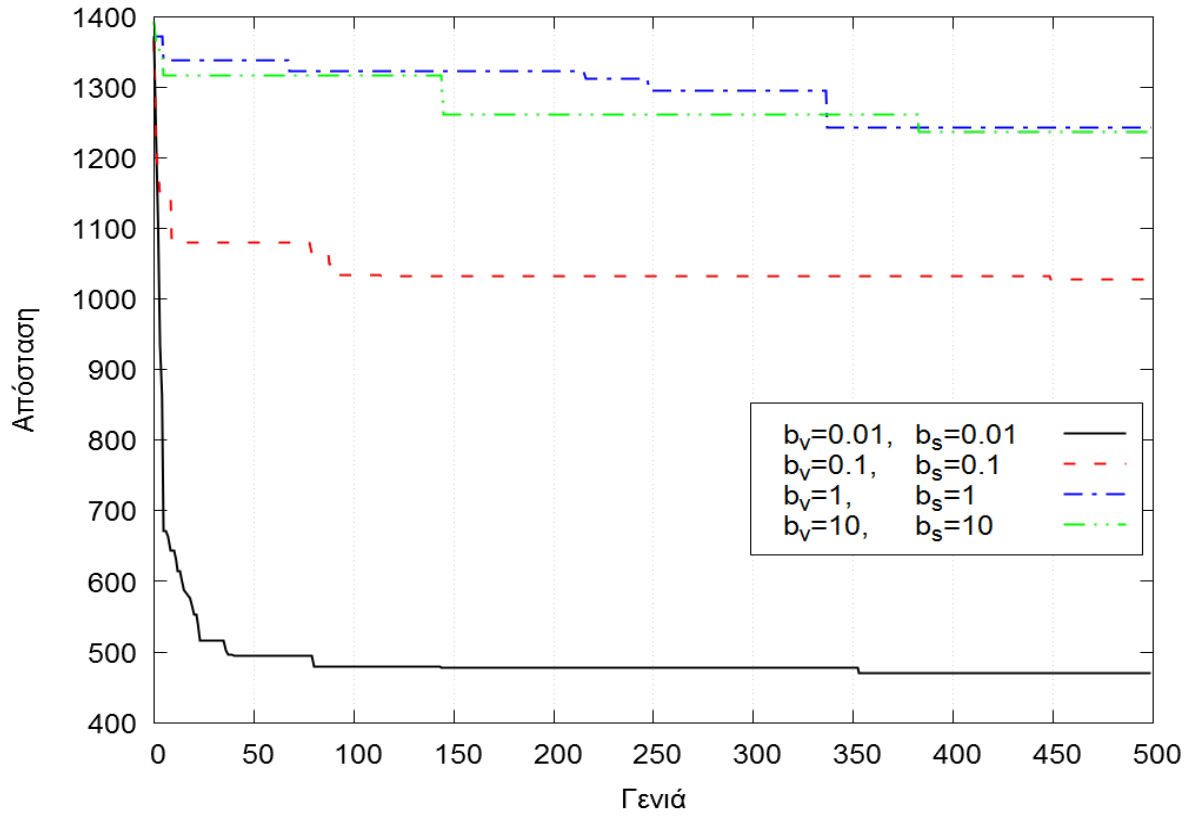


(β΄) 10000 γενιές

Σχήμα 16: Αποτελέσματα Πειράματος 2 για διαφορετικά b_v/b_s

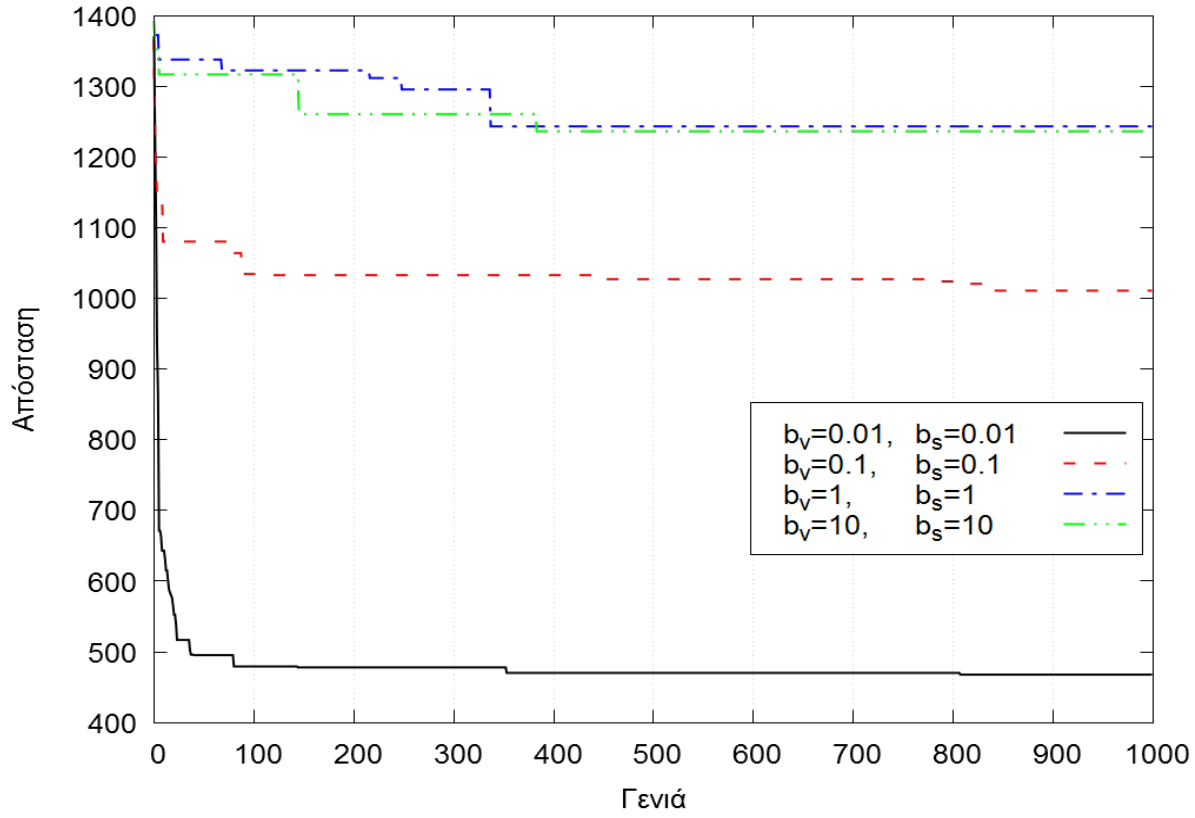


(α΄) 300 γενιές

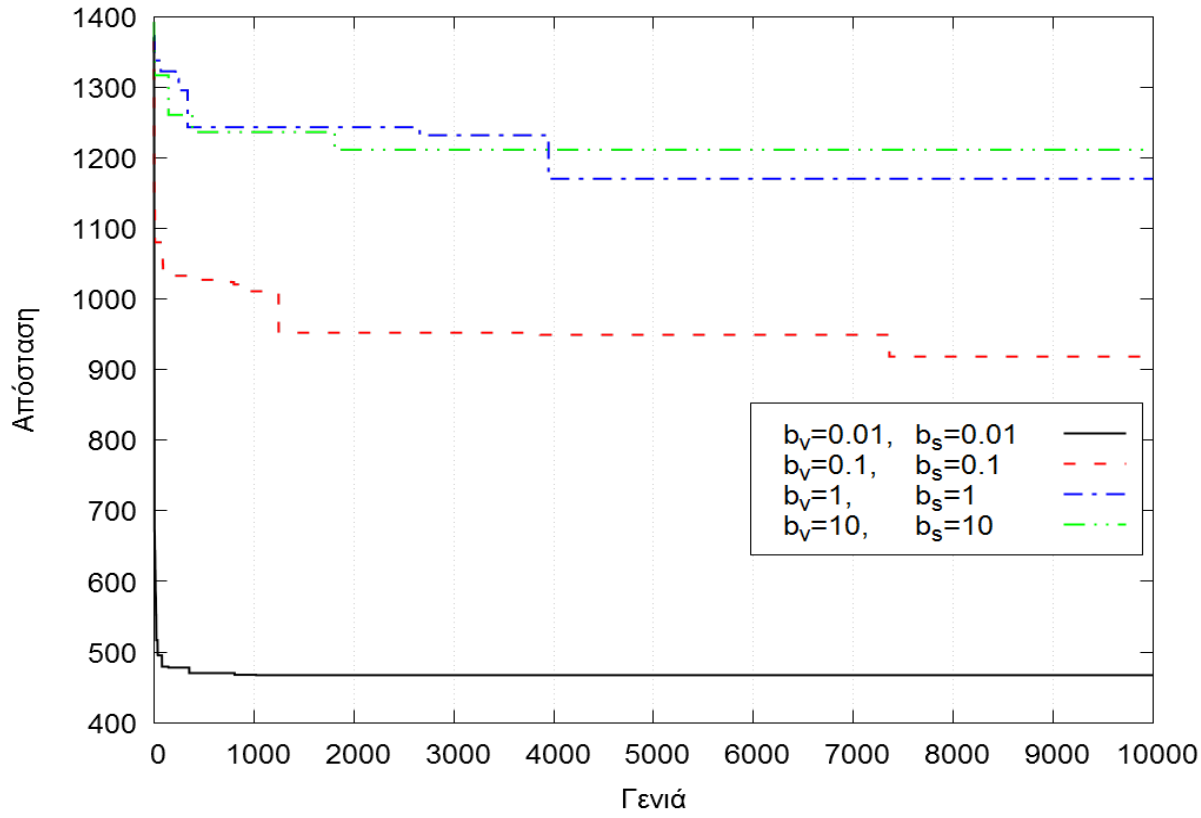


(β΄) 500 γενιές

Σχήμα 17: Αποτελέσματα Πειράματος 2 για διαφορετικά b_v/b_s

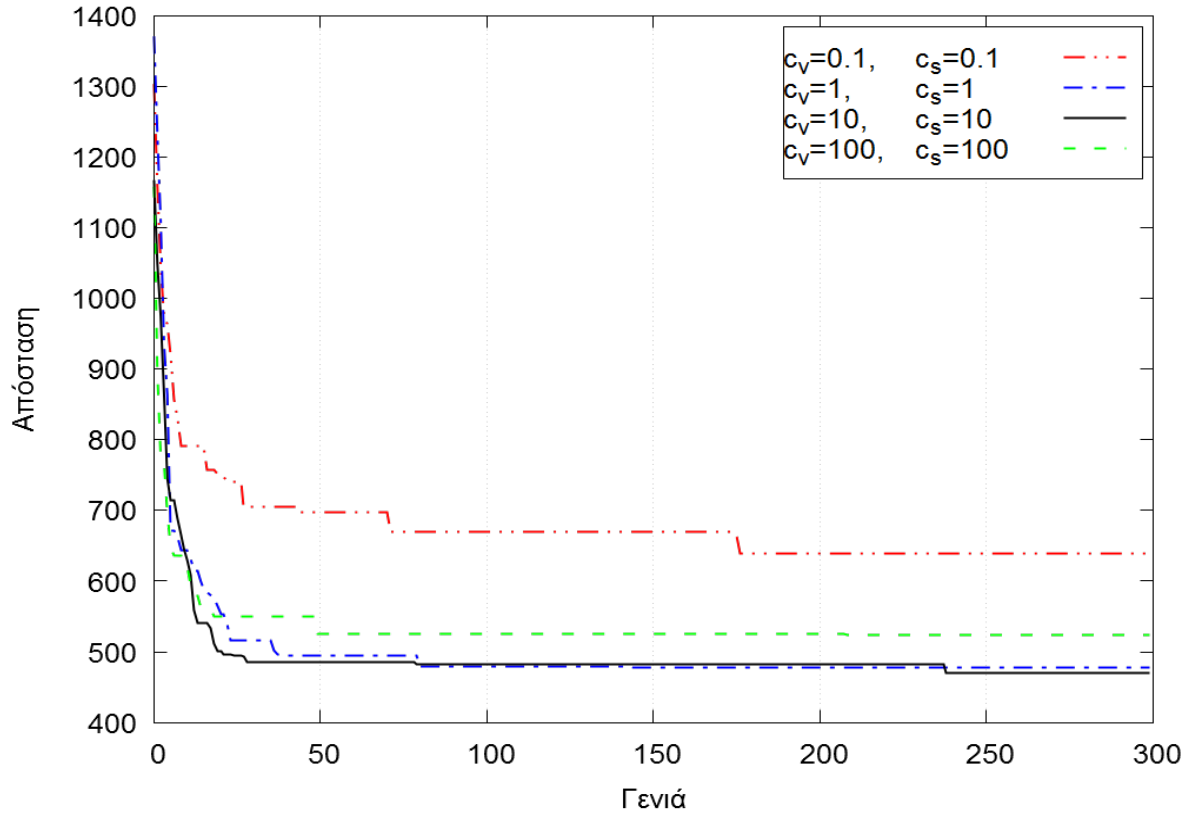


(α΄) 1000 γενιές

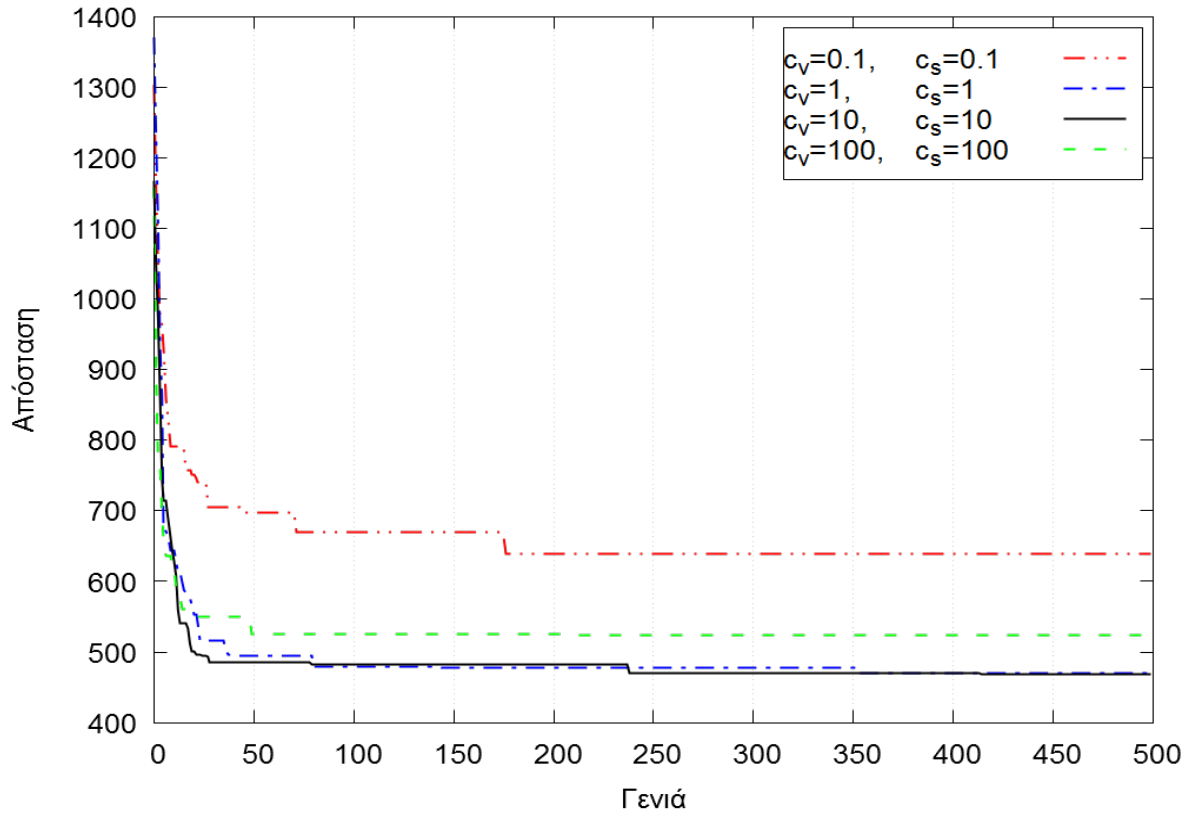


(β΄) 10000 γενιές

Σχήμα 18: Αποτελέσματα Πειράματος 2 για διαφορετικά c_v/c_s

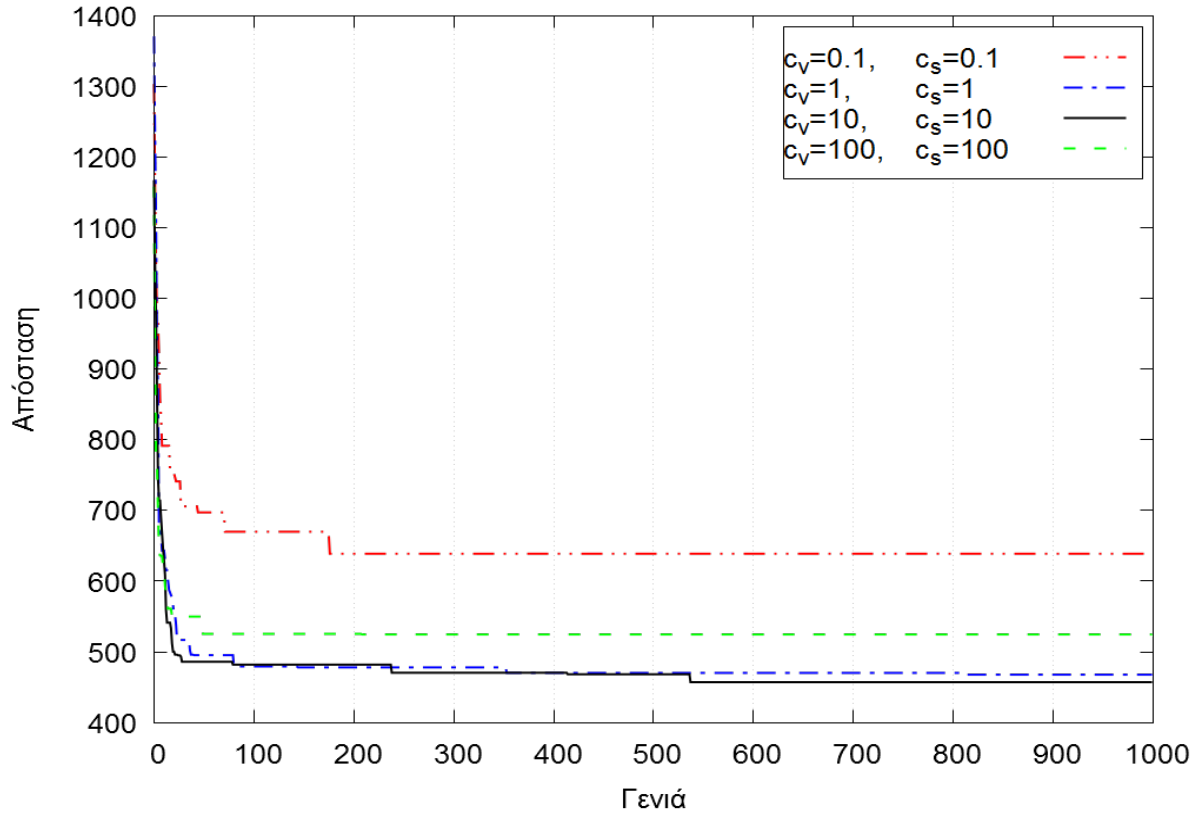


(α΄) 300 γενιές

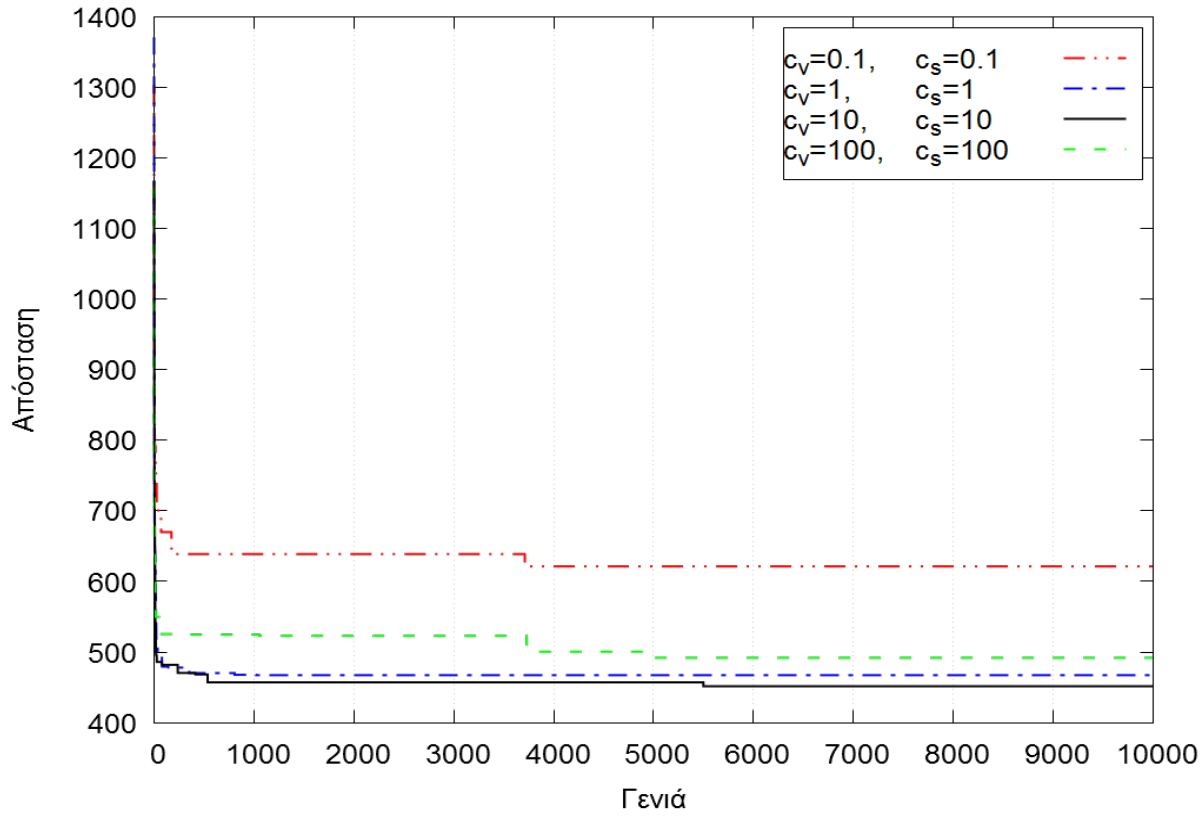


(β΄) 500 γενιές

Σχήμα 19: Αποτελέσματα Πειράματος 2 για διαφορετικά c_v/c_s

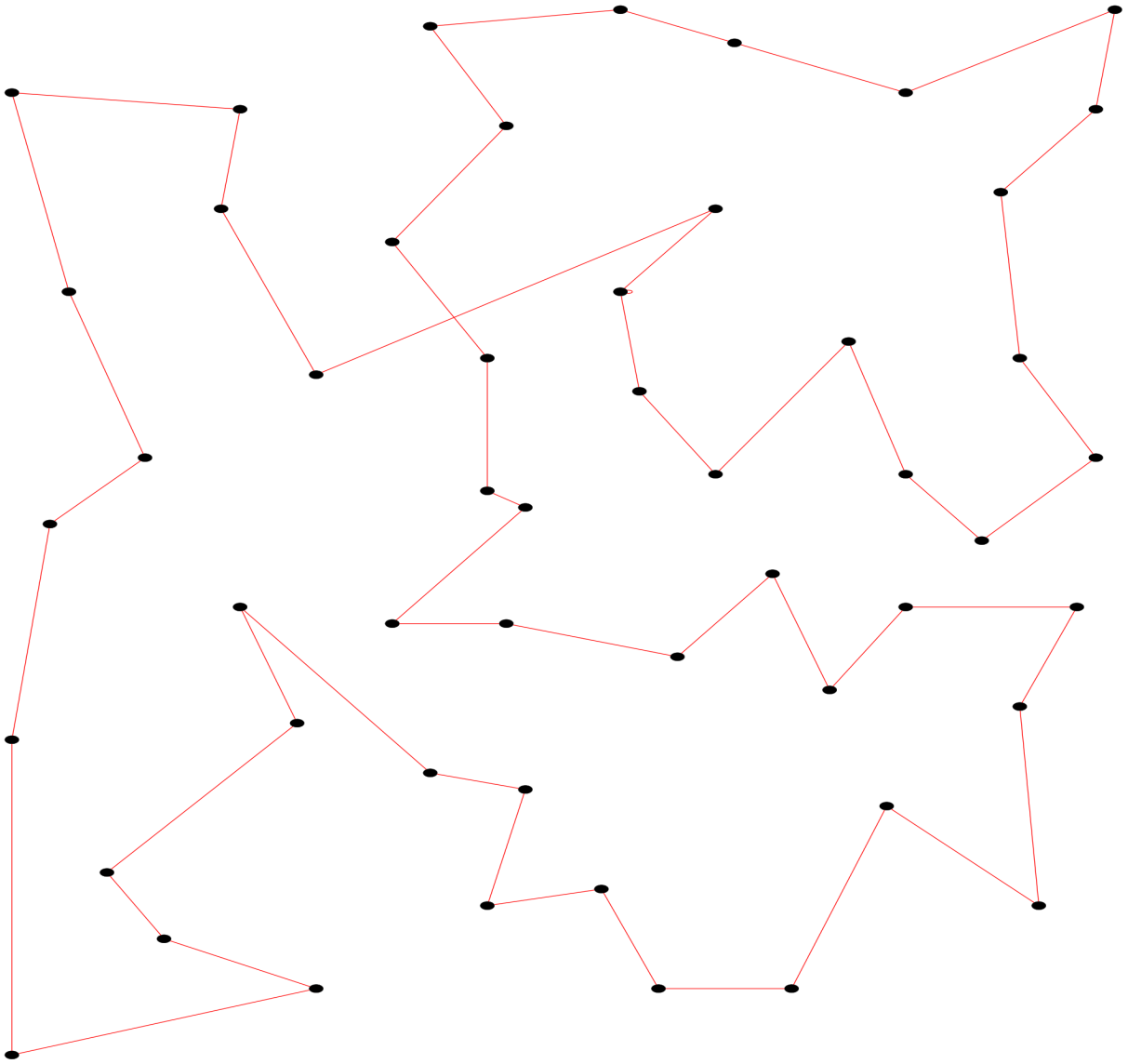


(α΄) 1000 γενιές

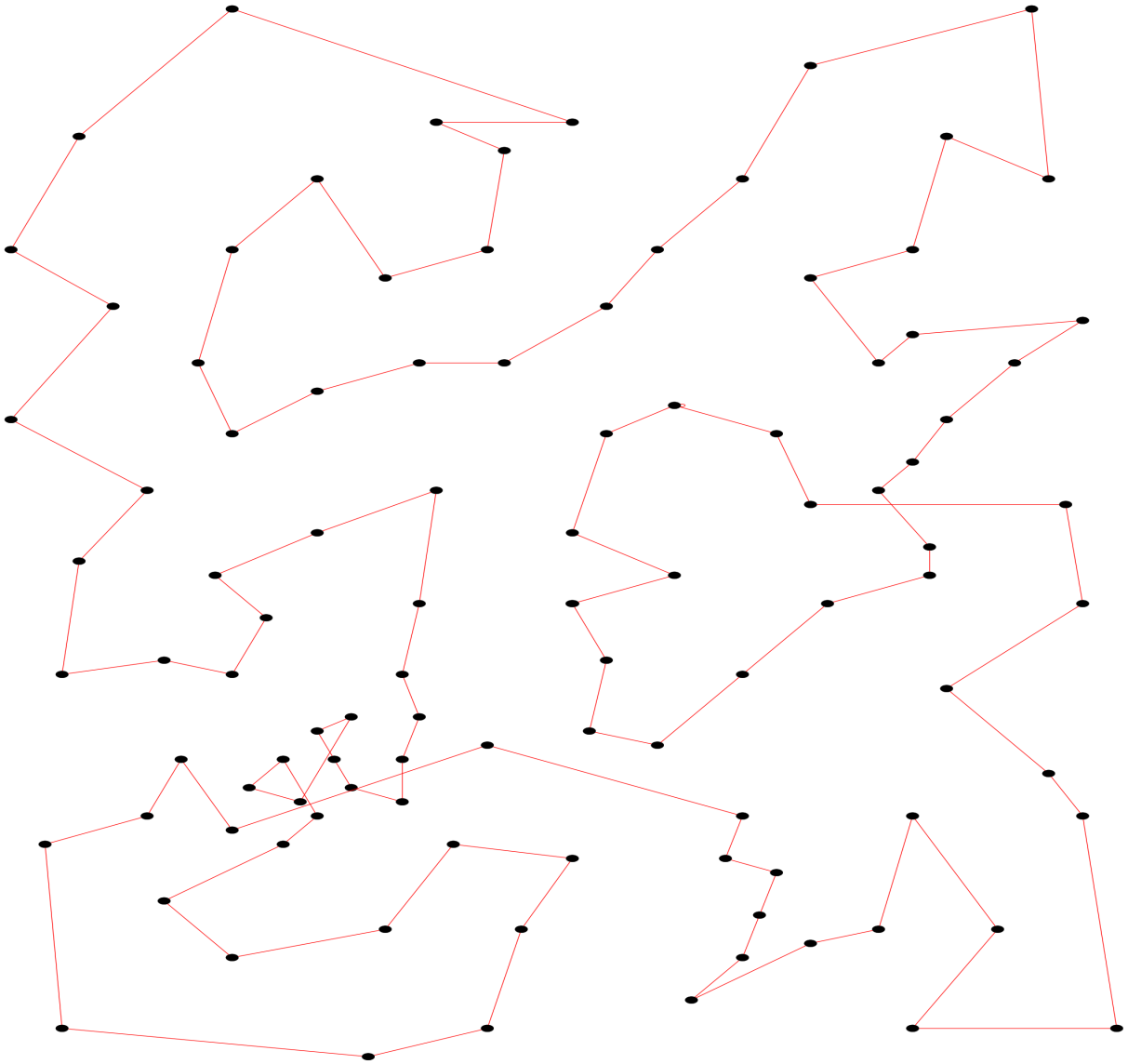


(β΄) 10000 γενιές

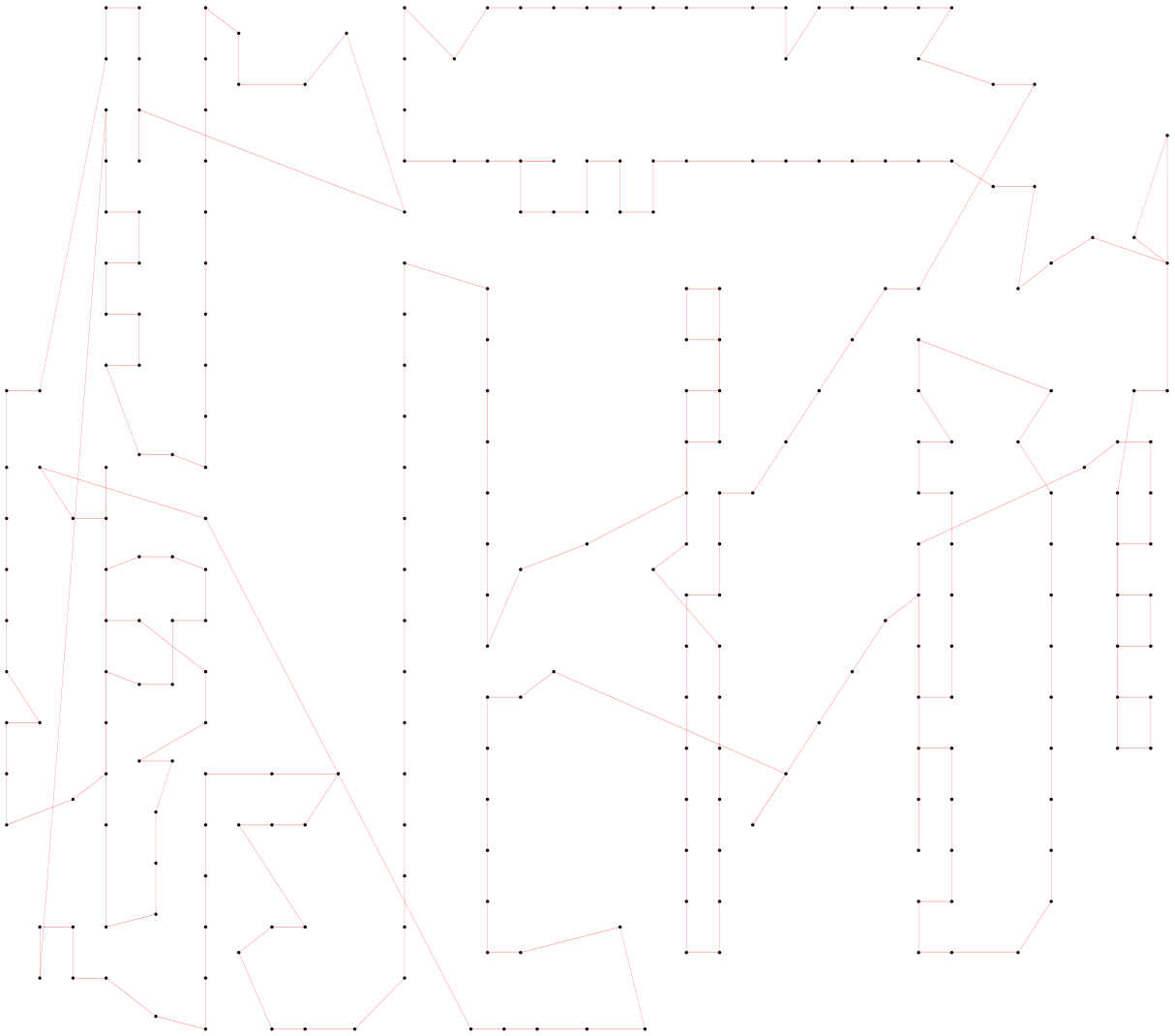
10.1 Αναπαράσταση καλύτερων λύσεων που βρέθηκαν



Σχήμα 20: Αναπαράσταση βέλτιστης (σύμφωνα με τα αποτελέσματά μας) διαδρομής για eil51



Σχήμα 21: Αναπαράσταση βέλτιστης (σύμφωνα με τα αποτελέσματά μας) διαδρομής για eil101



Σχήμα 22: Αναπαράσταση βέλτιστης (σύμφωνα με τα αποτελέσματά μας) διαδρομής για a280

11 Ακρώνυμα

Ακρώνυμα

ACO Ant Colony Optimization. 3

CIL Common Intermediate Language. 13

CLI Command Line Interface. 14

CLI Commom Language Infrastructure. 13–15

CLR Commom Language Runtime. 13, 15

HUD Heuristic Undesirability. 6

IDE Integrated Development Environment. 7

IWD Intelligent Water Drop. 2

JVM Java Virtual Machine. 13

RNG Random Number Generator. 12

STL Standard Template Library. 13

TSP Travelling Salesman Problem. 2–4

UML Unified Modeling Language. 11

VS2015 Visual Studio 2015. 7

12 Βιβλιογραφία

[1] J. van Leeuwen, *Handbook of theoretical computer science*, vol. A. Amsterdam [u.a.] Elsevier [u.a.], second ed., 1998.

[2] J. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975.

[3] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," 1991.

[4] D. Karaboga, "An idea based on honey bee swarm for numerical optimization, technical report - tr06," 01 2005.

[5] D. T. Pham and M. Castellani, "The bees algorithm: Modelling foraging behaviour to solve continuous optimization problems," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 223, no. 12, pp. 2919–2938, 2009.

[6] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "Gsa: A gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232 – 2248, 2009. Special Section on High Order Fuzzy Sets.

[7] Z. W. Geem, J. H. Kim, and G. Loganathan, "A new heuristic optimization algorithm: Harmony search," *SIMULATION*, vol. 76, no. 2, pp. 60–68, 2001.

[8] K. Kaipa and S. S. Unsubscribe, "Glowworm swarm optimization,"

[9] M. Eusuff, K. Lansey, and F. Pasha, "Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization," *Engineering Optimization*, vol. 38, no. 2, pp. 129–154, 2006.

[10] X.-S. Yang, *Nature Inspired Cooperative Strategies for Optimization*, ch. A New Metaheuristic Bat-Inspired Algorithm. Springer, Berlin, Heidelberg, 2010.

[11] G. Beni, *From Swarm Intelligence to Swarm Robotics*, pp. 1–9. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[12] L. Rosenberg, "Artificial swarm intelligence, a human-in-the-loop approach to a.i.," 2016.

[13] H. S. Hosseini, "Problem solving by intelligent water drops," *2007 IEEE Congress on Evolutionary Computation*, 2007.

[14] H. Shah-Hosseini, "Intelligent water drops algorithm: A new optimization method for solving the multiple knapsack problem," *International Journal of Intelligent Computing and Cybernetics*, vol. 1, no. 2, pp. 193–212, 2008.

[15] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 53–66, 1996.

[16] H. Shah-Hosseini, "The intelligent water drops algorithm: a nature-inspired swarm-based optimisation algorithm," *Int. J. Bio-Inspired Computation*, vol. 1, no. 1/2, pp. 71–79, 2009.

[17] L. Caccetta and A. Kulanoot, "Computational aspects of hard knapsack problems," vol. 47, pp. 5547–5558, 08 2001.

[18] J. J. Watkins, *Across the Board: The Mathematics of Chessboard Problems*. Princeton University Press, 2004.

[19] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," in *Proceedings of the 1961 16th ACM National Meeting*, ACM '61, (New York, NY, USA), pp. 71.201–71.204, ACM, 1961.

[20] "C++/cli language specification," standard, Ecma International, Geneva, CH, dec 2005.

[21] G. Reinelt, *TSPLIB 95*. Universitat Heidelberg Institut fur Angewandte Mathematik Im Neuenheimer, Feld 294 D-69120 Heidelberg. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

[22] M. M. Msallam and M. Hamdan, "Improved intelligent water drops algorithm using adaptive schema," *International Journal of Bio-Inspired Computation*, vol. 3, no. 2, pp. 103–111, 2011. PMID: 39909.

13 Πηγαίος κώδικας και αρχεία

13.1 Πηγαίος κώδικας βιβλιοθηκών

Utilities.hpp

```
1
2  #pragma once
3  #include <chrono>
4  #include <string>
5  #include <algorithm>
6  #include <regex>
7  #include <fstream>
8  #include <stdexcept>
9  #include "IWD/Settings.hpp"
10 #include "args.hxx"
11 #include "json.hpp"
12 #include "Graph.h"
13
14 #include <fcntl.h>
15 #include <corecrt_io.h>
16
17 #define GREEK(x) {              \
18   _setmode(_fileno(stdout), _O_U8TEXT); \
19   std::wcout << x << std::endl;     \
20   _setmode(_fileno(stdout), _O_TEXT);   \
21 }
22 using json = nlohmann::json;Για
23
24 // μέτριση χρόνου εκτέλεσης
25 template<typename TimeT = std::chrono::microseconds,
26   typename ClockT = std::chrono::high_resolution_clock,
27   typename DurationT = double>
28   class Stopwatch {
29   private:
30     std::chrono::time_point<ClockT> _start, _end;
31   public:
32     Stopwatch() { start(); }
33     void start() { _start = _end = ClockT::now(); }
34     DurationT stop() { _end = ClockT::now(); return elapsed(); }
35     DurationT elapsed() {
36       auto delta = std::chrono::duration_cast<TimeT>(_end - _start);
37       return delta.count();
38     }
39 };//https://codereview.stackexchange.com/a/48884Όνομα
40
41 // αρχείου απο πλήρη διαδρομή
42 inline std::string basename(const std::string& pathname) {
43   return{ std::find_if(pathname.rbegin(), pathname.rend(), [&](char
      c) { return (c == '/' || c == '\\'); }).base(),
44     pathname.end() };
```

```
}//https://stackoverflow.com/questions/8520560/get-a-file-name-from-a-pathΔιαβάζουμε

// έναν γράφο απο αρχείο
inline auto graphFileReader(const std::string& filepath) {

  std::ifstream tsp_file(filepath);
  if(!tsp_file) {
    throw std::invalid_argument("Unable to read file " + filepath);
  }

  enum SEARCH_STATE { EDGE_TYPE_SEARCH, FIRST_NODE_SEARCH,
   READ_NODES } search_state = EDGE_TYPE_SEARCH;

  std::string t_line;

  std::regex edge_weight_type(R"(EDGE_WEIGHT_TYPE\s*:\s*EUC_2D)");
  std::regex node_coord_start(R"(\s*NODE_COORD_SECTION)");

  Graph g;

  /*
  * Εδώ έχουμε ένα μικρό state machine. Ψάχνει όλο το αρχείο γραμμή
   γραμμή
  * για συγκεκριμένα μοτίβα. Το αρχείο πρέπει να έχει συγκεκριμένη μορφή
   για
  * να μπορέσουμε να το προσπελλάσουμε.
  */
  while(std::getline(tsp_file, t_line) && t_line != "EOF") {
    switch(search_state) {
    case EDGE_TYPE_SEARCH:
      if(std::regex_match(t_line, edge_weight_type)) {
        search_state = FIRST_NODE_SEARCH;
      } else if(tsp_file.eof()) {
        throw std::invalid_argument("Invalid input file type. Must
   be EUC_2D");
      }
      break;
    case FIRST_NODE_SEARCH:
      if(std::regex_match(t_line, node_coord_start)) search_state =
   READ_NODES;
      tsp_file.unget(); //Hack για να ξαναδιαβάσουμε την γραμμή.
   γενικά έχει θέμα όταν χρησιμοποιείς getline και operator>>
      break;
    case READ_NODES:
      unsigned int label;
      double x = 0.0, y = 0.0;
      if(tsp_file >> label >> x >> y) {
        g.addNode(x, y);
      }
      break;
```

```
89      }
90    }
91    return std::move(g);
92  }
93
94  inline void printSettings(const iwd::Settings& s) {
95    GREEK(LΑρχικό" soil \t: " << s.initial_soil);
96    GREEK(LΑρχικό" velocity \t:" << s.initial_velocity);
97    GREEK(Lε" = " << s.e);
98    GREEK(LΠαράμετροι" ανανέωσης velocity \t" <<
99      "  a = " << s.av << " b = " << s.bv << " c = " << s.cv);
100   GREEK(LΠαράμετροι" ανανέωσης soil \t" <<
101     "  a = " << s.as << " b = " << s.bs << " c = " << s.cs);
102   GREEK(LΤοπικήολική"/ παράμετρος ανανέωσης soil : p = " << s.p);
103 }Διαβάζουμε
104
105 // τις ρυθμίσεις απο ένα αρχείο JSON
106 inline iwd::Settings readJsonSettings(const std::string filepath) {
107   std::ifstream ifs(filepath);
108   if(!ifs) {
109     throw std::invalid_argument("Can't read options file.");
110   }
111   json settings_json;
112   ifs >> settings_json;
113
114   auto s = iwd::Settings();
115
116   s.initial_soil = settings_json.count("initial_soil") ?
       settings_json["initial_soil"] : s.initial_soil;
117   s.initial_velocity = settings_json.count("initial_velocity") ?
       settings_json["initial_velocity"] : s.initial_velocity;
118   s.av = settings_json.count("av") ? settings_json["av"] : s.av;
119   s.bv = settings_json.count("bv") ? settings_json["bv"] : s.bv;
120   s.cv = settings_json.count("cv") ? settings_json["cv"] : s.cv;
121   s.as = settings_json.count("as") ? settings_json["as"] : s.as;
122   s.bs = settings_json.count("bs") ? settings_json["bs"] : s.bs;
123   s.cs = settings_json.count("cs") ? settings_json["cs"] : s.cs;
124   s.e = settings_json.count("e") ? settings_json["e"] : s.e;
125   s.p  = settings_json.count("p") ? settings_json["p"] : s.p   ;
126   s.random_initial_node = settings_json.count("random_initial_node")
        ? true : false;
127   s.seed = settings_json.count("seed") ? settings_json["seed"] : s.
       seed;
128   s.max_iterations = settings_json.count("max_iterations") ?
       settings_json["max_iterations"] : s.max_iterations;
129   s.out_filename = settings_json.count("out_filename") ?
       settings_json["out_filename"] : s.out_filename;
130   s.no_files = settings_json.count("no_files") ? true : false;
131
132   return s;
133 }
```

```
1
2  #pragma once
3  #include <vector>
4  #include <sstream>
5  #include "IWD/Settings.hpp"
6  /*
7   * Μία πολύ απλή υλοποίηση ενός γράφου ακμές( και κόμβοι)
8   */To
9
10
11 // ορίζουμε σαν soil για να μπορέσουνε να το αλλάξουμε αν χρειαστεί
       χωρίς πρόβλημα
12 using soil = double;
13
14 struct Node {
15   double x, y;
16   unsigned int id;
17
18
19   Node(double x_i = 0, double y_i = 0, unsigned int id_i = 0) : x(
       x_i), y(y_i), id(id_i) {};
20
21   //Utils
22   std::string str() const{
23     std::ostringstream out;
24     out << "N" << id << "[" << x << "," << y << "]";
25     return out.str();
26   }
27 };
28
29
30 class Graph {
31 public:
32   Graph() = default;
33   Graph(int node_count, iwd::Settings);
34   std::vector<Node> nodes;
35   std::vector<std::vector<double>> edgeSoil;
36
37   iwd::Settings settings;
38
39   Graph* addNode(int x_i, int y_i);
40   Graph* addNode(double x_i, double y_i);
41   soil getSoil(const Node& n1, const Node& n2) const;
42
43   Graph * setSoil(const Node& n1, const Node& n2, soil);
44
45   void initiateSoil(soil initial_soil) {
46     edgeSoil = move(std::vector< std::vector<soil>>(nodes.size(),
       std::vector<soil>(nodes.size(), initial_soil)));
47   }
```

```
48
49
50 };
```

Graph.cpp

```
1  #include "Graph.h"
2
3  Graph::Graph(int node_count, iwd::Settings s) {
4    nodes.reserve(node_count);
5    edgeSoil = std::vector< std::vector<soil>>(node_count, std::vector
       <soil>(node_count,s.initial_soil));
6  }
7
8  Graph* Graph::addNode(int x_i, int y_i) {
9    nodes.push_back(Node(x_i, y_i, nodes.size()));
10   edgeSoil.push_back(std::vector<soil>(settings.initial_soil));
11   return this;
12 }
13
14 Graph* Graph::addNode(double x_i, double y_i) {
15   nodes.push_back(Node(x_i, y_i, nodes.size()));
16   edgeSoil.push_back(std::vector<soil>(settings.initial_soil));
17   return this;
18 }
19
20
21 soil Graph::getSoil(const Node& n1, const Node& n2) const{
22   return edgeSoil[n1.id][n2.id];
23 }
24
25 Graph * Graph::setSoil(const Node& n1, const Node& n2, soil new_soil
       ) {
26   edgeSoil[n1.id][n2.id] = new_soil;
27   edgeSoil[n2.id][n1.id] = new_soil;
28   return this;
29 }
```

Drop.cpp

```
1
2  #include "Drop.h"
3
4  namespace iwd {
5
6    Drop::Drop(Graph& g_i, Settings s_i) : graph(g_i),settings(s_i){
7      this->visited_node_map = std::vector<bool>(graph.nodes.size()+1,
         false);
8      this->visited_nodes.reserve(graph.nodes.size()+1);
9      if(settings.random_initial_node) {
10       unsigned min = 0, max = graph.nodes.size() - 1;
11       auto initial_node_id = min + rand() % (int)(max - min + 1);
```

```
12        this->initial_node = &this->graph.nodes[initial_node_id];
13      } else {
14        this->initial_node = &this->graph.nodes[0];
15      }
16    }
17
18    void Drop::setVisited(Node* n) {
19      this->visited_nodes.push_back(n);
20      this->visited_node_map[n->id] = true;
21    }
22
23    void Drop::travel() {
24      if(this->graph.nodes.size() == 0) {
25        GREEK(LO" γράφος δεν έχει κόμβους");
26        return;
27      }
28      έχουμε// το 1ο node
29      this->current_node = this->initial_node;
30      Αρχικοποιούμε// κάποιες μεταβλητές
31      this->visited_nodes.clear();
32      this->visited_node_map = std::vector<bool>(graph.nodes.size(),
       false);
33      this->velocity = this->settings.initial_velocity;
34      this->path_length = 0;
35      this->soil = 0;
36
37      auto loop_done = false;
38      while(!loop_done) {
39        this->setVisited(this->current_node);
40        Παίρνουμε// όλους τους κόμβους που δεν έχουμε επισκευθεί
41        auto valid_nodes = this->getValidNodes();
42        Επιλέγουμε// τον επόμενο κόμβο
43        if(valid_nodes.size() >= 1) {//>1 valid nodes
44          Node* next_node = roulete(valid_nodes);
45
46          Αυξάνουμε// το path length της τωρινής διαδρομής.
47          this->path_length += this->euclidDist(*current_node,*
       next_node);
48
49          Πάμε// στον επόμενο κόμβο και ανανεώνουμε τις μεταβλητές κατα
       την μετάβαση
50          auto target_edge_soil = graph.getSoil(*current_node, *
       next_node);
51          Ανανεώνουμε// το soil και velocity
52          //> Ανανεώνουμε το velocity της σταγόνας
53          this->velocity += this->settings.av / (this->settings.bv +
       this->settings.cv * pow(target_edge_soil, 2));
54          //> ανανεώνουμε το soil της διαδρομής
55          auto new_soil = (1 - this->settings.p) * target_edge_soil -
       this->settings.p * deltaSoil(*next_node);
56          this->graph.setSoil(*current_node,*next_node, new_soil);
```

```
57          this->soil += deltaSoil(*next_node);
58          this->current_node = next_node;
59        } else {// 0 πιθανοί κόβοι -> τέλος διαδρομής
60          Επιστρέφουμε// στο o1 κόμβο για να ολοκληρώσουμε το ταξίδι μας
61          Node* next_node = this->visited_nodes[0];
62          this->path_length += this->euclidDist(*current_node, *
     next_node);
63          auto target_edge_soil = graph.getSoil(*current_node, *
     next_node);
64          this->velocity += this->settings.av / (this->settings.bv +
     this->settings.cv * pow(target_edge_soil, 2));
65          auto new_soil = (1 - this->settings.p) * target_edge_soil -
     this->settings.p * deltaSoil(*next_node);
66          this->graph.setSoil(*current_node, *next_node, new_soil);
67          this->soil += deltaSoil(*next_node);
68          this->setVisited(next_node);
69          loop_done = true;
70        }
71      }
72    }
73    //fsoil(i,j)
74    double Drop::f(const Node& target_node, double minimum_soil) {
     Υπολογίζουμε// g(soil(i,j))
75      auto g = [&, target_node]() -> double {
76        return minimum_soil >= 0 ? minimum_soil : graph.getSoil(*this
     ->current_node, target_node) - minimum_soil;
77      };
78      return 1 / (0.01 + g()); //Hardcode το εs
79    }
80    // Heuristic Undesirability
81    double Drop::HUD(const Node& target_node) const {
82      auto x1 = this->current_node->x, x2 = target_node.x;
83      auto y1 = this->current_node->y, y2 = target_node.y;
84      return std::sqrt(std::pow(x2 - x1, 2) + std::pow(y2 - y1, 2));
85    }
86    Δ//soil(i,j)
87    double Drop::deltaSoil(const Node& target_node) const {
88      auto a{ this->settings.as }, b{ this->settings.bs }, c{ this->
     settings.cs };
89      return a /( b + c * (HUD(target_node) / std::max(this->settings.
     e, this->velocity)));
90    }
91
92    //      f(soill(i,j))
93    //Pi(j) = ---------------
94    //         ∑f(soil(i,k))   | k !in visitedNodes
95    //hosseini2009 p 2
96    auto Drop::calculateP(collection& valid_nodes) {
97      Πιθανότητες// για κάθε node απο αυτό που βρισκόμαστε τώρα
98      std::vector<std::pair<Node*, double>> Pi;
99      Pi.reserve(valid_nodes.size());
```

```
100       Βρίσκουμε// το ελλάχιστον soil στους ελεύθερους κόμβους
101       auto minimum_soil = ([&]() {
102         auto min = graph.getSoil(*this->current_node, *(valid_nodes
        [0]));
103         for(auto n : valid_nodes) {
104           min = std::min(min, graph.getSoil(*this->current_node, *n));
105         }
106         return min;
107       })();
108
109       Υπολογίζουμε// το Σf(i,j)
110       auto sumf = 0.0;
111       for(auto const target_node : valid_nodes) {
112         sumf += f(*target_node, minimum_soil);
113       }
114
115       for(auto node : valid_nodes) {
116         auto p = f(*node, minimum_soil) / sumf;
117         Pi.push_back(std::make_pair(node, p));
118       }
119       return move(Pi);
120     }
121     Η// συνάρτηση επιλογής του επόμενου κόμβου
122     βάση// των πιθανοτήτων που μας δίνει η calculateP()
123     Node* Drop::roulete(collection& nodes) {
124       if(nodes.size() == 1) return nodes[0]; Αν// έχουμε μόνο ένανα
        κόμβο, δεν ψάχνουμε
125       Node* next_node = nullptr;
126       Επιλέγουμε// τον καλύτερο κόμβο βάση του τύπου
127       auto Pi = calculateP(nodes);
128
129       static std::mt19937 gen(this->settings.seed);
130       std::uniform_real_distribution<> dis(0, 1);
131
132       double cumulative = 0;
133       auto t = dis(gen);
134       for(auto& p : Pi) {
135         if(t <= cumulative + p.second) {
136           return p.first;
137         }
138         cumulative += p.second;
139       }
140     }
141
142     Ευκλείδια// απόσταση μεταξύ 2 κόμβων
143     double Drop::euclidDist(Node& n1, Node& n2) const{
144       auto x1 = n1.x, x2 = n2.x;
145       auto y1 = n1.y, y2 = n2.y;
146       return std::sqrt(std::pow(x2 - x1, 2) + std::pow(y2 - y1, 2));
147     }
148
```

```
149   Επιστρέφει// το μήκος της τωρινής διαδρομής της σταγόνας
150   double  Drop::getCurrentPathLength() {
151     double length = 0;
152     for(unsigned i = 1; i < this->visited_nodes.size() ; i++) {
153       length += euclidDist(*this->visited_nodes[i - 1], *this->
      visited_nodes[i]);
154     }
155     return length;
156   }
157 }
```

Drop.h

```
1
2  #pragma once
3  #include <vector>
4  #include <map>
5  #include <iostream>
6  #include "Settings.hpp"
7  #include "../Graph.h"
8  #include <algorithm>
9  #include <random>
10 #include <math.h>
11 #include <fcntl.h>
12 #include <corecrt_io.h>
13
14 #define GREEK(x) {              \
15   _setmode(_fileno(stdout), _O_U8TEXT); \
16   std::wcout << x << std::endl;     \
17   _setmode(_fileno(stdout), _O_TEXT);   \
18 }
19 namespace iwd {
20
21   typedef std::vector<Node*> collection;
22   class Drop {
23   public:
24
25     Graph& graph;
26     Settings settings;
27
28     double velocity = 0;
29     soil soil = 0;
30
31     Node* current_node = nullptr;
32     Node* initial_node = nullptr;
33     collection visited_nodes;
34     std::vector<bool> visited_node_map;
35
36     long double path_length = 0;
37
38     Drop(Graph& g_i, Settings s_i);
39
```

```
40      collection getValidNodes() {
41        collection valid;
42        valid.reserve(this->graph.nodes.size());
43        for(size_t i = 0; i < this->graph.nodes.size(); i++) {
44          if(!this->visited_node_map[this->graph.nodes[i].id]) {
45            valid.push_back(&graph.nodes[i]);
46          }
47        }
48        return move(valid);
49      }
50
51      auto calculateP(collection& valid_nodes);
52      Node* roulete(collection&);
53      double euclidDist(Node& n1, Node& n2) const;
54      //Utils
55      void setVisited(Node* n);
56      auto getVisitedNodes() {
57        return this->visited_nodes;
58      }
59
60      void travel();
61
62      Ορίζουμε// την συνάρτηση f
63      double f(const Node& target_node, double minimum_soil);
64
65      //HUD ευκλίδεια( απόσταση)
66      Επειδή// στην περίπτωσή μας το HUD συμπίπτει με την απόσταση των 2
        κόμβων
67      χρησιμοποιώ// την ίδια συνάρτηση
68      double HUD(const Node& target_node) const;
69
70      double deltaSoil(const Node& target_node) const;
71
72      static void printPath(const collection& nodes)  {
73        for(auto *node : nodes) {
74          std::cout << node->id << " ";
75        }
76        std::cout << std::endl;
77      }
78
79      double getCurrentPathLength();
80    };
81
82
83  }
```

Settings.hpp

```
1
2   #pragma once
3   #include <string>
4
```

```
namespace iwd {
  class Settings {
  public:
    int initial_velocity = 200, initial_soil = 1000;
    double av = 1;
    double bv = 0.01;
    double cv = 1;
    double as = 1;
    double bs = 0.01;
    double cs = 1;
    double e = 0.0001;
    double p = 0.9; ΤοπικήΟλική/// παράμετρος μεταβολής soil [0,1]
    Άλλες// ρυρθμίσεις
    bool random_initial_node = false;
    int seed = 0;
    unsigned max_iterations = 100;
    std::string out_filename = "output";
    bool no_files = false;
  };
}
```

13.2 Πηγαίος κώδικας `iwd-console`

iwd-console.cpp

```
1
2  #include <iostream>
3  #include <fstream>
4  #include <string>
5  #include <vector>
6  #include <sstream>
7  #include <fcntl.h>
8  #include <corecrt_io.h>
9
10 //Macro για την εμφάνιση ελληνικών(utf8) μηνυμάτων στην κονσόλα
11 #define GREEK(x) {              \
12   _setmode(_fileno(stdout), _O_U8TEXT); \
13   std::wcout << x << std::endl;     \
14   _setmode(_fileno(stdout), _O_TEXT);   \
15 }
16
17 #include "fmt/format.h"
18 #include "IWD/Drop.h"
19 #include "Utilities.hpp"
20 #include "args.hxx"
21
22 using namespace std;
23
24 int main(int argc, char** argv) {
25   Ορίζουμε// τα ορίσματα που δέχεται το πρόγραμμα
26   args::ArgumentParser parser("");
27
28   args::HelpFlag help(parser, "help", "Display this help menu", { 'h
      ', "help" });
29   args::ValueFlag<unsigned> iterations(parser, "iterations", "Number
       of iterations", { 'i' });
30   args::ValueFlag<std::string> graph_file(parser, "filepath", "input
       file to get graph from", { 'g',"graph" });
31   args::ValueFlag<int> seed(parser, "number", "seed to use in random
       functions", { 's',"seed" });
32   args::ValueFlag<std::string> settings_file(parser, "filename", "
      input file to get settings from", { "settings" });
33   args::ValueFlag<std::string> output_file(parser, "filename", "name
       of output file", { 'o',"output" });
34   args::Flag random_initial_node(parser, "random initial node", "if
      set randomizes the 1st node when traveling", { 'r',"random-
      initial-node" });
35   args::Flag print_settings(parser, "print settings", "prints the
      settings used before runing the algorithm", { "po","print-
      settings" });
36   args::Flag no_files(parser, "no files", "does not create files.
      just runs the algorithm", { "no-files" });
37
```

```
38  try {
39    parser.ParseCLI(argc, argv);
40  } catch(args::Help&) {
41    //μφανίζουμεΕ το μενού βοήθειας
42    Χρησιμοποιούμε// το macro επειδή η βιβλιοθήκη args δεν υποστηρίζει
     wchar μεταβλητές
43    GREEK(ΛΕΠΙΛΟΓΕΣ"");
44    std::cout << std::endl;
45    GREEK(fmt::format(L"\t{:<35} {}", L"-h, --help", ΛΕμφανίζει" αυτό
     το κείμενο βοήθειας"));
46    GREEK(fmt::format(L"\t{:<35} {}", L"-i επαναλήψεις[]", ΛΑριθμός"
     επαναλήψεων"));
47    GREEK(fmt::format(L"\t{:<35} {}", L"-s αριθμός[], --seed
     αριθμός=[]", ΛΑριθμός" αρχικοποίησης συναρτήσεων τυχαίων μεταβλητών"))
     ;
48    GREEK(fmt::format(L"\t{:<35} {}", L"-g αρχείο[], --graph
     αρχείο=[]", ΛΕμφανίζει" αυτό το κείμενο βοήθειας"));
49    GREEK(fmt::format(L"\t{:<35} {}", L"--settingsαρχείο=[]", Λ
     Αρχείο" με τις παραμέτρους εκτέλεσης"));
50    GREEK(fmt::format(L"\t{:<35} {}", L"-o αρχείο[], --output
     αρχείο=[]", ΛΤο" όνομα των αρχείων εξόδου"));
51    GREEK(fmt::format(L"\t{:<35} {}", L"-r, --random-initial-node",
     ΛΟ" ος1 κόμβος που επισκέπτεται μια σταγόνα είναι τυχαίος"));
52    GREEK(fmt::format(L"\t{:<35} {}", L"--po, --print-settings", Λ
     Εμφανίζει" τις παραμέτρους εκτέλεσης πριν την εκτέλεση"));
53    GREEK(fmt::format(L"\t{:<35} {}", L"--no-files", ΛΔεν" δημιουργεί
     αρχεία εξόδου."));
54    return 0;
55  } catch(args::ParseError& e) {
56    std::cerr << e.what() << std::endl;
57    return 1;
58  } catch(args::ValidationError& e) {
59    std::cerr << e.what() << std::endl;
60    return 1;
61  }
62  iwd::Settings settings;
63
64  Αν// μας έχει δωθεί αρχείο με τις ρυθμίσεις, τις θέτουμε απο αυτό
65  if(settings_file) {
66    try {
67      settings = readJsonSettings(args::get(settings_file));
68    } catch(exception e) {
69      std::cerr << e.what() << endl;
70      settings = iwd::Settings();
71    }
72  }
73  if(output_file) {
74    settings.out_filename = args::get(output_file);
75  }
76
77  Αν// δεν μας έχει δωθεί αρχείο γράφου δεν συνεχίζουμε
```

```
78    if(!graph_file) {
79      GREEK(LTo" αρχείο του γράφου είναι απαραίτητο");
80      return 1;
81    }
82
83    Graph graph;
84
85    Θέτουμε// τις ρυθμίσεις βάση των ορισμάτων που μας έχουν δωθεί
86    if(iterations) {
87      settings.max_iterations = args::get(iterations);
88    }
89
90    try {
91      graph = graphFileReader(args::get(graph_file));
92    } catch(std::invalid_argument & e) {
93      std::cerr << e.what();
94      return 1;
95    }
96    /*
97    Θέτουμε το αρχικό seed που χρησιμοποιείται στην συνάρτηση
98    rand() και mt19937::gen() για την παραγωγή τυχαίων αριθμών.
99    Αν δεν μας έχει δωθεί κάτι ή μας δωθεί ο αριθμός 0, θέτουμε
100   τυχαίο αρχικό seed. Αυτό γίνεται για να μπορούν να αναπαραχθούν
101   πλήρως τα αποτελέσματα των πειραμάτων.
102   */
103   if(seed) {
104     settings.seed = args::get(seed);
105     srand(seed);
106   } else if(settings.seed != 0) {
107     srand(settings.seed);
108   } else {
109     srand(time(nullptr));
110     settings.seed = rand();
111   }
112
113   if(random_initial_node) {
114     settings.random_initial_node = true;
115   }
116
117   if(no_files) settings.no_files = no_files;
118
119   GREEK(fmt::format(LKόμβοι" : {}. Εκτέλεση για {} επαναλήψεις. Αρχικό
       seed : {}", graph.nodes.size(), settings.max_iterations, settings
       .seed));
120   if(print_settings) {
121     printSettings(settings);
122   }
123
124   //Debug μεταβλητές
125   auto best_iteration = 0;
126   auto last_iteration_change = 0; //unused
```

```
127   auto iterations_without_change = 0;
128
129   Stopwatch<chrono::milliseconds> all;
130
131   Αρχικοποίηση// μεταβλητών
132   auto graph_size = graph.nodes.size();
133   graph.initiateSoil(settings.initial_soil);
134   auto drops = vector<iwd::Drop>(graph.nodes.size(), iwd::Drop(graph
       , settings));
135
136   iwd::Drop* best_drop = nullptr; Δείκτης// προς την σταγόνα με την
       βέλτιστη λύση
137   Ορίζουμε// το αρχικό μήκος βέλτιστης διαδρομής στο άπειρο
138   auto best_tour_length = std::numeric_limits<double>::infinity();
139
140   Μεταβλητή// που κρατάει τους κόμβους της βέλτιστης διαδρομής
141   iwd::collection best_tour;
142   std::ostringstream pdataΜεταβητή;// που κρατάει τα plot data
143   for(auto iteration = 0u; iteration < settings.max_iterations ;
       iteration++) {
144     Stopwatch<chrono::milliseconds> it;
145     Κρατάμε// μία σταγόνα για κάθε node
146     for(auto& drop : drops) {
147       auto drop_start = Stopwatch<>();
148
149       drop.travel();
150
151       drop_start.stop();
152       assert(drop.visited_nodes.size() == drop.graph.nodes.size()+1)
      ;
153       Βρίσκουμε// την σταγόνα με την καλύτερημικρότερη() διαδρομή
154       if(drop.path_length < best_tour_length) {
155         best_drop = &drop;
156         best_tour_length = drop.path_length;
157         best_iteration = iteration;
158         best_tour = drop.visited_nodes;
159         iterations_without_change = 0;
160       } else {
161         ++iterations_without_change;
162       }
163     }
164     pdata << iteration << " " << best_tour_length << std::endl;
165     Έυρεση// της τοπικής βέλτιστης διαδρομής
166     auto Nib = graph_size;
167     auto p = settings.p;
168     Ανανεώνουμε// το soil στην τοπική βέλτιστη διαδρομή
169     for(unsigned id = 1; id  < graph_size ; id ++) {
170       auto n1 = *best_drop->visited_nodes[id - 1], n2 = *best_drop->
      visited_nodes[id];
171       auto prev_soil = best_drop->graph.getSoil(n1,n2);
```

```
172       soil new_soil = (1 + p) * prev_soil - (p * (1 / (Nib - 1))*
     best_drop->soil);
173       graph.setSoil(n1,n2,new_soil);
174      }
175
176      it.stop();
177    }
178
179    Εμφάνιση// αποτελεσμάτων
180    GREEK(fmt::format(LΟλική" βέλτιστη διαδρομή : {} μετά απο {}
     επαναλήψεις", best_tour_length, best_iteration));
181    GREEK(fmt::format(L"{:=^80}", LΒΕΛΤΙΣΤΗ" ΔΙΑΔΡΟΜΗ"));
182    iwd::Drop::printPath(best_tour);
183    std::cout << std::string(80,'=') << std::endl;
184
185    all.stop();
186    if(!settings.no_files) {
187      GREEK(LΔημιουργία" αρχείων...")
188      Γράφουμε// το αρχείο με τα δεδομένα του γραφήματος και το κλείνουμε
189      std::ofstream f_pdata(settings.out_filename + ".pdat");
190      f_pdata << pdata.str();
191      f_pdata.close();
192
193      Εξαγωγή// του γράφου σε neato format
194      // graph G{
195      //  n0[pos = "0,0!"];
196      //    n1[pos = "2,0!"];
197      //    n0 -- n1 -- n0;
198      // }
199      std::ostringstream neato;
200      std::ostringstream neato_node_connection;
201      neato << "graph G {" << std::endl;
202      neato << "node [style=filled,color=black];" << std::endl;
203      for(auto * node : best_tour) {
204        neato << "n" << node->id << "[pos = \"" << node->x << "," <<
     node->y << "!\"];" << std::endl;
205        neato_node_connection << "n" << node->id << " -- ";
206      }
207      neato_node_connection << "n" << best_tour[0]->id << "[color=red,
     penwidth=3.0];" << std::endl;
208      neato << neato_node_connection.str();
209      neato << "}";
210      Γράφουμε// τα δεδομένα για το πρόγραμμα neato
211      std::ofstream best_tour_file(settings.out_filename + ".ndat");
212      best_tour_file << neato.str();
213      best_tour_file.close();
214    }
215
216    GREEK(L"\t ΤΕΛΟΣ ΣΕ " << all.elapsed() << " ms");
217    system("pause");
218    return 0;
```

```
219 }
```

13.3 Πηγαίος κώδικας `iwd-gui`

Main.h

```
1
2  #pragma once
3  #include <string>
4  #include <msclr\marshal_cppstd.h>
5  #include <exception>
6
7  #include "IWD/Drop.h"
8  #include "IWD/Settings.hpp"
9  #include "Utilities.hpp"
10 #include "ProgressForm.h"
11
12 #include "IwdHelpClasses.h"
13
14
15
16 namespace iwdgui {
17
18   using namespace System;
19   using namespace System::Collections::Generic;
20   using namespace System::ComponentModel;
21   using namespace System::Collections;
22   using namespace System::Windows::Forms;
23   using namespace System::Data;
24   using namespace System::Drawing;
25   using namespace msclr::interop;
26
27   /// <summary>
28   /// Summary for Main
29   /// </summary>
30   public ref class Main : public System::Windows::Forms::Form {
31   public:
32     Main(void) {
33       InitializeComponent();
34       this->resetSettings();
35       windows = gcnew List<ProgressForm^>();
36     }
37     void resetSettings();
38     iwd::Settings settingsFromInput();
39
40     Λίστα// με όλα τα παράθυρα που εκτελούν εργασία
41   private: List<ProgressForm^>^ windows;
42
43   protected:
44     /// <summary>
45     /// Clean up any resources being used.
46     /// </summary>
47     ~Main() {
48       if(components) {
```

```
49          delete components;
50        }
51      }
52
53    private: System::Windows::Forms::GroupBox^  groupBox3;
54    private: System::Windows::Forms::GroupBox^  groupBox2;
55    private: System::Windows::Forms::CheckBox^  var_no_files;
56    private: System::Windows::Forms::CheckBox^
       var_random_initial_node;
57    private: System::Windows::Forms::TextBox^  var_max_iterations;
58    private: System::Windows::Forms::Label^  label14;
59    private: System::Windows::Forms::TextBox^  var_output_filename;
60    private: System::Windows::Forms::Label^  label12;
61    private: System::Windows::Forms::GroupBox^  groupBox1;
62    private: System::Windows::Forms::TextBox^  var_initial_velocity;
63    private: System::Windows::Forms::Label^  label11;
64    private: System::Windows::Forms::TextBox^  var_initial_soil;
65    private: System::Windows::Forms::Label^  label9;
66    private: System::Windows::Forms::TextBox^  var_e;
67    private: System::Windows::Forms::Label^  label5;
68    private: System::Windows::Forms::TextBox^  var_p;
69    private: System::Windows::Forms::Label^  label6;
70    private: System::Windows::Forms::TextBox^  var_cv;
71    private: System::Windows::Forms::Label^  label7;
72    private: System::Windows::Forms::TextBox^  var_cs;
73    private: System::Windows::Forms::Label^  label8;
74    private: System::Windows::Forms::TextBox^  var_bv;
75    private: System::Windows::Forms::Label^  label3;
76    private: System::Windows::Forms::TextBox^  var_bs;
77    private: System::Windows::Forms::Label^  label4;
78    private: System::Windows::Forms::TextBox^  var_av;
79    private: System::Windows::Forms::Label^  label2;
80    private: System::Windows::Forms::TextBox^  var_as;
81    private: System::Windows::Forms::Label^  label1;
82    private: System::Windows::Forms::Button^  btn_reset;
83    private: System::Windows::Forms::OpenFileDialog^  graphOpenDialog;
84    private: System::Windows::Forms::Button^  btn_open_file;
85    private: System::Windows::Forms::TextBox^  graphFileInput;
86    private: System::Windows::Forms::Label^  label10;
87    private: System::Windows::Forms::Button^  btn_run;
88    protected:
89    private:
90    /// <summary>
91    /// Required designer variable.
92    /// </summary>
93    System::ComponentModel::IContainer^  components;
94
95
96  #pragma region Windows Form Designer generated code
97      /// <summary>
98      /// Required method for Designer support - do not modify
```

```
99      /// the contents of this method with the code editor.
100     /// </summary>
101     void InitializeComponent(void) {
102       System::ComponentModel::ComponentResourceManager^  resources =
        (gcnew System::ComponentModel::ComponentResourceManager(Main::
        typeid));
103       this->graphOpenDialog = (gcnew System::Windows::Forms::
        OpenFileDialog());
104       this->btn_open_file = (gcnew System::Windows::Forms::Button())
        ;
105       this->graphFileInput = (gcnew System::Windows::Forms::TextBox
        ());
106       this->label10 = (gcnew System::Windows::Forms::Label());
107       this->btn_run = (gcnew System::Windows::Forms::Button());
108       this->groupBox3 = (gcnew System::Windows::Forms::GroupBox());
109       this->groupBox2 = (gcnew System::Windows::Forms::GroupBox());
110       this->var_no_files = (gcnew System::Windows::Forms::CheckBox()
        );
111       this->var_random_initial_node = (gcnew System::Windows::Forms
        ::CheckBox());
112       this->var_max_iterations = (gcnew System::Windows::Forms::
        TextBox());
113       this->label14 = (gcnew System::Windows::Forms::Label());
114       this->var_output_filename = (gcnew System::Windows::Forms::
        TextBox());
115       this->label12 = (gcnew System::Windows::Forms::Label());
116       this->groupBox1 = (gcnew System::Windows::Forms::GroupBox());
117       this->var_initial_velocity = (gcnew System::Windows::Forms::
        TextBox());
118       this->label11 = (gcnew System::Windows::Forms::Label());
119       this->var_initial_soil = (gcnew System::Windows::Forms::
        TextBox());
120       this->label9 = (gcnew System::Windows::Forms::Label());
121       this->var_e = (gcnew System::Windows::Forms::TextBox());
122       this->label5 = (gcnew System::Windows::Forms::Label());
123       this->var_p = (gcnew System::Windows::Forms::TextBox());
124       this->label6 = (gcnew System::Windows::Forms::Label());
125       this->var_cv = (gcnew System::Windows::Forms::TextBox());
126       this->label7 = (gcnew System::Windows::Forms::Label());
127       this->var_cs = (gcnew System::Windows::Forms::TextBox());
128       this->label8 = (gcnew System::Windows::Forms::Label());
129       this->var_bv = (gcnew System::Windows::Forms::TextBox());
130       this->label3 = (gcnew System::Windows::Forms::Label());
131       this->var_bs = (gcnew System::Windows::Forms::TextBox());
132       this->label4 = (gcnew System::Windows::Forms::Label());
133       this->var_av = (gcnew System::Windows::Forms::TextBox());
134       this->label2 = (gcnew System::Windows::Forms::Label());
135       this->var_as = (gcnew System::Windows::Forms::TextBox());
136       this->label1 = (gcnew System::Windows::Forms::Label());
137       this->btn_reset = (gcnew System::Windows::Forms::Button());
138       this->groupBox3->SuspendLayout();
```

```
139         this->groupBox2->SuspendLayout();
140         this->groupBox1->SuspendLayout();
141         this->SuspendLayout();
142         //
143         // graphOpenDialog
144         //
145         this->graphOpenDialog->AddExtension = false;
146         this->graphOpenDialog->FileOk += gcnew System::ComponentModel
    ::CancelEventHandler(this, &Main::graphOpenDialog_FileOk);
147         //
148         // btn_open_file
149         //
150         this->btn_open_file->Location = System::Drawing::Point(405,
    27);
151         this->btn_open_file->Name = L"btn_open_file";
152         this->btn_open_file->Size = System::Drawing::Size(75, 23);
153         this->btn_open_file->TabIndex = 17;
154         this->btn_open_file->Text = LΆνοιγμα"";
155         this->btn_open_file->UseVisualStyleBackColor = false;
156         this->btn_open_file->Click += gcnew System::EventHandler(this,
     &Main::btn_open_file_Click);
157         //
158         // graphFileInput
159         //
160         this->graphFileInput->Location = System::Drawing::Point(16,
    29);
161         this->graphFileInput->Name = L"graphFileInput";
162         this->graphFileInput->Size = System::Drawing::Size(383, 20);
163         this->graphFileInput->TabIndex = 0;
164         this->graphFileInput->DoubleClick += gcnew System::
    EventHandler(this, &Main::graphFileInput_DoubleClick);
165         //
166         // label10
167         //
168         this->label10->AutoSize = true;
169         this->label10->Location = System::Drawing::Point(16, 13);
170         this->label10->Name = L"label10";
171         this->label10->Size = System::Drawing::Size(83, 13);
172         this->label10->TabIndex = 18;
173         this->label10->Text = LΑρχείο" γράφου";
174         //
175         // btn_run
176         //
177         this->btn_run->Dock = System::Windows::Forms::DockStyle::
    Bottom;
178         this->btn_run->Font = (gcnew System::Drawing::Font(L"Microsoft
     Sans Serif", 10, System::Drawing::FontStyle::Regular, System::
    Drawing::GraphicsUnit::Point,
179             static_cast<System::Byte>(204)));
180         this->btn_run->Image = (cli::safe_cast<System::Drawing::Image
    ^>(resources->GetObject(L"btn_run.Image")));
```

```
181         this->btn_run->ImageAlign = System::Drawing::ContentAlignment
    ::MiddleLeft;
182         this->btn_run->Location = System::Drawing::Point(0, 315);
183         this->btn_run->Name = L"btn_run";
184         this->btn_run->Size = System::Drawing::Size(493, 32);
185         this->btn_run->TabIndex = 20;
186         this->btn_run->Text = LΕκτέλεση"";
187         this->btn_run->TextAlign = System::Drawing::ContentAlignment::
    MiddleRight;
188         this->btn_run->TextImageRelation = System::Windows::Forms::
    TextImageRelation::TextBeforeImage;
189         this->btn_run->UseVisualStyleBackColor = true;
190         this->btn_run->Click += gcnew System::EventHandler(this, &Main
    ::btn_run_Click);
191         //
192         // groupBox3
193         //
194         this->groupBox3->Controls->Add(this->groupBox2);
195         this->groupBox3->Controls->Add(this->groupBox1);
196         this->groupBox3->Controls->Add(this->btn_reset);
197         this->groupBox3->Font = (gcnew System::Drawing::Font(L"
    Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular,
    System::Drawing::GraphicsUnit::Point,
198           static_cast<System::Byte>(204)));
199         this->groupBox3->Location = System::Drawing::Point(12, 58);
200         this->groupBox3->Name = L"groupBox3";
201         this->groupBox3->Size = System::Drawing::Size(468, 227);
202         this->groupBox3->TabIndex = 21;
203         this->groupBox3->TabStop = false;
204         this->groupBox3->Text = LΕπιλογές" εκτέλεσης αλγορίθμου";
205         //
206         // groupBox2
207         //
208         this->groupBox2->Controls->Add(this->var_no_files);
209         this->groupBox2->Controls->Add(this->var_random_initial_node);
210         this->groupBox2->Controls->Add(this->var_max_iterations);
211         this->groupBox2->Controls->Add(this->label14);
212         this->groupBox2->Controls->Add(this->var_output_filename);
213         this->groupBox2->Controls->Add(this->label12);
214         this->groupBox2->Dock = System::Windows::Forms::DockStyle::
    Right;
215         this->groupBox2->Font = (gcnew System::Drawing::Font(L"
    Microsoft Sans Serif", 8.25F, System::Drawing::FontStyle::Regular
    , System::Drawing::GraphicsUnit::Point,
216           static_cast<System::Byte>(204)));
217         this->groupBox2->Location = System::Drawing::Point(234, 22);
218         this->groupBox2->Name = L"groupBox2";
219         this->groupBox2->Size = System::Drawing::Size(231, 179);
220         this->groupBox2->TabIndex = 21;
221         this->groupBox2->TabStop = false;
222         this->groupBox2->Text = LΛοιπές" παράμετροι && ρυθμίσεις";
```

```
223         //
224         // var_no_files
225         //
226         this->var_no_files->AutoSize = true;
227         this->var_no_files->Location = System::Drawing::Point(19, 100)
    ;
228         this->var_no_files->Name = L"var_no_files";
229         this->var_no_files->Size = System::Drawing::Size(127, 17);
230         this->var_no_files->TabIndex = 26;
231         this->var_no_files->Text = LΔημιουργία" αρχείων";
232         this->var_no_files->UseVisualStyleBackColor = true;
233         //
234         // var_random_initial_node
235         //
236         this->var_random_initial_node->AutoSize = true;
237         this->var_random_initial_node->Location = System::Drawing::
    Point(19, 77);
238         this->var_random_initial_node->Name = L"
    var_random_initial_node";
239         this->var_random_initial_node->Size = System::Drawing::Size
    (146, 17);
240         this->var_random_initial_node->TabIndex = 25;
241         this->var_random_initial_node->Text = LΤυχαίος" αρχικός κόμβος";
242         this->var_random_initial_node->UseVisualStyleBackColor = true;
243         //
244         // var_max_iterations
245         //
246         this->var_max_iterations->Location = System::Drawing::Point
    (147, 45);
247         this->var_max_iterations->Name = L"var_max_iterations";
248         this->var_max_iterations->Size = System::Drawing::Size(75, 20)
    ;
249         this->var_max_iterations->TabIndex = 24;
250         //
251         // label14
252         //
253         this->label14->AutoSize = true;
254         this->label14->Location = System::Drawing::Point(16, 48);
255         this->label14->Name = L"label14";
256         this->label14->Size = System::Drawing::Size(91, 13);
257         this->label14->TabIndex = 23;
258         this->label14->Text = LΓενιές" (iterations)";
259         //
260         // var_output_filename
261         //
262         this->var_output_filename->Location = System::Drawing::Point
    (147, 19);
263         this->var_output_filename->Name = L"var_output_filename";
264         this->var_output_filename->Size = System::Drawing::Size(75,
    20);
265         this->var_output_filename->TabIndex = 20;
```

```
266         //
267         // label12
268         //
269         this->label12->AutoSize = true;
270         this->label12->Location = System::Drawing::Point(16, 22);
271         this->label12->Name = L"label12";
272         this->label12->Size = System::Drawing::Size(124, 13);
273         this->label12->TabIndex = 19;
274         this->label12->Text = LΌνομα" αρχείων εξόδου";
275         //
276         // groupBox1
277         //
278         this->groupBox1->Controls->Add(this->var_initial_velocity);
279         this->groupBox1->Controls->Add(this->label11);
280         this->groupBox1->Controls->Add(this->var_initial_soil);
281         this->groupBox1->Controls->Add(this->label9);
282         this->groupBox1->Controls->Add(this->var_e);
283         this->groupBox1->Controls->Add(this->label5);
284         this->groupBox1->Controls->Add(this->var_p);
285         this->groupBox1->Controls->Add(this->label6);
286         this->groupBox1->Controls->Add(this->var_cv);
287         this->groupBox1->Controls->Add(this->label7);
288         this->groupBox1->Controls->Add(this->var_cs);
289         this->groupBox1->Controls->Add(this->label8);
290         this->groupBox1->Controls->Add(this->var_bv);
291         this->groupBox1->Controls->Add(this->label3);
292         this->groupBox1->Controls->Add(this->var_bs);
293         this->groupBox1->Controls->Add(this->label4);
294         this->groupBox1->Controls->Add(this->var_av);
295         this->groupBox1->Controls->Add(this->label2);
296         this->groupBox1->Controls->Add(this->var_as);
297         this->groupBox1->Controls->Add(this->label1);
298         this->groupBox1->Dock = System::Windows::Forms::DockStyle::
    Left;
299         this->groupBox1->Font = (gcnew System::Drawing::Font(L"
    Microsoft Sans Serif", 8.25F, System::Drawing::FontStyle::Regular
    , System::Drawing::GraphicsUnit::Point,
300           static_cast<System::Byte>(204)));
301         this->groupBox1->Location = System::Drawing::Point(3, 22);
302         this->groupBox1->Name = L"groupBox1";
303         this->groupBox1->Size = System::Drawing::Size(231, 179);
304         this->groupBox1->TabIndex = 20;
305         this->groupBox1->TabStop = false;
306         this->groupBox1->Text = LΠαράμετροι" μεταβολής IWD";
307         //
308         // var_initial_velocity
309         //
310         this->var_initial_velocity->Location = System::Drawing::Point
    (147, 151);
311         this->var_initial_velocity->Name = L"var_initial_velocity";
```

```
312         this->var_initial_velocity->Size = System::Drawing::Size(75,
        20);
313         this->var_initial_velocity->TabIndex = 20;
314         //
315         // label11
316         //
317         this->label11->AutoSize = true;
318         this->label11->Location = System::Drawing::Point(16, 154);
319         this->label11->Name = L"label11";
320         this->label11->Size = System::Drawing::Size(117, 13);
321         this->label11->TabIndex = 19;
322         this->label11->Text = LAρχικό" velocity (InitVel)";
323         //
324         // var_initial_soil
325         //
326         this->var_initial_soil->Location = System::Drawing::Point(147,
         126);
327         this->var_initial_soil->Name = L"var_initial_soil";
328         this->var_initial_soil->Size = System::Drawing::Size(75, 20);
329         this->var_initial_soil->TabIndex = 18;
330         //
331         // label9
332         //
333         this->label9->AutoSize = true;
334         this->label9->Location = System::Drawing::Point(16, 129);
335         this->label9->Name = L"label9";
336         this->label9->Size = System::Drawing::Size(98, 13);
337         this->label9->TabIndex = 16;
338         this->label9->Text = LAρχικό" soil (InitSoil)";
339         //
340         // var_e
341         //
342         this->var_e->Location = System::Drawing::Point(147, 97);
343         this->var_e->Name = L"var_e";
344         this->var_e->Size = System::Drawing::Size(75, 20);
345         this->var_e->TabIndex = 15;
346         //
347         // label5
348         //
349         this->label5->AutoSize = true;
350         this->label5->Location = System::Drawing::Point(122, 100);
351         this->label5->Name = L"label5";
352         this->label5->Size = System::Drawing::Size(13, 13);
353         this->label5->TabIndex = 14;
354         this->label5->Text = Lε"";
355         //
356         // var_p
357         //
358         this->var_p->Location = System::Drawing::Point(41, 97);
359         this->var_p->Name = L"var_p";
360         this->var_p->Size = System::Drawing::Size(73, 20);
```

```
        this->var_p->TabIndex = 13;
        //
        // label6
        //
        this->label6->AutoSize = true;
        this->label6->Location = System::Drawing::Point(17, 100);
        this->label6->Name = L"label6";
        this->label6->Size = System::Drawing::Size(13, 13);
        this->label6->TabIndex = 12;
        this->label6->Text = Lρ"";
        //
        // var_cv
        //
        this->var_cv->Location = System::Drawing::Point(147, 71);
        this->var_cv->Name = L"var_cv";
        this->var_cv->Size = System::Drawing::Size(75, 20);
        this->var_cv->TabIndex = 11;
        //
        // label7
        //
        this->label7->AutoSize = true;
        this->label7->Location = System::Drawing::Point(122, 74);
        this->label7->Name = L"label7";
        this->label7->Size = System::Drawing::Size(19, 13);
        this->label7->TabIndex = 10;
        this->label7->Text = L"cv";
        //
        // var_cs
        //
        this->var_cs->Location = System::Drawing::Point(41, 71);
        this->var_cs->Name = L"var_cs";
        this->var_cs->Size = System::Drawing::Size(73, 20);
        this->var_cs->TabIndex = 9;
        //
        // label8
        //
        this->label8->AutoSize = true;
        this->label8->Location = System::Drawing::Point(17, 74);
        this->label8->Name = L"label8";
        this->label8->Size = System::Drawing::Size(18, 13);
        this->label8->TabIndex = 8;
        this->label8->Text = L"cs";
        //
        // var_bv
        //
        this->var_bv->Location = System::Drawing::Point(147, 45);
        this->var_bv->Name = L"var_bv";
        this->var_bv->Size = System::Drawing::Size(75, 20);
        this->var_bv->TabIndex = 7;
        //
        // label3
```

```
412         //
413         this->label3->AutoSize = true;
414         this->label3->Location = System::Drawing::Point(122, 48);
415         this->label3->Name = L"label3";
416         this->label3->Size = System::Drawing::Size(19, 13);
417         this->label3->TabIndex = 6;
418         this->label3->Text = L"bv";
419         //
420         // var_bs
421         //
422         this->var_bs->Location = System::Drawing::Point(41, 45);
423         this->var_bs->Name = L"var_bs";
424         this->var_bs->Size = System::Drawing::Size(73, 20);
425         this->var_bs->TabIndex = 5;
426         //
427         // label4
428         //
429         this->label4->AutoSize = true;
430         this->label4->Location = System::Drawing::Point(17, 48);
431         this->label4->Name = L"label4";
432         this->label4->Size = System::Drawing::Size(18, 13);
433         this->label4->TabIndex = 4;
434         this->label4->Text = L"bs";
435         //
436         // var_av
437         //
438         this->var_av->Location = System::Drawing::Point(147, 19);
439         this->var_av->Name = L"var_av";
440         this->var_av->Size = System::Drawing::Size(75, 20);
441         this->var_av->TabIndex = 3;
442         //
443         // label2
444         //
445         this->label2->AutoSize = true;
446         this->label2->Location = System::Drawing::Point(122, 22);
447         this->label2->Name = L"label2";
448         this->label2->Size = System::Drawing::Size(19, 13);
449         this->label2->TabIndex = 2;
450         this->label2->Text = L"av";
451         //
452         // var_as
453         //
454         this->var_as->Location = System::Drawing::Point(41, 19);
455         this->var_as->Name = L"var_as";
456         this->var_as->Size = System::Drawing::Size(73, 20);
457         this->var_as->TabIndex = 1;
458         //
459         // label1
460         //
461         this->label1->AutoSize = true;
462         this->label1->Location = System::Drawing::Point(17, 22);
```

```
463         this->label1->Name = L"label1";
464         this->label1->Size = System::Drawing::Size(18, 13);
465         this->label1->TabIndex = 0;
466         this->label1->Text = L"as";
467         //
468         // btn_reset
469         //
470         this->btn_reset->Dock = System::Windows::Forms::DockStyle::
    Bottom;
471         this->btn_reset->Font = (gcnew System::Drawing::Font(L"
    Microsoft Sans Serif", 8.25F, System::Drawing::FontStyle::Regular
    , System::Drawing::GraphicsUnit::Point,
472           static_cast<System::Byte>(204)));
473         this->btn_reset->Location = System::Drawing::Point(3, 201);
474         this->btn_reset->Name = L"btn_reset";
475         this->btn_reset->Size = System::Drawing::Size(462, 23);
476         this->btn_reset->TabIndex = 16;
477         this->btn_reset->Text = LEπαναφορά"";
478         this->btn_reset->UseVisualStyleBackColor = true;
479         this->btn_reset->Click += gcnew System::EventHandler(this, &
    Main::btn_reset_Click);
480         //
481         // Main
482         //
483         this->AutoSizeMode = System::Windows::Forms::AutoSizeMode::
    GrowAndShrink;
484         this->ClientSize = System::Drawing::Size(493, 347);
485         this->Controls->Add(this->groupBox3);
486         this->Controls->Add(this->btn_run);
487         this->Controls->Add(this->label10);
488         this->Controls->Add(this->graphFileInput);
489         this->Controls->Add(this->btn_open_file);
490         this->FormBorderStyle = System::Windows::Forms::
    FormBorderStyle::FixedSingle;
491         this->Icon = (cli::safe_cast<System::Drawing::Icon^>(resources
    ->GetObject(L"$this.Icon")));
492         this->MaximizeBox = false;
493         this->Name = L"Main";
494         this->Text = LEφαργμογή" IWD GUI :: TEI Κεντρικής Μακεδονίας -
    2017";
495         this->groupBox3->ResumeLayout(false);
496         this->groupBox2->ResumeLayout(false);
497         this->groupBox2->PerformLayout();
498         this->groupBox1->ResumeLayout(false);
499         this->groupBox1->PerformLayout();
500         this->ResumeLayout(false);
501         this->PerformLayout();
502
503      }
504 #pragma endregion
505
```

```
506   private: System::Void btn_open_file_Click(System::Object^  sender,
        System::EventArgs^  e) {
507     graphOpenDialog->ShowDialog();
508   }
509
510   private: System::Void graphOpenDialog_FileOk(System::Object^
       sender, System::ComponentModel::CancelEventArgs^  e) {
511     graphFileInput->Text = graphOpenDialog->FileName;
512   }
513
514   private: System::Void btn_run_Click(System::Object^  sender,
       System::EventArgs^  e) {
515     Ελέγχουμε// αν έχει δοθεί όνομα αρχείου γράφου...
516     if(String::IsNullOrEmpty(graphFileInput->Text)) {
517       αν//... όχι, εμφανίζουμε τον διάλογο επιλογής αρχείου γράφου.
518       graphOpenDialog->ShowDialog();
519     } else {
520       auto filepath = ToStdString(graphFileInput->Text);
521       try {
522         auto graph = graphFileReader(filepath);
523         auto settings = settingsFromInput();
524         auto p = gcnew ProgressForm(graphFileInput->Text, settings);
525         windows->Add(p);
526       } catch(std::invalid_argument e) {
527         MessageBox::Show(LΠρόβλημα" κατά την ανάγνωση του αρχείου
       γράφου.", LΣφάλμα"", MessageBoxButtons::OK, MessageBoxIcon::Error)
       ;
528         return;
529       } catch(Exception^ ex) {
530         MessageBox::Show(LΛανθασμένοι" παράμετροι εκτέλεσης. Παρακαλώ
       ελέγξτε τις τιμές που εισάγατε.", LΠροσοχή"", MessageBoxButtons::OK,
        MessageBoxIcon::Information);
531       }
532     }
533   }
534
535   private: System::Void btn_reset_Click(System::Object^  sender,
       System::EventArgs^  e) {
536     this->resetSettings();
537   }
538
539   private: System::Void graphFileInput_DoubleClick(System::Object^
       sender, System::EventArgs^  e) {
540     if(String::IsNullOrWhiteSpace(graphFileInput->Text)) {
541       graphOpenDialog->ShowDialog();
542     }
543   }
544 };
545 }
```

```
1
2  #include "Main.h"
3
4  using namespace System;
5  using namespace System::Windows::Forms;
6
7  [STAThread]
8  void Main(array<String^>^ args) {
9    Application::EnableVisualStyles();
10   Application::SetCompatibleTextRenderingDefault(false);
11
12   iwdgui::Main form;
13   Application::Run(%form);
14 }
15
16 void iwdgui::Main::resetSettings() {
17   iwd::Settings ds; // Default Settings
18   var_av->Text = Convert::ToString(ds.av);
19   var_as->Text = Convert::ToString(ds.as);
20   var_bv->Text = Convert::ToString(ds.bv);
21   var_bs->Text = Convert::ToString(ds.bs);
22   var_cv->Text = Convert::ToString(ds.cv);
23   var_cs->Text = Convert::ToString(ds.cs);
24   var_p->Text = Convert::ToString(ds.p);
25   var_e->Text = Convert::ToString(ds.e);
26   var_initial_soil->Text = Convert::ToString(ds.initial_soil);
27   var_initial_velocity->Text = Convert::ToString(ds.initial_velocity
      );
28   var_no_files->Checked = !ds.no_files;
29   var_random_initial_node->Checked = ds.random_initial_node;
30   var_max_iterations->Text = Convert::ToString(ds.max_iterations);
31   var_output_filename->Text = gcnew String(ds.out_filename.c_str());
32 }
33
34 iwd::Settings iwdgui::Main::settingsFromInput() {
35   iwd::Settings s;
36   try {
37     s.as = Convert::ToDouble(var_as->Text);
38     s.av = Convert::ToDouble(var_av->Text);
39     s.bs = Convert::ToDouble(var_bs->Text);
40     s.bv = Convert::ToDouble(var_bv->Text);
41     s.cs = Convert::ToDouble(var_cs->Text);
42     s.cv = Convert::ToDouble(var_cv->Text);
43     s.e = Convert::ToDouble(var_e->Text);
44     s.p = Convert::ToDouble(var_p->Text);
45     s.random_initial_node = var_random_initial_node->Checked;
46     s.no_files = !var_no_files->Checked;
47     s.initial_soil = Convert::ToDouble(var_initial_soil->Text);
48     s.initial_velocity = Convert::ToDouble(var_initial_velocity->
      Text);
```

```
49      s.out_filename = marshal_as<std::string>(var_output_filename->
        Text);
50      s.max_iterations = Convert::ToUInt32(var_max_iterations->Text);
51    } catch(Exception^ e) {
52      throw e;
53    }
54    return s;
55  }
```

ProgressForm.h

```
1
2   #pragma once
3   #include <string>
4   #include <msclr\marshal_cppstd.h>
5
6   #include "IWD/Drop.h"
7   #include "IWD/Settings.hpp"
8   #include "Utilities.hpp"
9   #include "IwdHelpClasses.h"
10
11  #define ToNetString(x) Convert::ToString(x)
12
13  namespace iwdgui {
14
15    using namespace System;
16    using namespace System::ComponentModel;
17    using namespace System::Collections;
18    using namespace System::Windows::Forms;
19    using namespace System::Data;
20    using namespace System::Drawing;
21
22    /// <summary>
23    /// Φόρμα προόδου εκτέλεσης μιας σταγόνας
24    /// </summary>
25    public ref class ProgressForm : public System::Windows::Forms::
        Form
26    {
27    private:
28      bool closeWhenFinished = false;
29      DateTime opStart;
30    public: ProgressForm(String^ filepath, iwd::Settings& s) {
31        InitializeComponent();
32        // Νέο work state
33        auto workState = gcnew IwdWorkerState();
34        workState->filepath = filepath;
35        workState->settings = gcnew iwdgui::Settings(s);
36        workState->status = LΔημιουργία" εργασίας";
37        workState->max_iterations = workState->settings->
          max_iterations;
38
39        Αρχικοποίηση// δεδομένων παραθύρου
```

```
40        this->lblCurrentIteration->Text = "0 / " + workState->settings
      ->max_iterations;
41        this->lblStatus->Text = LΕκκίνηση"";
42        this->lblPathLength->Text = Convert::ToString(workState->
      pathLength);
43        String^ filename = System::IO::Path::GetFileName(filepath);
44        this->Text = filename + L" - Πρόοδος";
45        lbl_av->Text += Convert::ToString(workState->settings->av);
46        lbl_as->Text += Convert::ToString(workState->settings->as);
47        lbl_bv->Text += Convert::ToString(workState->settings->bv);
48        lbl_bs->Text += Convert::ToString(workState->settings->bs);
49        lbl_cv->Text += Convert::ToString(workState->settings->cv);
50        lbl_cs->Text += Convert::ToString(workState->settings->cs);
51        lbl_p->Text += Convert::ToString(workState->settings->p);
52        lbl_e->Text += Convert::ToString(workState->settings->e);
53        lbl_initial_soil->Text += Convert::ToString(workState->
      settings->initial_soil);
54        lbl_initial_velocity->Text += Convert::ToString(workState->
      settings->initial_velocity);
55        lbl_no_files->Visible = !workState->settings->no_files;
56        lbl_output_filename->Visible = !workState->settings->no_files;
57        lbl_random_initial_node->Visible = workState->settings->
      random_initial_node;
58        lbl_output_filename->Text += workState->settings->out_filename
      ;
59
60        opStart = DateTime::Now;
61        iwdTimer->Start();
62
63        Εμφάνιση// παραθύρου
64        this->Show();
65        iwdWorker->RunWorkerAsync(workState);
66      };
67
68    protected:
69      /// <summary>
70      /// Clean up any resources being used.
71      /// </summary>
72      ~ProgressForm()
73      {
74        if (components)
75        {
76          delete components;
77        }
78      }
79    protected:
80    private: System::ComponentModel::BackgroundWorker^  iwdWorker;
81    private: System::Windows::Forms::Label^  label3;
82    private: System::Windows::Forms::Label^  lblStatus;
83    private: System::Windows::Forms::GroupBox^  groupBox1;
84    private: System::Windows::Forms::RichTextBox^  dropPath;
```

```
85
86    private: System::Windows::Forms::Label^  lbl_initial_velocity;
87    private: System::Windows::Forms::Label^  lbl_initial_soil;
88    private: System::Windows::Forms::Label^  lbl_e;
89    private: System::Windows::Forms::Label^  lbl_p;
90    private: System::Windows::Forms::Label^  lbl_cv;
91    private: System::Windows::Forms::Label^  lbl_cs;
92    private: System::Windows::Forms::Label^  lbl_bv;
93    private: System::Windows::Forms::Label^  lbl_bs;
94    private: System::Windows::Forms::Label^  lbl_av;
95    private: System::Windows::Forms::Label^  lbl_as;
96    private: System::Windows::Forms::Label^  lbl_output_filename;
97    private: System::Windows::Forms::Label^  lbl_no_files;
98    private: System::Windows::Forms::Label^  lbl_random_initial_node;
99    private: System::Windows::Forms::Label^  label4;
100   private: System::Windows::Forms::StatusStrip^  iwdStatusStrip;
101   private: System::Windows::Forms::ToolStripProgressBar^
       iwdWorkProgressBar;
102   private: System::Windows::Forms::ToolStripStatusLabel^
       lblCurrentIteration;
103   private: System::Windows::Forms::Label^  lblNodes;
104   private: System::Windows::Forms::GroupBox^  groupBox2;
105   private: System::Windows::Forms::Label^  lblPathLength;
106   private: System::Windows::Forms::Label^  label1;
107   private: System::Windows::Forms::Label^  lblTimer;
108   private: System::Windows::Forms::Button^  btnCancel;
109   private: System::Windows::Forms::Timer^  iwdTimer;
110
111   private: System::ComponentModel::IContainer^  components;
112
113
114   protected:
115
116   private:
117     /// <summary>
118     /// Required designer variable.
119     /// </summary>
120
121
122 #pragma region Windows Form Designer generated code
123     /// <summary>
124     /// Required method for Designer support - do not modify
125     /// the contents of this method with the code editor.
126     /// </summary>
127     void InitializeComponent(void)
128     {
129       this->components = (gcnew System::ComponentModel::Container())
      ;
130       this->btnCancel = (gcnew System::Windows::Forms::Button());
131       this->iwdWorker = (gcnew System::ComponentModel::
      BackgroundWorker());
```

```
132         this->label3 = (gcnew System::Windows::Forms::Label());
133         this->lblStatus = (gcnew System::Windows::Forms::Label());
134         this->groupBox1 = (gcnew System::Windows::Forms::GroupBox());
135         this->lblNodes = (gcnew System::Windows::Forms::Label());
136         this->lbl_random_initial_node = (gcnew System::Windows::Forms
        ::Label());
137         this->lbl_output_filename = (gcnew System::Windows::Forms::
        Label());
138         this->lbl_no_files = (gcnew System::Windows::Forms::Label());
139         this->lbl_initial_velocity = (gcnew System::Windows::Forms::
        Label());
140         this->lbl_initial_soil = (gcnew System::Windows::Forms::Label
        ());
141         this->lbl_e = (gcnew System::Windows::Forms::Label());
142         this->lbl_p = (gcnew System::Windows::Forms::Label());
143         this->lbl_cv = (gcnew System::Windows::Forms::Label());
144         this->lbl_cs = (gcnew System::Windows::Forms::Label());
145         this->lbl_bv = (gcnew System::Windows::Forms::Label());
146         this->lbl_bs = (gcnew System::Windows::Forms::Label());
147         this->lbl_av = (gcnew System::Windows::Forms::Label());
148         this->lbl_as = (gcnew System::Windows::Forms::Label());
149         this->dropPath = (gcnew System::Windows::Forms::RichTextBox())
        ;
150         this->label4 = (gcnew System::Windows::Forms::Label());
151         this->iwdStatusStrip = (gcnew System::Windows::Forms::
        StatusStrip());
152         this->iwdWorkProgressBar = (gcnew System::Windows::Forms::
        ToolStripProgressBar());
153         this->lblCurrentIteration = (gcnew System::Windows::Forms::
        ToolStripStatusLabel());
154         this->groupBox2 = (gcnew System::Windows::Forms::GroupBox());
155         this->lblPathLength = (gcnew System::Windows::Forms::Label());
156         this->label1 = (gcnew System::Windows::Forms::Label());
157         this->lblTimer = (gcnew System::Windows::Forms::Label());
158         this->iwdTimer = (gcnew System::Windows::Forms::Timer(this->
        components));
159         this->groupBox1->SuspendLayout();
160         this->iwdStatusStrip->SuspendLayout();
161         this->groupBox2->SuspendLayout();
162         this->SuspendLayout();
163         //
164         // btnCancel
165         //
166         this->btnCancel->Location = System::Drawing::Point(272, 344);
167         this->btnCancel->Name = L"btnCancel";
168         this->btnCancel->Size = System::Drawing::Size(95, 23);
169         this->btnCancel->TabIndex = 0;
170         this->btnCancel->Text = LAκύρωση"";
171         this->btnCancel->UseVisualStyleBackColor = true;
172         this->btnCancel->Click += gcnew System::EventHandler(this, &
        ProgressForm::btnCancel_Click);
```

```
173         //
174         // iwdWorker
175         //
176         this->iwdWorker->WorkerReportsProgress = true;
177         this->iwdWorker->WorkerSupportsCancellation = true;
178         this->iwdWorker->DoWork += gcnew System::ComponentModel::
        DoWorkEventHandler(this, &ProgressForm::iwdWorker_DoWork);
179         this->iwdWorker->ProgressChanged += gcnew System::
        ComponentModel::ProgressChangedEventHandler(this, &ProgressForm::
        iwdWorker_ProgressChanged);
180         this->iwdWorker->RunWorkerCompleted += gcnew System::
        ComponentModel::RunWorkerCompletedEventHandler(this, &
        ProgressForm::iwdWorker_RunWorkerCompleted);
181         //
182         // label3
183         //
184         this->label3->AutoSize = true;
185         this->label3->Font = (gcnew System::Drawing::Font(L"Microsoft
        Sans Serif", 10, System::Drawing::FontStyle::Regular, System::
        Drawing::GraphicsUnit::Point,
186           static_cast<System::Byte>(204)));
187         this->label3->Location = System::Drawing::Point(12, 9);
188         this->label3->Name = L"label3";
189         this->label3->Size = System::Drawing::Size(79, 17);
190         this->label3->TabIndex = 6;
191         this->label3->Text = LΚατάσταση"";
192         //
193         // lblStatus
194         //
195         this->lblStatus->AutoSize = true;
196         this->lblStatus->Font = (gcnew System::Drawing::Font(L"
        Microsoft Sans Serif", 10, System::Drawing::FontStyle::Regular,
        System::Drawing::GraphicsUnit::Point,
197           static_cast<System::Byte>(204)));
198         this->lblStatus->Location = System::Drawing::Point(121, 9);
199         this->lblStatus->Name = L"lblStatus";
200         this->lblStatus->Size = System::Drawing::Size(13, 17);
201         this->lblStatus->TabIndex = 7;
202         this->lblStatus->Text = L"-";
203         //
204         // groupBox1
205         //
206         this->groupBox1->Controls->Add(this->lblNodes);
207         this->groupBox1->Controls->Add(this->lbl_random_initial_node);
208         this->groupBox1->Controls->Add(this->lbl_output_filename);
209         this->groupBox1->Controls->Add(this->lbl_no_files);
210         this->groupBox1->Controls->Add(this->lbl_initial_velocity);
211         this->groupBox1->Controls->Add(this->lbl_initial_soil);
212         this->groupBox1->Controls->Add(this->lbl_e);
213         this->groupBox1->Controls->Add(this->lbl_p);
214         this->groupBox1->Controls->Add(this->lbl_cv);
```

```
215         this->groupBox1->Controls->Add(this->lbl_cs);
216         this->groupBox1->Controls->Add(this->lbl_bv);
217         this->groupBox1->Controls->Add(this->lbl_bs);
218         this->groupBox1->Controls->Add(this->lbl_av);
219         this->groupBox1->Controls->Add(this->lbl_as);
220         this->groupBox1->Location = System::Drawing::Point(12, 30);
221         this->groupBox1->Name = L"groupBox1";
222         this->groupBox1->Size = System::Drawing::Size(353, 151);
223         this->groupBox1->TabIndex = 12;
224         this->groupBox1->TabStop = false;
225         this->groupBox1->Text = LΠαράμετροι" && ρυθμίσεις";
226         //
227         // lblNodes
228         //
229         this->lblNodes->AutoSize = true;
230         this->lblNodes->Location = System::Drawing::Point(52, 77);
231         this->lblNodes->Name = L"lblNodes";
232         this->lblNodes->Size = System::Drawing::Size(49, 13);
233         this->lblNodes->TabIndex = 33;
234         this->lblNodes->Text = LΚόμβοι" : ";
235         //
236         // lbl_random_initial_node
237         //
238         this->lbl_random_initial_node->AutoSize = true;
239         this->lbl_random_initial_node->Location = System::Drawing::
    Point(4, 133);
240         this->lbl_random_initial_node->Name = L"
    lbl_random_initial_node";
241         this->lbl_random_initial_node->Size = System::Drawing::Size
    (121, 13);
242         this->lbl_random_initial_node->TabIndex = 32;
243         this->lbl_random_initial_node->Text = LΤυχαίο" αρχικός κόμβος";
244         this->lbl_random_initial_node->Visible = false;
245         //
246         // lbl_output_filename
247         //
248         this->lbl_output_filename->AutoSize = true;
249         this->lbl_output_filename->Location = System::Drawing::Point
    (6, 111);
250         this->lbl_output_filename->Name = L"lbl_output_filename";
251         this->lbl_output_filename->Size = System::Drawing::Size(133,
    13);
252         this->lbl_output_filename->TabIndex = 31;
253         this->lbl_output_filename->Text = LΌνομα" αρχείων εξόδου : ";
254         this->lbl_output_filename->Visible = false;
255         //
256         // lbl_no_files
257         //
258         this->lbl_no_files->AutoSize = true;
259         this->lbl_no_files->Location = System::Drawing::Point(6, 98);
260         this->lbl_no_files->Name = L"lbl_no_files";
```

```
261         this->lbl_no_files->Size = System::Drawing::Size(146, 13);
262         this->lbl_no_files->TabIndex = 30;
263         this->lbl_no_files->Text = LΔημιουργία" αρχείων εξόδου";
264         this->lbl_no_files->Visible = false;
265         //
266         // lbl_initial_velocity
267         //
268         this->lbl_initial_velocity->AutoSize = true;
269         this->lbl_initial_velocity->Location = System::Drawing::Point
    (13, 64);
270         this->lbl_initial_velocity->Name = L"lbl_initial_velocity";
271         this->lbl_initial_velocity->Size = System::Drawing::Size(88,
    13);
272         this->lbl_initial_velocity->TabIndex = 29;
273         this->lbl_initial_velocity->Text = LΑρχικό" velocity : ";
274         //
275         // lbl_initial_soil
276         //
277         this->lbl_initial_soil->AutoSize = true;
278         this->lbl_initial_soil->Location = System::Drawing::Point(31,
    51);
279         this->lbl_initial_soil->Name = L"lbl_initial_soil";
280         this->lbl_initial_soil->Size = System::Drawing::Size(70, 13);
281         this->lbl_initial_soil->TabIndex = 28;
282         this->lbl_initial_soil->Text = LΑρχικό" soil  : ";
283         //
284         // lbl_e
285         //
286         this->lbl_e->AutoSize = true;
287         this->lbl_e->Location = System::Drawing::Point(257, 29);
288         this->lbl_e->Name = L"lbl_e";
289         this->lbl_e->Size = System::Drawing::Size(25, 13);
290         this->lbl_e->TabIndex = 27;
291         this->lbl_e->Text = Lε" = ";
292         //
293         // lbl_p
294         //
295         this->lbl_p->AutoSize = true;
296         this->lbl_p->Location = System::Drawing::Point(257, 16);
297         this->lbl_p->Name = L"lbl_p";
298         this->lbl_p->Size = System::Drawing::Size(25, 13);
299         this->lbl_p->TabIndex = 26;
300         this->lbl_p->Text = Lρ" = ";
301         //
302         // lbl_cv
303         //
304         this->lbl_cv->AutoSize = true;
305         this->lbl_cv->Location = System::Drawing::Point(172, 29);
306         this->lbl_cv->Name = L"lbl_cv";
307         this->lbl_cv->Size = System::Drawing::Size(31, 13);
308         this->lbl_cv->TabIndex = 25;
```

```
309         this->lbl_cv->Text = L"cv = ";
310         //
311         // lbl_cs
312         //
313         this->lbl_cs->AutoSize = true;
314         this->lbl_cs->Location = System::Drawing::Point(172, 16);
315         this->lbl_cs->Name = L"lbl_cs";
316         this->lbl_cs->Size = System::Drawing::Size(30, 13);
317         this->lbl_cs->TabIndex = 24;
318         this->lbl_cs->Text = L"cs = ";
319         //
320         // lbl_bv
321         //
322         this->lbl_bv->AutoSize = true;
323         this->lbl_bv->Location = System::Drawing::Point(92, 29);
324         this->lbl_bv->Name = L"lbl_bv";
325         this->lbl_bv->Size = System::Drawing::Size(31, 13);
326         this->lbl_bv->TabIndex = 23;
327         this->lbl_bv->Text = L"bv = ";
328         //
329         // lbl_bs
330         //
331         this->lbl_bs->AutoSize = true;
332         this->lbl_bs->Location = System::Drawing::Point(92, 16);
333         this->lbl_bs->Name = L"lbl_bs";
334         this->lbl_bs->Size = System::Drawing::Size(30, 13);
335         this->lbl_bs->TabIndex = 22;
336         this->lbl_bs->Text = L"bs = ";
337         //
338         // lbl_av
339         //
340         this->lbl_av->AutoSize = true;
341         this->lbl_av->Location = System::Drawing::Point(13, 29);
342         this->lbl_av->Name = L"lbl_av";
343         this->lbl_av->Size = System::Drawing::Size(31, 13);
344         this->lbl_av->TabIndex = 21;
345         this->lbl_av->Text = L"av = ";
346         //
347         // lbl_as
348         //
349         this->lbl_as->AutoSize = true;
350         this->lbl_as->Location = System::Drawing::Point(13, 16);
351         this->lbl_as->Name = L"lbl_as";
352         this->lbl_as->Size = System::Drawing::Size(30, 13);
353         this->lbl_as->TabIndex = 20;
354         this->lbl_as->Text = L"as = ";
355         //
356         // dropPath
357         //
358         this->dropPath->Location = System::Drawing::Point(9, 32);
359         this->dropPath->Name = L"dropPath";
```

```
360         this->dropPath->ReadOnly = true;
361         this->dropPath->Size = System::Drawing::Size(335, 69);
362         this->dropPath->TabIndex = 14;
363         this->dropPath->Text = L"";
364         //
365         // label4
366         //
367         this->label4->AutoSize = true;
368         this->label4->Location = System::Drawing::Point(6, 16);
369         this->label4->Name = L"label4";
370         this->label4->Size = System::Drawing::Size(54, 13);
371         this->label4->TabIndex = 15;
372         this->label4->Text = LΔιαδρομή"";
373         //
374         // iwdStatusStrip
375         //
376         this->iwdStatusStrip->AutoSize = false;
377         this->iwdStatusStrip->Items->AddRange(gcnew cli::array< System
    ::Windows::Forms::ToolStripItem^  >(2) {
378           this->iwdWorkProgressBar,
379             this->lblCurrentIteration
380         });
381         this->iwdStatusStrip->Location = System::Drawing::Point(0,
    370);
382         this->iwdStatusStrip->Name = L"iwdStatusStrip";
383         this->iwdStatusStrip->Size = System::Drawing::Size(379, 29);
384         this->iwdStatusStrip->SizingGrip = false;
385         this->iwdStatusStrip->TabIndex = 16;
386         this->iwdStatusStrip->Text = L"statusStrip1";
387         //
388         // iwdWorkProgressBar
389         //
390         this->iwdWorkProgressBar->Name = L"iwdWorkProgressBar";
391         this->iwdWorkProgressBar->Size = System::Drawing::Size(240,
    23);
392         //
393         // lblCurrentIteration
394         //
395         this->lblCurrentIteration->Name = L"lblCurrentIteration";
396         this->lblCurrentIteration->Size = System::Drawing::Size(18,
    24);
397         this->lblCurrentIteration->Text = L" - ";
398         //
399         // groupBox2
400         //
401         this->groupBox2->Controls->Add(this->lblPathLength);
402         this->groupBox2->Controls->Add(this->label1);
403         this->groupBox2->Controls->Add(this->dropPath);
404         this->groupBox2->Controls->Add(this->label4);
405         this->groupBox2->Location = System::Drawing::Point(12, 187);
406         this->groupBox2->Name = L"groupBox2";
```

```
407         this->groupBox2->Size = System::Drawing::Size(350, 151);
408         this->groupBox2->TabIndex = 17;
409         this->groupBox2->TabStop = false;
410         this->groupBox2->Text = LΑποτελέσματα"";
411         //
412         // lblPathLength
413         //
414         this->lblPathLength->AutoSize = true;
415         this->lblPathLength->Location = System::Drawing::Point(80,
    104);
416         this->lblPathLength->Name = L"lblPathLength";
417         this->lblPathLength->Size = System::Drawing::Size(16, 13);
418         this->lblPathLength->TabIndex = 17;
419         this->lblPathLength->Text = L" - ";
420         //
421         // label1
422         //
423         this->label1->AutoSize = true;
424         this->label1->Location = System::Drawing::Point(6, 104);
425         this->label1->Name = L"label1";
426         this->label1->Size = System::Drawing::Size(65, 13);
427         this->label1->TabIndex = 16;
428         this->label1->Text = LΑπόσταση" :";
429         //
430         // lblTimer
431         //
432         this->lblTimer->AutoSize = true;
433         this->lblTimer->Location = System::Drawing::Point(12, 345);
434         this->lblTimer->Name = L"lblTimer";
435         this->lblTimer->Size = System::Drawing::Size(16, 13);
436         this->lblTimer->TabIndex = 18;
437         this->lblTimer->Text = L" - ";
438         //
439         // iwdTimer
440         //
441         this->iwdTimer->Tick += gcnew System::EventHandler(this, &
    ProgressForm::iwdTimer_Tick);
442         //
443         // ProgressForm
444         //
445         this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
446         this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::
    Font;
447         this->ClientSize = System::Drawing::Size(379, 399);
448         this->Controls->Add(this->lblTimer);
449         this->Controls->Add(this->groupBox2);
450         this->Controls->Add(this->iwdStatusStrip);
451         this->Controls->Add(this->btnCancel);
452         this->Controls->Add(this->groupBox1);
453         this->Controls->Add(this->lblStatus);
454         this->Controls->Add(this->label3);
```

```
455         this->FormBorderStyle = System::Windows::Forms::
    FormBorderStyle::FixedSingle;
456         this->MaximizeBox = false;
457         this->MinimizeBox = false;
458         this->Name = L"ProgressForm";
459         this->ShowIcon = false;
460         this->Text = LΠρόοδος"";
461         this->FormClosing += gcnew System::Windows::Forms::
    FormClosingEventHandler(this, &ProgressForm::
    ProgressForm_FormClosing);
462         this->groupBox1->ResumeLayout(false);
463         this->groupBox1->PerformLayout();
464         this->iwdStatusStrip->ResumeLayout(false);
465         this->iwdStatusStrip->PerformLayout();
466         this->groupBox2->ResumeLayout(false);
467         this->groupBox2->PerformLayout();
468         this->ResumeLayout(false);
469         this->PerformLayout();
470 
471      }
472 #pragma endregion
473 
474    private: System::Void iwdWorker_DoWork(System::Object^  sender,
     System::ComponentModel::DoWorkEventArgs^  e) {
475 
476      auto state = static_cast<IwdWorkerState^>(e->Argument);
477      auto tmp = state->filepath;
478      std::string filepath = ToStdString(tmp);   //???
479 
480      auto settings = state->settings->getOriginal();
481 
482      srand(time(nullptr));
483      settings.seed = rand();
484 
485      Graph graph;
486      try {
487         state->status = LAνάγνωση" αρχείου γράφου";
488         iwdWorker->ReportProgress(0, state);
489         graph = graphFileReader(filepath);
490      } catch(std::invalid_argument e) {
491         state->status = LΣφάλμα"";
492         iwdWorker->ReportProgress(0, state);
493         MessageBox::Show(LTo" αρχείο" + (wchar_t)(filepath.c_str()) + L
    " δεν βρέθηκε.", LΠροσοχή"", MessageBoxButtons::OK, MessageBoxIcon
    ::Information);
494         return;
495      }
496 
497      Stopwatch<std::chrono::milliseconds> all;
498      Αρχικοποίηση// μεταβλητών
499      state->status = LΑρχικοποίηση" μεταβλητών";
```

```
500      iwdWorker->ReportProgress(0, state);
501
502      auto graph_size = graph.nodes.size();
503      graph.initiateSoil(settings.initial_soil);
504      auto drops = std::vector<iwd::Drop>(graph.nodes.size(), iwd::
       Drop(graph, settings));
505      iwd::Drop* best_drop = nullptr; Δείκτης// προς την σταγόνα με την
       βέλτιστη λύση
506
507                       Ορίζουμε// το αρχικό μήκος βέλτιστης διαδρομής στο
       άπειρο
508      auto best_tour_length = std::numeric_limits<double>::infinity();
509
510      Μεταβλητή// που κρατάει τους κόμβους της βέλτιστης διαδρομής
511      iwd::collection best_tour;
512      std::ostringstream pdataΜεταβητή;// που κρατάει τα plot data
513
514                   //Debug μεταβλητές
515      auto best_iteration = 0;
516      auto last_iteration_change = 0; //unused
517      auto iterations_without_change = 0;
518
519      state->status = LΕκτέλεση"";
520      state->nodes = graph.nodes.size();
521      iwdWorker->ReportProgress(0, state);
522
523      // Μεταβλητές για το ποσοστό ολοκλήρωσης
524      double drops_done = 1;
525      unsigned percent = 0, new_percent = 0;
526      for(auto iteration = 0u; iteration < settings.max_iterations;
       iteration++) {
527        if(iwdWorker->CancellationPending) {
528          e->Cancel = true;
529          return;
530        } else {
531          Stopwatch<std::chrono::milliseconds> it;
532          Κρατάμε// μία σταγόνα για κάθε node
533          for(auto& drop : drops) {
534            if(iwdWorker->CancellationPending) {
535              e->Cancel = true;
536              return;
537            } else {
538              auto drop_start = Stopwatch<>();
539              drop.travel();
540
541              drop_start.stop();
542              Βρίσκουμε// την σταγόνα με την καλύτερημικρότερη() διαδρομή
543              if(drop.path_length < best_tour_length) {
544                best_drop = &drop;
545                best_tour_length = drop.path_length;
546                best_iteration = iteration;
```

```
547             best_tour = drop.visited_nodes;
548             iterations_without_change = 0;
549             state->pathLength = best_tour_length;
550             iwdWorker->ReportProgress(-1, state);
551           } else {
552             ++iterations_without_change;
553           }
554           // Ποσοστό ολοκλήρωσης
555           new_percent = (drops_done++ / (settings.max_iterations *
      graph_size)) * 100;
556           if(percent < new_percent) {
557             percent = new_percent;
558             state->currentIteration = iteration+1;
559             // Παίρνουμε την καλύτερη διαδρομή
560             String^ path;
561             for(auto drop : best_tour) {
562               path += drop->id + " ";
563             }
564             state->path = path;
565             iwdWorker->ReportProgress(percent,state);
566           }
567         }
568       }
569       pdata << iteration << " " << best_tour_length << std::endl;
570       Έυρεση// της τοπικής βέλτιστης διαδρομής
571       auto Nib = graph_size;
572       auto p = settings.p;
573       Ανανεώνουμε// το soil στην τοπική βέλτιστη διαδρομή
574       for(unsigned id = 1; id < graph_size; id++) {
575         auto n1 = *best_drop->visited_nodes[id - 1], n2 = *
    best_drop->visited_nodes[id];
576         auto prev_soil = best_drop->graph.getSoil(n1, n2);
577         soil new_soil = (1 + p) * prev_soil - (p * (1 / (Nib - 1))
    *best_drop->soil);
578         graph.setSoil(n1, n2, new_soil);
579       }
580       it.stop();
581     }
582   }
583   state->pathLength = best_tour_length;
584   iwdWorker->ReportProgress(-1, state);
585   all.stop();
586
587   // Παίρνουμε την καλύτερη διαδρομή
588   String^ path;
589   for(auto drop : best_tour) {
590     path += drop->id + " ";
591   }
592   state->path = path;
593   iwdWorker->ReportProgress(-1, state);
594
```

```
595      if(!settings.no_files) {
596        bool writeOK = true;
597        state->status = LΔημιουργία" αρχείων εξόδου";
598        iwdWorker->ReportProgress(100, state);
599
600        Γράφουμε// το αρχείο με τα δεδομένα του γραφήματος και το κλείνουμε
601        std::ofstream f_pdata(settings.out_filename + ".pdat");
602        if(f_pdata.good()) {
603          f_pdata << pdata.str();
604        } else {
605          writeOK = false;
606        }
607        f_pdata.close();
608
609        Εξαγωγή// του γράφου σε neato format
610        // graph G{
611        //  n0[pos = "0,0!"];
612        //    n1[pos = "2,0!"];
613        //    n0 -- n1 -- n0;
614        // }
615        std::ostringstream neato;
616        std::ostringstream neato_node_connection;
617        neato << "graph G {" << std::endl;
618        neato << "node [style=filled,color=black];" << std::endl;
619        for(auto * node : best_tour) {
620          neato << "n" << node->id << "[pos = \"" << node->x << "," <<
       node->y << "!\"];" << std::endl;
621          neato_node_connection << "n" << node->id << " -- ";
622        }
623        neato_node_connection << "n" << best_tour[0]->id << "[color=
      red,penwidth=3.0];" << std::endl;
624        neato << neato_node_connection.str();
625        neato << "}";
626        Γράφουμε// τα δεδομένα για το πρόγραμμα neato
627        std::ofstream best_tour_file(settings.out_filename + ".ndat",
      std::ofstream::out);
628        if(best_tour_file.good()) {
629          best_tour_file << neato.str();
630        } else {
631          writeOK = false;
632        }
633        best_tour_file.close();
634        if(!writeOK) {
635          MessageBox::Show(LΠαρουσιάστηκε" πόβλημα κατά την δημιουργεία
      των αρχείων εξόδου");
636        }
637      }
638    }
639
640    private: System::Void iwdWorker_ProgressChanged(System::Object^
       sender, System::ComponentModel::ProgressChangedEventArgs^  e) {
```

```
641       if(e->ProgressPercentage >= 0)
642         iwdWorkProgressBar->Value = e->ProgressPercentage;
643       auto state = static_cast<IwdWorkerState^>(e->UserState);
644       lblPathLength->Text = state->pathLength.ToString("F2");
645       lblCurrentIteration->Text = LΓενιές" : " + Convert::ToString(
      state->currentIteration + " / "+state->max_iterations);
646       lblNodes->Text = LΚόμβοι" : " + Convert::ToString(state->nodes);
647       lblStatus->Text = state->status;
648       if(!String::IsNullOrEmpty(state->path)) dropPath->Text = state->
      path;
649     }
650
651     private: System::Void iwdWorker_RunWorkerCompleted(System::Object^
         sender, System::ComponentModel::RunWorkerCompletedEventArgs^  e
      ) {
652       iwdTimer->Stop();
653       if(e->Error != nullptr) {
654         MessageBox::Show(e->Error->Message);
655       } else if(e->Cancelled) {
656         lblStatus->Text = LΑκυρώθηκε"";
657       } else {
658         lblStatus->Text = LΟλοκληρώθηκε"";
659         btnCancel->Enabled = false;
660       }
661       if(this->closeWhenFinished) this->Close();
662     }
663
664     private: System::Void btnCancel_Click(System::Object^  sender,
      System::EventArgs^  e) {
665         cancelWork();
666     }
667
668     private: System::Void ProgressForm_FormClosing(System::Object^
      sender, System::Windows::Forms::FormClosingEventArgs^  e) {
669       if(iwdWorker->IsBusy) {
670         cancelWork();
671         e->Cancel = true;
672         this->closeWhenFinished = true;
673       }
674     }
675
676     void cancelWork() {
677       btnCancel->Enabled = false;
678       if(iwdWorker->IsBusy) {
679         lblStatus->Text = LΑκυρώνεται"...";
680         iwdWorker->CancelAsync();
681       }
682     }
683
684     private: System::Void iwdTimer_Tick(System::Object^  sender,
      System::EventArgs^  e) {
```

```
685     auto elapsed = DateTime::Now.Subtract(opStart);
686     lblTimer->Text = elapsed.ToString("h\\:mm\\:ss\\.ff");
687   }
688 };
689 }
```

IwdHelpClasses.h

```
1
2  #pragma once
3  #include <string>
4  #include <msclr\marshal_cppstd.h>
5  #include <exception>
6
7  #include "IWD/Drop.h"
8  #include "IWD/Settings.hpp"
9  #include "Utilities.hpp"
10 #include "ProgressForm.h"
11
12 #define ToStdString(x) msclr::interop::marshal_as<std::string>(x);
13
14 namespace iwdgui {
15
16   using namespace System;
17
18   Αντιγραφή// της κλάσης iwd::Settings για managed κώδικα
19   public ref class Settings : public System::Object {
20   public:
21     int initial_velocity = 200;
22     int initial_soil = 1000;
23     double av = 1;
24     double bv = 0.01;
25     double cv = 1;
26     double as = 1;
27     double bs = 0.01;
28     double cs = 1;
29     double e = 0.0001;
30     double p = 0.9;
31     bool random_initial_node = false;
32     int seed = 0;
33     unsigned int max_iterations = 100;
34     String^ out_filename = "output";
35     bool no_files = false;
36
37     Settings() {};
38
39     Settings(iwd::Settings& s) {
40       initial_velocity = s.initial_velocity;
41       initial_soil = s.initial_soil;
42       av = s.av;
43       bv = s.bv;
44       cv = s.cv;
```

```
45        as = s.as;
46        bs = s.bs;
47        cs = s.cs;
48        e = s.e;
49        p = s.p;
50        random_initial_node = s.random_initial_node;
51        seed = s.seed;
52        max_iterations = s.max_iterations;
53        out_filename = gcnew String(s.out_filename.c_str());
54        no_files = s.no_files;
55      }
56
57      // Επιστρέφει τις ρυθμίσεις στην αρχική τους μορφή. Αυτό επειδή
58      // μεγάλο μέρος του αλγορίθμου τα χρειάζεται σε αυτή την μορφή.
59      iwd::Settings getOriginal() {
60        iwd::Settings s;
61        s.initial_velocity             = this->initial_velocity;
62        s.initial_soil                 = this->initial_soil;
63        s.av                           = this->av;
64        s.bv                           = this->bv;
65        s.cv                           = this->cv;
66        s.as                           = this->as;
67        s.bs                           = this->bs;
68        s.cs                           = this->cs;
69        s.e                            = this->e;
70        s.p                            = this->p;
71        s.random_initial_node          = this->random_initial_node;
72        s.seed                         = this->seed;
73        s.max_iterations               = this->max_iterations;
74        String^ tmp = this->out_filename;
75        s.out_filename           = ToStdString(tmp);
76        s.no_files                     = this->no_files;
77
78        return s;
79      }
80
81    };
82
83    /// <summary>
84    /// Περιέχει τα στοιχεία της κατάστασης εκτέλεσης της σταγόνας
85    /// Αυτά μεταφέρονται απο το ένα thread στο άλλο για να υπάρχει
86    /// επικοινωνίασυγχρονισμός/ μεταξύ τους.
87    /// </summary>
88    public ref struct IwdWorkerState : public System::Object {
89      unsigned int currentIteration = 0;
90      unsigned int max_iterations = 0;
91      unsigned int nodes = 0;
92      double pathLength = std::numeric_limits<double>::infinity();
93      String^ path;
94      Settings^ settings;
95      String^ status;
```

```
96      String^ filepath;
97    };
98
99  }
```

13.4 Αρχείο γράφου eil51.tsp

Αρχείο γράφου eil51.tsp

```
NAME : eil51
COMMENT : 51-city problem (Christofides/Eilon)
TYPE : TSP
DIMENSION : 51
EDGE_WEIGHT_TYPE : EUC_2D
NODE_COORD_SECTION
1 37 52
2 49 49
3 52 64
4 20 26
5 40 30
6 21 47
7 17 63
8 31 62
9 52 33
10 51 21
11 42 41
12 31 32
13 5 25
14 12 42
15 36 16
16 52 41
17 27 23
18 17 33
19 13 13
20 57 58
21 62 42
22 42 57
23 16 57
24 8 52
25 7 38
26 27 68
27 30 48
28 43 67
29 58 48
30 58 27
31 37 69
32 38 46
33 46 10
34 61 33
35 62 63
36 63 69
37 32 22
38 45 35
39 59 15
40 5 6
41 10 17
42 21 10
```

```
43 5 64
44 30 15
45 39 10
46 32 39
47 25 32
48 25 55
49 48 28
50 56 37
51 30 40
EOF
```

13.5 Αρχείο γράφου eil101.tsp

Αρχείο γράφου eil101.tsp

```
NAME : eil101
COMMENT : 101-city problem (Christofides/Eilon)
TYPE : TSP
DIMENSION : 101
EDGE_WEIGHT_TYPE : EUC_2D
NODE_COORD_SECTION
1 41 49
2 35 17
3 55 45
4 55 20
5 15 30
6 25 30
7 20 50
8 10 43
9 55 60
10 30 60
11 20 65
12 50 35
13 30 25
14 15 10
15 30 5
16 10 20
17 5 30
18 20 40
19 15 60
20 45 65
21 45 20
22 45 10
23 55 5
24 65 35
25 65 20
26 45 30
27 35 40
28 41 37
29 64 42
30 40 60
31 31 52
32 35 69
33 53 52
34 65 55
35 63 65
36 2 60
37 20 20
38 5 5
39 60 12
40 40 25
41 42 7
42 24 12
```

```
43 23 3
44 11 14
45 6 38
46 2 48
47 8 56
48 13 52
49 6 68
50 47 47
51 49 58
52 27 43
53 37 31
54 57 29
55 63 23
56 53 12
57 32 12
58 36 26
59 21 24
60 17 34
61 12 24
62 24 58
63 27 69
64 15 77
65 62 77
66 49 73
67 67 5
68 56 39
69 37 47
70 37 56
71 57 68
72 47 16
73 44 17
74 46 13
75 49 11
76 49 42
77 53 43
78 61 52
79 57 48
80 56 37
81 55 54
82 15 47
83 14 37
84 11 31
85 16 22
86 4 18
87 28 18
88 26 52
89 26 35
90 31 67
91 15 19
92 22 22
93 18 24
```

```
94 26 27
95 25 24
96 22 27
97 25 21
98 19 21
99 20 26
100 18 18
101 35 35
EOF
```

13.6 Αρχείο γράφου a280.tsp

Αρχείο γράφου a280.tsp

```
NAME : a280
COMMENT : drilling problem (Ludwig)
TYPE : TSP
DIMENSION: 280
EDGE_WEIGHT_TYPE : EUC_2D
NODE_COORD_SECTION
  1 288 149
  2 288 129
  3 270 133
  4 256 141
  5 256 157
  6 246 157
  7 236 169
  8 228 169
  9 228 161
 10 220 169
 11 212 169
 12 204 169
 13 196 169
 14 188 169
 15 196 161
 16 188 145
 17 172 145
 18 164 145
 19 156 145
 20 148 145
 21 140 145
 22 148 169
 23 164 169
 24 172 169
 25 156 169
 26 140 169
 27 132 169
 28 124 169
 29 116 161
 30 104 153
 31 104 161
 32 104 169
 33  90 165
 34  80 157
 35  64 157
 36  64 165
 37  56 169
 38  56 161
 39  56 153
 40  56 145
 41  56 137
 42  56 129
```

```
 43  56 121
 44  40 121
 45  40 129
 46  40 137
 47  40 145
 48  40 153
 49  40 161
 50  40 169
 51  32 169
 52  32 161
 53  32 153
 54  32 145
 55  32 137
 56  32 129
 57  32 121
 58  32 113
 59  40 113
 60  56 113
 61  56 105
 62  48  99
 63  40  99
 64  32  97
 65  32  89
 66  24  89
 67  16  97
 68  16 109
 69   8 109
 70   8  97
 71   8  89
 72   8  81
 73   8  73
 74   8  65
 75   8  57
 76  16  57
 77   8  49
 78   8  41
 79  24  45
 80  32  41
 81  32  49
 82  32  57
 83  32  65
 84  32  73
 85  32  81
 86  40  83
 87  40  73
 88  40  63
 89  40  51
 90  44  43
 91  44  35
 92  44  27
 93  32  25
```

```
 94  24  25
 95  16  25
 96  16  17
 97  24  17
 98  32  17
 99  44  11
100  56   9
101  56  17
102  56  25
103  56  33
104  56  41
105  64  41
106  72  41
107  72  49
108  56  49
109  48  51
110  56  57
111  56  65
112  48  63
113  48  73
114  56  73
115  56  81
116  48  83
117  56  89
118  56  97
119 104  97
120 104 105
121 104 113
122 104 121
123 104 129
124 104 137
125 104 145
126 116 145
127 124 145
128 132 145
129 132 137
130 140 137
131 148 137
132 156 137
133 164 137
134 172 125
135 172 117
136 172 109
137 172 101
138 172  93
139 172  85
140 180  85
141 180  77
142 180  69
143 180  61
144 180  53
```

```
145 172  53
146 172  61
147 172  69
148 172  77
149 164  81
150 148  85
151 124  85
152 124  93
153 124 109
154 124 125
155 124 117
156 124 101
157 104  89
158 104  81
159 104  73
160 104  65
161 104  49
162 104  41
163 104  33
164 104  25
165 104  17
166  92   9
167  80   9
168  72   9
169  64  21
170  72  25
171  80  25
172  80  25
173  80  41
174  88  49
175 104  57
176 124  69
177 124  77
178 132  81
179 140  65
180 132  61
181 124  61
182 124  53
183 124  45
184 124  37
185 124  29
186 132  21
187 124  21
188 120   9
189 128   9
190 136   9
191 148   9
192 162   9
193 156  25
194 172  21
195 180  21
```

```
196 180  29
197 172  29
198 172  37
199 172  45
200 180  45
201 180  37
202 188  41
203 196  49
204 204  57
205 212  65
206 220  73
207 228  69
208 228  77
209 236  77
210 236  69
211 236  61
212 228  61
213 228  53
214 236  53
215 236  45
216 228  45
217 228  37
218 236  37
219 236  29
220 228  29
221 228  21
222 236  21
223 252  21
224 260  29
225 260  37
226 260  45
227 260  53
228 260  61
229 260  69
230 260  77
231 276  77
232 276  69
233 276  61
234 276  53
235 284  53
236 284  61
237 284  69
238 284  77
239 284  85
240 284  93
241 284 101
242 288 109
243 280 109
244 276 101
245 276  93
246 276  85
```

```
247 268  97
248 260 109
249 252 101
250 260  93
251 260  85
252 236  85
253 228  85
254 228  93
255 236  93
256 236 101
257 228 101
258 228 109
259 228 117
260 228 125
261 220 125
262 212 117
263 204 109
264 196 101
265 188  93
266 180  93
267 180 101
268 180 109
269 180 117
270 180 125
271 196 145
272 204 145
273 212 145
274 220 145
275 228 145
276 236 145
277 246 141
278 252 125
279 260 129
280 280 133
EOF
```

13.7 Βέλτιστη διαδρομή γράφου eil51.tsp

Βέλτιστη διαδρομή γράφου eil51.tsp

```
NAME : eil51.opt.tour
COMMENT : Optimal tour for eil51.tsp  (426)
TYPE : TOUR
DIMENSION : 51
TOUR_SECTION
1
22
8
26
31
28
3
36
35
20
2
29
21
16
50
34
30
9
49
10
39
33
45
15
44
42
40
19
41
13
25
14
24
43
7
23
48
6
27
51
46
12
47
```

```
18
4
17
37
5
38
11
32
-1
EOF
```

13.8 Βέλτιστη διαδρομή γράφου eil101.tsp

Βέλτιστη διαδρομή γράφου eil101.tsp

```
NAME : eil101.opt.tour
COMMENT : Optimum tour for eil101.tsp (Length 629)
TYPE : TOUR
DIMENSION : 101
TOUR_SECTION
1
69
27
101
53
28
26
12
80
68
29
24
54
55
25
4
39
67
23
56
75
41
22
74
72
73
21
40
58
13
94
95
97
87
2
57
15
43
42
14
44
38
86
```

```
16
61
85
91
100
98
37
92
59
93
99
96
6
89
52
18
83
60
5
84
17
45
8
46
47
36
49
64
63
90
32
10
62
11
19
48
82
7
88
31
70
30
20
66
71
65
35
34
78
81
9
```

```
51
33
79
3
77
76
50
-1
EOF
```

13.9 Παράδειγμα αρχείου εξόδου `.ndat` για eil51

Παράδειγμα αρχείου εξόδου `.ndat` για eil51

```
graph G {
node [style=filled,color=black];
n0[pos = "37,52!"];
n31[pos = "38,46!"];
n45[pos = "32,39!"];
n50[pos = "30,40!"];
n11[pos = "31,32!"];
n46[pos = "25,32!"];
n3[pos = "20,26!"];
n17[pos = "17,33!"];
n24[pos = "7,38!"];
n13[pos = "12,42!"];
n5[pos = "21,47!"];
n47[pos = "25,55!"];
n26[pos = "30,48!"];
n10[pos = "42,41!"];
n37[pos = "45,35!"];
n4[pos = "40,30!"];
n48[pos = "48,28!"];
n8[pos = "52,33!"];
n9[pos = "51,21!"];
n29[pos = "58,27!"];
n33[pos = "61,33!"];
n49[pos = "56,37!"];
n20[pos = "62,42!"];
n28[pos = "58,48!"];
n19[pos = "57,58!"];
n34[pos = "62,63!"];
n35[pos = "63,69!"];
n2[pos = "52,64!"];
n27[pos = "43,67!"];
n30[pos = "37,69!"];
n7[pos = "31,62!"];
n25[pos = "27,68!"];
n22[pos = "16,57!"];
n6[pos = "17,63!"];
n42[pos = "5,64!"];
n23[pos = "8,52!"];
n12[pos = "5,25!"];
n40[pos = "10,17!"];
n18[pos = "13,13!"];
n39[pos = "5,6!"];
n41[pos = "21,10!"];
n43[pos = "30,15!"];
n36[pos = "32,22!"];
n16[pos = "27,23!"];
n14[pos = "36,16!"];
n44[pos = "39,10!"];
```

```
49 n32[pos = "46,10!"];
50 n38[pos = "59,15!"];
51 n15[pos = "52,41!"];
52 n1[pos = "49,49!"];
53 n21[pos = "42,57!"];
54 n0[pos = "37,52!"];
55 n0 -- n31 -- n45 -- n50 -- n11 -- n46 -- n3 -- n17 -- n24 -- n13 --
      n5 -- n47 -- n26 -- n10 -- n37 -- n4 -- n48 -- n8 -- n9 -- n29 --
       n33 -- n49 -- n20 -- n28 -- n19 -- n34 -- n35 -- n2 -- n27 --
      n30 -- n7 -- n25 -- n22 -- n6 -- n42 -- n23 -- n12 -- n40 -- n18
      -- n39 -- n41 -- n43 -- n36 -- n16 -- n14 -- n44 -- n32 -- n38 --
       n15 -- n1 -- n21 -- n0 -- n0[color=red,penwidth=3.0];
56 }
```

13.10 Παράδειγμα αρχείου εξόδου `.pdat` για eil51

Παράδειγμα αρχείου εξόδου `.pdat` για eil51

```
0 1113.93
1 1005.87
2 959.543
3 875.42
4 867.151
5 845.406
6 819.464
7 811.815
8 789.954
9 789.954
10 720.468
11 720.468
12 720.468
13 715.967
14 705.363
15 667.622
16 643.666
17 628.266
18 628.266
19 620.135
20 611.588
21 588.083
22 570.675
23 570.675
24 570.675
25 569.185
26 569.185
27 569.185
28 542.088
29 542.088
30 542.088
31 542.088
32 542.088
33 542.088
34 542.088
35 542.088
36 542.088
37 542.088
38 542.088
39 542.088
40 542.088
41 542.088
42 542.088
43 542.088
44 542.088
45 542.088
46 542.088
47 542.088
```

```
48 542.088
49 542.088
50 542.088
51 542.088
52 542.088
53 542.088
54 524.118
55 524.118
56 524.118
57 524.118
58 520.61
59 520.61
60 520.61
61 520.61
62 520.61
63 520.61
64 520.61
65 520.61
66 520.61
67 520.61
68 520.61
69 520.61
70 520.61
71 520.61
72 520.61
73 520.61
74 520.61
75 520.61
76 520.61
77 520.61
78 520.61
79 520.61
80 520.61
81 520.61
82 520.61
83 520.61
84 520.61
85 520.61
86 520.61
87 520.61
88 520.61
89 520.61
90 520.61
91 520.61
92 520.61
93 520.61
94 520.61
95 520.61
96 520.61
97 520.61
98 520.61
```

```
99 520.61
100 520.61
101 520.61
102 520.61
103 520.61
104 520.61
105 520.61
106 520.61
107 520.61
108 520.61
109 520.61
110 520.61
111 520.61
112 520.61
113 520.61
114 520.61
115 520.61
116 520.61
117 520.61
118 520.61
119 520.61
120 520.61
121 520.61
122 520.61
123 520.61
124 520.61
125 520.61
126 520.61
127 520.61
128 520.61
129 520.61
130 520.61
131 520.61
132 520.61
133 520.61
134 520.61
135 520.61
136 520.61
137 520.61
138 520.61
139 520.61
140 520.61
141 520.61
142 520.61
143 520.61
144 520.61
145 520.61
146 520.61
147 520.61
148 520.61
149 520.61
```

```
150 520.61
151 520.61
152 520.61
153 505.988
154 493.912
155 493.912
156 493.912
157 493.912
158 493.912
159 493.912
160 493.912
161 493.912
162 493.912
163 493.912
164 493.912
165 493.912
166 493.912
167 493.912
168 493.912
169 493.912
170 493.912
171 493.912
172 493.912
173 493.912
174 493.912
175 493.912
176 493.912
177 493.912
178 493.912
179 493.912
180 493.912
181 493.912
182 493.912
183 493.912
184 493.912
185 493.912
186 493.912
187 493.912
188 493.912
189 493.912
190 493.912
191 493.912
192 493.912
193 493.912
194 493.912
195 493.912
196 493.912
197 493.912
198 493.912
199 493.912
200 493.912
```

```
201 493.912
202 493.912
203 493.912
204 493.912
205 493.912
206 493.912
207 483.32
208 483.32
209 483.32
210 483.32
211 483.32
212 483.32
213 483.32
214 483.32
215 483.32
216 483.32
217 483.32
218 483.32
219 483.32
220 483.32
221 483.32
222 483.32
223 483.32
224 483.32
225 483.32
226 483.32
227 483.32
228 483.32
229 483.32
230 483.32
231 483.32
232 483.32
233 483.32
234 483.32
235 483.32
236 483.32
237 483.32
238 483.32
239 483.32
240 483.32
241 483.32
242 483.32
243 483.32
244 483.32
245 483.32
246 483.32
247 483.32
248 483.32
249 483.32
250 483.32
251 483.32
```

```
252 483.32
253 483.32
254 483.32
255 483.32
256 483.32
257 483.32
258 483.32
259 483.32
260 483.32
261 483.32
262 483.32
263 483.32
264 483.32
265 483.32
266 483.32
267 483.32
268 483.32
269 483.32
270 483.32
271 483.32
272 483.32
273 481.353
274 481.353
275 481.353
276 481.353
277 481.353
278 481.353
279 481.353
280 481.353
281 481.353
282 481.353
283 481.353
284 481.353
285 481.353
286 481.353
287 481.353
288 481.353
289 481.353
290 481.353
291 481.353
292 481.353
293 481.353
294 481.353
295 481.353
296 481.353
297 481.353
298 481.353
299 481.353
```