
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

# Cooperative Scheme for Tasks Scheduling in Internet of Things

Κωνσταντίνος Μακρίδης

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Σταμούλης Γεώργιος

Καθηγητής Πανεπιστημίου Θεσσαλίας

ΣΥΝΕΠΙΒΛΕΠΩΝ

Κολομβάτσος Κωνσταντίνος

Επίκουρος Καθηγητής Πανεπιστημίου Θεσσαλίας

Λαμία 15 Οκτωβρίου έτος 2020

*«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:*

*1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

*2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

*3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια*

*4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.*

Ημερομηνία: 15/10/2020

Ο Δηλ.

*(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση*
*του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων*
*σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»*

# Περίληψη

Η παρούσα πτυχιακή εργασία στοχεύει στην ανάλυση του διαδικτύου των πράγματων και της διαχείρισης εργασιών σε αυτό. Το διαδίκτυο των πραγμάτων είναι μία ολοένα και περισσότερο αναπτυσσόμενη τεχνολογία, που σταδιακά θα κερδίσει και άλλο έδαφος στο άμεσο μέλλον. Η ανάπτυξη αυτή περιλαμβάνει την ένταξη πολλών νέων συσκευών στο διαδίκτυο, προκαλώντας ερωτηματικά σχετικά με την διαχείριση του δικτύου σε ότι αφορά την κατανομή και διαχείριση εργασιών. Η διαχείριση εργασιών στα υπολογιστικά συστήματα έχει σχεδιαστεί με βάση πρότυπα που έχουν δημιουργηθεί για τον άνθρωπο και τη διαχείριση έργου στον άνθρωπο. Λαμβάνοντας υπόψιν τις ανάγκες για άμεση χρονική απόκριση, αποδοτικότητα, συμβατότητα καθώς και εγκυρότητα αποτελεσμάτων. Σε συνδυασμό με τη συνεχή καταγραφή των ενεργειών που εξελίσσονται στο δίκτυο, έχουν αναπτυχθεί μηχανισμοί στα λειτουργικά συστήματα προκειμένου να συντονίσουν και να διαχωρίσουν όλες αυτές τις συσκευές. Στα πλαίσια την μελέτης των παραπάνω θεματικών ενοτήτων συντάχθηκε μια εφαρμογή προσομοιώνοντας ένα δίκτυο με πολλούς κόμβους. Σκοπός της εφαρμογής είναι η μελέτη πολυκριτηριακών μεθόδων λήψης αποφάσεων, σε συνδυασμό με μεθόδους λήψης αποφάσεων με ένα κριτήριο. Μέσα από την καταγραφή και τη μελέτη των δεδομένων που δημιουργήθηκαν από την εφαρμογή, είμαστε σε θέση να έχουμε μετρήσεις ποιοτικές και ποσοτικές. Τέλος καταλήγουμε στο να θεωρούμε ότι κάθε μέθοδος έχει διαφορετικά οφέλη. Αδιαμφησβήτητα όμως οι πολυκριτηριακές μέθοδοι παράγουν αποτέλεσμα σε κάθε περίπτωση επιθυμητό εξασφαλίζοντας συνέπεια αποτελεσμάτων σε όλους τους τομείς.

# Abstract

The aim of this thesis is to enlarge upon the Internet of Things and the Task Management on it. The Internet of Things is an ever-expanding technology that is gradually gaining ground. This kind of development encompasses the incorporation of an increasing number of devices in the Internet, which results in questioning its ability to distribute and manage different tasks. The task management that takes place in computer systems is custom-built so as to meet the related individual needs. Considering the needs for immediate responses, efficiency, compatibility, validity in results coupled with constant record keeping in the Internet, various mechanisms have been proposed in order to moderate and classify all these devices. Alongside with this research, an application that simulates a net with many nodes was created. The goal of this application is to study the decision making through multiple criteria decision making, in conjunction with methods that include just one criterion. By recording and studying the data provided by the application, quality and quantity measurements arise. In conclusion, each method leads to different benefits. Undeniably though, methods with multiple criteria bring about desirable results ensuring consistency in results in each and every domain.

# Περιεχόμενα

# Εισαγωγή

Η ανάπτυξη και η εξάπλωση τεχνολογιών που σχετίζονται με το διαδίκτυο, γίνεται ολοένα και πιο διαδεδομένη. Καθημερινά η ανθρωπότητα έρχεται αντιμέτωπη με ολοένα και πιο έξυπνες συσκευές οι οποίες είναι υπεύθυνες για τη βελτίωση, και την πιο εύκολη διαχείριση καθημερινών προβλημάτων. Κάνουμε λόγο για συσκευές που συνδέονται στο διαδύκτιο και με ευφυή τρόπο απλουστεύουν χρονοβόρα προβλήματα και διαδικασίες της ανθρώπινης και όχι μόνο καθημερινότητας. Ο λόγος γίνεται για το διαδύκτιο των πραγμάτων που σε πολλούς φαντάζει κάτι μακρινό ενώ στην πραγματικότητα βρίσκεται ήδη στις ζωές όλων.

Καινοτομίες τόσο σε επίπεδο software όσο και σε επίπεδο hardware κατακλύζουν την παγκόσμια κοινότητα εντάσσοντας τα οφέλη του διαδύκτιο των πραγμάτων στις ζωές των ανθρώπων. Πολλές από τις συσκευές που συναντάμε γύρω μας, λαμβάνουν τον όρο έξυπνες λαμβάνοντας χρήσιμες πληροφορίες για την απλοποίηση την καθημερινότητας. Συσκευές όπως το ψυγείο, μια λάμπα ακόμα και ένα αυτοκίνητο που ουδεμία σχέση εκ΄ πρώτης όψεως δείχνουν να έχουν με το ίντερνετ, λαμβάνουν μέρος και ενσωματώνονται σε αυτό.

Κλάδοι πιο εξειδικευμένοι εντάσσουν κάθε είδους συσκευές στο διαδύκτυο των πραγμάτων στην προσπάθεια να αποσπάσουν όσο το δυνατόν περισσότερες πληροφορίες. Με εφαρμογές στο σπίτι, στο δρόμο, στην πόλη, στη βιομηχανία ακόμα και στην ιατρική η ανάπτυξη του διαδίκτυο των πραγμάτων είναι ραγδαία και επιφέρει πολύ σημαντικές και ριζικές αλλαγές.

Εκτός από απλές συσκευές προφανώς περιλαμβάνονται και υπολογιστικοί κόμβοι. Είναι σαφές ότι χωρίς αυτούς δεν θα μπορούσε να υπάρχει συντονισμός και στοχευμένη λειτουργικότητα. Δημιουργείται λοιπόν ένα πολύ μεγάλο δίκτυο που λαμβάνει μια αρκετά μεγάλη γεωγραφική έκταση καθώς οι συσκευές που περιλαμβάνονται, δεν είναι δυνατόν σε όλες τις περιπτώσεις να συνορεύουν γεωγραφικά.

Όπως όλα τα δίκτυα έτσι και στο διαδίκτυο των πραγμάτων δημιουργούνται εργασίες για τις ανάγκες του δικτύου ενώ η συνεχής αποστολή δεδομένων από τις συσκευές-αισθητήρες πρέπει με κάποιο τρόπο να διαχειριστούν. Κάθε ενέργεια καταγράφεται τόσο για να υπάρχει μια εικόνα της κατάστασης του δικτύου όσο και σε περίπτωση

σφαλμάτων να μπορούν να γίνουν διορθώσεις. Η ποιοτικότερη απόδοση προϋποθέτει μια άρτια και σωστά κατανεμημένη διαχείριση στην διαχείριση των εργασιών του δικτύου.

Τη λύση στο πρόβλημα έρχεται να προσφέρει η επιστήμη της διαχείρισης έργου κάνοντας μία αναφορά σε αυτήν. Κάθε έργο για να αποδώσει τα μέγιστα οργανώνεται η διαχείριση του βάση κάποιων θεμελιωδών κανόνων. Η σχέση ανάμεσα στην πραγματική διαχείριση έργου στον άνθρωπο και ένα υπολογιστικό σύστημα είναι άμεσα συνδεδεμένη. Αρχικά γιατί είναι ανθρώπινο δημιούργημα κάθε συσκευή που ανήκει στην περίπτωση μας στο διαδίκτυοτων πραγμάτων και επιβάλλεται η διαχείριση των εργασιών να υπόκειται σε κάποιες αρχές. Στα πλαίσια της συγκεκριμένης εργασίας θα αναπτυχθούν οι τεχνικές αυτές καθώς και οι τρόποι με τους οποίους το δίκτυο επιτυγχάνει την διαχείριση. Όλα αυτά πραγματοποιούνται με ένα διαχωρισμό μεταξύ των συσκευών, ώστε να είναι σε θέση το λειτουργικό σύστημα του δικτύου να γνωρίζει ποιες συσκευές μπορούν να εκτελέσουν εργασίες και ποιες είναι αυτές. Τα επιμέρους κριτήρια για τον διαμοιρασμό των εργασιών θα είμαστε σε θέση με την ολοκλήρωση της μελέτης να τα παρατηρήσουμε, να τα συγκρίνουμε καθώς και να πραγματοποιήσουμε μετρήσεις πάνω σε αυτά.

Αναπτύξαμε μία εφαρμογή σε γλώσσα Java η οποία προσομοιώνει ένα δίκτυο με μεταβλητό πλήθος κόμβων. Οι κόμβοι αυτοί δέχονται πλήθος διεργασιών, ενώ κάθε κόμβος διέπεται από χαρακτηριστικά όπως ταχύτητα, αναγνωριστικό, φορτίο καθώς και κόστος επικοινωνίας μεταξύ δύο κόμβων. Το πείραμα που εκτελείται αποστέλλει τυχαία σε κόμβους διεργασίες και μέσα από την εφαρμογή πολυκριτηριακών αλλά και μη μεθόδων λήψης αποφάσεων, καθορίζεται αν θα εκτελεσθεί τοπικά ή θα σταλεί σε κάποιον άλλο κόμβο. Καταγράφουμε και παρουσιάζουμε μετρικές που αντλούνται από την εφαρμογή και είναι ικανές να μας οδηγήσουν σε χρήσιμα συμπεράσματα.

# 1.Internet of Things

## 1.1 Εισαγωγή

Το Internet of Things (IoT), ελληνιστί Διαδίκτυο των Πραγμάτων, αποτελεί μία σύνθετη και πολύπλοκη ως προς τον ορισμό της έννοια. Υπέρ αυτής της διαπίστωσης συνηγορεί ο ποικίλος σημασιολογικός φόρτος τον οποίο επωμίζεται ο όρος κάθε φορά από τον εκάστοτε εξεταστή-αναλυτή του. Σε γενικές γραμμές, ωστόσο, ο όρος χρησιμοποιείται ώστε να αποδώσει την έννοια «έξυπνων» και αποκλειστικά αναγνωρίσιμων συσκευών, των «πραγμάτων», τα οποία πρωτευόντως συνδέονται και αλληλοεπιδρούν μεταξύ τους και με τους ανθρώπους και δευτερευόντως, συνδεόμενα στο διαδίκτυο παρέχουν τη δυνατότητα για βελτιστοποίηση του τρόπου ζωής και εργασίας των ανθρώπων. Τα «Πράγματα όσον αφορά στο Διαδίκτυο των Πραγμάτων, είναι αντικείμενα του φυσικού κόσμου δηλαδή φυσικά πράγματα ή αντικείμενα του κόσμου της πληροφορίας. Όταν ανήκουν στον τελευταίο, υπάρχει η δυνατότητα εντοπισμού τους και ενσωμάτωσής τους σε δίκτυα επικοινωνιών» [1].

Η Διεθνής Ένωση Τηλεπικοινωνιών (ITU), ένας οργανισμός των Ηνωμένων Εθνών με έδρα τη Γενεύη αρμόδιος για τις τηλεπικοινωνίες και τη διακίνηση της πληροφορίας, σε μία προσπάθεια επισκόπησης του Διαδικτύου των Πραγμάτων αποσαφηνίζει τον όρο ως εξής: « Μία παγκόσμια υποδομή για την κοινωνία της πληροφορίας, η οποία ενεργοποιεί προηγμένες υπηρεσίες μέσω της διασύνδεσης (φυσικών και εικονικών) πραγμάτων βασισμένων σε υπαρκτή και εξελισσόμενη διαλειτουργική πληροφόρηση και τηλεπικοινωνιακές τεχνολογίες» [1].

Η διασύνδεση αυτή του φυσικού με τον εικονικό κόσμο ανοίγει νέους ορίζοντες στον τρόπο ζωής των ανθρώπων, οι οποίοι μπορούν πλέον να προηγηθούν στον κόσμο της πληροφορίας όποτε και από όπου το θελήσουν. Ταυτοχρόνως, ωστόσο, ανοίγει σε αυτόν τον άγνωστο για τους πολλούς νέο κόσμο και ένα ολόκληρο πεδίο απειλών και κινδύνων ασφαλείας. Το IoT κάνει χρήση αισθητήρων προκειμένου για τη συλλογή πληροφοριών. Οι αισθητήρες δεν είναι βέβαια άγνωστοι στην καθημερινή ζωή του ευρέως κοινού και δη στο κομμάτι χρήσης τεχνολογικών μέσων. Κάμερες, αισθητήρες φωτός, εγγύτητας, τοποθεσίας, δακτυλικών αποτυπωμάτων, βηματόμετρα, θερμόμετρα είναι μερικοί μόνο από τους ενσωματωμένους αισθητήρες που διαθέτει καθένα από τα «έξυπνα» τηλέφωνα και που πολύς κόσμος είτε αγνοεί την ύπαρξη είτε την ξεχνάει. Εντούτοις, σε ομιλία του ο Kevin Ashton, θεμελιωτής

του όρου «Διαδίκτυο των Πραγμάτων», δήλωσε μεταξύ άλλων πως « …Το Διαδίκτυο των Πραγμάτων συμβαίνει ήδη, αλλά ίσως δεν μπορούμε να το δούμε εν συγκρίσει με τα έξυπνα τηλέφωνα που είναι και ορατά και απτά. Η Ταυτοποίηση μέσω Ραδιοσυχνοτήτων (RFID) είναι μία τέτοια τεχνολογία του Διαδικτύου των Πραγμάτων που υπάρχει μεν αλλά δεν είναι και απαραιτήτως ορατή. Επομένως, το Διαδίκτυο των Πραγμάτων μπορεί να εξελιχθεί πάρα πολύ πριν γίνει ορατό σε όλους» [2].

## 1.2 Επιπλέον ορισμοί του όρου Διαδίκτυο των Πραγμάτων

Ο όρος Διαδίκτυο των Πραγμάτων έχει χρησιμοποιηθεί σε πληθώρα από περιβάλλοντα, επιφορτισμένος ποικιλοτρόπως εννοιολογικά, προερχόμενος κάθε φορά από διαφορετικούς πομπούς, άλλοτε ιδιώτες και άλλοτε εταιρίες . Υπό αυτό το πρίσμα, πλήθος ερευνητικών εργασιών και επιστημονικών βιβλίων έχουν ασχοληθεί με τον όρο. Οι εξειδικευμένες τεχνικές εταιρίες, οι οποίες ήδη τρόπον τινά είναι μέρος του Διαδικτύου των Πραγμάτων και που διαβλέπουν μια επιχειρηματική προοπτική, ικανή να τους εξασφαλίσει ένα βιώσιμο και κερδοφόρο μέλλον, χρησιμοποιούν κυρίως τον όρο για να περιγράψουν έναν τρόπο βελτίωσης της αποδοτικότητας στην παραγωγή τους, έναν δρόμο προς την καινοτομία[3].

Η Cisco Systems, Inc. ορίζει το IoT ως μία ιδέα κατά την οποία ολοένα και περισσότερα πράγματα θα είναι συνδεδεμένα στο Διαδίκτυο με στόχο την διευκόλυνση της καθημερινότητας στον κόσμο. Ωστόσο, όσο περισσότερα Πράγματα συνδέουμε στο Διαδίκτυο, η ανάγκη για IPv6, για μεγάλο όγκο δεδομένων και για υπολογιστικό νέφος συνεχώς θα αυξάνει και το Διαδίκτυο των Πραγμάτων θα μετουσιωθεί σταδιακά σε Διαδίκτυο των Πάντων. Η Cisco, εν τέλει, θεωρεί το IoT ένα δίκτυο όπου ο αριθμός των συνδεδεμένων συσκευών αυξάνεται συνεχώς, ενώ η κατάσταση αυτή αλλάζει άπαξ και συνδεθούν τα πάντα[4].

Η IBM ορίζει το IoT περισσότερο ως σύνδεση συστημάτων μεταξύ τους, παρά ως σύνδεση συσκευών. Για αυτό τον λόγο, ως εταιρεία, επιχειρεί να δημιουργήσει ένα «σύστημα συστημάτων». Η IBM περιγράφει το Διαδίκτυο των Πραγμάτων ως ένα μέσο για δημιουργία ενός «εξυπνότερου» πλανήτη. Το μέσο αυτό χωρίζεται ώστε να εξυπηρετήσει δύο επιμέρους σκοπούς «Ο πρώτος είναι να γίνει περισσότερο (το Διαδίκτυο των Πραγμάτων) αποδοτικό, λιγότερο επιζήμιο, να συνδέει διαφορετικές

πτυχές της ζωής, οι οποίες θα επηρεάζουν η μία την άλλη με περισσότερο συνειδητούς, σκόπιμους και έξυπνους τρόπους. Ο δεύτερος είναι να παράγει θεμελιωδώς καινούργια γνώση, δραστηριότητα αλλά και νέες μορφές κοινωνικών σχέσεων»[5].

Όσον αφορά στους ορισμούς από άτομα, δηλαδή, επιμέρους εξεταστές-αναλυτές του όρου, αξιοσημείωτος είναι αυτός που έχει ειπωθεί από τον Δρ. John Barrett, επικεφαλής των Ακαδημαϊκών Σπουδών στον τομέα Embedded Systems Research του Ινστιτούτου Τεχνολογίας Cork, στο πλαίσιο μιας TEDx ομιλίας του. «Στο γενικό πλαίσιο του IoT όλα τα πράγματα θα χρειαστούν έναν μοναδικό τρόπο ταυτοποίησης (IPv6), ικανότητα για επικοινωνία, με κάποιον τρόπο ικανότητα να αισθάνονται ( μέσω όρασης, όσφρησης, αφής, κλπ.) αλλά και να ελέγχονται. Με όλα τα διαθέσιμα δεδομένα, υπάρχει μια ανάγκη για έναν πρακτικό και αποδοτικό τρόπο για παρουσίαση των δεδομένων που είναι σχετικά με ένα συγκεκριμένο συγκείμενο. Η απόφαση του τι είναι σχετικό γίνεται μια βασική ερώτηση. Εξαρτάται από τα πράγματα αφ' εαυτά ώστε να αποφασίσουν για το τι είναι σχετικό και το τι όχι. Σε ορισμένες περιπτώσεις, τα «σχετικά» δεδομένα μπορεί και χρησιμοποιηθούν λανθασμένα, με ένα έναν τρόπο που επηρεάζει αρνητικά τους ανθρώπους. Για παράδειγμα, μια συσκευή που παρακολουθεί την υγεία σου, μπορεί να στείλει ειδοποίηση στο νοσοκομείο αν η υγεία σου βρίσκεται σε κρίσιμη κατάσταση. Όμως, με τη χρησιμοποίηση των ίδιων πληροφοριών, η ασφαλιστική εταιρία σου αυτομάτως αυξάνει το ασφάλιστρο υγείας σου [6] .

Ένας ακόμη ορισμός είναι και αυτός του Kevin Ashton, ο οποίος διαβλέπει στο Διαδίκτυο των Πραγμάτων μια θεμελιώδη προοπτική για τη δημιουργία λύσεων σε μελλοντικά προβλήματα. Αυτός ορίζει το Διαδίκτυο των Πραγμάτων ως υπολογιστές που διαισθάνονται τον πραγματικό κόσμο μέσω των συστημάτων τους και για τον εαυτό τους, επομένως οι πληροφορίες για Πράγματα στον κόσμο μπορούν να είναι διαθέσιμες μέσω του Διαδικτύου. Το ζήτημα με το IoT είναι πως να μην παραταχθούν αισθητήρες παντού, αλλά αντ' αυτού να δημιουργηθεί ένα σύστημα που θα είναι ικανό να εκμεταλλευτεί όλα τα διαθέσιμα δεδομένα και αυτομάτως θα είναι σε θέση να διαχωρίσει τι σημαίνει το καθετί[2].

Το CASAGRAS (Coordination and Support Action for Global RFID-related Activities and Standardisation) είναι ένα ερευνητικό σχέδιο χρηματοδοτημένο από το

7ο Πρόγραμμα Πλαίσιο, το οποίο εστιάζει στις διεθνείς διαστάσεις που σχετίζονται με τους κανονισμούς, την προτυποποίηση και άλλες προϋποθέσεις για την υλοποίηση του IoT. Ορίζεται, λοιπόν, το Διαδίκτυο των Πραγμάτων ως « Ένα παγκόσμιο δίκτυο υποδομής, που συνδέει φυσικά και εικονικά πράγματα μέσω της εκμετάλλευσης δεσμευμένων δεδομένων και ικανοτήτων επικοινωνίας. Αυτή η υποδομή συμπεριλαμβάνει υπαρκτά και εξελισσόμενα επιτεύγματα του Διαδικτύου και δικτύων. Θα προσφέρει συγκεκριμένη αναγνώριση πραγμάτων, αισθητήρες και ικανότητα σύνδεσης ως βάση για την ανάπτυξη ανεξάρτητων συνεργαζόμενων υπηρεσιών και εφαρμογών. Αυτές θα χαρακτηρίζονται από έναν υψηλό βαθμό δέσμευσης δεδομένων, μεταφορά γεγονότων, συνδεσιμότητα στα δίκτυα και διαλειτουργικότητα». Προκειμένου για τη δημιουργία ενός τέτοιου Διαδικτύου θα χρειαστεί αρχικά ένα βασικό πλαίσιο ώστε να οριστεί και να φιλοξενηθεί η ανάπτυξή του[7]

## 1.3 Ιστορική Αναδρομή

Το 1832 ο Baron Schilling δημιούργησε έναν ηλεκτρομαγνητικό τηλέγραφο στη Ρωσία και ένα χρόνο αργότερα ο Carl Friedrich Gauss μαζί με τον Wilhelm Weber δημιούργησαν έναν κώδικα που τους επέτρεπε να επικοινωνούν μεταξύ τους σε απόσταση 1200 μέτρων στο Γκέτινγκεν της Γερμανίας. Το1844 ο Samuel Morse έστειλε τον πρώτο κωδικοποιημένο μήνυμα με τηλέγραφο από την Ουάσινγκτον, Π.Κ στη Βαλτιμόρη.

Το 1926 ο Nikola Tesla σε συνέντευξή του στο περιοδικό Colliers είχε δηλώσει: « Όταν η ασύρματη σύνδεση εφαρμοστεί τέλεια, ολόκληρη η γη θα μετατραπεί σε έναν τεράστιο εγκέφαλο, που στην πραγματικότητα ισοδυναμεί με το να είναι όλα τα πράγματα πολύ μικρά κομμάτια ενός πραγματικού και ρυθμικού όλου και τα όργανα μέσω των οποίων θα είμαστε σε θέση να κάνουμε αυτά θα είναι εκπληκτικά απλά ιδίως σε σχέση με το τωρινό μας τηλέφωνο».

Το 1964 στο βιβλίο Understanding Media: The Extensions of Man (The MIT Press) ο Marshall McLuhan δήλωσε: «με τα ηλεκτρικά μέσα, δημιουργούμε μία δυναμική μέσω της οποίας όλες οι προηγούμενες τεχνολογίες- συμπεριλαμβανομένων και των πόλεων- θα μεταφραστούν σε πληροφοριακά συτήματα». Δύο χρόνια μετά ο Karl

Steinbuch, Γερμανός επιστήμονας και καινοτόμος είπε: «Μέσα σε λίγες δεκαετίες, οι υπολογιτές θα είναι συνυφασμένοι σχεδόν με κάθε βιομηχανικό προϊόν»[8].

Η υλοποίηση της βασικής ιδέας και αντίληψης ενός δικτύου πραγμάτων φαίνεται πως συντελέστηκε επίσημα για πρώτη φορά το 1982, όταν τέσσερις φοιτητές του τμήματος της Επιστήμης των Υπολογιστών του πανεπιστημίου Carnegie Mellon εγκατέστησαν διακόπτες σε έναν αυτόματο πωλητή αναψυκτικών. Με αυτόν τον τρόπο, κατάφεραν να μετρήσουν πόσα μπουκάλια αναψυκτικών παρέμεναν στον αυτόματο πωλητή, σε ποιες σειρές και για πόσο χρονικό διάστημα. Στην περίπτωση που κάποιο μπουκάλι παρέμενε για μεγάλο χρονικό διάστημα σε μια σειρά, αυτό χαρακτηριζόταν ως κρύο. Η εφεύρεση αυτή, γνωστή ως ARPANET-connected coke machine, αποτέλεσε αντικείμενο έμπνευσης για πολλούς εφευρέτες ώστε να αναπτύξουν και τα δικά τους τέτοια συστήματα.

Το 1989 Tim Berners-Lee εφηύρε τον Παγκόσμιο Ιστό συντελώντας καταλυτικά στην ανάπτυξη του Διαδικτύου των Πραγμάτων λόγω της διευκόλυνσης της διασύνδεσης των διαφόρων συσκευών. Μόλις έναν χρόνο αργότερα, ο John Romkey παρουσίασε την πρώτη συσκευή του IoT, μία φρυγανιέρα συνδεδεμένη στο Διαδίκτυο[9].

Ακολούθησε μία εποχή από το 1991 έως και το 1998, στην οποία έγιναν πολλές προσπάθειες ορισμού αυτού του πρωτόγνωρου για την εποχή αλλά και αδιαμφισβήτητα νεωτερικού δικτύου-σύλληψης.

Το 1991 ο Mark Weiser σε μία εργασία με θέμα τη διαδεδομένη χρήση των υπολογιστών του με τίτλο "The Computer of the 21st Century" δημιούργησε το σύγχρονο όραμα του IoT ενώ ανέφερε μεταξύ άλλων «Οι πιο σημαντικές τεχνολογίες είναι αυτές που εξαφανίζονται. Υφαίνουν τους εαυτούς τους γύρω από το ύφασμα της καθημερινής ζωής μέχρι που δεν ξεχωρίζουν καθόλου από αυτό» [10][8]. Τρία χρόνια αργότερα, ο Reza Raji περιέγραψε την ιδέα του Διαδικτύου των Πραγμάτων στο περιοδικό IEEE Spectrum ως « (μετακινούμενα) μικρά πακέτα δεδομένων σε μεγάλα σετ κόμβων, προκειμένου να ενσωματώσουν και να αυτοματοποιήσουν τα πάντα από απλές οικιακές συσκευές μέχρι και ολόκληρα εργοστάσια» [11]. Την ίδια χρονιά ο Steve Mann δημιούργησε την πρώτη φορητή κάμερα που συνδεόταν με το Διαδίκτυο την WearCam [9].

Τον Οκτώβριο του 1997, το πανεπιστήμιο Carnegie Mellon, το Ινστιτούτο Τεχνολογίας της Τζόρτζια και το MIT συνδιοργάνωσαν το πρώτο διεθνές συμπόσιο

για Wearable Computers, στο οποίο συζητήθηκαν μεταξύ άλλων η απόδοση της υποστήριξης στους wearable computers, οι μηχανισμοί καταχώρησης και οι επικοινωνιακές ικανότητες. Το 1998, έγινε μία εισαγωγή στο IPv6 της πιο πρόσφατης αναθεώρησης του πρωτοκόλλου του Διαδικτύου. Μέχρι τότε ακόμη και η ασύλληπτη η ύπαρξη ενός ευρέως Δικτύου IoT λόγω κυρίως της έλλειψης αρκετών διευθύνσεων IP προκειμένου για εγγραφή όλων των συνδεδεμένων συσκευών.

Το 1999 αποτελεί ένα έτος ορόσημο για την γενικότερη ανάπτυξη της τεχνολογίας αλλά και την έναρξη της ακμής του IoT. Ο Kevin Ashton, διοικητικός υπεύθυνος στα Auto-ID Labs του MIT, σε μία παρουσίασή του στην εταιρεία Procter and Gamble χρησιμοποίησε για πρώτη φορά τον όρο Διαδίκτυο των Πραγμάτων. Μέχρι τότε προκειμένου να αποδοθεί η ιδέα χρησιμοποιούνταν ο κατά τα άλλα ανακριβής όρος «ενσωματωμένο διαδίκτυο». Στην ίδια παρουσίαση, ο Ashton συνέδεσε με καινοτόμο τρόπο το Διαδίκτυο με την ταυτοποίηση μέσω ραδιοσυχνοτήτων (RFID). Αργότερα την ίδια χρονιά, ο Kevin Ashton, ο David Brock και ο Sanjay Sharma εκτέλεσαν της ιδέα του πρώτου σχετικά με την συνδεσιμότητα RFID και ίδρυσαν τα Auto-ID Labs, ένα σύστημα ακαδημαϊκών εργαστηρίων για έρευνα με στόχο την ανάπτυξη νέων τεχνολογιών και την ενσωμάτωσή τους στον εμπορικό τομέα [9]. Την ίδια χρονιά, τέλος, ο Neil Gross δήλωσε στο περιοδικό Bloomberg Businessweek: « Στα επόμενα εκατό χρόνια ο πλανήτης γη θα έχει φορέσει μία ηλεκτρονική στολή . Θα χρησιμοποιεί το Διαδίκτυο ως ικρίωμα για να υποστηρίξει και να μεταδώσει τις αισθήσεις του. Αυτή η στολή ήδη ράβεται. Αποτελείται από εκατομμύρια ενσωματωμένες συσκευές μέτρησης: θερμοστάτες, μετρητές πίεσης, ανιχνευτές μόλυνσης, κάμερες, μικρόφωνα, αισθητήρες γλυκόζης, καρδιογραφήματα, ηλεκτροεγκεφαλογραφήματα. Αυτά θα εξετάσουν και ελέγξουν τις πόλεις και τα απειλούμενα είδη, την ατμόσφαιρα, τα πλοία, τους αυτοκινητοδρόμους, τους στόλου φορτηγών, τις συζητήσεις, τα σώματά μας- ακόμη και τα όνειρά μας»[8].

Οι δύο δεκαετίες που ακολούθησαν ήρθαν να επιβεβαιώσουν πανηγυρικώς την τελευταία αυτήν πρόβλεψη. Πράγματι, αρχικά με αργά αλλά σταθερά βήματα και στην συνέχεια με ιλιγγιώδη ταχύτητα, το Διαδίκτυο των Πραγμάτων εξαπλώθηκε και συνεχίζει να εξαπλώνεται σε ολόκληρο τον πλανήτη, σε σημείο τέτοιο ώστε είναι εξαιρετικά δύσκολο να παρακολουθήσει κανείς την κάθε επιμέρους εξέλιξη.

Από το 2004 και έπειτα ο όρος Διαδίκτυο των Πραμάτων απέκτησε δημοφιλία και ευρύτατη αναγνώριση. Στους επιστημονικούς κύκλους, βιβλία, εφημερίδες, περιοδικά άρχισαν να χρησιμοποιούν ολοένα και περισσότερο τον όρο. Μερικά από τα σημαντικότερα επιστημονικά φύλλα ήταν αυτά των The Guardian, Scientific American και the Boston Globe. Συγχρόνως, η ταυτοποίηση μέσω ραδιοσυχνοτήτων (RFID) γίνεται γνωστή και υιοθετείται σε μεγάλο βαθμό από το Υπουργείο Άμυνας των ΗΠΑ ως μέρος του προγράμματος Savi καθώς και από μεγάλη αλυσίδα καταστημάτων Walmart επεκτείνοντας έτσι και στον εμπορικό κόσμο τη χρήση της RFID.

Το 2005 το IoT έφτασε σε ένα νέο επίπεδο με τη δημοσίευση της αναφοράς πάνω στο θέμα από τη Διεθνή Ένωση Τηλεπικοινωνιών. Τρία χρόνια αργότερα, διοργανώθηκε η Πρώτη Διεθνής Συνδιάσκεψη για το Διαδίκτυο των Πραγμάτων στην Ζυρίχη της Ελβετίας. Αξιοσημείωτο είναι πως το 2008 σημειώθηκε μία έκρηξη στον αριθμό των συσκευών που ήταν συνδεδεμένες στο IoT καθώς την χρονιά εκείνη περισσότερα Πράγματα ήταν συνδεδεμένα στο Διαδίκτυο παρά άνθρωποι καθώς το 2010 οι συνδεδεμένες στο Διαδίκτυο συσκευές ανήλθαν στις 12,5 δισεκατομμύρια ενώ ο παγκόσμιος πληθυσμός την ίδια χρονιά αυξήθηκε σε 6,8 δισεκατομμύρια. Δεδομένου αυτού η διετία 2008-2009 χαρακτηρίστηκε από το Cisco Internet Business Solutions Group (IBSG) ως η διετία γέννησης του Διαδικτύου των Πραγμάτων[8][9].

Το 2010, ο πρωθυπουργός της Κίνας Wen Jiabao σε μία προσπάθεια αντιμετώπισης της οικονομικής κρίσης και άλλων ευαίσθητων θεμάτων στη χώρα, αποφάσισε να δοθεί ιδιαίτερη προσοχή στο Διαδίκτυο των Πραγμάτων ανακηρύσσοντάς το σε τομέα-κλειδί για την ανάπτυξη της χώρας. Ο Wen Jiabao θεώρησε το IoT ως μια τέλεια τεχνολογία για να εφαρμόσει σε όλες τις κορυφαίες στρατηγικές βιομηχανίες. Η ιδέα της απομακρυσμένης διοίκησης και αναγνώρισης διαφορετικών Πραγμάτων φάνηκε τέλεια για την ανάπτυξη των υποδομών και υπηρεσιών της Κίνας. Αυτή ήταν και η πρώτη φορά που αναγνωρίστηκε από μία κυβέρνηση σε όλο τον κόσμο το Διαδίκτυο των Πραγμάτων.

Το 2011, διοργανώθηκε από τις εταιρίες Facebook, Google, Yahoo, Akamai Technologies και Limelight Networks μια παγκόσμια ημέρα IPv6 ώστε να δώσει κίνητρο στους παρόχους του Διαδικτύου, τις εταιρίες που φιλοξενούν τον παγκόσμιο ιστό και γενικότερα την βιομηχανία αυτή να προετοιμάσουν την μετάβαση από IPv4

σε IPv6. Το νέο πρωτόκολλο προβλέπει $2^{128}$ διευθύνσεις ή όπως το έθεσε ο Steven Leibson: «θα μπορούσα να αναθέσουμε μία διεύθυνση IPv6 σε κάθε άτομο της επιφάνεια της γης και ακόμη να έχουν απομείνει αρκετές διευθύνσεις για να καλύψουμε το μέγεθος της γης άλλες100+ φορές» [8][9].

Η ανάπτυξη του Διαδικτύου των Πραγμάτων ανά τα χρόνια παρουσιάζεται ιδιαίτερα εντυπωσιακή. Σημαντικότατο, όμως, διαφαίνεται και το μέλλον του δικτύου αυτού. Η εταιρία τεχνολογικών αναλύσεων IDC προβλέπει μία αύξηση των συνδεμένων στο IoT συσκευών ή Πραγμάτων σε 41,6 δισεκατομμύρια μέχρι και το τέλος του 2025. Μία δεύτερη τεχνολογική εταιρία αναλύσεων, η Gartner, προβλέπει ότι τομείς όπως η αυτοκίνηση και οι επιχειρήσεις θα είναι οι κύριοι υπεύθυνοι για την αύξηση των συνδεδεμένων συσκευών σε 5,8 δισεκατομμύρια στο τέλος του 2020, ένα τέταρτο σχεδόν περισσότερη από ότι το 2019 [12].

## 1.4 Εφαρμογές του Διαδικτύου των Πραγμάτων

Το Διαδίκτυο των Πραγμάτων αφορά σε ένα ευρύτατο πεδίο τομέων της ανθρώπινης δραστηριότητας, από την κατανάλωση, τη βιομηχανία μέχρι και τη φροντίδα του περιβάλλοντος και τη συντήρηση και καλλωπισμό του οικιστικού συνόλου των πόλεων.

### 1.4.1.Το έξυπνο σπίτι

Το Διαδίκτυο των Πραγμάτων εμπεριέχει σε μεγάλο βαθμό την ιδέα υλοποίησης της αυτοματοποίησης των σπιτιών, της απομακρυσμένης δηλαδή παρακολούθησης και διαχείρισης των διαφόρων οικιακών συσκευών όπως των ηλεκτρονικών, του φωτισμού, της θέρμανσης και του κλιματισμού, των συστημάτων ασφαλείας και κλειστών συστημάτων παρακολούθησης των χώρων ακόμη και των συστημάτων ποτίσματος των φυτών [13],[14].

Ένα έξυπνο σπίτι θα μπορούσε να βασίζεται σε μία πλατφόρμα ή ακόμη και σε κόμβους που θα ελέγχουν τις διάφορες έξυπνες και οικιακές συσκευές[15]. Πλήθος εταιριών έχουν επιχειρήσει μέχρι στιγμής να δημιουργήσουν έναν όσο το δυνατό πληρέστερο εξοπλισμό ο οποίος θα παρέχεται στους κατασκευαστές σπιτιών προκειμένου για την υλοποίηση ενός τέτοιου εγχειρήματος. Η Apple με το HomeKit στοχεύει ακριβώς στη δημιουργία ενός σπιτιού του οποίου όλες οι συσκευές και τα εξαρτήματα θα ελέγχονται μέσα από μία εφαρμογή συσκευών iOS όπως το iPhone

και το Apple Watch [16],[17]. Η εταιρία Lenovo βασιζόμενη σε αυτήν την ιδέα ανακοίνωσε μία σειρά έξυπνων οικιακών συσκευών που ελέγχονται μέσω της εφαρμογής Apple's Home ήτοι Siri, την Lenovo's Smart Home Essentials [18]. Εκτός αυτών, έχουν δημιουργηθεί και πάρα πολλοί άλλοι κόμβοι έξυπνων οικιακών συσκευών που προσφέρονται ως αυτοδύναμες πλατφόρμες όπως τα Amazon Echo, Google Home, Apple's HomePod και SmartThings Hub της Samsung [19].

### 1.4.2. Υγεία και ιατροφαρμακευτική περίθαλψη

Το Διαδίκτυο των Πραγμάτων, σε αυτή του την έκφανση, μετουσιώνεται σε Διαδίκτυο των Ιατρικών Πραγμάτων ( Internet of Medical Things -IoMT) ώστε να εξυπηρετήσει σκοπούς σχετικούς με την ιατροφαρμακευτική περίθαλψη, την περισυλλογή δεδομένων και την χρησιμοποίηση αυτών για έρευνα και παρακολούθηση [20],[21],[22],[23],[24]. Το IoMT συχνά αναφέρεται και ως « Έξυπνο Ιατροφαρμακευτικό Σύστημα», ως η τεχνολογία για ψηφιοποίηση του ιατρικού συστήματος με την ένωση διαθέσιμων ιατρικών πηγών και ιατροφαρμακευτικών υπηρεσιών [25],[26].

Οι συσκευές του Διαδικτύου των Πραγμάτων μπορούν να χρησιμοποιηθούν για τη ενεργοποίηση της εξ’ αποστάσεως παρακολούθησης της υγείας αλλά και για συστήματα ειδοποίησης σε περίπτωση ανάγκης. Αυτές οι συσκευές παρακολούθησης της υγείας μπορεί να κυμαίνονται από πιεσόμετρα, ζώνες καρδιακών παλμών έως και προηγμένες συσκευές, ικανές να ελέγξουν εξειδικευμένα εμφυτεύματα, όπως βηματοδότες και ακουστικά βαρηκοΐας [27]. Ειδικά σχεδιασμένοι αισθητήρες μπορούν ακόμη και να εξοπλίσουν τα σπίτια των ηλικιωμένων πολιτών για τον έλεγχο της υγείας τους και της γενικότερης ευημερίας τους και παράλληλα να εξασφαλίσουν ότι τους παρέχεται η κατάλληλη θεραπευτική αλλά και η βοήθεια για ανάκτηση της ικανότητας κίνησης μέσω ειδικής θεραπείας [28]. Αυτοί οι αισθητήρες δημιουργούν ένα σύστημα έξυπνων αισθητήρων που είναι ικανοί να περισυλλέξουν, να επεξεργαστούν, να μεταφέρουν και να αναλύσουν πολύτιμες πληροφορίες σε διαφορετικά περιβάλλοντα, όπως συνδεδεμένες εσωτερικές συσκευές επιτήρησης στα νοσοκομειακά συστήματα [25].

Υπάρχουν, τέλος, και άλλες καταναλωτικές συσκευές που στοχεύουν στην βελτίωση του τρόπου ζωής των χρηστών και την εξυγίανση αυτού μέσω του εντοπισμού της καθημερινής δραστηριότητας και της κατάστασης της υγείας και συνακόλουθα της

αποτροπής των βλαβερών συνηθειών [29] ενώ πλατφόρμες παρακολούθησης του IoT είναι διαθέσιμες για προγεννητικούς και χρόνιους ασθενείς, βοηθώντας στους στην διαχείριση ζωτικής σημασίας θεμάτων της υγείας τους αλλά και των επαναλαμβανόμενων αναγκών για φαρμακευτική αγωγή [30].

### 1.4.3. Βιομηχανική Παραγωγή

Το Διαδίκτυο των Πραγμάτων πραγματοποιεί μία απρόσκοπτη ενσωμάτωση διαφορετικών παραγωγικών συσκευών εξοπλισμένες με ικανότητα αίσθησης, αναγνώρισης, επεξεργασίας, δραστηριοποίησης και δικτύωσης. Βασισμένο σε ένα άρρηκτα συνυφασμένο έξυπνο κυβερνο-φυσικό χώρο, το IoT ανοίγει νέους ορίζοντες για τη δημιουργία ευκαιριών για άσκηση καινούργιων επιχειρηματικών δραστηριοτήτων στην παραγωγική διαδικασία [31]. Τα έξυπνα συστήματα του Διαδικτύου των Πραγμάτων δίνουν την δυνατότητα για ταχύτατη παραγωγή νέων αγαθών, δυναμική γρήγορη απάντηση στην παραγωγική ζήτηση, βελτιστοποίηση σε πραγματικό χρόνο της παραγωγικής διαδικασίας και του δικτύου της εφοδιαστικής αλυσίδας, μέσω δικτυωμένων μηχανημάτων, αισθητήρων και συστημάτων ελέγχου [27].

Η τόνωση της παραγωγικότητας, η αποταμίευση πόρων και η μείωση των κερδών παραγωγής είναι κάποιοι μόνο από τους λόγους της επέκτασης του IoT στην βιομηχανία. Επιπροσθέτως, η διαχείριση των επενδυτικών αγαθών μέσω της προγνωστικής συντήρησης, της στατιστικής αξιολόγησης και των μετρήσεων μπορούν να οδηγήσουν σε μία τόνωση της αξιοπιστίας στην επιχείρηση. Τα βιομηχανικά συστήματα διοίκησης μπορούν να ενσωματώσουν smart grids επιτρέποντας βελτιστοποίηση σε πραγματικό χρόνο. Μετρήσεις, αυτοματοποιημένα συστήματα ελέγχου, βελτιστοποίηση των εργοστασιακών εγκαταστάσεων, βελτίωσης των περιβαλλοντικών πολιτικών ασφαλείας, συντήρηση της ποιότητας και της σταθερότητας στην παραγωγική διαδικασίας καθώς και πολλές άλλες λειτουργίες παρέχονται από έναν μεγάλο αριθμό αισθητήρων του δικτύου [27],[32].

Το Βιομηχανικό Διαδίκτυο των Πραγμάτων αυξάνει την δυναμική του σε τέτοιο βαθμό ώστε θα μπορούσε να οδηγήσει σε μία τόσο μεγάλη αύξηση της αξίας των επιχειρήσεων που μία Τέταρτη Βιομηχανική Επανάσταση να είναι φυσικό επακόλουθο. Η δυνητική αύξηση από τη εφαρμογή του IoT στην βιομηχανική

παραγωγή μπορεί να αυξήσει το Ακαθάριστο Εγχώριο Προϊόν σε 12 τρισεκατομμύρια δολάρια μέχρι το 2030 [33].

### 1.4.4. Έργα Υποδομής και Περιβάλλον

Το εγχείρημα βιώσιμων υποδομών τόσο στα αστικά κέντρα όσο και στην εξοχή, όπως γέφυρες, σιδηροδρομικές γραμμές και αιολικά πάρκα είναι ιδιαίτερο κρίσιμο για το Διαδίκτυο των Πραγμάτων [34]. Το IoT μπορεί να ωφελήσει την κατασκευή βιομηχανιών μέσω της μείωσης του κόστους, την μείωση του χρόνου, της καλύτερη ποιότητας των εργάσιμων ημερών, της μείωσης της γραφειοκρατίας και της αύξησης της παραγωγικότητας. Μπορεί, ακόμη, να βοηθήσει στην ανάληψη πιο γρήγορων αποφάσεων ενώ μπορεί να χρησιμοποιηθεί για τον προγραμματισμό επισκευών και των δραστηριοτήτων συντήρησης με έναν αποτελεσματικό τρόπο, μέσω του συντονισμού των δραστηριοτήτων μεταξύ διαφορετικών παρόχων υπηρεσιών και χρηστών αυτών των υποδομών [27].Η χρήση συσκευών του IoT είναι πολύ πιθανό να βελτιώσει τη διαχείριση επειγουσών περιστατικών, την ποιότητα των υπηρεσιών και να μειώσει το κόστος δραστηριοποίησης σε όλες της σχετιζόμενες με υποδομές τομείς [35]. Ακόμη και τομείς όπως η διαχείριση αποβλήτων μπορούν να επωφεληθούν από την αυτοματοποίηση και βελτιστοποίηση που μπορεί να επιφέρει το Διαδίκτυο των Πραγμάτων.

Εφαρμογές για επιτήρηση του IoT χρησιμοποιούν αισθητήρες προκειμένου για την προστασία του περιβάλλοντος μέσω του ελέγχου της ποιότητας του αέρα, του νερού, των ατμοσφαιρικών συνθηκών και των συνθηκών του εδάφους και μπορούν ακόμη και να συμπεριλάβουν τομείς όπως η επιτήρηση της πορείας της άγριας ζωής και του οικοτόπου [36],[37],[38],[39]. Άλλες εφαρμογές περιλαμβάνουν συστήματα έγκαιρης προειδοποίησης για σεισμούς και τσουνάμι ενώ αξιοσημείωτο είναι πως τέτοιες συσκευές εκτείνονται σε μεγάλες γεωγραφικές περιοχές ενώ μπορεί να είναι φορητές [27]. Η τυποποίηση του IoT που επέρχεται στους ασύρματους αισθητήρες έχει υποστηριχθεί πως θα επιφέρει μία επανάσταση σε αυτόν τον τομέα [40].

# 2.Task management in IoT

Σε αυτό το κεφάλαιο θα ασχοληθούμε με την διαχείριση εργασιών τόσο γενικά όσο και συγκεκριμένα στο Internet of Things. Θα αναπτύξουμε έννοιες όπως εργασία, κόμβος και πως οι όροι αυτοί συνδέονται μεταξύ τους. Οι σκοποί που επιτυγχάνονται με τη σωστή διαχείριση των διεργασιών και την αποδοτικότητα των συστημάτων καθώς και του Internet of Things.

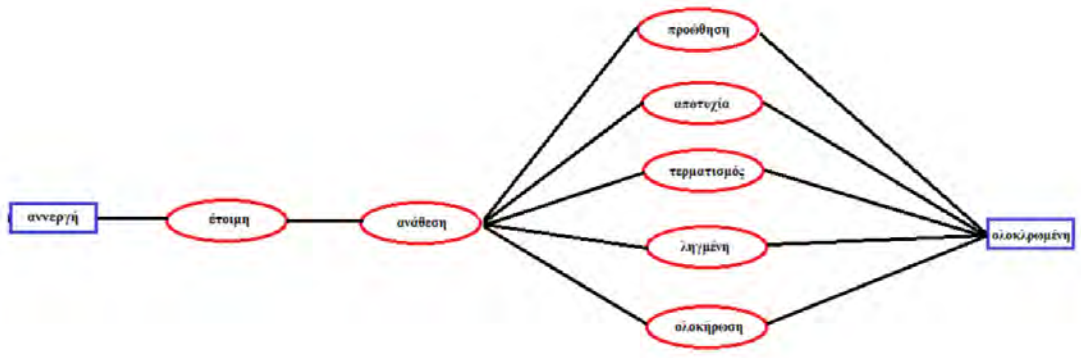
## 2.1 Task

Με τον όρο task στα ελληνικά εκφράζουμε μία εργασία, μία ενέργεια την οποία πρέπει να εκτελέσουμε είτε εμείς είτε να την διεκπεραιώσουμε με κάποιον τρόπο ώστε να ολοκληρωθεί. Στην πληροφορική με τον συγκεκριμένο όρο εκφράζουμε τα επιμέρους τμήματα μίας διεργασίας η οποία χωρίζεται σε μικρότερα τμήματα τα tasks.

Ξεκινώντας θα καλύψουμε τον ορισμό της διεργασίας, η οποία αποτελεί μέρος ενός μεγαλύτερου προγράμματος όπου μπορεί και να περιέχει περισσότερες από μία διεργασίες. Η διεργασία είναι υπεύθυνη για το κομμάτι ενός προγράμματος που εκτελείται είναι κατά κάποιο τρόπο το highlight ενός προγράμματος μίας και μέσα από την διεργασία διεκπεραιώνεται σταδιακά ολόκληρο ή κάποιο μέρος του προγράμματος.

Με την ίδια μοντελοποίηση και η διεργασία χωρίζεται σε μικρότερα τμήματα τα task επιτυγχάνοντας ταχύτερη και ευκολότερη διαχείριση. Τα tasks περιλαμβάνουν διάφορα στάδια - χαρακτηρισμούς ώστε να μπορεί ένα σύστημα να διαχωρίσει την κατάσταση που βρίσκονται βάση του χαρακτηρισμού τους. Πιο συγκεκριμένα ένα task έχει δύο καθοριστικούς χαρακτηρισμούς, έτσι κατατάσσεται προς εκτέλεση ή όχι. Οι χαρακτηρισμοί αυτοί είναι **ανενεργό** και **ολοκληρωμένο**, στην πρώτη περίπτωση δεν έχει εισέλθει ακόμα σε κατάσταση εκτέλεσης ενώ στην δεύτερη περίπτωση το task έχει εξέλθει από κατάσταση εκτέλεσης είτε αυτή ολοκληρώθηκε με επιτυχία είτε όχι. Εφόσον εισαχθεί ένα task σε ενεργή κατάσταση και είναι έτοιμο για εκτέλεση έχει συγκεκριμένα μονοπάτια μου μπορεί να ακολουθήσει για να προβεί τελικά σε κατάσταση ολοκλήρωσης. Αρχικά, οι όροι με τους οποίους γίνεται η μετάβαση από μία κατάσταση σε μία άλλη ορίζονται από το σύστημα όπου εκτελείται ένα task. Μπορεί να ακυρωθεί, οπότε να χαρακτηριστεί σε τερματική κατάσταση άρα να γίνει μετάβαση σε ολοκληρωμένο με αυτό τον τρόπο. Άλλη εκδοχή είναι να

παραμείνει για αρκετή ώρα σε ενεργή κατάσταση οπότε το σύστημα να τη χαρακτηρίσει ως 'ληγμένη' και έπειτα ολοκληρωμένη. Από την ενεργή κατάσταση η μετάβαση σε κατάσταση ανάθεσης γίνεται εφόσον υπάρξει εξουσιοδότηση, ενώ το αντίστροφο μπορεί να γίνει εφόσον μεσολαβήσει απόρριψη. Οι καταστάσεις - χαρακτηρισμοί που ακολουθούν αποτελούν και τα μονοπάτια που μπορεί να ακολουθήσει ένα task για να θεωρηθεί ολοκληρωμένο. Η κατάσταση τερματισμού που υποδηλώνει μία μη φυσιολογική ροή άρα γίνεται λόγος για κατά κάποιο τρόπο επιβαλλόμενο τερματισμό. Η κατάσταση - χαρακτηρισμός που αναπτύχθηκε και νωρίτερα αλλά με διαφορετικό μονοπάτι μετάβασης όταν ένα task λήγει, κάτι τέτοιο προκύπτει έπειτα από την μη τήρηση του χρονικού ορίου που έχει αποδοθεί. Σειρά λαμβάνει η φυσική ολοκλήρωση του task που στην πλειοψηφία αποτελεί και την ιδανική ροή για να χαρακτηριστεί ολοκληρωμένο. Για το τέλος έμειναν δύο χαρακτηρισμοί – καταστάσεις που λαμβάνει ένα task την αποτυχία ολοκλήρωσης και την προώθηση. Η πρώτη υποδηλώνει ότι η δομή του task δεν μπορεί να οδηγηθεί σε ολοκλήρωση ενώ, η δεύτερη έχει δύο έννοιες η πρώτη είναι ότι εφόσον μπήκε σε κατάσταση ανάθεσης ήταν έτοιμο για ολοκλήρωση ή η προώθηση δημιουργεί μία γέφυρα μεταξύ δύο ή περισσοτέρων tasks για να υπάρξει ομαδική ολοκλήρωση. Οι καταστάσεις – χαρακτηρισμοί που περιγράφονται μέχρι αυτό το σημείο σχηματικά αναπαρίστανται από την εικόνα 1. [50] [51]




*Εικόνα 1 Απεικόνιση των χαρακτηρισμών – καταστάσεων ενός task .*

## 2.2 Task management

Η διαχείριση των εργασιών είναι ίσως ένας από τους πιο καθοριστικούς παράγοντες για την αποδοτικότητα, την ευελιξία και την αποτελεσματικότητα ενός συστήματος. Αρχικά, πρέπει να τονιστεί ότι η διαχείριση των tasks αποτελεί και αυτό μία εργασία που το σύστημα καλείται να διεκπεραιώσει, περιλαμβάνει και αυτή η εργασία

επιμέρους εργασίες tasks, δημιουργώντας μία εμφωλευμένη ακολουθία με όσα έχουν αναφερθεί μέχρι τώρα για τα tasks. Όμως πριν γίνει αναφορά σε συστήματα και υπολογιστές είναι αξιοσημείωτο το γεγονός ότι η διαχείριση εργασιών βασίζεται και στην καθημερινή ζωή του ανθρώπου σε επίπεδο εταιριών και γενικώς σε περιβάλλοντας που αναπτύσσονται σχέσεις εργασιακές. Η διαχείριση εργασιών έχει σαν στόχο την ένωση των πόρων επιτυγχάνοντας πρόοδο στο γενικό έργο που έχει ανατεθεί οδεύοντας προς την ολοκλήρωση του.Με την ίδια ακριβώς λογική η διαχείριση εργασιών λειτουργεί σε συστήματα υπολογιστών. Για να γίνει αποτελεσματική η διαχείριση εργασιών πρέπει να ληφθούν υπόψιν αρκετές παράμετροι, η διαχείριση του χρόνου, η απόδοση προτεραιοτήτων σε κάθε εργασία και τέλος το αποτέλεσμα άρα την κατάσταση που μεταβαίνουν άλλες εργασίες. Υπάρχουν συγκεκριμένα στοιχεία που αποτελούν θεμελιώδεις αρχές για την αποδοτική διαχείριση εργασιών. Τα θεμελιώδη αυτά στοιχεία για την αποτελεσματική διαχείριση εργασιών είναι έξι και αναπτύσσονται παρακάτω. Πρώτο θεμελιώδες στοιχείο αποτελεί ο στόχος των εργασιών. Πιο συγκεκριμένα η στόχευση των εργασιών σε κομβικά σημεία του συνολικού έργου είναι πολύ σημαντικός παράγοντας για την γενική πρόοδο του έργου. Σε ένα υπολογιστικό σύστημα λαμβάνοντας κάποιες μετρικές υλοποιείται στοχευμένη παρατήρηση της προόδου των εργασιών με αποτέλεσμα τον καθορισμό σε ότι αφορά την πρόοδο. Δεύτερο θεμελιώδες στοιχείο αποτελεί η προτεραιότητα, είναι βασικό να μπορεί να διαχωριστεί πια εργασία προηγείται κάποιας άλλης λόγω κρισιμότητας στη συνολική πρόοδο του έργου. Σε ένα υπολογιστικό σύστημα η προτεραιότητα είναι αναγκαία και πρέπει να είναι πάντα προκαθορισμένη ώστε να μην οδηγείται τος σύστημα σε αδιέξοδα και αστοχίες. Τρίτο θεμελιώδες στοιχείο αποτελεί η κατηγοριοποίηση των εργασιών. Να καθορίζεται δηλαδή ποιος είναι ο ρόλος και ο στόχος της εργασίας αν αποτελεί κρίσιμη εργασία ώστε να της δοθεί προτεραιότητα και τέλος να είναι ευδιάκριτος ο στόχος της εργασίας αυτής. Σε ένα υπολογιστικό σύστημα οι διεργασίες πρέπει να ταξινομούνται με γνώμονα όσα προαναφέρθηκαν ώστε, να είναι σε θέση από μόνο του αν κληθεί να επιλέξει μία εργασία να το κάνει βάση του τύπου όπου ανήκει. Τέταρτο θεμελιώδες στοιχείο είναι η κατάσταση στην οποία βρίσκεται μία εργασία. Πιο συγκεκριμένα στον κύκλο ζωής μια εργασίας που έγινε αναφορά νωρίτερα και στην εικόνα υπάρχουν κάποιες καταστάσεις οι οποίες παίζουν καθοριστικό ρόλο για την πορεία τόσο της ίδιας της εργασίας όσο και για άλλες εργασίες που είναι αλληλένδετες με εκείνη. Το ίδιο συμβαίνει και σε ένα

υπολογιστικό σύστημα με τις εργασίες μόνο που εδώ είναι ζωτικής σημασία η τήρηση του συγκεκριμένου θεμελιώδους στοιχείου. Πέμπτο θεμελιώδες στοιχείο είναι η συνεργασία, να γίνεται κατανομή των εργασιών με βάση το φορτίο εργασία και συλλογικές ενέργειες για την περάτωση τους. Πρέπει αν χρειαστεί σε κάποιο σημείο να υπάρξει ενίσχυση για την υλοποίηση και αυτή η ενίσχυση να είναι πάντα υλοποιήσιμη σαν επιλογή. Σε υπολογιστικό επίπεδο αυτό πραγματοποιείται με το διαμοιρασμό εργασιών στους πόρους και με διάφορες τεχνικές ανίχνευσης σε επίπεδο φορτίου των διαθέσιμων πόρων, την ταχύτητα με την οποία μπορεί ένας πόρος να ανταποκριθεί καθώς και το κόστος. Λαμβάνοντας αυτές τις μετρικές το σύστημα πρέπει να κατανέμει συνεργατικά σε πόρους εργασίες ή να τις μεταφέρει από πόρο σε πόρο αν υπάρχει κάποιος καταλληλότερος διαθέσιμος. Έκτο και τελευταίο θεμελιώδες στοιχείο αποτελεί η παρακολούθηση της συμμόρφωσης. Η αναφορά γίνεται για εργασίες που έχουν παραμείνει για μεγάλο χρονικό διάστημα χωρίς να ολοκληρωθούν προκαλώντας γενική καθυστέρηση στην εκτέλεση της εργασίας. Με την καταγραφή συμβάντων και την καταγραφή αρχείου υπάρχει εικόνα γενικά για εργασίες που μένουν σε αρχικό στάδιο ενώ θα έπρεπε να έχουν υλοποιηθεί. Σε επίπεδο πληροφοριακών συστημάτων η παρακολούθηση έχει χαρακτήρα καταγραφής στα στάδια των tasks που αποτυγχάνουν να ολοκληρωθούν είτε λόγο χρόνου, είτε λόγο προώθησης σε άλλες εργασίες και στην συνέχεια να απορριφθούν από το σύστημα.

Η διαχείριση εργασιών τόσο στην κανονική ζωή των ανθρώπων όσο και στους υπολογιστές έχει θεμελιωθεί με συγκεκριμένες αρχές και τακτικές. Η αναφορά τους γίνεται με βάση τα πρότυπα που υπάρχουν διαθέσιμα από την βιβλιογραφία και σε αυτό το σημείο θα ακολουθήσει η συμπεριφορά του ίντερνετ των πραγμάτων σε ότι αφορά την διαχείριση εργασιών.

## 2.3 IoT και διαχείριση εργασιών

Το διαδίκτυο των πραγμάτων εξολοκλήρου σαν οντότητα αποτελεί ένα σύνολο συσκευών που αν και μετρά πολλά μέλη συμπεριφέρεται σαν ενιαία οντότητα. Οι συσκευές που απαρτίζουν το διαδίκτυο των πραγμάτων παρατηρήσαμε και νωρίτερα ότι ανήκουν σε διάφορες κατηγορίες, όπως κεντρικούς υπολογιστές, έξυπνους κόμβους, μετρητές αισθητήρες αλλά και κάθε είδους συσκευή που μπορεί να γίνει μέρος αυτής της μεγάλης οντότητας. Παρατηρώντας όλες αυτές τις συσκευές ευδιάκριτα διαχωρίζονται σε συσκευές που απλά πραγματοποιούν μετρήσεις,

συσκευές που επεξεργάζονται δεδομένα από άλλες συσκευές και συσκευές που πραγματοποιούν και τις δύο λειτουργίες. Είναι προφανές ότι δεν μπορεί στη διαχείριση εργασιών να ληφθούν υπόψιν συσκευές που δεν είναι σε θέση να εκτελέσουν εργασίες που θα τους ανατεθούν παρά μόνο μετρήσεις και αποστολή δεδομένων στο διαδίκτυο. Το γεγονός αυτό δεν της καθιστά λιγότερο σημαντικές από τις υπόλοιπες μιας και το διαδίκτυο των πραγμάτων δεν θα μπορούσε να λειτουργήσει χωρίς αυτές, ίσως να μην υπήρχε και η σύσταση του χωρίς τη συμμέτοχή τέτοιων συσκευών.

Εστιάζοντας στη διαχείριση εργασιών στο διαδίκτυο των πραγμάτων είναι κρίσιμο να σχολιάσουμε αναφερθούμε στο λειτουργικό σύστημα που διαχειρίζεται το πλήθος των συσκευών του δικτύου. Στην προηγούμενη ενότητα αναλύθηκαν κάποια στοιχεία και τεχνικές σχετικά με τα task και τις τεχνικές που η διαχείριση εργασιών πρέπει να ακολουθήσει για να εξασφαλιστεί ορθή λειτουργία. Το λειτουργικό σύστημα λειτουργεί με γνώμονα τις τεχνικές – αρχές αυτές με την προσθήκη πάντα λειτουργιών βασισμένων στις ανάγκες που δημιουργούνται, την κίνηση και το φορτίο του δικτύου, αλλά όλα αυτά βασίζονται στις συσκευές που απαρτίζουν το δίκτυο. Κάθε λειτουργικό σύστημα είναι σε θέση να διαχωρίσει την λειτουργία, τα χαρακτηριστικά (μνήμη, υπολογιστική ισχύ, αισθητήρες, ταχύτητα απόκρισης, φορτίο αλλά και κόστος επικοινωνίας). Στο σύνολο των συσκευών του δικτύου των πράγματων έρχονται σε πολλές περιπτώσεις και προστίθενται υπηρεσίες, αποθηκευτικοί χώροι καθώς και υπολογιστική ισχύ. Η προσθήκη αυτή είναι σημαντική διότι όλες αυτές οι προσθήκες μπορεί γεωγραφικά να απλώνονται ακόμα και σε ολόκληρη την υφήλιο.

Το διαδίκτυο των πραγμάτων έχει καταφέρει χάρη στην προσαρμογή του λειτουργικού συστήματος που διαχειρίζεται το δίκτυο, να επιτυγχάνει την ενσωμάτωση πολλαπλών συσκευών με διαφορές όχι μόνο σε λειτουργίες αλλά και σε μεγέθη καθώς και τοποθεσίες. Η διαχείριση των εργασιών κατανέμεται σε συσκευές που απαρτίζουν το δίκτυο ακόμα και αν ανήκουν στο υπολογιστικό νέφος (cloud computing). Η εξίσωση που καλείται να λύσει το λειτουργικό σύστημα είναι διαθέσιμες συσκευές που μπορούν να πραγματοποιήσουν τη συγκεκριμένη εργασία.Το φορτίο που έχουν τησυγκεκριμένη χρονική στιγμή, την ταχύτητα που έχουν και μπορούν να αποφέρουν αποτέλεσμα. Και όλα αυτά σε συνδυασμό με τις ήδη διαμοιρασμένες εργασίες καθώς και αυτές που πρόκειται να διαμοιραστούν, ώστε

να υπολογιστεί αν το αποτέλεσμα θα είναι διαθέσιμο στο επιθυμητό χρονικό διάστημα. Όλα τα παραπάνω περιβάλλονται με ένα κόστος όπου και αυτό είναι αρκετά υψηλά στη λίστα με τις σημαντικότερες παραμέτρους. Όσα αναφέρονται αποτελώντας κατά κάποιο τρόπο εμπόδια στην ομαλή λειτουργία του δικτύου, αποτελούν τον τρόπο λειτουργίας του λειτουργικού συστήματος σε ότι αφορά την διαχείριση εργασιών στο διαδίκτυο των πραγμάτων.

Αλγόρυθμοι που εντάσσονται σε εφαρμογές του διαδικρύου των πραγμάτων ανήκουν στην κατηγορία των πολυκρητιριακών. Τέτοιοι αλγόριθμοι έχουν συμπεριληφθοί και στην αφαρμογή που έχει αναπτυχθεί παρακάτω. Αλγόρυθμοι όπως οι AHP, TOPSIS και WPM λαμβάνοντας βάρη για κάθε κριτήριο που διαθέτει η κάθε εναλλακτική που προσφέρεται στην εφαρμογή. Η διαχείριση των πόρων για την κατανομή των εργασιών πραγματοποιείται λοιπόν με τις παραμέτρους που θέτει η εφαρμογή που εντάσσεται στο διαδίκτυο των πράγματων.Τέλος ο κάθε αλγόριθμος χρησιμοποιείται με βάση τον σχεδιασμό και τον σκοπό την εκάστοτε εφαρμογής. Έτσι κάθε εφαρμογή περιλαμβάνει έναν ή και περισσότερους αλγορύθμους που καλύπτουν τις εκάστοτε ενάγκες της εγαρμοφής.

Σε κάθε στάδιο και χρονική στιγμή, πριν από κάθε κίνηση γίνεται καταγραφή τόσο μετρικών όσο και στοιχείων που αφορούν την κατάσταση του δικτύου. Η καταγραφή αυτή γίνεται και σε τοπικό επίπεδο, πιο συγκεκριμένα σε κάθε κόμβο – συσκευή που είναι συνδεδεμένη στο δίκτυο. Στα πλαίσια της συγκεκριμένης πτυχιακής εργασίας δημιουργήθηκε μία προγραμματιστική εφαρμογή που σκοπό έχει να πραγματοποιήσει μετρήσεις προσομοιώνοντας όσα έχουν αναφερθεί μέχρι εδώ. Στο επόμενο κεφάλαιο γίνεται πιο ενδελεχής ανάλυση σχετικά με τα χαρακτηριστικά της εφαρμογής που έχει συμπεριληφθεί. [48],[49]

# 3. ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ

Στα πλαίσια της παραπάνω ανασκόπησης του Διαδικτύου των Πραγμάτων (IoT) καθώς και του Task scheduling στο Διαδίκτυο των Πραγμάτων (IoT), αναπτύχθηκε μία εφαρμογή με σκοπό τη μοντελοποίηση καθώς και την εξαγωγή αποτελεσμάτων προκειμένου να δημιουργηθούν κάποια συμπεράσματα. Τα συμπεράσματα αφορούν το Task scheduling στο Διαδίκτυο των Πραγμάτων (IoT) με τη χρήση συγκεκριμένων μεθόδων, τη χρήση της Java και μεθόδων – αλγορίθμων επιλογής αποφάσεων.

## 3.1. Περιγραφή της εφαρμογής

Με τη χρήση της αντικειμενοστραφούς γλώσσας προγραμματισμού Java αναπτύξαμε μία εφαρμογή που προσομοιώνει μεθόδους επιλογής βάση κριτηρίων.

Πιο συγκεκριμένα έχοντας δύο βασικά σενάρια που το καθένα επίκειται σε διαφορετικές παραμέτρους, επιτυγχάνουμε την προσομοίωση ενός δικτύου από κόμβους που είναι όλοι συνδεδεμένοι με όλους. Στα δύο αυτά σενάρια το πλήθος των κόμβων μεταβάλλεται ξεκινώντας με 50 κόμβους, στη συνέχεια γίνονται 100 και τέλος γίνονται 500. Και οι πέντε μέθοδοι εφαρμόζονται ξεχωριστά σε κάθε πλήθος κόμβου επιδιώκοντας την άντληση συμπερασμάτων. Το πείραμα αφορά και στα δύο σενάρια την προσομοίωση 1000 διεργασιών στο δίκτυο κάνοντας όμως διαφορετικό διαμοιρασμό σε κάθε σενάριο. Κάθε κόμβος έχει κάποια χαρακτηριστικά που τον προσδιορίζουν, έχει ταχύτητα (speed), φορτίο (load), αναγνωριστικό (id) καθώς επίσης κόστος επικοινωνίας με κάθε κόμβο (communication cost), ενώ κρατά και μία λίστα με τα στοιχεία όλων των κόμβων προκειμένου να επιτυγχάνεται ευκολότερα η διαδικασία επιλογής κόμβου για κάθε εργασία. Η ταχύτητα παράγεται τυχαία στο διάστημα [0, 1000], το ίδιο συμβαίνει και με το φορτίο στο διάστημα [0, 100] το αναγνωριστικό είναι ο αύξων αριθμός του κόμβου, ενώ το κόστος επικοινωνίας είναι μηδενικό σε περίπτωση που τρέξει τοπικά η εργασία ενώ παράγεται τυχαία για τους υπόλοιπους κόμβους στο διάστημα [0, 10000].

Στο πρώτο σενάριο γίνεται τυχαία επιλογή ενός κόμβου για την αποστολή της εργασίας. Στην συνέχεια και ενώ ανατρέξει ο αλγόριθμος-μέθοδο η εργασία αποστέλλεται στον καταλληλότερο κόμβο προκειμένου να διεκπεραιωθεί. Στο

συγκεκριμένο σενάριο όταν η εργασία αποσταλεί σε έναν κόμβο τότε αυξάνεται το φορτίο του κατά 0,05 ενώ κάθε 50 διεργασίες που εισάγονται στο δίκτυο αναδημιουργούμε τυχαία το φορτίο όλων των κόμβων.

Στο δεύτερο σενάριο η προσομοίωση είναι παραπλήσια. Πιο συγκεκριμένα με την αποστολή μιας εργασίας σε κάποιο κόμβο προστίθεται στο φορτίο του 0.05 , όμως σε αντίθεση με το πρώτο σενάριο για κάθε τρεις διεργασίες που εισάγονται στο δίκτυο το φορτίο όλων των κόμβων επεξεργάζεται. Η επεξεργασία αυτή φέρει αλλαγές στο φορτίο όλων των κόμβων στο διάστημα [-0.5, +0.5] που εξάγεται μέσω της μεθόδου random.

Για να υπάρξει ομοιομορφία στα αποτελέσματα και στα δύο σενάρια σώζεται λίστα με τα αρχικά χαρακτηριστικά κάθε κόμβου. Το γεγονός αυτό εξισώνει τα δύο σενάρια καθώς μεταξύ του πλήθους των κόμβων που τρέχει κάθε σενάριο επικρατεί ομοιογένεια και οι κόμβοι έχουν στο ξεκίνημα κάθε σεναρίου ίδια χαρακτηριστικά (φορτίο, ταχύτητα, κόστος). Θα ακολουθήσει παρακάτω πιο ενδελεχείς ανάλυση της εφαρμογής και του κώδικα, των μεθόδων, των εργαλείων ανάπτυξης καθώς και των αποτελεσμάτων.

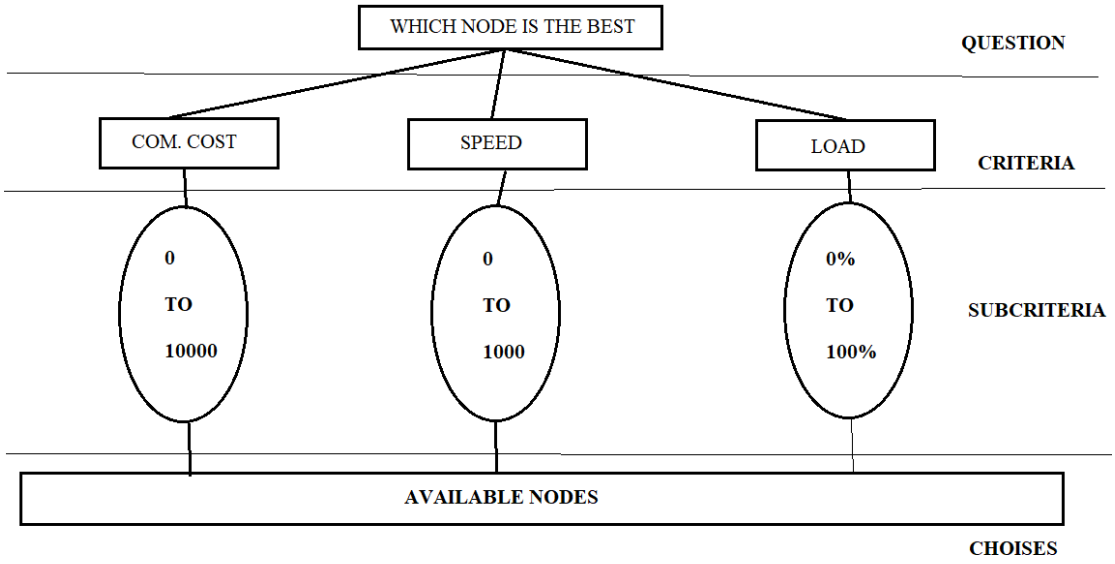
## 3.2 Μέθοδοι επιλογής κόμβων

Στο σημείο αυτό θα αναλύσουμε τις μεθόδους – αλγορίθμους που έχουν χρησιμοποιηθεί στην εφαρμογή. Με την χρήση τριών γνωστών μεθόδων - αλγορίθμων καθώς και με άλλες δύο που βασίζονται κυρίως στην λογική η εφαρμογή είναι σε θέση να μοντελοποίηση ένα δίκτυο με 50,100 και 500 κόμβων πραγματοποιώντας μετρήσεις σύμφωνα με το εκάστοτε σενάριο.

### 3.2.1 AHP

Η Analytic hierarchy process γνωστή και με τα αρχικά AHP με τον όρο στα ελληνικά να μεταφράζεται ως διαδικασία αναλυτικής ιεραρχίας. Η συγκεκριμένη μέθοδος βασίζεται κατά κύριο λόγο στα μαθηματικά και την ψυχολογία και χρησιμοποιείται για το δομημένο τρόπο λήψης, καθώς και την ανάλυση σύνθετων αποφάσεων. Εμπνευστής της «για λόγους συντομίας» AHP είναι ο Thomas L. Saaty όπου την σχεδίασε μέσα στην δεκαετία του 1970, σε συνεργασία με τον Ernest Forman περί το 1983 δημιούργησαν το Expert Choice όπου τον οδήγηση στην εκτενέστερη ανάλυση της μεθόδου με αποτέλεσμα την τελειοποίηση της,

Η Analytic hierarchy process χρησιμοποιείται μέχρι και σήμερα για να δώσει λύση σε πολυκριτηριακά προβλήματα χάρη στη δομή και στον τρόπο που αναπτύσσεται η μέθοδος και τους τομείς που χωρίζει το πρόβλημα.




*Εικόνα 2 απεικόνιση του δέντρου – διαγράμματος που δημιουργείται στην Analytic hierarchy process στο πρόβλημα «Επιλογής Κόμβου».*

Πιο συγκεκριμένα δημιουργείται μία ιεραρχία με το αρχικό ερώτημα – ζήτημα να τοποθετείται στην κορυφή ενός δέντρο, ακριβώς από κάτω να τοποθετούνται τα κριτήρια που υπάρχουν και στην βάση - φύλλα του δέντρου τοποθετούνται οι εναλλακτικές που υπάρχουν. Μία τέτοια ιεραρχική μορφή προβλημάτων μας οδηγεί στο συμπέρασμα ότι τα υψηλότερα επίπεδα επηρεάζουν τις αποφάσεις μας για τα κατώτερα επίπεδα. Στη συγκεκριμένη μέθοδο όμως δεν υπάρχει μία συγκεκριμένη δομή που να μας επιτρέπει απλά να τοποθετούμε τα κριτήρια και τα υποκριτήρια του προβλήματος. Αντ' αυτού όποιος θέτει το πρόβλημα πρέπει να προσδιορίσει με βάση τη θεμελιώδη κλίμακα της Analytic hierarchy process όπως φαίνεται παρακάτω, για τις διμερείς συγκρίσεις κάθε κριτηρίου.

| Κλίμακα | Σημασία | Επεξήγηση |
|---|---|---|
| **1** | Ίση σημασία | Τα δύο στοιχεία είναι ισότιμα |
| **2** | Ασθενές | Ενδιάμεση τιμή |
| **3** | Μέτρια σημασία | Το ένα στοιχείο έχει ένα πλεονέκτημα |
| **4** | Μέτρια προς ισχυρή σημασία | Ενδιάμεση τιμή |
| **5** | Ισχυρή σημασία | Το ένα στοιχείο είναι σαφώς καλύτερο από το άλλο |
| **6** | Αρκετά ισχυρή | Ενδιάμεση τιμή |
| **7** | Πολύ ισχυρή σημασία | Το ένα στοιχείο είναι πολύ καλύτερο |
| **8** | Πάρα πολύ ισχυρή σημασία | Ενδιάμεση τιμή |
| **9** | Εξαιρετικά σημαντικό | Το ένα στοιχείο είναι πανίσχυρο έναντι του άλλου |

*Πίνακας 3 Θεμελιώδης κλίμακα Analytic hierarchy process.*

Το επόμενο στάδιο της μεθόδου είναι η δημιουργία ενός νέου πίνακα, με δεδομένη την βαθμονόμηση μεταξύ των διμερών συγκρίσεων μεταξύ των κριτηρίων. Δημιουργείται ένας διαγώνιος πίνακας, όπως απεικονίζεται στον πίνακα 3.2. Παράλληλα υπολογίζεται το σύνολο κάθε στήλης του πίνακα που χρησιμοποιείται σε επόμενο στάδιο της Analytic hierarchy process.

| | load | cost | speed |
|---|---|---|---|
| load | 1,00 | 3,00 | 5,00 |
| cost | 0,33 | 1,00 | 3,00 |
| speed | 0,20 | 0,33 | 1,00 |
| total | 1,53 | 4,33 | 9,00 |

*Πίνακας 4 Διαγώνιος πίνακας βαρών μεταξύ των κριτηρίων επιλογής κόμβου, έπειτα από τον προσδιορισμό βάση της θεμελιώδους κλίμακας στις διμερής σχέσεις των κριτηρίων.*

Επόμενο στάδιο της μεθόδου είναι η κανονικοποίηση του πίνακα 4. Η κανονικοποίηση προκύπτει από την διαίρεση κάθε στοιχείου της στήλης με το άθροισμα που υπολογίστηκε νωρίτερα. Ο πίνακας που δημιουργείτε ονομάζεσαι κανονικοποιημένος, εφόσον υπολογιστεί ο μέσος όρος κάθε γραμμής το αποτέλεσμα είναι το βάρος κάθε κριτηρίου.

| | load | cost | speed | | Weights |
|---|---|---|---|---|---|
| Load | 0,6535 | 0,69 | 0,56 | | 0,6345 |
| cost | 0,2156 | 0,23 | 0,33 | | 0,2585 |
| speed | 0,1307 | 0,08 | 0,11 | | 0,1069 |

*Πίνακας 5 Κανονικοποιημένος πίνακας και υπολογισμός βαρών κάθε κριτηρίου στο πρόβλημα επιλογής κόμβου.*

Επόμενο βήμα είναι ο υπολογισμός του δείκτη συνέπειας που ορίζεται ως CI (consistency index) και υπολογίζεται με βάση τον παρακάτω τύπο. $CI = \frac{\lambda \max - n}{\mathrm{n}-1}$ ως λmax ορίζεται το άθροισμα του βάρους κάθε κριτηρίου πολλαπλασιασμένο με το άθροισμα που έχει υπολογιστή στον πίνακα 4, η μεταβλητή n είναι ο αριθμός των κριτηρίων. Επόμενο στάδιο που αφορά τον έλεγχο των αποτελεσμάτων είναι το ποσοστό συνέπειας που απεικονίζεται ως (consistency rate) και υπολογίζεται από τον τύπο $CR = \frac{CI}{RI} < 0.1{\sim}10\%$ , επομένως για να είναι έγκυρο το αποτέλεσμα πρέπει το

αποτέλεσμα στο συγκεκριμένο τύπο δηλαδή το ποσοστό συνέπειας να είναι μικρότερο του 10%. Σε περίπτωση μη έγκυρου ποσοστού πρέπει να βαθμονομηθούν εκ νέου οι διμερείς σχέσεις των κριτηρίων.

| N | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| RI | 0 | 0 | 0.58 | 0.9 | 1.12 | 1.24 | 1.32 | 1.41 | 1.45 | 1.49 |

*Πίνακας 6. Τιμές random consistency index σε συνάρτηση με τε το πλήθος των κριτηρίων που αποδίδονται ως N.*

#### *3.2.1.1 Προγραμματιστική ανάλυση της AHP*

Σε προγραμματιστικό επίπεδο έγινε προσπάθεια η Analytic hierarchy process να συνταχθεί με κώδικα και τεχνικές απλές, προκειμένου μελλοντικά να είναι εύκολη η παραμετροποίηση του.

Ξεκινώντας την ανάπτυξη της Analytic hierarchy process καλώντας την κλάση που δημιουργήσαμε δίνουμε ως όρισμα τον αριθμό των κριτηρίων της μεθόδου. Το γεγονός αυτό εξυπηρετεί στη δημιουργία των επιμέρους πινάκων που χρειάζεται η μέθοδος προκειμένου να υλοποιηθούν όλα τα στάδια της. Παράλληλα στην κύρια κλάση της εφαρμογής στο τμήμα που καλείται η Analytic hierarchy process δημιουργείται ένας πίνακας που με στατικό τρόπο παίρνει τιμές και τις στέλνει στην μέθοδο, ώστε να καθοριστούν οι προτεραιότητες βάση της θεμελιώδους κλίμακας στις διμερείς σχέσεις των κριτηρίων. Ουσιαστικά όλα τα παραπάνω στάδια προετοιμάζουν την κλάση της Analytic hierarchy process ώστε να είναι έτοιμη να δεχθεί τους κόμβους να εκτελέσει τις συγκρίσεις δίνοντας ως αποτέλεσμα τον επικρατέστερο κόμβο. Κάνοντας κλήση της υπομεθόδου ‘set rank’ και δίνοντας ως όρισμα τη λίστα με τους διαθέσιμους κόμβους, πιο συγκεκριμένα τη λίστα του κόμβου που έχει σταλεί η εργασία ώστε να βρεθεί βάση της Analytic hierarchy process ποιος είναι ο καταλληλότερος κόμβος για να εκτελεσθεί. Εντός της κλάσης Analytic hierarchy process υλοποιείται ότι ακριβώς έχει αναλυθεί στη θεωρητική ανάπτυξη της μεθόδου. Πριν την εξαγωγή του αποτελέσματος, δηλαδή του κόμβου που έχει επιλεγεί γίνεται και έλεγχος του δείκτη ποσοστού συνέπειας προκειμένου να εξασφαλιστεί η εγκυρότητα του αποτελέσματος. Κλείνοντας να σημειωθεί ότι δημιουργείται λίστα ταξινομημένη σε κάθε κλήση της μεθόδου με όλους τους κόμβους αλλά στην εφαρμογή αξιοποιούμε μόνο τον πρώτο κατά συνέπεια τον επικρατέστερο. [41],[42]

### 3.2.2 TOPSIS

Όπως η Analytic hierarchy process έτσι και η TOPSIS ανήκουν στην κατηγορία των μεθόδων αποφάσεων επίλυσης - ανάλυσης πολυκριτηριακών προβλημάτων. Η Technique for Order of Preference by Similarity to Ideal Solution γνωστή με τα αρχικά TOPSIS, με τη μετάφραση του όρου στα ελληνικά προσεγγιστικά να είναι η τεχνική για τη διάταξη των προτιμήσεων βάσει της ομοιότητας με την ιδεατή λύση. Η Technique for Order of Preference by Similarity to Ideal Solution αναπτύχθηκε από τους Yoon και Hwang το 1981 σαν μία ακόμα εναλλακτική στην οικογένεια μεθόδων ELECTRE. Με την TOPSIS επιδιώκουμε την σύγκριση της ιδεατής λύσης με την πραγματική.

Αναλύοντας τα βήματα της TOPSIS ξεκινάμε με την εισαγωγή των εναλλακτικών που έχουμε, όπου στην περίπτωση μας είναι τα χαρακτηρίστηκα των κόμβων. Οι κόμβοι αποτελούν τις εναλλακτικές και τα χαρακτηριστικά τους τα κριτήρια, όπου στον πίνακα που σχηματίζεται δίνεται η ονομασία του πίνακα απόδοσης.

| | Load | Speed | cost |
|---|---|---|---|
| Node 1 | 90 | 595 | 9500 |
| Node 2 | 15 | 200 | 5000 |
| Node 3 | 50 | 1000 | 10000 |
| Node 4 | 88 | 20 | 100 |

*Πίνακας 7 αναπαράσταση του πίνακα απόδοσης.*

Επόμενο βήμα της μεθόδου TOPSIS είναι η κανονικοποίηση του πίνακα απόδοσης. Το συγκεκριμένο βήμα επιτυγχάνεται εφαρμόζοντας το μαθηματικό τύπο

$r_{ij} = \frac{x_{ij}}{\sum_{i=1}^{m} x_{ij}^2}$. Όπου $x_{ij}$ είναι το κριτήριο και το $r_{ij}$ είναι η κανονικοποιοημένη απόδοση του αρχικού μας πίνακα σε σχέση με το χαρακτηριστικό $x_j$. Στη συνέχεια υπολογίζουμε τον σταθμισμένο κανονικοποιημένο πίνακα, αυτό επιτυγχάνεται με τον πολλαπλασιασμό του κανονικοποιημένου πίνακα με τον πίνακα βαρών. Ως πίνακα βαρών ορίζουμε ένα πίνακα που περιλαμβάνει την βαρυτική απόδοση ενός είδους χαρακτηριστικών για το σύνολο των εναλλακτικών.

| | load | Speed | Cost |
|---|---|---|---|
| Weight | 0.60 | 0.15 | 0.25 |

*Πίνακας 8 αναπαράσταση του πίνακα βαρών.*

Σειρά έχει η εύρεση της ιδανικής και της αρνητικά ιδανικής λύσης μέσα από τον πίνακα που προέκυψε σε προηγούμενο βήμα. Αναζητώντας στο σταθμισμένο κανονικοποιημένο πίνακα τη μέγιστη και ελάχιστη τιμή του κάθε χαρακτηριστικού από αυτά που περιλαμβάνει το πρόβλημα που θέλουμε να επιλύσουμε. Η εύρεση μέγιστης και ελάχιστης τιμής μας οδηγεί στο επόμενο βήμα της μεθόδου TOPSIS που είναι η εύρεση των μέτρων απόστασης των τιμών αυτών. Πιο συγκεκριμένα χρησιμοποιούμε τους παρακάτω μαθηματικούς τύπους: $S_i^+ = \sqrt{\sum_{j=1}^{n}(a_{ij} - a_j^+)^2}$ ο συγκεκριμένος τύπος εκφράζει την απόσταση κάθε εναλλακτικής από την ιδανική λύση, ενώ ο τύπος $S_i^- = \sqrt{\sum_{j=1}^{n}(a_{ij} - a_j^-)^2}$ εκφράζει την απόσταση της κάθε εναλλακτικής από την ιδανικά αρνητική λύση. Στους δύο τύπους περιλαμβάνονται, $a_j^+$ η μέγιστη τιμή κάθε χαρακτηριστικού και αντίστοιχα με $a_j^-$ την ελάχιστη, ενώ με $a_{ij}$ αντιπροσωπεύουμε τα στοιχεία του σταθμισμένου κανονικοποιημένου πίνακα. Τελευταίο βήμα για την ολοκλήρωση της μεθόδου είναι ο υπολογισμός της σχετικής κοντινότητας στην ιδανική λύση. $K_i = \frac{S_i^-}{S_i^+ + S_i^-}$ Ο συγκεκριμένος τύπος επιλύει το πρόβλημα μας στο τελικό του στάδιο με το $K_i$ να κυμαίνεται μεταξύ του 0 και του 1, συμπερασματικά λοιπόν όσο πιο κοντά στο 1 είναι κάποια εναλλακτική τόσο μεγαλύτερο βαθμό προτεραιότητας λαμβάνει δίνοντας μας τον ιδανικότερο κόμβο.[43],[44],[45]

### 3.2.2.1 Προγραμματιστική ανάπτυξη *TOPSIS*

Όπως και με την Analytic hierarchy process έτσι και στην TOPSIS επιλέξαμε μορφή και τεχνικές ώστε ο κώδικας να είναι κατανοητός και εύκολα παραμετροποιήσιμος. Η χρήση της μεθόδου γίνεται καλώντας την ομώνυμη κλάση TOPSIS και δίνοντας ως όρισμα την λίστα με τους διαθέσιμους κόμβους που περιλαμβάνει όλα τους τα χαρακτηριστικά. Επόμενη ενέργεια είναι η δημιουργία του κανονικοποιημένου πίνακα όπου γίνεται με τη χρήση της math.pow καθώς και κατάλληλη χρήση δομών επανάληψης για την επιθυμητή διάσχιση του πίνακα. Τα βάρη έχοντας οριστεί όπως απεικονίζονται στον πίνακα 8 πολλαπλασιάζονται με τον κανονικοποιημένο πίνακα

και έτσι δημιουργήται ο σταθμισμένος κανονικοποιημένος πίνακας. Επόμενο βήμα είναι η εύρεση μέγιστης και ελάχιστης τιμής ανά χαρακτηριστικό με την χρήση δομών επανάληψης και ο προσδιορισμός της λύσης, αρνητικής και μη. Κάνοντας χρήση και σε αυτό το σημείο της math.pow που υπολογίζει τις δυνάμεις με κατάλληλη τροποποίηση υπολογίζουμε και τις ρίζες των μέτρων απόστασης. Τέλος δημιουργούμε έναν πίνακα που φιλοξενεί την τελική κατάταξη δηλαδή τα αποτελέσματα της σχετικής κοντινότητας στην ιδανική λύση, εφόσον ταξινομηθεί επιστρέφει το πρώτο στοιχείο του πίνακα που είναι και ο κόμβος που αναζητούμε.

### 3.2.3 Weighted Product Model

Η τρίτη κατά σειρά μέθοδος που αναλύουμε και εντάσσουμε στην εφαρμογή μας είναι η weighted product method η οποία ανήκει και αυτή στις μεθόδους λήψης αποφάσεων σε πολυκριτηριακό επίπεδο. Δεν υπάρχουν αναφορές που να υποδηλώνουν τον εμπνευστή της μεθόδου που είναι γνωστή με τα αρχικά WPM. Ο τρόπος που εξελίσσεται η συγκεκριμένη μέθοδος δεν διαφέρει και πολύ από της προηγούμενες με μία μικρή διαφορά σε ότι αφορά τον χαρακτηρισμό των χαρακτηριστικών σε ωφέλημα και μη. Ξεκινώντας όμως η μέθοδος αρχικά πρέπει να λάβει το σύνολο των εναλλακτικών που έχει με τα χαρακτηριστικά που περιλαμβάνουν. Σε δεύτερο επίπεδο πρέπει να γίνει ο χαρακτηρισμός που αναφέραμε νωρίτερα στα χαρακτηριστικά, όμως ο χαρακτηρισμός αυτός δεν έχει να κάνει με το χαρακτηριστικό αν μας ωφελεί ή όχι αλλά αν εμείς θέλουμε να έχει όσο το δυνατόν υψηλότερη τιμή για να χαρακτηριστεί ωφέλιμο ή , όσο το δυνατόν χαμηλότερη τιμή για να χαρακτηριστεί μη ωφέλιμο. Επομένως όπως αναπαρίσταται και στον παρακάτω πίνακα έχουμε το χαρακτηρισμό των χαρακτηριστικών μας.

| | Load | Speed | Cost |
|---|---|---|---|
| Χαρακτηρισμός | Μη ωφέλιμο | Ωφέλιμο | Μη ωφέλιμο |
| Node 1 | 90 | **5000** | 7000 |
| Node2 | **50** | 800 | 9820 |
| Node 3 | 75 | 50 | **200** |

*Πίνακας 9 απεικόνιση των χαρακτηρισμών των χαρακτηριστικών του προβλήματος.*

Το χαρακτηριστικό ‘ταχύτητα’ πήρε το χαρακτηρισμό ωφέλιμο μιας και επιδιώκουμε για τη μέγιστη τιμή του. Ενώ τα χαρακτηριστικά φορτίο και κόστος πήραν τον χαρακτηρισμό του μη ωφέλιμου διότι επιδιώκουμε να έχουμε τις ελάχιστες τιμές. Βρίσκουμε αντίστοιχα μέγιστη τιμή στα ωφέλιμα χαρακτηριστικά και ελάχιστη στα μη ωφέλιμα και διαιρούμε με το ελάχιστο ή το μέγιστο κάθε στήλης - χαρακτηριστικό.

| | Load | Speed | Cost |
|---|---|---|---|
| Χαρακτηρισμός | Μη ωφέλιμο | Ωφέλιμο | Μη ωφέλιμο |
| Node 1 | 90/50 | **5000/5000** | 7000/200 |
| Node2 | **50/50** | 800/5000 | 9820/200 |
| Node 3 | 75/50 | 50/5000 | **200/200** |

*Πίνακας 10 απεικόνιση της εύρεσης μεγίστων και ελαχίστων τιμών ανά χαρακτηριστικό ανάλογα με τον χαρακτηρισμό που διαθέτει..*

Με τον τρόπο αυτό δημιουργείται ο κανονικοποιημένος πίνακας που περιλαμβάνει τις εναλλακτικές του προβλήματος, δηλαδή τους κόμβους και τα χαρακτηριστικά δηλαδή τα κριτήρια. Επόμενο βήμα είναι ο προσδιορισμός των βαρών, ο πίνακας αυτός είναι ίδιος με τη μέθοδο TOPSIS όπως απεικονίζεται στον πίνακα 8.. Συγκεντρώνοντας όλες τις πληροφορίες έχουμε τη δημιουργία του κανονικοποιημένου πίνακα αποφάσεων, όπου θα υψώσουμε σε δύναμη το βάρος που έχει αποδοθεί σε κάθε χαρακτηριστικό.$Index = x_{i1}^{weight_1}, x_{i2}^{weight_2} x_{i3}^{weight_3} x_{in}^{weight_n}$.

| | Load | Speed | Cost |
|---|---|---|---|
| weight | 0.60 | 0.15 | 0.25 |
| Node 1 | $1.8^{0.6}$ | $1^{0.15}$ | $35^{0.25}$ |
| Node2 | $1^{0.6}$ | $0.16^{0.15}$ | $49.1^{0.25}$ |
| Node 3 | $1.5^{0.6}$ | $0.01^{0.15}$ | $1^{0.25}$ |

*Πίνακας 11 απεικόνιση των χαρακτηρισμών των χαρακτηριστικών του προβλήματος*

Για να φτάσουμε στη λύση δηλαδή την εύρεση του καταλληλότερου κόμβου αρκεί να υπολογιστούν οι δυνάμεις που αποδόθηκαν με βάση τα βάρη κάθε χαρακτηριστικού. Τέλος πολλαπλασιάζουμε τα χαρακτηριστικά κάθε εναλλακτικής δημιουργώντας ένα δείκτη απόδοσης που μας βοηθάει να παράξουμε την τελική κατάταξη των εναλλακτικών. $PS = x_{i1} * x_{i2} * x_{i3} * x_{in}$

| | Load | Speed | Cost | Performance score |
|---|---|---|---|---|
| Node 1 | 1.42286436586 | 1 | 2.43229927909 | **3.4608316** |
| Node2 | 1 | 0.75965779293 | 0.63245553203 | **0.4804496** |
| Node 3 | 1.27542450062 | 0.50118723362 | 1 | **0.6392264** |

*Πίνακας 12 απεικόνιση του δείκτη απόδοσης .*

Όσο υψηλότερο δείκτη απόδοσης έχει μία εναλλακτική – κόμβος τόσο πιο κατάλληλος είναι. Ο κόμβος με το μέγιστο δείκτη απόδοσης είναι αυτός που επιλέγει το σύστημα ως το καταλληλότερο.[46],[47]

#### 3.2.3.1 Προγραμματιστική ανάπτυξη WPM

Στα ίδια προγραμματιστικά πλαίσια που κυμαίνονται οι προηγούμενες μέθοδοι είναι και η weighted product method. Χρησιμοποιήθηκαν όσο το δυνατόν πιο απλές μορφές και δομές προκειμένου να είναι κατανοητό κάθε βήμα και εύκολα παραμετροποιήσιμο. Ξεκινώντας κατά την κλήση της κλάσης WPM ως όρισμα δίνουμε τη λίστα των διαθέσιμων κόμβων που περιλαμβάνουν όλα τους τα χαρακτηριστικά. Σε πρώτο στάδιο διαβάζουμε όλη τη λίστα βρίσκοντας σε κάθε χαρακτηριστικό μέγιστη και ελάχιστη τιμή. Με αυτό τον τρόπο δημιουργούμε δυο πίνακες έναν με τα μέγιστα κάθε χαρακτηριστικού και ένα με τις ελάχιστες τιμές. Στη συνέχεια αξιοποιώντας ανάλογα το χαρακτηρισμό του κάθε χαρακτηριστικού στη δημιουργία του κανονικοποιημένου πίνακα, εκτελώντας τις αντίστοιχες πράξεις που αναφερθήκαμε νωρίτερα στην ανάλυση της μεθόδου WPM. Σε αυτό το σημείο υψώνουμε κάθε στοιχείο του πίνακα που προέκυψε στη δύναμη που έχει προκύψει βάση του βάρους που έχει αποδοθεί σε κάθε χαρακτηριστικό. Τελευταίο βήμα της μεθόδου είναι ο υπολογισμός του δείκτη απόδοσης κάθε εναλλακτικής, δηλαδή κάθε κόμβου. Εφόσον ταξινομήσουμε τον τελικό πίνακα εξάγουμε τον κόμβο που είναι ο καταλληλότερος βάση του δείκτη απόδοσής του.

### 3.2.4 Lower load

Ξεκινώντας αναφέραμε ότι έγινε χρήση δύο μεθόδων - αλγορίθμων που βασίζονται στη λογική. Η συγκεκριμένη μέθοδος όπως υποδηλώνει και το όνομα της, επιλέγει κόμβους για να στείλει την κάθε εργασία να εκτελεσθεί λαμβάνοντας υπόψιν μόνο το φορτίο κάθε κόμβου. Έχοντας ως μοναδικό κριτήριο το φορτίο η LOWER LOAD μέθοδος δεν χρειάζεται να κάνει πολλούς ελέγχους στο δίκτυο, επιτυγχάνοντας μία γρήγορη και με χαμηλό κόστος εξαγωγή αποτελεσμάτων.

Ειδικότερα καλούμε την LOWER LOAD για 50,100 αλλά και για 500 κόμβους καθώς επίσης και για τα δύο σενάρια. Κατά την κατασκευή της προγραμματίστηκε να λαμβάνει ως όρισμα την λίστα με τους κόμβους η οποία περιέχει όλα τα χαρακτηρίστηκα των κόμβων του δικτύου. Στη συνέχεια διαπερνά την λίστα εστιάζοντας στο να αναδείξει το κόμβο με το χαμηλότερο φορτίο load, επιστρέφοντας στη βασική μας μέθοδο (της εφαρμογής) το αναγνωριστικό id του κόμβου. Με αυτό τον τρόπο δίνει τη σκυτάλη σε άλλες μεθόδους (της εφαρμογής) να ενημερώσουν την λίστα και να αλλάξουν τις παραμέτρους του κόμβου που επιλέχθηκε για να αποσταλεί και να εκτελεσθεί η εργασία.

### 3.2.5 Highest Speed

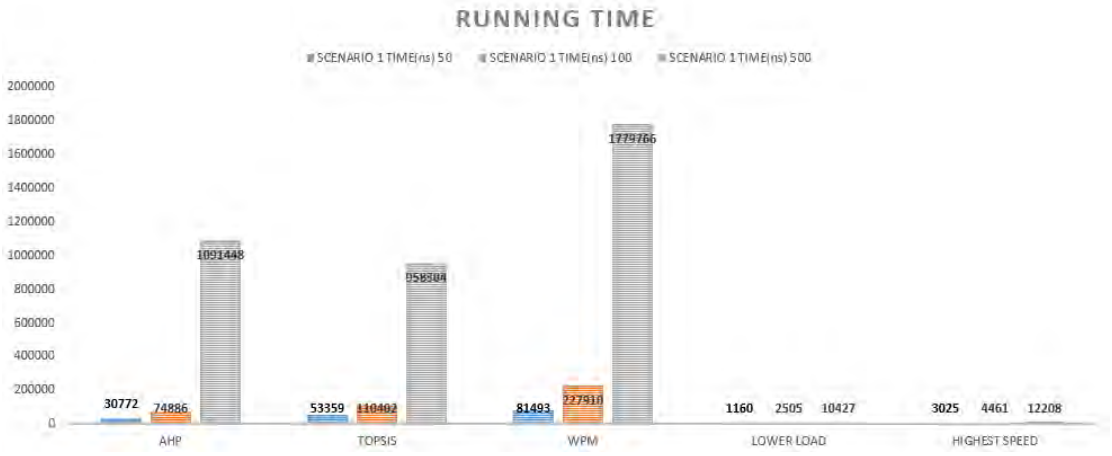
Η δεύτερη μέθοδος – αλγόριθμος που βασίζεται στη λογική και τελευταία για την εφαρμογή είναι εκείνη της υψηλότερης ταχύτητας. Όπως και η προηγούμενη έτσι και η συγκεκριμένη μέθοδος – αλγόριθμος όπως και η ονομασία της προδίδει ότι επιλέγει κόμβο για να αποστείλει για εκτέλεση την εργασία με κριτήριο την ταχύτητα. Η Highest Speed μέθοδος δεν κάνει πολλούς ελέγχους σε κριτήρια και χαρακτηρίστηκα των κόμβων παρά μόνο στην ταχύτητα επιλέγοντας τον ταχύτερο.

Κάνοντας κλήση της μεθόδου Highest Speed στην εφαρμογή μας εκτελείται για 50,100,500κόμβους καθώς επίσης και για τα δυο σενάρια. Όπως και η μέθοδος LOWER LOAD έτσι και η Highest Speed κατασκευάστηκε έχοντας ως όρισμα κατά την κλήση της μία λίστα, την λίστα με τους κόμβους πιο συγκεκριμένα προκειμένου να την διαπεράσει. Έτσι η μέθοδος – αλγόριθμος Highest Speed ελέγχει μόνο την ταχύτητα κάθε κόμβου και βρίσκει τον ταχύτερο επιστρέφοντας στην βασική μέθοδο (της εφαρμογής) το αναγνωριστικό id του κόμβου αυτού. Ολοκληρώνοντας την αποστολή της, παραδίδει την σκυτάλη σε άλλες μεθόδους της εφαρμογής να

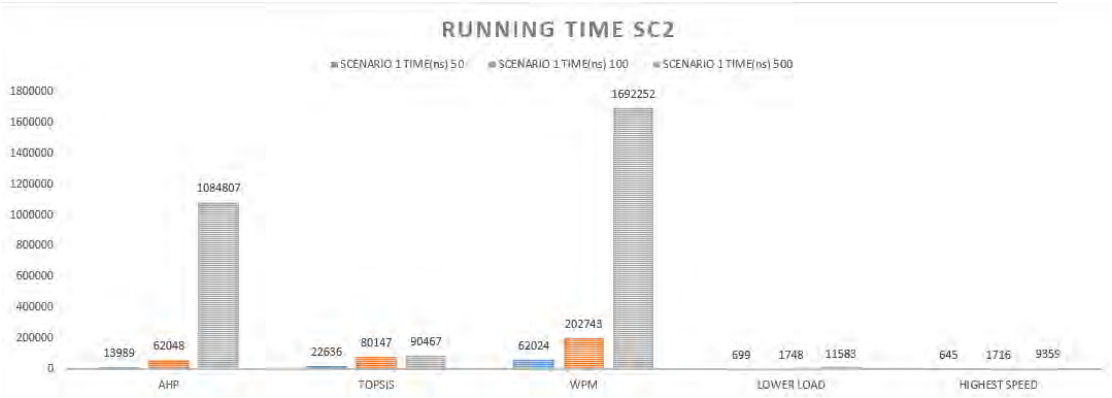
ενημερώσουν την λίστα συμπεριλαμβάνοντας και τον κόμβο που επιλέχθηκε ως ο κατάλληλος για να αποσταλεί η εργασία που ανατέθηκε στο σύστημα.

## 3.3 Μετρικές και αποτελέσματα

Στα πλαίσια της εφαρμογής που υλοποιήθηκε για την πρακτική προσομοίωση των τεχνικών – μεθόδων που αναπτύχθηκαν παραπάνω, παρατίθενται σε αυτό το σημείο τα δεδομένα που συλλέχθηκαν. Σε όλη την έκταση της εφαρμογής που υλοποιήθηκε , τοποθετήθηκαν σε κομβικά σημεία μετρητές ώστε να είμαστε σε θέση να πραγματοποιήσουμε διάφορες μετρήσεις. Οι μετρήσεις αφορούν χρονικά δεδομένα, ποσοτικά δεδομένα καθώς επίσης και ποιοτικά δεδομένα. Επίσης όλες οι μετρήσεις, τα διαγράμματα όπως και τα αποτελέσματα έχουν συλλεχθεί από μία τυχαία εκτέλεση της εφαρμογής. Το γεγονός αυτό δεν αποκλείει το ενδεχόμενο να υπάρξουν διαφορές σε μελλοντικές εκτελέσεις της εφαρμογής, έχει προαναφερθεί και παραπάνω ότι η εφαρμογή δεν κρατάει στατικά τα δεδομένα παρά μόνο το πλήθος των κόμβων. Τέλος σταθερές μεταβλητές θεωρούνται, η κατανομή των βαρών μεταξύ των διμερών σχέσεων των κριτηρίων δηλαδή των χαρακτηριστικών κάθε κόμβου που συγκρίνουν οι μέθοδοι που έχουν ενσωματωθεί στη εφαρμογή για την επιλογή του καταλληλότερου κόμβου.

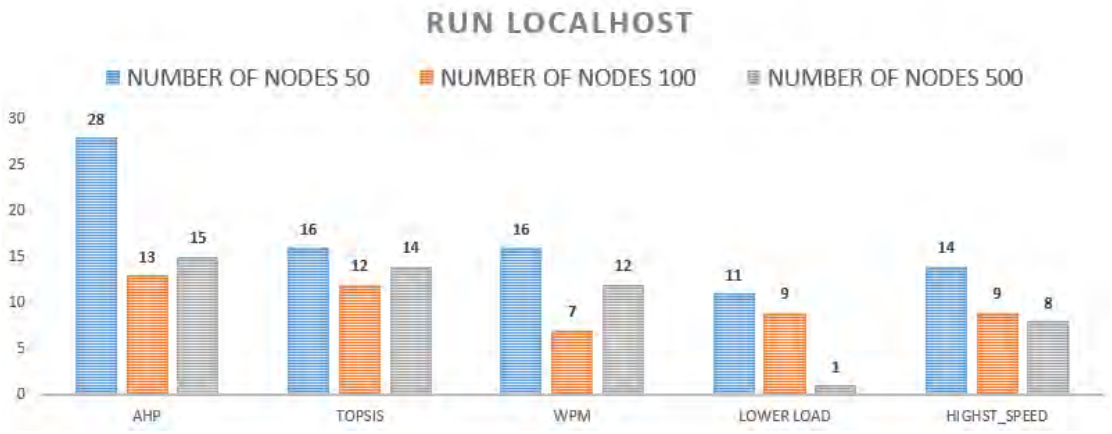



*Διάγραμμα 13 Αναπαράσταση του χρόνου εκτέλεσης των μεθόδων στο εκάστοτε πλήθος κόμβων στο πρώτο σενάριο.*

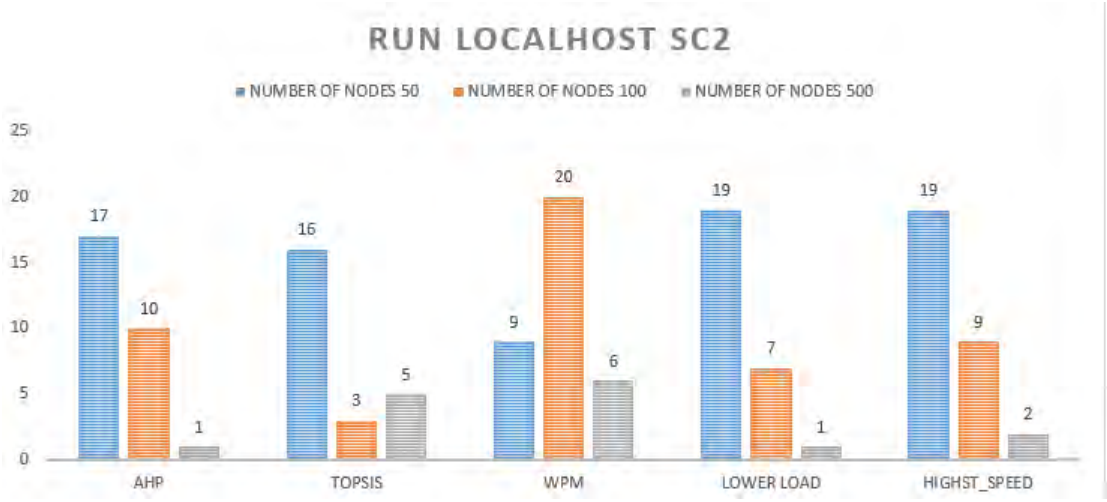



*Διάγραμμα 14 Αναπαράσταση του χρόνου εκτέλεσης των μεθόδων στο εκάστοτε πλήθος κόμβων στο δεύτερο σενάριο.*

Το διάγραμμα παραπάνω αναπαριστά το χρόνο περάτωσης της εκάστοτε μεθόδου σε συνάρτηση με το πλήθος κόμβων που αποτελείται το δίκτυο. Πιο συγκεκριμένα απεικονίζεται ο χρόνος σε nanosecond που η εκάστοτε μέθοδος εκτελέσθηκε κατά μέσο όρο. Από τη στιγμή που στέλνουμε μία εργασία στον κόμβο προς εκτέλεση, μέχρι τη στιγμή που η μέθοδος θα εξάγει σε πιο κόμβο θα σταλεί για εκτέλεση η εργασία ή αν θα μείνει να εκτελεσθεί τοπικά. Σε επίπεδο συμπερασμάτων παρατηρείται ότι η ταχύτερη μέθοδος σε επίπεδο χρόνου περάτωσης είναι η LOWER LOAD, η οποία βάση των αποτελεσμάτων που έχουμε ακολουθεί αναλογική αύξηση του χρόνου σε συνάρτηση με τους κόμβους του δικτύου, ενώ στο ίδια πλαίσια κινείται και η Highest Speed. Οι υπόλοιπες μέθοδοι έχουν σε κάθε περίπτωση ευκρινέστατη διαφορά αλλά και αύξηση του χρόνου με γνώμονα το πλήθος των κόμβων, όμως η αύξηση αυτή δεν ανταποκρίνεται σε αναλογική αλλά σε υπέρ πολλαπλάσια όταν το πλήθος των κόμβων είναι το μέγιστο. Τέλος οι διαφορές εμπίπτουν στις παραμέτρους του κάθε σεναρίου.

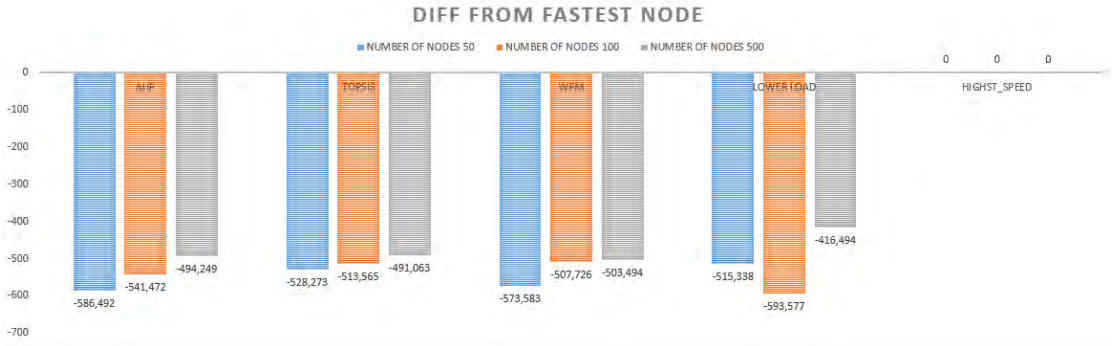



*Διάγραμμα 15 Αναπαράσταση των διεργασιών που εκτελέσθηκαν τοπικά στον κόμβο που εστάλησαν αρχικά κατά την εισαγωγή τους στο δίκτυο στο πρώτο σενάριο.*

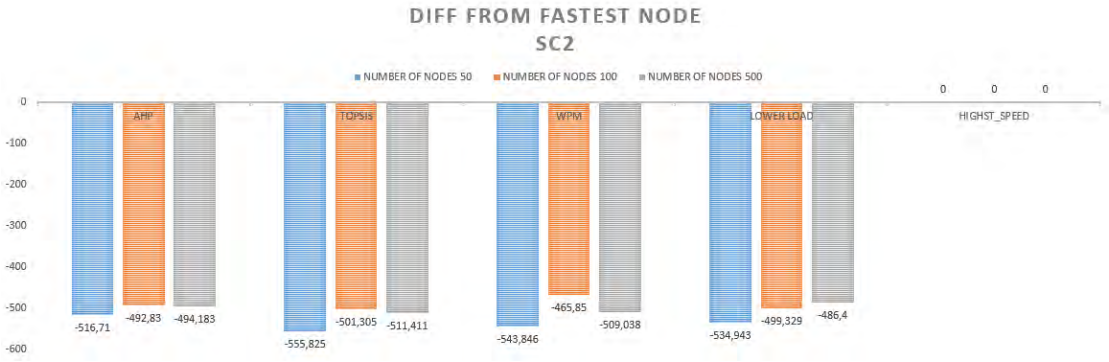



*Διάγραμμα 16 Αναπαράσταση των διεργασιών που εκτελέσθηκαν τοπικά στον κόμβο που εστάλησαν αρχικά κατά την εισαγωγή τους στο δίκτυο στο δεύτερο σενάριο.*

Όπως απεικονίζεται στο διάγραμμα 15 καθώς και στο 16 το πλήθος των διεργασιών που εκτελέσθηκαν στον κόμβο όπου εστάλησαν αρχικά, ο οποίος επιλέχθηκε τυχαία. Η αναπαράσταση όπως και στο προηγούμενο διάγραμμα αφορά τις πέντε μεθόδους επιλογής που συμπεριλαμβάνονται στην εφαρμογή σε συνάρτηση με το πλήθος των κόμβων του δικτύου. Σε αυτήν την περίπτωση τα συμπεράσματα που αντλούνται από το συγκεκριμένο διάγραμμα δεν είναι τα αναμενόμενα. Το γεγονός αυτό προκύπτει μιας και η κοινή λογική θα μας οδηγούσε στο γρήγορο αλλά λανθασμένο συμπέρασμα ότι όσο αυξάνονται οι κόμβοι τόσο θα μειώνονται και οι διεργασίες που θα εκτελούνται τοπικά. Αυτό συμβαίνει γιατί η φύση κάθε μεθόδου είναι διαφορετική και το γεγονός ότι οι μέθοδοι που συμπεριλαμβάνονται, είναι στην πλειοψηφία τους πολυκριτηριακές με αποτέλεσμα να μην εμπίπτουν στην ¨κοινή¨ λογική. Τέλος οι διαφορές εμπίπτουν στις παραμέτρους του κάθε σεναρίου.

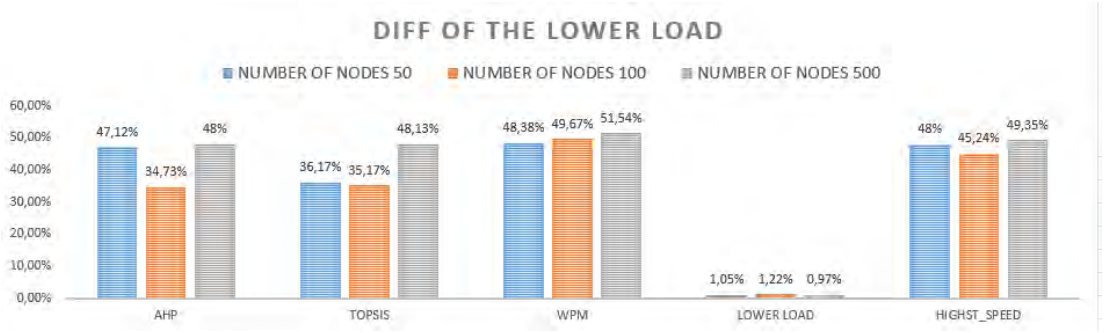



*Διάγραμμα 17 Αναπαράσταση της μέσης διαφοράς της ταχύτητας του κόμβου που επιλέχθηκε στο πρώτο σενάριο.*

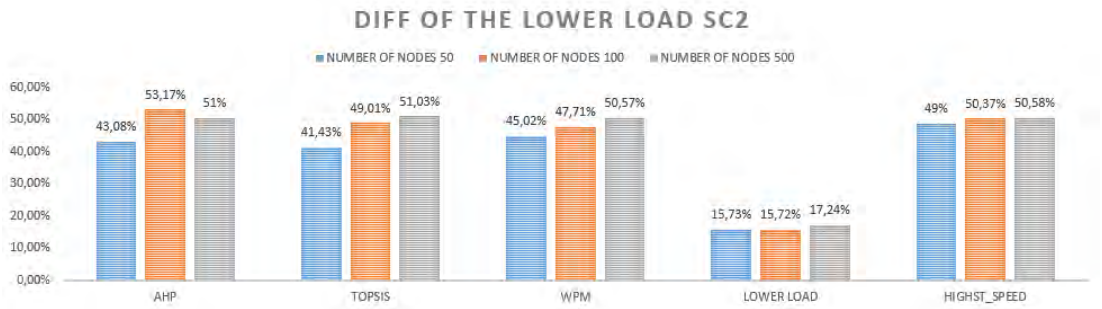



*Διάγραμμα 18 Αναπαράσταση της μέσης διαφοράς της ταχύτητας του κόμβου που επιλέχθηκε στο δεύτερο σενάριο.*

Στο σημείο αυτό γίνεται αναπαράσταση της μέσης διαφοράς της ταχύτητας του κόμβου που επιλέχθηκε από τον ταχύτερο του δικτύου. Η σύγκριση είναι σε επίπεδο μεθόδων με τους κόμβους του δικτύου. Όπως και στις προηγούμενες περιπτώσεις έτσι και εδώ δεν υπάρχει αναλογική διαφορετικά στη μείωση της διαφοράς από τον ταχύτερο στον κόμβο που επιλέχθηκε, τουλάχιστον στο δεύτερο σενάριο με εξαίρεση τη μέθοδο του χαμηλότερου φορτίου. Στο πρώτο σενάριο γίνεται το αντίστροφο με τη μέθοδο του χαμηλότερου φορτίου να έχει δυσανάλογα διαμορφώσει τα αποτελέσματα και τις τρεις πρώτες μεθόδους να μειώνονται. Τέλος και στα δύο σενάρια όπως είναι και αναμενόμενο η μέθοδος Highest Speed έχει μηδενική διαφορά από τον ταχύτερο κόμβο του δικτύου ο κόμβος που επέλεξε.

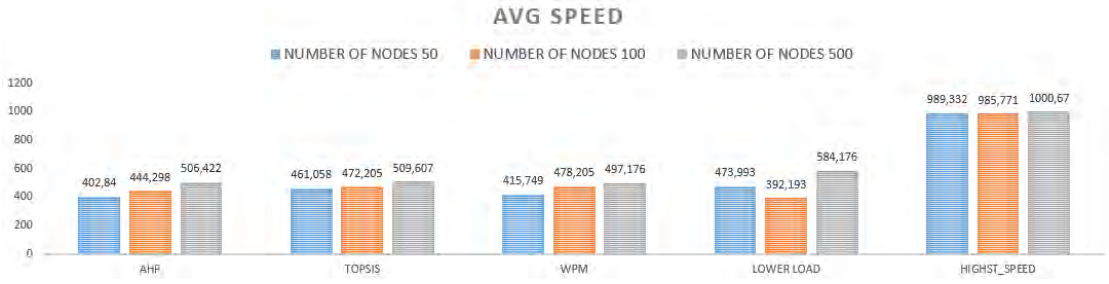



*Διάγραμμα 19 Αναπαράσταση της μέσης διαφοράς του φορτίου του κόμβου που επιλέχθηκε στο πρώτο σενάριο.*

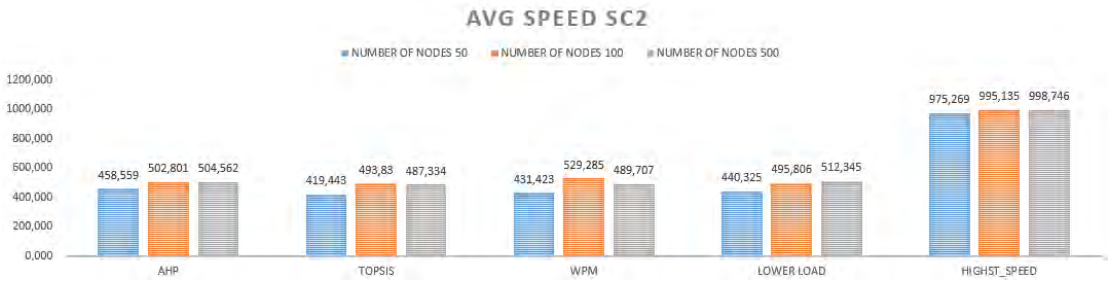



*Διάγραμμα 20 Αναπαράσταση της μέσης διαφοράς του φορτίου του κόμβου που επιλέχθηκε στο δεύτερο σενάριο.*

Σε αυτά τα διαγράμματα απεικονίζουμε την ποσοστιαία διαφορά του φορτίου του κόμβου που έχει επιλεχθεί από τις μεθόδους της εφαρμογής μας έναντι του κόμβου με το χαμηλότερο φορτίο στο δίκτυο. Και σε αυτό το σημείο η απεικόνιση είναι μεταξύ των μεθόδων σε συνάρτηση με το πλήθος κόμβων του δικτύου. Τόσο στο πρώτο όσο και στο δεύτερο σενάριο τα αποτελέσματα δεν έχουν κάποιο μοτίβο σε συνάρτηση με το πλήθος κόμβων. Το σημείο που εστιάζουμε και εκ πρώτης όψεως μας προκαλεί ερωτήματα είναι τα αποτελέσματα της μεθόδου LOWER LOAD, όπου φαινομενικά έπρεπε να είναι μηδενική η διαφορά του κόμβου που έχει επιλεχθεί από τη μέθοδο με τον κόμβο που έχει το χαμηλότερο φορτίο στο δίκτυο. Το συγκεκριμένο αποτέλεσμα προκύπτει διότι στα δύο σενάρια ορίζεται επαναπροσδιορισμός του φορτίου όλων των κόμβων του δικτύου με διαφορετικές ποσοτικές παραμέτρους σε καθένα από αυτά όπως έχουμε αναφέρει και νωρίτερα. Επομένως είναι δικαιολογημένο το αποτέλεσμα που εξάγεται βάση των σεναρίων.

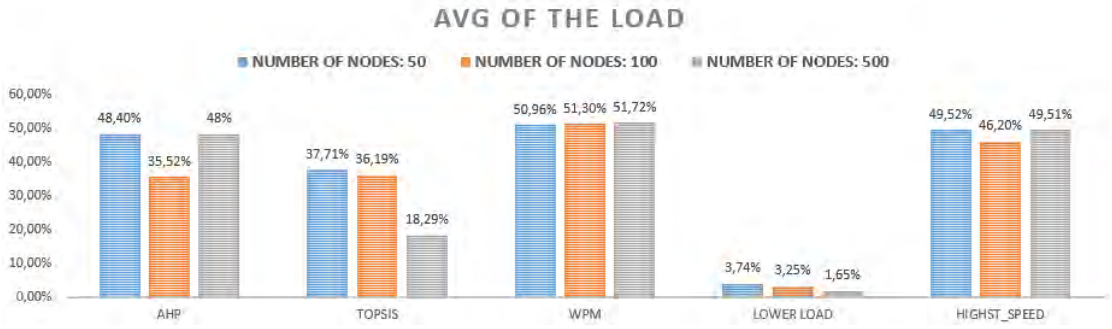



*Διάγραμμα 21 Αναπαράσταση της μέσης ταχύτητας του κόμβου που επιλέχθηκε στο πρώτο σενάριο.*

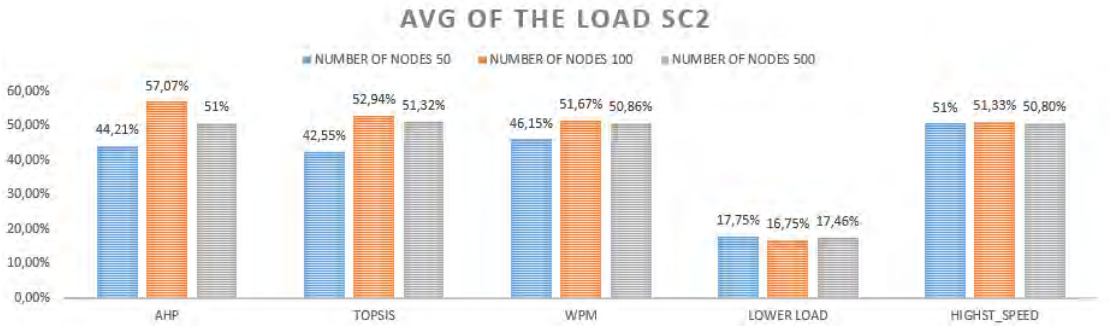



*Διάγραμμα 22 Αναπαράσταση της μέσης ταχύτητας του κόμβου που επιλέχθηκε στο δεύτερο σενάριο.*

Τα διαγράμματα που αναπαριστούν τη μέση ταχύτητα των κόμβων που επιλέχθηκαν από τις μεθόδους της εφαρμογής μας, αποτελούν μία αρκετά κατανοητή μορφή. Οι ταχύτητες που έχουν μεταξύ των δύο σεναρίων όλες οι μέθοδοι είναι πανομοιότυπες και όπως είναι αναμενόμενο και στα δύο σενάρια η μέθοδος της μέγιστης ταχύτητας έχει καταγράψει και τις υψηλότερες ταχύτητες. Οι υπόλοιπες τόσο μεταξύ του πλήθους κόμβων όσο και συγκριτικά με άλλες μεθόδους έχουν τιμές οριακά διαφορετικές.




*Διάγραμμα 23 Αναπαράσταση του μέσου φορτίου του κόμβου που επιλέχθηκε στο πρώτο σενάριο.*



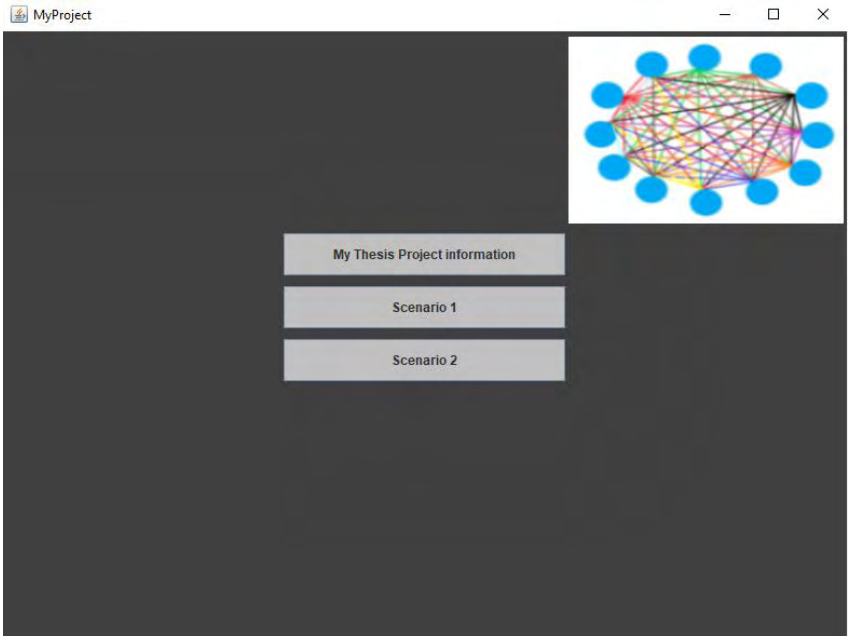

*Διάγραμμα 24 Αναπαράσταση του μέσου φορτίου του κόμβου που επιλέχθηκε στο δεύτερο σενάριο.*

Τα παραπάνω διαγράμματα απεικονίζουν το μέσο φορτίο του κόμβου που επιλέχθηκε από τις μεθόδους της εφαρμογής μας. Στα συγκεκριμένα διαγράμματα παρατηρούνται κάποια ενδιαφέροντα αποτελέσματα. Αρχικά η AHP και στα δύο σενάρια έχει απροσδιόριστο μοτίβο σε ότι αφορά τις τιμές που πήρε, το ίδιο και η μέθοδος Topsis. Οι υπόλοιπες μέθοδοι έχουν διατηρήσει πιο σταθερή καθώς και πιο μικρή την τιμή διακύμανσης των τιμών, τόσο σε επίπεδο σεναρίων όσο και σε επίπεδο πλήθους κόμβων στο δίκτυο. Όπως ήταν αναμενόμενο η μέθοδος του ελάχιστου φορτίου κατέγραψε το χαμηλότερο μέσο όρο μεταξύ των μεθόδων, χαρακτηριστικό είναι το γεγονός ότι λόγω της μορφής των σεναρίων στο πρώτο σενάριο καταγραφήκαν οι γενικά χαμηλότερες τιμές σε ότι αφορά το φορτίο.

## 3.4 Πλοήγηση

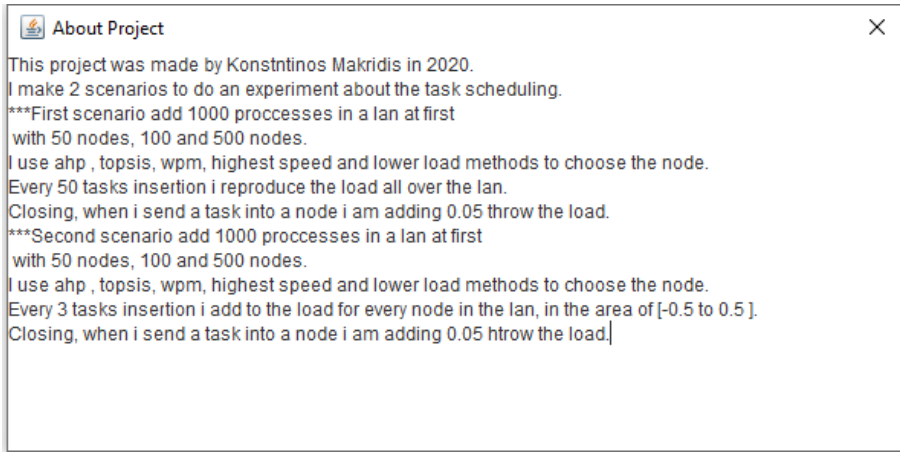
Σε αυτό το σημείο θα αναπτύξουμε και θα παρουσιάσουμε την εφαρμογή από την σκοπιά που παρουσιάζεται στο χρήστη. Βασιζόμενοι σε ένα μοτίβο μινιμαλιστικής αλλά και κατανοητής σχεδίασης καθώς επίσης και εστιάζοντας κατά κύριο λόγο στην υλοποίηση και όχι τόσο στην εμφάνιση.

Με την εκκίνηση της εφαρμογής, ο χρήστης οδηγείται σε ένα μενού έχοντας τρείς επιλογές. Οι δύο επιλογές αφορούν τα δύο σενάρια που έχουμε υλοποιήσει και προσομοιώνουμε και το τρίτο κουμπί μεταφέρει το χρήστη σε μία λειτουργία πληροφόρησης που θα αναδείξουμε παρακάτω.



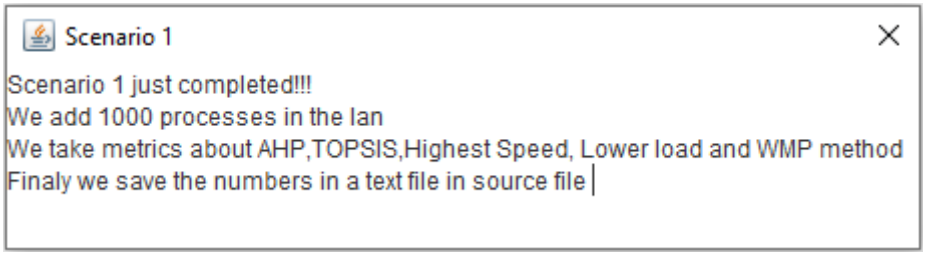

*Εικόνα 25 Στιγμιότυπο από το κεντρικό μενού της εφαρμογής που να αναπτύχθηκε στα πλαίσια της μελέτης.*

Αρχικά η εικόνα που λαμβάνει χώρα και έχει ενσωματωθεί στην εφαρμογή είναι προϊόν προσωπικής σχεδίασης και υλοποίησης και δεν έχει ληφθεί από κάποια πηγή διαδικτυακή και μη. Σε επίπεδο χρήστη το πλήκτρο «My thesis project information» εμφανίζει στο χρήστη το παρακάτω αναδυόμενο παράθυρο. Στο παράθυρο διαλόγου που γίνεται η μετάβαση αναγράφονται πληροφορίες που αφορούν την εφαρμογή. Περιλαμβάνεται μια σύντομη περιγραφή των σεναρίων και της δομής τους, έχοντας σκοπό την ένταξη του χρήστη στον τρόπο λειτουργίας της εφαρμογής καθώς επίσης και να προϊδεάσει για τα αποτελέσματα που θα ακολουθήσουν.

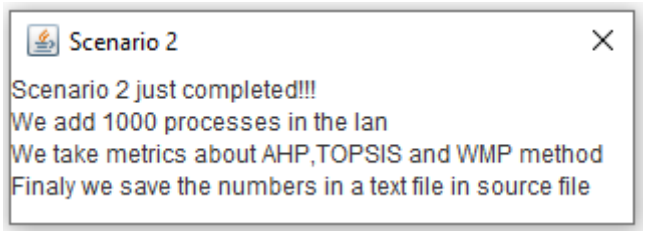



*Εικόνα 26 Στιγμιότυπο από το αναδυόμενο παράθυρο διαλόγου όταν ο χρήστης επιλέγει το πλήκτρο για τις πληροφορίες της εφαρμογής.*

Ο χρήστης έχει στη διάθεση του άλλες δύο επιλογές να τρέξει το πρώτο σενάριο της εφαρμογής ή το δεύτερο. Φυσικά και η επιλογή της σειράς εκτέλεσης των σεναρίων δεν επηρεάζει καθόλου τα αποτελέσματα και τις μετρήσεις που πραγματοποιεί και εξάγει η εφαρμογή. Ανεξάρτητα με την επιλογή του χρήστη όσο ‘τρέχει‘ η εφαρμογή δεν εμφανίζεται κάποιο μήνυμα ή κάποιο παράθυρο διαλόγου που να τον ενημερώνει για την πρόοδο του σεναρίου. Το αναδυόμενο παράθυρο διαλόγου όπως απεικονίζεται στις εικόνες 27 και 28 κατά αντιστοιχία στα δύο σενάρια της εφαρμογής.




*Εικόνα 27 Αναδυόμενο παράθυρο διαλόγου που εμφανίζεται στον χρήστη εφόσον ολοκληρωθεί η εκτέλεση του πρώτου σεναρίου.*




*Εικόνα 28 Αναδυόμενο παράθυρο διαλόγου που εμφανίζεται στον χρήστη εφόσον ολοκληρωθεί η εκτέλεση του δεύτερου σεναρίου.*

Κλείνοντας πρέπει να σημειωθεί ότι με την εκτέλεση κάθε σεναρίου στον φάκελο που εδρεύει ο πηγαίος κώδικας της εφαρμογής, δημιουργείται ένα αρχείο κειμένου .txt και περιλαμβάνει το αποτέλεσμα των μετρήσεων που έλαβαν χώρα κατά την εκτέλεση.

```
Scenario        1

nodes:  50

AHP average time to send a process to a node(nanoSec):  33650
average load from nodes which choose with AHP (%):      41.2091588379489
average speed from nodes which choose with AHP :        510.45462749673806
average load of diference from smaller load to node which choose with AHP (%):  39.508904970040746
average speed from fastest node to the node we choose with AHP :        -477.38941320039635
processes which run local(out of 1000): 21

Topsis average time to send a process to a node(nanoSec):       49523
average load from nodes which choose with Topsis (%):   44.95263266221077
average speed from nodes which choose with Topsis :     517.941075732041
average load of diference from smaller load to node which choose with Topsis (%):       43.45015648410683
average speed from fastest node to node we choose with Topsis : -469.9029649650931
processes which run local(out of 1000): 12

WPM average time to send a process to a node(nanoSec):  79495
average load from nodes which choose with WPM (%):      50.08403016369253
average speed from nodes which choose with WPM :        545.279137713918
average load of diference from smaller load to node which with from WPM (%):    47.66426583664611
average speed from fastest node to node we choose with WPM :    -442.56490298321626
processes which run local(out of 1000): 31

Lower load average time to send a process to a node(nanoSec):   1122
average load from nodes which choose with Lower load (%):       4.312021098360908
average speed from nodes which choose with Lower load : 486.0835820572162
average load of diference from smaller load to node which choose with Lower load (%):   1.0898570799824163
average speed from fastest node to node we choose with Lower load :     -501.76045863991175
processes which run local(out of 1000): 26

Highest speed average time to send a process to a node(nanoSec):        2795
average load from nodes which choose with Highest speed (%):    49.14252637806861
average speed from nodes which choose with Highest speed :      987.8440406971163
average load of diference from smaller load to node which choose with Highest speed (%):        47.97978449445302
average speed from fastest node to node we choose with Highest speed :  0.0
processes which run local(out of 1000): 14
```

*Εικόνα 29 Ενδεικτικά περιεχόμενα του αρχείου κειμένου που δημιουργεί η εφαρμογή κατά την ολοκλήρωση του εκάστοτε σεναρίου.*

Τα αποτελέσματα και οι μετρήσεις τοποθετούνται και χωρίζονται ανάλογα το πλήθος των κόμβων του δικτύου, τη μέθοδο και φυσικά ανά κατηγορία μετρήσεων. Σε επόμενη ενότητα θα σχολιάσουμε και θα αναπτύξουμε συμπεράσματα, παρατηρήσεις καθώς επίσης και ανάπτυξη αλλά και βελτιστοποίηση της εφαρμογής με γνώμονα τις παραπάνω μετρήσεις.

# 4.Συμπεράσματα

Ένα συνεργατικό σχήμα διαχείρισης εργασιών στο διαδίκτυο των πραγμάτων, αναδεικνύεται επιτυγχάνοντας τη μελέτη και τη μέτρηση παραμέτρων. Το διαδίκτυο των πραγμάτων αποτελεί μια ολοένα αναπτυσσόμενη τεχνολογία που σκοπεύει στην όσο το δυνατόν μεγαλύτερή εξάπλωση και ευρεία χρήση. Έχει αναπτυχθεί και αποτελεί αναπόσπαστο κομμάτι πολλών κλάδων όπως η βιομηχανική χρήση, η οικιακή χρήση ακόμα και έξυπνες πόλεις συμμετέχουν επωφελούμενες από αυτά που προσφέρει το διαδίκτυο των πραμάτων. Μελετήσαμε τον τρόπο λειτουργίας καθώς και τη φιλοσοφία που κρύβει το διαδίκτυο των πραγμάτων. Οι συσκευές που το απαρτίζουν συλλέγουν δεδομένα από πληθώρα αισθητήρων που είναι προσαρμοσμένοι πάντα στις ανάγκες κάτω από τις οποίες γεννήθηκε η ένταξη στο διαδίκτυο των πραγμάτων. Το πλήθος όλων των συσκευών που συσσωρεύονται στο διαδίκτυο των πραγμάτων δημιουργεί την ανάγκη για περαιτέρω μελέτη σε ότι αφορά τη διαχείριση των συσκευών αυτών καθώς και τον τρόπο με τον οποίο διαμοιράζονται οι εργασίες για να επιτευχθεί μια γρήγορη, αξιόπιστη και πλήρως διαχειρίσημη απόκριση από την πλευρά του συστήματος - δικτύου. Αναπτύξαμε την έννοια της εργασίας - task η οποία περιλαμβάνει αρκετά χαρακτηριστικά προκειμένου να μπορεί να τύχει διαχείρισης σωστά και αποτελεσματικά από την εκάστοτε εφαρμογή. Το task με τις καταστάσεις που μεταβαίνει στον κύκλο της ζωής του εξασφαλίζει μια εύκολη και πλήρως κατανοητή προς την εφαρμογή , διοχετεύοντας πληροφορίες τόσο για την ολοκλήρωση του καθώς και για το στάδιο στο οποίο βρίσκεται ακόμα και αν δεν έχει ολοκληρωθεί. Παρατηρήσαμε ότι η διαχείριση εργασιών αντλεί μεθόδους από τον άνθρωπο έχοντας πολλά κοινά σημεία με την διαχείριση έργου. Με τον τρόπο αυτό οι εφαρμογές που διαχειρίζονται το δίκτυο είναι σε θέση να διαχειριστούν και τα task τα οποία προκύπτουν. Η διαχείριση των tasks συγκεκριμένα στο διαδίκτυο των πραγμάτων είναι πιο πολύπλοκη καθώς πρέπει να συμπεριλάβει περισσότερα κριτήρια για την κατανομή εργασιών. Δεν είναι όλες οι συσκευές που εντάσσονται στο δίκτυο ικανές να εκτελέσουν πάσης φύσεων έργα που ανατίθενται στο δίκτυο. Για το λόγο αυτό περιλαμβάνονται μηχανισμοί που εκτελούν ενέργειες όπως ο διαχωρισμός των συσκευών και εντάσσονται τεχνολογίες όπως το cloud computing παρέχοντας υπηρεσίες και υπολογιστικός πόρους. Έτσι συμπεραίνουμε ότι η διαχείριση εργασιών στο διαδίκτυο των πραγμάτων έχει πολυκριτηριακό χαρακτήρα και η λήψη αποφάσεων για τη διαχείριση έχει αυξημένη

πολυπλοκότητα. Από τη μελέτη αυτή δεν θα μπορούσε να λείπει και μία προγραμματιστική εφαρμογή που να προσομοιώνει μια αντίστοιχη διαχείριση εργασιών όπως στο διαδύκτυο των πραγμάτων. Εντάχθηκαν αλγόριθμοι λήψης αποφάσεων, σε ένα δίκτυο αποτελούμενο από μεταβλητό πλήθος κόμβων. Πραγματοποιήθηκαν μετρήσεις στους πολυκριτηριακούς αλγορίθμους που εντάχθηκαν καθώς και απλούστερης μορφής αλγορίθμους λήψης αποφάσεων. Σε κάθε περίπτωση η διαχείριση των εργασιών επιτυγχάνεται θέτοντας τις παραμέτρους με τις οποίες επιθυμούμε να ληφθούν οι αποφάσεις επηρεάζοντας με αυτό τον τρόπο το τελικό αποτέλεσμα. Είναι λοιπόν σκόπιμο να υπάρχει σωστή ανάλυση πριν τη σχεδίαση και υλοποίηση ενός μηχανισμού διαχείρισης εργασιών, προκειμένου να είμαστε σε θέση να θέσουμε ιεραρχία, σε κάθε κριτήριο που περιλαμβάνεται στο πρόβλημα της διαχείρισης εργασιών στο διαδίκτυο των πραγμάτων.

# Βιβλιογραφικές Αναφορές


1. ITU Telecommunication Standardization Sector, "ITU-T Recommendation database," 2012. [Online]. http://handle.itu.int/11.1002/1000/11559-en?locatt=format:pdf&auth.
2. K. Ashton, “Kevin Ashton. The Internet of Things. Seoul, June 19, 2014 - YouTube,” 19 June 2014. [Online].https://www.youtube.com/watch?v=xSYkp8_Dn2E
3. Postscapes, "Internet of Things Definition- Postscapes," Postscapes, [Online]: https://www.postscapes.com/iot/#definition
4. Cisco Systems, "Internet of Things (IoT) - Cisco Systems," [Online]. http://www.cisco.com/web/solutions/trends/iot/overview.html.
5. IBM, "The Internet of Things - YouTube," IBM Social Media, 15 March 2010. [Online].https://www.youtube.com/watch?v=sfEbMV295Kk.
6. J. Barrett, "The Internet of Things Dr. John Barrett at TEDxCIT - YouTube," TEDx Talks, 5 October 2012. [Online]. https://www.youtube.com/watch?v=QaTIt1C5R-M.
7. “CASAGRAS an EU Framework 7 Project (Coordination and Support Action for 64 | References Global RFID-related Activities and Standardisation),” [Online]. http://grifsproject.uniweb.be/data/File/CASAGRAS%20FinalReport%20(2).pdf.
8. https://www.postscapes.com/iot-history/
9. https://hqsoftwarelab.com/blog/the-history-of-iot-a-comprehensive-timeline-of-major-events-infographic/
10. Weiser, Mark (1991). "The Computer for the 21st Century".
11. Raji, R.S. (1994). "Smart networks for control". *IEEE Spectrum*. doi:10.1109/6.284793

12. https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/
13. Kang, Won Min; Moon, Seo Yeon; Park, Jong Hyuk (5 March 2017). "An enhanced security framework for home appliances in smart home". *Human-centric Computing and Information Sciences*. **7** (6). doi:10.1186/s13673-017-0087-4
14. "How IoT & smart home automation will change the way we live". *Business Insider*. Retrieved 10 November 2017.
15. Greengard, Samuel (2015). The Internet of Things. Cambridge, MA: MIT Press. p. 90.
16. Inc., Apple. "HomeKit – Apple Developer". *developer.apple.com*. Retrieved 19 September 2018.
17. Wollerton, Megan (3 June 2018). "Here's everything you need to know about Apple HomeKit". *CNET*. Retrieved 19 September 2018.
18. Lovejoy, Ben (31 August 2018). "HomeKit devices getting more affordable as Lenovo announces Smart Home Essentials line"
19. Prospero, Mike (12 September 2018). "Best Smart Home Hubs of 2018"
20. da Costa, CA; Pasluosta, CF; Eskofier, B; da Silva, DB; da Rosa Righi, R (July 2018). "Internet of Health Things: Toward intelligent vital signs monitoring in hospital wards". *Artificial Intelligence in Medicine*. **89**: 61–69. doi:10.1016/j.artmed.2018.05.005. PMID 29871778.
21. Engineer, A; Sternberg, EM; Najafi, B (21 August 2018). "Designing Interiors to Mitigate Physical and Cognitive Deficits Related to Aging and to Promote Longevity in Older Adults: A Review"
22. Kricka, LJ (2019). "History of disruptions in laboratory medicine: what have we learned from predictions?"
23. Gatouillat, Arthur; Badr, Youakim; Massot, Bertrand; Sejdic, Ervin (2018). "Internet of Medical Things: A Review of Recent Contributions Dealing with Cyber-Physical Systems in Medicine" (PDF). *IEEE Internet of Things Journal*. **5** (5): 3810–3822. doi:10.1109/jiot.2018.2849014
24. Topol, Eric (2016). *The Patient Will See You Now: The Future of Medicine Is in Your Hands*. Basic Books.
25. Dey, Nilanjan; Hassanien, Aboul Ella; Bhatt, Chintan; Ashour, Amira S.; Satapathy, Suresh Chandra (2018). *Internet of things and big data analytics toward next-generation intelligence*
26. "Deloitte Centre for Health Solutions" (PDF). *Deloitte*.
27. Ersue, M.; Romascanu, D.; Schoenwaelder, J.; Sehgal, A. (4 July 2014). "Management of Networks with Constrained Devices: Use Cases". IETF Internet Draft.
28. Istepanian, R.; Hu, S.; Philip, N.; Sungoor, A. (2011). "The potential of Internet of m-health Things "m-IoT" for non-invasive glucose level sensing". *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. **2011**. pp. 5264–6
29. Swan, Melanie (8 November 2012). "Sensor Mania! The Internet of Things, Wearable Computing, Objective Metrics, and the Quantified Self 2.0". *Journal of Sensor and Actuator Networks*.

30. *Taiwan Information Strategy, Internet and E-commerce Development Handbook - Strategic Information, Regulations, Contacts*. IBP, Inc. USA. 2016. p. 79.
31. Yang, Chen; Shen, Weiming; Wang, Xianbin (January 2018). "The Internet of Things in Manufacturing: Key Issues and Potential Applications". *IEEE Systems, Man, and Cybernetics Magazine*.
32. an, Lu; Wang, Neng (20–22 August 2010). Future Internet: The Internet of Things. 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE). 5. pp. 376–380.
33. Daugherty, Paul; Negm, Walid; Banerjee, Prith; Alter, Allan. "Driving Unconventional Growth through the Industrial Internet of Things"
34. Gubbi, Jayavardhana; Buyya, Rajkumar; Marusic, Slaven; Palaniswami, Marimuthu (24 February 2013). "Internet of Things (IoT): A vision, architectural elements, and future directions". *Future Generation Computer Systems*.
35. Chui, Michael; Löffler, Markus; Roberts, Roger. "The Internet of Things". *McKinsey Quarterly*. McKinsey & Company. Retrieved 10 July 2014.
36. Davies, Nicola. "How the Internet of Things will enable 'smart buildings'". *Extreme Tech*.
37. "Molluscan eye"
38. Li, Shixing; Wang, Hong; Xu, Tao; Zhou, Guiping (2011). *Application Study on Internet of Things in Environment Protection Field*. *Lecture Notes in Electrical Engineering Volume* (Submitted manuscript). Lecture Notes in Electrical Engineering. **133**. pp. 99–106.
39. "Use case: Sensitive wildlife monitoring". *FIT French Project*.
40. Hart, Jane K.; Martinez, Kirk (1 May 2015). "Toward an environmental Internet of Things". *Earth & Space Science*.
41. Marimuthu, G., & Ramesh, G. (2016). Comparison among Original AHP, Ideal AHP and Moderate AHP Models. *International Research Journal of Engineering, IT & Scientific Research, 2*(5), 40. doi:10.21744/irjeis.v2i5.26
42. Simon, Joel & Ali, Adamu & Abdulkadir, Ahmed & Henry, Akpensuen. (2019). Analytical Hierarchy Process (AHP) Model for Prioritizing Alternative Strategies for Malaria Control. Asian Journal of Probability and Statistics. 1-8. 10.9734/ajpas/2019/v5i130124.
43. Pavić, Zlatko & Novoselac, Vedran. (2013). Notes on TOPSIS Method. International Journal of Research in Engineering and Science. 1. 5-12.
44. Yakup Çelikbilek & Fatih Tüysüz (2020) An in-depth review of theory of the TOPSIS method: An experimental analysis, Journal of Management Analytics, 7:2, 281-300, DOI: 10.1080/23270012.2020.1748528
45. Divya C., Raju L.S., Singaravel B. (2020) A Review of TOPSIS Method for Multi Criteria Optimization in Manufacturing Environment. In: Dawn S., Balas V., Esposito A., Gope S. (eds) Intelligent Techniques and Applications in Science and Technology. ICIMSAT 2019. Learning and Analytics in Intelligent Systems, vol 12. Springer, Cham. https://doi.org/10.1007/978-3-030-42363-6_84

46. AIP Conference Proceedings 1977, 020049 (2018); https://doi.org/10.1063/1.5042905 Published Online: 26 June 2018
47. A. DAGHOURI, K. MANSOURI and M. QBADOU, "Multi Criteria Decision Making methods for Information System Selection: A Comparative Study," *2018 International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS)*, Kenitra, 2018, pp. 1-5, doi: 10.1109/ICECOCS.2018.8610579.
48. Sinha, A., Kumar, P., Rana, N.P. *et al.* Impact of internet of things (IoT) in disaster management: a task-technology fit perspective. *Ann Oper Res* **283,** 759–794 (2019). https://doi.org/10.1007/s10479-017-2658-1
49. Podriguez-Zurrunero, R.; Utrilla, R.; Rozas, A.; Araujo, A. Process Management in IoT Operating Systems: Cross-Influence between Processing and Communication Tasks in End-Devices. *Sensors* **2019**, *19*, 805.
50. Cleland, D. and W. King. “Project management handbook.” (1983).
51. Matt Bower. 2008. A taxonomy of task types in computing. SIGCSE Bull. 40, 3 (September 2008), 281–285. DOI: https://doi.org/10.1145/1597849.1384346

# ΠΑΡΑΡΤΗΜΑ

## MAIN

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package project.test;

import java.util.*;
import java.lang.Math;


/**
 *
 * @author Κωστής
 */
public class ProjectTest {


    public static void main(String[] args){

       Graphics g = new Graphics();

    }


}
```

## NODE.java

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package project.test;

import java.util.ArrayList;
import java.util.Arrays;

/**
 *
 * @author Κωστής
 */
public class Node {
    double[][] tmparray_nodes,tmp;
    double[] tmp2,tmpload,tmpspeed;
    ArrayList<double [][]> lan = new ArrayList();
    ArrayList<double [][]> lan2 = new ArrayList();
    double[][] copy,copy2,copy3;
    int value;



    Node(int nfn){
        this.tmparray_nodes = new double [nfn][4];
    this.copy = new double[nfn][3];
    this.value = nfn;
  }

  public void setlan(int nodes){
    int j;
    for(j=0;j<nodes;j++){
        /*initialize the current node informations */
      tmparray_nodes[j][0]=j;
      tmparray_nodes[j][1] =(Math.random() * ((100.0 - 0.0) + 1)) +
0.0;
      // set random load range [0.0,100.0]
      tmparray_nodes[j][2]=(Math.random() * ((1000 - 1) + 1)) + 1;
      // set random speed range [1,1000]
      tmparray_nodes[j][3] = 0;
      // set the cost to zero in the current node
    }
  }

  public void set_list(int n){
    for(int i=0;i<n;i++){
      tmp = new double [n][4];
      for(int j=0;j<n;j++){
        tmp[j][0] = j;
        tmp[j][1] = tmparray_nodes[j][1];
        tmp[j][2] = tmparray_nodes[j][2];
        if(j == i){
          tmp[j][3] = 0;
        }else{
          tmp[j][3] = (int)(Math.random() * 10000) + 1;
          //i set a random cost from every node to another
```

```
      }
    }
    lan.add(tmp);
    lan2.add(tmp);

  }
}

public double[][] get_nodelist(int x){
  return lan.get(x);
}


public void remake_listsc1(){
  for(int i=0;i<value;i++){
    copy2 = new double[value][4];
    for(int j=0;j<value;j++){
      copy2[j][0] = lan2.get(i)[j][0];
      copy2[j][1] = copy[j][1];
      copy2[j][2] = lan2.get(i)[j][2];
      copy2[j][3] = lan2.get(i)[j][3];


    }
    lan.add(copy2);
  }
}

public void remake_loadsc1(){
  lan.removeAll(lan);
  for(int i=0;i<value;i++){
    copy2 = new double[value][4];
    for(int j=0;j<value;j++){
      copy2[j][0] = lan2.get(i)[j][0];
      copy2[j][1] = (Math.random() * ((100.0 - 0.0) + 1)) + 0.0;
      // set random load range [0.0,100.0]
      copy2[j][2] = lan2.get(i)[j][2];
      copy2[j][3] = lan2.get(i)[j][3];
    }
    lan.add(copy2);
  }

}

public void plusloadsc1(int x){
  for(int i = 0;i<lan.get(x).length;i++){
    copy[i][0] = lan.get(x)[i][0];
    copy[i][1] = lan.get(x)[i][1];
    copy[i][2] = lan.get(x)[i][2];
  }
  copy[x][1] += 0.05 ;

  lan.removeAll(lan);

}


public void remake_loadsc2(){
   lan.removeAll(lan);
  for(int i=0;i<value;i++){
    copy2 = new double[value][4];
```

```
      for(int j=0;j<value;j++){
        copy2[j][0] = lan2.get(i)[j][0];
        copy2[j][1] = ((Math.random() * ((0.5 - (-0.5)) + 1)) + (-
0.5))
         +lan2.get(i)[j][1];
        //set the load as before plus random range [-0.5,0.5]
        copy2[j][2] = lan2.get(i)[j][2];
        copy2[j][3] = lan2.get(i)[j][3];
      }
      lan.add(copy2);
    }
  }

  public void printlan(){
    for(int i=0;i<lan.size();i++){
      System.out.println("Node: "+i);
      for(int j=0;j<tmp.length;j++){
        System.out.println("id: "+lan.get(i)[j][0]+" load: "
            +lan.get(i)[j][1]+" speed: "+lan.get(i)[j][2]+" cost: "
            +lan.get(i)[j][3]);
      }
    }
  }

  public void reset_list(){
    lan.removeAll(lan);
    for(int i=0;i<value;i++){
      copy2 = new double[value][4];
      for(int j=0;j<value;j++){
        copy2[j][0] = lan2.get(i)[j][0];
        copy2[j][1] = lan2.get(i)[j][1];
        copy2[j][2] = lan2.get(i)[j][2];
        copy2[j][3] = lan2.get(i)[j][3];
      }
      lan.add(copy2);
    }
  }

  public double get_load(int x){
    return lan.get(x)[x][1];
  }

  public double get_speed(int x){
    return lan.get(x)[x][2];
  }

  public double get_maxspeed(){
    tmpspeed = new double[value];
    for(int i=0;i<value;i++){
      tmpspeed[i] = lan.get(0)[i][2];
    }
    Arrays.sort(tmpspeed);
    return tmpspeed[value-1];
  }

  public double get_minload(){
    tmpload = new double[value];
    for(int i=0;i<value;i++){
      tmpload[i] = lan.get(0)[i][1];
    }
    Arrays.sort(tmpload);
```

```
    return tmpload[0];
  }

  public int get_lenght(){
    return lan.size();
  }
}
```

## Graphics.java

```
package project.test;

import java.awt.*;
import java.awt.Canvas;
import java.awt.GridBagLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Random;
import java.util.Scanner;
import javax.swing.*;
/**
 *
 * @author Κωστής
 */
public class Graphics extends JFrame implements ActionListener{

  private JFrame frame_hp;
  private JPanel
pan1_hp,pan2_hp,pan3_hp,pan4_hp,pan5_hp,pan6_hp,pan7_hp,
      pan8_hp,pan9_hp;
  private JButton b1,b2,b3,b4,b5;
  private JLabel jl;
  int flag = 0;
  Write_to_file wt = new Write_to_file();




  public Graphics(){
    GUI();
  }

  public void GUI(){
    frame_hp = new JFrame("MyProject");
    frame_hp.setVisible(true);
    frame_hp.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame_hp.setSize(800,600);
    frame_hp.setLayout(new GridLayout(3,3));
    jl = new JLabel();
    jl.setIcon(new ImageIcon("C:\\Users\\Κωστής\\Documents"
        + "\\NetBeansProjects\\Project test\\logo.png"));
    jl.setVisible(true);
```

```
    pan1_hp = new JPanel();
    pan2_hp = new JPanel();
    pan3_hp = new JPanel();
    pan4_hp = new JPanel();
    pan5_hp = new JPanel();
    pan6_hp = new JPanel();
    pan7_hp = new JPanel();
    pan8_hp = new JPanel();
    pan9_hp = new JPanel();
    pan1_hp.setBackground(Color.darkGray);
    pan2_hp.setBackground(Color.darkGray);
    pan3_hp.setBackground(Color.darkGray);
    pan4_hp.setBackground(Color.darkGray);
    pan5_hp.setBackground(Color.darkGray);
    pan6_hp.setBackground(Color.darkGray);
    pan7_hp.setBackground(Color.darkGray);
    pan8_hp.setBackground(Color.darkGray);
    pan9_hp.setBackground(Color.darkGray);
    b1 = new JButton("My Thesis Project information");
    b1.setBackground(Color.LIGHT_GRAY);
    b1.addActionListener(this);
    b2 = new JButton("Scenario 1");
    b2.setBackground(Color.LIGHT_GRAY);
    b2.addActionListener(this);
    b3 = new JButton("Scenario 2");
    b3.setBackground(Color.LIGHT_GRAY);
    b3.addActionListener(this);
    b5 = new JButton("i");
    b5.setBackground(Color.LIGHT_GRAY);


    pan3_hp.add(jl);
    pan3_hp.validate();
    pan5_hp.add(b1);
    pan5_hp.add(b2);
    pan5_hp.add(b3);
    pan5_hp.setLayout(new GridLayout(4,1,10,10));
    frame_hp.add(pan1_hp);
    frame_hp.add(pan2_hp);
    frame_hp.add(pan3_hp);
    frame_hp.add(pan4_hp);
    frame_hp.add(pan5_hp);
    frame_hp.add(pan6_hp);
    frame_hp.add(pan7_hp);
    frame_hp.add(pan8_hp);
    frame_hp.add(pan9_hp);
    frame_hp.validate();


  }

   public void actionPerformed(ActionEvent e){
     String text = "This project was made by Konstntinos Makridis in
"
         + "2020. " +"\n" +
            "I make 2 scenarios to do an experiment about the task"
         + " scheduling." +"\n" +
            "***First scenario add 1000 proccesses in a lan at first"
         +"\n" +
            " with 50 nodes, 100 and 500 nodes."+"\n" +
            "I use ahp , topsis, wpm, highest speed and lower load "
```

```
            + "methods to choose the node."+"\n" +
               "Every 50 tasks insertion i reproduce the load all over "
            + "the lan."+"\n" +
               "Closing, when i send a task into a node i am adding
0.05"
            + " throw the load."+"\n" +
               "***Second scenario add 1000 proccesses in a lan at "
            + "first"+"\n" +
               " with 50 nodes, 100 and 500 nodes."+"\n" +
               "I use ahp , topsis, wpm, highest speed and lower load "
            + "methods to choose the node."+"\n" +
               "Every 3 tasks insertion i add to the load for every "
            + "node in the lan, in the area of [-0.5 to 0.5 ]."+"\n" +
               "Closing, when i send a task into a node i am adding "
            + "0.05 htrow the load.";

        if(e.getSource() == b1){
          JDialog d = new JDialog(frame_hp, "About Project");
          JTextArea l = new JTextArea();
          l.setRows(10);
          l.setText(text);
          l.setLineWrap(true);
          l.setWrapStyleWord(true);
          d.add(l);
          d.setSize(600,300);
          d.setVisible(true);
        }
        else if(e.getSource() == b2){
          wt.open_file();;
          int i,y=0,ahplocal = 0,topsislocal = 0,wpmlocal = 0,lllocal =
0,
              hslocal = 0;
          int nodes;
          long ahpstart = 0,ahpfinish = 0,ahptotal = 0,topsisstart = 0,
              topsisfinish = 0,topsistotal = 0,wpmstart = 0,
              wpmfinish = 0,wpmtotal = 0,llstart = 0,llfinish = 0,
              lltotal = 0,
              hsstart = 0,hsfinish = 0,hstotal = 0;
          double [] priority = new double [3];
          double ahplavg = 0,topsislavg = 0,wpmlavg = 0;
          double ahpsavg = 0,topsissavg = 0,wpmsavg = 0;
          double ahpdlavg = 0,topsisdlavg = 0,wpmdlavg = 0;
          double ahpdsavg = 0,topsisdsavg = 0,wpmdsavg = 0;
          double lllavg = 0,llsavg = 0,lldlavg = 0, lldsavg = 0;
          double hslavg = 0,hssavg = 0,hsdlavg = 0, hsdsavg = 0;
          ArrayList<Double> ahpload= new ArrayList();
          ArrayList<Double> topsisload= new ArrayList();
          ArrayList<Double> wpmload= new ArrayList();
          ArrayList<Double> llload= new ArrayList();
          nodes = 50; // number of the nodes static
          AHP A = new AHP(3); //call ahp
          WPM W;//initial weighted product model
          TOPSIS T;
          Lower_load ll = new Lower_load();
          Highest_speed hs = new Highest_speed();
          Node node = new Node(nodes); // init the nodes
          /*set pairwise matrix*/
          priority[0] = 3.0;
          priority[1] = 5.0;
          priority[2] = 3.0;
```

```
A.setpairwisemtr(priority);

node.setlan(nodes); //set up the lan's specs
node.set_list(nodes);
//1000 processes insert with AHP method
wt.addtext("Scenario", "1");
wt.space();
wt.addtext("nodes: ", "50");
wt.space();
for(i=0;i<1000;i++){
  int z = (int)(Math.random() * (nodes-1)) + 1;
  ahpstart = System.nanoTime();
  A.setrank(node.get_nodelist(z));
  y = (int)A.getrank(z);
  ahpfinish = System.nanoTime();
  ahplavg += node.get_load(y);
  ahpsavg += node.get_speed(y);
  ahpdsavg += (node.get_speed(y) - node.get_maxspeed());
  ahpdlavg += (node.get_load(y) - node.get_minload());
  ahpload.add(node.get_load(y));
  node.plusloadsc1(y); //assign the process random in a node
  node.remake_listsc1();
  if(i%50==0){
    node.remake_loadsc1();
  }
  if(y == z){
    ahplocal ++;
  }
  ahptotal += ahpfinish -ahpstart;
}
wt.addtext("AHP average time to send a process to a node"
    + "(nanoSec): ", String.valueOf(ahptotal/1000));
wt.addtext("average load from nodes which choose with "
    + "AHP (%):", String.valueOf(ahplavg/1000));
wt.addtext("average speed from nodes which choose with "
    + "AHP :", String.valueOf(ahpsavg/1000));
wt.addtext("average load of diference from smaller load"
    + " to node which choose with AHP (%):",
    String.valueOf(ahpdlavg/1000));
wt.addtext("average speed from fastest node to the"
    + " node we choose with AHP :",
    String.valueOf(ahpdsavg/1000));
wt.addtext("processes which run local(out of 1000):",
    String.valueOf(ahplocal));
wt.space();
node.reset_list();
//1000 processes isert with Topsis method /*
 for(i=0;i<1000;i++){
  int z = (int)(Math.random() * (nodes-1)) + 1;
  topsisstart = System.nanoTime();
  T = new TOPSIS(node.get_nodelist(z)); // call TOPSIS
  y = (int)T.getpi(z);
  topsisfinish = System.nanoTime();
  topsislavg += node.get_load(y);
  topsissavg += node.get_speed(y);
  topsisdsavg += (node.get_speed(y) - node.get_maxspeed());
  topsisdlavg += (node.get_load(y) - node.get_minload());
  topsisload.add(node.get_load(y));
  node.plusloadsc1(y); //assign the process random in a node
  node.remake_listsc1();
  if(i%50==0){
```

```
    node.remake_loadsc1();
  }
  if(y == z){
    topsislocal ++;
  }
  topsistotal += topsisfinish - topsisstart;
}
wt.addtext("Topsis average time to send a process to a "
    + "node(nanoSec): ", String.valueOf(topsistotal/1000));
wt.addtext("average load from nodes which choose with Topsis "
    + "(%):", String.valueOf(topsislavg/1000));
wt.addtext("average speed from nodes which choose with "
    + "Topsis :", String.valueOf(topsissavg/1000));
wt.addtext("average load of diference from smaller load "
    + "to node which choose with Topsis (%):"
    , String.valueOf(topsisdlavg/1000));
wt.addtext("average speed from fastest node to node"
    + " we choose with Topsis :"
    , String.valueOf(topsisdsavg/1000));
wt.addtext("processes which run local(out "
    + "of 1000):", String.valueOf(topsislocal));
wt.space();
node.reset_list();
//1000 processe insert with WPM method
for(i=0;i<1000;i++){
  int z = (int)(Math.random() * (nodes-1)) + 1;
  wpmstart = System.nanoTime();
  W = new WPM(node.get_nodelist(z)); // call WPM
  y = (int)W.getnormal(z);
  wpmfinish = System.nanoTime();
  wpmlavg += node.get_load(y);
  wpmsavg += node.get_speed(y);
  wpmdsavg += (node.get_speed(y) - node.get_maxspeed());
  wpmdlavg += (node.get_load(y) - node.get_minload());
  wpmload.add(node.get_load(y));
  node.plusloadsc1(y); //assign the process random in a node
  node.remake_listsc1();
  if(i%50==0){
    node.remake_loadsc1();
  }
  if(y == z){
    wpmlocal ++;
  }
  wpmtotal += wpmfinish - wpmstart;
}
wt.addtext("WPM average time to send a process to a"
    + " node(nanoSec): ", String.valueOf(wpmtotal/1000));
wt.addtext("average load from nodes which choose "
    + "with WPM (%):", String.valueOf(wpmlavg/1000));
wt.addtext("average speed from nodes which choose "
    + "with WPM :", String.valueOf(wpmsavg/1000));
wt.addtext("average load of diference from smaller"
    + " load to node which with from WPM (%):"
    , String.valueOf(wpmdlavg/1000));
wt.addtext("average speed from fastest node to node "
    + "we choose with WPM :", String.valueOf(wpmdsavg/1000));
wt.addtext("processes which run local(out of 1000):"
    , String.valueOf(wpmlocal));
wt.space();
node.reset_list();
//1000 processe insert with lower load method
```

```
        for(i=0;i<1000;i++){
          int z = (int)(Math.random() * (nodes-1)) + 1;
          llstart = System.nanoTime();
          ll.find_lowestload(node.get_nodelist(z));
          y = (int)ll.getid();
          llfinish = System.nanoTime();
          lllavg += node.get_load(y);
          llsavg += node.get_speed(y);
          lldsavg += (node.get_speed(y) - node.get_maxspeed());
          lldlavg += ((node.get_load(y) - node.get_minload()));
          llload.add(node.get_load(y));
          node.plusloadsc1(y); //assign the process random in a node
          node.remake_listsc1();
          if(i%50==0){
            node.remake_loadsc1();
          }
          if(y == z){
            lllocal ++;
          }
          lltotal += llfinish - llstart;
        }
        wt.addtext("Lower load average time to send a process"
            + " to a node(nanoSec): ", String.valueOf(lltotal/1000));
        wt.addtext("average load from nodes which choose with"
            + " Lower load (%):", String.valueOf(lllavg/1000));
        wt.addtext("average speed from nodes which choose with"
            + " Lower load :", String.valueOf(llsavg/1000));
        wt.addtext("average load of diference from smaller load to
node"
            + " which choose with Lower load (%):"
            , String.valueOf(lldlavg/1000));
        wt.addtext("average speed from fastest node to node "
            + "we choose with Lower load :"
            , String.valueOf(lldsavg/1000));
        wt.addtext("processes which run local(out of 1000):"
            , String.valueOf(lllocal));
        wt.space();
        node.reset_list();
        //1000 processe insert with highest speed method
        for(i=0;i<1000;i++){
          int z = (int)(Math.random() * (nodes-1)) + 1;
          hsstart = System.nanoTime();
          hs.find_highestspeed(node.get_nodelist(z));
          y = (int)hs.getid();
          hsfinish = System.nanoTime();
          hslavg += node.get_load(y);
          hssavg += node.get_speed(y);
          hsdsavg += (node.get_speed(y) - node.get_maxspeed());
          hsdlavg += (node.get_load(y) - node.get_minload());
          ahpload.add(node.get_load(y));
          node.plusloadsc1(y); //assign the process random in a node
          node.remake_listsc1();
          if(i%50==0){
            node.remake_loadsc1();
          }
          if(y == z){
            hslocal ++;
          }
          hstotal += hsfinish - hsstart;
        }
        wt.addtext("Highest speed average time to send a process "
```

```
            + "to a node(nanoSec): ", String.valueOf(hstotal/1000));
        wt.addtext("average load from nodes which choose with Highest "
            + "speed (%):", String.valueOf(hslavg/1000));
        wt.addtext("average speed from nodes which choose with Highest"
            + " speed :", String.valueOf(hssavg/1000));
        wt.addtext("average load of diference from smaller load to
node"
            + " which choose with Highest speed (%):",
            String.valueOf(hsdlavg/1000));
        wt.addtext("average speed from fastest node to node we choose"
            + " with Highest speed :", String.valueOf(hsdsavg/1000));
        wt.addtext("processes which run local(out of 1000):"
            , String.valueOf(hslocal));
        wt.space();
        node.reset_list();
        nodes = 100;
        node = new Node(nodes);
        node.setlan(nodes); //set up the lan's specs
        node.set_list(nodes);
        //1000 processes insert with AHP method
        ahplavg = 0;topsislavg = 0;
        wpmlavg = 0;ahpsavg = 0;
        topsissavg = 0;wpmsavg = 0; ahpdlavg = 0;
        topsisdlavg = 0;wpmdlavg = 0;
        ahpdsavg = 0;topsisdsavg = 0;wpmdsavg = 0;
        lllavg = 0;llsavg = 0;lldlavg = 0; lldsavg = 0;
        hslavg = 0;hssavg = 0;hsdlavg = 0; hsdsavg = 0;
        ahplocal = 0;topsislocal = 0;wpmlocal = 0;
        lllocal = 0; hslocal = 0;
        wt.addtext("nodes:", "100");
        wt.space();
        for(i=0;i<1000;i++){
          int z = (int)(Math.random() * (nodes-1)) + 1;
          ahpstart = System.nanoTime();
          A.setrank(node.get_nodelist(z));
          y = (int)A.getrank(z);
          ahpfinish = System.nanoTime();
          ahplavg += node.get_load(y);
          ahpsavg += node.get_speed(y);
          ahpdsavg += (node.get_speed(y) - node.get_maxspeed());
          ahpdlavg += (node.get_load(y) - node.get_minload());
          ahpload.add(node.get_load(y));
          node.plusloadsc1(y); //assign the process random in a node
          node.remake_listsc1();
          if(i%50==0){
            node.remake_loadsc1();
          }
          if(y == z){
            ahplocal ++;
          }
          ahptotal += ahpfinish -ahpstart;
        }
        wt.addtext("AHP average time to send a process to a
node(nanoSec): ", String.valueOf(ahptotal/1000));
        wt.addtext("average load from nodes which choose with AHP
(%):", String.valueOf(ahplavg/1000));
        wt.addtext("average speed from nodes which choose with AHP :",
String.valueOf(ahpsavg/1000));
        wt.addtext("average load of diference from smaller load to node
which choose with AHP (%):", String.valueOf(ahpdlavg/1000));
```

```
      wt.addtext("average speed from fastest node to node we choose
with AHP :", String.valueOf(ahpdsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(ahplocal));
      wt.space();
      node.reset_list();
      //1000 processes isert with Topsis method /*
       for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        topsisstart = System.nanoTime();
        T = new TOPSIS(node.get_nodelist(z)); // call TOPSIS
        y = (int)T.getpi(z);
        topsisfinish = System.nanoTime();
        topsislavg += node.get_load(y);
        topsissavg += node.get_speed(y);
        topsisdsavg += (node.get_speed(y) - node.get_maxspeed());
        topsisdlavg += (node.get_load(y) - node.get_minload());
        topsisload.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%50==0){
          node.remake_loadsc1();
        }
        if(y == z){
          topsislocal ++;
        }
        topsistotal += topsisfinish - topsisstart;
      }
      wt.addtext("Topsis average time to send a process to a
node(nanoSec): ", String.valueOf(topsistotal/1000));
      wt.addtext("average load from nodes which choose with Topsis
(%):", String.valueOf(topsislavg/1000));
      wt.addtext("average speed from nodes which choose with Topsis
:", String.valueOf(topsissavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with Topsis (%):", String.valueOf(topsisdlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
with Topsis :", String.valueOf(topsisdsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(topsislocal));
      wt.space();
      node.reset_list();
      //1000 processe insert with WPM method
      for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        wpmstart = System.nanoTime();
        W = new WPM(node.get_nodelist(z)); // call WPM
        y = (int)W.getnormal(z);
        wpmfinish = System.nanoTime();
        wpmlavg += node.get_load(y);
        wpmsavg += node.get_speed(y);
        wpmdsavg += (node.get_speed(y) - node.get_maxspeed());
        wpmdlavg += (node.get_load(y) - node.get_minload());
        wpmload.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%50==0){
          node.remake_loadsc1();
        }
        if(y == z){
          wpmlocal ++;
```

```
        }
        wpmtotal += wpmfinish - wpmstart;
      }
      wt.addtext("WPM average time to send a process to a
node(nanoSec): ", String.valueOf(wpmtotal/1000));
      wt.addtext("average load from nodes which choose with WPM
(%):", String.valueOf(wpmlavg/1000));
      wt.addtext("average speed from nodes which choose with WPM :",
String.valueOf(wpmsavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with WPM (%):", String.valueOf(wpmdlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
with WPM :", String.valueOf(wpmdsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(wpmlocal));
      wt.space();
      node.reset_list();
      //1000 processe insert with lower load method
      for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        llstart = System.nanoTime();
        ll.find_lowestload(node.get_nodelist(z));
        y = (int)ll.getid();
        llfinish = System.nanoTime();
        lllavg += node.get_load(y);
        llsavg += node.get_speed(y);
        lldsavg += (node.get_speed(y) - node.get_maxspeed());
        lldlavg += ((node.get_load(y) - node.get_minload()));
        llload.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%50==0){
          node.remake_loadsc1();
        }
        if(y == z){
          lllocal ++;
        }
        lltotal += llfinish - llstart;
      }
      wt.addtext("Lower load average time to send a process to a
node(nanoSec): ", String.valueOf(lltotal/1000));
      wt.addtext("average load from nodes which choose with Lower
load (%):", String.valueOf(lllavg/1000));
      wt.addtext("average speed from nodes which choose with Lower
load :", String.valueOf(llsavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with Lower load (%):", String.valueOf(lldlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
with Lower load :", String.valueOf(lldsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(lllocal));
      wt.space();
      node.reset_list();
      //1000 processe insert with highest speed method
      for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        hsstart = System.nanoTime();
        hs.find_highestspeed(node.get_nodelist(z));
        y = (int)hs.getid();
        hsfinish = System.nanoTime();
        hslavg += node.get_load(y);
```

```
        hssavg += node.get_speed(y);
        hsdsavg += (node.get_speed(y) - node.get_maxspeed());
        hsdlavg += (node.get_load(y) - node.get_minload());
        ahpload.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%50==0){
          node.remake_loadsc1();
        }
        if(y == z){
          hslocal ++;
        }
        hstotal += hsfinish - hsstart;
      }
      wt.addtext("Highest speed average time to send a process to a
node(nanoSec): ", String.valueOf(hstotal/1000));
      wt.addtext("average load from nodes which choose with Highest
speed (%):", String.valueOf(hslavg/1000));
      wt.addtext("average speed from nodes which choose with Highest
speed :", String.valueOf(hssavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with Highest speed (%):", String.valueOf(hsdlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
with Highest speed :", String.valueOf(hsdsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(hslocal));
      wt.space();
      node.reset_list();
      nodes = 500;
      node = new Node(nodes);
      node.setlan(nodes); //set up the lan's specs
      node.set_list(nodes);
      //1000 processes insert with AHP method
      ahplavg = 0;topsislavg = 0;
      wpmlavg = 0;ahpsavg = 0;
      topsissavg = 0;wpmsavg = 0; ahpdlavg = 0;
      topsisdlavg = 0;wpmdlavg = 0;
      ahpdsavg = 0;topsisdsavg = 0;wpmdsavg = 0;
      lllavg = 0;llsavg = 0;lldlavg = 0; lldsavg = 0;
      hslavg = 0;hssavg = 0;hsdlavg = 0; hsdsavg = 0;
      lllocal = 0; hslocal = 0;
      wt.addtext("nodes: ", "500");
      wt.space();
      for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        ahpstart = System.nanoTime();
        A.setrank(node.get_nodelist(z));
        y = (int)A.getrank(z);
        ahpfinish = System.nanoTime();
        ahplavg += node.get_load(y);
        ahpsavg += node.get_speed(y);
        ahpdsavg += (node.get_speed(y) - node.get_maxspeed());
        ahpdlavg += (node.get_load(y) - node.get_minload());
        ahpload.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%50==0){
          node.remake_loadsc1();
        }
        if(y == z){
          ahplocal ++;
```

```
        }
        ahptotal += ahpfinish -ahpstart;
      }
      wt.addtext("AHP average time to send a process to a
node(nanoSec): ", String.valueOf(ahptotal/1000));
      wt.addtext("average load from nodes which choose with AHP
(%):", String.valueOf(ahplavg/1000));
      wt.addtext("average speed from nodes which choose with AHP :",
String.valueOf(ahpsavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with AHP (%):", String.valueOf(ahpdlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
with AHP :", String.valueOf(ahpdsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(ahplocal));
      wt.space();
      node.reset_list();
      //1000 processes isert with Topsis method /*
       for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        topsisstart = System.nanoTime();
        T = new TOPSIS(node.get_nodelist(z)); // call TOPSIS
        y = (int)T.getpi(z);
        topsisfinish = System.nanoTime();
        topsislavg += node.get_load(y);
        topsissavg += node.get_speed(y);
        topsisdsavg += (node.get_speed(y) - node.get_maxspeed());
        topsisdlavg += (node.get_load(y) - node.get_minload());
        topsisload.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%50==0){
          node.remake_loadsc1();
        }
        if(y == z){
          topsislocal ++;
        }
        topsistotal += topsisfinish - topsisstart;
      }
      wt.addtext("Topsis average time to send a process to a
node(nanoSec): ", String.valueOf(topsistotal/1000));
      wt.addtext("average load from nodes which choose with Topsis
(%):", String.valueOf(topsislavg/1000));
      wt.addtext("average speed from nodes which choose with Topsis
:", String.valueOf(topsissavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with Topsis (%):", String.valueOf(topsisdlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
with Topsis :", String.valueOf(topsisdsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(topsislocal));
      wt.space();
      node.reset_list();
      //1000 processe insert with WPM method
      for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        wpmstart = System.nanoTime();
        W = new WPM(node.get_nodelist(z)); // call WPM
        y = (int)W.getnormal(z);
        wpmfinish = System.nanoTime();
        wpmlavg += node.get_load(y);
```

```
        wpmsavg += node.get_speed(y);
        wpmdsavg += (node.get_speed(y) - node.get_maxspeed());
        wpmdlavg += (node.get_load(y) - node.get_minload());
        wpmload.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%50==0){
          node.remake_loadsc1();
        }
        if(y == z){
          wpmlocal ++;
        }
        wpmtotal += wpmfinish - wpmstart;
      }
      wt.addtext("WPM average time to send a process to a
node(nanoSec): ", String.valueOf(wpmtotal/1000));
      wt.addtext("average load from nodes which choose with WPM
(%):", String.valueOf(wpmlavg/1000));
      wt.addtext("average speed from nodes which choose with WPM :",
String.valueOf(wpmsavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with WPM (%):", String.valueOf(wpmdlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
with WPM :", String.valueOf(wpmdsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(wpmlocal));
      node.reset_list();
      wt.space();
      //1000 processe insert with lower load method
      for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        llstart = System.nanoTime();
        ll.find_lowestload(node.get_nodelist(z));
        y = (int)ll.getid();
        llfinish = System.nanoTime();
        lllavg += node.get_load(y);
        llsavg += node.get_speed(y);
        lldsavg += (node.get_speed(y) - node.get_maxspeed());
        lldlavg += ((node.get_load(y) - node.get_minload()));
        llload.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%50==0){
          node.remake_loadsc1();
        }
        if(y == z){
          lllocal ++;
        }
        lltotal += llfinish - llstart;
      }
      wt.addtext("Lower load average time to send a process to a
node(nanoSec): ", String.valueOf(lltotal/1000));
      wt.addtext("average load from nodes which choose with Lower
load (%):", String.valueOf(lllavg/1000));
      wt.addtext("average speed from nodes which choose with Lower
load :", String.valueOf(llsavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with Lower load (%):", String.valueOf(lldlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
with Lower load :", String.valueOf(lldsavg/1000));
```

```
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(lllocal));
      wt.space();
      node.reset_list();
      //1000 processe insert with highest speed method
      for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        hsstart = System.nanoTime();
        hs.find_highestspeed(node.get_nodelist(z));
        y = (int)hs.getid();
        hsfinish = System.nanoTime();
        hslavg += node.get_load(y);
        hssavg += node.get_speed(y);
        hsdsavg += (node.get_speed(y) - node.get_maxspeed());
        hsdlavg += (node.get_load(y) - node.get_minload());
        ahpload.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%50==0){
          node.remake_loadsc1();
        }
        if(y == z){
          hslocal ++;
        }
        hstotal += hsfinish - hsstart;
      }
      wt.addtext("Highest speed average time to send a process to a
node(nanoSec): ", String.valueOf(hstotal/1000));
      wt.addtext("average load from nodes which choose with Highest
speed (%):", String.valueOf(hslavg/1000));
      wt.addtext("average speed from nodes which choose with Highest
speed :", String.valueOf(hssavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with Highest speed (%):", String.valueOf(hsdlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
with Highest speed :", String.valueOf(hsdsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(hslocal));
      wt.space();
      node.reset_list();
      wt.closeFile();
      JDialog d = new JDialog(frame_hp, "Scenario 1");
      JTextArea l = new JTextArea();
      l.setRows(10);
      l.setText("Scenario 1 just completed!!!\n"
          + "We add 1000 processes in the lan\n"
      +"We take metrics about AHP,TOPSIS,Highest Speed, Lower load
and WMP method\n"
          + "Finaly we save the numbers in a text file in source file
");
      l.setLineWrap(true);
      l.setWrapStyleWord(true);
      d.add(l);
      d.setSize(300,300);
      d.setVisible(true);


    }
    else if(e.getSource() == b3){
      wt.open_filesc2();
```

```
      int i,y=0,ahplocal = 0,topsislocal = 0,wpmlocal = 0,lllocal =
0,hslocal = 0;
      int nodes;
      long ahpstart = 0,ahpfinish = 0,ahptotal = 0,
          topsisstart = 0,topsisfinish = 0,
          topsistotal = 0,wpmstart = 0,wpmfinish = 0,wpmtotal =
0,llstart = 0,llfinish = 0,lltotal = 0,
          hsstart = 0,hsfinish = 0,hstotal = 0;
      double [] priority = new double [3];
      double ahplavg = 0,topsislavg = 0,wpmlavg = 0;
      double ahpsavg = 0,topsissavg = 0,wpmsavg = 0;
      double ahpdlavg = 0,topsisdlavg = 0,wpmdlavg = 0;
      double ahpdsavg = 0,topsisdsavg = 0,wpmdsavg = 0;
      double lllavg = 0,llsavg = 0,lldlavg = 0, lldsavg = 0;
      double hslavg = 0,hssavg = 0,hsdlavg = 0, hsdsavg = 0;
      ArrayList<Double> ahpload50 = new ArrayList();
      ArrayList<Double> topsisload50 = new ArrayList();
      ArrayList<Double> wpmload50 = new ArrayList();
      ArrayList<Double> ahpload100 = new ArrayList();
      ArrayList<Double> topsisload100 = new ArrayList();
      ArrayList<Double> wpmload100 = new ArrayList();
      ArrayList<Double> ahpload500 = new ArrayList();
      ArrayList<Double> topsisload500 = new ArrayList();
      ArrayList<Double> wpmload500 = new ArrayList();
      ArrayList<Double> llload50 = new ArrayList();
      ArrayList<Double> llload100 = new ArrayList();
      ArrayList<Double> llload500 = new ArrayList();
      ArrayList<Double> hsload50 = new ArrayList();
      ArrayList<Double> hsload100 = new ArrayList();
      ArrayList<Double> hsload500 = new ArrayList();
      nodes = 50; // number of the nodes static
      AHP A = new AHP(3); //call ahp
      WPM W;//initial weighted product model
      TOPSIS T;
      Lower_load ll = new Lower_load();
      Highest_speed hs = new Highest_speed();
      Node node = new Node(nodes); // init the nodes
      /*set pairwise matrix*/
      priority[0] = 3.0;
      priority[1] = 5.0;
      priority[2] = 3.0;

      A.setpairwisemtr(priority);

      node.setlan(nodes); //set up the lan's specs
      node.set_list(nodes);
      wt.addtext("Scenario ", "2");
      wt.space();
      wt.addtext("nodes: ", "50");
      wt.space();
      //1000 processes insert with AHP method
      for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        ahpstart = System.nanoTime();
        A.setrank(node.get_nodelist(z));
        y = (int)A.getrank(z);
        ahpfinish = System.nanoTime();
        ahplavg += node.get_load(y);
        ahpsavg += node.get_speed(y);
        ahpdsavg += (node.get_speed(y) - node.get_maxspeed());
        ahpdlavg += (node.get_load(y) - node.get_minload());
```

```
        ahpload50.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%3==0){
          node.remake_loadsc2();
        }
        if(y == z){
          ahplocal ++;
        }
        ahptotal += ahpfinish -ahpstart;
      }
      wt.space();
      wt.addtext("AHP average time to send a process to a
node(nanoSec): ", String.valueOf(ahptotal/1000));
      wt.addtext("average load from nodes which choose with AHP
(%):", String.valueOf(ahplavg/1000));
      wt.addtext("average speed from nodes which choose with AHP :",
String.valueOf(ahpsavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with AHP (%):", String.valueOf(ahpdlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
with AHP :", String.valueOf(ahpdsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(ahplocal));
      wt.space();
      node.reset_list();
      //1000 processes isert with Topsis method /*
       for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        topsisstart = System.nanoTime();
        T = new TOPSIS(node.get_nodelist(z)); // call TOPSIS
        y = (int)T.getpi(z);
        topsisfinish = System.nanoTime();
        topsislavg += node.get_load(y);
        topsissavg += node.get_speed(y);
        topsisdsavg += (node.get_speed(y) - node.get_maxspeed());
        topsisdlavg += (node.get_load(y) - node.get_minload());
        topsisload50.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%3==0){
          node.remake_loadsc2();
        }
        if(y == z){
          topsislocal ++;
        }
        topsistotal += topsisfinish - topsisstart;
      }
      wt.addtext("Topsis average time to send a process to a
node(nanoSec): ", String.valueOf(topsistotal/1000));
      wt.addtext("average load from nodes which choose with Topsis
(%):", String.valueOf(topsislavg/1000));
      wt.addtext("average speed from nodes which choose with Topsis
:", String.valueOf(topsissavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with Topsis (%):", String.valueOf(topsisdlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
with Topsis :", String.valueOf(topsisdsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(topsislocal));
      wt.space();
```

```
      node.reset_list();
      //1000 processe insert with WPM method
      for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        wpmstart = System.nanoTime();
        W = new WPM(node.get_nodelist(z)); // call WPM
        y = (int)W.getnormal(z);
        wpmfinish = System.nanoTime();
        wpmlavg += node.get_load(y);
        wpmsavg += node.get_speed(y);
        wpmdsavg += (node.get_speed(y) - node.get_maxspeed());
        wpmdlavg += (node.get_load(y) - node.get_minload());
        wpmload50.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%3==0){
          node.remake_loadsc2();
        }
        if(y == z){
          wpmlocal ++;
        }
        wpmtotal += wpmfinish - wpmstart;
      }
      wt.addtext("WPM average time to send a process to a
node(nanoSec): ", String.valueOf(wpmtotal/1000));
      wt.addtext("average load from nodes which choose with WPM
(%):", String.valueOf(wpmlavg/1000));
      wt.addtext("average speed from nodes which choose with WPM :",
String.valueOf(wpmsavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with WPM (%):", String.valueOf(wpmdlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
with WPM :", String.valueOf(wpmdsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(wpmlocal));
      node.reset_list();
      wt.space();
      //1000 processe insert with lower load method
      for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        llstart = System.nanoTime();
        ll.find_lowestload(node.get_nodelist(z));
        y = (int)ll.getid();
        llfinish = System.nanoTime();
        lllavg += node.get_load(y);
        llsavg += node.get_speed(y);
        lldsavg += (node.get_speed(y) - node.get_maxspeed());
        lldlavg += ((node.get_load(y) - node.get_minload()));
        llload50.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%3==0){
          node.remake_loadsc2();
        }
        if(y == z){
          lllocal ++;
        }
        lltotal += llfinish - llstart;
      }
      wt.addtext("Lower load average time to send a process to a
node(nanoSec): ", String.valueOf(lltotal/1000));
```

```
      wt.addtext("average load from nodes which choose with Lower
load (%):", String.valueOf(lllavg/1000));
      wt.addtext("average speed from nodes which choose with Lower
load :", String.valueOf(llsavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with Lower load (%):", String.valueOf(lldlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
with Lower load :", String.valueOf(lldsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(lllocal));
      wt.space();
      node.reset_list();
      //1000 processe insert with highest speed method
      for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        hsstart = System.nanoTime();
        hs.find_highestspeed(node.get_nodelist(z));
        y = (int)hs.getid();
        hsfinish = System.nanoTime();
        hslavg += node.get_load(y);
        hssavg += node.get_speed(y);
        hsdsavg += (node.get_speed(y) - node.get_maxspeed());
        hsdlavg += (node.get_load(y) - node.get_minload());
        hsload50.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%3==0){
          node.remake_loadsc2();
        }
        if(y == z){
          hslocal ++;
        }
        hstotal += hsfinish - hsstart;
      }
      wt.addtext("Highest speed average time to send a process to a
node(nanoSec): ", String.valueOf(hstotal/1000));
      wt.addtext("average load from nodes which choose with Highest
speed (%):", String.valueOf(hslavg/1000));
      wt.addtext("average speed from nodes which choose with Highest
speed :", String.valueOf(hssavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with Highest speed (%):", String.valueOf(hsdlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
with Highest speed :", String.valueOf(hsdsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(hslocal));
      wt.space();
      node.reset_list();
      nodes = 100;
      node = new Node(nodes);
      node.setlan(nodes); //set up the lan's specs
      node.set_list(nodes);
      //1000 processes insert with AHP method
      ahplavg = 0;topsislavg = 0;
      wpmlavg = 0;ahpsavg = 0;
      topsissavg = 0;wpmsavg = 0; ahpdlavg = 0;
      topsisdlavg = 0;wpmdlavg = 0;
      ahpdsavg = 0;topsisdsavg = 0;wpmdsavg = 0;
      lllavg = 0;llsavg = 0;lldlavg = 0; lldsavg = 0;
      hslavg = 0;hssavg = 0;hsdlavg = 0; hsdsavg = 0;
      ahplocal = 0;topsislocal = 0;wpmlocal = 0;
```

```
      lllocal = 0; hslocal = 0;
      wt.addtext("nodes:", "100");
      wt.space();
      //1000 processes insert with AHP method
      for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        ahpstart = System.nanoTime();
        A.setrank(node.get_nodelist(z));
        y = (int)A.getrank(z);
        ahpfinish = System.nanoTime();
        ahplavg += node.get_load(y);
        ahpsavg += node.get_speed(y);
        ahpdsavg += (node.get_speed(y) - node.get_maxspeed());
        ahpdlavg += (node.get_load(y) - node.get_minload());
        ahpload100.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%3==0){
          node.remake_loadsc2();
        }
        if(y == z){
          ahplocal ++;
        }
        ahptotal += ahpfinish -ahpstart;
      }
      wt.space();
      wt.addtext("AHP average time to send a process to a
node(nanoSec): ", String.valueOf(ahptotal/1000));
      wt.addtext("average load from nodes which choose with AHP
(%):", String.valueOf(ahplavg/1000));
      wt.addtext("average speed from nodes which choose with AHP :",
String.valueOf(ahpsavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with AHP (%):", String.valueOf(ahpdlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
with AHP :", String.valueOf(ahpdsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(ahplocal));
      wt.space();
      node.reset_list();
      //1000 processes insert with Topsis method /*
       for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        topsisstart = System.nanoTime();
        T = new TOPSIS(node.get_nodelist(z)); // call TOPSIS
        y = (int)T.getpi(z);
        topsisfinish = System.nanoTime();
        topsislavg += node.get_load(y);
        topsissavg += node.get_speed(y);
        topsisdsavg += (node.get_speed(y) - node.get_maxspeed());
        topsisdlavg += (node.get_load(y) - node.get_minload());
        topsisload100.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%3==0){
          node.remake_loadsc2();
        }
        if(y == z){
          topsislocal ++;
        }
        topsistotal += topsisfinish - topsisstart;
```

```
      }
      wt.addtext("Topsis average time to send a process to a
node(nanoSec): ", String.valueOf(topsistotal/1000));
      wt.addtext("average load from nodes which choose with Topsis
(%):", String.valueOf(topsislavg/1000));
      wt.addtext("average speed from nodes which choose with Topsis
:", String.valueOf(topsissavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with Topsis (%):", String.valueOf(topsisdlavg/1000));
      wt.addtext("average speed from fastest nodeto node we choose
with Topsis :", String.valueOf(topsisdsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(topsislocal));
      wt.space();
      node.reset_list();
      //1000 processe insert with WPM method
      for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        wpmstart = System.nanoTime();
        W = new WPM(node.get_nodelist(z)); // call WPM
        y = (int)W.getnormal(z);
        wpmfinish = System.nanoTime();
        wpmlavg += node.get_load(y);
        wpmsavg += node.get_speed(y);
        wpmdsavg += (node.get_speed(y) - node.get_maxspeed());
        wpmdlavg += (node.get_load(y) - node.get_minload());
        wpmload100.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%3==0){
          node.remake_loadsc2();
        }
        if(y == z){
          wpmlocal ++;
        }
        wpmtotal += wpmfinish - wpmstart;
      }
      wt.addtext("WPM average time to send a process to a
node(nanoSec): ", String.valueOf(wpmtotal/1000));
      wt.addtext("average load from nodes which choose with WPM
(%):", String.valueOf(wpmlavg/1000));
      wt.addtext("average speed from nodes which choose with WPM :",
String.valueOf(wpmsavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with WPM (%):", String.valueOf(wpmdlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
with WPM :", String.valueOf(wpmdsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(wpmlocal));
      node.reset_list();
      wt.space();
      //1000 processe insert with lower load method
      for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        llstart = System.nanoTime();
        ll.find_lowestload(node.get_nodelist(z));
        y = (int)ll.getid();
        llfinish = System.nanoTime();
        lllavg += node.get_load(y);
        llsavg += node.get_speed(y);
        lldsavg += (node.get_speed(y) - node.get_maxspeed());
```

```
        lldlavg += ((node.get_load(y) - node.get_minload()));
        llload100.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%3==0){
          node.remake_loadsc2();
        }
        if(y == z){
          lllocal ++;
        }
        lltotal += llfinish - llstart;
      }
      wt.addtext("Lower load average time to send a process to a
node(nanoSec): ", String.valueOf(lltotal/1000));
      wt.addtext("average load from nodes which choose with Lower
load (%):", String.valueOf(lllavg/1000));
      wt.addtext("average speed from nodes which choose with Lower
load :", String.valueOf(llsavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with Lower load (%):", String.valueOf(lldlavg/1000));
      wt.addtext("average speed from fastest nodeto node we choose
with Lower load :", String.valueOf(lldsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(lllocal));
      wt.space();
      node.reset_list();
      //1000 processe insert with highest speed method
      for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        hsstart = System.nanoTime();
        hs.find_highestspeed(node.get_nodelist(z));
        y = (int)hs.getid();
        hsfinish = System.nanoTime();
        hslavg += node.get_load(y);
        hssavg += node.get_speed(y);
        hsdsavg += (node.get_speed(y) - node.get_maxspeed());
        hsdlavg += (node.get_load(y) - node.get_minload());
        hsload100.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%3==0){
          node.remake_loadsc2();
        }
        if(y == z){
          hslocal ++;
        }
        hstotal += hsfinish - hsstart;
      }
      wt.addtext("Highest speed average time to send a process to a
node(nanoSec): ", String.valueOf(hstotal/1000));
      wt.addtext("average load from nodes which choose with Highest
speed (%):", String.valueOf(hslavg/1000));
      wt.addtext("average speed from nodes which choose with Highest
speed :", String.valueOf(hssavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with Highest speed (%):", String.valueOf(hsdlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
with Highest speed :", String.valueOf(hsdsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(hslocal));
      wt.space();
```

```
        node.reset_list();
        nodes = 500;
        node = new Node(nodes);
        node.setlan(nodes); //set up the lan's specs
        node.set_list(nodes);
        //1000 processes insert with AHP method
        ahplavg = 0;topsislavg = 0;
        wpmlavg = 0;ahpsavg = 0;
        topsissavg = 0;wpmsavg = 0; ahpdlavg = 0;
        topsisdlavg = 0;wpmdlavg = 0;
        ahpdsavg = 0;topsisdsavg = 0;wpmdsavg = 0;
        lllavg = 0;llsavg = 0;lldlavg = 0; lldsavg = 0;
        hslavg = 0;hssavg = 0;hsdlavg = 0; hsdsavg = 0;
        ahplocal = 0;topsislocal = 0;wpmlocal = 0;
        lllocal = 0; hslocal = 0;
        wt.addtext("nodes:", "1000");
        wt.space();
        //1000 processes insert with AHP method
        for(i=0;i<1000;i++){
          int z = (int)(Math.random() * (nodes-1)) + 1;
          ahpstart = System.nanoTime();
          A.setrank(node.get_nodelist(z));
          y = (int)A.getrank(z);
          ahpfinish = System.nanoTime();
          ahplavg += node.get_load(y);
          ahpsavg += node.get_speed(y);
          ahpdsavg += (node.get_speed(y) - node.get_maxspeed());
          ahpdlavg += (node.get_load(y) - node.get_minload());
          ahpload500.add(node.get_load(y));
          node.plusloadsc1(y); //assign the process random in a node
          node.remake_listsc1();
          if(i%3==0){
            node.remake_loadsc2();
          }
          if(y == z){
            ahplocal ++;
          }
          ahptotal += ahpfinish -ahpstart;
        }
        wt.space();
        wt.addtext("AHP average time to send a process to a
node(nanoSec): ", String.valueOf(ahptotal/1000));
        wt.addtext("average load from nodes which choose with AHP
(%):", String.valueOf(ahplavg/1000));
        wt.addtext("average speed from nodes which choose with AHP :",
String.valueOf(ahpsavg/1000));
        wt.addtext("average load of diference from smaller load to node
which choose with AHP (%):", String.valueOf(ahpdlavg/1000));
        wt.addtext("average speed from fastest nodeto node we choose
with AHP :", String.valueOf(ahpdsavg/1000));
        wt.addtext("processes which run local(out of 1000):",
String.valueOf(ahplocal));
        wt.space();
        node.reset_list();
        //1000 processes insert with Topsis method /*
         for(i=0;i<1000;i++){
          int z = (int)(Math.random() * (nodes-1)) + 1;
          topsisstart = System.nanoTime();
          T = new TOPSIS(node.get_nodelist(z)); // call TOPSIS
          y = (int)T.getpi(z);
          topsisfinish = System.nanoTime();
```

```
        topsislavg += node.get_load(y);
        topsissavg += node.get_speed(y);
        topsisdsavg += (node.get_speed(y) - node.get_maxspeed());
        topsisdlavg += (node.get_load(y) - node.get_minload());
        topsisload500.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%3==0){
          node.remake_loadsc2();
        }
        if(y == z){
          topsislocal ++;
        }
        topsistotal += topsisfinish - topsisstart;
      }
      wt.addtext("Topsis average time to send a process to a
node(nanoSec): ", String.valueOf(topsistotal/1000));
      wt.addtext("average load from nodes which choose with Topsis
(%):", String.valueOf(topsislavg/1000));
      wt.addtext("average speed from nodes which choose with Topsis
:", String.valueOf(topsissavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with Topsis (%):", String.valueOf(topsisdlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
with Topsis :", String.valueOf(topsisdsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(topsislocal));
      wt.space();
      node.reset_list();
      //1000 processe insert with WPM method
      for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        wpmstart = System.nanoTime();
        W = new WPM(node.get_nodelist(z)); // call WPM
        y = (int)W.getnormal(z);
        wpmfinish = System.nanoTime();
        wpmlavg += node.get_load(y);
        wpmsavg += node.get_speed(y);
        wpmdsavg += (node.get_speed(y) - node.get_maxspeed());
        wpmdlavg += (node.get_load(y) - node.get_minload());
        wpmload500.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%3==0){
          node.remake_loadsc2();
        }
        if(y == z){
          wpmlocal ++;
        }
        wpmtotal += wpmfinish - wpmstart;
      }
      wt.addtext("WPM average time to send a process to a
node(nanoSec): ", String.valueOf(wpmtotal/1000));
      wt.addtext("average load from nodes which choose with WPM
(%):", String.valueOf(wpmlavg/1000));
      wt.addtext("average speed from nodes which choose with WPM :",
String.valueOf(wpmsavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with WPM (%):", String.valueOf(wpmdlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
from WPM :", String.valueOf(wpmdsavg/1000));
```

```
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(wpmlocal));
      node.reset_list();
      wt.space();
      //1000 processe insert with lower load method
      for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        llstart = System.nanoTime();
        ll.find_lowestload(node.get_nodelist(z));
        y = (int)ll.getid();
        llfinish = System.nanoTime();
        lllavg += node.get_load(y);
        llsavg += node.get_speed(y);
        lldsavg += (node.get_speed(y) - node.get_maxspeed());
        lldlavg += ((node.get_load(y) - node.get_minload()));
        llload500.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%3==0){
          node.remake_loadsc2();
        }
        if(y == z){
          lllocal ++;
        }
        lltotal += llfinish - llstart;
      }
      wt.addtext("Lower load average time to send a process to a
node(nanoSec): ", String.valueOf(lltotal/1000));
      wt.addtext("average load from nodes which choose with Lower
load (%):", String.valueOf(lllavg/1000));
      wt.addtext("average speed from nodes which choose with Lower
load :", String.valueOf(llsavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with Lower load (%):", String.valueOf(lldlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
with Lower load :", String.valueOf(lldsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(lllocal));
      wt.space();
      node.reset_list();
      //1000 processe insert with highest speed method
      for(i=0;i<1000;i++){
        int z = (int)(Math.random() * (nodes-1)) + 1;
        hsstart = System.nanoTime();
        hs.find_highestspeed(node.get_nodelist(z));
        y = (int)hs.getid();
        hsfinish = System.nanoTime();
        hslavg += node.get_load(y);
        hssavg += node.get_speed(y);
        hsdsavg += (node.get_speed(y) - node.get_maxspeed());
        hsdlavg += (node.get_load(y) - node.get_minload());
        hsload500.add(node.get_load(y));
        node.plusloadsc1(y); //assign the process random in a node
        node.remake_listsc1();
        if(i%3==0){
          node.remake_loadsc2();
        }
        if(y == z){
          hslocal ++;
        }
        hstotal += hsfinish - hsstart;
```

```
      }
      wt.addtext("Highest speed average time to send a process to a
node(nanoSec): ", String.valueOf(hstotal/1000));
      wt.addtext("average load from nodes which choose with Highest
speed (%):", String.valueOf(hslavg/1000));
      wt.addtext("average speed from nodes which choose with Highest
speed :", String.valueOf(hssavg/1000));
      wt.addtext("average load of diference from smaller load to node
which choose with Highest speed (%):", String.valueOf(hsdlavg/1000));
      wt.addtext("average speed from fastest node to node we choose
from Highest speed :", String.valueOf(hsdsavg/1000));
      wt.addtext("processes which run local(out of 1000):",
String.valueOf(hslocal));
      wt.closeFilesc2();
      JDialog d = new JDialog(frame_hp, "Scenario 2");
      JTextArea l = new JTextArea();
      l.setRows(10);
      l.setText("Scenario 2 just completed!!!\n"
          + "We add 1000 processes in the lan\n"
      +"We take metrics about AHP,TOPSIS and WMP method\n"
          + "Finaly we save the numbers in a text file in source file
");
      l.setLineWrap(true);
      l.setWrapStyleWord(true);
      d.add(l);
      d.setSize(300,300);
      d.setVisible(true);
    }



  }

}
```

## Write to file.java

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package project.test;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException; // error handling
import java.util.Formatter;
/**
 *
 * @author Κωστής
 */
public class Write_to_file {

   FileWriter f;

  public void open_file () {
    try{
      f = new FileWriter("Scenario_1.txt");
    }catch(IOException e){
```

```
      System.out.println(e.getMessage());
    }
  }

  public void open_filesc2() {
    try{
      f = new FileWriter("Scenario_2.txt");
    }catch(IOException e){
      System.out.println(e.getMessage());
    }
  }

  public void addtext(String tmp,String tmp2){
    try{
      f.write(tmp);
      f.write("\t");
      f.write(tmp2);
      f.write("\n");
    }catch(IOException e){
      System.out.println(e.getMessage());
    }
  }



  public void space(){
     try{
      f.write("\n");
    }catch(IOException e){
      System.out.println(e.getMessage());
    }
  }

  public void closeFile(){
     try{
      f.close();
    }catch(IOException e){
      System.out.println(e.getMessage());
    }
  }

  public void closeFilesc2(){
     try{
      f.close();
    }catch(IOException e){
      System.out.println(e.getMessage());
    }
  }
}
```

## Ahp.java

```
package project.test;


import java.util.Arrays;

public class AHP{
  int num_atr;
  double[][] pairwisemtr,tmp;
  double[] criteria_weight;
```

```
  double [] weighted_sum,sum;
  double RI[] = {0.0, 0.0, 0.58, 0.9, 1.12, 1.24, 1.32, 1.41, 1.45,
1.49};
  double[] larray;
  double ci,lmax,cr;
  double[][] rank;
  double[] min,max;


  AHP(int num_atr){
    this.num_atr = num_atr;
    pairwisemtr = new double [num_atr][num_atr];
    tmp = new double [num_atr][num_atr];
    criteria_weight = new double [num_atr];
    weighted_sum = new double [num_atr];
    larray = new double [num_atr];
    sum = new double [num_atr];
    min = new double [num_atr];
    max = new double [num_atr];
    /**/
    for(int i = 0;i < num_atr ; i++){
      for(int j = 0;j < num_atr ; j++){
        if(i==j) pairwisemtr[i][j] = 1.0;
      }
    }
  }

  public void setpairwisemtr(double[] temp){
    int x = 0;
    for(int i=0;i< num_atr;i++){
      for(int j=i+1;j< num_atr;j++){
        pairwisemtr[i][j] = temp[x];
        pairwisemtr[j][i] = 1.0/temp[x];
        x++;
      }
    }


    for(int i=0;i< num_atr;i++){
      for(int j=0;j< num_atr;j++){
        tmp[i][j]=pairwisemtr[i][j];
      }
    }

    for(int i=0;i< num_atr;i++){
      for(int j=0;j< num_atr;j++){
        sum[j] = pairwisemtr[i][j] + sum[j];
      }
    }
    for(int i=0;i< num_atr;i++){
      for(int j=0;j< num_atr;j++){
        pairwisemtr[i][j] = pairwisemtr[i][j]/sum[j];
      }
    }

    for(int i=0;i< num_atr;i++){
      for(int j=0;j< num_atr;j++){
        criteria_weight[i] += pairwisemtr[i][j];
      }
      criteria_weight[i] = criteria_weight[i]/num_atr;
    }
```

```
  for(int i=0;i< num_atr;i++){
    for(int j=0;j< num_atr;j++){
      tmp[i][j] = tmp[i][j]*criteria_weight[j];

    }
  }
  for(int i=0;i< num_atr;i++){
    for(int j=0;j< num_atr;j++){
      weighted_sum[i] = tmp[i][j] + weighted_sum[i];
    }
  }

  for(int i=0;i< num_atr;i++){
    larray[i] = weighted_sum[i]/criteria_weight[i];
  }

}

public double getlmax() {
  lmax=0;
  for(int i=0;i< num_atr;i++){
    lmax += larray[i];
  }
  return lmax/3;
}

public double getci() {
  return ci=(getlmax()-num_atr)/(num_atr-1);
}

public double getcr(){
  return cr=ci/RI[num_atr];
}



public void setrank(double[][] b){
  int l = b.length,ll;
  for(int i=0;i< num_atr;i++){
    max[i] = 0;
    min[i] = 9999;
  }
  for(int i = 0;i < l ; i++){
    for(int j = 0;j < max.length ; j++){
      if(max[j] < b[i][j+1]){
        max[j] = b[i][j+1];
      }
      if(min[j] > b[i][j+1] ){
        min[j] = b[i][j+1];
      }
    }
  }
  for(int i = 0;i < l ; i++){
    for(int j = 1;j <max.length ; j++){
      if(j==1) b[i][j] = (b[i][j]/max[j]);
      if(j==2) b[i][j] = (b[i][j]/max[j]);
      if(j==3) b[i][j] = (min[j]/b[i][j]);
    }
```

```
    }
    rank = new double[b.length][b[0].length+1];
    ll = rank.length;
    for(int i = 0;i < ll ; i++){
      for(int j = 0;j < b[i].length ; j++){
        if(j<rank[i].length){
          rank[i][j] = b[i][j];
        }
        rank[i][4] = (rank[i][1] * criteria_weight[0]) + (rank[i][2]
           * criteria_weight[1]) + (rank[i][3] * criteria_weight[2]);
      }

    }
    double[] tmp = new double[rank[0].length];
    for(int i = 0;i < ll ; i++){
      for(int j = 1;j < (ll-i) ; j++){
        if(rank[j-1][4] < rank[j][4]){
          tmp[0] = rank[j-1][0];
          tmp[1] = rank[j-1][1];
          tmp[2] = rank[j-1][2];
          tmp[3] = rank[j-1][3];
          tmp[4] = rank[j-1][4];
          rank[j-1][0] = rank[j][0];
          rank[j-1][1] = rank[j][1];
          rank[j-1][2] = rank[j][2];
          rank[j-1][3] = rank[j][3];
          rank[j-1][4] = rank[j][4];
          rank[j][0] = tmp[0];
          rank[j][1] = tmp[1];
          rank[j][2] = tmp[2];
          rank[j][3] = tmp[3];
          rank[j][4] = tmp[4];
        }
      }
    }
    /*
    for(int i = 0;i < rank.length ; i++){
      System.out.println("id: "+rank[i][0]+"precent: "+rank[i][4]);
    }
     */
  }
  public double getrank(int x){
    return rank[x][0];
  }
}
```

Topsis.java

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package project.test;

import java.util.Arrays;
```

```java
/**
 *
 * @author KOSTIS
 */
public class TOPSIS {
  double col1,col2,col3;
  double[] weight = {0.6,0.15,0.25};
  double[] best = new double[3];
  double[] worst= new double[3];
  double[] siplus;
  double[] siminus;
  double[][] pi;



  TOPSIS(double[][] a){
    for(int i=0;i<a.length;i++){
      for(int j=0;j<a[i].length;j++){
        if(j==1) col1 = col1 + Math.pow(a[i][j], 2);
        if(j==2) col2 = col2 + Math.pow(a[i][j], 2);
        if(j==3) col3 = col3 + Math.pow(a[i][j], 2);
      }
    }
    col1 = Math.sqrt(col1);
    col2 = Math.sqrt(col2);
    col3 = Math.sqrt(col3);
    for(int i=0;i<a.length;i++){
      for(int j=0;j<a[i].length;j++){
        if(j==1) a[i][j] = a[i][j]/col1 ;
        if(j==2) a[i][j] = a[i][j]/col2 ;
        if(j==3) a[i][j] = a[i][j]/col3 ;
      }

    }
    for(int i=0;i<a.length;i++){
      for(int j=1;j<a[i].length;j++){
        if(j>0 && j<3){
          a[i][j] = a[i][j] * weight[j-1];
        }
      }

    }
    for(int i=0;i<best.length;i++){
      best[i] = 0;
      worst[i] = 0;
    }

    for(int i=0;i<a.length;i++){
      for(int j=1;j<a[i].length;j++){
        if(j==1){
          if(a[i][1]<best[0]) best[0]= a[i][1];
          if(a[i][1]>worst[0]) worst[0]= a[i][1];
        }
        if(j==2){
          if(a[i][2]>best[1]) best[1]= a[i][2];
          if(a[i][2]<worst[1]) worst[1]= a[i][2];
        }
        if(j==3) {
          if(a[i][3]<best[2]) best[2] = a[i][3];
          if(a[i][3]>worst[2]) worst[2] = a[i][3];
        }
```

```
      }
    }

    siplus = new double[a.length];
    siminus = new double[a.length];
    for(int i=0;i<siplus.length;i++){
      siplus[i] = Math.pow(Math.pow(a[i][1]-best[1], 2)
      + Math.pow(a[i][2]-best[2], 2) + Math.pow(a[i][3]-best[1], 2),
0.5);
      siminus[i] = Math.pow(Math.pow(a[i][1]-worst[1], 2)
     + Math.pow(a[i][2]-worst[2], 2) + Math.pow(a[i][3]-worst[1], 2),
0.5);
    }
    pi = new double[a.length][2];
    for(int i=0;i<pi.length;i++){
      pi[i][0] = a[i][0];
      pi[i][1] = siplus[i]/(siplus[i] + siminus[i]);
    }

    double[] tmp = new double[2];
    for(int i = 0;i < pi.length ; i++){
      for(int j = 1;j < (pi.length-i) ; j++){
        if(pi[j-1][1] < pi[j][1]){
          tmp[0] = pi[j-1][0];
          tmp[1] = pi[j-1][1];
          pi[j-1][0] = pi[j][0];
          pi[j-1][1] = pi[j][1];
          pi[j][0] = tmp[0];
          pi[j][1] = tmp[1];
        }
      }
    }
  }

  public double getpi(int x){
    return pi[x][0];
  }


}
```

WPM.java

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package project.test;

/**
 *
 * @author KOSTIS
 * WEIGHTED PRODUCT MODEL
```

```
 */
public class WPM {
  double[] weight = {0.6,0.15,0.25};
  double[] min = {99999.0,99999.0,99999.0};
  double[][] normal;
  double[] max = {0.0,0.0,0.0};
  double[] rank;
  double[] f;

  WPM(double[][] a){
    normal = new double [a.length][a[0].length+1];
    rank = new double [a.length];
    f = new double [a.length];

    for(int i=0;i<a.length;i++){
      for(int j=0;j<max.length;j++){
        if (a[i][j+1] > max[j]) {
          max[j] = a[i][j+1];
        }
        if (a[i][j+1] < min[j] && a[i][j+1] < 0) {
          min[j] = a[i][j+1];
        }
      }
    }

    for(int i=0;i<normal.length;i++){
      normal[i][0] = a[i][0];
      normal[i][1] = a[i][1]/min[0];
      normal[i][2] = a[i][2]/max[1];
      if(a[i][3] > 0){
        normal[i][3] = min[2]/a[i][3];
      }else{
        normal[i][3] = 1;
      }

      normal[i][4] = 0;

    }
    for(int i=0;i<normal.length;i++){
      for(int j=1;j<normal[i].length-1;j++){
        if(j>0 && j<3){
          normal[i][j] = Math.pow(normal[i][j],weight[j-1]);
        }
      }
    }
    for(int i=0;i<a.length;i++){
      normal[i][4] = normal[i][1] * normal[i][2] * normal[i][3];
    }
    double[] tmp = new double[normal[0].length];
    for(int i = 0;i < a.length ; i++){
      for(int j = 1;j < (a.length-i) ; j++){
        if(normal[j-1][4] < normal[j][4]){
          tmp[0] = normal[j-1][0];
          tmp[1] = normal[j-1][1];
          tmp[2] = normal[j-1][2];
          tmp[3] = normal[j-1][3];
          tmp[4] = normal[j-1][4];
          normal[j-1][0] = normal[j][0];
          normal[j-1][1] = normal[j][1];
          normal[j-1][2] = normal[j][2];
          normal[j-1][3] = normal[j][3];
```

```
            normal[j-1][4] = normal[j][4];
            normal[j][0] = tmp[0];
            normal[j][1] = tmp[1];
            normal[j][2] = tmp[2];
            normal[j][3] = tmp[3];
            normal[j][4] = tmp[4];
          }
        }
      }

  }

  public double getnormal(int x){
    return normal[x][0];
  }

}
```

## Highest speed.java

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package project.test;

/**
 *
 * @author Κωστής
 */
public class Highest_speed {
  double id,speed;

  public void find_highestspeed(double[][] x){
    id = 99999;speed = -1;
    for(int i=0;i<x.length;i++){
      if(x[i][2]>speed){
        id = x[i][0];
        speed = x[i][2];
      }
    }
  }

  public double getid(){
    return id;
  }
}
```

## Lower load.java

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```java
package project.test;

import java.util.Arrays;

/**
 *
 * @author Κωστής
 */
public class Lower_load {
  double id ,load;

  public void find_lowestload(double[][] x){
    load = 100000;
    id = 9999;
    for(int i=0;i<x.length;i++){
      if(x[i][1]<load){
        id = x[i][0];
        load = x[i][1];
      }
    }
  }
  public double getid(){
    return id;
  }
}
```