



Δημοκρίτειο Πανεπιστήμιο Θράκης
Πολυτεχνική Σχολή Ξάνθης
Τμήμα Μηχανικών Παραγωγής και Διοίκησης

Επιβλέποντες:

Αναπληρωτής Καθηγητής Κατσαβούνης Στέφανος,
Δρ. Κουλίνας Γεώργιος

**Ανάπτυξη Αλγορίθμου Σμήνους Μελισσών για την
Επίλυση του Προβλήματος RCPSP - DC και Υλοποίηση
σε Γλώσσα C++**

```
elite_bees[i].setStart(elite_bees[i].getSol(inputs - 1  
elite_bees[i].setNEFinish(elite_bees[i].getSol(inputs  
elite_bees[i].setDurationNEW(elite_bees[i].getFinish(e  
vector<vector<int>> gg;  
gg = elite_bees[i].getGannt();  
gg.resize(elite_bees[i].getDuration());  
elite_bees[i].setGannt(gg);
```

Διπλωματική Εργασία

Καραδουλαμάς Δημήτριος

A.M.: 91786

Ξάνθη, Νοέμβριος 2022

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία εκπονήθηκε στα πλαίσια του προπτυχιακού προγράμματος σπουδών του Τμήματος Μηχανικών Παραγωγής και Διοίκησης, της Πολυτεχνικής Σχολής του Δημοκρίτειου Πανεπιστημίου Θράκης υπό την καθοδήγηση του Αναπληρωτή Καθηγητή κ. Στέφανου Κατσαβούνη και του Δρ. κ. Γεώργιου Κουλίνα.

Θα ήθελα να ευχαριστήσω τον κ. Κατσαβούνη για την εξαιρετική του συνεργασία καθ' όλη την διάρκεια της διόρθωσης της εργασίας καθώς και για την πολυετή του καθοδήγηση κατά την διάρκεια των σπουδών μου. Στην συνέχεια θα ήθελα να ευχαριστήσω τον κ. Κουλίνα για τον χρόνο που αφιέρωσε και τις γνώσεις που μοιράστηκε μαζί μου κατά την διαδικασία ανάπτυξης του αλγορίθμου. Τελικά, θα ήθελα να ευχαριστήσω τους γονείς μου που παρείχαν ατελείωτη υποστήριξη όλα αυτά τα χρόνια.

ΠΕΡΙΛΗΨΗ

Στο σύγχρονο επιχειρηματικό περιβάλλον, κατά τη φάση σχεδίασης ενός επενδυτικού πλάνου, οι επιχειρήσεις καλούνται να διαχειριστούν μεγάλο όγκο πληροφοριών. Με την εγγενή πολυπλοκότητα έργων και χρονοπρογραμμάτων επικρατεί μία ασάφεια και μία αβεβαιότητα σχετικά με την ελκυστικότητα μιας επένδυσης. Η λήψη της πλέον κατάλληλης απόφασης από τους αρμόδιους διαχειριστές για τη διασφάλιση της απόδοσης της σχετικής επένδυσης επιβάλλει και απαιτεί χρήση αποτελεσματικών και αναγνωρισμένων κριτηρίων αξιολόγησης επενδύσεων και σχεδίαση σχετικών επιχειρηματικών μοντέλων.

Ένα από τα πιο βασικά μοντέλα αξιολόγησης χρησιμοποιεί το κριτήριο αξιολόγησης της Καθαρής Παρούσας Αξίας (*NPV*: Net Present Value). Στόχος του μοντέλου αυτού είναι η μεγιστοποίηση της *NPV* ενός επενδυτικού πλάνου. Επιλέγεται το κριτήριο αυτό καθώς το αποτέλεσμα της αξιολόγησης αποτελεί μία και μοναδική τιμή η οποία δίνει μία καθαρή και χωρίς ασάφειες αρχική εικόνα για την ελκυστικότητα της επένδυσης.

Το πρόβλημα χρονοπρογραμματισμού έργων με περιορισμό πόρων (RCPSP-Resource Constrained Project Scheduling Problem) αποτελεί το κύριο προς

επίλυση πρόβλημα για τους διαχειριστές έργων. Στόχος του είναι η ελαχιστοποίηση της διάρκειας ενός έργου χωρίς την παραβίαση των περιορισμών διαθεσιμότητας των πόρων. Το πρόβλημα χρονοπρογραμματισμού έργων με περιορισμό πόρων με προεξοφλημένες ταμειακές ροές (RCPSP - Discounted Cashflows) αποτελεί μία επέκταση του κλασικού προβλήματος του RCPSP και στοχεύει στη μεγιστοποίηση της NPV . Λόγω της εγγενούς πολυπλοκότητας των προβλημάτων αυτών είναι αναγκαία η ανάπτυξη μεθόδων και αλγορίθμων προσαρμοσμένων στα χαρακτηριστικά του επενδυτικού πλάνου και της επιχείρησης που θα συμβάλλουν στην εύρεση ποιοτικών λύσεων.

Στην παρούσα διπλωματική εργασία γίνεται ανάπτυξη ενός μεταευρετικού αλγορίθμου με τη μέθοδο συμπεριφοράς του σμήνους μελισσών κατά τη διαδικασία εύρεσης τροφής για την επίλυση του προβλήματος RCPSP - DC. Αρχικά, γίνεται η μαθηματική μοντελοποίηση του προβλήματος, και η μορφή των δεδομένων που χρησιμοποιεί ο αλγόριθμος. Ακολουθεί η παρουσίαση των βημάτων της εκτέλεσης του αλγορίθμου και αναλύεται ο τρόπος προσομοίωσης του σμήνους μελισσών. Αναπτύσσεται πηγαίος κώδικας σε C++ με τη μέθοδο του σμήνους μελισσών καθώς και με τη μέθοδο της προσομοίωσης απόπτησης (Simulated Annealing). Έπειτα, ακολουθεί η ανάπτυξη γραφικής διεπαφής χρήστη (GUI: Graphical User Interface) με τη χρήση της βιβλιοθήκης wxWidgets. Τέλος, εφαρμόζονται τρεις διαφορετικές προσεγγίσεις για το σμήνος μελισσών, γίνεται σύγκριση των αποτελεσμάτων μεταξύ τους ως προς το κριτήριο της NPV παράλληλα με τη χρήση των πόρων. Η εργασία ολοκληρώνεται με την ανάλυση ευαισθησίας.

Λέξεις - Κλειδιά: Χρονοπρογραμματισμός Έργων, Metaheuristic, Bees Algorithm, NPV , RCPSP, RCPSP - DC, C++, GUI, wxWidgets

ABSTRACT

Bee Swarm algorithm development for solving the problem of RCPSP - DC and implementation in C++.

Thesis submitted to the Department of Production and Management Engineering, School of Engineering, Democritus University of Thrace, Greece, on November 2022 for the degree

Diploma in Production Engineering and Management (Dip. Eng.)

Supervisors: Stefanos Katsavounis, Associate Professor, Georgios K. Koulinas (Dip.Eng, Msc, PhD)

In the contemporary business environment, during the stage of investment planning, enterprises are called to manage an exorbitant amount of information. With the inherent complexity of projects and schedules there is an ambiguity and an uncertainty about the attractiveness of an investment. Making the most suitable decision to ensure the efficiency of the relative investment requires and demands the exploitation of effective

and recognized investment evaluation criteria and designing relative business plans by the managers.

One of the most essential evaluation methods is the Net Present Value evaluation criterion (*NPV*). This method aims to maximize the *NPV* of an investment plan. This method is selected due to the result being a single and unique value that gives a clear and without ambiguity first estimate for the attractiveness of the investment.

The resource constrained project scheduling problem (RCPSP) constitutes the main problem to be solved by the project managers. It focuses on minimizing the duration of a project without exceeding the resource availability. The RCPSP with Discounted Cashflows (RCPSP - DC) is an extension of the classic RCPSP problem and it's solution aims to maximize the *NPV*. Due to the intrinsic complexity of the aforementioned problems the development of methods and algorithms adjusted to the characteristics of an investment plan and its business is a necessity for finding quality solutions.

In the current diploma thesis a metaheuristic algorithm is developed with the method of foraging bees swarm behavior for solving the problem of RCPSP - DC. Initially, the mathematical model of the problem is proposed and the format of the algorithm data is presented. Consequently, the steps of the algorithm execution are presented and the bee swarm foraging emulation is analyzed. Source code is developed in C++ employing the bees swarm foraging method as well as simulated annealing method. Afterwards, the development of the Graphical User Interface (GUI) follows with the use of the library wxWidgets. Finally, three different approaches are implemented for the bees swarm, the results of the *NPV* are measured against each other in parallel with the usage of resources. The thesis is completed with the sensitivity analysis.

Key words: Project Scheduling, Metaheuristic, Bees Algorithm, *NPV*, RCPSP, RCPSP - DC, C++, GUI, wxWidgets

Περιεχόμενα

Ευχαριστίες	i
Περίληψη	iii
Abstract	v
1 Εισαγωγή	1
2 Θεωρητικό Υπόβαθρο	7
2.1 Καθαρή Παρούσα Αξία	8
2.1.1 Η Καθαρή Παρούσα Αξία	8
2.1.2 Η Διαχρονική Αξία του Χρήματος	10
2.2 Υπολογιστική Πολυπλοκότητα	12
2.3 Απλοί Ευρετικοί Αλγόριθμοι	13
2.4 Μεταευρετικοί Αλγόριθμοι	13
2.4.1 Συμπεριφορά Μελισσών	15
2.5 Εργαλεία Προγραμματισμού	16
3 Παρουσίαση του Προβλήματος	19
3.1 Καθορισμός του Προβλήματος	20
3.1.1 Μαθηματική Μοντελοποίηση	20
3.1.2 Παράμετροι για την σχεδίαση του αλγορίθμου	22
3.1.3 Οι κλάσεις του αλγορίθμου	24
3.2 Παρουσίαση του Αλγορίθμου	29
3.3 Προεπισκόπηση Σημαντικών Βημάτων	31
3.3.1 Αρχικός Πληθυσμός	31
3.3.2 Τοπική Έρευνα	32

3.3.3	Καθυστέρηση	34
3.3.4	Έλεγχος Χρονοπρογράμματος	38
3.4	Ανάλυση Πηγαίου Κώδικα	42
3.4.1	Βήμα 1 - Εισαγωγή Δεδομένων	42
3.4.2	Βήμα 2 - Σχεσιακό και Οικονομικό Μοντέλο	42
3.4.3	Βήμα 3 - Αρχικός Πληθυσμός	43
3.4.4	Βήμα 4 - Τοπική Έρευνα και Επιλογή Καλύτερων Λύσεων	48
3.4.5	Βήμα 5 - Καθυστέρηση Δραστηριοτήτων	50
3.4.6	Βήμα 6 - Έλεγχος Χρονοπρογράμματος	54
3.4.7	Βήμα 7 - Εξαγωγή Αποτελεσμάτων	55
3.5	Ανάλυση Simulated Annealing	56
4	Υλοποίηση Γραφικής Διεπαφής Χρήστη - GUI	59
4.1	Προσέγγιση Μέσω Διεπαφής	60
4.2	Οι Κλάσεις του GUI	63
4.2.1	Το αρχείο MainFrame.h	63
4.2.2	Το αρχείο MainFrame.cpp	64
4.2.3	Το αρχείο rcpsp.h	64
4.2.4	Το αρχείο bba.h	65
4.2.5	Το αρχείο chartcontrol.h	65
4.2.6	Το αρχείο chartcontrol.cpp	66
4.3	Χρήση του GUI	66
4.3.1	Εισαγωγή δεδομένων	68
4.3.2	Εκτέλεση του αλγόριθμου	68
4.3.3	Εξαγωγή Αποτελεσμάτων	71
4.3.4	Αρχειοποίηση GUI	72
5	Υπολογιστική Εμπειρία - Αποτελέσματα	75
5.1	Επιλογή Αρχείων	76
5.2	Σμήνος Μελισσών vs Simulated Annealing	78
5.3	Συμπεριφορά Αλγορίθμου	81
5.3.1	Γενική Μελέτη	81
5.3.2	Περιπτώσεις Εστιασμένης Μελέτης	88

5.4	Ανάλυση Ευαισθησίας	105
6	Συμπεράσματα και Περαιτέρω Έρευνα	109
6.1	Συμπεράσματα	110
6.2	Περαιτέρω Έρευνα	111
	Βιβλιογραφία	115
	Α΄ Το αρχείο chartcontrol.h	117
	Β΄ Το αρχείο chartcontrol.cpp	121
	Γ΄ Το αρχείο repsp.h	137
	Δ΄ Το αρχείο bba.h	141
	Ε΄ Το αρχείο MainFrame.h	153
	ΣΤ Το αρχείο MainFrame.cpp	159

Κατάλογος Πινάκων

3.1	Αντιστοιχία μεταβλητών μαθηματικού μοντέλου - δεδομένων πηγαίου κώδικα.	21
3.2	Δεδομένα - Μέλη Rcpcp Class	25
3.3	Δεδομένα - Μέλη In_Pop Class	25
3.4	Δεδομένα - Μέλη Foragers Class	26
3.5	Δεδομένα - Μέλη Best Class	27
3.6	Δεδομένα - Μέλη Elite και Opt Class	27
3.7	Παράδειγμα λίστας και διανυσμάτων προτεραιότητας για πρόβλημα 30 δραστηριοτήτων του αρχείου J301_1.RCP.	32
4.1	Παράδειγμα δομής συναρτήσεων του GUI.	61
5.1	Πίνακας Σύγκρισης Τοπικής Έρευνας με Simulated Annealing.	80
5.2	Αποτελέσματα αρχείων χαμηλής απαίτησης πόρων και χαμηλής Εντατικοποίησης.	82
5.3	Αποτελέσματα αρχείων μέτριας απαίτησης πόρων και χαμηλής Εντατικοποίησης.	82
5.4	Αποτελέσματα αρχείων υψηλής απαίτησης πόρων και χαμηλής Εντατικοποίησης.	83
5.5	Αποτελέσματα αρχείων χαμηλής απαίτησης πόρων και μέτριας Εντατικοποίησης.	83
5.6	Αποτελέσματα αρχείων μέτριας απαίτησης πόρων και μέτριας Εντατικοποίησης.	84
5.7	Αποτελέσματα αρχείων υψηλής απαίτησης πόρων και μέτριας Εντατικοποίησης.	84

5.8	Αποτελέσματα αρχείων χαμηλής απαίτησης πόρων και υψηλής Εντατικοποίησης.	85
5.9	Αποτελέσματα αρχείων μέτριας απαίτησης πόρων και υψηλής Εντατικοποίησης.	85
5.10	Αποτελέσματα αρχείων υψηλής απαίτησης πόρων και υψηλής Εντατικοποίησης.	86
5.11	Αποτελέσματα αρχείων χαμηλής απαίτησης πόρων, υψηλής Εντατικοποίησης και διπλής συνθήκης.	86
5.12	Αποτελέσματα αρχείων μέτριας απαίτησης πόρων, υψηλής Εντατικοποίησης και διπλής συνθήκης.	87
5.13	Αποτελέσματα αρχείων υψηλής απαίτησης πόρων, υψηλής Εντατικοποίησης και διπλής συνθήκης.	87
5.14	Αποτελέσματα αρχείου X16_4 με constant = 100.0.	89
5.15	Αποτελέσματα αρχείου X16_4 με constant = 350.0.	92
5.16	Αποτελέσματα αρχείου X16_4 με constant = 700.0.	94
5.17	Αποτελέσματα αρχείου X54_4 με constant = 20.0.	97
5.18	Αποτελέσματα αρχείου X54_4 με constant = 300.0.	100
5.19	Αποτελέσματα αρχείου X54_4 με constant = 700.0.	102
5.20	Ποσοστά χρήσης πόρων.	105
5.21	Αποτελέσματα ανάλυσης ευαισθησίας αρχείου X16_4 με constant = 100.0.	106
5.22	Αποτελέσματα ανάλυσης ευαισθησίας αρχείου X16_4 με constant = 200.0.	107
5.23	Αποτελέσματα ανάλυσης ευαισθησίας αρχείου X16_4 με constant = 300.0.	108

Κατάλογος Σχημάτων

2.1	Μελλοντική αξία επένδυσης 10.000 € με ανατοκισμό ύψους 5%.	11
3.1	Ενδεικτικό σχήμα δικτύου AoN, $N = 7$ δραστηριοτήτων.	20
3.2	Διάταξη αρχείου 30 δραστηριοτήτων, J301_1.RCP.	24
3.3	Διάγραμμα κλάσεων	28
3.4	Διάγραμμα Ροής του Αλγορίθμου.	30
3.5	Διάγραμμα Ροής Εμπρόσθιας Σειριακής Μεθόδου Παραγωγής Χρονοπρογραμμάτων	31
3.6	Ενδεικτικό σχήμα πρώτου και δεύτερου τρόπου αλλαγής προτεραιότητας	32
3.7	Εύρεση καλύτερης λύσης και επανάληψη Τοπικής Έρευνας	33
3.8	Αναλυτικό διάγραμμα ροής υπολογισμού <i>set1</i> και <i>set2</i> με βάση τις σχέσεις εξάρτησης. Πηγή: (Leyman & Vanhoucke, 2015).	35
3.9	Αναλυτικό διάγραμμα ροής υπολογισμού <i>set1</i> βάση τους χρόνους λήξης. Πηγή: (Leyman & Vanhoucke, 2015).	37
3.10	Αναλυτικό διάγραμμα ροής υπολογισμού <i>set2</i> βάση τους χρόνους λήξης. Πηγή: (Leyman & Vanhoucke, 2015).	38
3.11	Διάγραμμα ροής Ελέγχου Χρονοπρογράμματος	38
3.12	Παράδειγμα χρονοπρογράμματος 30 δραστηριοτήτων, αρχείο J3048_9.RCP, πριν την κλήση της Check_Schedule.	40
3.13	Παράδειγμα χρονοπρογράμματος 30 δραστηριοτήτων, αρχείο J3048_9.RCP, μετά την κλήση της Check_Schedule.	41
3.14	Διάγραμμα Ροής - Simulated Annealing.	57
4.1	Πίνακας Γεγονότων.	61
4.2	Η εντολή Bind.	62
4.3	Το GUI του αλγορίθμου.	66

4.4	Το GUI του αλγορίθμου εστιασμένο.	67
4.5	Μπάρα επιλογών (MenuBar) του GUI.	68
4.6	Το κουμπί "Run" ενεργοποιημένο.	69
4.7	Ράβδοι προόδου του GUI.	70
4.8	Το GUI του αλγορίθμου μετά την ολοκλήρωση του υπολογισμού των λύσεων.	70
4.9	Προσωρινά παράθυρα για την ανάθεση τιμών επιτοκίων.	71
4.10	Επιλογή διαχωρισμού τιμών.	72
4.11	Το κουμπί "Initialize Solution" ενεργοποιημένο.	73
5.1	Ενδεικτική μορφή αρχείου αποτελεσμάτων.	78
5.2	Διάγραμμα χρήσης πόρου R1 αρχείου X16_4 με constant = 100.0. . . .	90
5.3	Διάγραμμα χρήσης πόρου R2 αρχείου X16_4 με constant = 100.0. . . .	90
5.4	Διάγραμμα χρήσης πόρου R3 αρχείου X16_4 με constant = 100.0. . . .	91
5.5	Διάγραμμα χρήσης πόρου R4 αρχείου X16_4 με constant = 100.0. . . .	91
5.6	Διάγραμμα χρήσης πόρου R1 αρχείου X16_4 με constant = 350.0. . . .	92
5.7	Διάγραμμα χρήσης πόρου R2 αρχείου X16_4 με constant = 350.0. . . .	93
5.8	Διάγραμμα χρήσης πόρου R3 αρχείου X16_4 με constant = 350.0. . . .	93
5.9	Διάγραμμα χρήσης πόρου R4 αρχείου X16_4 με constant = 350.0. . . .	94
5.10	Διάγραμμα χρήσης πόρου R1 αρχείου X16_4 με constant = 700.0. . . .	95
5.11	Διάγραμμα χρήσης πόρου R2 αρχείου X16_4 με constant = 700.0. . . .	95
5.12	Διάγραμμα χρήσης πόρου R3 αρχείου X16_4 με constant = 700.0. . . .	96
5.13	Διάγραμμα χρήσης πόρου R4 αρχείου X16_4 με constant = 700.0. . . .	96
5.14	Διάγραμμα χρήσης πόρου R1 αρχείου X54_4 με constant = 20.0. . . .	98
5.15	Διάγραμμα χρήσης πόρου R2 αρχείου X54_4 με constant = 20.0. . . .	98
5.16	Διάγραμμα χρήσης πόρου R3 αρχείου X54_4 με constant = 20.0. . . .	99
5.17	Διάγραμμα χρήσης πόρου R4 αρχείου X54_4 με constant = 20.0. . . .	99
5.18	Διάγραμμα χρήσης πόρου R1 αρχείου X54_4 με constant = 300.0. . . .	100
5.19	Διάγραμμα χρήσης πόρου R2 αρχείου X54_4 με constant = 300.0. . . .	101
5.20	Διάγραμμα χρήσης πόρου R3 αρχείου X54_4 με constant = 300.0. . . .	101
5.21	Διάγραμμα χρήσης πόρου R4 αρχείου X54_4 με constant = 300.0. . . .	102
5.22	Διάγραμμα χρήσης πόρου R1 αρχείου X54_4 με constant = 700.0. . . .	103
5.23	Διάγραμμα χρήσης πόρου R2 αρχείου X54_4 με constant = 700.0. . . .	103

5.24	Διάγραμμα χρήσης πόρου R3 αρχείου X54_4 με $\text{constant} = 700.0$	104
5.25	Διάγραμμα χρήσης πόρου R4 αρχείου X54_4 με $\text{constant} = 700.0$	104
5.26	Διάγραμμα NPV - Επιτοκίων με $\text{constant} = 100.0$	106
5.27	Διάγραμμα NPV - Επιτοκίων με $\text{constant} = 200.0$	107
5.28	Διάγραμμα NPV - Επιτοκίων με $\text{constant} = 300.0$	108

ΚΕΦΑΛΑΙΟ

1

ΕΙΣΑΓΩΓΗ

Κάθε επιχείρηση έχει ως απώτερο σκοπό της αύξηση της αντικειμενικής της αξίας και των εσόδων της με οποιονδήποτε δυνατό τρόπο. Η επίτευξη του στόχου αυτού γίνεται από την ομαλή και σωστή λειτουργία της επιχείρησης, την τεχνογνωσία των μελών της, την τεχνολογία των υποδομών, την διοίκηση και τις σωστές λήψεις αποφάσεων από τα διοικητικά μέλη. Ένα άκρως ανταγωνιστικό περιβάλλον τόσο σε εθνικό, όσο και σε παγκόσμιο επίπεδο, προτρέπει τις επιχειρήσεις να προχωρούν στη ανάπτυξη και ενσωμάτωση σύγχρονων μοντέλων αξιολόγησης, μεθόδων σχεδιασμού και υλοποίησης χρονοπρογραμματισμού ώστε να ληφθούν οι σωστές αποφάσεις και να μπορέσουν να ανταπεξέλθουν στις απαιτήσεις της αγοράς.

Ένα από τα πιο βασικά μοντέλα αξιολόγησης είναι το κριτήριο αξιολόγησης της Καθαρής Παρούσας Αξίας (*NPV*: Net Present Value). Η χρήση του μοντέλου αυτού προτιμάται ιδιαιτέρως διότι στόχος του είναι η μεγιστοποίηση της παρούσας αξίας του κεφαλαίου ενός επενδυτικού πλάνου, λαμβάνει βασικές έννοιες υπόψη όπως η διαχρονική αξία του χρήματος μέσω του προεξοφλητικού επιτοκίου και επιτρέπει τη χρήση κυμαινόμενου επιτοκίου. Η αξιολόγηση της *NPV* οδηγεί σε μία και μοναδική τιμή η οποία αποτελεί έναν σαφή δείκτη για την επένδυση. Συνήθως χρησιμοποιείται παράλληλα με το μοντέλο του Εσωτερικού Βαθμού Απόδοσης (*IRR*: Internal Rate of Return) του οποίου ο στόχος είναι η εύρεση του προεξοφλητικού επιτοκίου που μηδενίζει την *NPV* ενός επενδυτικού πλάνου.

Το πρόβλημα χρονοπρογραμματισμού έργων με περιορισμό πόρων με προεξοφλημένες ταμειακές ροές (*RCPS* - *DC*) αποτελεί μια επέκταση του κλασικού προβλήματος του *RCPS* και στοχεύει στην μεγιστοποίηση της *NPV*. Η αποτελεσματική αξιοποίηση των πόρων σε υψηλό ποσοστό κατά την διάρκεια εκτέλεσης του χρονοπρογράμματος αποτελεί αναπόσπαστο κομμάτι της μελέτης ενός επενδυτικού πλάνου. Είτε αφορά εργατικό δυναμικό είτε ειδικό εξοπλισμό ή υποδομές, το πρόβλημα *RCPS* - *DC* στοχεύει στην υψηλή αξιοποίηση πόρων σε συνδυασμό με την μεγιστοποίηση της *NPV* και την αποφυγή υπέρβασης ορίων διαθεσιμότητας πόρων.

Λόγω της έμφυτης *NP* - *Hard* πολυπλοκότητας καθιστά αναγκαία την ανάπτυξη

και χρήση ευρετικών μεθόδων και αλγορίθμων προσαρμοσμένων στα χαρακτηριστικά του επενδυτικού πλάνου και της επιχείρησης. Η εύρεση της βέλτιστης κατανομής των πόρων και ακολουθίας εκτέλεσης των δραστηριοτήτων πραγματοποιείται με την χρήση κανόνων προτεραιότητας. Το αποτέλεσμα της εφαρμογής των κανόνων αυτών εξαρτάται από τις σχέσεις εξάρτησης των δραστηριοτήτων καθώς και από τις διαθεσιμότητες πόρων.

Το πρώτο κεφάλαιο εισάγει τον αναγνώστη στο αντικείμενο που εξετάζεται στην παρούσα διπλωματική εργασία. Αναφέρονται γενικές πληροφορίες σχετικά για την σημαντικότητα του αντικειμένου και περιγράφεται συνοπτικά η δομή της εργασίας.

Το δεύτερο κεφάλαιο είναι αφιερωμένο στο θεωρητικό υπόβαθρο που χρειάζεται να κατέχει ο αναγνώστης για την κατανόηση του αντικειμένου, του προβλήματος προς επίλυση, του τρόπου λειτουργίας του αλγορίθμου και του περιβάλλοντος υλοποίησής του. Αποτελείται από πέντε ενότητες.

- Στην πρώτη ενότητα γίνεται αναφορά σε όρους οικονομικού περιεχομένου, που είναι απαραίτητοι για την κατανόηση του κριτηρίου της NPV .
- Στη δεύτερη ενότητα επεξηγείται η ιδιαιτερότητα των προβλημάτων που παρουσιάζουν πολυπλοκότητα $NP - Hard$.
- Στην τρίτη και τέταρτη ενότητα αναφέρονται η φύση και παραδείγματα ευρετικών και μεταευρετικών αλγορίθμων αντίστοιχα και αναλύεται η συμπεριφορά κατά τη διαδικασία εύρεσης τροφής του σμήνους μελισσών.
- Τέλος, στην πέμπτη ενότητα γίνεται η παρουσίαση του περιβάλλοντος υλοποίησης του αλγορίθμου και τα εργαλεία προγραμματισμού.

Στο τρίτο κεφάλαιο γίνεται η παρουσίαση του προβλήματος, αποτελείται από πέντε ενότητες.

- Στην πρώτη ενότητα γίνεται ο καθορισμός του προβλήματος, η μαθηματική μοντελοποίηση του προβλήματος, αναφορά σε σημαντικές παραμέτρους και παρουσιάζονται οι κλάσεις του αλγορίθμου.

- Στη δεύτερη ενότητα παρουσιάζονται τα βήματα του αλγορίθμου
- Στην τρίτη ενότητα γίνεται μία περιφραστική ανάλυση των σημαντικών βημάτων του αλγορίθμου.
- Στην τέταρτη ενότητα υπάρχει η ανάλυση του πηγαίου κώδικα
- Στην πέμπτη ενότητα αναλύεται ο δευτερεύων αλγόριθμος της προσομοιωμένης ανόπτησης που υλοποιήθηκε για τη σύγκριση των αποτελεσμάτων.

Το τέταρτο κεφάλαιο είναι αφιερωμένο στην ανάδειξη και παρουσίαση της Γραφικής Διεπαφής Χρήστη (GUI) και αποτελείται από τρεις ενότητες.

- Στην πρώτη ενότητα γίνεται η σύγκριση και διαφοροποίηση του διαδικαστικού προγραμματισμού με τον προγραμματισμό χειρισμού γεγονότων.
- Στη δεύτερη ενότητα παρουσιάζονται τα αρχεία που αποτελούν το GUI καθώς και το περιεχόμενό τους
- Στην τρίτη ενότητα παρουσιάζεται βηματικά η χρήση του GUI.

Στο πέμπτο κεφάλαιο παρουσιάζεται η υπολογιστική εμπειρία και τα αποτελέσματα που προέκυψαν από την εκτέλεση του αλγορίθμου. Αποτελείται από τρεις ενότητες:

- Η πρώτη ενότητα περιέχει τα κριτήρια επιλογής των αρχείων δεδομένων που χρησιμοποιήθηκαν και γίνεται αναφορά στην μορφή των εξαγόμενων αρχείων με τα αποτελέσματα.
- Η δεύτερη ενότητα περιέχει τα αποτελέσματα από τη σύγκριση των αλγορίθμων του σμήνους μελισσών με τον αλγόριθμο προσομοίωσης ανόπτησης
- Η τρίτη και τελευταία ενότητα περιέχει τα αποτελέσματα από την εκτέλεση του πηγαίου κώδικα με τα επιλεγμένα αρχεία δεδομένων που αναφέρονται στην πρώτη ενότητα του κεφαλαίου.

Στο έκτο κεφάλαιο περιέχονται τα συμπεράσματα από την εκπόνηση της διπλωματικής εργασίας και γίνονται προτάσεις για περαιτέρω έρευνα πάνω στο

αντικείμενο.

Στο παράρτημα παρατίθεται ο πηγαίος κώδικας της γραφικής διεπαφής χρήστη του αλγορίθμου σε γλώσσα C++.

ΚΕΦΑΛΑΙΟ

— 2 —

ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

2.1 Καθαρή Παρούσα Αξία

2.1.1 Η Καθαρή Παρούσα Αξία

Μία επιχείρηση έχει ως απώτερο σκοπό τη μεγιστοποίηση του πραγματικού, καθώς και του πιθανού, κέρδους που μπορεί να παράξει. Για τη μεγιστοποίηση του κέρδους, η επιχείρηση θα προβεί σε επενδύσεις σε διάφορους τομείς που άπτονται του αντικειμένου της, οι οποίες δύναται να είναι από εξαιρετικά κερδοφόρες έως καταστροφικές για την ίδια, καθώς και τα στελέχη που τη διοικούν. Λόγω αυτής της αβεβαιότητας εφαρμόζονται συγκεκριμένα μοντέλα αξιολόγησης επενδύσεων, είτε εμπειρικά μοντέλα (Βαβάτσικος, 2020):

- Απόδοσης Επένδυσης (ROI: Return On Investmen) και
- Περίοδος Αποπληρωμής Επενδυμένων Κεφαλαίων (Payback Period).

είτε χρηματοοικονομικά μοντέλα:

- Προεξοφλημένη Περίοδος Αποπληρωμής Επενδεδυμένων Κεφαλαίων (Discounted Payback Period),
- Αναλύσεις Οφέλους Κόστους (Benefit/Cost Analysis Ratios),
- Εσωτερικός Βαθμός (ποσοστό) Απόδοσης (IRR: Internal Rate of Return),
- Τροποποιημένος Εσωτερικός Βαθμός Απόδοσης (MIRR: Modified Internal Rate of Return) και
- και Καθαρή Παρούσα Αξία (NPV: Net Present Value)

Το συχνότερο και κυριότερο εργαλείο για την αξιολόγηση μιας επένδυσης (Remer & Nieto, 1995) είναι η Ανάλυση Προεξοφλημένων Ταμειακών Ροών (Discount Cash Flow Analysis), η οποία επιτυγχάνεται με δύο από τα προαναφερθέντα χρηματοοικονομικά μοντέλα, τα: IRR και NPV, όπου το τελευταίο είναι το μοντέλο που υλοποιείται στην παρούσα εργασία. Ο ορισμός της Καθαρής Παρούσας Αξίας, πλέον θα αναφέρεται ως NPV, έχει λάβει διάφορες και πολυάριθμες προσεγγίσεις, όμως σύμφωνα με τους (Αραβώσης κ.ά., 2011), ο ορισμός που αποδίδουν είναι: Η τιμή που προκύπτει από την αφαίρεση του συνόλου των προεξοφλημένων ταμειακών εισροών μίας επένδυσης

από το σύνολο των αντίστοιχων εκροών. Μαθηματικά ο ορισμός της NPV (Crundwell, 2008) αποδίδεται ως:

$$NPV = \sum_{t=0}^n \frac{CF_t}{(1+i)^t} \quad (2.1)$$

Όπου:

- t η χρονική περίοδος της επένδυσης,
- CF_t η χρηματοροή (Cash Flow) τη χρονική περίοδο t ,
- n ο κύκλος ζωής της επένδυσης και
- i το προεξοφλητικό επιτόκιο.

Για τα επενδυτικά έργα που υλοποιούνται είτε από δημόσιους, είτε από ιδιωτικούς φορείς υφίστανται τεχνικές αξιολόγησης τόσο πριν την υλοποίηση της επένδυσης, όσο και κατά τη διάρκειά της. Η NPV αποτελεί μία τεχνική αξιολόγησης που πραγματοποιείται κατά τη φάσης σύλληψης και σχεδίασης ενός επενδυτικού έργου, οπότε έχει ως σκοπό να αξιολογήσει τη βιωσιμότητά του και να παρέχει πληροφορίες στους αρμόδιους λήπτες αποφάσεων προκειμένου να αποφασίσουν αν η υπό μελέτη επένδυση είναι πραγματοποιήσιμη ή όχι. Τα εύρη τιμών που μπορεί να έχει η NPV είναι τρία: θετική, μηδέν και αρνητική. Σε κάθε περίπτωση αξιολόγησης επιδιώκεται η τιμή της NPV να είναι θετική. Η αξιολόγηση αυτή μπορεί να εκφραστεί ως εξής:

- $NPV > 0$. Το εύρος αυτό υποδηλώνει πως τα προβλεπόμενα κέρδη της επένδυσης υπερβαίνουν το αναμενόμενο κόστος της σε τρέχουσες χρηματικές μονάδες.
- $NPV = 0$. Η επένδυση είναι αδιάφορη, κοινώς δεν αποδίδει κέρδος, ωστόσο ούτε αποτελεί ζημιά για την επιχείρηση.
- $NPV < 0$. Το εύρος αυτό υποδηλώνει πως η εξεταζόμενη επένδυση θα έχει καταστροφικές συνέπειες για την επιχείρηση και δεν θα πρέπει να ληφθεί υπόψιν.

Ωστόσο, μία θετική NPV δεν υπονοεί αυτόματα πως αποτελεί καλή και ποιοτική επιλογή. Η ποιότητα της αξιολόγησης βασίζεται άμεσα στην ποιότητα των προβλέψεων και των υπολογισμών των χρηματοροών, καθώς και του προεξοφλητικού επιτοκίου.

2.1.2 Η Διαχρονική Αξία του Χρήματος

Αναφέρθηκε προηγουμένως, στα εύρη τιμών της NPV, ότι αυτή χρησιμοποιεί τρέχουσες (σημερινές) χρηματικές μονάδες. Για να γίνει κατανοητή η έννοια αυτή, η χρήση και η σημαντικότητα της NPV, πρέπει σε πρώτο στάδιο να αναλυθεί η αξία του χρήματος σε συνάρτηση με τον χρόνο και έπειτα το προεξοφλητικό επιτόκιο.

Μελλοντική Αξία και Ανατοκισμός

Πρωταρχικό χαρακτηριστικό στη φάση σχεδίασης ενός επενδυτικού έργου και της αρχικής αξιολόγησής του, είναι ο καθορισμός μίας διορίας (deadline - χρονικό περιθώριο) εντός της οποίας θα ολοκληρωθεί το εν λόγω έργο. Δεδομένου ότι η υλοποίηση μιας επένδυσης απαιτεί τη δέσμευση ενός μεγάλου κεφαλαίου, είναι λογικό οι επενδύσεις να αξιολογούνται σε βάθος χρόνου. Για το λόγο αυτό, εφαρμόζονται τεχνικές μετατροπής χρηματικών ποσών σε μελλοντική αξία μέσω ανατοκισμού σύμφωνα με τη θεωρία της διαχρονικής αξίας των χρημάτων (time value of money). Η θεωρία της διαχρονικής αξίας των χρημάτων είναι μια βασική οικονομική αρχή η οποία δηλώνει πως μία χρηματική μονάδα του παρόντος αξίζει περισσότερο από την ίδια χρηματική μονάδα στο μέλλον λόγω μεταβλητών όπως ο πληθωρισμός και τα επιτόκια. (Lioudis, 2022)

Με σκοπό να γίνει πιο κατανοητή η έννοια της μελλοντικής αξίας, θα χρησιμοποιηθεί το εξής παράδειγμα: « Έστω μία επένδυση αξίας 10.000€ με σύνθετο τόκο 5%. Ποια αξία της επένδυσης θα είναι η αξία της επένδυσης μετά από 10 χρόνια;». Μαθηματικά η μελλοντική αξία ορίζεται ως:

$$FV = PV \times (1 + r)^n \quad (2.2)$$

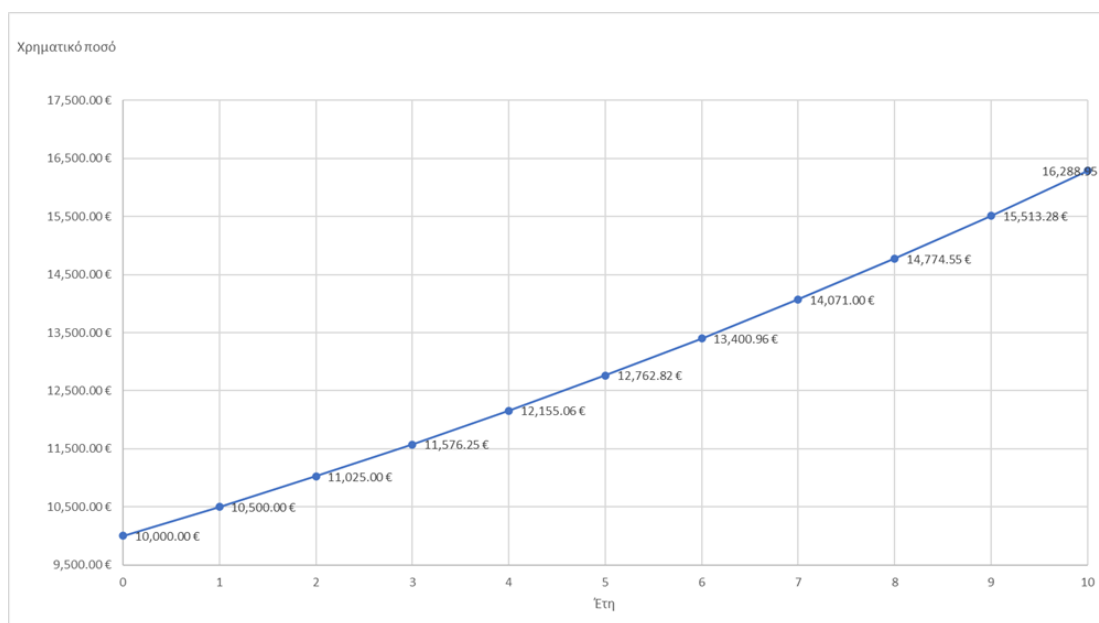
Όπου:

- FV η μελλοντική αξία,
- PV η παρούσα αξία,
- r το επιτόκιο και

- n η περίοδος ανατοκισμού.

Οπότε σύμφωνα με την Εξίσωση 2.2, αν η περίοδος ανατοκισμού είναι ο ένας χρόνος, τότε η αξία της επένδυσης θα είναι:

- Μετά από 1 χρόνο θα είναι ίση με $FV = 10.000 \times (1 + 0.05)^1 = 10.500 \text{ €}$.
- Μετά από 2 χρόνια θα είναι ίση με $FV = 10.000 \times (1 + 0.05)^2 = 11.025 \text{ €}$.
- Μετά από 10 χρόνια θα είναι ίση με $FV = 10.000 \times (1 + 0.05)^{10} = 16.288,95 \text{ €}$.



Σχήμα 2.1: Μελλοντική αξία επένδυσης 10.000 € με ανατοκισμό ύψους 5%.

Μέσο Σταθμικό Κόστος Κεφαλαίου

Ο καθορισμός του επιτοκίου δεν είναι πάντα ξεκάθαρος και γι' αυτόν το λόγο πολλές επιχειρήσεις υπολογίζουν το Μέσο Σταθμικό Κόστος Κεφαλαίου (WACC: Weighted Average Cost of Capital) και το χρησιμοποιούν ως το προεξοφλητικό τους επιτόκιο (Majaski, 2022). Η τιμή του εξαρτάται άμεσα από το είδος της χρηματοδότησης της επένδυσης. Η χρηματοδότηση αυτή συνήθως αντλείται από Ίδια κεφάλαια E με κόστος κεφαλαίου R_E και μέσω δανεισμού κεφαλαίου D με κόστος δανεισμού R_D .

(Βαβάτσικος, 2020).

Ως Μέσο Σταθμικό Κόστος Κεφαλαίου ορίζεται η ελάχιστη απόδοση που αναμένεται να καταβάλλει η επιχείρηση κατά μέσο όρο στους μετόχους της για τη χρηματοδότηση των περιουσιακών της στοιχείων. Το $WACC$ υπολογίζεται ως εξής:

$$WACC = \left(\frac{E}{E + D} \times R_E \right) + \left(\frac{D}{E + D} \times R_D \times (1 - T_C) \right) \quad (2.3)$$

Όπου:

- E (Equity), Ίδια κεφάλαια,
- R_E κόστος κεφαλαίου,
- D (Debt), κεφάλαιο Δανεισμού
- R_D κόστος δανεισμού και
- T_C ο φορολογικός συντελεστής (Hargrave, 2022).

2.2 Υπολογιστική Πολυπλοκότητα

Στη θεωρία της υπολογιστικής πολυπλοκότητας, ως NP πολυπλοκότητα (Non Polynomial Complexity - μη αιτιοκρατικός (non-deterministic) πολυώνυμος χρόνος) ορίζεται η καθοριστική ιδιότητα μιας κατηγορίας προβλημάτων που είναι ανεπίσημα "τουλάχιστον τόσο σκληρά όσο τα πιο δύσκολα προβλήματα NP". Ως NP ορίζεται το σύνολο των μη αιτιοκρατικών αλγορίθμων που επιλύονται σε πολυώνυμο χρόνο ενώ P ορίζεται ως το σύνολο των αιτιοκρατικών αλγορίθμων που επιλύονται σε πολυώνυμο χρόνο (Garey & Johnson, 1979).

Μερικά παραδείγματα προβλημάτων πολυωνυμικής (P) πολυπλοκότητας είναι:

- Linear Search, με χρονική πολυπλοκότητα n ,
- Binary Search, με χρονική πολυπλοκότητα $\log n$ και
- Merge Sort, με χρονική πολυπλοκότητα $n \log n$

Ως NP - Hard ορίζονται τα μη αιτιοκρατικά προβλήματα με εκθετική χρονική πολυπλοκότητα για τα οποία όχι μόνο δεν είναι γνωστός κάποιος αλγόριθμος για την επίλυση του προβλήματος, αλλά δεν ενδέχεται να είναι πιθανή η ύπαρξη ενός τέτοιου αλγορίθμου. Τα NP - Hard για τα οποία έχουν αναπτυχθεί μη αιτιοκρατικοί αλγόριθμοι για την επίλυσή τους ορίζονται ως NP - Complete.

2.3 Απλοί Ευρετικοί Αλγόριθμοι

Οι απλοί Ευρετικοί Αλγόριθμοι αποτελούν μία τεχνική εύρεσης λύσεων σε προβλήματα NP-Complete και NP-Hard οι οποίοι εξάγουν «καλά» αποτελέσματα συγκριτικά με συμβατικές μεθόδους και χρησιμοποιούνται συνήθως σε προβλήματα με υπολογιστική πολυπλοκότητα NP - Complete και NP-Hard. Είναι σχεδιασμένοι ώστε να μπορούν να παράξουν λύσεις γρήγορα θυσιάζοντας την ακρίβεια και την επιδίωξη της βέλτιστης λύσης. Οι αλγόριθμοι αυτοί, παρότι δεν αποτελούν μεθόδους βελτιστοποίησης, μπορούν μεμονωμένα να παράξουν ποιοτική λύση ή να χρησιμοποιηθούν ως βάση σε συνδυασμό με έναν αλγόριθμο βελτιστοποίησης (Cook, 1983).

Οι κατηγορίες των Ευρετικών Αλγορίθμων είναι οι εξής:

1. Άπληστοι Αλγόριθμοι (greedy algorithms),
2. Προσεγγιστικοί αλγόριθμοι (approximation algorithms) και
3. Αλγόριθμοι τοπικής αναζήτησης (local search algorithms).

Οι δύο πρώτες κατηγορίες αλγορίθμων παράγουν μία λύση ενώ η τρίτη κατηγορία αλγορίθμων αποτελεί μια μέθοδο βελτίωσης μιας προϋπάρχουσας λύσης (Μαρινάκης κ.ά., 2011).

2.4 Μεταευρετικοί Αλγόριθμοι

Με την εφαρμογή της μεθόδου της τοπικής αναζήτησης ο αλγόριθμος έχει την τάση να συγκλίνει γύρω από τη γειτονία μιας λύσης. Η μέθοδος αυτή επίσης παρουσιάζει μεγάλη ευαισθησία στην αρχική λύση η οποία έχει την ικανότητα, ανάλογα με την ποιότητά της, να κατευθύνει τον αλγόριθμο σε ένα τοπικό βέλτιστο και να εγκλωβίζεται

εκεί. Ο εγκλωβισμός αυτός αποτελεί πρόβλημα καθώς καθιστά τον αλγόριθμο ανίκανο να βελτιώσει περαιτέρω τη λύση του προβλήματος.

Για τον απεγκλωβισμό από τα τοπικά βέλτιστα έχουν εφαρμοστεί πολλοί αλγόριθμοι, ονομάζονται μεταευρετικοί ή μεθευρετικοί αλγόριθμοι. Ο διαχωρισμός των κατηγοριών των μεταευρετικών αλγορίθμων γίνεται με κριτήριο το πλήθος των λύσεων (Μαρινάκης κ.ά., 2011):

1. Χρήση μίας λύσης και αναζήτηση στη γειτονιά αυτής της λύσης,
2. Χρήση πληθυσμού λύσεων και αναζήτηση σε όλο το χώρο λύσεων και
3. Υβριδικές μορφές των παραπάνω (hybrid ή memetic αλγόριθμοι).

Οι αλγόριθμοι μίας λύσης και αναζήτησης στη γειτονιά της λύσης διαχωρίζονται σε τέσσερις κατηγορίες ανάλογα με τον τρόπο αποφυγής εγκλωβισμού σε τοπικό βέλτιστο:

1. Επαναληπτικές διαδικασίες που ξεκινούν από διαφορετικές αρχικές λύσεις:
 - Αλγόριθμοι πολυεναρκτήριας τοπικής αναζήτησης (multistart local search),
 - Αλγόριθμοι επαναληπτικής τοπικής αναζήτησης (iterated local search) και
 - Διαδικασία άπληστης τυχαιοποιημένης προσαρμοστικής αναζήτησης (GRASP: Greedy Randomized Adaptive Search Procedure),
2. Αποδοχή χειρότερης γειτονικής λύσης που δεν βελτιώνουν τη λύση:
 - Προσομοίωση ανόπτησης (Simulated Annealing) και
 - Περιορισμένη αναζήτηση (Tabu Search).
3. Αλλαγή στη γειτονιά αναζήτησης:
 - Αλγόριθμος μεταβλητής γειτονιάς αναζήτησης (VNS: Variable Neighborhood Search).
 - Αλγόριθμος επέκτασης της γειτονιάς αναζήτησης (ENS: Expanding Neighborhood Search).
4. Αλλαγή αντικειμενικής συνάρτησης ή δεδομένων του προβλήματος:

- Αλγόριθμος καθοδηγούμενης τοπικής αναζήτησης (Guided Local Search).

Οι μεταεureτικοί αλγόριθμοι που χρησιμοποιούν πληθυσμό λύσεων είναι έντονα εμπνευσμένοι από τις φυσικές διεργασίες, συγκεκριμένα διακρίνονται σε δύο μεγάλες κατηγορίες αλγορίθμων με τη δεύτερη να έχει πέντε υποκατηγορίες:

1. Εξελικτικοί και Γενετικοί Αλγόριθμοι και
2. Νοημοσύνη Σμήνους και Αλγόριθμοι Εμπνευσμένοι από τη Φύση:
 - i. Αλγόριθμος βελτιστοποίησης Αποικίας Μυρμηγκιών,
 - ii. Αλγόριθμος βελτιστοποίησης Σμήνους Σωματιδίων (Particle Swarm Optimization),
 - iii. Αλγόριθμοι που βασίζονται σε Συμπεριφορές Μελισσών,
 - iv. Αλγόριθμοι Τεχνητών Ανοσοποιητικών Συστημάτων (Artificial Immune Systems) και
 - v. Αλγόριθμοι εμπνευσμένοι από Φυσικές Διεργασίες.

2.4.1 Συμπεριφορά Μελισσών

Οι αλγόριθμοι νοημοσύνης σμήνους που βασίζονται στη συμπεριφορά των μελισσών χωρίζονται σε δύο κατηγορίες που αφορούν φυσικές διαδικασίες:

1. Συμπεριφορά κατά τη διάρκεια αναζήτησης τροφής (foraging) και
2. Συμπεριφορά κατά τη διάρκεια ζευγαρώματος (mating).

Η παρούσα διπλωματική πραγματεύεται την ανάπτυξη ενός αλγορίθμου βασισμένου στη συμπεριφορά των μελισσών κατά τη διάρκεια αναζήτησης τροφής. Για να γίνει κατανοητός ο τρόπος λειτουργίας του σμήνους μελισσών πρέπει να αναλυθεί η διαδικασία όπως πραγματοποιείται στη φύση.

Αρχικά οι μέλισσες φεύγουν από την κυψέλη για να ξεκινήσουν την αναζήτηση της τροφής. Μόλις εντοπίσουν την τροφή, λόγω μεγέθους, δεν είναι ικανές να τη συλλέξουν μόνες τους και να τη μεταφέρουν στην κυψέλη, επιστρέφουν, ενημερώνουν και προσπαθούν να πείσουν τις υπόλοιπες μέλισσες να βοηθήσουν στην περισυλλογή

της τροφής. Η ενημέρωση γίνεται μέσω μια διαδικασίας ειδικών κινήσεων που αναφέρεται ως χορός (waggle dance).

Κατά τη διάρκεια του χορού η κατεύθυνση που έχει η μέλισσα προσδιορίζει την κατεύθυνση της τροφής σε σχέση με τη θέαση του ήλιου, η ένταση των κινήσεων προσδιορίζει την απόσταση από την κυψέλη και η διάρκεια την ποσότητα της τροφής. Ακόμα, η μέλισσα ακουμπά με τις κεραίες τις όσο το δυνατόν περισσότερες ώστε να μάθουν το άρωμα και τη γεύση της τροφής.

Στην παρούσα διπλωματική η έννοια του χορού ενσωματώθηκε στο κριτήριο της αντικειμενικής συνάρτησης και θεωρήθηκε ως δεδομένη η στρατολόγηση σταθερού πλήθους μελισσών με την εύρεση καλύτερης λύσης.

2.5 Εργαλεία Προγραμματισμού

Ο αλγόριθμος που αναπτύχθηκε κατά την εκπόνηση της διπλωματικής εργασίας υλοποιήθηκε στο περιβάλλον Microsoft Visual Studio 2019 και η γλώσσα που επιλέχθηκε ήταν η C++, συγκεκριμένα η έκδοση C++20 Standard. Για την ενσωμάτωση παράλληλου προγραμματισμού και τη δημιουργία των threads χρησιμοποιήθηκε η διεπαφή προγραμματισμού εφαρμογών (API: Application Programming Interface) OpenMP (Open Multi-Processing).

Η OpenMP αποτελεί μία διεπαφή η οποία υποστηρίζει τον προγραμματισμό πολυεπεξεργασίας κοινής μνήμης σε πολλαπλές πλατφόρμες σε γλώσσες C, C++ και Fortran. Αποτελείται από ένα σύνολο οδηγιών του compiler, βιβλιοθήκες και περιβαλλοντικές μεταβλητές που επηρεάζουν τον χρόνο εκτέλεσης του προγράμματος. Η υποστήριξη της OpenMP μπορεί να γίνει κατευθείαν από τις ρυθμίσεις του Visual Studio και δεν απαιτεί την εγκατάσταση βιβλιοθηκών από εξωτερικές πηγές.

Για την υλοποίηση της γραφικής διεπαφής χρήστη, χρησιμοποιήθηκε η εξωτερική βιβλιοθήκη wxWidgets. Η βιβλιοθήκη wxWidgets αποτελεί μια διεπαφή προγραμματισμού εφαρμογών για την ανάπτυξη γραφικών διεπαφών οι οποίες

υιοθετούν τα στοιχεία και τους χειρισμούς του εκάστοτε λειτουργικού συστήματος στο οποίο αναπτύσσεται. Ακόμα, οι διεπαφές που έχουν αναπτυχθεί με την wxWidgets έχουν υποστήριξη ανεξαρτήτως πλατφόρμας και μπορούν να λειτουργήσουν χωρίς αλλαγή στον πηγαίο κώδικά τους.

ΚΕΦΑΛΑΙΟ

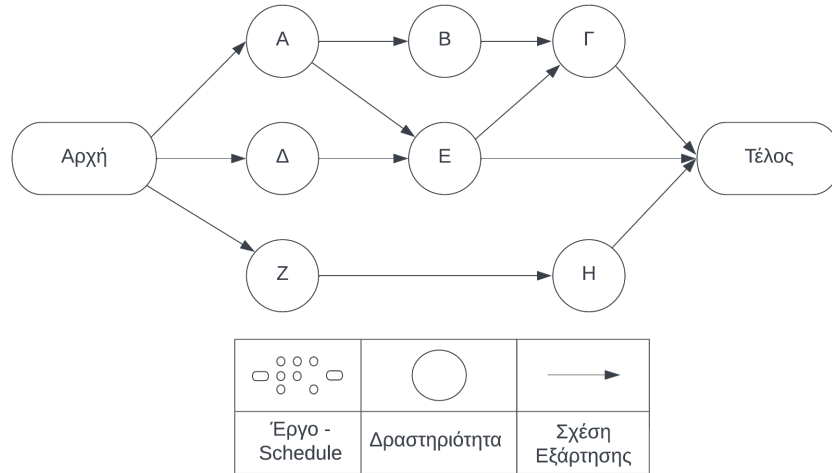
— 3 —

ΠΑΡΟΥΣΙΑΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

3.1 Καθορισμός του Προβλήματος

3.1.1 Μαθηματική Μοντελοποίηση

Ένα έργο μπορεί να αναπαρασταθεί ως ένα δίκτυο $M(N, A)$ με δραστηριότητες στους κόμβους (Activity on Node – AoN), Σχήμα 3.1, όπου N οι δραστηριότητες - κόμβοι και A οι σχέσεις εξάρτησης. Κάθε δραστηριότητα από την πλασματική Αρχή 0 έως την πλασματική Τέλος $N + 1$ έχει διάρκεια d_i , απαίτηση πόρων r_{ik} και χρηματοροή CF_i , η οποία υπολογίζεται στον χρόνο λήξης της δραστηριότητας f_i . Κάθε πόρος έχει όριο διαθεσιμότητας u_k και το έργο υφίσταται μια καθυστέρηση για την δημιουργία της διορίας Δ του έργου, η οποία ισούται με $\Delta = 1.15 \times f_{N+1}$. Η τιμή του Δ καθορίστηκε ώστε να μην αψηφά τον σκοπό της καθυστέρησης. Πιο μικρές τιμές δεν θα βελτιώναν σημαντικά την λύση, ενώ μεγαλύτερες δεν θα παρήγαγαν ρεαλιστικές λύσεις.



Σχήμα 3.1: Ενδεικτικό σχήμα δικτύου AoN, $N = 7$ δραστηριοτήτων.

Ο υπολογισμός της χρηματοροής είναι ως εξής:

$$CF_i = \text{const} - \sum_{p=0}^k r_{ip} \times c_p \quad (3.1)$$

Όπου:

- k το πλήθος των διαφορετικών πόρων,

- p ο πόρος,
- c_p η αξία μία μονάδας του πόρου p , ίση με 15.0 για όλους τους πόρους
- i η δραστηριότητα και
- $const$ μία σταθερά που ισούται με 100.0.

Οι τιμές αυτές επιλέχθηκαν πειραματικά ως σταθερές ώστε πρώτον, να υπάρχει ευκολία στην σύγκριση των λύσεων και δεύτερον να παράγονται αρνητικές και θετικές npv στις δραστηριότητες.

Στους μαθηματικούς ορισμούς που αναφέρθηκαν οι μεταβλητές που χρησιμοποιούνται δεν είναι οι αντιπροσωπευτικές ονομασίες των μεταβλητών που υπάρχουν στον πηγαίο κώδικα, διότι στην πραγματικότητα είναι ιδιαίτερα μεγάλες. Για την σαφήνεια λοιπόν του μαθηματικού μοντέλου, θα ορισθούν στη συνέχεια πίνακας αντιστοιχίας 3.1.

Πίνακας 3.1: Αντιστοιχία μεταβλητών μαθηματικού μοντέλου - δεδομένων πηγαίου κώδικα.

Μαθηματικό Μοντέλο	Πηγαίος Κώδικας
i	<i>activity</i>
d_i	<i>duration</i>
k	<i>resourceNum</i>
r_{ik}	<i>demand</i>
u_k	<i>resourceCap</i>
CF_i	<i>cf</i>
s_i	<i>start</i>
f_i	<i>start + duration</i>
j	<i>successors</i>
c_p	<i>ResourceCost</i>
$const$	<i>ConstantV</i>
Δ	<i>del</i>

Το πρόβλημα που επιλύει ο αλγόριθμος που αναπτύχθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας ανήκει στην κατηγορία των RCPSP με Προεξοφλημένες Ταμειακές Ροές (Discounted Cash flows), εν συντομία RCPSP-DC. Το πρόβλημα

μοντελοποιείται ως εξής:

$$\max \sum_{i=1}^N \frac{CF_i}{(1+i)^{f_i}} \quad (3.2)$$

Υποκείμενη στους περιορισμούς:

$$f_i \leq s_j, \quad \forall (i, j) \in A, \quad (3.3)$$

$$f_{N+1} \leq \Delta \quad (3.4)$$

$$\sum_{i \in P_t} r_{ik} \leq u_k, \quad t = 1, \dots, f_{N+1} + \delta \text{ και } k \in R \quad (3.5)$$

$$f_0 = 0, \quad d_0 = 0 \quad (3.6)$$

$$\Delta \text{ ακέραιος} \quad (3.7)$$

Στόχος της επίλυσης της αντικειμενικής συνάρτησης 3.2 είναι η εύρεση λύσης ώστε να μεγιστοποιείται η συνολική τιμή της npv .

Ο περιορισμός 3.3 εξασφαλίζει την μη παραβίαση των σχέσεων εξάρτησης καθ' όλη την διάρκεια του έργου, με j τους διάδοχους της δραστηριότητας i .

Ο περιορισμός 3.4 εξασφαλίζει πως η διάρκεια του έργου δεν θα είναι μεγαλύτερη της προθεσμίας.

Ο περιορισμός 3.5 ορίζει την διαθεσιμότητα του πόρου r , η οποία παραμένει σταθερή σε όλη την διάρκεια του έργου και δεν μπορεί να παραβιάζεται σε οποιαδήποτε στιγμή, δηλαδή το άθροισμα ζήτησης του πόρου r_{ik} από όλες τις δραστηριότητες P_t την χρονική περίοδο t , δεν μπορεί να υπερβαίνει την διαθεσιμότητα u_k .

Τέλος, ο περιορισμός 3.6 ορίζει πως το έργο ξεκινά την χρονική στιγμή μηδέν και ο περιορισμός 3.7 ότι η διορία του έργου θα είναι ακέραιος αριθμός.

3.1.2 Παράμετροι για την σχεδίαση του αλγορίθμου

Για την εκκίνηση του αλγορίθμου που υλοποιήθηκε απαιτείται ένα σύνολο παραμέτρων τα περισσότερα εκ των οποίων οι τιμές τους καθορίζονται από τον χρήστη:

- `initial`, τύπου `int` όπου καθορίζει τον αρχικό πληθυσμό και κατ' επέκταση το πλήθος των καλύτερων (`best`) και των ελίτ (`elite`) λύσεων που θα προκύψουν από τους υπολογισμούς.
- `frg`, τύπου `int` όπου προσδιορίζει την έκταση της τοπικής έρευνας για κάθε ένα άτομο του αρχικού πληθυσμού (`initial`).
- `rate`, τύπου `double` που καθορίζει την τιμή του προεξοφλητικού επιτοκίου.
- `s_rates` ένα διάνυσμα τύπου `double`, όπου ο χρήστης μπορεί να ορίσει δυναμικά το πλήθος των διαφορετικών προεξοφλητικών επιτοκίων και στην συνέχεια τις τιμές αυτών στην περίπτωση όπου θελήσει να προβεί σε ανάλυση ευαισθησίας του προβλήματος.

Επίσης, για την εκτίμηση της ορθότητας και της αποτελεσματικότητας του αλγόριθμου, αλλά και την εξαγωγή συγκρίσεων με παρόμοιες προσεγγίσεις θα χρησιμοποιηθούν διεθνώς χρησιμοποιούμενα σύνολα δεδομένων (`data sets`) που υπάρχουν και αντλούνται από την επίσημη βιβλιοθήκη [PSPLIB](#) (Kolisch & Sprecher, 1997). Τα αρχεία που χρησιμοποιούνται από την επίσημη βιβλιοθήκη αποτελούν το πρόβλημα προς επίλυση και ποικίλουν σε πολυπλοκότητα, ωστόσο ακολουθούν την ίδια διάταξη και μορφή, η οποία είναι η μορφή Patterson (OR-AS, 2015). Η χρήση της συγκεκριμένης μορφής έχει εφαρμογή για την εκπροσώπηση των δικτύων AoN, Σχήμα 3.1, σε μορφή απλού κειμένου. Παρακάτω οι ονομασίες των μεταβλητών που παρουσιάζονται συναντούνται έτσι και στον πηγαίο κώδικα. Τα αρχεία αυτά περιέχουν κατά σειρά:

1. Το πλήθος των δραστηριοτήτων (`inputs`), μαζί με τις πλασματικές δραστηριότητες Αρχή και Τέλος, και το πλήθος των διαφορετικών ανανεώσιμων πόρων (`resourceNum`),
2. Τη διαθεσιμότητα κάθε ανανεώσιμου πόρου (`resourceCap`),
3. Από την τρίτη γραμμή μέχρι την τελευταία ακολουθούν οι πληροφορίες κάθε δραστηριότητας οι οποίες είναι:
 - i. Η διάρκεια (`duration`),
 - ii. Οι απαιτήσεις πόρου για κάθε πόρο (`demand`),
 - iii. Το πλήθος των διάδοχων (`numofsucc`) και

iv. Τα ID των διάδοχων (successors)

Στο Σχήμα 3.2 παρουσιάζεται η διάταξη των αρχείων .RCP ή .txt που χρησιμοποιούνται.

Inputs resourceNum	32	4							
resourceCap	12	13	4	12					
duration 4*demand numofsucc successors	0	0	0	0	0	3	2	3	4

	0	0	0	0	0	0			

Σχήμα 3.2: Διάταξη αρχείου 30 δραστηριοτήτων, J301_1.RCP.

3.1.3 Οι κλάσεις του αλγόριθμου

Παρακάτω παρουσιάζονται οι κλάσεις που χρησιμοποιούνται στον αλγόριθμο:

1. Rrcp Class,
2. In_Pop Class,
3. Foragers Class,
4. Best Class,
5. Elite Class και
6. Opt Class.

Η κλάση Rrcp αποτελεί ένα αντικείμενο αποθήκευσης των δεδομένων που φορτώνονται από το αρχείο και παρέχουν πληροφορίες για την κάθε δραστηριότητα. Οι υπόλοιπες κλάσεις αποτελούν μια μέθοδο αποθήκευσης των λύσεων στα κομβικά σημεία κατά της εκτέλεση του αλγόριθμου.

Συγκεκριμένα σε κάθε κλάση αποθηκεύονται:

- Στην In_Pop ο αρχικός πληθυσμός,
- Στην Foragers η Τοπική Έρευνα,
- Στην Best οι καλύτερες λύσεις της Τοπικής Έρευνας,

- Στην Elite οι λύσεις μετά την καθυστέρηση και
- Στην Opt η καλύτερη λύση που παρήγαγε ο αλγόριθμος.

Παρακάτω βρίσκονται οι πίνακες 3.2 - 3.6 με τα δεδομένα και του τύπους των δεδομένων κάθε κλάσης:

Πίνακας 3.2: Δεδομένα - Μέλη Recp Class

DESIGNATION	VARIABLE TYPE	DESCRIPTION
activity	int	Η δραστηριότητα
duration	int	Διάρκεια
demand	vector<int>	Απαίτηση πόρων
numofsucc	int	Πλήθος διάδοχων
successors	vector<int>	Διάδοχοι (Επόμενες δραστηριότητες)
predecessors	vector<int>	Προαπαιτούμενες (Προηγούμενες δραστηριότητες)
cf	double	Χρηματοροή
start	int	Χρονική στιγμή εκκίνησης
activityCheck	bool	Μεταβλητή ελέγχου
NPV	double	Τιμή της NPV της δραστηριότητας

Πίνακας 3.3: Δεδομένα - Μέλη In_Pop Class

DESIGNATION	VARIABLE TYPE	DESCRIPTION
solution	vector<int>	Η σειρά των δραστηριοτήτων
NPV	double	Η συνολική τιμή της NPV
duration	int	Η διάρκεια της λύσης
prio	vector<double>	Οι τιμές του διανύσματος προτεραιότητας για κάθε δραστηριότητα

Πίνακας 3.4: Δεδομένα - Μέλη Foragers Class

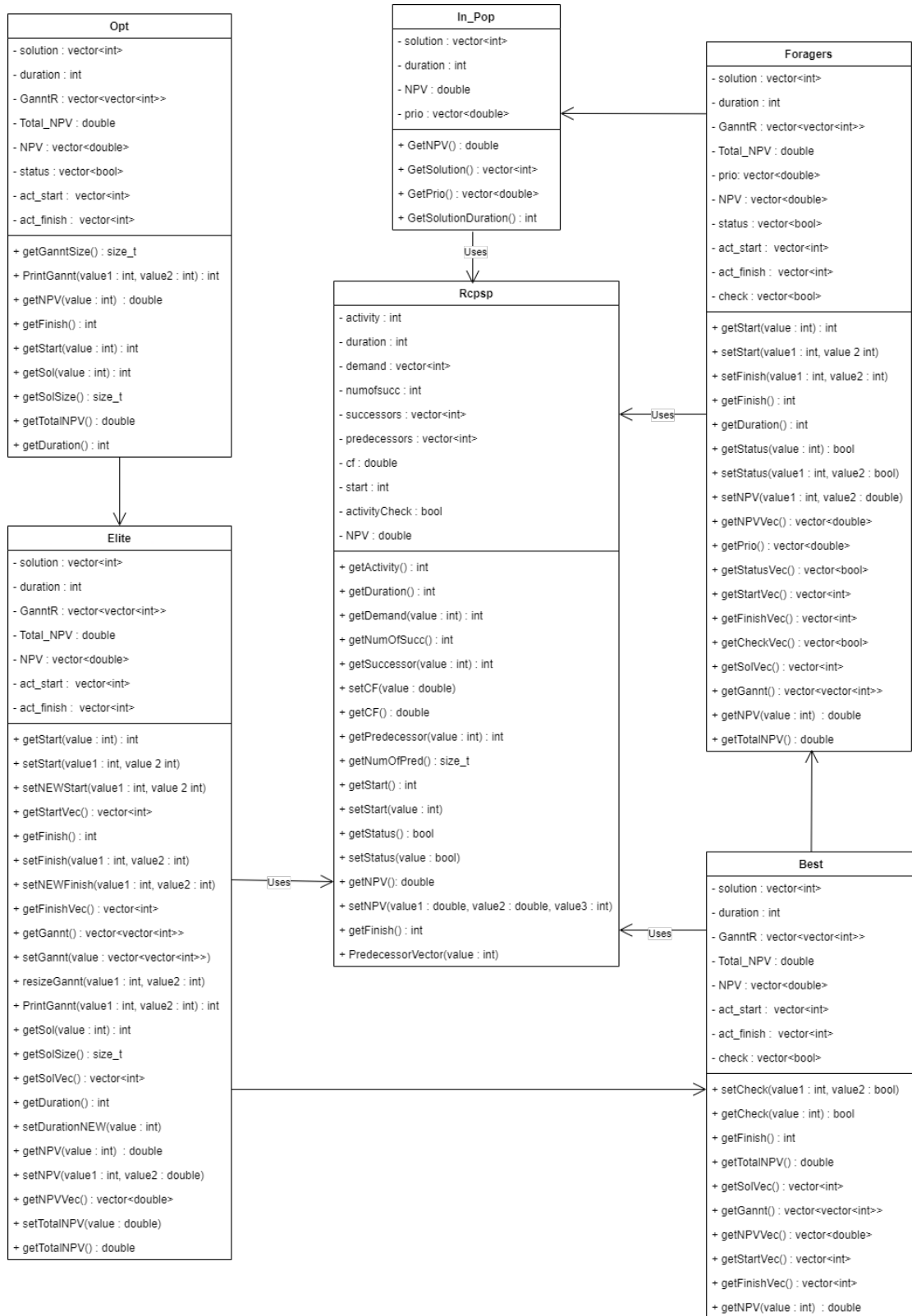
DESIGNATION	VARIABLE TYPE	DESCRIPTION
solution	vector<int>	Η σειρά των δραστηριοτήτων
duration	int	Η διάρκεια της λύσης
Gannt	vector<vector<int> >	Πίνακας χρήσης πόρων
Total_NPV	double	Η συνολική τιμή της NPV
NPV	vector<double>	NPV κάθε δραστηριότητας της λύσης
status	vector<bool>	Μεταβλητή ελέγχου για κάθε δραστηριότητας της λύσης
act_start	vector<int>	Χρονική στιγμή εκκίνησης κάθε δραστηριότητας της λύσης
act_finish	vector<int>	Χρονική στιγμή λήξης κάθε δραστηριότητας της λύσης
prio	vector<double>	Οι τιμές του διανύσματος προτεραιότητας για κάθε δραστηριότητα
check	vector<bool>	Μεταβλητή ελέγχου για καθυστέρηση

Πίνακας 3.5: Δεδομένα - Μέλη Best Class

DESIGNATION	VARIABLE TYPE	DESCRIPTION
solution	vector<int>	Η σειρά των δραστηριοτήτων
duration	int	Η διάρκεια της λύσης
Gannt	vector<vector<int> >	Πίνακας χρήσης πόρων
Total_NPV	double	Η συνολική τιμή της NPV
NPV	vector<double>	NPV κάθε δραστηριότητας της λύσης
act_start	vector<int>	Χρονική στιγμή εκκίνησης κάθε δραστηριότητας της λύσης
act_finish	vector<int>	Χρονική στιγμή λήξης κάθε δραστηριότητας της λύσης
check	vector<bool>	Μεταβλητή ελέγχου για καθυστέρηση

Πίνακας 3.6: Δεδομένα - Μέλη Elite και Opt Class

DESIGNATION	VARIABLE TYPE	DESCRIPTION
solution	vector<int>	Η σειρά των δραστηριοτήτων
duration	int	Η διάρκεια της λύσης
Gannt	vector<vector<int> >	Πίνακας χρήσης πόρων
Total_NPV	double	Η συνολική τιμή της NPV
NPV	vector<double>	NPV κάθε δραστηριότητας της λύσης
act_start	vector<int>	Χρονική στιγμή εκκίνησης κάθε δραστηριότητας της λύσης
act_finish	vector<int>	Χρονική στιγμή λήξης κάθε δραστηριότητας της λύσης



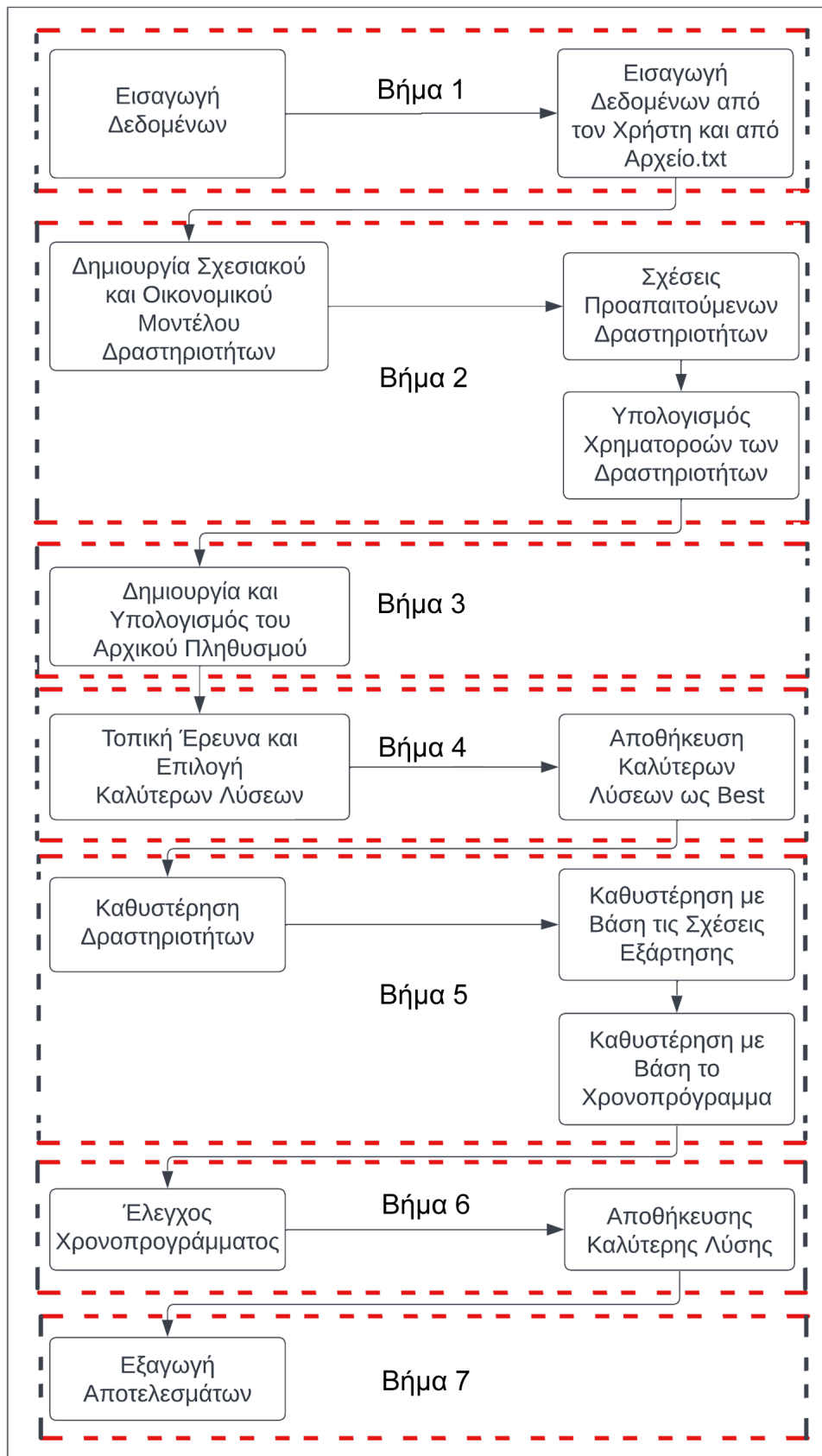
Σχήμα 3.3: Διάγραμμα κλάσεων

3.2 Παρουσίαση του Αλγορίθμου

Ο πηγαίος κώδικας του αλγόριθμου που υλοποιήθηκε αποτελείται από τα παρακάτω βήματα:

1. Εισαγωγή δεδομένων,
2. Δημιουργία Σχεσιακού και Οικονομικού Μοντέλου,
3. Δημιουργία και Υπολογισμός του Αρχικού Πληθυσμού,
4. Τοπική Έρευνα και Επιλογή Καλύτερων Λύσεων,
5. Καθυστέρηση Δραστηριοτήτων,
6. Έλεγχος Χρονοπρογράμματος,
7. Εξαγωγή Αποτελεσμάτων.

Στο Σχήμα 3.4 παρουσιάζεται το διάγραμμα ροής του αλγόριθμου όπου φαίνεται αναλυτικά η σειρά με την οποία εκτελούνται τα βήματα και οι επιμέρους διαδικασίες των βημάτων. Προτού ξεκινήσει η ανάλυση του πηγαίου κώδικα, σε αυτό το σημείο αξίζει να αναφερθεί πως στις παρακάτω υποενότητες θα γίνει η ανάλυση της λειτουργίας του αλγόριθμου και όχι της διεπαφής χρήστη. Ο τρόπος που μπορεί ο χρήστης να αλληλεπιδράσει με τις αρχικές τιμές, να επιλέξει το αρχείο του προβλήματος, να επιλέξει αν το πρόγραμμα θα εκτελέσει ανάλυση ευαισθησίας, όπως αναφέρθηκε στην υποενότητα 3.1.2 και τα γραφήματα που μπορεί να εξάγει το πρόγραμμα θα αναφερθούν στο επόμενο κεφάλαιο όπου θα αναλυθεί η διεπαφή χρήστη για λόγους σαφήνειας.



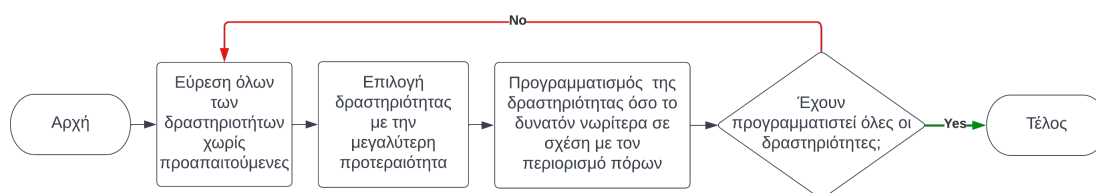
Σχήμα 3.4: Διάγραμμα Ροής του Αλγορίθμου.

3.3 Προεπισκόπηση Σημαντικών Βημάτων

Πριν την ανάλυση βήμα προς βήμα, θα αναλυθούν περιεκτικά τα σημαντικότερα και μεγαλύτερα βήματα του αλγορίθμου, ώστε να είναι πιο εύκολη και κατανοητή η λειτουργία τους.

3.3.1 Αρχικός Πληθυσμός

Για την εύρεση των λύσεων και την ποικιλομορφία τους, καθ' όλη την διάρκεια του υπολογισμού του αλγορίθμου χρησιμοποιείται η Εμπρόσθια Σειριακή Μέθοδος Παραγωγής Χρονοπρογραμμάτων (Forward SSGS - Forward Serial Schedule Generation Scheme) (Kolisch, 1996) με τη χρήση διανυσμάτων προτεραιότητας. Με τον όρο Εμπρόσθια (Forward) προσδιορίζεται πως το χρονοπρόγραμμα σχεδιάζεται από την αρχική δραστηριότητα προς την τελική. Τα βήματα της μεθοδολογίας αυτής παρουσιάζονται στο διάγραμμα ροής στο Σχήμα 3.5.



Σχήμα 3.5: Διάγραμμα Ροής Εμπρόσθιας Σειριακής Μεθόδου Παραγωγής Χρονοπρογραμμάτων

Τα διανύσματα προτεραιότητας παράγονται εν μέσω μίας γεννήτριας τυχαίων αριθμών με την κλήση των κλάσεων `random_device` και `uniform_real_distribution`. Όπως φαίνεται και από την ονομασία της κλάσης, οι αριθμοί που παράγονται ακολουθούν μία ομοιόμορφη κατανομή ώστε να υπάρχει ίση πιθανότητα εμφάνισης ολόκληρου του εύρους τιμών και να αποφευχθεί η αλγοριθμική μεροληψία. Με τη χρήση τους αναθέτονται τιμές από 0.001 έως και 0.999 στις δραστηριότητες του προβλήματος, με τις πλασματικές δραστηριότητες «Αρχή» ή 0 και «Τέλος» ή $n + 1$ να λαμβάνουν τιμές 1.00 και 0.00 αντίστοιχα ώστε να εισαχθούν στις αντίστοιχες θέσεις. Ο Πίνακας 3.7 αποτελεί ένα παράδειγμα που δημιουργήθηκε από τον αλγόριθμο.

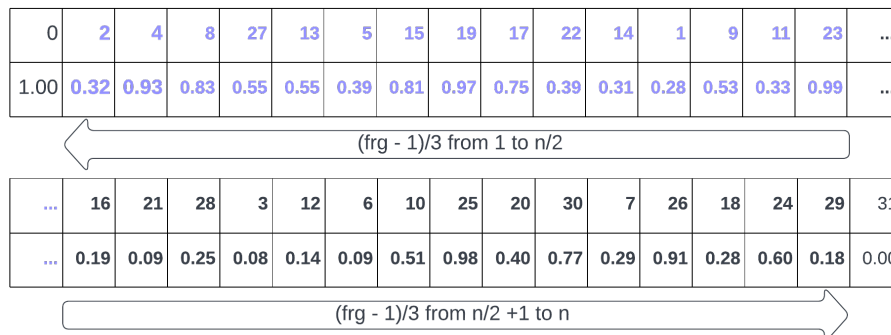
Πίνακας 3.7: Παράδειγμα λίστας και διανυσμάτων προτεραιότητας για πρόβλημα 30 δραστηριοτήτων του αρχείου J301_1.RCP.

Activity:	0	3	2	1	14	9	21	6	...	30	31
Priority:	1.00	0.81	0.63	0.56	0.36	0.19	0.76	0.18	...	0.69	0.00

3.3.2 Τοπική Έρευνα

Η διαδικασία της Τοπικής Έρευνας αποτελεί το πιο χρονοβόρο τμήμα του αλγορίθμου καθώς εφαρμόζεται σε ολόκληρο τον πληθυσμό (*initial*) και εκτελεί έναν μεγάλο αριθμό επαναλήψεων (*frg*). Η διαδικασία ουσιαστικά πραγματοποιεί αλλαγές στα διανύσματα προτεραιότητας των αρχικών λύσεων, χωρίς όμως να αλλάζει τις τιμές των πλασματικών δραστηριοτήτων 0 και $n + 1$. Αρχικά αποθηκεύεται η αρχική λύση και στην συνέχεια ακολουθούν οι αλλαγές στα διανύσματα προτεραιότητας για την παραγωγή νέων λύσεων με τρεις διαφορετικούς τρόπους:

1. Αλλαγή των διανυσμάτων προτεραιότητας από την πρώτη δραστηριότητα έως την μεσαία δραστηριότητα, $n/2$, της λύσης για $(frg - 1)/3$ λύσεις.
2. Αλλαγή των διανυσμάτων προτεραιότητας από την επόμενη της μεσαίας δραστηριότητας, $n/2 + 1$, έως την τελευταία δραστηριότητα της λύσης για $(frg - 1)/3$ λύσεις.
3. Τυχαίες αλλαγές σε ολόκληρη την λύση για $(frg - 1)/3$ λύσεις.

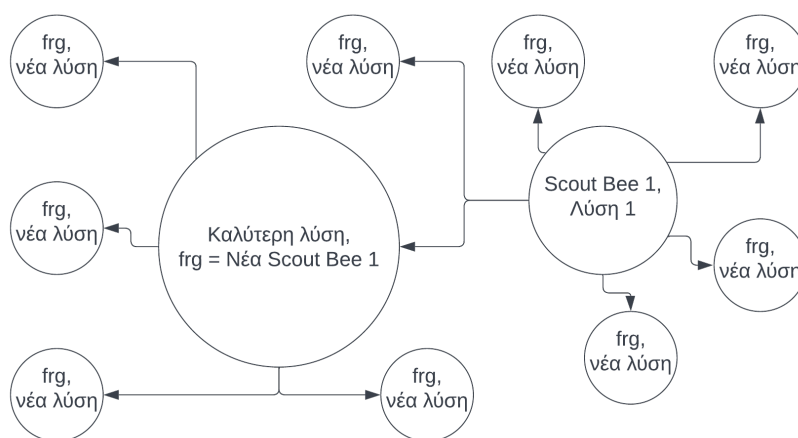


Σχήμα 3.6: Ενδεικτικό σχήμα πρώτου και δεύτερου τρόπου αλλαγής προτεραιότητας

Για παράδειγμα, σε ένα πρόβλημα $n = 30$ δραστηριοτήτων, με αρχικό πληθυσμό $initial = 1$ και αριθμό επαναλήψεων $frg = 100$:

- Θα διατηρηθεί η λύση του αρχικού πληθυσμού ως νέα λύση 0,
- Θα παραχθούν 33 λύσεις με κάθε διαφορετικό τρόπο,
- Αν βρεθεί καλύτερη λύση από την αρχική (initial) τότε η νέα λύση γίνεται η αρχική και ξεκινά εκ νέου η αλλαγή των διανυσμάτων προτεραιότητας,
- Η διαδικασία αυτή επαναλαμβάνεται για κάθε μέλος του πληθυσμού έως ότου με το πέρας των αλλαγών να μην βρεθεί καλύτερη λύση.

Ο διαχωρισμός της Τοπικής Έρευνας με αυτόν τον τρόπο δίνει στον αλγόριθμο το πλεονέκτημα να εκτελεί εστιασμένη αναζήτηση καλύτερης λύσης με τους τρόπους που αναδεικνύονται στο Σχήμα 3.6, κρατώντας σε κάθε περίπτωση την μισή λύση ατόφια και πραγματοποιώντας αλλαγές στην υπόλοιπη μισή. Το μέγεθος της εντατικοποίησης επιλέχθηκε πειραματικά καθώς βρέθηκε πως το συγκεκριμένο πλήθος αλλαγών είναι ικανό να παράξει καλύτερη λύση. Με τον τρίτο τρόπο αλλαγής ο αλγόριθμος εκτελεί τη διαφοροποίηση των λύσεων με σκοπό να τα απεγκλωβιστεί από τοπικές βέλτιστες λύσεις ώστε να παράξει την καλύτερη δυνατή λύση.



Σχήμα 3.7: Εύρεση καλύτερης λύσης και επανάληψη Τοπικής Έρευνας

Ένα ακόμα πλεονέκτημα του διαχωρισμού αποτελεί το γεγονός πως η συγκεκριμένη συνάρτηση έχει δομηθεί κατάλληλα ώστε να μπορεί να παράγει τις λύσεις παράλληλα για κάθε τρόπο. Αυτό επιτυγχάνεται με την ανάθεση ενός thread για κάθε διαφορετικό τρόπο, συνολικά τρία threads, μειώνοντας τον χρόνο υπολογισμού σε σημαντικό βαθμό.

Στο τέλος της διαδικασίας, αποθηκεύεται η καλύτερη λύση από κάθε μέλος του πληθυσμού ως ο νέος πληθυσμός (Best λύσεις), ενώ ο παλιός διαγράφεται.

3.3.3 Καθυστέρηση

Η διαδικασία της καθυστέρησης είναι σημαντική καθώς είναι διακριτό από την την εξίσωση 2.1 στη σελίδα 9, πως με την αύξηση του χρόνου λήξης μιας δραστηριότητας με αρνητική χρηματοροή, μπορεί να αυξηθεί η ολική nrv του έργου. Η διαδικασία της καθυστέρησης υφίσταται για όλα τα μέλη του νέου πληθυσμού και πραγματοποιείται σε δύο στάδια:

1. Με βάση τις σχέσεις εξάρτησης και
2. Με βάση τους χρόνους λήξης των δραστηριοτήτων

Τα κριτήρια που θα προσδιορίσουν αν θα γίνει καθυστέρηση μιας ή ενός σετ δραστηριοτήτων θα αναλυθεί παρακάτω. Η διαδικασία δύο σταδίων υλοποιήθηκε από τους (Leyman & Vanhoucke, 2015) και εφαρμόστηκε στην παρούσα εργασία.

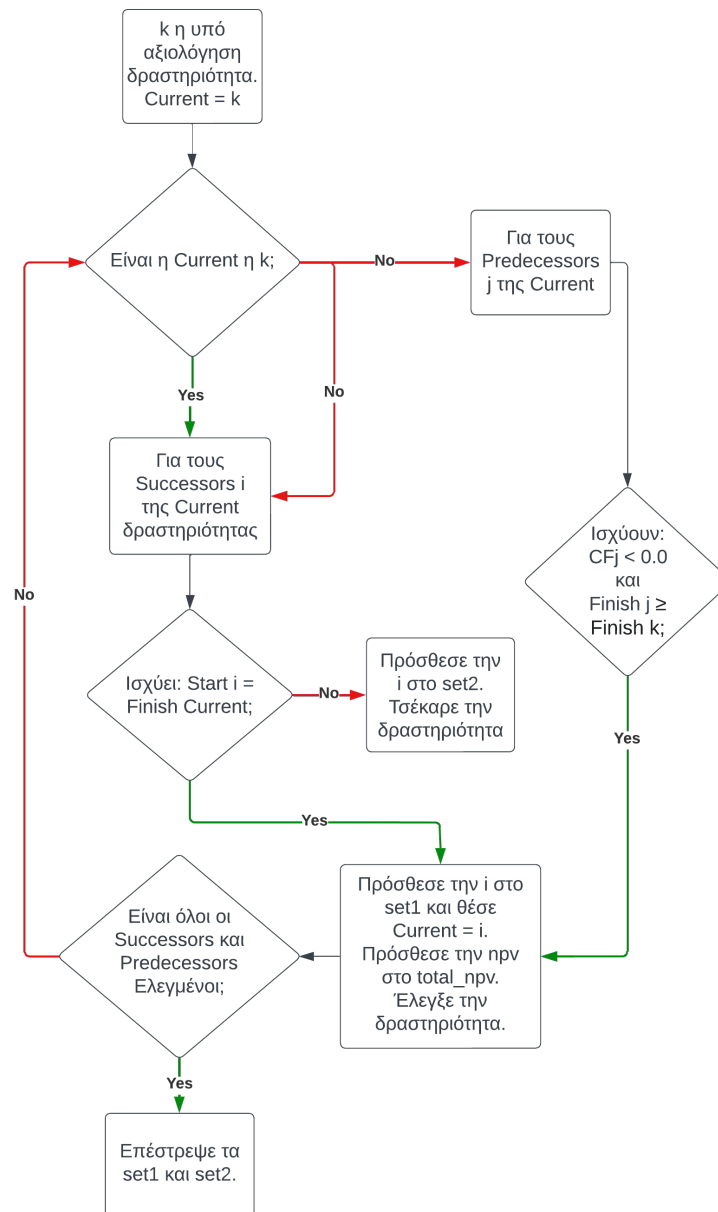
Αρχικά, στο πρώτο στάδιο εφαρμόζεται η καθυστέρηση Δ στο χρονοπρόγραμμα και ξεκινώντας από την τελευταία δραστηριότητας της λύσης (λίστα προτεραιότητας), αξιολογείται η χρηματοροή κάθε δραστηριότητας. Μέσω μιας επαναληπτικής μεθόδου επιστρέφεται το σετ δραστηριοτήτων προς καθυστέρηση ($set1$), το άθροισμα των nrv των δραστηριοτήτων του $set1$ (nrv_total) και το σετ δραστηριοτήτων για τον υπολογισμό της καθυστέρησης ($set2$). Ο αλγόριθμος υπολογισμού των set φαίνεται στο Σχήμα 3.8.

Έπειτα γίνεται η αξιολόγηση του συνόλου nrv_total , όπου αν είναι θετική, η διαδικασία της καθυστέρησης παραλείπεται και η διαδικασία προχωράει στην προηγούμενη δραστηριότητα της λίστας προτεραιότητας. Ωστόσο, αν είναι αρνητική, υπολογίζεται η μέγιστη επιτρεπτή καθυστέρηση που μπορεί να εφαρμοστεί στο $set1$. Η τιμή αυτή υπολογίζεται ως εξής:

$$\delta = \min_{i \in set1, k \in set2, k \in S_i} s_k - f_i \quad (3.8)$$

Όπου:

- S_i οι διάδοχοι (*Successors* της δραστηριότητας i),
- s_k ο χρόνος έναρξης του διάδοχου του i και
- f_i ο χρόνος λήξης του i .



Σχήμα 3.8: Αναλυτικό διάγραμμα ροής υπολογισμού $set1$ και $set2$ με βάση τις σχέσεις εξάρτησης. Πηγή: (Leyman & Vanhoucke, 2015).

Όπου:

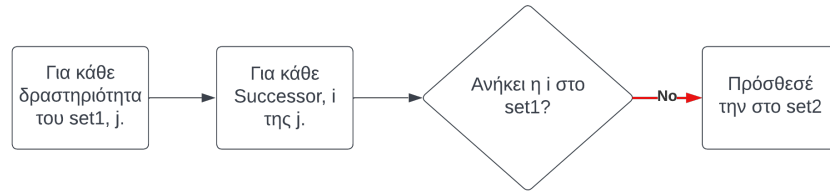
- k η αρχική υπό αξιολόγηση δραστηριότητα,
- $Finish_k$ χρόνος λήξης της δραστηριότητας k ,
- $Current$, η δραστηριότητα που αξιολογείται,
- $Finish_{Current}$ ο χρόνος λήξης της δραστηριότητας που αξιολογείται,
- i οι διάδοχοι (Successors) της $Current$,
- $Start_i$, ο χρόνος έναρξης της δραστηριότητας i ,
- j οι προαπαιτούμενες (Predecessors) της $Current$,
- $Start_j$ ο χρόνος έναρξης της δραστηριότητας j ,
- $Finish_j$ ο χρόνος λήξης της δραστηριότητας j ,
- CF_j η χρηματοροή της δραστηριότητας j και
- $total_npv$ το άθροισμα της npv των δραστηριοτήτων του $set1$

Αφότου έχει υπολογιστεί η τιμή της εξίσωσης 3.8, η διαδικασία συνεχίζει με την εφαρμογή της καθυστέρησης στο $set1$ και τον έλεγχο παραβίασης διαθεσιμότητας πόρων. Στην περίπτωση που η καθυστέρηση είναι αδύνατη, η διαδικασία συνεχίζει την αξιολόγηση στις προηγούμενες δραστηριότητες της λίστας προτεραιότητας. Αν στο τέλος της διαδικασίας του ελέγχου της λίστας προτεραιότητας έχει γίνει τουλάχιστον μία καθυστέρηση στο χρονοπρόγραμμα, η διαδικασία αξιολόγησης της λίστας προτεραιότητας ξεκινά εκ νέου, έως ότου να είναι αδύνατη η περαιτέρω καθυστέρηση των δραστηριοτήτων.

Το δεύτερο στάδιο της καθυστέρησης λειτουργεί με την ίδια αρχή του πρώτου σταδίου με την μόνη διαφορά στα κριτήρια δημιουργίας του $set1$.

Όπου:

- k η αρχική υπό αξιολόγηση δραστηριότητα,
- $Finish_k$ χρόνος λήξης της δραστηριότητας k ,



Σχήμα 3.10: Αναλυτικό διάγραμμα ροής υπολογισμού *set2* βάση τους χρόνους λήξης. Πηγή: (Leyman & Vanhoucke, 2015).

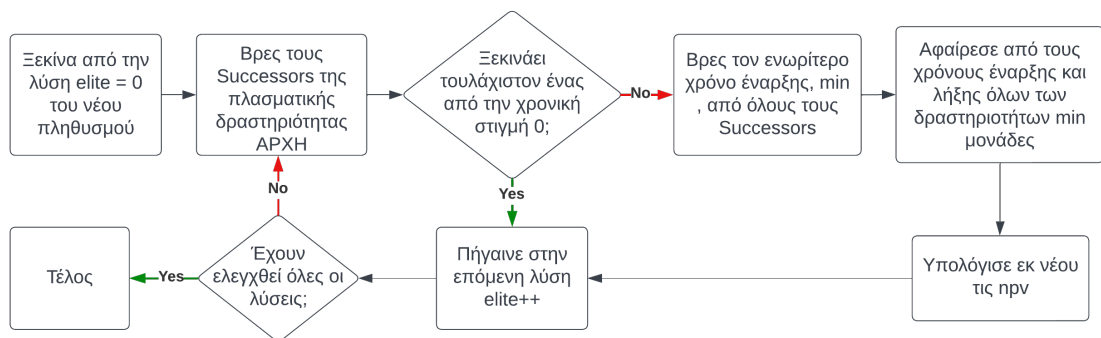
3.3.4 Έλεγχος Χρονοπρογράμματος

Πρέπει να σημειωθεί πως σε προβλήματα όπου ένα μεγάλο πλήθος ή όλες οι δραστηριότητες παρουσιάζουν αρνητική χρηματοροή, όπως είναι αναμενόμενο, ο αριθμός δραστηριοτήτων που καθυστερούν είναι πιθανόν να είναι υπερβολικός. Ως αποτέλεσμα, το τελικό χρονοπρόγραμμα ενδέχεται να μην ξεκινά από την χρονική στιγμή 0, αλλά αργότερα. Για τον λόγο αυτόν εφαρμόζεται μια διόρθωση σε όλες τις δραστηριότητες ίση με:

$$\min_{i \in S_0} f_i - d_i \quad (3.9)$$

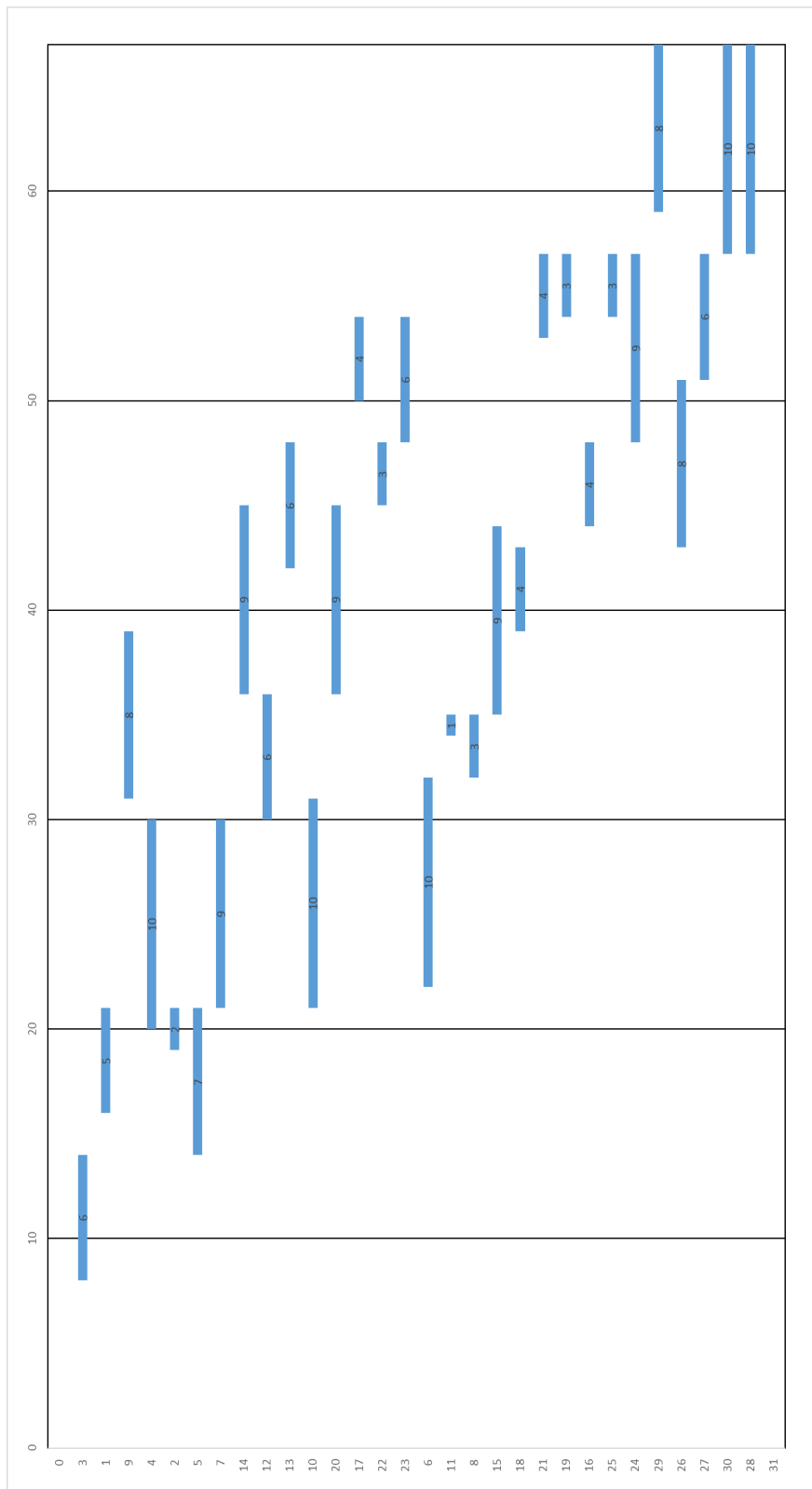
Όπου:

- S_0 το σετ των *successors* της πλασματικής δραστηριότητας 0,
- i ο *successor*,
- f_i ο χρόνος λήξης του *successor* και
- d_i η διάρκεια του *successor*.

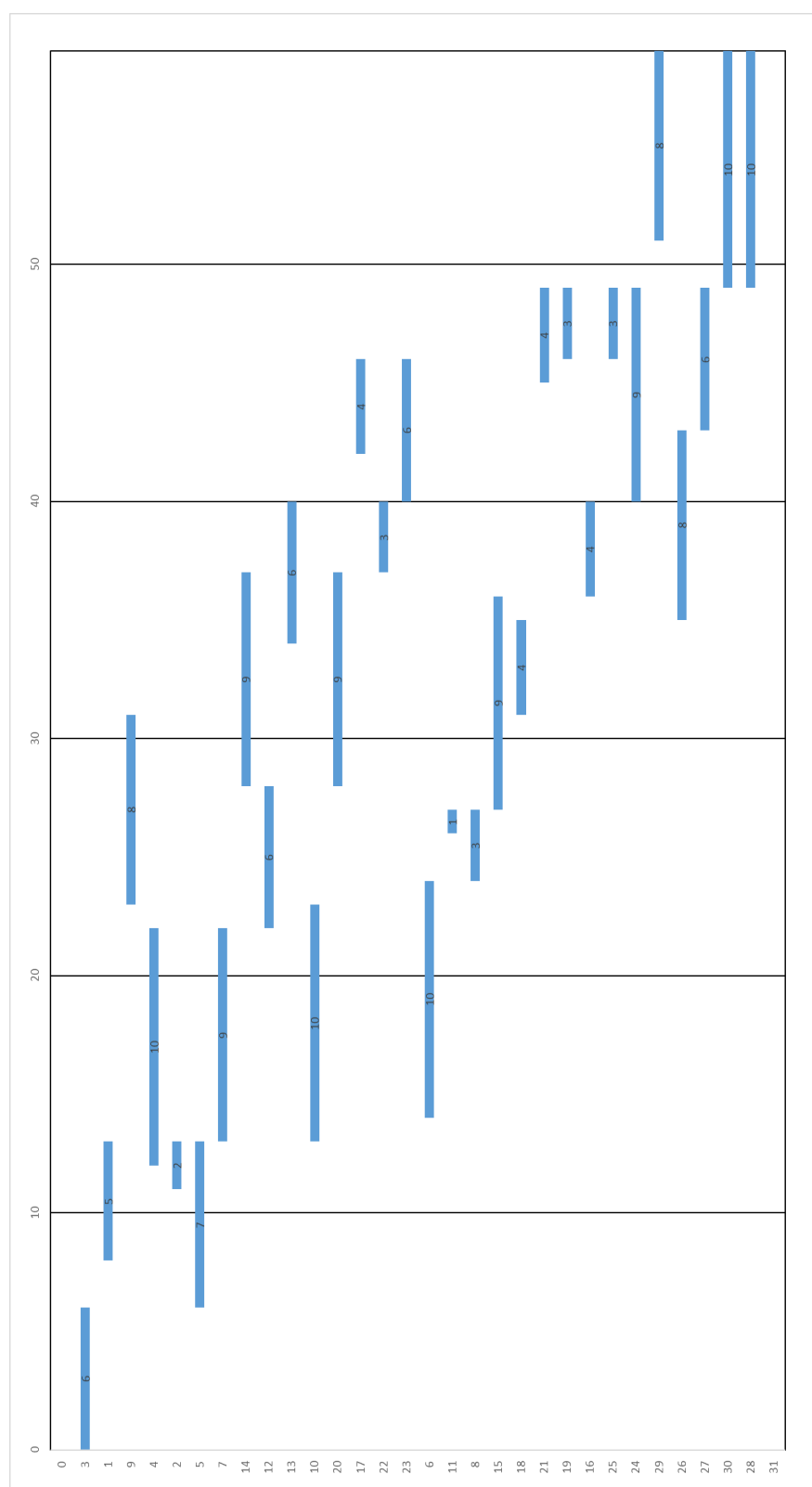


Σχήμα 3.11: Διάγραμμα ροής Ελέγχου Χρονοπρογράμματος

Έπειτα, υπολογίζεται εκ νέου οι *nrv* των δραστηριοτήτων και κατά συνέπεια η συνολική. Στο Σχήμα 3.12 φαίνεται το χρονοπρόγραμμα που έχει υποστεί καθυστέρηση αλλά είναι εσφαλμένο λόγω της καθυστέρησης σε όλες τις δραστηριότητες ενώ στο Σχήμα 3.13 είναι το ίδιο χρονοπρόγραμμα διορθωμένο.



Σχήμα 3.12: Παράδειγμα χρονοπρογράμματος 30 δραστηριοτήτων, αρχείο J3048_9.RCP, πριν την κλήση της Check_Schedule.



Σχήμα 3.13: Παράδειγμα χρονοπρογράμματος 30 δραστηριοτήτων, αρχείο J3048_9.RCP, μετά την κλήση της Check_Schedule.

3.4 Ανάλυση Πηγαίου Κώδικα

Παρακάτω ξεκινά η ανάλυση βήμα προς βήμα του πηγαίου κώδικα για όλα τα βήματα του αλγορίθμου. Η ανάλυση που ακολουθεί αφορά τον τρόπο λειτουργίας και εκτέλεσης του αλγορίθμου με την προσέγγιση των μελισσών. Στην επόμενη ενότητα, 3.5, θα αναλυθεί ο μεταερευτικός αλγόριθμος Simulated Annealing που υλοποιήθηκε για την σύγκριση των λύσεων της Τοπικής Έρευνας.

3.4.1 Βήμα 1 - Εισαγωγή Δεδομένων

Το πρώτο στάδιο αποτελεί την εισαγωγή των παραμέτρων που αφορούν το μέγεθος του αρχικού πληθυσμού, το ύψος του προεξοφλητικού επιτοκίου και το αρχείο του προβλήματος που θα χρησιμοποιηθεί. Εφόσον ο χρήστης επιλέξει τα προαναφερθέντα στοιχεία, καλείται η συνάρτηση **data** όπου εκτελεί τα εξής βήματα:

- Κάλυψε την συνάρτηση **data**.
- Δημιούργησε dummy μεταβλητές για την προσωρινή αποθήκευση των στοιχείων κατά την ανάγνωση του αρχείου.
- Διάβασε το αρχείο.
- Όρισε το μέγεθος του αντικειμένου της κλάσης *Repsp* ίσο με *inputs*.
- Μετέφερε τα στοιχεία από τις dummy μεταβλητές στο αντικείμενο της κλάσης *Repsp*.
- Όρισε μια πλασματική μεταβλητή *MaxTime* τύπου *int* για τον ορισμό του μεγέθους του πίνακα πόρων.

3.4.2 Βήμα 2 - Σχεσιακό και Οικονομικό Μοντέλο

Από το αρχείο, όπως αναφέρθηκε και στο Σχήμα 3.2, μας δίνεται έτοιμο το μοντέλο των διαδόχων για κάθε δραστηριότητα. Είναι αναγκαίος όμως ο υπολογισμός του μοντέλου εξάρτησης των προαπαιτούμενων δραστηριοτήτων πρώτον για την διευκόλυνση στους υπολογισμούς των λύσεων και δεύτερον απαιτείται στην διαδικασία του υπολογισμού της καθυστέρησης. Ακόμα, εφόσον η αντικειμενική συνάρτηση που βελτιστοποιεί ο

αλγόριθμος είναι η 3.2, πρέπει να υπολογιστούν οι χρηματοροές. Τα βήματα είναι τα εξής:

- Κάλεσε την συνάρτηση **CreatePredecessors**.
 - Για κάθε δραστηριότητα πάρε τους διάδοχους.
 - Για κάθε διάδοχο της δραστηριότητας κάνε `push_back` το `predecessors vector` της δραστηριότητας.
- Κάλεσε την συνάρτηση **createCF**.
- Για κάθε δραστηριότητα υπολόγισε την *cf* σύμφωνα με την εξίσωση 3.1.

3.4.3 Βήμα 3 - Αρχικός Πληθυσμός

Για την δημιουργία και την αποθήκευση των λύσεων του αρχικού πληθυσμού, γίνεται η χρήση της κλάσης `In_Pop`. Συγκεκριμένα τα βήματα είναι:

- Όρισε το μέγεθος του αντικειμένου `scout_bees` της κλάσης `In_Pop` ίσο με *inputs*.
- Κάλεσε τον default constructor της κλάσης `In_Pop` για την αρχικοποίηση των μελών του αντικειμένου `scout_bees`, Πίνακας 3.3.

Ο αλγόριθμος που καλείται είναι η συνάρτηση **Initial_Population** η οποία έχει ως παράμετρος:

- Το μέγεθος του πληθυσμού, *initial*.
- Ένα vector *random*, ως χώρο αποθήκευσης των διανυσμάτων προτεραιότητας.
- Ένα vector *solution*, ως χώρο αποθήκευσης της λίστας προτεραιότητας.

Ακολουθεί η ανάθεση τιμών στο vector `random` με την χρήση της γεννήτριας τυχαίων αριθμών και η κλήση της συνάρτησης για την δημιουργία των λύσεων **SSGS_Forward**. Ο αλγόριθμος της συνάρτησης αυτής είναι δομημένος έτσι ώστε να καλείται για την δημιουργία του αρχικού πληθυσμού καθώς όμως και κατά την διάρκεια της Τοπικής Έρευνας. Επειδή η διαδικασία δημιουργίας λύσεων σε αυτές τις περιπτώσεις είναι ίδιες, η **SSGS_Forward** όπως επίσης και οι συναρτήσεις που καλούνται διαδοχικά κατά την διάρκεια υπολογισμού των λύσεων έχουν ως παράμετρο, μεταξύ άλλων, έναν δείκτη

$trigger$ ο οποίος καθορίζει αν ο αλγόριθμος βρίσκεται στην διαδικασία δημιουργίας λύσεων για τον αρχικό πληθυσμό, $trigger = 0$, ή αν βρίσκεται στην διαδικασία της Τοπικής Έρευνας, $trigger = 1$. Οι παράμετροι της **SSGS_Foward** είναι:

- Το vector *random*.
- Ο δείκτης s_index , ο οποίος καθορίζει για ποιο μέλος του αρχικού πληθυσμού αντιστοιχεί η λύση που υπολογίζεται.
- Ο δείκτης frg , ο οποίος καθορίζει σε ποιο σημείο της Τοπικής Έρευνας βρίσκεται ο αλγόριθμος.
- Ο δείκτης $trigger$, που ορίζει αν θα γίνει δημιουργία αρχικού πληθυσμού ή Τοπική Έρευνα.

Οι συναρτήσεις που χρησιμοποιούνται μέσα την για τον υπολογισμό των λύσεων και στις δύο περιπτώσεις, καλούνται με την παρακάτω σειρά και είναι οι εξής:

- **SSGS_Foward**, κύρια συνάρτηση, δημιουργεί και μεταβιβάζει παραμέτρους.
 - **Forward_Activity_List**, παράγει την λίστα προτεραιότητας με βάση τις σχέσεις εξάρτησης των δραστηριοτήτων και τα διανύσματα προτεραιότητας.
 - **Forward_Activity_Check**, η συνάρτηση αυτή αποτελεί τον έλεγχο των υποψήφιων δραστηριοτήτων για να μουν στην λίστα προτεραιότητας.
 - **Forward_Resource_Check**, αποτελεί τον έλεγχο διαθεσιμότητας πόρων.
 - **Early_Start**, υπολογίζει την νωρίτερη έναρξη μιας δραστηριότητας με κριτήριο τους χρόνους λήξης των προαπαιτούμενών της.

Παρακάτω, προτού παρουσιαστούν τα βήματα του αλγορίθμου, θα αναλυθούν οι παράμετροι των προαναφερθέντων συναρτήσεων διότι καλούνται σε δύο διαφορετικές φάσεις της επίλυσης του προβλήματος και είναι σημαντικό να είναι ξεκάθαρη η λειτουργία του την εκάστοτε χρονική στιγμή.

Η **Forward_Activity_List**, επιστρέφει στην κύρια συνάρτηση την λίστα προτεραιότητας και έχει ως παραμέτρους:

- Το vector *random* με τις τιμές των διανυσμάτων προτεραιότητας.
- Τον δείκτη *trigger*.
- Τον δείκτη *s_index*.
- Τον δείκτη *frg*.

Η **Forward_Activity_Check**, επιστρέφει ένα vector με όλες τις υποψήφιες δραστηριότητες για να μπουν στην λίστα προτεραιότητας και έχει ως παραμέτρους:

- Το vector *next_act* με τις υποψήφιες δραστηριότητες.
- Έναν pointer *act* της δραστηριότητας που θα μπει στην λίστα προτεραιότητας.
- Το vector *random*.
- Τον δείκτη *trigger*.
- Τον δείκτη *s_index*.
- Τον δείκτη *frg*.

Η **Forward_Resource_Check** επιστρέφει τον πίνακα χρήσης πόρων και έχει ως παραμέτρους:

- Τον πίνακα χρήσης πόρων *GanntR*.
- Την λίστα προτεραιότητας vector *Solution* που παρήχθη από την **Forward_Activity_List**.
- Έναν pointer *npv* για τον υπολογισμό της συνολικής *npv* όταν εκτελείται ο αλγόριθμος της Τοπικής Έρευνας.
- Έναν pointer *finish* που επιστρέφει την χρονική διάρκεια της λύσης.
- Τον δείκτη *trigger*.
- Τον δείκτη *s_index*.
- Τον δείκτη *frg*.

Τελευταία, η **Early_Start** έχει ως παραμέτρους:

- Την δραστηριότητα *act* που εξετάζεται.

- Τον δείκτη *trigger*.
- Τον δείκτη *s_index*.
- Τον δείκτη *frg*.

Εφόσον έχει ορισθεί η δομή των συναρτήσεων, μπορεί να πλέον να αναλυθεί βήμα προς βήμα η διαδικασία δημιουργίας των λύσεων. Από εδώ και έπειτα το διάνυσμα προτεραιότητας θα αναφέρεται ως *prio*, όπως έχει ορισθεί στον Πίνακα 3.3. Υπενθυμίζεται, πως στην παρούσα φάση ο αλγόριθμος εκτελείται με την τιμή του *trigger* = 0 εφόσον βρίσκεται στο στάδιο παραγωγής λύσεων για τον αρχικό πληθυσμό. Τα βήματα είναι τα εξής:

- Δημιούργησε ένα vector *Solution* για την αποθήκευση της λίστας προτεραιότητας.
- Δημιούργησε έναν πίνακα από vector *GanntR* με μέγεθος *MaxTime* και αρχικοποίησέ τον.
- Αν *trigger* = 0 θέσε το *status* όλων των δραστηριοτήτων του αντικειμένου *scout_bees*, *status* = *false* εκτός της 0. Θέσε την 0, *status* = *true*.
- Κάλυψε την **Forward_Activity_List**, βρες την αμέσως επόμενη δραστηριότητα *next* μετά την Αρχή και πρόσθεσέ την στην λίστα προτεραιότητας, *Solution*.
- Θέσε *check* = 1.
- Αν *trigger* = 0, θέσε το *status* της *next* του αντικειμένου της κλάσης *Rcpsp*, *status* = *true*, διαφορετικά θέσε *true* στο αντικείμενο της *Foragers*.
- Όρισε ένα vector *next_act* το οποίο θα περιέχει σε κάθε επανάληψη τις υποψήφιες δραστηριότητες για να μπουν στην λίστα.
- Πρόσθεσε του διάδοχους της Αρχής και τους διάδοχους της *next* στο *next_act*.
- Πρόσθεσε την *next* στην λίστα και θέσε *check* + +.
- Μέχρι το *check* = *inputs*.
 - Κάλυψε την **Forward_Activity_Check**, θέσε *max* = 0.0 και όρισε ένα vector *temp*.

- Για κάθε στοιχείο του *next_act*:
 - Για κάθε προαπαιτούμενη του στοιχείου το *next_act*:
 - Έλεγε αν το *status* των προαπαιτούμενων ισούται με *true*.
 - Αν το όλες οι προαπαιτούμενες του στοιχείου είναι *true* και το *prio* > *max*, επέλεξε το στοιχείο ως την δραστηριότητα και θέσε *max* = *prio*.
 - Αν *trigger* = 0, θέσε *true* το *status* της δραστηριότητας, διαφορετικά θέσε *true* στο αντικείμενο της κλάσης *Foragers*.
 - Ανανέωσε το *next_act* με τα προϋπάρχοντα στοιχεία και τους *successors* της δραστηριότητας, πλην την δραστηριότητα που επιλέχθηκε.
 - Επέστρεψε την δραστηριότητα και το *next_act*.
 - Πρόσθεσε την δραστηριότητα στην λίστα και *check* ++.
- Κάλυψε την **Forward_Resource_Check**.
 - Όρισε τις μεταβλητές τύπου *int*: *counter* = 1, και *act* = 1 και την μεταβλητή τύπου *double* *sum* = 0.0.
 - Όσο *counter* < *solution.size*
 - Κάλυψε την **Early_Start** για την δραστηριότητα στην θέση *act* της λίστας προτεραιότητας για την εύρεση του χρόνου εκκίνησής της.
 - Αν *trigger* = 0 πάρε τα *finish* από το αντικείμενο *scout_bees* διαφορετικά από το αντικείμενο της κλάσης *Foragers*.
 - Για όλους του *predecessors* της δραστηριότητας στην θέση *act*, αν το *finish* > *max*, τότε *max* = *finish*.
 - Επέστρεψε το *max* και θέσε το ως *ES*.
 - Από την χρονική στιγμή *ES* έως *ES* + *duration*
 - Έλεγε αν δεν παραβιάζονται οι διαθεσιμότητες των πόρων.
 - Αν παραβιάζονται θέσε *ES* = *ES* + 1 έως ότου να μην παραβιάζονται.
 - Πρόσθεσε τις απαιτήσεις πόρων στον πίνακα πόρων *GanntR*.
 - Αν *trigger* = 0, θέσε τον χρόνο έναρξης της δραστηριότητας στην θέση *act*, *start* = *ES* στο αντικείμενο *scout_bees*.

- Αν $trigger = 1$: θέσε $start$, $finish$, npv στο αντικείμενο της κλάσης Foragers και άθροισε στο sum τις npv της δραστηριότητας.
- Θέσε $act++$ και $counter++$.
- Αν $trigger = 0$, θέσε τον $pointer * finish$ ίσο με τον χρόνο λήξης της τελευταίας δραστηριότητας του αντικειμένου $scout_bees$.
- Αν $trigger = 1$, θέσε τον $pointer * finish$ ίσο με τον χρόνο λήξης της τελευταίας δραστηριότητας του αντικειμένου της κλάσης Foragers και θέσε τον $pointer * npv = sum$.
- Επέστρεψε τον πίνακα $GanntR$.
- Άλλαξε το μέγεθος του πίνακα πόρων $GanntR$ στην χρονική στιγμή $finish$.
- Αν $trigger = 0$, θέσε τον πίνακα $GanntR$, το vector $random$ και το $finish$ στο αντικείμενο $scout_bees$ της κλάσης In_Pop.
- Αν $trigger = 1$, θέσε τα παραπάνω στο αντικείμενο της κλάσης Foragers.

3.4.4 Βήμα 4 - Τοπική Έρευνα και Επιλογή Καλύτερων Λύσεων

Σε αυτό το βήμα, εκτελείται η δημιουργία λύσεων του Βήματος 3, αυτήν την φορά ωστόσο με $trigger = 1$. Η συνάρτηση **Local_Search** έχει ως παραμέτρους:

- $local_size$, ίσο με την αρχική παράμετρο frg ,
- $scout$, για τον προσδιορισμό του μέλους του αρχικού πληθυσμού,
- $dummy_status$,
- $dummy_start$,
- $dummy_finish$,
- $dummy_npv$ και
- $dummy_check$.

Οι $dummy$ παράμετροι έχουν ως σκοπό την ανάθεση πλασματικών τιμών στο αντικείμενο της κλάσης Foragers για την αποφυγή υπερχείλισης μνήμης. Τα βήματα του αλγόριθμου της Τοπικής Έρευνας έχουν ως εξής:

- Κάλεσε τον default constructor της κλάσης Foragers, δημιούργησε το αντικείμενο της, *forager_bees*, με μέγεθος ίσο με *initial* και αρχικοποίησε το.
- Κάλεσε τον default constructor της κλάσης Best και αρχικοποίησε το αντικείμενο της *best_bees*, με μέγεθος ίσο με *initial* και αρχικοποίησε το.
- Όρισε dummy vectors για την ανάθεση πλασματικών τιμών για τα μέλη του *forager_bees*.
- Κάλεσε την **Local_Search** για ολόκληρο τον αρχικό πληθυσμό (*initial*).
- Όρισε *best_npv* και *best_dur* την συνολική *npv* και διάρκεια αντίστοιχα της λύσης του αντικειμένου *scout_bees*.
- Θέσε *prob* ίσο με τα, διανύσματα προτεραιότητας, *prio* του *scout_bees*.
- Θέσε *sol* ίσο με την λίστα προτεραιότητας, *Solution*, του *scout_bees*.
- Θέσε τον αριθμό των *threads* = 3.
- Θέσε *flag* = 0.
- Για $i = 0$ μέχρι $i = local_size - 1$ φορές εκτέλεσε παράλληλα:
 - Αν $i = 0$ διατήρησε την αρχική λύση *scout_bees*.
 - Αν $i \leq (local_size/3)$:
 - Άλλαξε το *prob* στην *sol* στις δραστηριότητες 1 έως $n/2$, (πρώτος τρόπος Σχήμα 3.6).
 - Κάλεσε τον constructor της Foragers για την ανάθεση τιμών στα μέλη του *forager_bees*.
 - Κάλεσε την **SSGS_Forward** με *trigger* = 1.
 - Αν $i \leq (local_size/3) \times 2 - 1$:
 - Άλλαξε το *prob* στην *sol* στις δραστηριότητες $n/2 + 1$ έως n , (δεύτερος τρόπος Σχήμα 3.6).
 - Κάλεσε τον constructor της Foragers για την ανάθεση τιμών στα μέλη του *forager_bees*.
 - Κάλεσε την **SSGS_Forward** με *trigger* = 1.
 - Αν $i > (local_size/3) \times 2 - 1$:

- Άλλαξε το *prob* στο 70% του *sol* σε τυχαίες δραστηριότητες.
- Κάλεσε τον constructor της Foragers για την ανάθεση τιμών στα μέλη του *forager_bees*.
- Κάλεσε την **SSGS_Forward** με *trigger* = 1.
- Αν η *npv* του *forager_bees* είναι $>$ της *best_npv* και η διάρκειά του είναι \leq της *best_dur*, τότε:
 - *best_npv* ισούται με την *npv* του *forager_bees*.
 - *best_dur* ισούται με την διάρκεια του *forager_bees*.
 - *flag* = 1.
- Αν *flag* = 1:
 - Βάλε τις τιμές του *forager_bees* στο *scout_bees*.
 - Ξανά κάλεσε της **Local_Search** για το ίδιο *scout_bees*.
- Αν *flag* = 0:
 - Βάλε τις τιμές του *forager_bees* στο *best_bees*.

3.4.5 Βήμα 5 - Καθυστέρηση Δραστηριοτήτων

Όπως έχει αναφερθεί, το βήμα αυτό πραγματοποιείται σε δύο διαδοχικά στάδια. Οι συναρτήσεις που καλούνται σε κάθε στάδιο είναι οι εξής:

1. **Network_Delay**,
 - **Find_Network_Set**,
 - **delay** και
 - **Push_Delay**.
2. **Schedule_Delay**,
 - **Find_Schedule_Set**,
 - **Schedule_Set_2**,
 - **delay** και

- **Push_Delay.**

Παρακάτω θα αναφερθούν οι παράμετροι των συναρτήσεων που θα χρησιμοποιηθούν:

Η **Network_Delay** έχει ως παραμέτρους:

- Την τιμή της καθυστέρησης, *deadline* και
- Τον δείκτη *bestbees* για τον καθορισμό του αντικειμένου.

Η **Find_Network_Set** αποτελεί την συνάρτηση που αναλύθηκε στο Σχήμα 3.8 και έχει ως παραμέτρους:

- Τον δείκτη *current*, η δραστηριότητα που εξετάζεται,
- Τον δείκτη *start*, η αρχική δραστηριότητα,
- Τον πίνακα *pt1*, το *set1*,
- Τον πίνακα *pt2*, το *set2*,
- Τον pointer **total_npv*, συνολική *npv*
- Τον δείκτη *bestbees*,
- Τον δείκτη *status*, έναυσμα για έλεγχο προαπαιτούμενων
- Τον δείκτη *count1*, μέγεθος *set1* και
- Τον δείκτη *count2*, μέγεθος *set2*.

Η **delay** έχει ως παραμέτρους:

- Το *set1*,
- Το *set2* και
- Τον δείκτη *bestbees*.

Η **Push_Delay** έχει ως παραμέτρους:

- Τον δείκτη *bestbees*.
- Την καθυστέρηση **delay**,

- Το *set1* και
- Την διορία του χρονοπρογράμματος *deadline*.

Η **Find_Schedule_Set** είναι η συνάρτηση που αναλύθηκε στο Σχήμα 3.9 και έχει ως παραμέτρους:

- Τον δείκτη *current*, η δραστηριότητα που εξετάζεται,
- Τον δείκτη *start*, η αρχική δραστηριότητα,
- Τον πίνακα *pt1*, το *set1*,
- Τον pointer **total_npv*, συνολική *npv*
- Τον δείκτη *bestbees*,
- Τον δείκτη *status*, έναυσμα για έλεγχο προαπαιτούμενων και
- Τον δείκτη *count1*, μέγεθος *set1*.

Τέλος η **Schedule_Set_2** είναι η συνάρτηση που αναλύθηκε στο Σχήμα 3.10 και έχει ως παραμέτρους:

- Το *set1* και
- Τον δείκτη *bestbees*.

Εφόσον αναφέρθηκαν οι παράμετροι των συναρτήσεων που θα χρησιμοποιηθούν, παρακάτω ακολουθεί η βήμα προς βήμα ανάλυση του αλγόριθμου:

- Κάλεσε τον default constructor της κλάσης Elite, δημιούργησε το αντικείμενο της, *elite_bees*, με μέγεθος ίσο με *initial* και αρχικοποίησε το.
- Θέσε $deadline = 1.15$ φορές την διάρκεια της λύσης της *best_bees* ως την διορία του έργου.
- Κάλεσε την **Network_Delay**.
- Κάλεσε τον constructor της Elite κλάσης και ανάθεσε τις τιμές του *best_bees* στο *elite_bees*.
- Θέσε τον χρόνο έναρξης και λήξης της πλασματικής δραστηριότητας Τέλος ίση με το *deadline*.

- Αύξησε το μέγεθος του πίνακα πόρων *GanntR* έως την χρονική στιγμή *deadline*.
- Θέσε *change_count* = 1.
- Όσο *change_count* > 0.
 - Για όλες τις δραστηριότητες της λίστας προτεραιότητας, αρχίζοντας από την τελευταία.
 - Αν η χρηματοροή της δραστηριότητας είναι αρνητική:
 - Θέσε *total* τη συνολική *npv* του *set1*.
 - Κάλεσε την **Find_Network_Set**.
 - Υπολόγισε τα *set1* και *set2* σύμφωνα με το Σχήμα 3.8.
 - Αν η συνολική *npv* του *set1* είναι αρνητική:
 - Κάλεσε την **delay** για την εύρεση της μέγιστης επιτρεπτής καθυστέρησης, δ .
 - Αν $\delta > 0$, κάλεσε την **Push_Delay** για την εφαρμογή της καθυστέρησης.
 - Για κάθε δραστηριότητα του *set1*:
 - Αφαίρεσε από το *GanntR* για κάθε πόρο, όλους τους πόρους που απαιτεί η δραστηριότητα.
 - Θέσε *ok* = *false*.
 - Έως $\delta \leq 0$ ή *ok* = *true*:
 - Για κάθε δραστηριότητα του *set1*:
 - Θέσε νέους χρόνους έναρξης και λήξης, $start + \delta$ και $finish + \delta$.
 - Έλεγξε αν στους νέους χρόνους υπάρχει έστω και μία παραβίαση πόρου για οποιονδήποτε πόρο.
 - Αν υπάρχει, τότε $\delta = \delta - 1$.
 - Αν δεν υπάρχει τότε *ok* = *true*.
 - Αν *ok* = *true*, τότε για κάθε δραστηριότητα του *set1*:
 - Θέσε στο *elite_bees* τους νέους χρόνους έναρξης και λήξης.
 - Υπολόγισε τις νέες *npv*.
 - Θέσε στο *elite_bees* το ανανεωμένο *GanntR*.

- Αν πραγματοποιήθηκε καθυστέρηση, θέσε $change_count++$.
- Κάλεσε την **Schedule_Delay**.
- Θέσε $change_count = 1$.
- Όσο $change_count > 0$.
 - Για όλες τις δραστηριότητες της λίστας προτεραιότητας, αρχίζοντας από την τελευταία.
 - Αν η χρηματοροή της δραστηριότητας είναι αρνητική:
 - Θέσε $total$ τη συνολική npv του $set1$.
 - Κάλεσε την **Find_Schedule_Set**.
 - Υπολόγισε το $set1$ σύμφωνα με το Σχήμα 3.9.
 - Υπολόγισε το $set2$ σύμφωνα με το Σχήμα 3.10.
 - Αν η συνολική npv του $set1$ είναι αρνητική:
 - Επανέλαβε κατά γράμμα την διαδικασία που αναφέρθηκε από το βήμα κλήσης της **delay** μέχρι το βήμα $change_count++$.
- Άθροισε την συνολική npv των δραστηριοτήτων.
- Αποθήκευσε την τιμή της συνολικής npv στο $elite_bees$.

3.4.6 Βήμα 6 - Έλεγχος Χρονοπρογράμματος

Σε αυτό το βήμα καλείται η συνάρτηση **Check_Schedule** με μόνη παράμετρο την $bees$, δηλαδή το μέγεθος του αρχικού πληθυσμού. Τα βήματα είναι τα εξής:

- Κάλεσε την **Check_Schedule**.
- Για όλες τις λύσεις του αλγόριθμου, $i = 0$ μέχρις $i = bees - 1$:
 - Έλεγε αν οι $successors$ της πλασματικής δραστηριότητας Αρχή ξεκινάνε από τη χρονική στιγμή 0.
 - Αν κανένας $successor$ δεν ξεκινά από την 0, για κάθε $successor$:
 - Πάρε τους χρόνους έναρξης τους.

- Βρες τον μικρότερο από τους χρόνους έναρξης, *min*.
- Για κάθε δραστηριότητα:
 - Διόρθωσε τον χρόνο έναρξης, *start - min*.
 - Διόρθωσε τον χρόνο λήξης, *finish - min*.
 - Υπολόγισε εκ νέου τις *npv*.
 - Θέσε τα παραπάνω στο *elite_beas*.
- Υπολόγισε εκ νέου την συνολική *npv* της λύσης.
- Διόρθωσε τον πίνακα πόρων *GanntR* ώστε να ξεκινά από την χρονική στιγμή 0 και αυτό.
- Θέσε το *GanntR* στο *elite_beas*.

3.4.7 Βήμα 7 - Εξαγωγή Αποτελεσμάτων

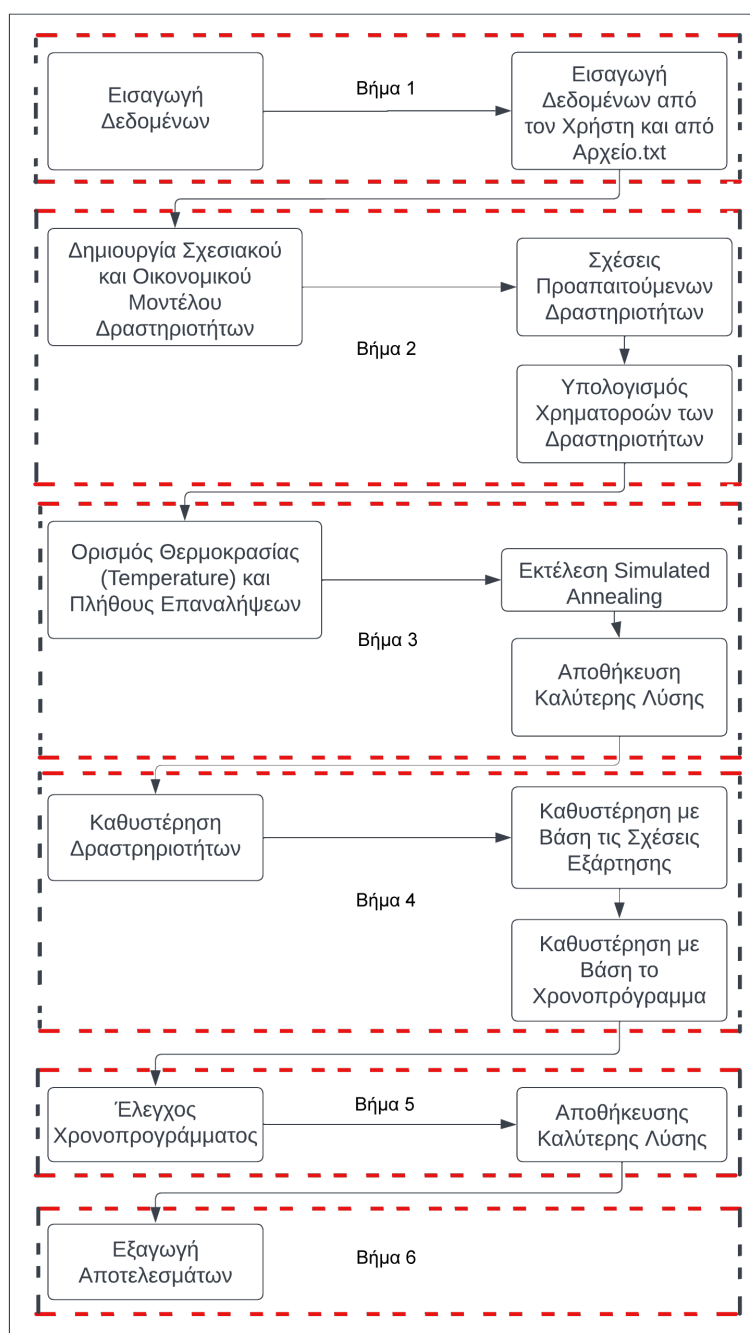
Η εξαγωγή αποτελεσμάτων θα αναφερθεί πιο αναλυτικά στο επόμενο κεφάλαιο όπου αφορά την Γραφική Διεπαφή Χρήστη. Ωστόσο η υποενότητα αυτή αποτελείται από τα εξής βήματα:

- Κάλεσε τον default constructor της κλάσης Opt, δημιούργησε το αντικείμενο της, *final*, με μέγεθος ίσο με 1 και αρχικοποίησε το.
- Όρισε *best_npv* την συνολική *npv* της λύσης 0 του αντικειμένου *elite_beas*.
- Όρισε *pos = 0*
- Για όλες τις λύσεις αποθηκευμένες στο *elite_beas*:
 - Αν η συνολική *npv* του *elite_beas* είναι $> best_npv$:
 - *best_npv* ισούται με τη συνολική *npv* του *elite_beas*.
 - *pos* ισούται με το *elite_beas*.
- Κάλεσε τον constructor της Opt για την ανάθεση των τιμών της *pos* στα μέλη του *final*.
- Εξήγαγε την τελική λύση σε αρχείο .CSV.
- Εξήγαγε διαγράμματα χρήσης πόρων.

3.5 Ανάλυση Simulated Annealing

Η προσέγγιση του Simulated Annealing διαφέρει από αυτήν της αποικίας μελισσών. Η κύρια διαφορά είναι ότι ο αλγόριθμος δεν λειτουργεί πλέον με έναν αρχικό πληθυσμό αλλά ενσωματώνει τα βήματα 3 και 4 του Σχήματος 3.4 στην διαδοχική σύγκριση δύο λύσεων και την ανανέωσή τους, Σχήμα 3.14. Η κλάση Rcpsr παραμένει η ίδια ενώ η κλάση In_Pop μετονομάζεται σε SA. Συγκεκριμένα τα βήματα είναι ως εξής:

- Όρισε την θερμοκρασία του αλγορίθμου ως τον pointer $Temperature = 10,000$.
- Όρισε το πλήθος των επαναλήψεων του αλγορίθμου ως $loops$.
- Δημιούργησε μία αρχική τρέχουσα λύση $sa1$ με την κλήση της συνάρτησης **SSGS_Forward** με μία πιθανότητα $probability_1$ από 0.001 έως 0.999.
- Δημιούργησε την λύση $best = sa1$ για την αποθήκευση της καλύτερης λύσης.
- Δημιούργησε μία δεύτερη γειτονική λύση $sa2$ με την κλήση της συνάρτησης **SSGS_Forward** με μία πιθανότητα $probability_2$ από 0.001 έως 0.999..
- Όρισε $n = 0$ και για όσο $n < loops$:
 - Όρισε $diff$ την διαφορά των npv των δύο λύσεων, $diff = npv_{sa1} - npv_{sa2}$.
 - Αν $diff < 0.0$:
 - Τότε η τρέχουσα $sa1$ ισούται με την $sa2$.
 - Αν $npv_{sa1} > npv_{best}$, τότε η $best$ ισούται με την $sa1$.
 - Αν $diff > 0.0$:
 - Θέσε την πιθανότητα $prob = \exp^{diff/Temperature}$.
 - Αν η $prob > probability_1$, τότε η τρέχουσα λύση $sa1$ ισούται με την $sa2$.
 - Αν η $prob < probability_1$, τότε διατηρείται η τρέχουσα λύση $sa1$ ωστόσο με $probability_1 = 1 - prob$.
 - Για κάθε 5 επαναλήψεις θέσε $Temperature = Temperature * 0.98$.
 - Θέσε $n = n + 1$.

Σχήμα 3.14: Διάγραμμα Ροής - *Simulated Annealing*.

ΚΕΦΑΛΑΙΟ

— 4 —

ΥΛΟΠΟΙΗΣΗ ΓΡΑΦΙΚΗΣ ΔΙΕΠΑΦΗΣ ΧΡΗΣΤΗ - GUI

4.1 Προσέγγιση Μέσω Διεπαφής

Για την διευκόλυνση του χρήστη και την άμεση αλληλεπίδραση με των αρχικών μεταβλητών, δημιουργήθηκε μία Γραφική Διεπαφή Χρήστη (Graphical User Interface - GUI) με την βοήθεια της βιβλιοθήκης wxWidgets. Κύριο χαρακτηριστικό στην υλοποίηση ενός GUI, είναι η διαφοροποίηση στην ροή εκτέλεσης του προγράμματος. Σε μία τυπική εφαρμογή δομημένου ή διαδικαστικού προγραμματισμού (Procedural programming), η ροή του προγράμματος είναι προκαθορισμένη, ελεγχόμενη από τον χρήστη και είναι σύνηθες να ξεκινά από την αρχή του πηγαίου κώδικα και να λήγει στο τέλος του.

Η δομή του GUI παρεκκλίνει από τα χαρακτηριστικά του διαδικαστικού προγραμματισμού και αντ' αυτού η προσέγγισή του προγραμματισμού της είναι αυτή του χειρισμού γεγονότων (Event - driven programming). Σε μία εφαρμογή με τη συγκεκριμένη δομή η εκτέλεση του προγράμματος καθορίζεται από το σύστημα. Ο προγραμματιστής είναι υπεύθυνος για την δημιουργία και δήλωση εναυσμάτων (triggers) τα οποία εκτελούνται όταν καλούνται συγκεκριμένα γεγονότα. Στη συνέχεια το σύστημα αναμένει την ενέργεια του χρήστη για να ξεκινήσουν τα γεγονότα και τελικά το σύστημα ανταποκρίνεται σε αυτά, ελέγχοντας έτσι την ροή της εκτέλεσης του προγράμματος.

Για κάθε στοιχείο του GUI όπου ο χρήστης μπορεί να αλληλεπιδράσει αντιστοιχεί μία συνάρτηση. Ένα στοιχείο, για παράδειγμα, μπορεί να είναι ένα κουμπί (Button), η μπάρα επιλογής του μενού του προγράμματος (MenuBar), επιλογή αριθμού αυξομειώνοντας ανά βαθμίδες (SpinControl), το πάτημα του δεξιού κλικ στο ποντίκι και ούτω καθεξής. Κατά την επιλογή ενός στοιχείου, καλείται το γεγονός όπου στην συνέχεια καλεί την συνάρτηση.

Πίνακας 4.1: Παράδειγμα δομής συναρτήσεων του GUI.

Συνάρτηση	Χειριστής Γεγονότων	Λειτουργία
OnMenuOpen (wxCommandEvent& evt)	wxCommandEvent	Γενική Χρήση
OnClose (wxCloseEvent& evt)	wxCloseEvent	Κλείσιμο GUI
OnResourceSpin (wxSpinEvent& evt)	wxSpinEvent	Αυξομείωση αριθμού

Υπάρχουν δύο τρόποι για τον χειρισμό γεγονότων με την βιβλιοθήκη wxWidgets. Ο πρώτος είναι με την χρήση μακροεντολών και δήλωση ενός Πίνακα Γεγονότων (EVENT_TABLE), Σχήμα 4.1, ο οποίος επιτρέπει την δέσμευση των γεγονότων και τον χειριστών γεγονότων (event handler) στατικά και απαιτεί τον προσδιορισμό της διεύθυνσης του στοιχείου του GUI. Ο δεύτερος γίνεται με την χρήση της εντολής Bind της κλάσης wxEventHandler η οποία επιτρέπει την δυναμική δέσμευση των παραπάνω. Ο σκοπός ενός χειριστή γεγονότων είναι ο καθορισμός του τύπου του γεγονότος.

```
wxBEGIN_EVENT_TABLE(MainFrame, wxFrame)
...
EVT_MENU(101, MainFrame::OnMenuOpen)
EVT_MENU(103, MainFrame::OnMenuExit)
EVT_MENU(110, MainFrame::ExportCSV)
EVT_CLOSE(MainFrame::OnClose)
...
EVT_MENU(130, MainFrame::OnHelp)
EVT_MENU(120, MainFrame::OnDefineRates)
EVT_MENU(121, MainFrame::OnSensitivity)
EVT_MENU(122, MainFrame::OnSense)
EVT_MENU(111, MainFrame::ExportSA)
...
wxEND_EVENT_TABLE()
```

Σχήμα 4.1: Πίνακας Γεγονότων.

Από το Σχήμα 4.1, λαμβάνοντας ως παράδειγμα την πρώτη προσθήκη στον πίνακα γεγονότων, η μορφή δήλωσης ενός γεγονότος μπορεί να αναλυθεί ως εξής:

- EVT_Menu, δήλωση του τύπου του γεγονότος. Αντιστοιχεί σε στοιχείο τύπου menu.

- 101, δήλωση της διεύθυνσης του στοιχείου στο GUI ώστε να γίνει η αντιστοιχία γεγονότος - στοιχείου.
- `MainFrame::OnMenuOpen`, δήλωση της συνάρτησης που αντιστοιχεί στο συγκεκριμένο γεγονός.

Παρατηρείται πως στην τέταρτη προθήκη, `EVT_CLOSE`, παραλείπεται η δήλωση της διεύθυνσης. Αυτό δεν αποτελεί πρόβλημα καθώς το κλείσιμο του παραθύρου δεν απαιτεί κάποια δήλωση διότι αποτελεί έμφυτο χαρακτηριστικό του GUI.

Από το Σχήμα 4.2, από την πρώτη σειρά του σχήματος, μπορεί να αναλυθεί η δομή της εντολή `Bind` η οποία είναι η εξής:

- `wxEVT_BUTTON`, δήλωση του τύπου του στοιχείου στο GUI (Event Tag). Αντιστοιχεί σε στοιχείο τύπου `button`.
- `MainFrame::OnResourceGraph`, δήλωση της συνάρτησης που αντιστοιχεί στο συγκεκριμένο γεγονός.
- `this`, δήλωση ενός pointer που αντιστοιχεί στο αντικείμενο που θα χειριστεί το γεγονός. Με τον όρο `this` στην συγκεκριμένη περίπτωση εννοείται το παράθυρο του GUI που είναι δηλωμένη η μεταβλητή `plot`.

```
plot->Bind(wxEVT_BUTTON, &MainFrame::OnResourceGraph, this);  
init->Bind(wxEVT_BUTTON, &MainFrame::OnSolInit, this);
```

Σχήμα 4.2: Η εντολή `Bind`.

Για την ομαλή λειτουργία του GUI, εφαρμόζεται η αρχή του παράλληλου προγραμματισμού με την δήλωση ενός `thread backgroundThread`. Σκοπός του `backgroundThread` είναι να παραλληλίζει τις διεργασίες που εκτελούνται ώστε να μην "παγώνει" η διεπαφή χρήστη σε διαδικασίες όπου απαιτούν αρκετό χρόνο και μεγάλα ποσά μνήμης. Έτσι, το GUI και οποιαδήποτε ενημέρωση σε αυτό, όπως μία ράβδος προόδου, εκτελούνται από το κύριο `thread` (Master thread) ενώ οι λοιπές διεργασίες να εκτελούνται από το `backgroundThread`.

4.2 Οι Κλάσεις του GUI

Τα αρχεία που αποτελούν το πρόγραμμα, ο σκοπός τους καθώς και τα περιεχόμενά τους. Το πρόγραμμα στο σύνολό του αποτελείται από έξι αρχεία:

1. MainFrame.h
2. MainFrame.cpp
3. rcpsp.h
4. bba.h
5. chartcontrol.h
6. chartcontrol.cpp

4.2.1 Το αρχείο MainFrame.h

Στο αρχείο MainFrame.h περιέχονται οι κλάσεις υπεύθυνες για το αρχικό παράθυρο του GUI και η δήλωση των συναρτήσεων του αλγόριθμου. Συγκεκριμένα αποτελείται από:

- Η κλάση MainFrame, παράγωγη της κλάσης wxFrame, η οποία είναι υπεύθυνη για την δημιουργία του παραθύρου του GUI,
- Η δήλωση της ονομασίας των συναρτήσεων του παραθύρου, υπό την κλάση MainFrame,
- Η δήλωση της ονομασίας των συναρτήσεων που αναφέρθηκαν στην ενότητα 3.4, υπό την κλάση MainFrame,
- Τα protected δεδομένα της κλάσης MainFrame, τα οποία περιλαμβάνουν οτιδήποτε εμφανίζεται στο παράθυρο και το thread backgroundThread,
- Η δήλωση του πίνακα γεγονότων,
- Η κλάσεις NumberDlg και RateDlg, παράγωγες της κλάσης wxDialog,
- Οι συναρτήσεις και τα protected δεδομένα τους των προαναφερθέντων κλάσεων.

Η κλάσεις τύπου wxDialog χρησιμοποιούνται στην περίπτωση που ο χρήστης επιθυμεί να πραγματοποιήσει ανάλυση ευαισθησίας. Η χρήση τους είναι να ανοίγουν ένα

νέο μικρό παράθυρο όπου μπορεί να γίνει η ανάθεση της τιμής του πλήθους των διαφορετικών επιτοκίων (NumberDlg) και οι τιμές αυτών (RateDlg).

4.2.2 Το αρχείο MainFrame.cpp

Το αρχείο MainFrame.cpp, το οποίο αποτελεί το αρχείο υλοποίησης του MainFrame.h και αποτελεί το μεγαλύτερο αρχείο του πηγαιού κώδικα, περιέχει:

- Την κλάση App, παράγωγη της wxApp, υπεύθυνη για την εκκίνηση του GUI,
- Την γεννήτρια των τυχαίων αριθμών, *random_device rd*,
- Την μεταβλητή *filename* τύπου string για την αποθήκευση της τοποθεσίας του αρχείου που θα χρησιμοποιηθεί,
- Την ανάλυση του Πίνακα Γεγονότων, Σχήμα 4.1,
- Τις μεταβλητές του αρχικού πληθυσμού *initial*, *frg*, *rate*, *s_rates*,
- Την μεταβλητή *res* για την επιλογή του πόρου του οποίου η γραφική απεικόνιση θα παραχθεί και
- Η ανάλυση όλων των συναρτήσεων της κλάσης MainFrame που ορίστηκαν στο αρχείο header MainFrame.h.

4.2.3 Το αρχείο rcpsp.h

Στο αρχείο rcpsp.h δηλώνεται η κλάση Rcrcp υπεύθυνη για την αποθήκευση των στοιχείων του αρχείου που θα χρησιμοποιηθεί. Τα περιεχόμενα είναι:

- Δήλωση των μελών της κλάσης όπως φαίνεται στον Πίνακα 3.2,
- Δήλωση των συναρτήσεων της κλάσης Rcrcp,
- Δήλωση του αντικειμένου της, *pointer element*, και
- Δήλωση μεταβλητής *inputs*, *resourceCap*, *MaxTime*, *TotalNPV*, *rcost*, *ConstantV* και *ResourceCost*,

4.2.4 Το αρχείο `bba.h`

Στο αρχείο `bba.h` δηλώνονται με τη σειρά που αναφέρονται οι κλάσεις υπεύθυνες για την αποθήκευση των λύσεων του αλγόριθμου:

- Η κλάση `In_Pop`, τα μέλη της, Πίνακας 3.3 και οι συναρτήσεις της,
- Δήλωση του αντικειμένου της `In_Pop`, pointer `scout_bees`,
- Η κλάση `Foragers`, τα μέλη της, Πίνακας 3.4 και οι συναρτήσεις της,
- Δήλωση του αντικειμένου της `Foragers`, τριπλός pointer `forager_bees`,
- Η κλάση `Best`, τα μέλη της, Πίνακας 3.5 και οι συναρτήσεις της,
- Δήλωση του αντικειμένου της `Best`, pointer `best_bees`,
- Η κλάση `Elite`, τα μέλη της, Πίνακας 3.6 και οι συναρτήσεις της,
- Δήλωση του αντικειμένου της `Elite`, pointer `elite_bees`,
- Η κλάση `Opt`, τα μέλη της, Πίνακας 3.6 και οι συναρτήσεις της και
- Δήλωση του αντικειμένου της `Opt`, pointer `final`.

4.2.5 Το αρχείο `chartcontrol.h`

Στο αρχείο `chartcontrol.h`, δηλώνονται οι κλάσεις υπεύθυνες για την δημιουργία γραφημάτων χρήσης πόρων και συμπεριφοράς της ολικής *nrv* σε μία ανάλυση ευαισθησίας. Τα περιεχόμενα είναι τα εξής:

- Δήλωση κλάσης `ChartControl`, παράγωγης της κλάσης `wxWindow` και
- Δήλωση κλάσης `ChartControlSA`, παράγωγης της κλάσης `wxWindow`.

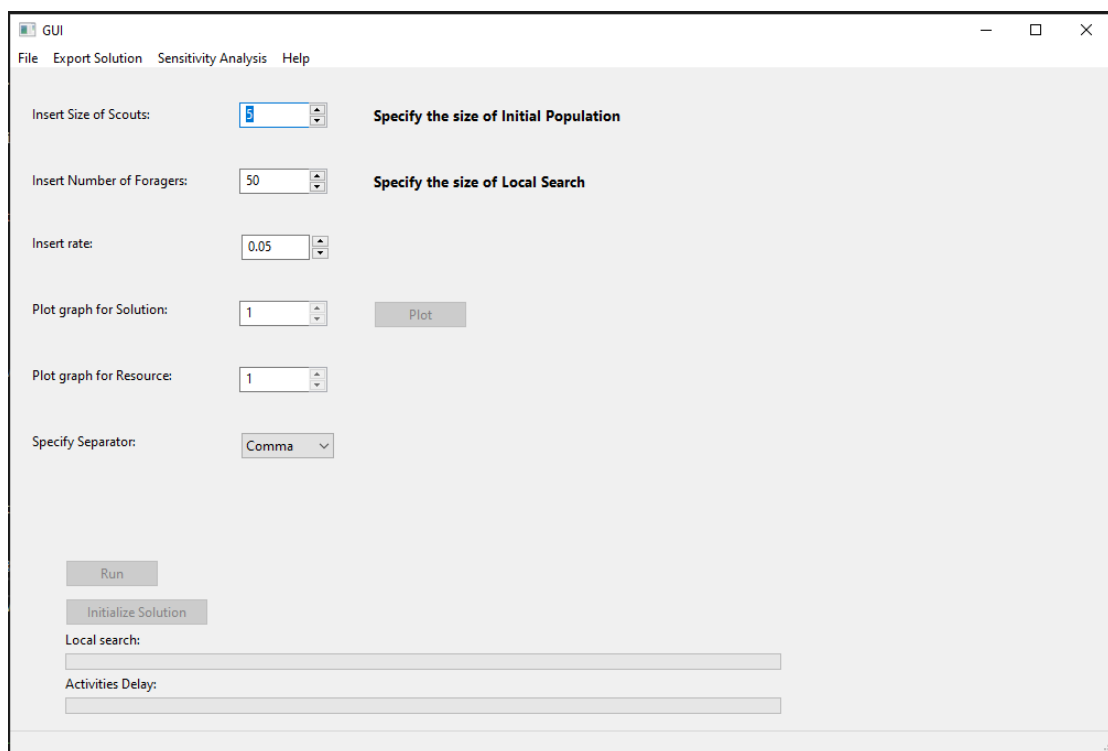
Η κλάση `ChartControl` καλείται όταν ο χρήστης θέλει να παράξει διαγράμματα χρήσης πόρων, ενώ η κλάση `ChartControlSA` όταν ο χρήστης θέλει να παράξει διάγραμμα *nrv* - επιτοκίων.

4.2.6 Το αρχείο chartcontrol.cpp

Τέλος, το αρχείο chartcontrol.cpp αποτελεί το αρχείο υλοποίησης του chartcontrol.h και περιέχει την ανάπτυξη των συναρτήσεων των κλάσεων ChartControl και ChartControlSA.

4.3 Χρήση του GUI

Κατά την εκκίνηση του προγράμματος ανοίγει το παράθυρο του Σχήματος 4.3.

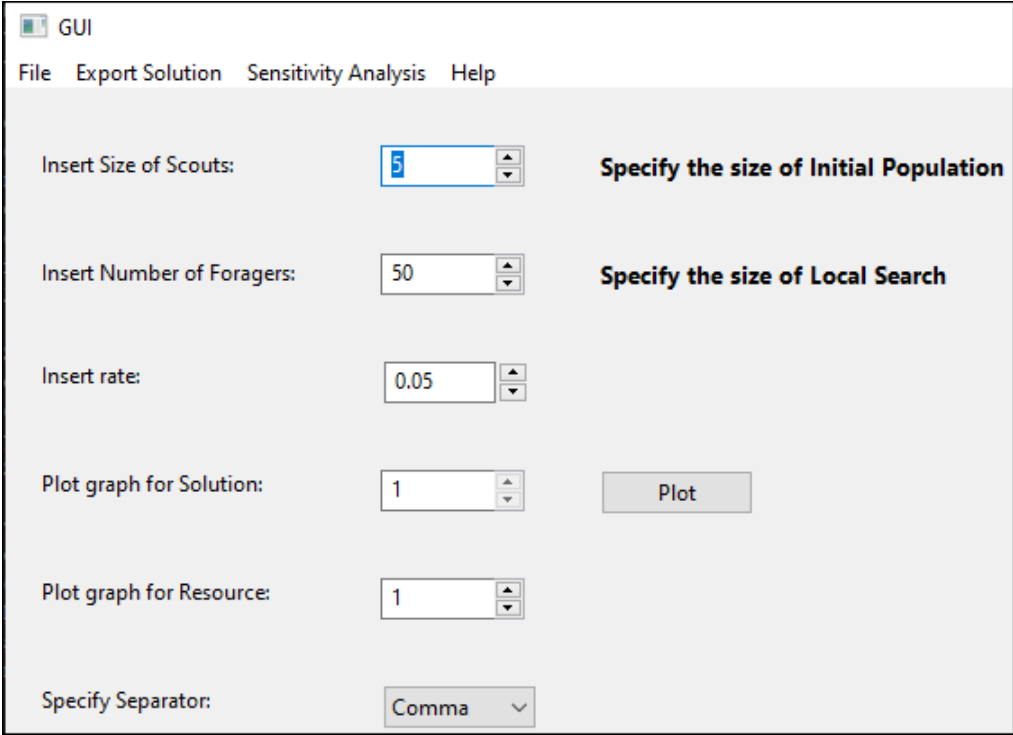


Σχήμα 4.3: Το GUI του αλγορίθμου.

Ο χρήστης μπορεί να διακρίνει την μπάρα επιλογών (MenuBar) με κάθε επιλογή να έχει το δικό της πτυσσόμενο μενού. Το μενού αναλύεται παρακάτω, στο Σχήμα 4.4 φαίνεται η μπάρα επιλογών εστιασμένη και στο Σχήμα 4.5 τα πτυσσόμενα μενού κάθε επιλογής:

- File, επιλογή αρχείου ή εξόδου του προγράμματος

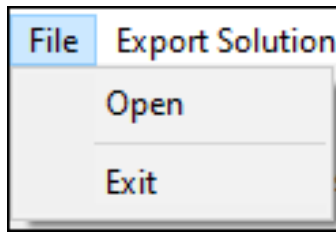
- Open, εκκίνηση περιήγησης και επιλογής του αρχείου,
- Exit, κλείσιμο προγράμματος
- Export Solution, εξαγωγή αποτελεσμάτων,
 - Export to CSV, εξαγωγή λύσης,
 - Export SA, εξαγωγή λύσεων ανάλυσης ευαισθησίας,
- Sensitivity Analysis, επιλογές σχετικά με την ανάλυση ευαισθησίας,
 - Define Rates, ορισμός πλήθους και τιμών επιτοκίων,
 - Run Sensitivity Analysis, εκκίνηση ανάλυσης ευαισθησίας,
 - Plot Sensitivity Analysis, σχεδίαση γραφήματος ανάλυσης ευαισθησίας,
- Help, διευκόλυνση του χρήστη,
 - Steps, εμφάνιση βημάτων.



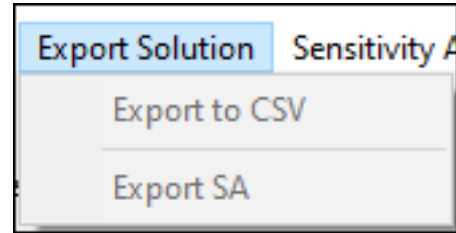
The screenshot shows a software window titled "GUI" with a menu bar containing "File", "Export Solution", "Sensitivity Analysis", and "Help". The main area contains several input fields and buttons:

- Insert Size of Scouts:** A numeric input field with the value "5" and a spinner control. To its right is the text "Specify the size of Initial Population".
- Insert Number of Foragers:** A numeric input field with the value "50" and a spinner control. To its right is the text "Specify the size of Local Search".
- Insert rate:** A numeric input field with the value "0.05" and a spinner control.
- Plot graph for Solution:** A numeric input field with the value "1" and a spinner control. To its right is a button labeled "Plot".
- Plot graph for Resource:** A numeric input field with the value "1" and a spinner control.
- Specify Separator:** A dropdown menu currently showing "Comma".

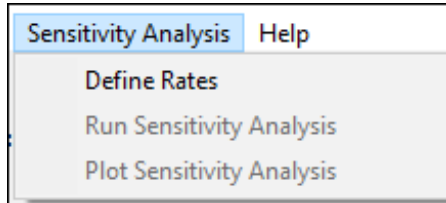
Σχήμα 4.4: Το GUI του αλγορίθμου εστιασμένο.



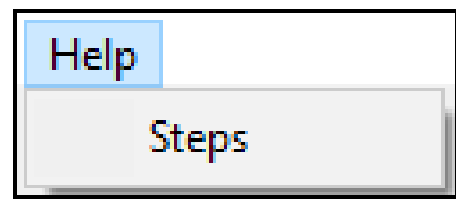
(α') Μενού File.



(β') Μενού Export Solution.



(γ') Μενού Sensitivity Analysis.



(δ') Μενού Help.

Σχήμα 4.5: Μπάρα επιλογών (MenuBar) του GUI.

4.3.1 Εισαγωγή δεδομένων

Για την εισαγωγή των δεδομένων του αρχικού πληθυσμού ο χρήστης χρειάζεται να γίνουν τα εξής βήματα ανεξαρτήτως σειράς:

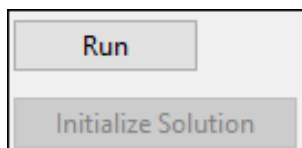
- Επιλογή του αρχείου με την επιλογή "Open" σύμφωνα με το Σχήμα 4.5α', κλήση συνάρτησης **OnMenuOpen**,
- Ορισμός μεγέθους αρχικού πληθυσμού (initial) στην επιλογή "Insert Size of Scouts", Σχήμα 4.4, κλήση συνάρτησης **OnInitialSpin**,
- Ορισμός μεγέθους Τοπικής Έρευνας (frg) στην επιλογή "Insert Number of Foragers", κλήση συνάρτησης **OnForagerSPCTRL**,
- Ορισμός τιμής προεξοφλητικού επιτοκίου (rate) στην επιλογή "Insert rate", κλήση συνάρτησης *OnRateSpin*.

4.3.2 Εκτέλεση του αλγόριθμου

Μόλις γίνει η επιλογή του αρχείου, το κουμπί "Run" ενεργοποιείται και το πρόγραμμα είναι έτοιμο να εκτελέσει τον αλγόριθμο.

Απλή Εκτέλεση του Προγράμματος

Με το πάτημα του κουμπιού καλείται η συνάρτηση **OnRun**. Στην περίπτωση του διαδικαστικού προγραμματισμού, η συγκεκριμένη συνάρτηση θα αποτελούσε την **main** συνάρτηση όπου εκτελείται ο πηγαίος κώδικας.



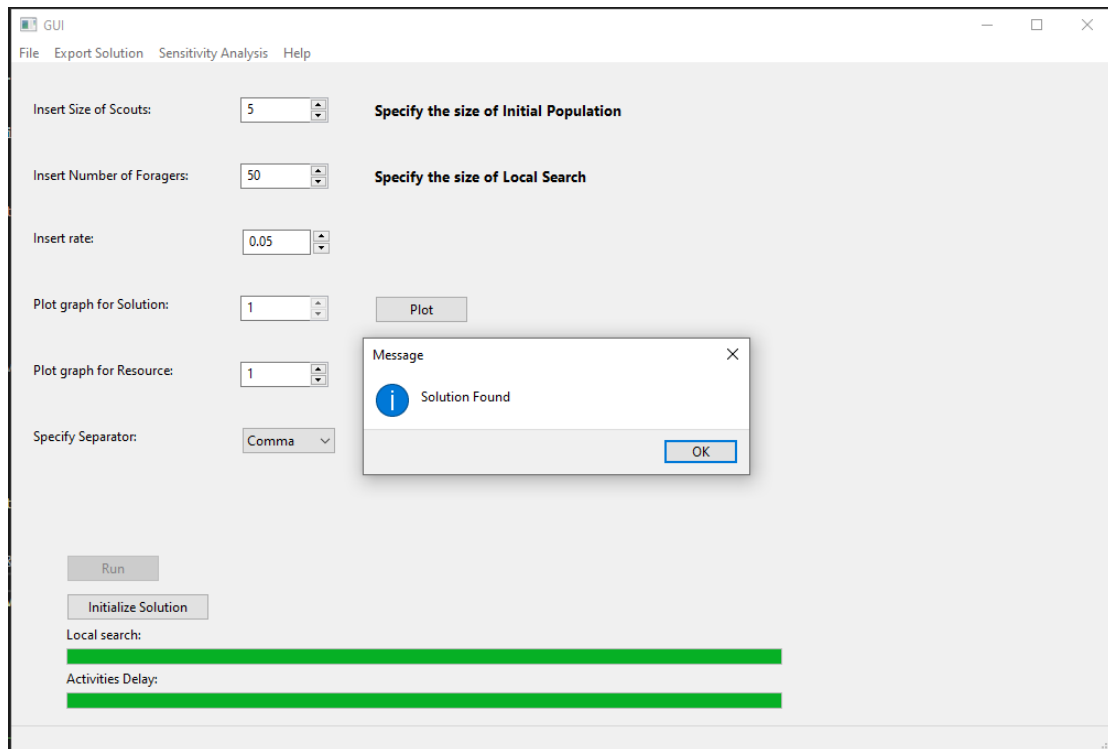
Σχήμα 4.6: Το κουμπί "Run" ενεργοποιημένο.

Κατά την διάρκεια της εκτέλεσης του αλγόριθμου, το κουμπί "Run" παραμένει απενεργοποιημένο για την αποφυγή ανάκλησης της συνάρτησης **OnRun** και κατά συνέπεια συντριβής του προγράμματος. Η **OnRun** συγκεκριμένα:

- Ορίζει το εύρος της ράβδου προόδου του Local Search και της ράβδου Activity Delay, Σχήμα 4.7,
- Δημιουργεί το thread backgroundThread,
- Καλεί την lambda συνάρτηση για την **wxGetApp().CallAfter** για την ανανέωση της ράβδου προόδου του Τοπικής Έρευνας μετά από κάθε κλήση της **Local_Search**,
- Καλεί την lambda συνάρτηση για την **wxGetApp().CallAfter** για την ανανέωση της ράβδου προόδου του καθυστέρησης δραστηριοτήτων πριν από κάθε κλήση της **Network_Delay**,
- Ενεργοποιεί την επιλογή "Export to CSV", Σχήμα 4.5β',
- Ενεργοποιεί το κουμπί "Plot" για την εξαγωγή διαγραμμάτων χρήσης πόρων Σχήμα 4.8, και
- Ενεργοποιεί το κουμπί "Initialize Solution" για την αρχικοποίηση των παραμέτρων.

Local search:	
Activities Delay:	

Σχήμα 4.7: Ράβδοι προόδου του GUI.



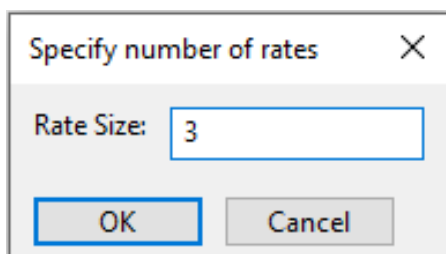
Σχήμα 4.8: Το GUI του αλγορίθμου μετά την ολοκλήρωση του υπολογισμού των λύσεων.

Εκτέλεση Ανάλυση Ευαισθησίας

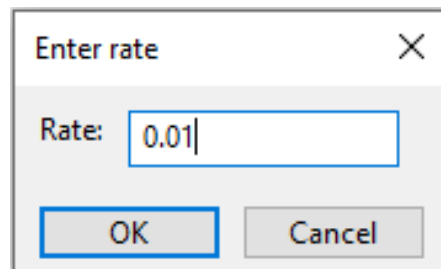
Στην περίπτωση αυτήν ο χρήστης πρέπει να επιλέξει την από το μενού στο Σχήμα 4.5γ' την επιλογή "Define Rates" η οποία θα καλέσει την συνάρτηση **OnDefineRates** η οποία:

- Ανοίγει ένα προσωρινό παράθυρο (window dialogue) για τον προσδιορισμό του πλήθους των διαφορετικών προεξοφλητικών επιτοκίων Σχήμα 4.9α',
- Ανοίγει ένα προσωρινό παράθυρο, όσες φορές όσο το πλήθος των επιτοκίων, για τον προσδιορισμό των τιμών των επιτοκίων, s_rates , Σχήμα 4.9β' και

- Ενεργοποιεί την επιλογή "Run Sensitivity Analysis".



(α') Παράθυρο εισαγωγής πλήθους επιτοκίων.



(β') Παράθυρο εισαγωγής τιμής επιτοκίου.

Σχήμα 4.9: Προσωρινά παράθυρα για την ανάθεση τιμών επιτοκίων.

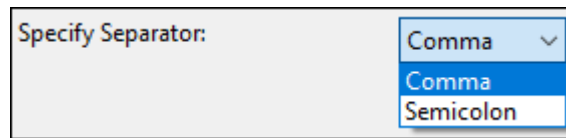
Με την επιλογή του "Run Sensitivity Analysis" γίνεται η κλήση της συνάρτησης **OnSensitivity**. Η λειτουργία της είναι παρόμοια με την συνάρτηση **OnRun**, με τις διαφορές ότι:

- Ορίζει το μέγεθος του αντικειμένου της κλάσης Opt ίσο με το πλήθος των διαφορετικών επιτοκίων, για παράδειγμα ίσο με 3, Σχήμα 4.9α',
- Εκτελεί τα βήματα 3 έως 6, Σχήμα 3.4, τόσες φορές όσες το πλήθος των διαφορετικών επιτοκίων,
- Ενεργοποιεί την επιλογή "Export SA", Σχήμα 4.5β',
- Ενεργοποιεί την επιλογή "Plot Sensitivity Analysis", Σχήμα 4.5γ',

4.3.3 Εξαγωγή Αποτελεσμάτων

Με τη λήξη του υπολογισμού των λύσεων, ο χρήστης μπορεί πλέον να προχωρήσει στην εξαγωγή των αποτελεσμάτων η οποία γίνεται με την παραγωγή ενός αρχείου τιμών διαχωρισμένων με κόμματα. Από την επιλογή "Specify Separator", Σχήμα 4.10, ο χρήστης μπορεί να επιλέξει αν ο διαχωρισμός θα γίνει:

1. Με κόμμα - Comma (,) ή
2. Με ελληνικό ερωτηματικό - Semicolon (;).



Σχήμα 4.10: Επιλογή διαχωρισμού τιμών.

ώστε η μορφή του αρχείου να είναι προσαρμοσμένη στο το σύστημα του χρήστη.

Έπειτα, για την δημιουργία διαγραμμάτων αρκεί να ο χρήστης:

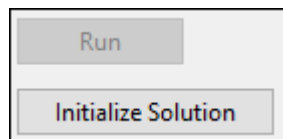
- Να επιλέξει τον πόρο που επιθυμεί να εξετάσει από την επιλογή "Plot graph for Resource", κλήση συνάρτησης **OnResourceSpin**,
- Να επιλέξει την λύση από την επιλογή "Plot graph for Solution", κλήση συνάρτησης **OnSolutionSpin**,
- Να καλέσει την συνάρτηση **OnResourceGraph** πατώντας το κουμπί "Plot", Σχήμα 4.4, και
- Αν ο αλγόριθμος έχει εκτελέσει ανάλυση ευαισθησίας, να επιλέξει την επιλογή "Plot Sensitivity Analysis", κλήση συνάρτησης **OnSense**.

4.3.4 Αρχικοποίηση GUI

Τέλος, εφόσον έχει λήξει η διαδικασία του υπολογισμού του προβλήματος, γίνεται διαθέσιμη η επιλογή "Initialize Solution", Σχήμα 4.11. Με το πάτημα του κουμπιού καλείται η συνάρτηση **OnSolInit** η οποία:

- Διαγράφει το αντικείμενο *final* της κλάσης **Opt**,
- Διαγράφει το αντικείμενο *element* της κλάσης **Rcpsp**,
- Διαγράφει τις τιμές των επιτοκίων για την ανάλυση ευαισθησίας,
- Αρχικοποιεί τις ράβδους προόδου,
- Καθιστά μη διαθέσιμες τις επιλογές:
 - "Plot",
 - "Initialize Solution"

- "Run Sensitivity Analysis"
 - "Plot Sensitivity Analysis",
 - "Export to CSV" και
 - "Export SA".
- Κάνει join το `backgroundThread` ώστε να μην υπάρξει συντριβή του προγράμματος.



Σχήμα 4.11: Το κουμπί "Initialize Solution" ενεργοποιημένο.

ΚΕΦΑΛΑΙΟ

— 5 —

ΥΠΟΛΟΓΙΣΤΙΚΗ ΕΜΠΕΙΡΙΑ - ΑΠΟΤΕΛΕΣΜΑΤΑ

5.1 Επιλογή Αρχείων

Για την εξαγωγή των αποτελεσμάτων επιλέχθηκαν 3 διαφορετικές κατηγορίες αρχείων από την επίσημη βιβλιοθήκη PSPLIB. Από αυτές, επιλέχθηκαν 3 υποκατηγορίες αρχείων με βάση την απαίτηση πόρων, δηλαδή:

1. Αρχεία 30 δραστηριοτήτων,
 - i. Χαμηλή απαίτηση πόρων,
 - ii. Μέτρια απαίτηση πόρων και
 - iii. Υψηλή απαίτηση πόρων.
2. Αρχεία 60 δραστηριοτήτων,
 - i. Χαμηλή απαίτηση πόρων,
 - ii. Μέτρια απαίτηση πόρων και
 - iii. Υψηλή απαίτηση πόρων.
3. Αρχεία 120 δραστηριοτήτων
 - i. Χαμηλή απαίτηση πόρων,
 - ii. Μέτρια απαίτηση πόρων και
 - iii. Υψηλή απαίτηση πόρων.

Για κάθε υποκατηγορία επιλέχθηκαν τρία αρχεία με το σύνολο τους να είναι 27 αρχεία. Για την καλύτερη παραγωγή αποτελεσμάτων, για κάθε αρχείο εκτελέστηκε ο αλγόριθμος από 3 φορές και επιλέχθηκε η καλύτερη λύση. Οι αρχικές παράμετροι που επιλέχθηκαν είναι:

1. $\text{initial} = 10$,
2. $\text{frg} = 300$ και
3. $\text{rate} = 0.05$ ή 5%.

Η διάταξη των αρχείων με τα αποτελέσματα έχουν την εξής ακολουθία ανά σειρά:

- Θέση του αρχείου στον υπολογιστή,
- Το μέγεθος του αρχικού πληθυσμού *initial*,
- Το μέγεθος της Τοπικής Έρευνας *frg*
- Οι συνολικές λύσεις (*loops*) που βρέθηκαν,
- Το ύψος του επιτοκίου,
- Ποιο μέλος του αρχικού πληθυσμού είναι η καλύτερη λύση,
- Ο χρόνος που χρειάστηκε για την επίλυση του προβλήματος,
- Ο χρόνος που χρειάστηκε για την διαδικασία της καθυστέρησης,
- Οι διάρκειες και οι *nrv* σε κάθε στάδιο του αλγορίθμου:
- Η λίστα προτεραιότητας,
- Ο χρόνος εκκίνησης των δραστηριοτήτων,
- Ο χρόνος λήξης των δραστηριοτήτων και
- Οι *nrv* των δραστηριοτήτων.
- Η απαίτηση πόρων για κάθε πόρο και
- Η χρονική στιγμή απαίτησης των πόρων.

Στο Σχήμα 5.1 παρουσιάζεται η μορφή του αρχείου αποτελεσμάτων με τις πληροφορίες που αναφέρθηκαν παραπάνω.

	A	B	C	D	E	F
1	C:\Users\jimar\Desktop\RCPSP\RCPSP\RCPSP\j120rcp\X1_2.RCP					
2	Initial:	10				
3	Frg:	750				
4	Loops:	111750				
5	Rate:	0.05				
6	Sol:	7				
7	prog_dur:	476817ms	476s			
8	del_dur:	1681ms	1s			
9		Duration	NPVCum			
10	Scout_b:	171	287.119			
11	Best_b:	136	383.171			
12	Elite_b4_Check:	156	383.171			
13	Final:	156	422.94			
14						
15	Act:	0	1	5	3	61
16	Start:	0	0	1	0	3
17	Finish:	0	1	5	3	5
18	NPV:	0	52.381	-39.1763	34.5535	54.8468
19						
20						
21	R1:	4	9	9	5	5
22	R2:	0	0	0	0	0
23	R3:	4	1	1	1	1
24	R4:	0	10	10	12	12
25	Time:	1	2	3	4	5

Σχήμα 5.1: Ενδεικτική μορφή αρχείου αποτελεσμάτων.

5.2 Σμήνος Μελισσών vs Simulated Annealing

Για τον καθορισμό της ποιότητας των λύσεων παρακάτω θα ακολουθήσει η σύγκριση στην παραγωγή των καλύτερων λύσεων της Τοπικής Έρευνας του αρχικού πληθυσμού

με την παραγωγή της λύσης του Simulated Annealing. Η σύγκριση γίνεται σε αυτό το σημείο καθώς, πρώτον, η διαδικασία της καθυστέρησης είναι η ίδια και για της δύο περιπτώσεις και δεύτερον στην παρούσα διπλωματική το κυριότερο σημείο που εξετάζει είναι η παραγωγή λύσεων με την μέθοδο συμπεριφοράς του σμήνους μελισσών και όχι η καθυστέρηση αυτή καθ'αυτή.

Για την σύγκριση χρησιμοποιήθηκε η εξής μεθοδολογία:

1. Τρέξε το αρχείο με Τοπική Έρευνα με αρχικό πληθυσμό $initial = 3$ και μέγεθος Τοπικής Έρευνας $frg = 600$.
2. Για όσες λύσεις παράξει ο αλγόριθμος, βλέπε Loops στο Σχήμα 5.1, θέσε $loops = Loops$ στον Simulated Annealing.
3. Τρέξε τον Simulated Annealing.
4. Τρέξε ξανά το αρχείο με Τοπική Έρευνα, αυτήν την φορά με αρχικό πληθυσμό $initial = 1$ και μέγεθος Τοπικής Έρευνας $frg = loops$.
5. Σύγκρινε τα τρία αποτελέσματα.

Ακολουθείται αυτή η μέθοδος ώστε με το βήμα 1 να παραχθεί μια αρχική λύση ως σημείο αναφοράς.

Με το βήμα 2 ορίζεται το μέγεθος του Simulated Annealing.

Ενώ με το βήμα 4 συγκρίνεται το ίδιο μέγεθος βελτίωσης και για τους δύο αλγόριθμους.

Για τη σύγκριση επιλέχθηκαν 4 αρχεία των 120 δραστηριοτήτων και 1 των 60, συνολικά 5 αρχεία. Παρακάτω παρουσιάζονται τα αποτελέσματα.

Πίνακας 5.1: Πίνακας Σύγκρισης Τοπικής Έρευνας με Simulated Annealing.

Αρχείο	Τύπος	NPV	Loops	Βελτίωση %
X16_4	S.A.	-632.86	12600	0.0
	Bees 3	-620.233	12600	2.04 %
	Bees 1	-581.724	63000	8.79 %
X3_6	S.A.	2894.32	9600	0.0
	Bees 3	2896.2	9600	0.06 %
	Bees 1	2925.67	67200	1.07 %
X60_10	S.A.	-2291.36	7800	0.0
	Bees 3	-2302.99	7800	-0.50 %
	Bees 1	-2226.54	23400	2.91 %
X60_8	S.A.	-2686.98	7200	0.0
	Bees 3	-2665.53	7200	0.80 %
	Bees 1	-2659.77	28800	1.02 %
J6014_4	S.A.	-1314.2	8400	0.0
	Bees 3	-1286.67	8400	2.14 %
	Bees 1	-1286.52	33600	2.15 %

Όπου:

- S.A. ο αλγόριθμος Simulated Annealing,
- Bees 3, η Τοπική Έρευνα με μέγεθος $inital = 3$ και
- Bees 1, η Τοπική Έρευνα με μέγεθος $inital = 1$.

Δοκιμάστηκε και η προσέγγιση του S.A. με $Temperature = 20,000$ και συντελεστή μείωσης 0.99, ωστόσο τα αποτελέσματα δεν παρουσίασαν βελτίωση συγκριτικά με τις αρχικές παραμέτρους. Ανεξαρτήτως της αρχικής θερμοκρασίας ο S.A. συγκλίνει και εγκλωβίζεται σε ένα τοπικό βέλτιστο.

5.3 Συμπεριφορά Αλγορίθμου

5.3.1 Γενική Μελέτη

Στα αρχεία που αναφέρθηκαν στην αρχή του κεφαλαίου εφαρμόστηκαν τέσσερις διαφορετικές προσεγγίσεις για την εξαγωγή των αποτελεσμάτων:

1. Χαμηλή εντατικοποίηση μονής συνθήκης (καλύτερης *npu*),
2. Μέτρια εντατικοποίηση μονής συνθήκης,
3. Υψηλή εντατικοποίηση μονής συνθήκης και
4. Υψηλή εντατικοποίηση διπλής συνθήκης (καλύτερης *npu* και καλύτερης ή ίσης διάρκειας).

Στους παρακάτω πίνακες παρουσιάζονται τα αρχεία με τα καλύτερα αποτελέσματα θα χρωματιστούν με **πράσινο φόντο** ενώ τα αρχεία που παρουσιάζουν ουδετερότητα και εμφανίζουν το ίδιο αποτέλεσμα ανεξαρτήτως προσεγγίσεως με **πορτοκαλί φόντο**.

Πίνακας 5.2: Αποτελέσματα αρχείων χαμηλής απαίτησης πόρων και χαμηλής Εντατικοποίησης.

File Name	Init. Dur.	Delay	Duration	NPV	Time	Avg. Time	Loops
30 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J301_2	47	54	54	133.38	17.7		15,300
J305_10	105	120	120	-9.27	23.75	22.82	19,500
J3041_3	117	134	134	-77.04	26.95		22,200
60 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J601_2	87	100	100	163.50	51.52		24,000
J6033_6	98	112	112	381.53	43.77	47.55	19,500
J6038_5	138	158	122	-327.29	48.35		19,500
120 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
X1_2	140	161	161	403.21	180.00		39,300
X3_6	113	129	129	642.06	129.85	215.90	29,700
X16_4	284	326	250	-1041.3	337.84		70,800

Πίνακας 5.3: Αποτελέσματα αρχείων μέτριας απαίτησης πόρων και χαμηλής Εντατικοποίησης.

File Name	Init. Dur.	Delay	Duration	NPV	Time	Avg. Time	Loops
30 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J3043_2	51	58	48	-1402.98	14.24		11,400
J3043_10	75	86	70	-773.49	11.30	11.01	9,900
J3044_4	57	65	57	-1275.21	7.36		6,000
60 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J6014_4	100	115	79	-1778.67	32.63		15,000
J6023_5	86	98	98	-27.95	34.46	39.25	14,400
J6030_5	101	116	100	-1189.02	50.45		23,400
120 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
X54_4	192	220	157	-960.65	214.06		38,100
X54_1	163	187	135	-1653.67	220.89	199.21	37,200
X55_1	166	190	140	-1073.88	162.66		27,600

Πίνακας 5.4: Αποτελέσματα αρχείων υψηλής απαίτησης πόρων και χαμηλής Εντατικοποίησης.

File Name	Init. Dur.	Delay	Duration	NPV	Time	Avg. Time	Loops
30 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J3048_1	63	72	63	-1757.13	8.06		6,000
J3048_3	50	57	55	-1682.08	6.49	7.13	6,000
J3048_10	54	62	54	-1673.50	6.84		6,000
60 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J6016_5	66	75	66	-2396.41	16.53		7,500
J6016_10	68	78	75	-1689.48	16.56	15.87	7,500
J6048_2	87	100	87	-1969.60	14.51		6,900
120 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
X60_8	150	172	125	-2264.34	188.63		30,600
X60_9	148	170	112	-2073.57	193.31	188.28	31,200
X60_10	127	146	109	-2355.35	182.88		28,500

Πίνακας 5.5: Αποτελέσματα αρχείων χαμηλής απαίτησης πόρων και μέτριας Εντατικοποίησης.

File Name	Init. Dur.	Delay	Duration	NPV	Time	Avg. Time	Loops
30 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J301_2	48	55	55	142.76	15.07		14,400
J305_10	113	129	129	-1.54	23.91	19.79	16,500
J3041_3	117	134	134	-77.04	20.39		18,900
60 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J601_2	87	100	100	159.72	44.14		16,800
J6033_6	98	112	112	382.02	36.08	39.52	15,300
J6038_5	139	159	103	-462.67	38.33		16,500
120 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
X1_2	154	177	177	399.40	125.71		18,300
X3_6	108	124	124	639.43	83.07	114.59	18,900
X16_4	268	308	268	-769.84	134.97		27,300

Πίνακας 5.6: Αποτελέσματα αρχείων μέτριας απαίτησης πόρων και μέτριας Εντατικοποίησης.

File Name	Init. Dur.	Delay	Duration	NPV	Time	Avg. Time	Loops
30 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J3043_2	51	58	49	−1390.55	11.55	10.11	10,800
J3043_10	75	86	67	−803.13	12.22		9,600
J3044_4	57	65	57	−1275.21	6.55		5,400
60 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J6014_4	97	111	90	−1305.76	24.90	28.69	11,100
J6023_5	86	98	98	−53.07	26.88		12,600
J6030_5	99	113	97	−1236.81	34.29		15,300
120 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
X54_4	172	197	148	−936.42	176.57	141.75	24,600
X54_1	155	178	132	−1649.60	133.66		20,400
X55_1	163	187	140	−1121.64	115.00		20,100

Πίνακας 5.7: Αποτελέσματα αρχείων υψηλής απαίτησης πόρων και μέτριας Εντατικοποίησης.

File Name	Init. Dur.	Delay	Duration	NPV	Time	Avg. Time	Loops
30 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J3048_1	63	72	63	−1757.13	6.97	7.13	6,300
J3048_3	50	57	55	−1682.08	6.90		6,000
J3048_10	54	62	54	−1673.50	7.80		6,900
60 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J6016_5	66	75	66	−2396.41	16.78	15.57	7,500
J6016_10	68	78	74	−1708.97	17.50		7,500
J6048_2	87	100	87	−1969.60	12.89		6,300
120 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
X60_8	145	166	124	−2761.24	103.93	98.97	18,300
X60_9	138	158	115	−1824.12	90.40		16,500
X60_10	132	151	114	−1946.18	102.56		18,900

Πίνακας 5.8: Αποτελέσματα αρχείων χαμηλής απαίτησης πόρων και υψηλής Εντατικοποίησης.

File Name	Init. Dur.	Delay	Duration	NPV	Time	Avg. Time	Loops
30 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J301_2	55	63	63	164.55	13.27		12,600
J305_10	108	124	124	-9.57	15.31	15.99	12,600
J3041_3	117	134	134	-77.04	19.39		16,200
60 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J601_2	87	100	100	162.20	32.61		15,300
J6033_6	96	110	110	382.09	43.28	36.88	14,400
J6038_5	136	156	108	-393.35	34.73		14,400
120 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
X1_2	155	178	178	408.31	70.00		15,600
X3_6	126	144	144	640.298	48.80	79.94	10,800
X16_4	265	304	268	-739.40	121.02		20,100

Πίνακας 5.9: Αποτελέσματα αρχείων μέτριας απαίτησης πόρων και υψηλής Εντατικοποίησης.

File Name	Init. Dur.	Delay	Duration	NPV	Time	Avg. Time	Loops
30 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J3043_2	57	65	45	-1593.07	7.98		7,800
J3043_10	75	86	70	-779.35	11.24	8.69	9,000
J3044_4	57	65	59	-1222.41	6.84		6,000
60 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J6014_4	100	115	86	-1494.58	24.50		11,100
J6023_5	88	101	101	-25.96	22.88	26.10	10,200
J6030_5	110	126	103	-1153.96	30.91		13,500
120 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
X54_4	189	217	156	-981.53	153.64		21,300
X54_1	153	175	127	-1840.28	92.76	122.72	15,900
X55_1	163	187	136	-1076.72	121.73		19,200

Πίνακας 5.10: Αποτελέσματα αρχείων υψηλής απαίτησης πόρων και υψηλής Εντατικοποίησης.

File Name	Init. Dur.	Delay	Duration	NPV	Time	Avg. Time	Loops
30 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J3048_1	63	72	63	-1757.13	7.00		6,000
J3048_3	50	57	55	-1682.08	6.67	6.83	6,300
J3048_10	54	62	54	-1673.50	6.80		6,300
60 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J6016_5	66	75	66	-2396.41	16.78		7,200
J6016_10	68	78	77	-1587.57	17.43	16.82	7,500
J6048_2	87	100	87	-1969.60	14.12		6,600
120 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
X60_8	135	155	120	-2527.50	88.74		15,300
X60_9	130	149	119	-1627.07	102.42	88.00	18,600
X60_10	127	146	124	-1490.46	72.81		13,200

Πίνακας 5.11: Αποτελέσματα αρχείων χαμηλής απαίτησης πόρων, υψηλής Εντατικοποίησης και διπλής συνθήκης.

File Name	Init. Dur.	Delay	Duration	NPV	Time	Avg. Time	Loops
30 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J301_2	54	62	62	161.76	13.43		12,000
J305_10	101	116	116	-26.67	12.62	13.12	10,500
J3041_3	112	128	128	-104.227	13.28		10,500
60 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J601_2	88	101	101	156.20	26.65		10,800
J6033_6	82	94	94	376.84	21.28	24.70	9,000
J6038_5	108	124	111	-345.88	26.27		10,800
120 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
X1_2	147	169	169	384.51	45.68		9,600
X3_6	116	133	133	636.87	46.47	53.17	10,200
X16_4	252	289	264	-855.87	67.34		12,600

Πίνακας 5.12: Αποτελέσματα αρχείων μέτριας απαίτησης πόρων, υψηλής Εντατικοποίησης και διπλής συνθήκης.

File Name	Init. Dur.	Delay	Duration	NPV	Time	Avg. Time	Loops
30 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J3043_2	57	65	51	−1329.95	10.15		9,600
J3043_10	65	74	72	−745.82	10.95	9.28	9,900
J3044_4	57	65	59	−1205.18	6.73		6,000
60 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J6014_4	85	97	97	−1096.30	19.84		8,700
J6023_5	85	97	97	−38.70	20.78	21.82	9,300
J6030_5	88	101	101	−1194.88	24.82		9,600
120 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
X54_4	154	177	149	−962.692	82.52		13,500
X54_1	133	152	134	−1588.81	76.64	81.08	12,900
X55_1	142	163	138	−990.18	84.08		10,500

Πίνακας 5.13: Αποτελέσματα αρχείων υψηλής απαίτησης πόρων, υψηλής Εντατικοποίησης και διπλής συνθήκης.

File Name	Init. Dur.	Delay	Duration	NPV	Time	Avg. Time	Loops
30 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J3048_1	63	72	63	−1757.13	6.49		6,000
J3048_3	50	57	55	−1682.08	6.41	6.65	6,000
J3048_10	54	62	54	−1673.50	7.07		6,000
60 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
J6016_5	66	75	66	−2396.41	15.59		7,200
J6016_10	68	78	78	−1546.06	16.40	15.78	7,200
J6048_2	87	100	87	−1969.60	15.34		7,200
120 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ							
X60_8	128	147	123	−2317.32	74.91		12,900
X60_9	118	135	116	−1890.66	63.66	65.51	9,900
X60_10	108	124	121	−1693.45	57.95		10,200

5.3.2 Περιπτώσεις Εστιασμένης Μελέτης

Από τα παραπάνω αρχεία επιλέχθηκαν δύο συγκεκριμένα, τα αρχεία X16_4 και X54_4, εκ των οποίων οι λύσεις είναι ιδιαίτερα μεγάλες σε χρονική διάρκεια συγκριτικά με τα υπόλοιπα ώστε να γίνει μια πιο εστιασμένη μελέτη του αλγορίθμου. Συγκεκριμένα σε κάθε αρχείο εφαρμόστηκαν οι εξής αλλαγές:

1. Μέγεθος αρχικού πληθυσμού, $initial = 10$,
2. Μέγεθος Τοπικής Έρευνας, $frg = 1000$,
3. Αλλαγή στην τιμή του constant για την δημιουργία αρνητικών χρηματοροών στις περισσότερες δραστηριότητες και
4. Αλλαγή στην τιμή του constant για την δημιουργία θετικών χρηματοροών στις περισσότερες δραστηριότητες αλλά και την ύπαρξη αρνητικών ώστε να γίνει καθυστέρηση.

Για κάθε μία από τις παραπάνω περιπτώσεις στην αλλαγή των χρηματοροών εφαρμόστηκαν οι παρακάτω προσεγγίσεις, όλες με υψηλή εντατικοποίηση:

1. Εφαρμογή μονής συνθήκης (καλύτερη nrv),
2. Εφαρμογή διπλής συνθήκης (καλύτερη nrv και καλύτερη ή ίση διάρκεια) και
3. Εφαρμογή νέου αντίστροφου κριτηρίου (καλύτερη nrv και χειρότερη διάρκεια).

Για την αποδοτικότερη προσέγγιση θα παρουσιαστούν στην συνέχεια τα διαγράμματα χρήσης πόρων όπου στον οριζόντιο άξονα προσδιορίζεται η χρονική στιγμή και στον κάθετο η κατανάλωση του πόρου την δεδομένη χρονική στιγμή.

Για κάθε διάγραμμα που θα ακολουθήσει παρακάτω οι μέγιστες τιμές των αξόνων είναι η διάρκεια του έργου και το όριο διαθεσιμότητας του πόρου για τον οριζόντιο και κάθετο άξονα αντίστοιχα. Τα επιλεγμένα αρχεία αφορούν προβλήματα με τέσσερις (4) διαφορετικούς ανανεώσιμους πόρους εκ των οποίων η ποσοστιαία συνολική χρήση (utilization) καθ' όλη την διάρκεια του χρονοπρογράμματος θα αναγράφεται μετά το διάγραμμα του κάθε πόρου.

Το Αρχείο X16_4

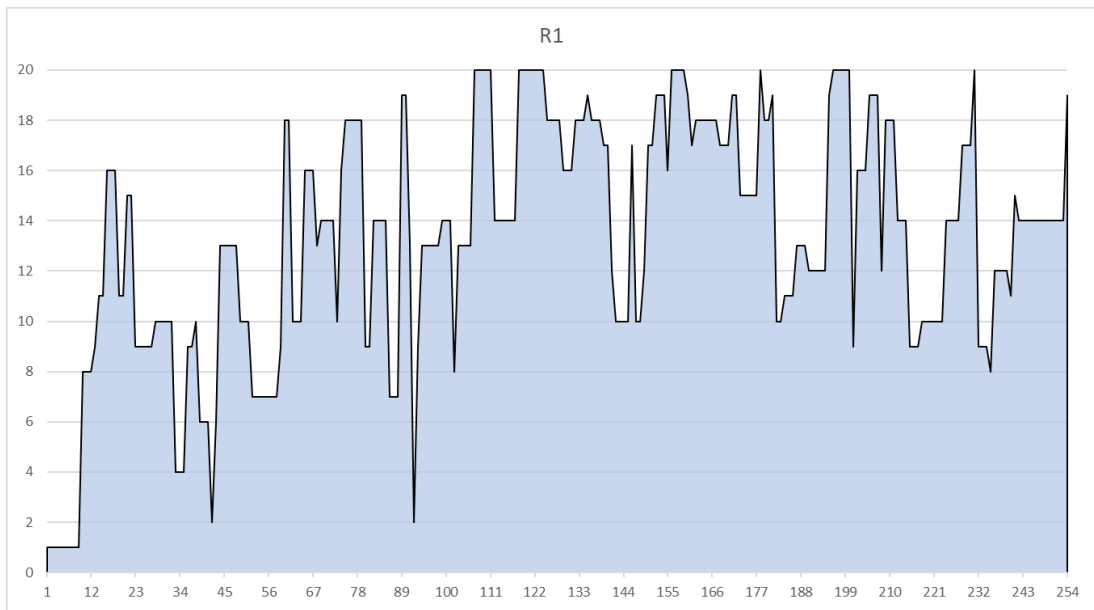
Σε αυτό το αρχείο η τιμή του constant και το πλήθος των αρνητικών χρηματοροών είναι ως εξής:

1. constant = 100.0 και πλήθος αρνητικών χρηματοροών 120,
2. constant = 350.0 και πλήθος αρνητικών χρηματοροών 42 και
3. constant = 700.0 και πλήθος αρνητικών χρηματοροών 0.

Πίνακας 5.14: Αποτελέσματα αρχείου X16_4 με constant = 100.0.

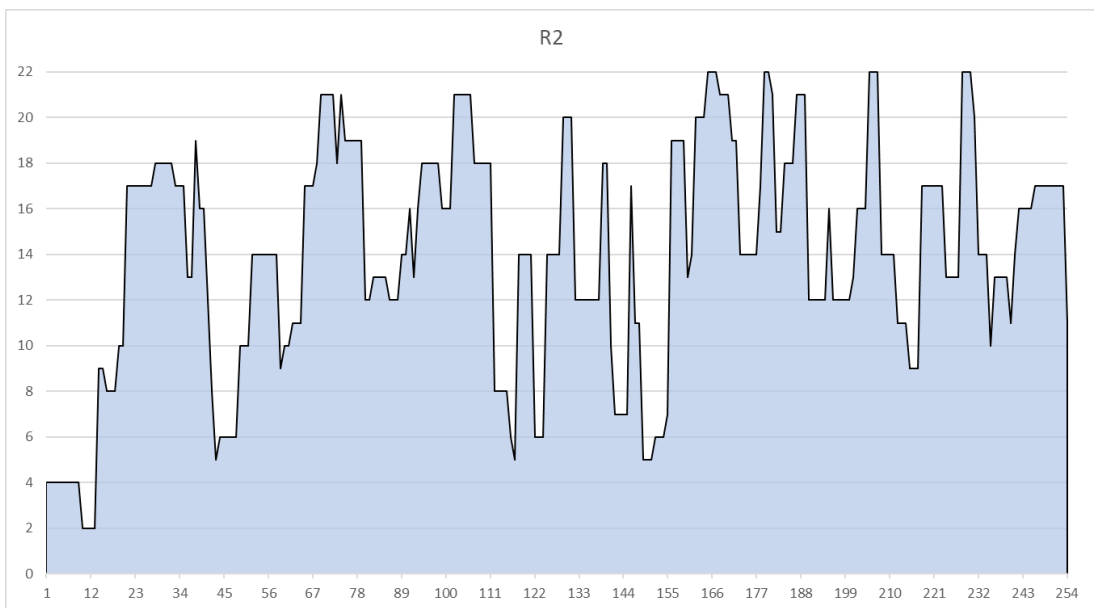
Init. Dur.	Delay	Init. NPV	Duration	NPV	NPV Opt.	Duration Opt.	Time
ΜΟΝΗ ΣΥΝΘΗΚΗ							
263	302	-1465.69	253	-1041.26	29%	4%	323
ΔΙΠΛΗ ΣΥΝΘΗΚΗ							
264	303	-1494.45	262	-980.089	34%	1%	241
ΔΙΠΛΗ ΑΝΤΙΣΤΡΟΦΗ ΣΥΝΘΗΚΗ							
287	330	-1482.61	254	-955.717	36%	11%	244

Παρακάτω παρουσιάζονται τα διαγράμματα χρήσης πόρων για το αρχείο X16_4 με την εφαρμογή της διπλής αντίστροφης συνθήκης:



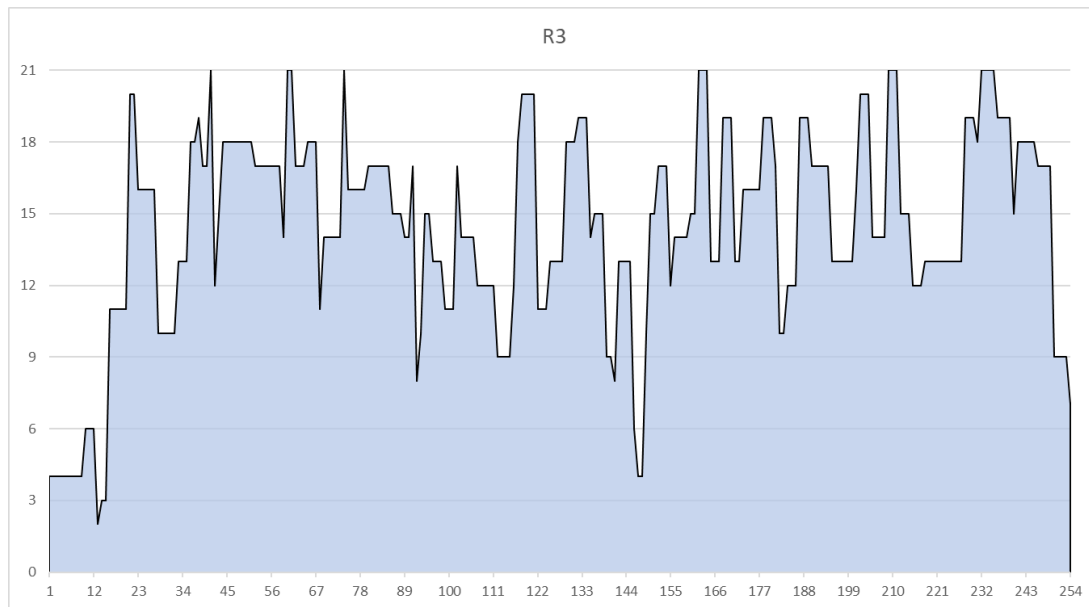
Σχήμα 5.2: Διάγραμμα χρήσης πόρου R1 αρχείου X16_4 με constant = 100.0.

Χρήση πόρου R1, Σχήμα 5.2, utilization = 66.54%.



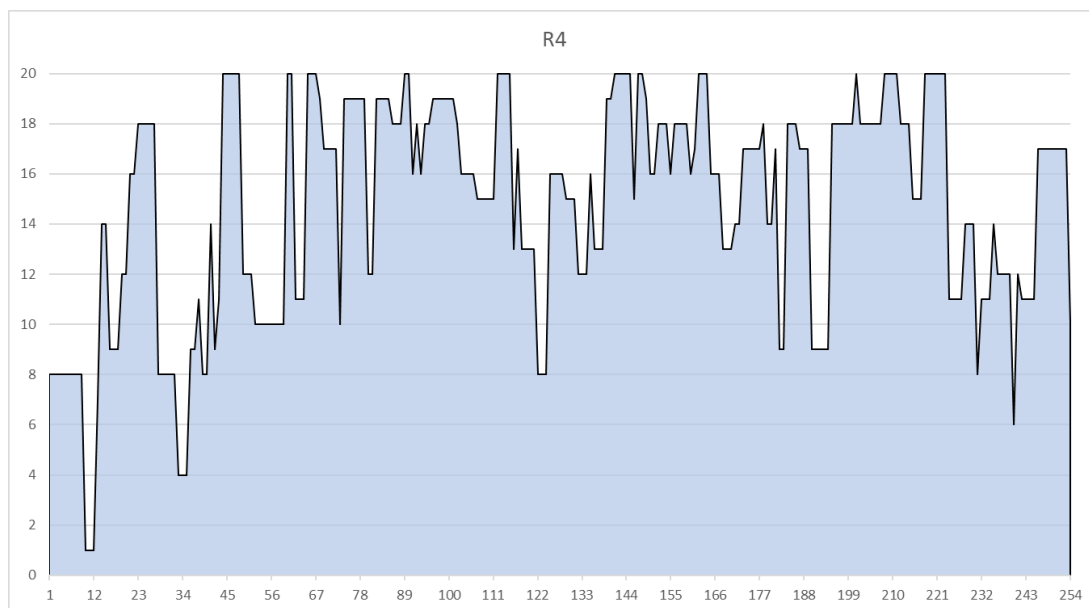
Σχήμα 5.3: Διάγραμμα χρήσης πόρου R2 αρχείου X16_4 με constant = 100.0.

Χρήση πόρου R2, Σχήμα 5.3, utilization = 63.55%.



Σχήμα 5.4: Διάγραμμα χρήσης πόρου R3 αρχείου X16_4 με constant = 100.0.

Χρήση πόρου R3, Σχήμα 5.4, utilization = 68.77%.



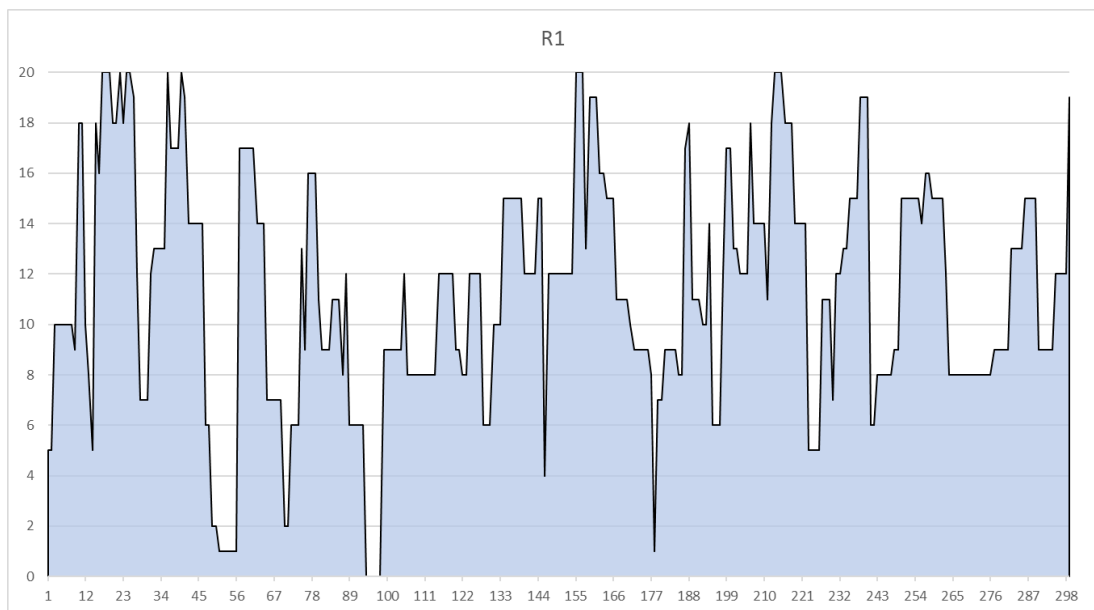
Σχήμα 5.5: Διάγραμμα χρήσης πόρου R4 αρχείου X16_4 με constant = 100.0.

Χρήση πόρου R4, Σχήμα 5.5, utilization = 73.88%.

Πίνακας 5.15: Αποτελέσματα αρχείου X16_4 με constant = 350.0.

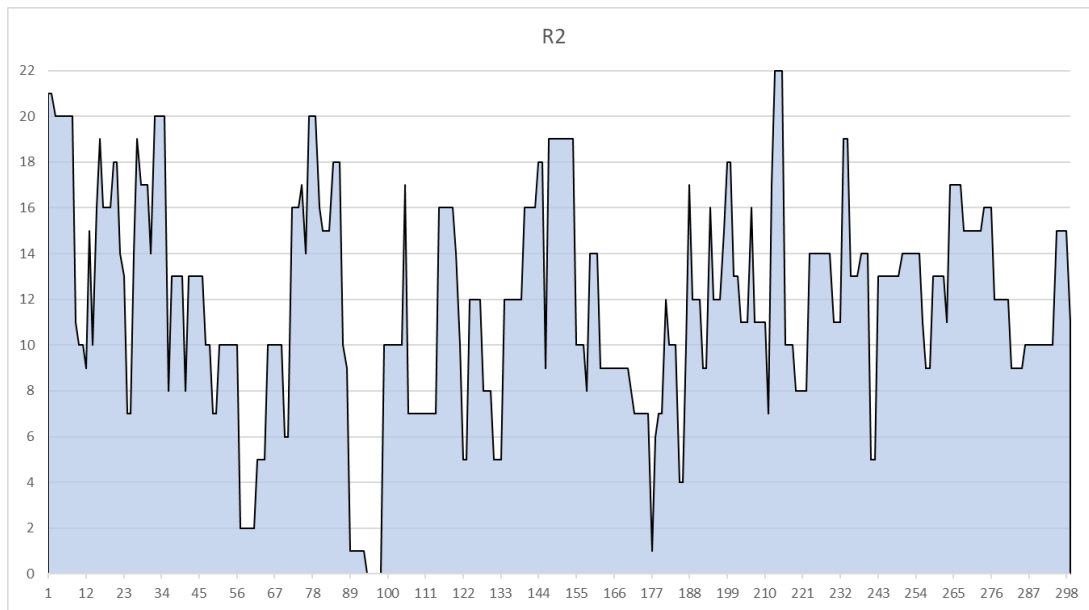
Init. Dur.	Delay	Init. NPV	Duration	NPV	NPV Opt.	Duration Opt.	Time
ΜΟΝΗ ΣΥΝΘΗΚΗ							
260	299	1030.53	299	1040.32	1%	-15%	216
ΔΙΠΛΗ ΣΥΝΘΗΚΗ							
257	295	951.15	295	982.408	3%	-15%	209
ΔΙΠΛΗ ΑΝΤΙΣΤΡΟΦΗ ΣΥΝΘΗΚΗ							
277	318	942.757	318	1000.77	6%	-15%	199

Παρακάτω παρουσιάζονται τα διαγράμματα χρήσης πόρων για το αρχείο X16_4 με την εφαρμογή της μονής συνθήκης:



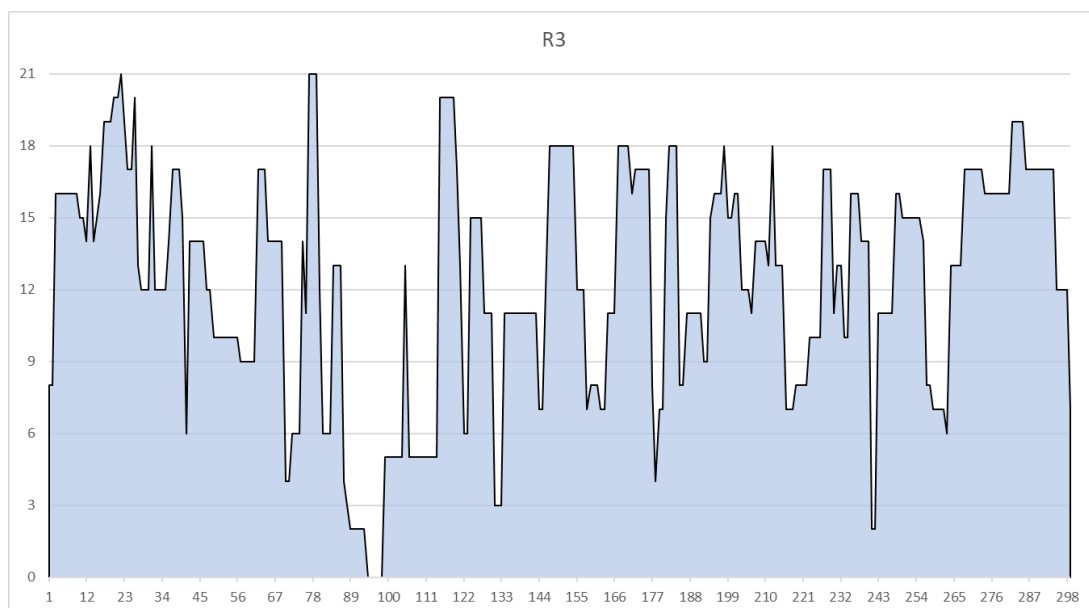
Σχήμα 5.6: Διάγραμμα χρήσης πόρου R1 αρχείου X16_4 με constant = 350.0.

Χρήση πόρου R1, Σχήμα 5.6, utilization = 56.52%.



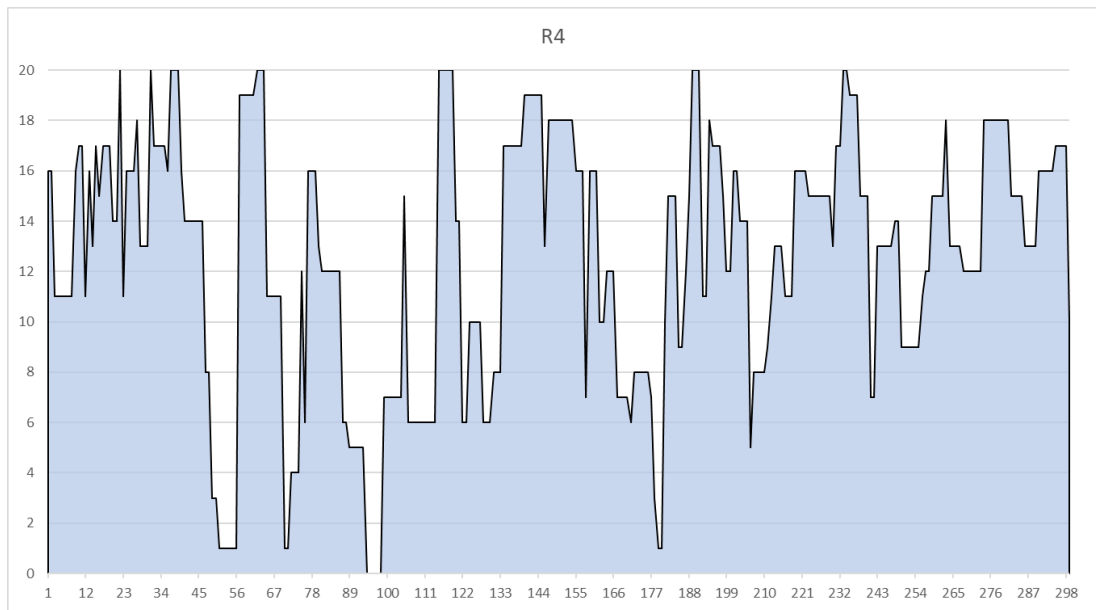
Σχήμα 5.7: Διάγραμμα χρήσης πόρου R2 αρχείου X16_4 με constant = 350.0.

Χρήση πόρου R2, Σχήμα 5.7, utilization = 53.98%.



Σχήμα 5.8: Διάγραμμα χρήσης πόρου R3 αρχείου X16_4 με constant = 350.0.

Χρήση πόρου R3, Σχήμα 5.8, utilization = 58.42%.



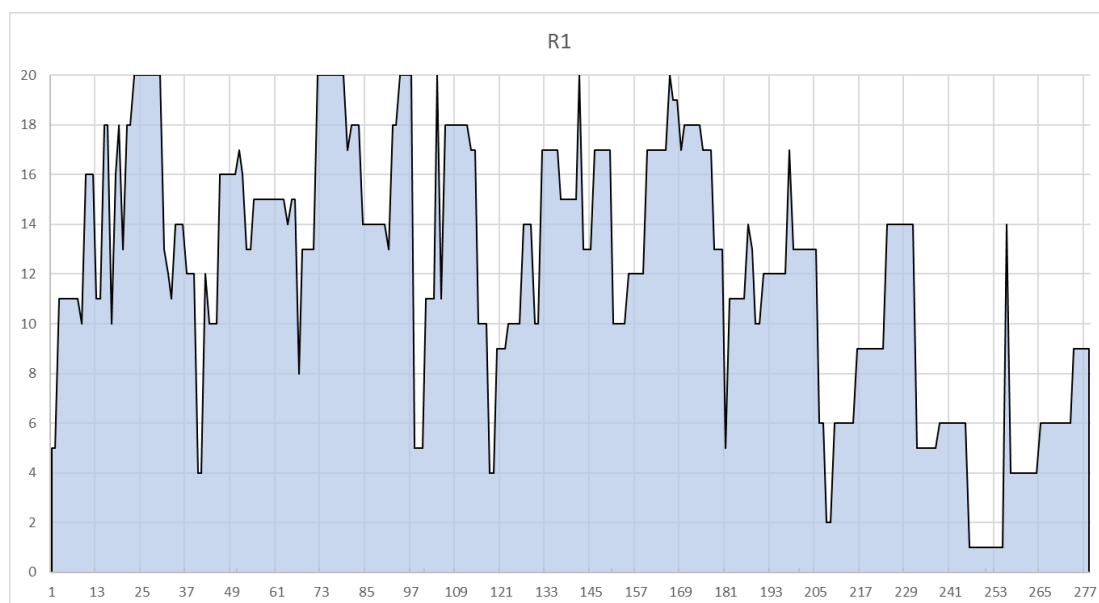
Σχήμα 5.9: Διάγραμμα χρήσης πόρου R4 αρχείου X16_4 με constant = 350.0.

Χρήση πόρου R4, Σχήμα 5.9, utilization = 62.76%.

Πίνακας 5.16: Αποτελέσματα αρχείου X16_4 με constant = 700.0.

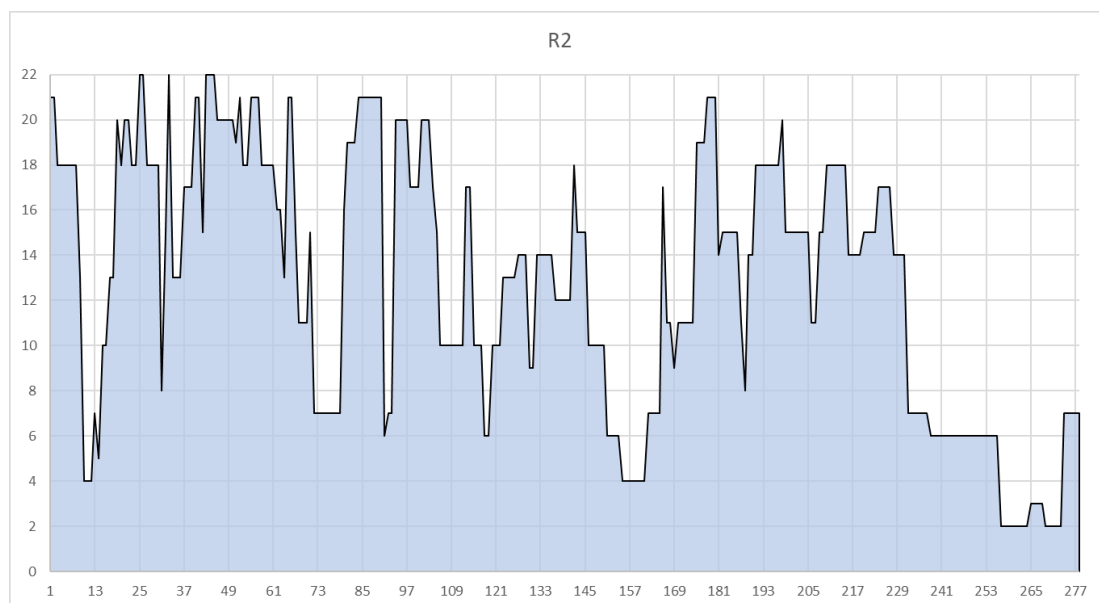
Init. Dur.	Delay	Init. NPV	Duration	NPV	NPV Opt.	Duration Opt.	Time
ΜΟΝΗ ΣΥΝΘΗΚΗ							
278	319	5388	278	5388	0%	0%	264
ΔΙΠΛΗ ΣΥΝΘΗΚΗ							
253	290	5223.94	253	5223.94	0%	0%	152
ΔΙΠΛΗ ΑΝΤΙΣΤΡΟΦΗ ΣΥΝΘΗΚΗ							
280	322	5228.47	280	5228.47	0%	0%	196

Παρακάτω παρουσιάζονται τα διαγράμματα χρήσης πόρων για το αρχείο X16_4 με την εφαρμογή της μονής συνθήκης:



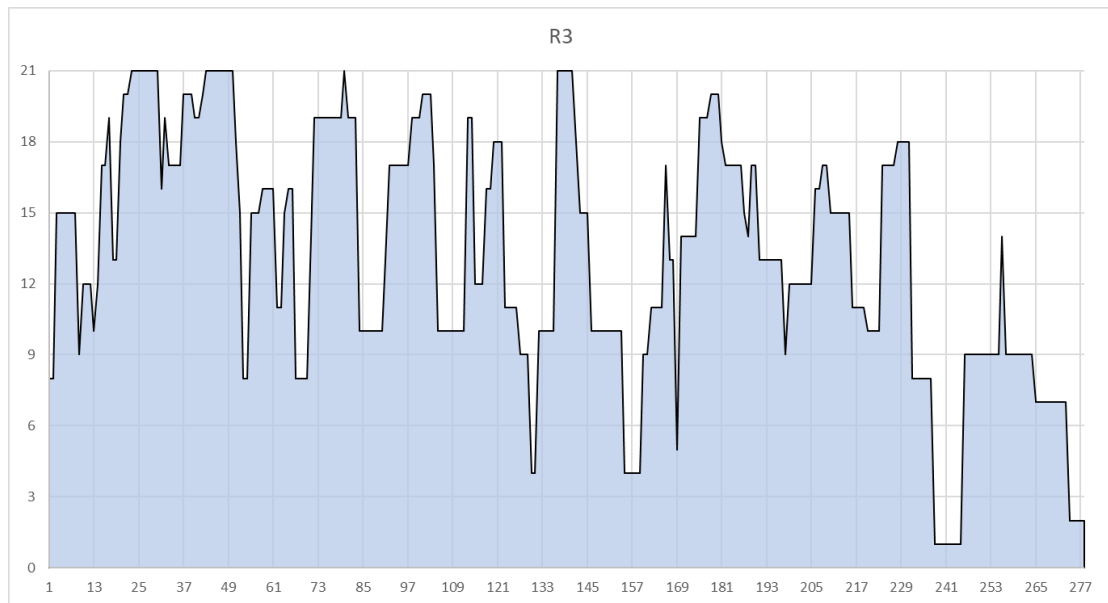
Σχήμα 5.10: Διάγραμμα χρήσης πόρου R1 αρχείου X16_4 με constant = 700.0.

Χρήση πόρου R1, Σχήμα 5.10, utilization = 60.79%.



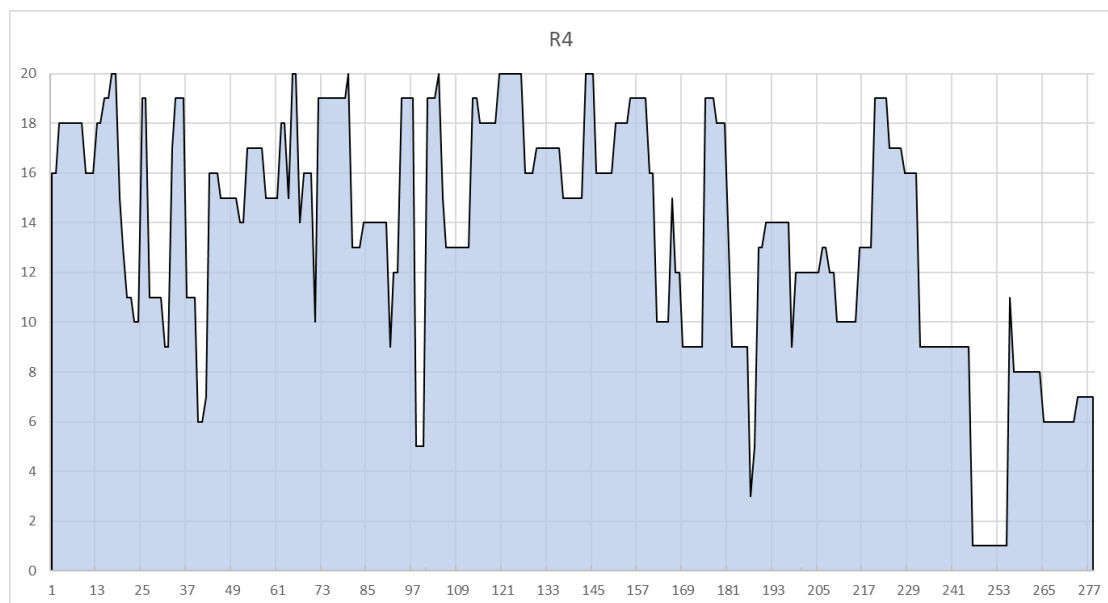
Σχήμα 5.11: Διάγραμμα χρήσης πόρου R2 αρχείου X16_4 με constant = 700.0.

Χρήση πόρου R2, Σχήμα 5.11, utilization = 58.06%.



Σχήμα 5.12: Διάγραμμα χρήσης πόρου R3 αρχείου X16_4 με constant = 700.0.

Χρήση πόρου R3, Σχήμα 5.12, utilization = 62.83%.



Σχήμα 5.13: Διάγραμμα χρήσης πόρου R4 αρχείου X16_4 με constant = 700.0.

Χρήση πόρου R4, Σχήμα 5.13, utilization = 67.50%.

Το Αρχείο X54_4

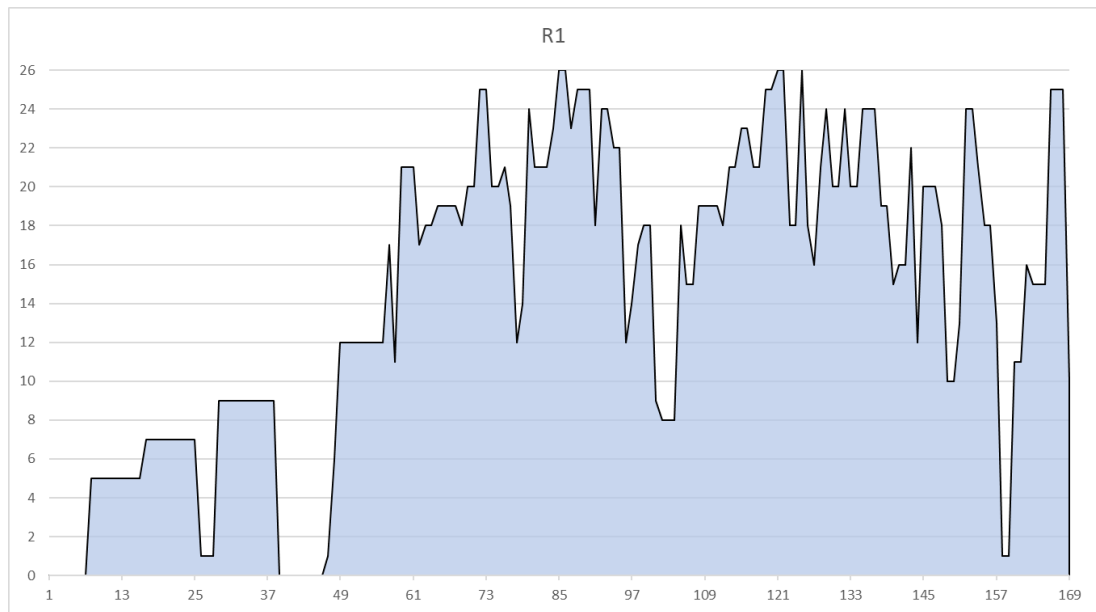
Σε αυτό το αρχείο η τιμή του constant και το πλήθος των αρνητικών χρηματοροών είναι ως εξής:

1. constant = 20.0 και πλήθος αρνητικών χρηματοροών 119,
2. constant = 300.0 και πλήθος αρνητικών χρηματοροών 41 και
3. constant = 700.0 και πλήθος αρνητικών χρηματοροών 0.

Πίνακας 5.17: Αποτελέσματα αρχείου X54_4 με constant = 20.0.

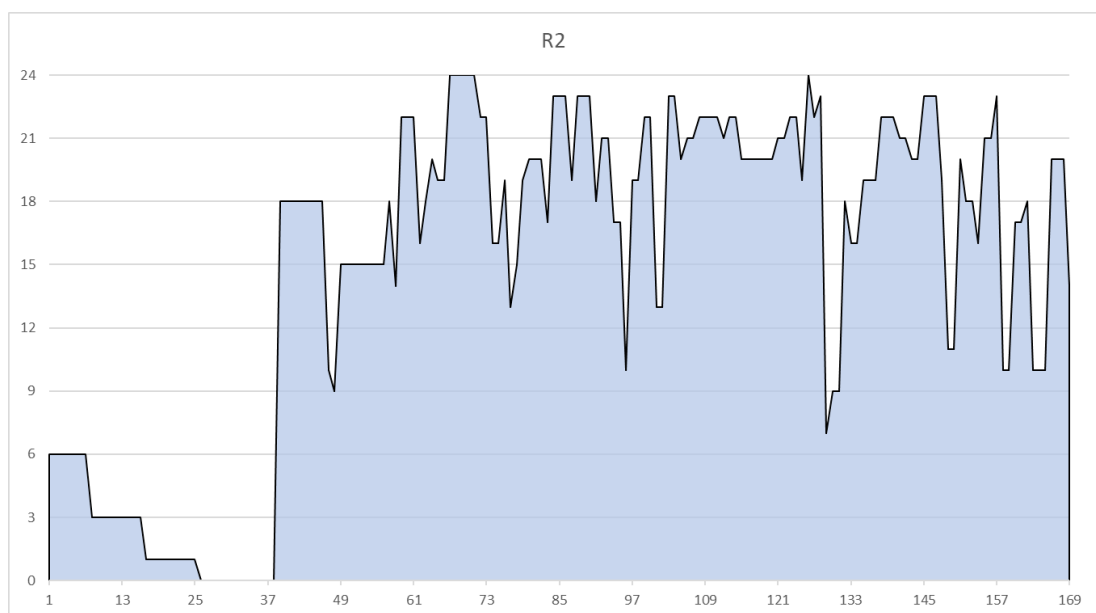
Init. Dur.	Delay	Init. NPV	Duration	NPV	NPV Opt.	Duration Opt.	Time
ΜΟΝΗ ΣΥΝΘΗΚΗ							
209	240	-2532.74	144	-1808.06	29%	31%	368
ΔΙΠΛΗ ΣΥΝΘΗΚΗ							
151	173	-2669.25	171	-812.357	70%	-13%	257
ΔΙΠΛΗ ΑΝΤΙΣΤΡΟΦΗ ΣΥΝΘΗΚΗ							
187	215	-2642.8	169	-779.127	71%	10%	308

Παρακάτω παρουσιάζονται τα διαγράμματα χρήσης πόρων για το αρχείο X54_4 με την εφαρμογή της διπλής αντίστροφης συνθήκης:



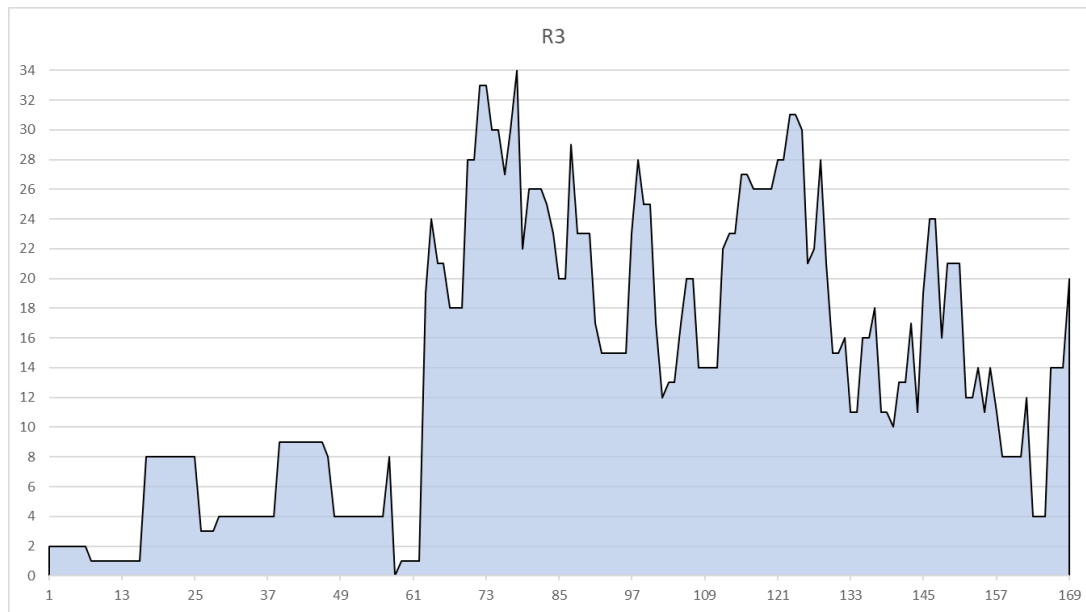
Σχήμα 5.14: Διάγραμμα χρήσης πόρου R1 αρχείου X54_4 με $\text{constant} = 20.0$.

Χρήση πόρου R1, Σχήμα 5.14, $\text{utilization} = 55.28\%$.



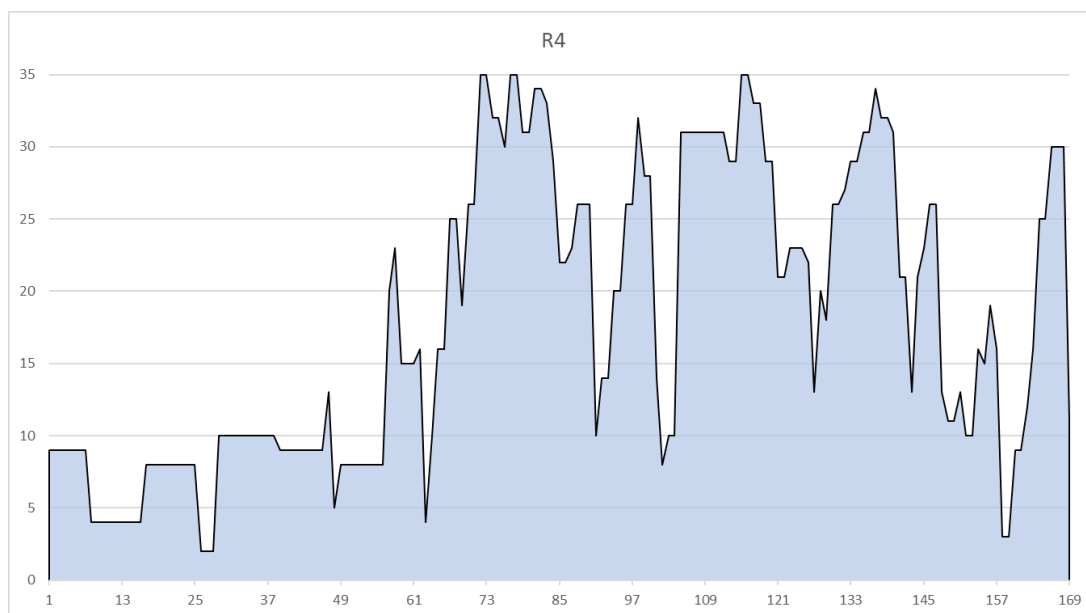
Σχήμα 5.15: Διάγραμμα χρήσης πόρου R2 αρχείου X54_4 με $\text{constant} = 20.0$.

Χρήση πόρου R2, Σχήμα 5.15, $\text{utilization} = 62.06\%$.



Σχήμα 5.16: Διάγραμμα χρήσης πόρου R3 αρχείου X54_4 με $\text{constant} = 20.0$.

Χρήση πόρου R3, Σχήμα 5.16, $\text{utilization} = 40.71\%$.



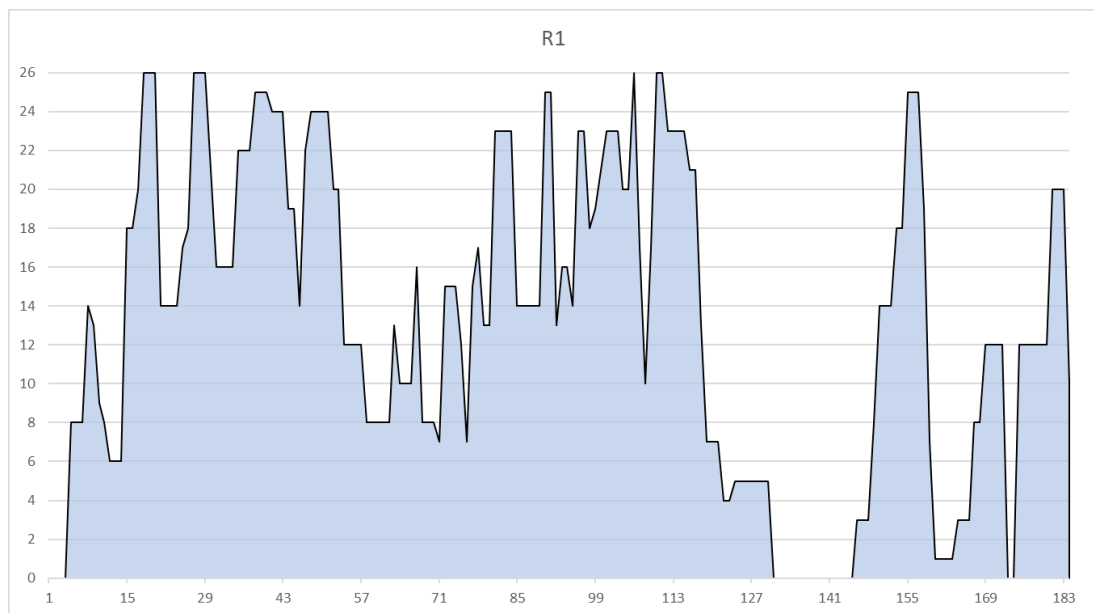
Σχήμα 5.17: Διάγραμμα χρήσης πόρου R4 αρχείου X54_4 με $\text{constant} = 20.0$.

Χρήση πόρου R4, Σχήμα 5.17, $\text{utilization} = 51.28\%$.

Πίνακας 5.18: Αποτελέσματα αρχείου X54_4 με constant = 300.0.

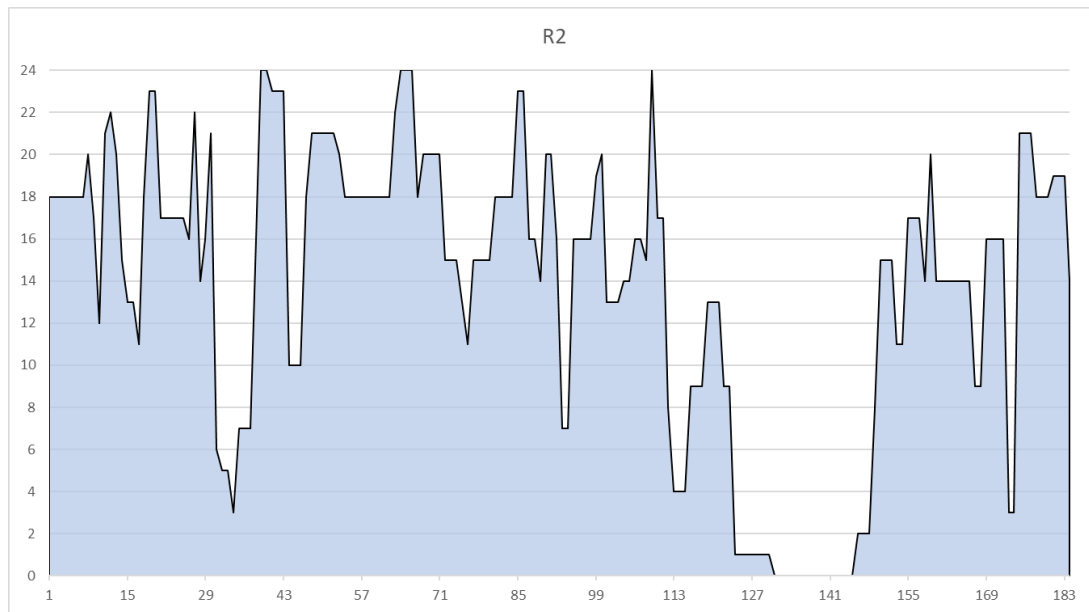
Init. Dur.	Delay	Init. NPV	Duration	NPV	NPV Opt.	Duration Opt.	Time
ΜΟΝΗ ΣΥΝΘΗΚΗ							
160	184	1268.11	184	1346.82	6%	-13%	345
ΔΙΠΛΗ ΣΥΝΘΗΚΗ							
143	164	1244.24	164	1280.8	3%	-13%	238
ΔΙΠΛΗ ΑΝΤΙΣΤΡΟΦΗ ΣΥΝΘΗΚΗ							
180	207	1273.84	207	1313.81	3%	-13%	247

Παρακάτω παρουσιάζονται τα διαγράμματα χρήσης πόρων για το αρχείο X54_4 με την εφαρμογή της μονής συνθήκης:



Σχήμα 5.18: Διάγραμμα χρήσης πόρου R1 αρχείου X54_4 με constant = 300.0.

Χρήση πόρου R1, Σχήμα 5.18, utilization = 50.77%.



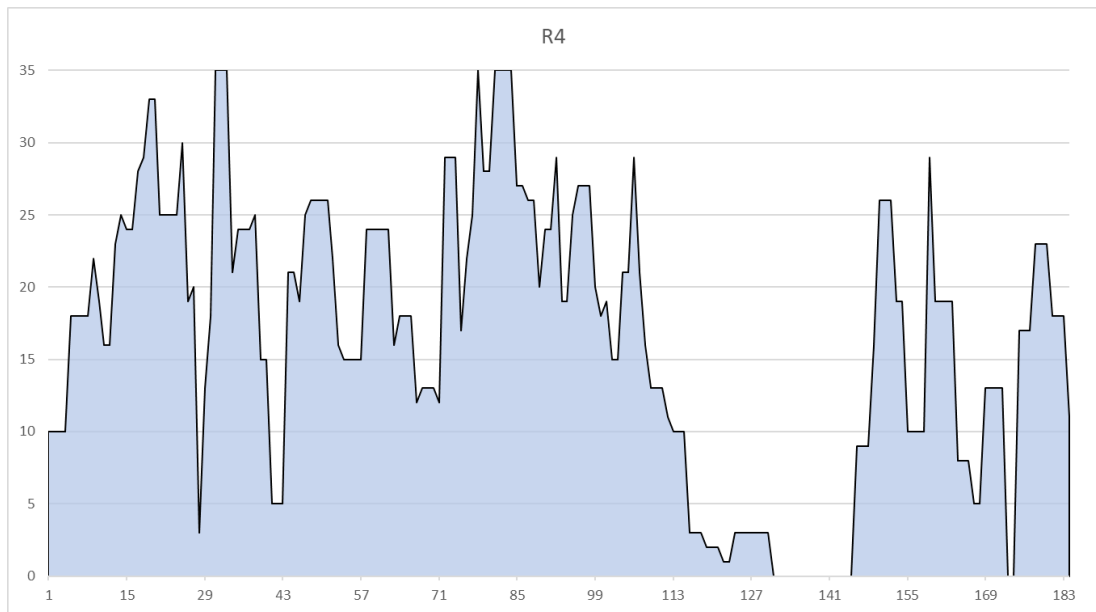
Σχήμα 5.19: Διάγραμμα χρήσης πόρου R2 αρχείου X54_4 με constant = 300.0.

Χρήση πόρου R2, Σχήμα 5.19, utilization = 57.00%.



Σχήμα 5.20: Διάγραμμα χρήσης πόρου R3 αρχείου X54_4 με constant = 300.0.

Χρήση πόρου R3, Σχήμα 5.20, utilization = 37.39%.



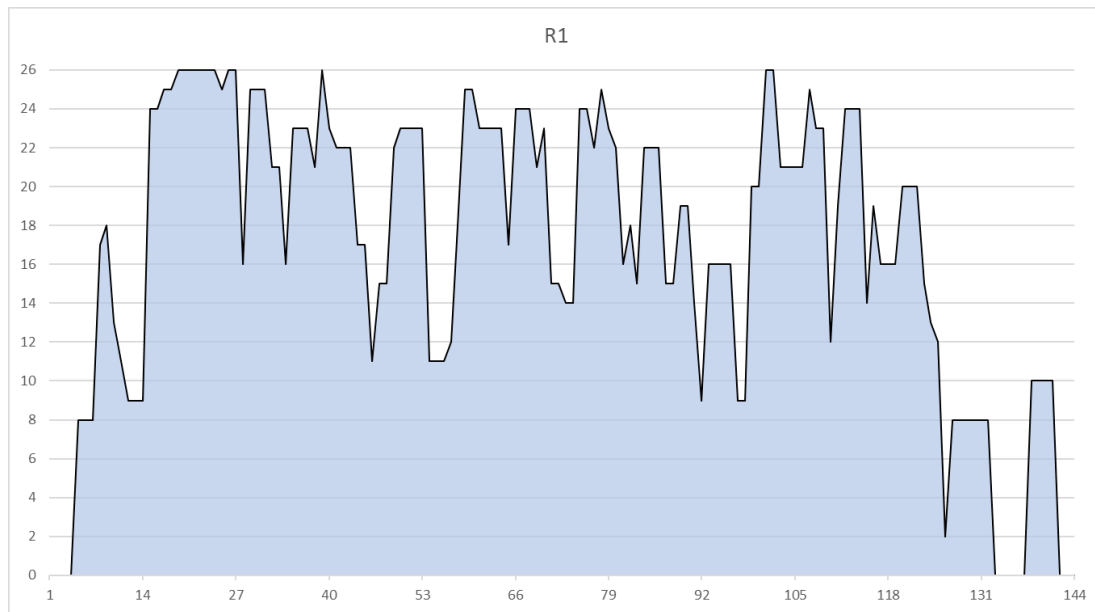
Σχήμα 5.21: Διάγραμμα χρήσης πόρου R4 αρχείου X54_4 με constant = 300.0.

Χρήση πόρου R4, Σχήμα 5.21, utilization = 47.10%.

Πίνακας 5.19: Αποτελέσματα αρχείου X54_4 με constant = 700.0.

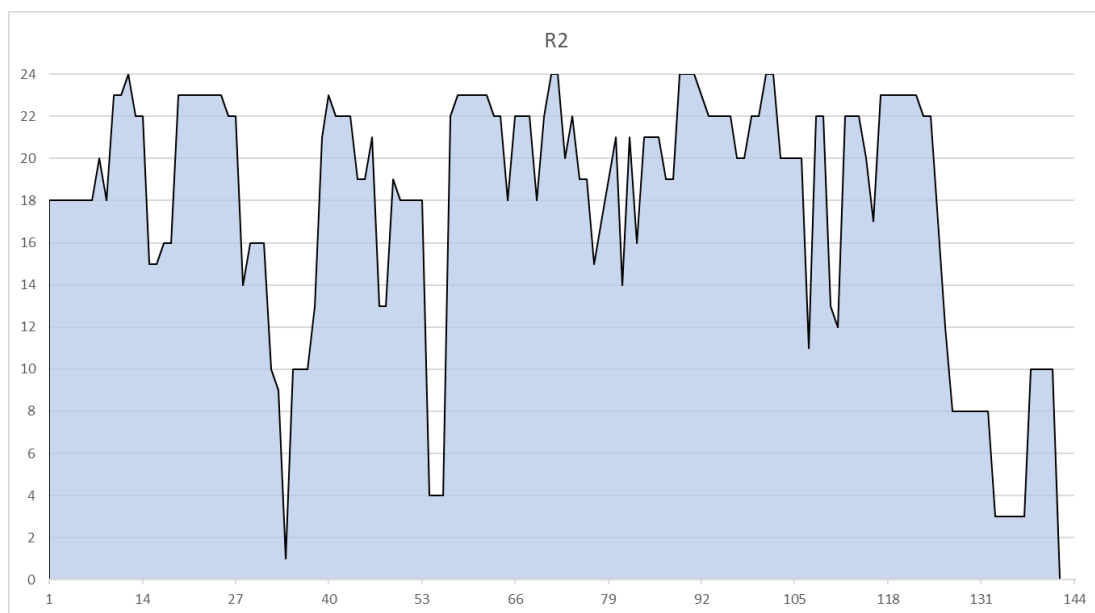
Init.	Dur.	Delay	Init. NPV	Duration	NPV	NPV Opt.	Duration Opt.	Time
ΜΟΝΗ ΣΥΝΘΗΚΗ								
144	165		8219.28	165	8219.28	0%	0%	263
ΔΙΠΛΗ ΣΥΝΘΗΚΗ								
149	171		8191.84	149	8191.84	0%	0%	198
ΔΙΠΛΗ ΑΝΤΙΣΤΡΟΦΗ ΣΥΝΘΗΚΗ								
157	180		8114.77	157	8114.77	0%	0%	214

Παρακάτω παρουσιάζονται τα διαγράμματα χρήσης πόρων για το αρχείο X54_4 με την εφαρμογή της μονής συνθήκης:



Σχήμα 5.22: Διάγραμμα χρήσης πόρου R1 αρχείου X54_4 με constant = 700.0.

Χρήση πόρου R1, Σχήμα 5.22, utilization = 64.88%.



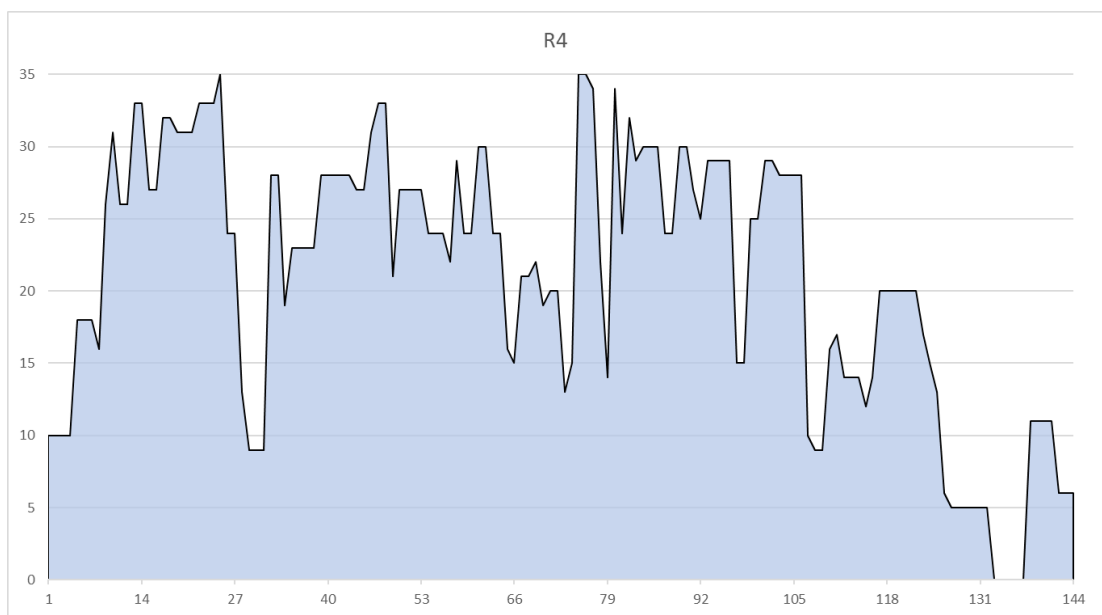
Σχήμα 5.23: Διάγραμμα χρήσης πόρου R2 αρχείου X54_4 με constant = 700.0.

Χρήση πόρου R2, Σχήμα 5.23, utilization = 72.83%.



Σχήμα 5.24: Διάγραμμα χρήσης πόρου R3 αρχείου X54_4 με constant = 700.0.

Χρήση πόρου R3, Σχήμα 5.24, utilization = 47.77%.



Σχήμα 5.25: Διάγραμμα χρήσης πόρου R4 αρχείου X54_4 με constant = 700.0.

Χρήση πόρου R4, Σχήμα 5.25, utilization = 60.18%.

Πίνακας 5.20: Ποσοστά χρήσης πόρων.

	X16_4			X54_4		
constant	100.0	350.0	700.0	20.0	300.0	700.0
R1	66.54%	56.52%	60.79%	55.28%	50.77%	64.88%
R2	63.55%	53.98%	58.06%	62.06%	57.00%	72.83%
R3	68.77%	58.42%	62.83%	40.71%	37.39%	47.77%
R4	73.88%	62.76%	67.50%	51.28%	47.10%	60.18%

5.4 Ανάλυση Ευαισθησίας

Σε αυτό το σημείο επιλέχθηκαν να πραγματοποιηθούν τρεις αναλύσεις ευαισθησίας στο αρχείο X16_4 με τις εξής παραμέτρους:

- Τιμή του constant:

1. 100.0,
2. 200.0 και
3. 300.0

- Αρχικός πληθυσμός, $initial = 10$,

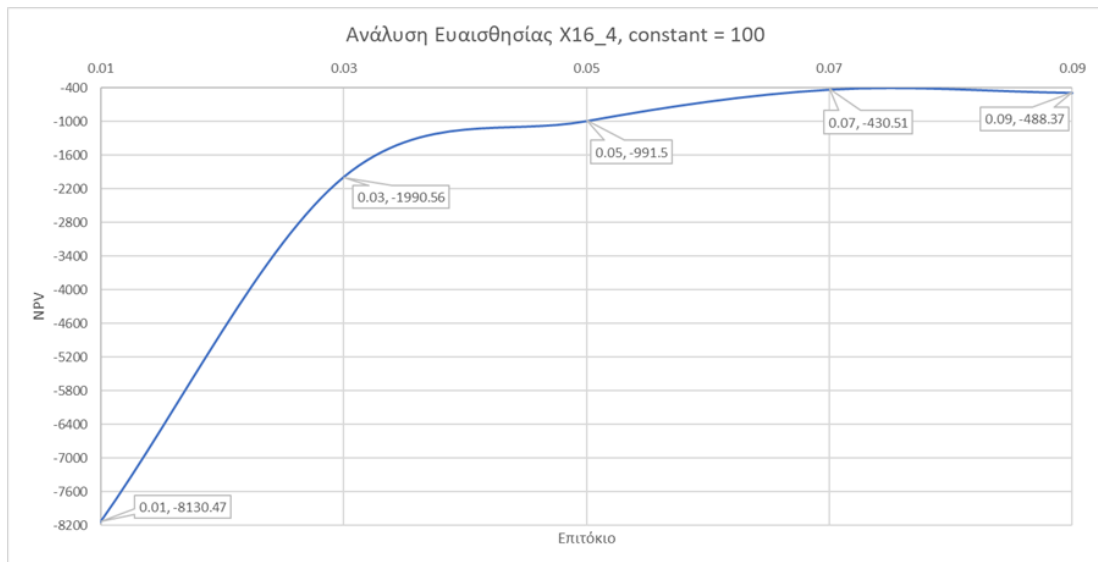
- Μέγεθος Τοπικής Έρευνας, $frg = 500$,

- Πλήθος διαφορετικών επιτοκίων $s_rates = 5$, με τιμές επιτοκίων:

1. 0.01,
2. 0.03,
3. 0.05,
4. 0.07 και
5. 0.09.

Πίνακας 5.21: Αποτελέσματα ανάλυσης ευαισθησίας αρχείου X16_4 με constant = 100.0.

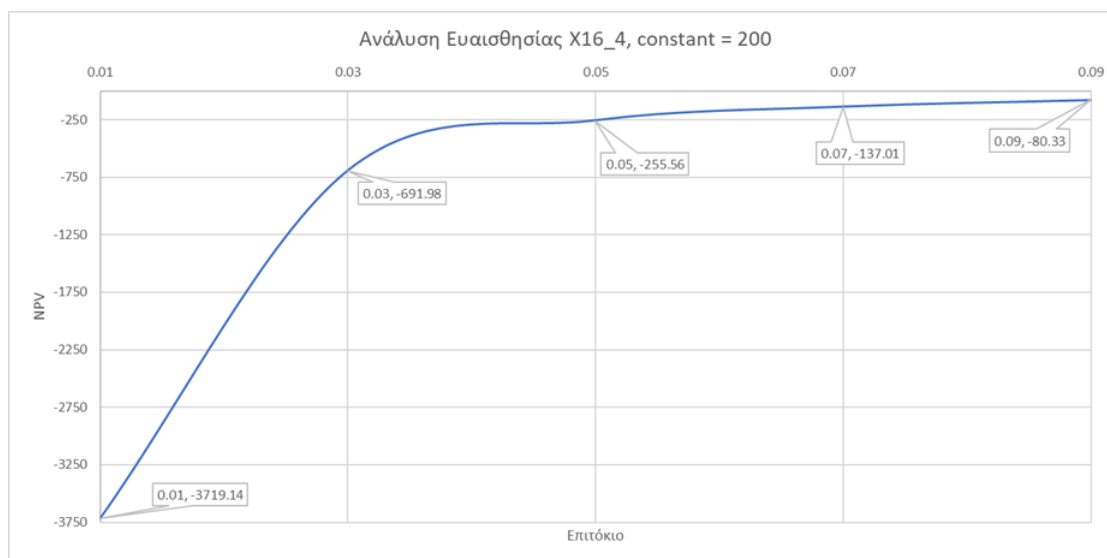
constant	Rate	Duration	NPV	Time	Loops
100	0.01	259	-8130.47	518.14	94500
	0.03	258	-1990.56		
	0.05	255	-991.5		
	0.07	261	-430.51		
	0.09	257	-488.37		



Σχήμα 5.26: Διάγραμμα NPV - Επιτοκίων με constant = 100.0.

Πίνακας 5.22: Αποτελέσματα ανάλυσης ευαισθησίας αρχείου X16_4 με constant = 200.0.

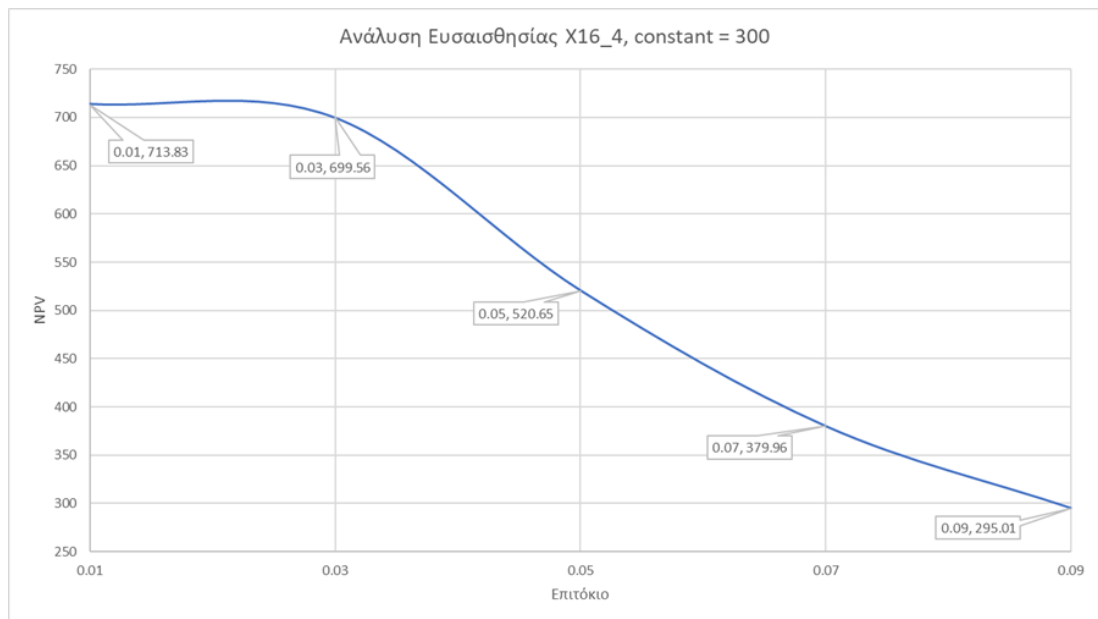
constant	Rate	Duration	NPV	Time	Loops
200	0.01	272	-3719.14	502.5	82000
	0.03	264	-691.98		
	0.05	267	-255.56		
	0.07	261	-137.01		
	0.09	255	-80.33		



Σχήμα 5.27: Διάγραμμα NPV - Επιτοκίων με constant = 200.0.

Πίνακας 5.23: Αποτελέσματα ανάλυσης ευαισθησίας αρχείου X16_4 με constant = 300.0.

constant	Rate	Duration	NPV	Time	Loops
300	0.01	295	541.52		
	0.03	310	676.14		
	0.05	293	514.8	673.75	84500
	0.07	294	373.05		
	0.09	301	288.44		



Σχήμα 5.28: Διάγραμμα NPV - Επιτοκίων με constant = 300.0.

ΚΕΦΑΛΑΙΟ

— 6 —

ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΕΡΑΙΤΕΡΩ ΕΡΕΥΝΑ

6.1 Συμπεράσματα

Από τον Πίνακα 5.1 είναι φανερό πως με την αύξηση του μεγέθους της Τοπικής Έρευνας βελτιώνεται συστηματικά η ποιότητα των λύσεων του αρχικού πληθυσμού. Οι τιμές των τελικών Loops στην περίπτωση του Bees 1 είναι αρκετά μεγαλύτερες διότι, υπενθυμίζεται από την υποενότητα 3.3.2 πως για κάθε καλύτερη λύση που δημιουργείται, ο αλγόριθμος επαναλαμβάνεται μέχρι να μην υπάρξει καλύτερη λύση.

Είναι σαφές λοιπόν πως η προσέγγιση των μελισσών παρουσιάζει μεγαλύτερες δυνατότητες στην παραγωγή ποιοτικότερων λύσεων από την προσέγγιση του S.A. υπό το κόστος των επαναλήψεων και κατά συνέπεια του χρόνου.

Από τους πίνακες της υποενότητας 5.3.1 παρατηρήθηκε πως:

1. Η χαμηλή εντατικοποίηση μονής συνθήκης παρουσίασε καλύτερες λύσεις σε 4 αρχεία,
2. Η μέτρια εντατικοποίηση μονής συνθήκης παρουσίασε τις λιγότερες καλύτερες λύσεις σε 2 αρχεία,
3. Η υψηλή εντατικοποίηση μονής συνθήκης παρουσίασε τις περισσότερες καλύτερες λύσεις σε 8 αρχεία,
4. Υψηλή εντατικοποίηση διπλής συνθήκης παρουσίασε καλύτερες λύσεις σε 7 αρχεία ενώ
5. Σε 6 αρχεία δεν παρουσιάστηκε βελτίωση.

Είναι φανερό πως η υψηλή εντατικοποίηση παρήγαγε τα περισσότερα και ποιοτικότερα αποτελέσματα από όλες τις προσεγγίσεις, ωστόσο η διπλή συνθήκη παρουσιάζει κάποια ιδιαιτερότητα, ιδίως σε προβλήματα όπου όλες ή σε πολύ μεγάλο βαθμό οι χρηματοροές των δραστηριοτήτων είναι αρνητικές. Για τον λόγο αυτό εφαρμόστηκε η ειδική μελέτη στα δύο αρχεία στην επόμενη υποενότητα.

Από την υποενότητα 5.3.2 μπορεί να διακριθεί πως σε μεγάλα προβλήματα όπου όλες οι δραστηριότητες ή σε πολύ μεγάλο αριθμό παρουσιάζουν αρνητικές

χρηματοροές, η διπλή αντίστροφη συνθήκη αποδίδει καλύτερα από τις υπόλοιπες.

Η απόρροια αυτή είναι αναμενόμενη καθώς ο αλγόριθμος δημιουργεί χειρότερες λύσεις, χρονικά, επιτρέποντας στον αλγόριθμο να εφαρμόσει μεγαλύτερη καθυστέρηση άρα κατά συνέπεια από την εξίσωση 3.2 περισσότερο χώρο για βελτίωση της npv . Η μεγάλη καθυστέρηση επιτρέπει την αναδιαμόρφωση του πίνακα χρήσης πόρων μεταφέροντας τους χρόνους έναρξης των δραστηριοτήτων αργότερα, δημιουργώντας με την κλήση της `Check_Schedule` ένα πιο πυκνό χρονοπρόγραμμα χωρίς να παραμένει άεργος κανένας πόρος.

Σε προβλήματα όπου οι αρνητικές χρηματοροές είναι λιγότερες από τις θετικές επικρατεί η μονή συνθήκη στην παραγωγή καλύτερων αποτελεσμάτων, ωστόσο σε προβλήματα αυτής της φύσεως με την εφαρμογή της καθυστέρησης είναι φανερό πως ενδέχεται να υπάρχουν χρονικές στιγμές όπου οι πόροι παραμένουν άεργοι.

Σε περιπτώσεις προβλημάτων όπου όλες οι χρηματοροές είναι θετικές η μονή συνθήκη παραμένει η αποδοτικότερη, ωστόσο δεν υπάρχει η εφαρμογή της καθυστέρησης και κατ' επέκτασης βελτίωση της npv .

Από τις αναλύσεις ευαισθησίας διακρίθηκε το γεγονός πως σε προβλήματα όπου οι περισσότερες δραστηριότητες είναι αρνητικές είναι επιθυμητό ένα πιο υψηλό επιτόκιο ωστόσο δεν υπάρχει τόσο σημαντική βελτίωση με την αύξηση του επιτοκίου. Στην αντίθετη περίπτωση είναι επιθυμητό ένα χαμηλό επιτόκιο και με την αύξηση του επιτοκίου παρουσιάστηκαν αισθητές μεταβολές στην τιμή της npv . Αυτό είναι αναμενόμενο καθώς από την εξίσωση 2.1 το επιτόκιο είναι αντιστρόφως ανάλογο της npv .

6.2 Περαιτέρω Έρευνα

Στην παρούσα διπλωματική ο ορισμός της διορίας, Δ , καθώς και οι τιμές των χρηματοροών προσδιορίστηκαν στατικά ώστε να υπάρχει μια σταθερή και ξεκάθαρη σύγκριση των αποτελεσμάτων. Αυτό δημιούργησε διάφορες επιπλοκές και γεννήθηκαν

τα ερωτήματα:

1. "Για κάθε πρόβλημα που επιλύει ο αλγόριθμος, ποια διάρκεια είναι αποδεκτή;
2. Με την λογική της καθυστέρησης, δεν μπορεί πρακτικά να εφαρμοστεί ιδιαίτερα μεγάλη καθυστέρηση ώστε να ελαχιστοποιηθεί κατά πολύ η *nprv*;"

Για την περαιτέρω αξιολόγηση του αλγορίθμου είναι θεμιτό να γίνει η εφαρμογή του πάνω σε προβλήματα όπου έχουν ξεκάθαρη διορία για το χρονοπρόγραμμα.

Σε αυτό το σημείο πρέπει να αναφερθεί το γεγονός ότι παρόλο που υλοποιήθηκε ο Simulated Annealing, δεν εφαρμόστηκε η καθυστέρηση σε αυτόν. Εφόσον δεν υπάρχει ξεκάθαρη σύγκριση, μπορεί η παρούσα διπλωματική να αξιοποιηθεί ως σημείο αναφοράς και σύγκρισης για μελλοντικές εργασίες.

Στο κεφάλαιο 3 αναφέρθηκε πως η δημιουργία του αρχικού πληθυσμού και των λύσεων της Τοπικής Έρευνας πραγματοποιήθηκε με την μέθοδο της Εμπρόσθιας Σειριακής Παραγωγής Χρονοπρογραμμάτων (Forward SSGS). Στο πρωταρχικό στάδιο κατά την υλοποίηση του αλγορίθμου είχε δημιουργηθεί και το μοντέλο της Σειριακής Παραγωγής Χρονοπρογραμμάτων Προς τα Πίσω (Backward SSGS) αλλά για λόγους χρονικούς διατηρήθηκε μόνο το Forward. Συμπερασματικά, ένας τρόπος βελτίωσης του αλγορίθμου είναι να εφαρμοστεί το Backward μοντέλο στην παρούσα διπλωματική και να επιλέγεται με βάση το πλήθος των αρνητικών χρηματοροών όπως το εφάρμοσαν οι Leyman και Vanhoucke, 2015.

Σε προβλήματα όπως του αρχείου X54_4 με $\text{constant} = 20.0$, Πίνακας 5.17, όπου η χρήση των πόρων είναι χαμηλή, από Σχήμα 5.14 έως Σχήμα 5.17 και Πίνακα 5.20, σε μελλοντικές έρευνες μπορεί να μελετηθεί:

- Η αποδέσμευση των πόρων σε χρονικά διαστήματα όπου παρουσιάζεται συγκεντρωτική χαμηλή απαίτηση των πόρων για την μεγιστοποίηση της χρήσης πόρων, ή
- Επίλυση του κλασικού προβλήματος RCPSP ή του RCPSP-DC με αντικειμενική συνάρτηση την χρήση των πόρων.

Τέλος, ένας ακόμη τρόπος βελτίωσης είναι να γίνει η λύση του προβλήματος RCPSP-DC με παραπάνω από μία αντικειμενική συνάρτηση. Στην παρούσα διπλωματική εφαρμόστηκαν τρία διαφορετικά κριτήρια στην υποενότητα 5.3.2, ωστόσο δεν μελετήθηκε εις βάθος η επίλυση του προβλήματος λαμβάνοντας υπόψη την βελτιστοποίηση ταυτόχρονα της *npu* και της χρονικής διάρκειας.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Άρθρα

- Cook, S. A. (1983). An Overview of Computational Complexity. *Commun. ACM*, 26(6), 400–408. <https://doi.org/10.1145/358141.358144>
- Garey, M. R., & Johnson, D. S. (1979). Computers and intractability. *A Guide to the*.
- Kolisch, R. (1996). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, 90(2), 320–333. [https://doi.org/https://doi.org/10.1016/0377-2217\(95\)00357-6](https://doi.org/https://doi.org/10.1016/0377-2217(95)00357-6)
- Kolisch, R., & Sprecher, A. (1997). PSPLIB - A project scheduling problem library: OR Software - ORSEP Operations Research Software Exchange Program. *European Journal of Operational Research*, 96(1), 205–216. [https://doi.org/https://doi.org/10.1016/S0377-2217\(96\)00170-1](https://doi.org/https://doi.org/10.1016/S0377-2217(96)00170-1)
- Leyman, P., & Vanhoucke, M. (2015). A new scheduling technique for the resource-constrained project scheduling problem with discounted cash flows. *International Journal of Production Research*, 53. <https://doi.org/10.1080/00207543.2014.980463>

Remer, D. S., & Nieto, A. P. (1995). A compendium and comparison of 25 project evaluation techniques. Part 1: Net present value and rate of return methods. *International Journal of Production Economics*, 42(1), 79–96. [https://doi.org/https://doi.org/10.1016/0925-5273\(95\)00104-2](https://doi.org/https://doi.org/10.1016/0925-5273(95)00104-2)

Βιβλία

Crundwell, F. (2008). *Finance for Engineers. Evalutaion and Funding of Capital Projects*.

Αραβώσης, Καρμπέρης & Σωτήρχος. (2011). *Τεχνοοικονομική Αξιολόγηση Επενδύσεων*. Οικονομική Βιβλιοθήκη.

Βαβάτσικος, Α. (2020). *Τεχνολογική Οικονομική*.

Μαρινάκης, Ι., Μαρινάκη, Μ., Μатσατσίνης, Ν., & Ζοπουνίδης, Κ. (2011). *Μεθευρετικοί και Εξελικτικοί Αλγόριθμοι σε Προβλήματα Δικοικητικής Επιστήμης*. ΚΛΕΙΔΑΡΙΘΜΟΣ.

Λοιπές Πηγές

Hargrave, M. (2022). Weighted Average Cost of Capital (WACC) Explained with Formula and Example. <https://www.investopedia.com/terms/w/wacc.asp>

Lioudis, N. (2022). Time Value of Money and the Dollar. <https://www.investopedia.com/ask/answers/032715/why-does-time-value-money-tvm-assume-dollar-today-worth-more-dollar-tomorrow.asp>

Majaski, C. (2022). Cost of Capital vs. Discount Rate: What's the Difference? <https://www.investopedia.com/ask/answers/052715/what-difference-between-cost-capital-and-discount-rate.asp>

OR-AS. (2015). Data files: The Patterson format. http://www.p2engine.com/p2reader/patterson_format

ΠΑΡΑΡΤΗΜΑ

— A' —

ΤΟ ΑΡΧΕΙΟ CHARTCONTROL.H

```
#pragma once
#include <wx/wx.h>
#include <vector>
#include <string>

class ChartControl : public wxWindow
{
public:
    ChartControl(wxWindow* parent, wxWindowID id,
        const wxPoint& pos, const wxSize& size);

    std::vector<double> values;
    std::string title;
    double max;
    double min;
    int time_value;

private:
    void OnPaint(wxPaintEvent& evt);
    std::tuple<int, double, double>
        calculateChartSegmentCountAndRange(double origLow,
            double origHigh);

    std::tuple<int, double, double>
        calculateChartSegmentCountAndRange2(double origLow,
            double origHigh);

};

class ChartControlSA : public wxWindow
{
public:
```

```
ChartControlSA(wxWindow* parent, wxWindowID id,
const wxPoint& pos, const wxSize& size);

    std::vector<double> rate_values;
    std::vector<double > npv_values;
    std::string title;
    int number = 0;
    double lv;
    double hv;
private:
    void OnPaint(wxPaintEvent& evt);
    std::tuple<int, double, double>
calculateChartSegmentCountAndRange(double origLow,
double origHigh);

    std::tuple<int, double, double>
calculateChartSegmentCountAndRange2(double origLow,
double origHigh);

};
```


ΠΑΡΑΡΤΗΜΑ

— Β' —

ΤΟ ΑΡΧΕΙΟ CHARTCONTROL.CPP

```

#include "chartcontrol.h"
#include <wx/settings.h>
#include <wx/graphics.h>
#include <wx/dcbuffer.h>
#include <algorithm>
#include "MainFrame.h"

```

```

ChartControl::ChartControl(wxWindow* parent, wxWindowID id,
const wxPoint& pos, const wxSize& size) :
wxWindow(parent, id, pos, size, wxFULL_REPAINT_ON_RESIZE)
{
    this->SetBackgroundStyle(wxBG_STYLE_PAINT);
    this->Bind(wxEVT_PAINT, &ChartControl::OnPaint, this);
}

```

```

void ChartControl::OnPaint(wxPaintEvent& evt)
{
    wxAutoBufferedPaintDC dc(this);
    dc.Clear();
    wxGraphicsContext* gc = wxGraphicsContext::Create(dc);

    if (gc && values.size() > 0)
    {
        wxFont titleFont = wxFont(wxNORMAL_FONT->GetPointSize() * 2.0,
        wxFONTFAMILY_DEFAULT, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_BOLD);
        gc->SetFont(titleFont, wxSystemSettings::
GetAppearance().IsDark() ? *wxWHITE:*wxBLACK);
        double tw, th;
        gc->GetTextExtent(this->title, &tw, &th);
        const double titleTopBottomMinimumMargin = this->FromDIP(10);
        wxRect2DDouble fullArea{ 0, 0, static_cast<double>(

```

```
GetSize().GetWidth()), static_cast<double>(
GetSize().GetHeight()) };

    const double marginX = fullArea.GetSize().GetWidth() / 8.0;
    const double marginTop = std::max(fullArea.GetSize().GetHeight()
/ 8.0, titleTopBottomMinimumMargin * 2.0 + th);
    const double marginBottom = fullArea.GetSize().GetHeight() / 8.0;
    double labelsToChartAreaMargin = this->FromDIP(10);
    wxRect2DDouble chartArea = fullArea;
    chartArea.Inset(marginX, marginTop, marginX, marginBottom);
    gc->DrawText(this->title, (fullArea.GetSize().GetWidth() - tw)
/ 2.0, (marginTop - th) / 2.0);

    wxAffineMatrix2D normalizedToChartArea{};
    normalizedToChartArea.Translate(chartArea.GetLeft(),
chartArea.GetTop());
    normalizedToChartArea.Scale(chartArea.m_width,
chartArea.m_height);

    double lowValue = min;
    double highValue = max;
    int hv = time_value;
    int lv = 0;

    const auto& [segmentCount, rangeLow, rangeHigh] =
calculateChartSegmentCountAndRange(lowValue, highValue);

    const auto& [segmentCountX, rangeLowX, rangeHighX] =
calculateChartSegmentCountAndRange2(lv, hv);

    double yValueSpan = rangeHigh - rangeLow;
    lowValue = rangeLow;
```

```
highValue = rangeHigh;

double yLinesCount = segmentCount + 1;

double xValueSpan = rangeHighX - rangeLowX;
lv = rangeLowX;
hv = rangeHighX;

double xLinesCount = segmentCountX + 1;

wxAffineMatrix2D normalizedToValue{};
normalizedToValue.Translate(0, highValue);
normalizedToValue.Scale(1, -1);
normalizedToValue.Scale(static_cast<double>(values.size() - 1),
yValueSpan);

wxAffineMatrix2D normalizedToValueX{};
normalizedToValueX.Translate(0, hv);
normalizedToValueX.Scale(1, 1);
normalizedToValueX.Scale(static_cast<int>(hv), 0);

gc->SetPen(wxPen(wxColor(128, 128, 128)));
gc->SetFont(*wxNORMAL_FONT, wxSystemSettings::
GetAppearance().IsDark() ? *wxWHITE : *wxBLACK);

for (int i = 0; i < yLinesCount; i++)
{
    double normalizedLineY = static_cast<double>(i)
/ (yLinesCount - 1);

    auto lineStartPoint = normalizedToChartArea.TransformPoint(
{ 0, normalizedLineY });
```

```
        auto lineEndPoint = normalizedToChartArea.TransformPoint(
{ 1, normalizedLineY });

        wxPoint2DDouble linePoints[] = { lineStartPoint,
lineEndPoint };
        gc->StrokeLines(2, linePoints);

        double valueAtLineY = normalizedToValue.TransformPoint(
{ 0, normalizedLineY }).m_y;

        auto text = wxString::Format("%.0f", valueAtLineY);
        text = wxControl::Ellipsize(text, dc, wxELLIPSIZE_MIDDLE,
chartArea.GetLeft() - labelsToChartAreaMargin);

        double tw, th;
        gc->GetTextExtent(text, &tw, &th);
        gc->DrawText(text, chartArea.GetLeft() -
labelsToChartAreaMargin - tw,
lineStartPoint.m_y - th / 2.0);
    }

    for (int i = 0; i < xLinesCount; i++)
    {
        double normalizedLineX = static_cast<double>(i)
/ (xLinesCount - 1);

        auto lineStartPoint = normalizedToChartArea.TransformPoint(
{ normalizedLineX , 0});
        auto lineEndPoint = normalizedToChartArea.TransformPoint(
{ normalizedLineX, 1 });

        wxPoint2DDouble linePoints[] = { lineStartPoint,
```

```
lineEndPoint };
gc->StrokeLines(2, linePoints);

double valueAtLineX = normalizedToValueX.TransformPoint(
{ normalizedLineX , 0 }).m_x;

auto text = wxString::Format("%.2f", valueAtLineX);
text = wxControl::Ellipsize(text, dc,
wxELLIPSIZE_MIDDLE, chartArea.GetBottom() -
labelsToChartAreaMargin);

double tw, th;
gc->GetTextExtent(text, &tw, &th);
gc->DrawText(text, lineStartPoint.m_x - tw/2.0,
chartArea.GetBottom() + th);
}

wxPoint2DDouble leftHLinePoints[] = {
normalizedToChartArea.TransformPoint({0, 0}),
normalizedToChartArea.TransformPoint({0, 1}) };

wxPoint2DDouble rightHLinePoints[] = {
normalizedToChartArea.TransformPoint({1, 0}),
normalizedToChartArea.TransformPoint({1, 1}) };

gc->StrokeLines(2, leftHLinePoints);
gc->StrokeLines(2, rightHLinePoints);

wxPoint2DDouble* pointArray =
new wxPoint2DDouble[values.size()];

wxAffineMatrix2D valueToNormalized = normalizedToValue;
```

```
        valueToNormalized.Invert();
        wxAffineMatrix2D valueToChartArea = normalizedToChartArea;
        valueToChartArea.Concat(valueToNormalized);

        for (int i = 0; i < values.size(); i++)
        {
            pointArray[i] = valueToChartArea.TransformPoint(
{ static_cast<double>(i), values[i] });
        }

        gc->SetPen(wxPen(wxSystemSettings::GetAppearance().IsDark()
? *wxCYAN : *wxBLUE, 3));
        gc->StrokeLines(values.size(), pointArray);

        delete[] pointArray;
        delete gc;
    }
}

ChartControlSA::ChartControlSA(wxWindow* parent, wxWindowID id,
const wxPoint& pos, const wxSize& size) : wxWindow(parent, id,
pos, size, wxFULL_REPAINT_ON_RESIZE)
{
    this->SetBackgroundStyle(wxBG_STYLE_PAINT);
    this->Bind(wxEVT_PAINT, &ChartControlSA::OnPaint, this);
}

void ChartControlSA::OnPaint(wxPaintEvent& evt)
{
    wxAutoBufferedPaintDC dc(this);
    dc.Clear();
    wxGraphicsContext* gc = wxGraphicsContext::Create(dc);
```

```
if (gc && rate_values.size() > 0)
{
    wxFont titleFont = wxFont(wxNORMAL_FONT->GetPointSize() * 2.0,
                                wxFONTFAMILY_DEFAULT, wxFONTSTYLE_NORMAL,
wxFONTWEIGHT_BOLD);

    gc->SetFont(titleFont, wxSystemSettings::
GetAppearance().IsDark() ? *wxWHITE : *wxBLACK);
    double tw, th;
    gc->GetTextExtent(this->title, &tw, &th);

    const double titleTopBottomMinimumMargin = this->FromDIP(10);

    wxRect2DDouble fullArea{ 0, 0, static_cast<double>(
GetSize().GetWidth()), static_cast<double>(
GetSize().GetHeight()) };

    const double marginX = fullArea.GetSize().GetWidth() / 8.0;
    const double marginTop = std::max(fullArea.GetSize().GetHeight()
/ 8.0, titleTopBottomMinimumMargin * 2.0 + th);
    const double marginBottom = fullArea.GetSize().GetHeight()
/ 8.0;
    double labelsToChartAreaMargin = this->FromDIP(10);
    wxRect2DDouble chartArea = fullArea;
    chartArea.Inset(marginX, marginTop, marginX, marginBottom);

    gc->DrawText(this->title, (fullArea.GetSize().GetWidth() - tw)
/ 2.0, (marginTop - th) / 2.0);

    wxAffineMatrix2D normalizedToChartArea{};
    normalizedToChartArea.Translate(chartArea.GetLeft(),
```



```
chartArea.GetTop());
    normalizedToChartArea.Scale(chartArea.m_width,
chartArea.m_height);

    double lowValue = *std::min_element(npv_values.begin(),
npv_values.end());
    double highValue = *std::max_element(npv_values.begin(),
npv_values.end());
    double val = lv;
    const auto& [segmentCount, rangeLow, rangeHigh] =
calculateChartSegmentCountAndRange(lowValue, highValue);

    const auto& [segmentCountX, rangeLowX, rangeHighX] =
calculateChartSegmentCountAndRange2(lv, hv);

    double yValueSpan = rangeHigh - rangeLow;
    lowValue = rangeLow;
    highValue = rangeHigh;

    double yLinesCount = segmentCount + 1;

    double xValueSpan = rangeHighX - rangeLowX;
    lv = rangeLowX;
    hv = rangeHighX;

    wxAffineMatrix2D normalizedToValue{};
    normalizedToValue.Translate(0, highValue);
    normalizedToValue.Scale(1, -1);
    normalizedToValue.Scale(static_cast<double>(
npv_values.size() - 1), yValueSpan);

    wxAffineMatrix2D normalizedToValueX{};
```

```

        normalizedToValueX.Translate(val, 0);
        normalizedToValueX.Scale(1, 0);
        normalizedToValueX.Scale(xValueSpan, 0);

        gc->SetPen(wxPen(wxColor(128, 128, 128)));
        gc->SetFont(*wxNORMAL_FONT, wxSystemSettings::
GetAppearance().IsDark() ? *wxWHITE : *wxBLACK);

        for (int i = 0; i < yLinesCount; i++)
        {
            double normalizedLineY = static_cast<double>(i) /
(yLinesCount - 1);

            auto lineStartPoint =
normalizedToChartArea.TransformPoint({ 0, normalizedLineY });
            auto lineEndPoint =
normalizedToChartArea.TransformPoint({ 1, normalizedLineY });

            wxPoint2DDouble linePoints[] = { lineStartPoint,
ineEndPoint };
            gc->StrokeLines(2, linePoints);

            double valueAtLineY = normalizedToValue.TransformPoint(
{ 0, normalizedLineY }).m_y;

            auto text = wxString::Format("%.1f", valueAtLineY);
            text = wxControl::Ellipsize(text, dc, wxELLIPSIZE_MIDDLE,
chartArea.GetLeft() - labelsToChartAreaMargin);

            double tw, th;
            gc->GetTextExtent(text, &tw, &th);
            gc->DrawText(text, chartArea.GetLeft() -

```

```
labelsToChartAreaMargin - tw,
lineStartPoint.m_y - th / 2.0);
    }

    wxPoint2DDouble leftHLinePoints[] = {
        normalizedToChartArea.TransformPoint({0, 0}),
        normalizedToChartArea.TransformPoint({0, 1}) };

    wxPoint2DDouble rightHLinePoints[] = {
        normalizedToChartArea.TransformPoint({1, 0}),
        normalizedToChartArea.TransformPoint({1, 1}) };

    gc->StrokeLines(2, leftHLinePoints);
    gc->StrokeLines(2, rightHLinePoints);

    wxPoint2DDouble* pointArray =
new wxPoint2DDouble[rate_values.size()];

    wxAffineMatrix2D valueToNormalized = normalizedToValue;
    valueToNormalized.Invert();
    wxAffineMatrix2D valueToChartArea = normalizedToChartArea;
    valueToChartArea.Concat(valueToNormalized);

    for (int i = 0; i < rate_values.size(); i++)
    {
        pointArray[i] = valueToChartArea.TransformPoint(
{ static_cast<double>(i), npv_values[i] });
    }

    gc->SetPen(wxPen(wxSystemSettings::GetAppearance().IsDark()
? *wxCYAN : *wxBLUE, 3));
    gc->StrokeLines(rate_values.size(), pointArray);
```

```
        delete[] pointArray;
        delete gc;
    }
}

std::tuple<int, double, double> ChartControlSA::
calculateChartSegmentCountAndRange(
double origLow, double origHigh)
{
    constexpr double rangeMults[] = { 0.2, 0.25, 0.5, 1.0, 2.0, 2.5, 5.0 };
    constexpr int maxSegments = 6;

    double magnitude = std::floor(std::log10(origHigh - origLow));

    for (auto r : rangeMults)
    {
        double stepSize = r * std::pow(10.0, magnitude);
        double low = std::floor(origLow / stepSize) * stepSize;
        double high = std::ceil(origHigh / stepSize) * stepSize;

        int segments = round((high - low) / stepSize);

        if (segments <= maxSegments)
        {
            return std::make_tuple(segments, low, high);
        }
    }
    return std::make_tuple(10, origLow, origHigh);
}

std::tuple<int, double, double> ChartControlSA::
```

```
calculateChartSegmentCountAndRange2(
double origLow, double origHigh)
{
    constexpr double rangeMults[] = { 0.2, 0.25, 0.5, 1.0, 2.0};
    constexpr int maxSegments = 6;

    double magnitude = std::floor(std::log10(origHigh - origLow));

    for (auto r : rangeMults)
    {
        double stepSize = r * std::pow(10.0, magnitude);
        double low = std::floor(origLow / stepSize) * stepSize;
        double high = std::ceil(origHigh / stepSize);
        int segments = round((high - low) / stepSize);

        if (segments <= maxSegments)
        {
            return std::make_tuple(segments, low, high);
        }
    }
    return std::make_tuple(10, origLow, origHigh);
}

std::tuple<int, double, double> ChartControl::
calculateChartSegmentCountAndRange(
double origLow, double origHigh)
{
    constexpr double rangeMults[] = { 0.1, 0.2, 0.25, 0.5, 1.0,
2.0, 2.5, 5.0 };
    constexpr int maxSegments = 30;

    double magnitude = std::floor(std::log10(origHigh - origLow));
```

```
for (auto r : rangeMults)
{
    double stepSize = r * std::pow(10.0, magnitude);
    double low = std::floor(origLow / stepSize) * stepSize;
    double high = std::ceil(origHigh / stepSize) * stepSize;

    int segments = round((high - low) / stepSize);

    if (segments <= maxSegments)
    {
        return std::make_tuple(segments, low, high);
    }
}
return std::make_tuple(10, origLow, origHigh);
}

std::tuple<int, double, double> ChartControl::
calculateChartSegmentCountAndRange2(
double origLow, double origHigh)
{
    constexpr double rangeMults[] = { 0.05, 0.1 ,0.2, 0.25, 0.5, 1.0,
2.0, 2.5, 5.0 };
    constexpr int maxSegments = 100;

    double magnitude = std::floor(std::log10(origHigh - origLow));

    for (auto r : rangeMults)
    {
        double stepSize = r * std::pow(10.0, magnitude);
        double low = std::floor(origLow / stepSize) * stepSize;
        double high = std::ceil(origHigh / stepSize) * stepSize;
```

```
    int segments = round((high - low) / stepSize);

    if (segments <= maxSegments)
    {
        return std::make_tuple(segments, low, high);
    }
}

return std::make_tuple(10, origLow, origHigh);
}
```


ΠΑΡΑΡΤΗΜΑ

Γ'

ΤΟ ΑΡΧΕΙΟ RCPSP.H

```
#ifndef RCPSP_H
#define RCPSP_H
#include <iostream>
#include <iomanip>
#include <algorithm>
#include <vector>
#include <string.h>
#include <fstream>
#include <sstream>
#include <numeric>
#include <cmath>
#include <iostream>
using namespace std;

class rcpsp
{
private:
    int activity;
    int duration;
    vector<int> demand;
    int numofsucc;
    vector<int> successors;
    vector<int> predecessors;
    double cf;
    int start;
    bool activityCheck;
    double NPV;
public:
    rcpsp()
    {
        activity = 0;
        duration = 0;
```

```
        demand.clear();
        numofsucc = 0;
        successors.clear();
        predecessors.clear();
        cf = 0.0;
        start = 0;
        activityCheck = false;
        NPV = 0.0;
    }

    rcpsp(int in_activity, int in_duration, vector<int> in_demand,
int in_numofsucc, vector<int> in_successors, vector<int> in_predecessors,
        double in_cf, int in_start, bool in_activityCheck, double in_NPV)
    {
        activity = in_activity;
        duration = in_duration;
        demand = in_demand;
        numofsucc = in_numofsucc;
        successors = in_successors;
        predecessors = in_predecessors;
        cf = in_cf;
        start = in_start;
        activityCheck = in_activityCheck;
        NPV = in_NPV;
    }

    ~rcpsp() {

    }

    int getActivity() { return activity; }
    int getDuration() { return duration; }
    int getDemand(int position) { return demand[position]; }
```

```
int getNumOfSucc() { return numofsucc; }
int getSuccessor(int position) { return successors[position]; }
double getCashFlow() { return cf; }
void setCF(double value) { cf = value; }
double getCF() { return cf; }
int getPredecessor(int position) { return predecessors[position]; }
size_t getNumOfPred() { return predecessors.size(); }
int getStart() { return start; }
void setStart(int value) { start = value; }
bool getStatus() { return activityCheck; }
void setStatus(bool stat) { activityCheck = stat; }
double getNPV() { return NPV; }
void setNPV(double cash, double rate, int period)
{ double t = pow(1.0 + rate, period); NPV = cash / t; }
int getFinish() { return start + duration; }
void PredecessorVector(int Act) { predecessors.push_back(Act); }
}*element;

int inputs;
int* resourceCap;
int resourceNum;
int MaxTime = 0;
double TotalNPV = 0.0;
vector<double> rcost;
double ConstantV = 100.0;
double ResourceCost = 15.0;
#endif
```

ΠΑΡΑΡΤΗΜΑ

Δ'

ΤΟ ΑΡΧΕΙΟ ΒΒΑ.Η

```
#ifndef BBA_H
#define BBA_H

#include <iostream>
#include <iomanip>
#include <algorithm>
#include <vector>
#include <numeric>
#include <cmath>
#include "rcpsp.h"

using namespace std;

class In_Pop
{
private:
    vector<int> solution;
    double NPV;
    int duration;
    vector<double> prio;

public:
    In_Pop()
    {
        solution.clear();
        NPV = 0.0;
        duration = 0;
        prio.clear();
    }

    In_Pop(vector<int> in_solution, double in_npv, int in_dur,
vector<double> in_prio)
```

```
{
    solution = in_solution;
    NPV = in_npv;
    duration = in_dur;
    prio = in_prio;
}
~In_Pop() {

}
double GetNPV() { return NPV; }
vector<int> GetSolution() { return solution; }
vector<double> GetPrio() { return prio; }
int GetSolutionDuration() { return duration; }
int Sol(int value) { return solution[value]; }
double Prio(int value) { return prio[value]; }
}*scout_bees;
```

```
class Foragers
{
private:
    vector<int> solution;
    int duration;
    vector<vector<int>> Gannt;
    double Total_NPV;
    vector<double> prio;
    vector<double> NPV;
    vector<bool> status;
    vector<int> act_start;
    vector<int> act_finish;
    vector<bool> check;
public:
```

```
Foragers()
{
    solution.clear();
    Total_NPV = 0.0;
    NPV.clear();
    duration = 0;
    prio.clear();
    Gannt.clear();
    status.clear();
    act_start.clear();
    act_finish.clear();
    check.clear();
}

Foragers(vector<int> in_sol, int in_dur, vector<vector<int>>
in_gannt, vector<double> in_prio, double in_total_npv,
vector<double> in_npv, vector<bool> in_status, vector<int>
in_start, vector<int> in_finish, vector<bool> in_check)
{
    solution = in_sol;
    duration = in_dur;
    Gannt = in_gannt;
    prio = in_prio;
    Total_NPV = in_total_npv;
    NPV = in_npv;
    status = in_status;
    act_start = in_start;
    act_finish = in_finish;
    check = in_check;
}

~Foragers() {
```



```
    }

    int getStart(int pos) { return act_start[pos]; }
    void setStart(int pos, int value)
{ act_start[pos] = value; }
    void setFinish(int act, int value)
{ act_finish[act] = act_start[act] + value; }
    int getFinish(int act) { return act_finish[act]; }
    int getDuration() { return duration; }
    bool getStatus(int pos) { return status[pos]; }
    void setStatus(int pos, bool stat)
{ status[pos] = stat; }
    void setNPV(int act, double value)
{ NPV[act] = value; }
    vector<double> getNPVVec() { return NPV; }
    vector<double> getPrio() { return prio; }
    vector<bool> getStatusVec() { return status; }
    vector<int> getStartVec() { return act_start; }
    vector<int> getFinishVec() { return act_finish; }
    vector<bool> getCheckVec() { return check; }
    vector<int> getSolVec() { return solution; }
    vector<vector<int>> getGannt() { return Gannt; }
    double getTotalNPV() { return Total_NPV; }

}***forager_bees;

class Best
{
private:
    vector<int> solution;
    int duration;
```

```
vector<vector<int>> Gannt;
double Total_NPV;
vector<double> NPV;
vector<int> act_start;
vector<int> act_finish;
vector<bool> check;
public:
    Best()
    {
        solution.clear();
        Total_NPV = 0.0;
        NPV.clear();
        duration = 0;
        Gannt.clear();
        act_start.clear();
        act_finish.clear();
        check.clear();
    }

    Best(vector<int> in_sol, int in_dur, vector<vector<int>>
in_gannt, double in_total_npv, vector<double> in_npv,
vector<int> in_start, vector<int> in_finish,
vector<bool> in_check)
    {
        solution = in_sol;
        duration = in_dur;
        Gannt = in_gannt;
        Total_NPV = in_total_npv;
        NPV = in_npv;
        act_start = in_start;
        act_finish = in_finish;
        check = in_check;
```

```
    }

    ~Best() {

    }

    void setCheck(int act, bool val) { check[act] = val; }
    bool getCheck(int act) { return check[act]; }
    int getFinish(int act) { return act_finish[act]; }
    double getTotalNPV() { return Total_NPV; }
    vector<int> getSolVec() { return solution; }
    vector<vector<int>> getGannt() { return Gannt; }
    vector<double> getNPVVec() { return NPV; }
    vector<int> getStartVec() { return act_start; }
    vector<int> getFinishVec() { return act_finish; }
    double getNPV(int act) { return NPV[act]; }

}*best_bees;

class Elite
{
private:
    vector<int> solution;
    int duration;
    vector<vector<int>> Gannt;
    double Total_NPV;
    vector<double> NPV;
    vector<int> act_start;
    vector<int> act_finish;
public:
    Elite()
    {
```

```

        solution.clear();
        Total_NPV = 0.0;
        NPV.clear();
        duration = 0;
        Gannt.clear();
        act_start.clear();
        act_finish.clear();
    }
    Elite(vector<int> in_sol, int in_dur, vector<vector<int>>
in_gannt, double in_total_npv, vector<double> in_npv,
vector<int> in_start, vector<int> in_finish)
    {
        solution = in_sol;
        duration = in_dur;
        Gannt = in_gannt;
        Total_NPV = in_total_npv;
        NPV = in_npv;
        act_start = in_start;
        act_finish = in_finish;
    }
    ~Elite() {

    }

    int getStart(int pos) { return act_start[pos]; }
    void setStart(int pos, int value)
    { act_start[pos] = value; }
    void setNEWStart(int pos, int value)
    { act_start[pos] = act_start[pos] + value; }
    vector<int> getStartVec() { return act_start; }
    int getFinish(int act) { return act_finish[act]; }
    void setFinish(int act, int value)
    { act_finish[act] = act_start[act] + value; }

```

```

    void setNEWFinish(int act, int value)
{ act_finish[act] = value; }
    vector<int> getFinishVec() { return act_finish; }
    vector<vector<int>> getGannt() { return Gannt; }
    void setGannt(vector<vector<int>> value)
{ Gannt = value; }
    void resizeGannt(int rows, int new_size)
{ Gannt.resize(rows, vector<int>(new_size)); }
    int PrintGannt(int time, int resource)
{ return Gannt[time][resource]; }
    int getSol(int pos) { return solution[pos]; }
    size_t getSolSize() { return solution.size(); }
    vector<int> getSolVec() { return solution; }
    int getDuration() { return duration; }
    void setDurationNEW(int value) { duration = value; }
    double getNPV(int act) { return NPV[act]; }
    void setNPV(int act, double value) { NPV[act] = value; }
    vector<double> getNPVVec() { return NPV; }
    void setTotalNPV(double val) { Total_NPV = val; }
    double getTotalNPV() { return Total_NPV; }

}*elite_bees;

class Opt
{
private:
    vector<int> solution;
    int duration;
    vector<vector<int>> Gannt;
    double Total_NPV;
    vector<double> NPV;
    vector<int> act_start;

```

```
    vector<int> act_finish;
public:
    Opt()
    {
        solution.clear();
        Total_NPV = 0.0;
        NPV.clear();
        duration = 0;
        Gannt.clear();
        act_start.clear();
        act_finish.clear();
    }
    Opt(vector<int> in_sol, int in_dur, vector<vector<int>>
in_gannt, double in_total_npv, vector<double> in_npv,
vector<int> in_start, vector<int> in_finish)
    {
        solution = in_sol;
        duration = in_dur;
        Gannt = in_gannt;
        Total_NPV = in_total_npv;
        NPV = in_npv;
        act_start = in_start;
        act_finish = in_finish;
    }
    ~Opt() {

    }
    size_t getGanntSize() { return Gannt.size(); }
    int PrintGannt(int time, int resource)
{ return Gannt[time][resource]; }
    double getNPV(int act) { return NPV[act]; }
    int getFinish(int act) { return act_finish[act]; }
```

```
    int getStart(int pos) { return act_start[pos]; }
    int getSol(int pos) { return solution[pos]; }
    size_t getSolSize() { return solution.size(); }
    double getTotalNPV() { return Total_NPV; }
    int getDuration() { return duration; }
}*final;
#endif
```


ΠΑΡΑΡΤΗΜΑ

Ε΄

ΤΟ ΑΡΧΕΙΟ MAINFRAME.H

```
#pragma once
#include "chartcontrol.h"
#include <wx/wx.h>
#include <wx/event.h>
#include <wx/spinctrl.h>
#include <thread>
#include <chrono>
#include <random>
#include <omp.h>
#include <iostream>
#include <vector>
using namespace std;

class MainFrame : public wxFrame
{
public:
MainFrame(const wxString& title);
// menu
void OnMenuOpen(wxCommandEvent& evt);
void OnMenuExit(wxCommandEvent& evt);
//menu1
void ExportCSV(wxCommandEvent& evt);
void ExportSA(wxCommandEvent& evt);
//menu2
void OnResourceGraph(wxCommandEvent& evt);
void OnSense(wxCommandEvent& evt);
//menu3
void OnHelp(wxCommandEvent& evt);
void OnSolInit(wxCommandEvent& evt);
void OnForagerSPCTRL(wxSpinEvent& evt);
void OnRateSpin(wxSpinDoubleEvent& evt);
void OnInitialSpin(wxSpinEvent& evt);
```

```
void OnResourceSpin(wxSpinEvent& evt);
void OnSolutionSpin(wxSpinEvent& evt);
void OnRun(wxCommandEvent& evt);
void OnChoice(wxCommandEvent& evt);
void OnClose(wxCloseEvent& evt);
void OnDefineRates(wxCommandEvent& evt);
void OnSensitivity(wxCommandEvent& evt);
void data();
void CreatePredecessors();
int createCF();
int Early_Start(int act, int trigger, int s_index, int frg);
int Latest_Finish(int act);
vector<int> Forward_Activity_Check(vector<int> next_act,
int* act, vector<double> random, int trigger,
int s_index, int frg);
vector<int> Forward_Activity_List(vector<double> random,
int trigger, int s_index, int frg);
vector<vector<int>> Forward_Resource_Check(vector<vector<int>> GanntR,
vector<int> solution, double* npv, int* finish, int trigger,
int s_index, int frg);
void SSGS_Forward(vector<double> random, int s_index, int frg,
int trigger);
void Local_Search(int local_size, int scout, vector<bool> dummy_status,
vector<int> dummy_start, vector<int> dummy_finish,
vector<double> dummy_npv, vector<bool> dummy_check);
vector<int> delay(vector<int> set, vector<int> set2, int sol);
bool Push_Delay(int bestbee, int delay, vector<int> set1, int deadline);
void Find_Network_Set(int current, int start, int pt1[], int pt2[],
double* total_npv, int sol, bool status, int count1, int count2);
void Network_Delay(int deadline, int bestbee);
void Find_Schedule_Set(int current, int start, int pt1[],
double* total_npv, int sol, bool status, int count1);
```

```
vector<int> Schedule_Set_2(vector<int> set1, int sol);
void Schedule_Delay(int bestbee, int deadline);
void Check_Schedule(int bees);
void Initial_Population(int N, vector<double> random,
vector<int> solution);

protected:
bool processing{ false };
std::atomic<bool> quitRequested{ false };
wxGauge* gauge;
wxGauge* Netgauge;
double dur = 0.0;
double dur_sec = 0.0;
wxString va;
wxPanel* panel;
int time_limit = 300;
int global_counter;
thread backgroundThread;
wxButton* run_button;
wxButton* init;
wxStaticText* gauge_text;
wxMenu* menu = new wxMenu();
wxMenu* menu1 = new wxMenu();
wxMenu* menu2 = new wxMenu();
wxMenu* menu3 = new wxMenu();
wxStaticText* label;
wxStaticText* label1;
int first = 0;
ChartControl* chart;
ChartControlSA* gchart;
wxSpinCtrl* resource_count;
wxSpinCtrl* solution_count;
```

```
wxButton* plot;
wxChoice* choice;
string ch =("Comma");
wxDECLARE_EVENT_TABLE();
};

class NumberDlg : public wxDialog
{
public:
NumberDlg() : wxDialog(NULL, wxID_ANY, _("Specify number of rates"))
{
wxFlexGridSizer* rate_size_win = new wxFlexGridSizer(2);
rate_size_win->Add(new wxStaticText(this, wxID_ANY,
_("&Rate Size:")), 0, wxALL | wxALIGN_RIGHT, 5);
rate_size = new wxTextCtrl(this, wxID_ANY);
rate_size_win->Add(rate_size, 0, wxALL, 5);
wxBoxSizer* mainSizer = new wxBoxSizer(wxVERTICAL);
mainSizer->Add(rate_size_win, 0, wxALL, 5);
mainSizer->Add(CreateButtonSizer(wxOK | wxCANCEL), 0, wxALL, 5);
SetSizerAndFit(mainSizer);
Centre();
}

int GetNumber() const
{
int test = wxAtoi(rate_size->GetValue());
return test;
};

void SetNumber(const wxString& user)
{
rate_size->SetValue(user);
}

private:
```

```
wxTextCtrl* rate_size;
};

class RateDlg : public wxDialog
{
public:
RateDlg() : wxDialog(NULL, wxID_ANY, _("Enter rate"))
{
wxFlexGridSizer* rate_win = new wxFlexGridSizer(2);
rate_win->Add(new wxStaticText(this, wxID_ANY,
_("&Rate:")), 0, wxALL | wxALIGN_RIGHT, 5);
rate_input = new wxTextCtrl(this, wxID_ANY);
rate_win->Add(rate_input, 0, wxALL, 5);
wxBoxSizer* mainSizer = new wxBoxSizer(wxVERTICAL);
mainSizer->Add(rate_win, 0, wxALL, 5);
mainSizer->Add(CreateButtonSizer(wxOK | wxCANCEL), 0, wxALL, 5);
SetSizerAndFit(mainSizer);
Centre();
}

double GetNumber() const
{
double test = wxAtof(rate_input->GetValue());
return test;
};

void SetNumber(const wxString& user)
{
rate_input->SetValue(user);
}

private:
wxTextCtrl* rate_input;
};
```

ΠΑΡΑΡΤΗΜΑ

— ΣΤ' —

ΤΟ ΑΡΧΕΙΟ MAINFRAME.CPP

```
#include "MainFrame.h"
#include "rcpsp.h"
#include "bba.h"
#include "chartcontrol.h"
#include <wx/wx.h>
#include <wx/textfile.h>
#include <wx/event.h>
#include <wx/spinctrl.h>
#include <wx/wfstream.h>
#include <wx/txtstrm.h>
#include <wx/hyperlink.h>
#include <wx/datetime.h>
#include <wx/thread.h>
#include "wx/wxprec.h"
#include <thread>
#include <chrono>
#include <random>
#include <omp.h>
#include <iostream>
#include <vector>
#include <algorithm>
#include <iterator>
using namespace std;
using namespace std::chrono;

class App : public wxApp
{
public:
    bool OnInit();
};

DECLARE_APP(App);
```



```
bool App::OnInit() {
    MainFrame* mainframe = new MainFrame("GUI");
    mainframe->SetClientSize(1000, 600);
    mainframe->Center();

    mainframe->Show();
    return true;
}
wxIMPLEMENT_APP(App);

int f_counter = 0;
constexpr double MIN = 0.001;
constexpr double MAX = 0.999;
random_device rd;
default_random_engine eng(rd());
uniform_real_distribution<double> distr(MIN, MAX);

string filename;

wxBEGIN_EVENT_TABLE(MainFrame, wxFrame)
EVT_MENU(101, MainFrame::OnMenuOpen)
EVT_MENU(103, MainFrame::OnMenuExit)
EVT_MENU(110, MainFrame::ExportCSV)
EVT_CLOSE(MainFrame::OnClose)
EVT_MENU(130, MainFrame::OnHelp)
EVT_MENU(120, MainFrame::OnDefineRates)
EVT_MENU(121, MainFrame::OnSensitivity)
EVT_MENU(122, MainFrame::OnSense)
EVT_MENU(111, MainFrame::ExportSA)
wxEND_EVENT_TABLE()
```

```
int initial = 5;
int frg = 10;
double rate = 0.05;
int res = 1;
int sa_sol = 0;
int separatot = 1;
vector<double> s_rates;

MainFrame::MainFrame(const wxString& title) : wxFrame(nullptr,
wxID_ANY, title) {
    panel = new wxPanel(this);
    wxMenuBar* menub = new wxMenuBar();
    this->SetMenuBar(menub);
    menu->Append(101, "Open");
    menu->AppendSeparator();
    menu->Append(103, "Exit");

    menu1->Append(110, "Export to CSV");
    menu1->Enable(110, false);
    menu1->AppendSeparator();
    menu1->Append(111, "Export SA");
    menu1->Enable(111, false);
    menu2->Append(120, "Define Rates");
    menu2->Append(121, "Run Sensitivity Analysis");
    menu2->Append(122, "Plot Sensitivity Analysis");
    menu2->Enable(122, false);

    menu3->Append(130, "Steps");

    menub->Append(menu, "File");
    menub->Append(menu1, "Export Solution");
```

```
menub->Append(menu2, "Sensitivity Analysis");
menub->Append(menu3, "Help");

wxStaticText* initial_text = new wxStaticText(panel,
wxID_ANY, "Insert Size of Scouts:", wxPoint(20, 33),
wxDefaultSize);

wxSpinCtrl* initial_count = new wxSpinCtrl(panel,
wxID_ANY, "", wxPoint(210, 30), wxSize(80, -1),
wxSP_ARROW_KEYS | wxSP_WRAP, 1, 100, 5);

wxStaticText* initial_size_text = new wxStaticText(
panel, wxID_ANY, "Specify the size of Initial Population",
wxPoint(330, 33), wxDefaultSize);

wxFont font = initial_size_text->GetFont();
font.SetPointSize(10);
font.SetWeight(wxFONTWEIGHT_EXTRABOLD);
initial_size_text->SetFont(font);

initial_count->Bind(wxEVT_SPINCTRL, &MainFrame::
OnInitialSpin, this);

wxStaticText* forager_text = new wxStaticText(panel,
wxID_ANY, "Insert Number of Foragers:", wxPoint(20, 93),
wxDefaultSize, wxALIGN_LEFT);

wxSpinCtrl* forager_count = new wxSpinCtrl(panel,
wxID_ANY, "", wxPoint(210, 90), wxSize(80, -1),
wxSP_ARROW_KEYS | wxSP_WRAP, 10, 2000, 5);

wxStaticText* forager_size_text = new wxStaticText(panel,
```

```
wxID_ANY, "Specify the size of Local Search",
wxPoint(330, 93), wxDefaultSize);

    wxFont font1 = forager_size_text->GetFont();
    font.SetPointSize(10);
    font.SetWeight(wxFONTWEIGHT_EXTRABOLD);
    forager_size_text->SetFont(font);

    forager_count->Bind(wxEVT_SPINCTRL, &MainFrame::
OnForagerSPCTRL, this);

    wxStaticText* rate_text = new wxStaticText(panel,
wxID_ANY, "Insert rate:", wxPoint(20, 150),
wxDefaultSize, wxALIGN_CENTER_HORIZONTAL);
    wxSpinCtrlDouble* rate_count = new wxSpinCtrlDouble(
panel, wxID_ANY, "", wxPoint(210, 150), wxSize(80, -1),
wxSP_ARROW_KEYS | wxSP_WRAP, 0.01, 0.99, 0.05, 0.01);

    rate_count->Bind(wxEVT_SPINCTRLDOUBLE, &MainFrame::OnRateSpin,
this);

    wxStaticText* solution_text = new wxStaticText(panel,
wxID_ANY, "Plot graph for Solution:", wxPoint(20, 210),
wxDefaultSize, wxALIGN_CENTER_HORIZONTAL);
    solution_count = new wxSpinCtrl(panel, wxID_ANY, "",
wxPoint(210, 210), wxSize(80, -1), wxSP_ARROW_KEYS | wxSP_WRAP,
1, 1, 1);
    solution_count->Bind(wxEVT_SPINCTRL, &MainFrame::
OnSolutionSpin, this);

    wxStaticText* resource_text = new wxStaticText(panel,
wxID_ANY, "Plot graph for Resource:", wxPoint(20, 270),
```

```
wxDefaultSize, wxALIGN_CENTER_HORIZONTAL);  
    resource_count = new wxSpinCtrl(panel, wxID_ANY, "",  
wxPoint(210, 270), wxSize(80, -1), wxSP_ARROW_KEYS | wxSP_WRAP,  
1, 1, 1);  
    resource_count->Bind(wxEVT_SPINCTRL, &MainFrame::OnResourceSpin,  
this);  
  
    plot = new wxButton(panel, wxID_ANY, "Plot", wxPoint(330, 210),  
wxSize(85, 25));  
  
    plot->Enable(false);  
  
    wxStaticText* choice_text = new wxStaticText(panel,  
wxID_ANY, "Specify Separator:", wxPoint(20, 330),  
wxDefaultSize, wxALIGN_CENTER_HORIZONTAL);  
    wxArrayString choices;  
    choices.Add("Comma");  
    choices.Add("Semicolon");  
  
    choice = new wxChoice(panel, wxID_ANY, wxPoint(210, 330),  
wxDefaultSize, choices);  
    choice->Bind(wxEVT_CHOICE, &MainFrame::OnChoice, this);  
    choice->Select(0);  
  
    wxStaticText* local_text = new wxStaticText(panel,  
wxID_ANY, "Local search:", wxPoint(50, 510),  
wxDefaultSize, wxALIGN_CENTER_HORIZONTAL);  
    gauge = new wxGauge(panel, wxID_ANY, initial,  
wxPoint(50, 530), wxSize(650, -1), wxGA_SMOOTH,  
wxDefaultValidator);  
    gauge->SetValue(0);  
    wxStaticText* delay_text = new wxStaticText(panel,
```

```
wxID_ANY, "Activities Delay:", wxPoint(50, 550),
    wxDefaultSize, wxALIGN_CENTER_HORIZONTAL);
    Netgauge = new wxGauge(panel, wxID_ANY, initial,
wxPoint(50, 570), wxSize(650, -1), wxGA_SMOOTH,
    wxDefaultValidator);
    Netgauge->SetValue(0);

    label = new wxStaticText(panel, 110, "", wxPoint(720, 570),
wxDefaultSize, wxALIGN_CENTER_HORIZONTAL);
    label1 = new wxStaticText(panel, 110, "", wxPoint(720, 550),
wxDefaultSize, wxALIGN_CENTER_HORIZONTAL);

    run_button = new wxButton(panel, wxID_ANY, "Run",
wxPoint(50, 445), wxSize(85, 25));
    init = new wxButton(panel, wxID_ANY, "Initialize Solution",
wxPoint(50, 480), wxSize(130, 25));
    run_button->Bind(wxEVT_BUTTON, &MainFrame::OnRun, this);

    plot->Bind(wxEVT_BUTTON, &MainFrame::OnResourceGraph, this);
    init->Bind(wxEVT_BUTTON, &MainFrame::OnSolInit, this);

    init->Enable(false);
    run_button->Enable(false);
    menu2->Enable(121, false);
CreateStatusBar();
}

void MainFrame::OnSolutionSpin(wxSpinEvent& evt) {
    sa_sol = evt.GetValue() - 1;
    wxLogStatus("Solution set to %d", evt.GetValue());
}
```

```
void MainFrame::OnChoice(wxCommandEvent& evt)
{
    ch = evt.GetString();
    wxLogStatus("Choice set to %s", evt.GetString());
}

void MainFrame::ExportSA(wxCommandEvent& evt) {
    if (backgroundThread.joinable()) {this->backgroundThread.join();}
    string c1("Semicolon");
    string c2("Comma");
    if (ch == c2) {
        wxFileDialog dlg(this, "Save Sensitivity Analysis", "", "",
        "Save Files(*.csv) | *.csv | All files(*.*) | *.*", wxFD_SAVE);
        if (dlg.ShowModal() == wxID_OK)
        {
            wxFile file(dlg.GetPath(), wxFile::write);
            if (file.IsOpened())
            {
                file.Write("Date: ,");
                wxString tt;
                wxDateTime now = wxDateTime::Now();
                wxString today_str = now.FormatDate();
                wxString hour_str = now.FormatTime();
                file.Write(today_str);
                file.Write(",");
                file.Write(hour_str);
                file.Write(wxT("\n"));
                file.Write(filename);
                file.Write("\nTime: ,");

                wxString txt1;
                txt1.Printf(wxT("%.2lf [ms] || %.2f [s]"), dur, dur_sec);
```

```
file.Write(txt1);
file.Write("\nLoops:");
int ll;
int l = int(s_rates.size());
ll = l * (initial * frg) + f_counter * frg;
wxString loop_text;
loop_text.Printf(wxT("%d"), ll);
file.Write(loop_text);
for (size_t i = 0; i < s_rates.size(); i++)
{
    file.Write("\n\nRate: ");
    wxString txt;
    txt.Printf(wxT("%.3lf"), s_rates[i]);
    file.Write(txt);
    file.Write("\n");

    file.Write("Duration: ");
    wxString txt3;
    txt3.Printf(wxT("%d"), final[i].getDuration());
    file.Write(txt3);
    file.Write("\n");
    file.Write("NPVCum: ");
    wxString txt4;
    txt4.Printf(wxT(", %5.2lf"),
final[i].getTotalNPV());
    file.Write(txt4);
    file.Write("\n");
    file.Write("Act: ");
    file.Write(",");
    for (int j = 0; j < final[i].getSolSize(); j++)
    {
        wxString txt;
```



```
        txt.Printf(wxT("%d,"), final[i].getSol(j));
        file.Write(txt);
    }
    file.Write("\n");
    file.Write("Start: ");
    file.Write(",");
    for (int j = 0; j < final[i].getSolSize(); j++)
    {
        wxString txt;
        txt.Printf(wxT("%d, "),
final[i].getStart(final[i].getSol(j)));
        file.Write(txt);
    }
    file.Write("\n");
    file.Write("Finish: ");
    file.Write(",");
    for (int j = 0; j < final[i].getSolSize(); j++)
    {
        wxString txt;
        txt.Printf(wxT("%d, "),
final[i].getFinish(final[i].getSol(j)));
        file.Write(txt);
    }
    file.Write("\n");
    file.Write("NPV: ");
    file.Write(",");
    for (int j = 0; j < final[i].getSolSize(); j++)
    {
        wxString txt;
        txt.Printf(wxT("%5.2lf,"),
final[i].getNPV(final[i].getSol(j)));
        file.Write(txt);
    }
```

```
    }
    file.Write("\n\n");
    for (int j = 0; j < resourceNum; j++)
    {
        wxString txt;
        txt.Printf(wxT("R%d,"), j + 1);
        file.Write(txt);
        for (int k = 0; k < final[i].getDuration(); k++)
        {
            wxString txt1;
            txt1.Printf(wxT("%d,"),
final[i].PrintGantt(k, j));
            file.Write(txt1);
        }
        file.Write("\n");
    }
    file.Write("Time:");
    for (int k = 0; k < final[i].getDuration(); k++)
    {
        wxString txt1;
        txt1.Printf(wxT("%d,"), k + 1);
        file.Write(txt1);
    }
    file.Write("\n");
}
file.Close();
wxMessageBox(".csv File Created.");
}
}

else if(ch == c1)
```

```
{
    wxFileDialog dlg(this, "Save Sensitivity Analysis", "", "",
"Save Files(*.csv) | *.csv | All files(*.*) | *.*", wxFD_SAVE);
    if (dlg.ShowModal() == wxID_OK)
    {
        wxFile file(dlg.GetPath(), wxFile::write);
        if (file.IsOpened())
        {
            file.Write("Date: ");
            wxString tt;
            wxDateTime now = wxDateTime::Now();
            wxString today_str = now.FormatDate();
            wxString hour_str = now.FormatTime();
            file.Write(today_str);
            file.Write(";");
            file.Write(hour_str);
            file.Write(wxT("\n"));
            file.Write(filename);
            file.Write("\nTime: ");

            wxString txt1;
            txt1.Printf(wxT("%.2lf [ms] || %.2f [s]"), dur, dur_sec);
            file.Write(txt1);
            file.Write("\nLoops:");
            int ll;
            int l = int(s_rates.size());
            ll = l * (initial * frg) + f_counter * frg;
            wxString loop_text;
            loop_text.Printf(wxT("%d"), ll);
            file.Write(loop_text);
            for (size_t i = 0; i < s_rates.size(); i++)
            {
```

```
file.Write("\n\nRate: ");
wxString txt;
txt.Printf(wxT("%1.3lf"), s_rates[i]);
file.Write(txt);
file.Write("\n");

file.Write("Duration:  ");
wxString txt3;
txt3.Printf(wxT("%d"), final[i].getDuration());
file.Write(txt3);
file.Write("\n");
file.Write("NPVCum: ");
wxString txt4;
txt4.Printf(wxT("; %5.2lf"), final[i].getTotalNPV());
file.Write(txt4);
file.Write("\n");
file.Write("Act: ");
file.Write(";");
for (int j = 0; j < final[i].getSolSize(); j++)
{
    wxString txt;
    txt.Printf(wxT("%d;"), final[i].getSol(j));
    file.Write(txt);
}
file.Write("\n");
file.Write("Start: ");
file.Write(";");
for (int j = 0; j < final[i].getSolSize(); j++)
{
    wxString txt;
    txt.Printf(wxT("%d; "),
final[i].getStart(final[i].getSol(j))));
```

```
        file.Write(txt);
    }
    file.Write("\n");
    file.Write("Finish: ");
    file.Write(";");
    for (int j = 0; j < final[i].getSolSize(); j++)
    {
        wxString txt;
        txt.Printf(wxT("%d; "),
final[i].getFinish(final[i].getSol(j)));
        file.Write(txt);
    }
    file.Write("\n");
    file.Write("NPV: ");
    file.Write(";");
    for (int j = 0; j < final[i].getSolSize(); j++)
    {
        wxString txt;
        txt.Printf(wxT("%5.2lf;"),
final[i].getNPV(final[i].getSol(j)));
        file.Write(txt);
    }
    file.Write("\n\n");
    for (int j = 0; j < resourceNum; j++)
    {
        wxString txt;
        txt.Printf(wxT("R%d;"), j + 1);
        file.Write(txt);
        for (int k = 0; k < final[i].getDuration(); k++)
        {
            wxString txt1;
            txt1.Printf(wxT("%d;"),
```

```
final[i].PrintGantt(k, j));
        file.Write(txt1);
    }
    file.Write("\n");
}
file.Write("Time:");
for (int k = 0; k < final[i].getDuration(); k++)
{
    wxString txt1;
    txt1.Printf(wxT("%d"), k + 1);
    file.Write(txt1);
}
file.Write("\n");
}
file.Close();
wxMessageBox(".csv File Created.");
}
}
return;
}

void MainFrame::OnDefineRates(wxCommandEvent& evt)
{
    NumberDlg dlg;
    RateDlg rdlg;
    int t = 0;
    while (t < 2)
    {
        dlg.ShowModal();
        t = dlg.GetNumber();
        wxLogStatus("Rate number: %d", t);
    }
}
```

```
        if (t <= 1)
        {
            if (wxMessageBox(_("Invalid value, try again?"),
                _("Question"), wxYES_NO) != wxYES)
            {
                break;
            }
        }
    }
    if (t > 1)
    {
        s_rates.clear();
        int flag = 1;
        double val = 0.0;
        int co = 0;
        while (co < t)
        {
            if (rdlg.ShowModal() == wxID_CANCEL)
            {
                flag = 0;
                break;
            }
            if (rdlg.GetNumber() > 0.00001)
            {
                s_rates.push_back(rdlg.GetNumber());
                co++;
            }
        }
        if (flag == 1) {
            f_counter = 0;
            menu2->Enable(121, true);
        }
    }
```

```
    }
    sort(s_rates.begin(), s_rates.end());
    wxString Foobar;
}

void MainFrame::OnSense(wxCommandEvent& evt)
{
    wxFrame* chart_fr = new wxFrame(this, wxID_ANY,
    "Chart Frame", wxPoint(1000, 300), wxSize(600, 500));
    gchart = new ChartControlSA(chart_fr, wxID_ANY,
    wxDefaultPosition, wxDefaultSize);
    auto text = wxString::Format("NPV - Rate Chart");
    gchart->title = text;
    gchart->lv = s_rates[0];
    gchart->hv = *std::max_element(s_rates.begin(), s_rates.end());
    gchart->number = s_rates.size();
    for (int i = 0; i < s_rates.size(); i++)
    {
        gchart->rate_values.push_back(s_rates[i]);
        gchart->npv_values.push_back(final[i].getTotalNPV());
    }

    chart_fr->Show();
}

void MainFrame::OnResourceGraph(wxCommandEvent& evt)
{
    wxFrame* chart_fr = new wxFrame(this, wxID_ANY, "Chart Frame",
    wxPoint(1000, 300), wxSize(600, 500));
    chart = new ChartControl(chart_fr, wxID_ANY, wxDefaultPosition,
    wxDefaultSize);
    auto text = wxString::Format("Resource %d Usage Chart,
```



```
Duration = %d, Solution %d, NPV = %.2f", res,
final[sa_sol].getDuration(), sa_sol + 1,
final[sa_sol].getTotalNPV());
    chart->title = text;
    chart->max = resourceCap[res - 1];
    chart->min = 0.0;
    chart->time_value = final[sa_sol].getDuration();

    for (int i = 0; i < final[sa_sol].getGanttSize(); i++)
        chart->values.push_back(final[sa_sol].PrintGantt(i, res - 1));

    chart_fr->Show();
}

void MainFrame::OnSensitivity(wxCommandEvent& evt)
{
    run_button->Enable(false);
    gauge->SetRange(initial);
    Netgauge->SetRange(initial);
    if (!this->processing)
    {
        this->processing = true;
        const auto f = [this]()
        {
            auto startTime = high_resolution_clock::now();

            double X;
            vector<double> random; random.clear();
            vector<int> solution; solution.clear();
            data();
            resource_count->SetRange(1, resourceNum);
            CreatePredecessors();
```

```
createCF();
vector<bool> dummy_status; dummy_status.clear();
vector<int> dummy_start; dummy_start.clear();
vector<int> dummy_finish; dummy_finish.clear();
vector<double> dummy_npv; dummy_npv.clear();
vector<bool> dummy_check; dummy_check.clear();
for (int i = 0; i < inputs; i++)
{
    if (i == 0) dummy_status.push_back(true);
    else dummy_status.push_back(false);
    dummy_start.push_back(0);
    dummy_finish.push_back(0);
    dummy_npv.push_back(0.0);

    dummy_check.push_back(false);
}
int rate_size = s_rates.size();
final = new Opt[rate_size];
int position = 0;
for (size_t s = 0; s < rate_size; s++)
{
    rate = s_rates[s];
    int nn = 0;
    scout_bees = new In_Pop[initial];
    for (int i = 0; i < initial; i++)
        scout_bees[i] = In_Pop();
    while (nn < initial)
    {
        random.clear();
        random.push_back(1.0);
        for (int i = 0; i < inputs - 1; i++)
        {
```

```
        X = distr(eng);
        random.push_back(X);
    }
    random.push_back(0.0);
    TotalNPV = 0.0;
    SSGS_Forward(random, nn, 0, 0);
    for (int i = 0; i < inputs; i++)
    {
        element[i].setNPV(element[i].getCashFlow(), rate,
element[i].getFinish());
        TotalNPV += element[i].getNPV();
    }
    scout_bees[nn] = In_Pop(scout_bees[nn].GetSolution(),
TotalNPV, scout_bees[nn].GetSolutionDuration(),
scout_bees[nn].GetPrio());
    nn++;
    solution.clear();
}

forager_bees = new Foragers * *[initial];
best_bees = new Best[initial];
elite_bees = new Elite[initial];
for (int i = 0; i < initial; i++)
{
    forager_bees[i] = new Foragers * [frg];
    for (int j = 0; j < frg; j++) {
        forager_bees[i][j] = new Foragers();
    }
}
for (int i = 0; i < initial; i++)
{
    Local_Search(frg, i, dummy_status, dummy_start,
```

```

dummy_finish, dummy_npv, dummy_check);
        wxGetApp().CallAfter([this, i]()
            {this->gauge->SetValue(i + 1); });
    }

    delete[] scout_bees;
    delete[] forager_bees;
    for (int n = 0; n < initial; n++)
    {
        wxGetApp().CallAfter([this, n]()
            {this->Netgauge->SetValue(n + 1); });
        int del = best_bees[n].getFinish(inputs - 1) +
int(best_bees[n].getFinish(inputs - 1) * 0.15);
        Network_Delay(del, n);
        double t = 0.0;
        for (int i = 0; i < inputs; i++)
        {
            t += elite_bees[n].getNPV(i);
        }
        elite_bees[n].setTotalNPV(t);
        t = 0.0;
        Schedule_Delay(n, del);
        for (int i = 0; i < inputs; i++)
        {
            t += elite_bees[n].getNPV(i);
        }
        elite_bees[n].setTotalNPV(t);
    }
    Check_Schedule(initial);
    double best_npv = best_bees[0].getTotalNPV();
    int best_time = 10000;
    for (int i = 0; i < initial; i++)

```

```

        {
            if (elite_bees[i].getTotalNPV() > best_npv)
            {
                best_npv = elite_bees[i].getTotalNPV();
                first = i;
            }
        }
        final[position] = Opt(elite_bees[first].getSolVec(),
elite_bees[first].getDuration(), elite_bees[first].getGannt(),
        elite_bees[first].getTotalNPV(), elite_bees[first].getNPVVec(),
        elite_bees[first].getStartVec(), elite_bees[first].getFinishVec());

        delete[] best_bees;
        delete[] elite_bees;
        position++;
    }
    menu2->Enable(122, true);
    menu1->Enable(111, true);
    plot->Enable(true);
    init->Enable(true);
    wxMessageBox("Solution Found");
    auto stopTime = high_resolution_clock::now();
    auto duration = stopTime - startTime;
    dur = std::chrono::duration<double,
std::milli>(duration).count();
    dur_sec = std::chrono::duration<double>(duration).count();
    this->label1->SetLabelText(wxString::Format("
Processing time: %.2f [ms]", std::chrono::duration<double,
std::milli>(duration).count()));

    this->label->SetLabelText(wxString::Format("
Processing time: %.2f [s]",

```

```

        std::chrono::duration<double>(duration).count());
        solution_count->SetRange(1, rate_size);
    };
    this->backgroundThread = thread{ f };
}

}

void MainFrame::OnHelp(wxCommandEvent& evt){
    wxFrame* help_fr = new wxFrame(this, wxID_ANY, "Steps",
wxPoint(1000, 300), wxSize(600, 500));
    wxPanel* panel1 = new wxPanel(help_fr);
    wxStaticText* test_txt = new wxStaticText(panel1, wxID_ANY,
"Step 1: Specify the size of the Initial Population
(Scout Bees).", wxPoint(15, 20), wxDefaultSize);
    wxStaticText* test_txt1 = new wxStaticText(panel1, wxID_ANY,
"Step 2: Specify the size of the Forager Bees
(loops per Scout Bee).", wxPoint(15, 50), wxDefaultSize);
    wxStaticText* note = new wxStaticText(panel1, wxID_ANY,
"Important: The pool of solutions will be at least the size
of Scout Bees * Forager Bees.", wxPoint(15, 70), wxDefaultSize);
    wxFont font = note->GetFont();
    font.SetPointSize(9);
    font.SetWeight(wxFONTWEIGHT_EXTRABOLD);
    note->SetFont(font);
    wxStaticText* test_txt2 = new wxStaticText(panel1, wxID_ANY,
"Step 3: Specify rate.", wxPoint(15, 95), wxDefaultSize);
    wxStaticText* test_txt3 = new wxStaticText(panel1, wxID_ANY,
"Step 4: Open .RCP or .txt file (", wxPoint(15, 120), wxDefaultSize);
    wxHyperlinkCtrl* thyper = new wxHyperlinkCtrl(panel1, wxID_ANY,
" Patterson ", "http://www.p2engine.com/p2reader
/patterson_format", wxPoint(171, 120), wxDefaultSize);
    wxStaticText* test_txt3_1 = new wxStaticText(panel1, wxID_ANY,

```

```
"format ONLY).", wxPoint(232, 120), wxDefaultSize);
    wxStaticText* test_txt4 = new wxStaticText(panel1, wxID_ANY,
"Step 5: Run algorithm.", wxPoint(15, 145), wxDefaultSize);
    wxStaticText* test_txt5 = new wxStaticText(panel1, wxID_ANY,
"Step 6: Export Solution.", wxPoint(15, 170), wxDefaultSize);
    help_fr->Show();
}

void MainFrame::OnRun(wxCommandEvent& evt)
{
    f_counter = 0;
    run_button->Enable(false);
    wxYield();
    gauge->SetRange(initial);
    Netgauge->SetRange(initial);
    if (!this->processing)
    {
        this->processing = true;
        const auto f = [this]()
        {
            auto startTime = high_resolution_clock::now();
            vector<double> random; random.clear();
            vector<int> solution; solution.clear();
            data();
            resource_count->SetRange(1, resourceNum);
            CreatePredecessors();
            createCF();
            vector<bool> dummy_status; dummy_status.clear();
            vector<int> dummy_start; dummy_start.clear();
            vector<int> dummy_finish; dummy_finish.clear();
            vector<double> dummy_npv; dummy_npv.clear();
            vector<bool> dummy_check; dummy_check.clear();
```

```

for (int i = 0; i < inputs; i++)
{
    if (i == 0) dummy_status.push_back(true);
    else dummy_status.push_back(false);
    dummy_start.push_back(0);
    dummy_finish.push_back(0);
    dummy_npv.push_back(0.0);
    dummy_check.push_back(false);
}

scout_bees = new In_Pop[initial];
for (int i = 0; i < initial; i++)
    scout_bees[i] = In_Pop();

Initial_Population(initial, random, solution);

forager_bees = new Foragers * *[initial];
best_bees = new Best[initial];
elite_bees = new Elite[initial];
for (int i = 0; i < initial; i++)
{
    forager_bees[i] = new Foragers * [frg];
    for (int j = 0; j < frg; j++) {
        forager_bees[i][j] = new Foragers();
    }
}
for (int i = 0; i < initial; i++)
{
    Local_Search(frg, i, dummy_status, dummy_start,
dummy_finish, dummy_npv, dummy_check);
    wxGetApp().CallAfter([this, i]()
        {this->gauge->SetValue(i + 1); });
}

```



```
}

delete[] scout_bees;
delete[] forager_bees;
for (int n = 0; n < initial; n++)
{
    wxGetApp().CallAfter([this, n]()
        {this->Netgauge->SetValue(n + 1); });
    int del = best_bees[n].getFinish(inputs - 1) +
int(best_bees[n].getFinish(inputs - 1) * 0.15);
    Network_Delay(del, n);
    double t = 0.0;
    for (int i = 0; i < inputs; i++)
    {
        t += elite_bees[n].getNPV(i);
    }
    elite_bees[n].setTotalNPV(t);
    t = 0.0;
    Schedule_Delay(n, del);
    for (int i = 0; i < inputs; i++)
    {
        t += elite_bees[n].getNPV(i);
    }
    elite_bees[n].setTotalNPV(t);
}
Check_Schedule(initial);
final = new Opt[1];
double best_npv = best_bees[0].getTotalNPV();
int best_time = 10000;
for (int i = 0; i < initial; i++)
{
    if (elite_bees[i].getTotalNPV() > best_npv)
```

```

        {
            best_npv = elite_bees[i].getTotalNPV();
            first = i;
        }
    }
    final[0] = Opt(elite_bees[first].getSolVec(),
elite_bees[first].getDuration(),
elite_bees[first].getGantt(),
elite_bees[first].getTotalNPV(),
            elite_bees[first].getNPVVec(),
elite_bees[first].getStartVec(),
elite_bees[first].getFinishVec());
    delete[] best_bees;
    delete[] elite_bees;

    menu1->Enable(110, true);
    plot->Enable(true);
    init->Enable(true);
    wxMessageBox("Solution Found");
    auto stopTime = high_resolution_clock::now();
    auto duration = stopTime - startTime;
    dur = std::chrono::duration<double,
std::milli>(duration).count();
    dur_sec = std::chrono::duration<double>(duration).count();
    this->label1->SetLabelText(wxString::Format(
"Processing time: %.2f [ms]", std::chrono::
duration<double, std::milli>(duration).count()));
    this->label->SetLabelText(wxString::Format(
"Processing time: %.2f [s]", std::chrono::
duration<double>(duration).count()));
    };
    this->backgroundThread = thread{ f };

```

```
    }  
}  
  
void MainFrame::OnResourceSpin(wxSpinEvent& evt)  
{  
    res = evt.GetValue();  
    wxLogStatus("Resource set to %d", evt.GetValue());  
}  
  
void MainFrame::OnSolInit(wxCommandEvent& evt) {  
    delete[] final;  
    delete[] element;  
    s_rates.clear();  
    menu2->Enable(121, false);  
    menu2->Enable(122, false);  
    init->Enable(false);  
    run_button->Enable(true);  
    menu1->Enable(110, false);  
    menu1->Enable(111, false);  
    plot->Enable(false);  
    gauge->SetValue(0);  
    Netgauge->SetValue(0);  
    this->label1->SetLabelText(wxString::Format(""));  
    this->label->SetLabelText(wxString::Format(""));  
    this->processing = false;  
    if (backgroundThread.joinable()) {this->backgroundThread.join();}  
    wxMessageBox("Solution Deleted.");  
}  
  
void MainFrame::ExportCSV(wxCommandEvent& evt) {  
    if (backgroundThread.joinable()) { this->backgroundThread.join();}  
    string c1("Semicolon");
```

```
string c2("Comma");
if (ch == c2) {
    wxFileDialog dlg(this, "Save Best Solutions", "", "",
"Save Files(*.csv) | *.csv | All files(*.*) | *.*",
wxFD_SAVE);
    if (dlg.ShowModal() == wxID_OK)
    {
        wxFile file(dlg.GetPath(), wxFile::write);
        if (file.IsOpened())
        {
            file.Write("Date: ,");
            wxString tt;
            wxDateTime now = wxDateTime::Now();
            wxString today_str = now.FormatDate();
            wxString hour_str = now.FormatTime();
            file.Write(today_str);
            file.Write(",");
            file.Write(hour_str);
            file.Write(wxT("\n"));
            file.Write(filename);
            file.Write("\nTime: ,");

            wxString txt1;
            txt1.Printf(wxT("%.2lf [ms] || %.2f [s]"), dur,
dur_sec);

            file.Write(txt1);
            file.Write("\nLoops: ,");
            int ll;
            ll = initial * frg + f_counter * frg;
            wxString loop_text;
            loop_text.Printf(wxT("%d"), ll);
            file.Write(loop_text);
```

```
file.Write("\nRate: ,");
wxString txt;
txt.Printf(wxT("%1.3lf"), rate);
file.Write(txt);
file.Write("\n");
file.Write(wxT("Solution: ,"));
wxString txt2;
txt2.Printf(wxT("%d"), first);
file.Write(txt2);
file.Write("\n");

file.Write("Duration: , ");
wxString txt3;
txt3.Printf(wxT("%d"), final[0].getDuration());
file.Write(txt3);
file.Write("\n");
file.Write("NPVCum: ");
wxString txt4;
txt4.Printf(wxT(", %5.2lf"), final[0].getTotalNPV());
file.Write(txt4);
file.Write("\n\n");
file.Write("Act: ");
file.Write(",");
for (int j = 0; j < final[0].getSolSize(); j++)
{
    wxString txt;
    txt.Printf(wxT("%d,"), final[0].getSol(j));
    file.Write(txt);
}
file.Write("\n");
file.Write("Start: ");
```

```
file.Write(",");
for (int j = 0; j < final[0].getSolSize(); j++)
{
    wxString txt;
    txt.Printf(wxT("%d, "),
final[0].getStart(final[0].getSol(j)));
    file.Write(txt);
}
file.Write("\n");
file.Write("Finish: ");
file.Write(",");
for (int j = 0; j < final[0].getSolSize(); j++)
{
    wxString txt;
    txt.Printf(wxT("%d, "),
final[0].getFinish(final[0].getSol(j)));
    file.Write(txt);
}
file.Write("\n");
file.Write("NPV: ");
file.Write(",");
for (int j = 0; j < final[0].getSolSize(); j++)
{
    wxString txt;
    txt.Printf(wxT("%5.2lf,"),
final[0].getNPV(final[0].getSol(j)));
    file.Write(txt);
}
file.Write("\n\n");
for (int j = 0; j < resourceNum; j++)
{
    wxString txt;
```

```
        txt.Printf(wxT("R%d,"), j + 1);
        file.Write(txt);
        for (int i = 0; i < final[0].getDuration(); i++)
        {
            wxString txt1;
            txt1.Printf(wxT("%d,"), final[0].PrintGantt(i,j));
            file.Write(txt1);
        }
        file.Write("\n");
    }
    file.Write("Time:");
    for (int i = 0; i < final[0].getDuration(); i++)
    {
        wxString txt1;
        txt1.Printf(wxT("%d,"), i + 1);
        file.Write(txt1);
    }
    file.Close();
    wxMessageBox(".csv File Created.");
}

}

else if (ch == c1)
{
    wxFileDialog dlg(this, "Save Best Solutions", "", "",
"Save Files(*.csv) | *.csv | All files(*.*) | *.*", wxFD_SAVE);
    if (dlg.ShowModal() == wxID_OK)
    {
        wxFile file(dlg.GetPath(), wxFile::write);
        if (file.IsOpened())
        {
```

```
file.Write("Date: ");
wxString tt;
wxDateTime now = wxDateTime::Now();
wxString today_str = now.FormatDate();
wxString hour_str = now.FormatTime();
file.Write(today_str);
file.Write(";");
file.Write(hour_str);
file.Write(wxT("\n"));
file.Write(filename);
file.Write("\nTime: ");

wxString txt1;
txt1.Printf(wxT("%.2lf [ms] || %.2f [s]"), dur,
dur_sec);

file.Write(txt1);
file.Write("\nLoops:");
int ll;
ll = initial * frg + f_counter * frg;
wxString loop_text;
loop_text.Printf(wxT("%d"), ll);
file.Write(loop_text);

file.Write("\nRate: ");
wxString txt;
txt.Printf(wxT("%.3lf"), rate);
file.Write(txt);
file.Write("\n");
file.Write(wxT("Solution: "));
wxString txt2;
txt2.Printf(wxT("%d"), first);
file.Write(txt2);
```



```
file.Write("\n");

file.Write("Duration:  ");
wxString txt3;
txt3.Printf(wxT("%d"), final[0].getDuration());
file.Write(txt3);
file.Write("\n");
file.Write("NPVCum: ");
wxString txt4;
txt4.Printf(wxT("; %5.2lf"), final[0].getTotalNPV());
file.Write(txt4);
file.Write("\n\n");
file.Write("Act: ");
file.Write(";");
for (int j = 0; j < final[0].getSolSize(); j++)
{
    wxString txt;
    txt.Printf(wxT("%d;"), final[0].getSol(j));
    file.Write(txt);
}
file.Write("\n");
file.Write("Start: ");
file.Write(";");
for (int j = 0; j < final[0].getSolSize(); j++)
{
    wxString txt;
    txt.Printf(wxT("%d; "),
final[0].getStart(final[0].getSol(j)));
    file.Write(txt);
}
file.Write("\n");
file.Write("Finish: ");
```

```
file.Write(";");
for (int j = 0; j < final[0].getSolSize(); j++)
{
    wxString txt;
    txt.Printf(wxT("%d; "),
final[0].getFinish(final[0].getSol(j)));
    file.Write(txt);
}
file.Write("\n");
file.Write("NPV: ");
file.Write(";");
for (int j = 0; j < final[0].getSolSize(); j++)
{
    wxString txt;
    txt.Printf(wxT("%5.2lf;"),
final[0].getNPV(final[0].getSol(j)));
    file.Write(txt);
}
file.Write("\n\n");
for (int j = 0; j < resourceNum; j++)
{
    wxString txt;
    txt.Printf(wxT("R%d;"), j + 1);
    file.Write(txt);
    for (int i = 0; i < final[0].getDuration(); i++)
    {
        wxString txt1;
        txt1.Printf(wxT("%d;"), final[0].PrintGannt(i, j));
        file.Write(txt1);
    }
    file.Write("\n");
}
```

```
        file.Write("Time:");
        for (int i = 0; i < final[0].getDuration(); i++)
        {
            wxString txt1;
            txt1.Printf(wxT("%d"), i + 1);
            file.Write(txt1);
        }
        file.Close();
        wxMessageBox(".csv File Created.");
        file.Close();
        wxMessageBox(".csv File Created.");
    }
}

return;
}

void MainFrame::OnInitialSpin(wxSpinEvent& evt) {
    initial = evt.GetValue();
    wxLogStatus("Initial Population Size set to %d", evt.GetValue());
}

void MainFrame::OnForagerSPCTRL(wxSpinEvent& evt) {
    frg = evt.GetValue();
    wxLogStatus("Foragers set to %d", evt.GetValue());
}

void MainFrame::OnRateSpin(wxSpinDoubleEvent& evt) {
    rate = evt.GetValue();
    wxLogStatus("Rate set to %2.3f", evt.GetValue());
}
```

```
void MainFrame::OnClose(wxCloseEvent& evt) {
    if (this->processing)
    {
        evt.Veto();
        if (backgroundThread.joinable()) this->backgroundThread.join();
        this->quitRequested = true;
    }
    else
    {
        if (backgroundThread.joinable()) this->backgroundThread.join();
        this->Destroy();
    }
    evt.Skip();
}

void MainFrame::OnMenuOpen(wxCommandEvent& evt) {
    wxFileDialog fileDlg(this, _("Choose file"), wxEmptyString,
    wxEmptyString,
    _("RCP file|*.RCP|TXT file|*.txt|All files|*..*"));

    if (fileDlg.ShowModal() == wxID_OK) {
        wxString path = fileDlg.GetPath();
        filename = path;
    }
    run_button->Enable(true);
}

void MainFrame::OnMenuExit(wxCommandEvent& evt) {
    Close();
    evt.Skip();
}
```

```
}
```

```
void MainFrame::data()
{
    ifstream infile(filename);
    string ch;
    int positive = 0;
    int line = 0;
    int act = 0, duration, * demand, numofSucc,
    * successors, start = 0;

    vector<int> activity;
    vector<int> act_duration;
    vector<int> in_vec_demand;
    vector<vector<int>> resource_array;
    vector<int> in_succ;
    vector<vector<int>> successors_array;
    vector<int> pred;
    vector<double> cash_flow;

    activity.clear();
    act_duration.clear();
    in_vec_demand.clear();
    resource_array.clear();
    in_succ.clear();
    successors_array.clear();
    pred.clear();
    cash_flow.clear();
    rcost.clear();

    if (!infile)
```

```
{
    exit(1);
}
else
{

    fstream infile(filename, fstream::in);
    while (!infile.eof())
    {
        if (line == 0)
        {
            infile >> inputs >> resourceNum;
            resourceCap = new int[resourceNum];
        }

        if (line == 1)
        {
            for (int i = 0; i < resourceNum; i++)
            {
                infile >> resourceCap[i];
            }
        }

        if (line > 1)
        {
            in_vec_demand.clear();
            in_succ.clear();

            infile >> duration;
            act_duration.push_back(duration);

            demand = new int[resourceNum];
```

```
        for (int i = 0; i < resourceNum; i++)
        {
            infile >> demand[i];
            in_vec_demand.push_back(demand[i]);
        }
        resource_array.push_back(in_vec_demand);
        delete[] demand;

        infile >> numofSucc;
        successors = new int[numofSucc];
        for (int i = 0; i < numofSucc; i++)
        {
            infile >> successors[i];
            successors[i]--;
            in_succ.push_back(successors[i]);
        }
        successors_array.push_back(in_succ);
        delete[] successors;

        activity.push_back(act);
        act++;
    }
    line++;
}

infile.close();
element = new rcpsp[inputs];
for (int i = 0; i < inputs; i++)
    element[i] = rcpsp();
for (int i = 0; i < inputs; i++)
{
    vector<int> rdemand;
```

```

        vector<int> succ;
        for (size_t j = 0; j < resource_array[i].size(); j++)
            rdemand.push_back(resource_array[i][j]);
        for (size_t k = 0; k < successors_array[i].size(); k++)
            succ.push_back(successors_array[i][k]);
        element[i] = rcpsp(activity[i], act_duration[i], rdemand,
int(successors_array[i].size()), succ, pred, 0.0, start,
false, 0.0);
    }
    for (int i = 0; i < inputs; i++)
        MaxTime += element[i].getDuration();
}

void MainFrame::CreatePredecessors()
{
    for (int i = 0; i < inputs; i++)
        for (int j = 0; j < element[i].getNumOfSucc(); j++)
            element[element[i].getSuccessor(j)].PredecessorVector(
element[i].getActivity());
}

int MainFrame::createCF()
{
    int count = 0;
    for (int i = 0; i < resourceNum; i++)
rcost.push_back(ResourceCost);
    double v;
    double vv = 0.0;
    for (int i = 1; i < inputs - 1; i++)
    {
        v = 0.0;
        for (int j = 0; j < rcost.size(); j++)

```



```
        v += element[i].getDemand(j) * rcost[j];
        vv = ConstantV - v;
        element[i].setCF(vv);
        if (vv > 0.0) count++;
    }
    return count;
}

int MainFrame::Early_Start(int act, int trigger, int s_index, int frg)
{
    int max = 0;
    if (trigger == 0)
    {
        for (int i = 0; i < element[act].getNumOfPred(); i++)
            if (element[element[act].getPredecessor(i)].getFinish() > max)
                max = element[element[act].getPredecessor(i)].getFinish();
    }
    else if (trigger == 1)
    {
        for (int i = 0; i < element[act].getNumOfPred(); i++)
            if (forager_bees[s_index][frg]->getStart(
element[act].getPredecessor(i)) + element[
element[act].getPredecessor(i)].getDuration() > max)
                max = forager_bees[s_index][frg]->getStart(
element[act].getPredecessor(i)) + element[
element[act].getPredecessor(i)].getDuration();
    }
    return max;
}

int MainFrame::Latest_Finish(int act)
{

```

```

    int max = 0;
    for (int i = 0; i < element[act].getNumOfSucc(); i++)
        if (element[element[act].getSuccessor(i)].getFinish() > max)
            max = element[element[act].getSuccessor(i)].getFinish();
    return max;
}

vector<int> MainFrame::Forward_Activity_Check(vector<int> next_act,
int* act, vector<double> random, int trigger, int s_index, int frg)
{
    double max = 0.0;
    vector<int> temp;
    if (trigger == 0)
    {
        for (size_t i = 0; i < next_act.size(); i++)
        {
            int count = 0;
            for (int j = 0; j < element[next_act[i]].getNumOfPred();
j++)
            {
                if (element[element[next_act[i]].getPredecessor(j)]
.getStatus() == true)
                // check if all the preds are checked
                    count++;
            }
            if ((count == element[next_act[i]].getNumOfPred()) &&
(random[next_act[i]] > max))
            {
                max = random[next_act[i]];
                *act = next_act[i];
            }
        }
    }
}

```

```

        element[*act].setStatus(true);

        for (size_t i = 0; i < next_act.size(); i++)
        {
            if ((element[next_act[i]].getStatus() == false) && (
find(temp.begin(), temp.end(), next_act[i]) == temp.end()))
                temp.push_back(next_act[i]);

            else if (element[next_act[i]].getStatus() == true)
                for (int j = 0; j < element[
next_act[i]].getNumOfSucc(); j++)
                {
                    if (find(temp.begin(), temp.end(), element[
next_act[i]].getSuccessor(j)) == temp.end())
                        temp.push_back(element[next_act[i]].
getSuccessor(j));
                }
        }
    }
    else if (trigger == 1)
    {
        for (size_t i = 0; i < next_act.size(); i++)
        {
            int count = 0;
            for (int j = 0; j < element[next_act[i]].getNumOfPred();
j++)
            {
                if (forager_bees[s_index][frg]->getStatus(
element[next_act[i]].getPredecessor(j)) == true)
                    count++;
            }
            if ((count == element[next_act[i]].getNumOfPred())

```

```

    && (random[next_act[i]] > max))
    {
        max = random[next_act[i]];
        *act = next_act[i];
    }
}
forager_bees[s_index][frg]->setStatus(*act, true);
for (size_t i = 0; i < next_act.size(); i++)
{
    if ((forager_bees[s_index][frg]->getStatus(
next_act[i]) == false) && (find(temp.begin(),
temp.end(), next_act[i]) == temp.end()))
        temp.push_back(next_act[i]);
    else if (forager_bees[s_index][frg]->getStatus(
next_act[i]) == true)
        for (int j = 0; j < element[next_act[i]].
getNumOfSucc(); j++)
        {
            if (find(temp.begin(), temp.end(), element[
next_act[i]].getSuccessor(j)) == temp.end())
                temp.push_back(element[next_act[i]].
getSuccessor(j));
        }
}
}
return temp;
}

vector<int> MainFrame:: Forward_Activity_List(vector<double> random,
int trigger, int s_index, int frg)
{
    vector<int> solution; solution.clear(); solution.push_back(0);

```

```
vector<int> next_act;    // candidate activities
int current = 0;        // prwta mpainei h 0 profanws
int next;
int act;
int check = 1;

if (trigger == 0)
{
    double max = 0.0;
    int suc = 0;
    for (int i = 0; i < element[current].getNumOfSucc(); i++)
        if (random[element[current].getSuccessor(i)] > max)
        {
            max = random[element[current].getSuccessor(i)];
            suc = element[current].getSuccessor(i);
        }
    next = suc;
    element[next].setStatus(true);

    for (int i = 0; i < element[current].getNumOfSucc(); i++)
        if (element[element[current].getSuccessor(i)].
getStatus() == false)
            next_act.push_back(element[current].getSuccessor(i));

    for (int i = 0; i < element[next].getNumOfSucc(); i++)
        next_act.push_back(element[next].getSuccessor(i));

    solution.push_back(next); check++;
    while (check < inputs)
    {
        next_act = Forward_Activity_Check(next_act, &act, random,
trigger, s_index, frg);
```

```
        solution.push_back(act);
        check++;
    }
}
else if (trigger == 1)
{
    double max = 0.0;
    int suc = 0;
    for (int i = 0; i < element[current].getNumOfSucc(); i++)
        if (random[element[current].getSuccessor(i)] > max)
        {
            max = random[element[current].getSuccessor(i)];
            suc = element[current].getSuccessor(i);
        }
    next = suc;
    forager_bees[s_index][frg]->setStatus(next, true);

    for (int i = 0; i < element[current].getNumOfSucc(); i++)
        if (forager_bees[s_index][frg]->getStatus(element[
current].getSuccessor(i)) == false)
            next_act.push_back(element[current].getSuccessor(i));
    for (int i = 0; i < element[next].getNumOfSucc(); i++)
        next_act.push_back(element[next].getSuccessor(i));

    solution.push_back(next); check++;
    while (check < inputs)
    {
        next_act = Forward_Activity_Check(next_act, &act, random,
trigger, s_index, frg);
        solution.push_back(act);
        check++;
    }
}
```

```
    }
    return solution;
}

vector<vector<int>> MainFrame::Forward_Resource_Check(
vector<vector<int>> GanttR, vector<int> solution, double* npv,
int* finish, int trigger, int s_index, int frg)
{
    int counter = 1;
    int count = 0;
    int act = 1;
    double sum = 0.0;
    while (counter < solution.size())
    {
        int ES = Early_Start(solution[act], trigger, s_index, frg);
        int dur = element[solution[act]].getDuration();
        int i = ES;
        while (i < ES + dur)
        {
            for (int j = 0; j < resourceNum; j++)
            {
                if (GanttR[i][j] + element[solution[act]].
getDemand(j) > resourceCap[j])
                {
                    ES++; i = ES - 1; count = 0;
                    break;
                }
                else count++;
            }
            if (count / resourceNum == dur)
            {
                for (int k = ES; k < i + 1; k++)
```

```

        for (int j = 0; j < resourceNum; j++)
            GanttR[k][j] += element[solution[act]].
getDemand(j);
    }
    i++;
}
if (trigger == 0) element[solution[act]].setStart(ES);
else if (trigger == 1)
{
    forager_bees[s_index][frg]->setStart(solution[act], ES);
    forager_bees[s_index][frg]->setFinish(solution[act],
element[solution[act]].getDuration());
    double val = element[solution[act]].getCashFlow() /
(pow(1.0 + rate, forager_bees[s_index][frg]->
getFinish(solution[act])));
    forager_bees[s_index][frg]->setNPV(solution[act], val);
    sum += val;
}
act++;
count = 0;
counter++;
}
if (trigger == 0) {*finish = element[inputs - 1].getFinish();}
else if (trigger == 1)
{
    *finish = forager_bees[s_index][frg]->getStart(inputs - 1);
    *npv = sum;
}
return GanttR;
}

```

```

void MainFrame::SSGS_Forward(vector<double> random, int s_index,

```



```
int frg, int trigger)
{
    vector<int> Solution; Solution.clear();
    vector<vector<int>> GanttR; GanttR.clear();
    vector<int> vec;
    int finish;
    double npv;

    for (int i = 0; i < resourceNum; i++) vec.push_back(0);
    for (int i = 0; i < MaxTime; i++) GanttR.push_back(vec);

    for (int i = 0; i < MaxTime; i++)
        for (int j = 0; j < resourceNum; j++)
            GanttR[i][j] = 0;
    if (trigger == 0)
    {
        element[0].setStatus(true);
        for (int i = 1; i < inputs; i++)
            element[i].setStatus(false);
    }
    Solution = Forward_Activity_List(random, trigger, s_index, frg);
    GanttR = Forward_Resource_Check(GanttR, Solution, &npv, &finish,
    trigger, s_index, frg);
    GanttR.resize(finish);

    if (trigger == 0) scout_bees[s_index] = In_Pop(Solution, 0.0,
    finish, random);
    else if (trigger == 1) *(forager_bees[s_index][frg]) = Foragers(
    Solution, finish, GanttR, forager_bees[s_index][frg]->getPrio(),
    npv, forager_bees[s_index][frg]->getNPVVec(), forager_bees[
    s_index][frg]->getStatusVec(), forager_bees[s_index][frg]->
    getStartVec(),
```

```
forager_bees[s_index][frg]->getFinishVec(),
forager_bees[s_index][frg]->getCheckVec());
}

void MainFrame::Local_Search(int local_size, int scout, vector<bool>
dummy_status, vector<int> dummy_start, vector<int> dummy_finish,
vector<double> dummy_npv, vector<bool> dummy_check)
{
    vector<vector<int>> gannt; gannt.clear();
    vector<int> sol;
    sol.clear();
    sol = scout_bees[scout].GetSolution();
    vector<double> prob;
    prob.clear();
    prob = scout_bees[scout].GetPrio();
    double best_npv = scout_bees[scout].GetNPV();
    int best_dur = scout_bees[scout].GetSolutionDuration();
    int index = 0;
    int flag = 0;

    omp_set_dynamic(0);
    omp_set_num_threads(3);
#pragma omp parallel
    {
#pragma omp for schedule(static)
        for (int i = 0; i < local_size; i++)
        {
            if (i == 0)
            {
                *(forager_bees[scout][i]) = Foragers(sol, 0, gannt,
prob, 0.0, dummy_npv, dummy_status, dummy_start,
```

```

dummy_finish, dummy_check);
    SSGS_Forward(prob, scout, i, 1);
}
else if (i <= (local_size / 3))
{
    vector<double> prob1; prob1.clear();
    prob1 = scout_bees[scout].GetPrio();
    vector<int> sol1 = scout_bees[scout].GetSolution();
    int* temp1 = (int*)calloc(inputs, sizeof(int));
    for (int j = 0; j < int(prob1.size() * 0.48); j++)
    {
        int a1;
        double aa1;
        do
        {
            a1 = rand() % (prob1.size() / 2 - 1) + 1;
            aa1 = distr(eng);
            temp1[sol[a1]]++;
        } while (temp1[sol[a1]] > 1);
        prob1[sol[a1]] = aa1;
    }
    free(temp1);
    sol1.clear();
    *(forager_bees[scout][i]) = Foragers(sol1, 0, gantt,
    prob1, 0.0, dummy_npv, dummy_status, dummy_start,
    dummy_finish, dummy_check);
    SSGS_Forward(prob1, scout, i, 1);
}
else if (i <= (local_size / 3) * 2 - 1)
{
    int* temp2 = (int*)calloc(inputs, sizeof(int));
    vector<double> prob2 = scout_bees[scout].GetPrio();

```

```

        vector<int> sol2 = scout_bees[scout].GetSolution();
        for (int k = 0; k < int(prob2.size() * 0.48); k++)
        {
            int a2;
            double aa2;
            do
            {
                a2 = rand() % int((prob2.size() / 2)) +
int(prob2.size()) / 2;
                aa2 = distr(eng);
                temp2[sol[a2]]++;
            } while (temp2[sol[a2]] > 1);
            prob2[sol[a2]] = aa2;
        }
        free(temp2);
        sol2.clear();
        *(forager_bees[scout][i]) = Foragers(sol2, 0, gannt,
prob2, 0.0, dummy_npv, dummy_status, dummy_start,
dummy_finish, dummy_check);
        SSGS_Forward(prob2, scout, i, 1);
    }
    else if (i > (local_size / 3) * 2 - 1)
    {
        vector<double> prob3 = scout_bees[scout].GetPrio();
        vector<int> sol3 = scout_bees[scout].GetSolution();
        int* temp3 = (int*)calloc(inputs, sizeof(int));
        for (int jj = 0; jj < int(prob3.size() * 0.96); jj++)
        {
            int a3;
            double aa3;
            do
            {

```

```

        a3 = rand() % (prob3.size() - 2) + 1;
        aa3 = distr(eng);
        temp3[sol[a3]]++;
    } while (temp3[sol[a3]] > 1);
    prob3[sol[a3]] = aa3;
}
free(temp3);
sol3.clear();
*(forager_bees[scout][i]) = Foragers(sol3, 0, gantt,
prob3, 0.0, dummy_npv, dummy_status, dummy_start,
dummy_finish, dummy_check);
    SSGS_Forward(prob3, scout, i, 1);
}
    if ((forager_bees[scout][i]->getTotalNPV() - best_npv > 0.0)
&& (forager_bees[scout][i]->getDuration() <= best_dur))
    {
        best_npv = forager_bees[scout][i]->getTotalNPV();
        index = i;
        flag = 1;
    }
    else if (flag == 0)
    {
        index = 0;
    }
}
}
if (flag == 1)
{
    f_counter++;
    scout_bees[scout] = In_Pop(forager_bees[scout][index]->
getSolVec(), forager_bees[scout][index]->getTotalNPV(),
forager_bees[scout][index]->getDuration(),

```

```

forager_bees[scout][index]->getPrio());
    Local_Search(local_size, scout, dummy_status, dummy_start,
dummy_finish, dummy_npv, dummy_check);
    }
    else if (flag == 0)
    {
        best_bees[scout] = Best(forager_bees[scout][index]->
getSolVec(), forager_bees[scout][index]->getDuration(),
forager_bees[scout][index]->getGannt(),
forager_bees[scout][index]->getTotalNPV(),
forager_bees[scout][index]->getNPVVec(),
forager_bees[scout][index]->getStartVec(),
forager_bees[scout][index]->getFinishVec(),
        forager_bees[scout][index]->getCheckVec());
    }
}

vector<int> MainFrame::delay(vector<int> set, vector<int> set2,
int bestbees)
{
    vector<int> delta;
    for (size_t k = 0; k < set2.size(); k++)
// gia kathe act tou set2
    {
        for (size_t i = 0; i < set.size(); i++)
        {
            for (int j = 0; j < element[set[i]].getNumOfSucc(); j++)
            {
                if (set2[k] == element[set[i]].getSuccessor(j))
// an h act tou set2 EINAI successor kapoias activity
// tou set1 tote
            {

```

```
        delta.push_back(elite_bees[bestbees].getStart(
    set2[k]) - elite_bees[bestbees].getFinish(set[i]));
// to delay einai act_start set2 - act_finish set1
    }
    }
    }
    sort(delta.begin(), delta.end());
// sort gia na exw to minimum prwto, delta[0]
    return delta;
}

bool MainFrame::Push_Delay(int bestbee, int delay, vector<int>
set1, int deadline)
{
    bool change = false;
// deikths opou true an ginei kathusterhsh tou set1
    vector<int> vir_start; vir_start.clear();
// act 29 -> 51
    vector<int> vir_finish; vir_finish.clear();
// act 29 -> 53 praktika 52 sto gannt einai to finish - 1 ****
    int dur = 0;
// total duration olou tou set1
    for (size_t i = 0; i < set1.size(); i++)
    {
        vir_start.push_back(elite_bees[bestbee].getStart(set1[i]));
// pairnw to start, finish kai duration olwn twv act tou set1
        vir_finish.push_back(elite_bees[bestbee].getFinish(set1[i]));
        dur += element[set1[i]].getDuration();
    }
    bool ok = false;
    int count = 0;
```

```

    int rcount = 0;
    int flag = 0;
    int flag1 = 0;
    vector<vector<int>> dummy_gannt = elite_bees[bestbee].getGannt();
// Gannt[i][j], i = xronos, j = poroi | dummy_gannt epeidh
// den allazw kateutheian sto original gannt ths lushs
    for (size_t k = 0; k < set1.size(); k++)
    {
        for (int i = vir_start[k]; i < vir_finish[k]; i++)
        // **** to thelw edw gia na paw sth swsth thesi kai
        // na mhn exw overflow sto vector
        {
            for (int j = 0; j < resourceNum; j++)
            // gia kathe act tou set1 AFAIRW OLOUS (resourceNum)
            // porous apo start -> finish
            {
                dummy_gannt[i][j] = dummy_gannt[i][j] -
                element[set1[k]].getDemand(j);
            }
        }
    }
    vector<vector<int>> backup = dummy_gannt;
    // orizw ena backup gannt = me to dummy_gannt me tous porous
    //na exoun afairethei tw n act tou set1
    while (true)
    {
        flag = 0;                // deikths gia break
        flag1 = 0;               // deikths gia break
        count = 0;
        rcount = 0;              // metrhths upervashs oriou porwn
        vector<int> virt_NEW_start; virt_NEW_start.clear();
        // orizw ta NEW start, finish, dhladh start + delay

```



```
// kai finish + delay **
vector<int> virt_NEW_finish; virt_NEW_finish.clear();
for (size_t k = 0; k < set1.size(); k++)
{
    virt_NEW_start.push_back(elite_bees[bestbee].
getStart(set1[k]) + delay); // **
    virt_NEW_finish.push_back(elite_bees[bestbee].
getFinish(set1[k]) + delay); // **
    if (virt_NEW_finish[k] > deadline)
    {
        flag1 = 1;
    }
}
if (flag1 == 1)
{
    break;
}
for (size_t k = 0; k < set1.size(); k++)
{
    for (int i = virt_NEW_start[k]; i < virt_NEW_finish[k]; i++)
    {
        for (int j = 0; j < resourceNum; j++)
        {
            if (dummy_gannt[i][j] + element[set1[k]].
getDemand(j) > resourceCap[j])
            {
                rcount++; // an uparxei upervash, rcount++
            }
        }
        if (rcount > 0)
        {
            delay--;
        }
    }
}
```

```

        dummy_gannt = backup;
        flag = 1;
        break;
    }
    else
    {
        for (int j = 0; j < resourceNum; j++)
        {
            {
                count++;
                dummy_gannt[i][j] = dummy_gannt[i][j]
+ element[set1[k]].getDemand(j);
                if (count == resourceNum * dur)
                {
                    ok = true;
                }
            }
        }
    }
    if (flag == 1) break;    // ****
}
if ((delay <= 0) || (ok == true))
{
    break;    // break thn while
}
}
if (ok == true)    // an kathusterhsan oles
{

    for (size_t i = 0; i < set1.size(); i++)
    {

```

```

        elite_bees[bestbee].setNEWStart(set1[i], delay);
        elite_bees[bestbee].setFinish(set1[i], element[
set1[i]].getDuration());
        double val = element[set1[i]].getCashFlow() /
(pow(1.0 + rate, elite_bees[bestbee].getFinish(set1[i])));
        elite_bees[bestbee].setNPV(set1[i], val);
    }
    elite_bees[bestbee].setGannt(dummy_gannt);
    change = true;
}
return change;
}

void MainFrame::Find_Network_Set(int current, int start, int pt1[],
int pt2[], double* total_npv, int bestbees, bool status, int count1,
int count2)
{
    int fi = elite_bees[bestbees].getFinish(current);
    int fk = elite_bees[bestbees].getFinish(start);
    best_bees[bestbees].setCheck(start, true);
    if (status == true)
    {
        for (int i = 0; i < element[current].getNumOfPred(); i++)
        {
            if ((best_bees[bestbees].getCheck(element[current].
getPredecessor(i)) == false) && (element[current].getPredecessor(i)
!= start) && (elite_bees[bestbees].getFinish(element[current].
getPredecessor(i)) >= fk) && (elite_bees[bestbees].getNPV(element[
current].getPredecessor(i)) < 0.0))
            {
                pt1[count1] = element[current].getPredecessor(i);
                count1++;
            }
        }
    }
}

```

```

        *total_npv += elite_bees[bestbees].getNPV(element[
current].getPredecessor(i));
        best_bees[bestbees].setCheck(element[current].
getPredecessor(i), true);
    }
}
for (int i = 0; i < element[current].getNumOfSucc(); i++)
{
    if ((elite_bees[bestbees].getStart(element[current].
getSuccessor(i)) == fi) && (element[current].getSuccessor(i)
!= inputs - 1) && (best_bees[bestbees].getCheck(element[current].
getSuccessor(i)) == false))
    {
        pt1[count1] = element[current].getSuccessor(i);
        count1++;
        *total_npv += elite_bees[bestbees].getNPV(element[
current].getSuccessor(i));
        best_bees[bestbees].setCheck(element[current].
getSuccessor(i), true);
        Find_Network_Set(element[current].getSuccessor(i), start,
pt1, pt2, total_npv, bestbees, true, count1, count2);
    }
    else if (best_bees[bestbees].getCheck(element[current].
getSuccessor(i)) == false)
    {
        pt2[count2] = element[current].getSuccessor(i);
        count2++;
        best_bees[bestbees].setCheck(element[current].
getSuccessor(i), true);
    }
}
}

```

```
}

void MainFrame::Network_Delay(int deadline, int bestbee)
{
    elite_bees[bestbee] = Elite(best_bees[bestbee].getSolVec(),
    deadline, best_bees[bestbee].getGannt(),
    best_bees[bestbee].getTotalNPV(), best_bees[bestbee].
getNPVVec(), best_bees[bestbee].getStartVec(),
best_bees[bestbee].getFinishVec());
    elite_bees[bestbee].setStart(inputs - 1, deadline);
    elite_bees[bestbee].setNEWFinish(inputs - 1, deadline);
    elite_bees[bestbee].resizeGannt(deadline, resourceNum);
    best_bees[bestbee].setCheck(0, true);
    int del = 0;
    bool change = false;
    int change_count = 1;
    while (change_count > 0)
    {
        change_count = 0;
        for (int j = inputs - 1; j > 0; j--)
        {
            vector<int> set1; set1.clear();
            vector<int> set2; set2.clear();
            double total = 0.0;
            vector<int> delta; delta.clear();
            int pos = elite_bees[bestbee].getSol(j);
            if (elite_bees[bestbee].getNPV(pos) < 0.0)
            {
                int* pt1 = NULL;
                int* pt2 = NULL;
                pt1 = (int*)calloc(inputs, sizeof(int));
                pt2 = (int*)calloc(inputs, sizeof(int));
```

```
int count1 = 0;
int count2 = 0;

set1.push_back(pos);
total = best_bees[bestbee].getNPV(pos);
Find_Network_Set(pos, pos, pt1, pt2, &total, bestbee,
false, count1, count2);
for (int i = 0; i < inputs; i++)
{
    if (pt1[i] > 0) set1.push_back(pt1[i]);
    if (pt2[i] > 0) set2.push_back(pt2[i]);
}

for (int k = 0; k < inputs; k++)
{
    best_bees[bestbee].setCheck(k, false);
}
free(pt1);
free(pt2);
if (total < 0.0)
{
    sort(set1.begin(), set1.end());
    set1.erase(unique(set1.begin(), set1.end()),
set1.end());

    sort(set2.begin(), set2.end());
    set2.erase(unique(set2.begin(), set2.end()),
set2.end());

    delta = delay(set1, set2, bestbee);
    if (delta.size() > 0)
    {
        for (size_t i = 0; i < delta.size(); i++)
        {
```



```

bestbees].getSol(j))) && (best_bees[bestbees].getCheck(
elite_bees[bestbees].getSol(j)) == false) && (elite_bees[bestbees].
getSol(j) != start) && (elite_bees[bestbees].getFinish(elite_bees[
bestbees].getSol(j)) >= fk) && (elite_bees[bestbees].getNPV(
elite_bees[bestbees].getSol(j)) < 0.0))
    {
        pt1[count1] = elite_bees[bestbees].getSol(j);
        count1++;
        *total_npv += elite_bees[bestbees].getNPV(elite_bees[
bestbees].getSol(j));
        best_bees[bestbees].setCheck(elite_bees[bestbees].
getSol(j), true);
    }
}

for (int j = 0; j < elite_bees[bestbees].getSolSize(); j++)
{
    if ((fi == elite_bees[bestbees].getStart(elite_bees[
bestbees].getSol(j))) && (elite_bees[bestbees].getSol(j)
!= inputs - 1) && (best_bees[bestbees].getCheck(elite_bees[
bestbees].getSol(j)) == false))
    {
        pt1[count1] = elite_bees[bestbees].getSol(j);
        count1++;
        *total_npv += elite_bees[bestbees].getNPV(elite_bees[
bestbees].getSol(j));
        best_bees[bestbees].setCheck(elite_bees[bestbees].
getSol(j), true);
        Find_Schedule_Set(elite_bees[bestbees].getSol(j), start,
pt1, total_npv, bestbees, true, count1);
    }
}

```



```
}

vector<int> MainFrame::Schedule_Set_2(vector<int> set1, int bestbees)
{
    vector<int> set2; set2.clear();
    for (size_t i = 0; i < set1.size(); i++)
    {
        for (int k = 0; k < element[set1[i]].getNumOfSucc(); k++)
        {
            if (best_bees[bestbees].getCheck(element[set1[i]].
getSuccessor(k)) == false)
                set2.push_back(element[set1[i]].getSuccessor(k));
        }
    }
    sort(set2.begin(), set2.end());
    return set2;
}

void MainFrame::Schedule_Delay(int bestbee, int deadline)
{
    best_bees[bestbee].setCheck(0, true);
    int del = 0;
    bool change = false;
    int change_count = 1;
    while (change_count > 0)
    {
        change_count = 0;
        for (int j = inputs - 1; j > 0; j--)
        {
            vector<int> set1; set1.clear();
            vector<int> set2; set2.clear();
            double total = 0.0;
```

```

vector<int> delta; delta.clear();
int pos = elite_bees[bestbee].getSol(j);
if (elite_bees[bestbee].getNPV(pos) < 0.0)
{
    int* pt1 = (int*)calloc(inputs, sizeof(int));
    int count1 = 0;

    set1.push_back(pos);
    total = best_bees[bestbee].getNPV(pos);
    Find_Schedule_Set(pos, pos, pt1, &total, bestbee,
false, count1);
    for (int i = 0; i < inputs; i++)
        if (pt1[i] > 0) set1.push_back(pt1[i]);

    set2 = Schedule_Set_2(set1, bestbee);
    for (int k = 0; k < inputs; k++)
    {
        best_bees[bestbee].setCheck(k, false);
    }
    free(pt1);
    if (total < 0.0)
    {
        sort(set1.begin(), set1.end());
        set1.erase(unique(set1.begin(), set1.end()),
set1.end());

        sort(set2.begin(), set2.end());
        set2.erase(unique(set2.begin(), set2.end()),
set2.end());

        delta = delay(set1, set2, bestbee);
        if (delta.size() > 0)
        {
            for (size_t i = 0; i < delta.size(); i++)

```

```

        {
            if (delta[i] > 0)
            {
                del = delta[i];
                break;
            }
        }
        if (del > 0)
        {
            change = Push_Delay(bestbee, del, set1,
deadline);

            if (change == true) {
                change_count++;
            }
        }
    }
}

void MainFrame::Check_Schedule(int bees)
{
    for (int i = 0; i < bees; i++)
    {
        size_t pre_size = element[inputs - 1].getNumOfPred();
        int pc = 0;
        for (int j = 0; j < pre_size; j++)
        {
            if (elite_bees[i].getFinish(element[inputs - 1].
getPredecessor(j)) != elite_bees[i].getDuration())

```

```
{
    pc++;
}
}
int max = 0;
if (pc == pre_size)
{
    for (int j = 0; j < pre_size; j++)
    {
        if (elite_bees[i].getStart(element[inputs - 1].
getPredecessor(j)) >= max)
        {
            max = elite_bees[i].getFinish(element[
inputs - 1].getPredecessor(j));
        }
    }
    elite_bees[i].setStart(elite_bees[i].getSol(inputs - 1),
max);
    elite_bees[i].setNEWFinish(elite_bees[i].getSol(
inputs - 1), max);
    elite_bees[i].setDurationNEW(elite_bees[i].getFinish(
elite_bees[i].getSol(inputs - 1)));
    vector<vector<int>> gg;
    gg = elite_bees[i].getGannt();
    gg.resize(elite_bees[i].getDuration());
    elite_bees[i].setGannt(gg);
}

int suc_size = element[0].getNumOfSucc();
int sc = 0;
for (int j = 0; j < suc_size; j++)
{
```

```

        if (elite_bees[i].getStart(element[0].getSuccessor(j))
!= 0) { sc++; }
    }
    int min = 500;
    if (sc == suc_size)
    {
        for (int j = 0; j < suc_size; j++)
        {
            if (elite_bees[i].getStart(element[0].getSuccessor(j))
<= min)
            {
                min = elite_bees[i].getStart(element[0].
getSuccessor(j));
            }
        }
        int old_start = min;
        int old_finish = elite_bees[i].getDuration();
        for (int k = 1; k < inputs; k++)
        {
            elite_bees[i].setStart(elite_bees[i].getSol(k),
elite_bees[i].getStart(elite_bees[i].getSol(k)) - min);
            elite_bees[i].setFinish(elite_bees[i].getSol(k),
element[elite_bees[i].getSol(k)].getDuration());
            double val = element[elite_bees[i].getSol(k)].
getCashFlow() / (pow(1.0 + rate, elite_bees[i].g
etFinish(elite_bees[i].getSol(k))));
            elite_bees[i].setNPV(elite_bees[i].getSol(k), val);
        }
        elite_bees[i].setDurationNEW(elite_bees[i].getFinish(
elite_bees[i].getSol(inputs - 1)));
        double tt = 0.0;
        for (int k = 0; k < inputs; k++)

```

```
        {
            tt += elite_bees[i].getNPV(elite_bees[i].getSol(k));
        }
        elite_bees[i].setTotalNPV(tt);
        vector<vector<int>> dg(elite_bees[i].getDuration(),
vector<int>(resourceNum, 0));
        int k = 0;
        for (int t = old_start; t < old_finish; t++)
        {
            for (int j = 0; j < resourceNum; j++)
            {
                dg[k][j] = elite_bees[i].PrintGannt(t, j);
            }
            k++;
        }
        elite_bees[i].setGannt(dg);
    }
}
```

```
void MainFrame::Initial_Population(int N, vector<double> random,
vector<int> solution)
{
    int nn = 0;
    double X = 0.0;
    while (nn < N)
    {
        random.clear();
        random.push_back(1.0);
        for (int i = 0; i < inputs - 1; i++)
        {
            X = distr(eng);
```

```
        random.push_back(X);
    }
    random.push_back(0.0);
    TotalNPV = 0.0;
    SSGS_Forward(random, nn, 0, 0);
    for (int i = 0; i < inputs; i++)
    {
        element[i].setNPV(element[i].getCashFlow(), rate,
element[i].getFinish());
        TotalNPV += element[i].getNPV();
    }
    scout_bees[nn] = In_Pop(scout_bees[nn].GetSolution(), TotalNPV,
scout_bees[nn].GetSolutionDuration(), scout_bees[nn].GetPrio());
    nn++;
    solution.clear();
}
}
```