ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ & ΔΙΟΙΚΗΣΗΣ

Διπλωματική Εργασία

ΑΡΙΘΜΗΤΙΚΗ ΕΠΙΛΥΣΗ ΑΛΥΣΙΔΩΝ MARKOV ΜΕΓΑΛΩΝ ΔΙΑΣΤΑΣΕΩΝ ΓΙΑ ΒΙΟΜΗΧΑΝΙΚΕΣ ΓΡΑΜΜΕΣ ΠΑΡΑΓΩΓΗΣ

ΜΠΟΥΣΔΕΚΗΣ ΑΛΕΞΑΝΔΡΟΣ

Τριμελής εξεταστική επιτροπή:
Κουϊκόγλου Βασίλειος, Καθηγητής (επιβλέπων)
Νικολός Ιωάννης, Επίκουρος Καθηγητής
Τσιναράκης Γεώργιος, Δρ., Διδάσκων Π.Δ. 407

Χανιά, 2010

ΠΡΟΛΟΓΟΣ

Στο σημείο αυτό θα ήθελα να ευχαριστήσω ορισμένους ανθρώπους, των οποίων η συμβολή κατά τη διάρκεια των σπουδών μου ήταν σημαντική:

Τον επιβλέποντα της διπλωματικής μου εργασίας, Καθηγητή Βασίλειο Κουϊκόγλου για την στήριξη, τη βοήθεια και την καθοδήγηση που μου πρόσφερε κατά τη διάρκεια εκπόνησης της διπλωματικής, καθώς και για την άριστη συνεργασία που είχαμε.

Τον πατέρα μου Δημήτρη, τη μητέρα μου Βασιλική και την αδελφή μου Μαρία για την ηθική και υλική στήριξη που μου πρόσφεραν όλα αυτά τα χρόνια.

Όλους τους φίλους και γνωστούς που έκανα στα Χανιά και έκαναν την περίοδο της φοιτητικής μου ζωής ακόμα πιο όμορφη και ενδιαφέρουσα.

ΠΕΡΙΕΧΟΜΕΝΑ

1 ΕΙΣΑΓΩΓΗ 6

1.1 Αντικείμενο της Διπλωματικής Εργασίας 6

1.2 Σύνοψη Μεθόδων Ανάλυσης Γραμμών Παραγωγής 8

2 ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ 11

2.1 Μηχανές και Αποθήκες 11

2.2 Καταστάσεις 12

2.3 Αλυσίδες Markov και Εξισώσεις Chapman- Kolmogorov 14

2.4 Επίλυση της εξίσωσης Chapman- Kolmogorov 15

2.4.1 Μέθοδος Jacobi 15

2.4.2 Μέθοδος Successive Over-Relaxation (SOR) 15

2.4.3 Μέθοδος Gauss-Seidel 19

3 ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ 21

3.1 Μεταβάσεις 21

3.1.1 Βλάβη της μηχανής M_i 21

3.1.2 Επισκευή της μηχανής M_i 22

3.1.3 Παραγωγή από τη μηχανή M_i 22

3.2 Κωδικοποίηση-Αποκωδικοποίηση .... **Σφάλμα! Δεν έχει οριστεί σελιδοδείκτης.**

3.2.1 Κωδικοποίηση ..... **Σφάλμα! Δεν έχει οριστεί σελιδοδείκτης.**

3.2.2 Αποκωδικοποίηση ..... **Σφάλμα! Δεν έχει οριστεί σελιδοδείκτης.**

3.3 Έλεγχος σύγκλισης 28

3.4 Υπολογισμός μέτρων απόδοσης 28

3.4.1 Μέσος ρυθμός παραγωγής 28

3.4.2 Μέση στάθμη αποθήκης B_i 28

3.4.3 Μέση στάθμη όλων των αποθηκών της γραμμής παραγωγής 29

4 ΑΡΙΘΜΗΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ 30

4.1 Σύστημα 4 μηχανών 30

4.1.1 Αρχικές τιμές συστήματος 30

4.1.2 Μεταβολή ρυθμού παραγωγής μ_i 31

4.1.3 Μεταβολή ρυθμού βλάβης p_i 33

4.1.4 Μεταβολή ρυθμού επισκευής r_i 35

4.1.5 Μεταβολή χωρητικότητας C_i 37

5 ΣΥΜΠΕΡΑΣΜΑΤΑ- ΕΠΕΚΤΑΣΕΙΣ 41

6 ΒΙΒΛΙΟΓΡΑΦΙΑ 41

ΠΑΡΑΡΤΗΜΑ 43

Κώδικας του αλγορίθμου με τη μέθοδο Jacobi Over- Relaxation 43

Κώδικας του αλγορίθμου με τη μέθοδο Gauss- Seidel Over- Relaxation 52

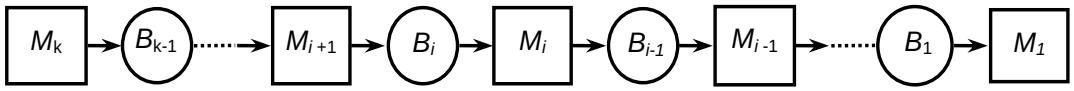
1 ΕΙΣΑΓΩΓΗ

1.1 Αντικείμενο της Διπλωματικής Εργασίας

Στην παρούσα διπλωματική εργασία, ασχολούμαστε με την ανάλυση των γραμμών παραγωγής και πιο συγκεκριμένα, με την αριθμητική επίλυση αλυσίδων Markov μεγάλων διαστάσεων για βιομηχανικές γραμμές παραγωγής.

Η βασική συμβολή της εργασίας είναι η ανάπτυξη ενός απλού αλγορίθμου για την κατάστρωση εξισώσεων. Άλλες προσεγγίσεις στο πρόβλημα αυτό (βλ. π.χ. [1]) οδήγησαν σε εξαιρετικά πολύπλοκους αλγορίθμους.

Η γραμμή παραγωγής είναι ένα σύνολο k μηχανών συνδεδεμένων σε σειρά με k-1 ενδιάμεσες αποθήκες, όπου στην πρώτη μηχανή M_k εισέρχονται πρώτες ύλες από μια πηγή άπειρων πρώτων υλών. Όταν ολοκληρωθεί η κατεργασία τους από τη μηχανή, προχωρούν και αποθηκεύονται στην αποθήκη που ακολουθεί. Στη συνέχεια, συνεχίζουν προς τη μηχανή M_{k-1}, την αποθήκη B_{k-1}, κ. ο. κ . Από την τελευταία μηχανή M_1 εξέρχεται το τελικό προϊόν και καταλήγει σε μια αποθήκη άπειρης χωρητικότητας. Το Σχήμα 1 αναπαριστά μια γραμμή παραγωγής με k μηχανές (τετράγωνα) και k-1 αποθήκες (κύκλοι):



Σχήμα 1: Γραμμή παραγωγής με k μηχανές και k-1 αποθήκες

Η μηχανή M_i τροφοδοτεί την αποθήκη B_{i-1} που έχει χωρητικότητα C_{i-1} κομμάτια, που περιμένουν επεξεργασία από τη M_{i-1}. Μια μηχανή αποκλείεται (μπλοκάρεται) όταν είναι έτοιμη να ελευθερώσει ένα κομμάτι και η αποθήκη της είναι γεμάτη. Δυαδικά, η M_{i+1} αποστερείται («πεινάει») όταν είναι έτοιμη να δεχθεί ένα κομμάτι και η αποθήκη που την τροφοδοτεί είναι άδεια. Παρόλα αυτά, η πρώτη μηχανή δεν αποστερείται ποτέ ενώ η τελευταία δεν μπλοκάρει ποτέ. Τα φαινόμενα αποκλεισμού και αποστέρησης αναγκάζουν τις μηχανές που επηρεάζονται να προσαρμόσουν το χρόνο κατεργασίας τους, παρατείνοντάς τον ανάλογα με την αιτία. Στην πράξη, εάν μια μηχανή αποστερηθεί, περιμένει μέχρι να υπάρξει διαθέσιμο προς επεξεργασία κομμάτι. Μετά το επεξεργάζεται και το ελευθερώνει προς την επόμενη αποθήκη, αλλά περιμένει ξανά μέχρι το επόμενο διαθέσιμο κομμάτι και ούτω καθεξής. Η μακροσκοπική της συμπεριφορά αναπαριστά παραγωγή με βραδύτερο ρυθμό. Μαζί με τον αποκλεισμό και την αποστέρηση, οι μηχανές μπορεί να αναγκαστούν να πάψουν να λειτουργούν προσωρινά, εξαιτίας διακοπής ρεύματος, μηχανικής βλάβης, αλλαγής εργαλείων, και προληπτικής συντήρησης. Διακοπές ρεύματος συμβαίνουν σε τυχαίες χρονικές στιγμές και είναι γνωστές ως βλάβες που εξαρτώνται από το χρόνο (χρονικής εξάρτησης). Τα άλλα γεγονότα εξαρτώνται από τη λειτουργία ή μη της μηχανής. Οι διακοπές εξαιτίας μηχανικής βλάβης και αλλαγής

εργαλείων οφείλονται στη φθορά χρήσης και λαμβάνουν χώρα μετά την παραγωγή ορισμένου αριθμού κομματιών. Η προληπτική συντήρηση συνήθως προβλέπεται μετά από έναν ορισμένο όγκο παραγωγής. Όλα αυτά τα φαινόμενα μπορούν εύκολα να ληφθούν υπόψη με την εισαγωγή κατάλληλών γεγονότων και μεταβλητών κατάστασης.

Οι αποθήκες έχουν πεπερασμένη χωρητικότητα, οι μηχανές παθαίνουν βλάβες και επισκευάζονται τυχαία και οι χρόνοι παραγωγής είναι τυχαίοι. Τα προβλήματα που παρουσιάζουν ενδιαφέρον στις γραμμές παραγωγής είναι η ανάλυση και η σύνθεση. Ανάλυση είναι η μέθοδος με την οποία υπολογίζονται οι μέσοι ρυθμοί παραγωγής, οι μέσες στάθμες στις αποθήκες κι άλλες παράμετροι που χαρακτηρίζουν μία γραμμή παραγωγής. Η ανάλυση των γραμμών παραγωγής γίνεται με τη βοήθεια μοντέλων προσομοίωσης ή μοντέλων Markov, τα οποία εξετάζουν τα διάφορα γεγονότα που λαμβάνουν χώρα στο σύστημα. Σύνθεση, εξάλλου, είναι το σύνολο των αποφάσεων που λαμβάνονται για τη σχεδίαση μιας νέας γραμμής παραγωγής, τη μεταβολή μιας υπάρχουσας και τον προγραμματισμό λειτουργίας της. Η λήψη αποφάσεων για το μέγεθος των αποθηκών, το είδος των μηχανών και τον τρόπο κυκλοφορίας των προϊόντων μέσα στη γραμμή, επηρεάζει τους ρυθμούς παραγωγής, τις μέσες στάθμες αποθηκών και κατά συνέπεια το κόστος λειτουργίας της. Έτσι, τις τελευταίες δεκαετίες υπάρχει συνεχές ερευνητικό ενδιαφέρον σχετικά με την ανάλυση γραμμών παραγωγής. Στην παρούσα διπλωματική εργασία ασχολούμαστε με την ανάλυση των γραμμών παραγωγής χρησιμοποιώντας μοντέλα Markov.

Θέλουμε να υπολογίσουμε τα εξής μέτρα απόδοσης: την παραγωγικότητα (TH) και το μέσο απόθεμα (N) που βρίσκεται στην γραμμή παραγωγής, δηλαδή το δεσμευμένο κεφάλαιο. Μπορούμε να χρησιμοποιήσουμε την παραγωγικότητα και το μέσο απόθεμα για να εφαρμόσουμε το νόμο Little και να υπολογίσουμε τον μέσο χρόνο καθυστέρησης μίας πρώτης ύλης μέχρι να παραχθεί, να βρούμε δηλαδή τον κύκλο παραγωγής.

Ο νομός του Little είναι:

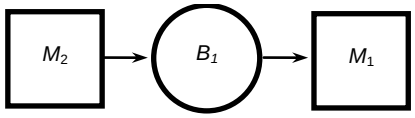
$$N = \lambda * T \Rightarrow T = \frac{N}{\lambda}$$, όπου αντί για λ έχουμε TH.

Για τον προσδιορισμό των μέτρων απόδοσης μέσο απόθεμα (N) και παραγωγικότητα (TH) του συστήματος απαιτείται η γνώση των πιθανοτήτων για όλες τις καταστάσεις του συστήματος. Όταν οι χρόνοι παραγωγής, βλάβης και επισκευής είναι εκθετικοί, το σύστημα περιγράφεται από μια αλυσίδα Markov και οι πιθανότητες υπολογίζονται λύνοντας ένα γραμμικό σύστημα εξισώσεων. Οι εξισώσεις αυτές είναι γνωστές ως εξισώσεις Chapman- Kolmogorov. Για την ανάπτυξη του μοντέλου λειτουργίας του συστήματος, θεωρούμε ότι η κάθε μηχανή μπορεί να έχει δύο καταστάσεις: 0 αν είναι χαλασμένη και 1 αν λειτουργεί, ενώ η κάθε αποθήκη έχει χωρητικότητα ίση με το άθροισμα των κομματιών που μπορούν να βρίσκονται στην αποθήκη και του κομματιού που μπορεί να βρίσκεται εντός της αντίστοιχης μηχανής και υφίσταται επεξεργασία. Για κάθε εφικτή κατάσταση, λοιπόν, της γραμμής παραγωγής εξετάζουμε αν μπορούμε να πάμε σε μια άλλη εφικτή κατάσταση και με τι ρυθμό μετάβασης. Σημειώνουμε ότι η γραμμή παραγωγής αναπαριστάται ως εξής:

$$M_k \rightarrow B_{k-1} \rightarrow M_{k-1} \rightarrow B_{k-2} \rightarrow \ldots \rightarrow B_2 \rightarrow M_2 \rightarrow B_1 \rightarrow M_1$$

Για παράδειγμα, αν έχουμε μια γραμμή παραγωγής με 2 μηχανές και 1 αποθήκη ($M_2 \rightarrow B_1 \rightarrow M_1$), όπου κάποια στιγμή η πρώτη μηχανή M_2 είναι χαλασμένη, στην αποθήκη βρίσκονται 2 κομμάτια και η δεύτερη μηχανή M_1 λειτουργεί, τότε η κατάσταση είναι 0-2-1 ($M_2 \rightarrow B_1 \rightarrow M_1$) και τα ακόλουθα γεγονότα μπορεί να συμβούν:

- Να επισκευαστεί η μηχανή M_2 και να προκύψει η κατάσταση: 1-2-1. Ο ρυθμός μετάβασης από την κατάσταση 0-2-1 στην 1-2-1 είναι ίσος με τον ρυθμό επισκευής της μηχανής M_2.
- Να χαλάσει η μηχανή M_1 και να προκύψει η κατάσταση: 0-2-0. Ο ρυθμός μετάβασης από την κατάσταση 0-2-1 στην 0-2-0 είναι ίσος με τον ρυθμό βλάβης της μηχανής M_1.
- Να παράγει η μηχανή M_1 ένα κομμάτι και να προκύψει η κατάσταση: 0-1-1. Ο ρυθμός μετάβασης από την κατάσταση 0-2-1 στην 0-1-1 είναι ίσος με τον ρυθμό παραγωγής της μηχανής M_1.



Σχήμα 2: Γραμμή παραγωγής με 2 μηχανές και 1 αποθήκη

1.2 Σύνοψη Μεθόδων Ανάλυσης Γραμμών Παραγωγής

Οι γραμμές παραγωγής μπορούν να αναλυθούν είτε με τη χρήση προσομοίωσης είτε με τη χρήση των μοντέλων Markov.

Αν οι σχέσεις που περιγράφουν την εξέλιξη του συστήματος είναι απλές τότε είναι δυνατή η εύρεση λύσεων κλειστής μορφής, οπότε λέμε ότι το μοντέλο επιλύεται **αναλυτικά**. Ωστόσο τα περισσότερα συστήματα έχουν διάνυσμα κατάστασης μεγάλων διαστάσεων και περιγράφονται από πολύπλοκα μοντέλα των οποίων η αναλυτική επίλυση είναι αδύνατη. Για τη μελέτη τους εφαρμόζονται οι λεγόμενες **αριθμητικές** μέθοδοι. Τέτοιες είναι η αριθμητική ανάλυση και η προσομοίωση. Η προσομοίωση συνίσταται στην ανάπτυξη ενός μοντέλου του υπό εξέταση συστήματος με τη μορφή προγράμματος σε υπολογιστή και στην εκτέλεση ενός (ή περισσοτέρων) πειράματος το οποίο καταγράφει την κατάσταση του συστήματος σε διαδοχικές χρονικές στιγμές αποτυπώνοντας ένα πιθανό σενάριο εξέλιξης του συστήματος στο χρόνο.

Η προσομοίωση βρίσκει εφαρμογές:
• στην ανάλυση και σχεδίαση συστημάτων παραγωγής (βιομηχανία)
• στον έλεγχο αποθεμάτων (βιομηχανία, εμπορικές επιχειρήσεις)

• στη μελέτη κυκλοφοριακών συστημάτων (οδικό δίκτυο, αεροδρόμια)
• στη μελέτη συστημάτων εξυπηρετήσεως πελατών (τράπεζες, νοσοκομεία, τηλεπικοινωνίες)
• στην αξιολόγηση αποφάσεων υπό αβεβαιότητα (χρηματιστήριο, επενδύσεις, marketing).

Με την προσομοίωση μπορεί κανείς να αξιολογήσει την αποτελεσματικότητα ή απόδοση ενός συστήματος πριν αυτό κατασκευασθεί με σκοπό τη βέλτιστη σχεδίασή του.

Προσομοίωση (simulation) είναι η μίμηση της λειτουργίας συστημάτων ή της εξέλιξης διαδικασιών μέσα στο χρόνο με τη βοήθεια υπολογιστή.

Διαδικασία ή **σύστημα** ονομάζεται ένα σύνολο στοιχείων τα οποία εξελίσσονται και αλληλεπιδρούν σύμφωνα με κάποιους κανόνες.

Οι κανόνες αυτοί εκφράζονται με μαθηματικές ή λογικές σχέσεις, και αποτελούν το **μοντέλο** του συστήματος.

Κατάσταση είναι το σύνολο των μεταβλητών οι οποίες δίνουν την απαραίτητη πληροφορία για την περιγραφή του συστήματος. ([2])

Για την εκτίμηση των πιθανοτήτων κάθε κατάστασης, εκτός από την προσομοίωση, εφαρμόζονται και μοντέλα αλυσίδων Markov. Τα μοντέλα αυτά απαιτούν την επίλυση συστημάτων γραμμικών εξισώσεων, οι οποίες περιγράφουν τις πιθανότητες μεταβάσεων από κάθε κατάσταση προς άλλες. Γενικά, οι επαναληπτικές μέθοδοι είναι οι πιο συχνά χρησιμοποιούμενες μέθοδοι για την απόκτηση του διανύσματος πιθανοτήτων μόνιμης κατάστασης από έναν πίνακα στοχαστικών μεταβάσεων. Υπάρχουν διάφοροι λόγοι για αυτή την επιλογή. Κατ' αρχήν, μια εξέταση των επαναληπτικών μεθόδων που χρησιμοποιούνται συνήθως δείχνει ότι η μόνη πράξη στην οποία εμπλέκονται οι πινάκες είναι πολλαπλασιασμοί μεταξύ ενός ή περισσότερων διανυσμάτων. Αυτές οι πράξεις δεν αλλάζουν τη μορφή του πίνακα. Έτσι, μειώνεται ο όγκος της μνήμης που απαιτείται για να αποθηκεύσει τον πίνακα και μάλιστα προσαρμόζεται στον πολλαπλασιασμό του πίνακα. Επειδή οι πίνακες που εμπλέκονται είναι συνήθως μεγάλοι και αραιοί, η οικονομία στη μνήμη του υπολογιστή που γίνεται με τέτοιες μεθόδους μπορεί να είναι σημαντική. Με τις ευθείες μεθόδους επίλυσης, η διαγραφή ενός μη μηδενικού στοιχείου του πίνακα κατά την φάση της μείωσης συχνά έχει σαν αποτέλεσμα τη δημιουργία πολλών μη μηδενικών στοιχείων σε θέσεις που πριν είχαν 0. Αυτό κάνει την οργάνωση του μικρού χώρου αποθήκευσης πιο δύσκολη, αφού η πρόβλεψη πρέπει να γίνει για τη διαγραφή και την εισαγωγή στοιχείων και επιπλέον, ο όγκος αυτού του «γεμίσματος» μπορεί να είναι τόσο εκτεταμένος που η διαθέσιμη μνήμη γρήγορα εξαντλείται.

Οι επαναληπτικές μέθοδοι έχουν άλλα πλεονεκτήματα. Η χρήση τους μπορεί να γίνει από καλές αρχικές εκτιμήσεις για το διάνυσμα που θα είναι και η λύση. Αυτό είναι χρήσιμο όταν μια σειρά σχετικών πειραμάτων διεξάγεται. Σε αυτές τις συνθήκες, οι παράμετροι ενός πειράματος διαφέρουν ελάχιστα από τις παραμέτρους του προηγούμενου, ενώ κάποιες παραμένουν ίδιες. Συνεπώς, αναμένεται η λύση του νέου πειράματος να είναι κοντά στη λύση του προηγούμενου, οπότε μας βοηθάει να χρησιμοποιήσουμε το προηγούμενο αποτέλεσμα ως τη νέα αρχική εκτίμηση της

λύσης. Εάν πράγματι υπάρχει μικρή διαφοροποίηση, θα πρέπει να αναμένουμε να υπολογίσουμε το νέο αποτέλεσμα σε σχετικά λίγες επαναλήψεις.

Μια επαναληπτική μέθοδος μπορεί να σταματήσει όταν ικανοποιηθεί ένα προαποφασισμένο κριτήριο, το οποίο μπορεί να είναι σχετικά χαλαρό. Για παράδειγμα δεν έχει νόημα να υπολογιστεί σωστά και με μεγάλη ακρίβεια η λύση ενός μαθηματικού μοντέλου όταν το ίδιο το μοντέλο περιέχει σφάλματα της τάξης του 5-10%. Μια ευθεία μέθοδος επίλυσης είναι υποχρεωμένη να συνεχίσει μέχρι να βρει την «ακριβή» τελική τιμή. Τέλος, με τις επαναληπτικές μεθόδους, ο πίνακας δεν αλλάζει ποτέ. ([3])

Για αυτούς τους λόγους, οι επαναληπτικές μέθοδοι προτιμώνται από τις ευθείες μεθόδους. Παρ' όλα αυτά, οι επαναληπτικές μέθοδοι έχουν κυρίως ένα μειονέκτημα: χρειάζονται συνήθως πιο πολύ χρόνο για να καταλήξουν στην επιθυμητή λύση. Πιο ανεπτυγμένες επαναληπτικές μέθοδοι (όπως αυτές που αναπτύσσονται στην παρούσα διπλωματική εργασία) είναι πιο γρήγορες από τις ευθείες μεθόδους.

Συνοψίζοντας, η προσομοίωση είναι ακριβής μέθοδος ανάλυσης γραμμών παραγωγής αλλά είναι πολύ χρονοβόρα, κάτι το οποίο πιθανόν να δημιουργεί προβλήματα σε μεγάλες γραμμές παραγωγής. Τα μοντέλα Markov είναι μοντέλα απεικόνισης καταστάσεων ενός συστήματος σε δεδομένες χρονικές στιγμές (ή περιόδους). Διακρίνονται σε ακριβή και προσεγγιστικά. Τα ακριβή μοντέλα Markov διαχειρίζονται πίνακες μεγάλων διαστάσεων και παρουσιάζουν δυσκολία στην εύρεση των πινάκων μετάβασης από τη μια κατάσταση στην άλλη. Από την άλλη, τα προσεγγιστικά μοντέλα Markov διαχειρίζονται πίνακες μικρών διαστάσεων, αλλά δεν είναι πάντα ακριβή.

2 ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

2.1 Μηχανές και Αποθήκες

Η γραμμή παραγωγής αποτελείται από k μηχανές και k-1 αποθήκες. Κάθε μηχανή έχει έναν ρυθμό παραγωγής μ_i, έναν ρυθμό βλάβης p_i και έναν ρυθμό επισκευής r_i. Αντίστοιχα, ο μέσος χρόνος παραγωγής της μηχανής M_i είναι $1/\mu_i$, ο μέσος χρόνος ζωής μέχρι τη βλάβη είναι $1/p_i$ και ο μέσος χρόνος επισκευής είναι $1/r_i$. Οι χρόνοι είναι εκθετικά κατανεμημένοι. Προκειμένου να βρούμε τις εφικτές μεταβάσεις από μια κατάσταση της γραμμής παραγωγής σε μια άλλη, πρέπει να πάρουμε υπόψιν μας τα εξής για τις μηχανές:

- Μια μηχανή που είναι αποστερημένη ή μπλοκαρισμένη δεν μπορεί να χαλάσει.
- Μια μηχανή που λειτουργεί (δηλαδή βρίσκεται στην κατάσταση 1) μπορεί είτε να χαλάσει είτε να παράξει ένα κομμάτι. Αν χαλάσει, η κατάσταση της συγκεκριμένης μηχανής γίνεται 0, ενώ αν παράξει ένα κομμάτι, το απόθεμα της προηγούμενης αποθήκης μειώνεται κατά 1 και το απόθεμα της επόμενης αποθήκης αυξάνεται κατά 1.
- Μια μηχανή που είναι χαλασμένη (δηλαδή βρίσκεται στην κατάσταση 0) μπορεί να επισκευαστεί και η κατάστασή της να γίνει 1.

Για τις αποθήκες ισχύουν τα εξής:

- Έχουν πεπερασμένη χωρητικότητα.
- Ο χρόνος για να διασχίσει ένα κομμάτι μια άδεια αποθήκη είναι αμελητέος.

Συνολικά, για την λειτουργία της γραμμής, γίνονται οι παρακάτω υποθέσεις:

1. Κάθε μηχανή είναι **λειτουργική** (1) ή **υπό επισκευή** (0).
2. Υπάρχει μια **άπειρη πηγή στην αρχή** και **μια άπειρη καταβόθρα στο τέλος της γραμμής.** Έτσι η πρώτη μηχανή ποτέ δεν «πεινάει» και η τελευταία ποτέ δεν αποκλείεται.
3. Οι χρόνοι παραγωγής των μηχανών είναι **τυχαίες μεταβλητές με γνωστή κατανομή.**
4. Οι μηχανές χαλούν μόνο μετά την παραγωγή ενός κομματιού.
5. Μια **πεινασμένη** ή **αποκλεισμένη μηχανή δεν χαλάει.**
6. Ο **χρόνος μεταφοράς** ενός κομματιού από τη μια μηχανή στην άλλη **είναι αμελητέος** ή περιλαμβάνεται στο χρόνο επεξεργασίας.
7. Ο χρόνος που μια μηχανή χαλάει είναι τυχαία μεταβλητή με γνωστή κατανομή. Η κατανομή θεωρείται ίδια για όλες τις μηχανές.
8. Ο χρόνος επισκευής για κάθε μηχανή είναι τυχαία μεταβλητή με γνωστή κατανομή που είναι ίδια για όλες τις μηχανές.

Οι κατανομές στα (3), (7) και (8) θα ληφθούν εκθετικές. Αυτό επιτρέπει την ανάλυση του συστήματος με αλυσίδες Markov. ([3])

2.2 Καταστάσεις

Η κατάσταση του συστήματος ορίζεται σαν μια ομάδα από αριθμούς που δείχνουν τις λειτουργικές καταστάσεις των μηχανών και τον αριθμό των κομματιών σε κάθε αποθήκη, όπως περιγράφεται παρακάτω.

Για κάθε μηχανή σε μια γραμμή παραγωγής k μηχανών, η μεταβλητή ορίζεται ως εξής:

$\alpha_i = 0$, αν η μηχανή i είναι χαλασμένη
$\alpha_i = 1$, αν η μηχανή i είναι λειτουργική

Ο όρος «λειτουργική» ορίζει ότι η μηχανή είναι ικανή να παράγει ένα κομμάτι ανεξάρτητα από το αν επεξεργάζεται πραγματικά ένα κομμάτι. Η μεταβλητή αυτή εξηγεί τις περιπτώσεις που η μηχανή είναι σε καλές λειτουργικές συνθήκες αλλά δεν επεξεργάζεται κομμάτια γιατί είναι αποστερημένη ή αποκλεισμένη.

Η μεταβλητή η_i ορίζεται ως ο αριθμός των κομματιών που περιέχει η αποθήκη i. Κάθε αποθήκη έχει χωρητικότητα:

$0 \le \eta_i \le C_i+2$, όπου C_i η χωρητικότητα της αποθήκης και 2 κομμάτια που πιθανόν να επεξεργάζονται η προηγούμενη και η επόμενη μηχανή

Επομένως, κάθε κατάσταση της γραμμής παραγωγής συμβολίζεται ως εξής:

$x = (\alpha_k, \eta_{k-1}, \alpha_{k-1}, \dots, \alpha_2, \eta_1, \alpha_1)$

Για γραμμές παραγωγής με 2 μηχανές M_2, M_1 χωρίς βλάβες και μια ενδιάμεση αποθήκη χωρητικότητας $K=C_1+2$ συμβολίζονται ως αναμονητικό σύστημα M|M|1|K. Ο μέσος ρυθμός παραγωγής της M_2 είναι λ και της M_1 είναι μ. Η κατάσταση του συστήματος είναι μόνο μια μεταβλητή $\eta=\eta_1$ για την οποία ισχύουν ([4]):

$$P_n = P_0 \left(\frac{\lambda}{\mu}\right)^n \quad \text{αν } n \le K$$

$$P_n = 0 \quad \text{αν } n>K$$

Επίσης, από την εξίσωση $P_0 + \dots + P_K = 1$, που δίνει την ολική πιθανότητα, προκύπτει:

$$P_0 = \frac{1-\frac{\lambda}{\mu}}{1-\left(\frac{\lambda}{\mu}\right)^{K+1}}$$

Οπότε:

$$P_n = \frac{1-\frac{\lambda}{\mu}}{1-\left(\frac{\lambda}{\mu}\right)^{K+1}}\left(\frac{\lambda}{\mu}\right)^n \quad \text{αν} \quad 0 \le n \le K$$

$$P_n = 0 \quad \text{αν} \quad n>K \text{ η } n<0$$

Μεγαλύτερες γραμμές παραγωγής με μηχανές που υφίστανται βλάβες μοντελοποιούνται σαν αλυσίδες Markov. Θεωρείται ότι το σύστημα είναι στη μόνιμη κατάσταση, δηλαδή ότι όλα τα προβλήματα των μεταβάσεων εκκίνησης έχουν ξεπερασθεί και ότι το σύστημα μπορεί να παρασταθεί από μια σταθερή κατανομή πιθανότητας. Ένα στοχαστικό σύστημα δεν είναι αμετάβλητο. Η υπόθεση μόνιμης κατάστασης δεν συνεπάγεται ότι το σύστημα δεν ταλαντώνεται. Αυτό που συνεπάγεται είναι ότι έχει περάσει μια αρκετά μεγάλη περίοδος του χρόνου από το ξεκίνημα, ούτως ώστε η γνώση της αρχικής κατάστασης του συστήματος δεν δίνει κάποια πληροφορία για την παρούσα κατάσταση του συστήματος. Για να βρούμε την οριακή κατανομή πιθανότητας σταθερής μόνιμης κατάστασης από μια αλυσίδα Markov n καταστάσεων είναι αναγκαίο να λύσουμε μια ομάδα από γραμμικές εξισώσεις μετάβασης με n αγνώστους. ([4]) Για γραμμές παραγωγής με k μηχανές και k-1 αποθήκες, όπου κάθε μηχανή έχει δυο καταστάσεις (0 και 1) και κάθε αποθήκη χωρητικότητας C_i εχει C_i+3 καταστάσεις (0,…,C_i+2), το n είναι ίσο με:

$$n = 2^k (C_1+3)(C_2+3)...(C_{k-1}+3) \Rightarrow n = 2^k \prod_{i=1}^{k-1}(C_i+3)$$

Οι εξισώσεις των πιθανοτήτων είναι γνωστές ως εξισώσεις Chapman-Kolmogorov και περιγράφονται στην επόμενη παράγραφο. Από την παραπάνω σχέση βλέπουμε ότι ο αριθμός των εξισώσεων αυξάνεται σε συνάρτηση με τις χωρητικότητες των αποθηκών. Επειδή λοιπόν το n είναι πολύ μεγάλο, απαιτείται ένας αποδοτικός τρόπος υπολογισμού των μέτρων απόδοσης της γραμμής παραγωγής. Για παράδειγμα, για μία γραμμή με k= 5 μηχανές και C_i=5 έχουμε 131072 εξισώσεις!

2.3 Αλυσίδες Markov και Εξισώσεις Chapman- Kolmogorov

Οι αλυσίδες Markov συνεχούς χρόνου περιγράφουν τυχαία φαινόμενα που εξελίσσονται στο χρόνο (στοχαστικές διαδικασίες) ως εξής:

1) Κάθε στιγμή το φαινόμενο περιγράφεται από μία μεταβλητή (κατάσταση) η οποία λαμβάνει τιμές από ένα αριθμήσιμο σύνολο, πχ. {1,2,3, ….} ή {y, y′, y″, …} . Για το προηγούμενο παράδειγμα των δύο μηχανών με ενδιάμεση αποθήκη διαμορφώνεται ένα σύνολο 12 καταστάσεων {000, 001, …, 121} ή {0, 1, …, 11}.
2) Η πιθανότητα ώστε σε χρονικό διάστημα δ η αλυσίδα να μεταβεί από την κατάσταση y′ σε μία άλλη y′ ισούται με

   δ R(y′, y) + (όροι ανωτέρας τάξεως του δ).

   Το R(y′, y) είναι ο μέσος στιγμιαίος ρυθμός μετάβασης από την y′ στην y. Για, παράδειγμα, αν ο μέσος ρυθμός παραγωγής της M_2 είναι 10 και ο ρυθμός βλάβης της είναι 1, τότε από την κατάσταση 101 του παραδείγματος μεταβαίνουμε:

   είτε στην 111 με ρυθμό R(101, 111) = 10

   είτε στην κατάσταση 001 με ρυθμό R(101, 001) = 1
3) Εφ’ όσον υπάρχει μόνιμη κατάσταση (καθώς ο χρόνος τείνει στο άπειρο οι πιθανότητες καταστάσεων συγκλίνουν), ισχύουν οι αλγεβρικές εξισώσεις Chapman-Kolmogorov:

   P(y′) × (συνολικός ρυθμός μετάβασης από y′ σε άλλες καταστάσεις) =

$$= \sum_{\text{άλλες } y \neq y'} P(y) * R(y, y')$$

Οι εξισώσεις Chapman-Kolmogorov σχηματίζουν ένα γραμμικό σύστημα που έχει τη μορφή:

$$P = AP$$

όπου P είναι το διάνυσμα πιθανοτήτων [P(y) P(y′) P(y″)…] διατεταγμένο σε στήλη ενώ A είναι ο τετραγωνικός πίνακας με στοιχεία τα πηλίκα

R(y, y′)/ (συνολικός ρυθμός μετάβασης από y′ σε άλλες καταστάσεις).

ή

$$A(\pi\rho o\varsigma\_ y', \alpha\pi o\_ y) = A(y', y) = \frac{R(y, y')}{R(y')}$$

όπου

$$R(y') = \sum_{\text{άλλες } y \neq y'} R(y', y) = \text{ (συνολικός ρυθμός μετάβασης από } y' \text{ σε άλλες καταστάσεις)}$$

Για μικρό πλήθος καταστάσεων το γραμμικό σύστημα εξισώσεων επιλύεται εύκολα. Για μεγάλες διαστάσεις εφαρμόζονται επαναληπτικές μέθοδοι όπως η Jacobi, Gauss Seidel, και παραλλαγές τους που θα περιγράψουμε αργότερα.

Οι πιθανότητες μόνιμης κατάστασης μπορούν να χρησιμοποιηθούν στον υπολογισμό δεικτών απόδοσης της στοχαστικής διαδικασίας που εξετάζεται. Θα δούμε τέτοιους δείκτες απόδοσης αργότερα για γραμμές παραγωγής.

2.4 Επίλυση της εξίσωσης Chapman- Kolmogorov

2.4.1 Μέθοδος Jacobi

Για την επίλυση της εξίσωσης Chapman- Kolmogorov που αφορά μια γραμμή παραγωγής k μηχανών και k-1 αποθηκών, με δεδομένους ρυθμούς παραγωγής, βλάβης και επισκευής κάθε μηχανής καθώς και δεδομένης χωρητικότητας κάθε αποθήκης, μπορούμε να χρησιμοποιήσουμε την μέθοδο Jacobi. Αρχικοποιούμε τις πιθανότητες Q(y) να είναι ίσες με:

$$Q(y) = \frac{1}{\Pi\lambda\eta\theta o\varsigma\_\varepsilon\phi\iota\kappa\tau\omega\nu\_\kappa\alpha\tau\alpha\sigma\tau\alpha\sigma\varepsilon\omega\nu}$$

για κάθε εφικτή κατάσταση y (Θα δούμε αργότερα ότι οι καταστάσεις διακρίνονται σε εφικτές και μη εφικτές).

Ξεκινώντας λοιπόν από την εξίσωση P=AP, το γραμμικό σύστημα των εξισώσεων είναι:

$$P(y') = \sum_{\forall y \varepsilon \phi \iota \kappa \tau \eta} A(y', y) * P(y)$$

Η μέθοδος Jacobi είναι επαναληπτική και χρησιμοποιεί τιμές P(y) στο δεξί μέλος των εξισώσεων που έχουν προκύψει από προηγούμενες επαναλήψεις. Ο επαναληπτικός τύπος είναι:

$$P(y') = \sum_{\forall y \varepsilon \phi \iota \kappa \tau \eta} \frac{R(y, y')}{R(y')} * Q(y) = \sum_{\forall y \varepsilon \phi \iota \kappa \tau \eta} A(y', y) * Q(y)$$

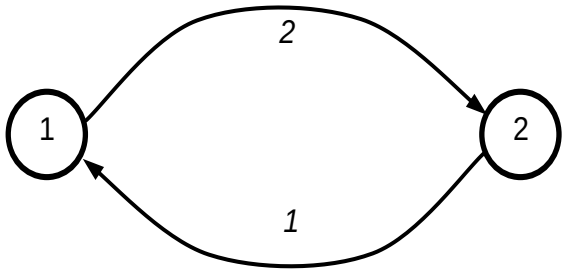
όπου Q(y) είναι η πιθανότητα από την προηγούμενη επανάληψη.

Κάνοντας επαναλήψεις για διάφορα πειράματα, παρατηρούμε ότι συχνά η μέθοδος δεν συγκλίνει. Γενικά, οι επαναληπτικές μέθοδοι δεν συγκλίνουν πάντα όπως δείχνουμε στη συνέχεια.

2.4.2 Μέθοδος Successive Over-Relaxation (SOR)

Θεωρούμε μια αλυσίδα Markov με δυο καταστάσεις 1 και 2. Από την 1 πάμε στη 2 με ρυθμό 2 και από την 2 πάμε στην 1 με ρυθμό 1. **Η αλυσίδα αυτή είναι περιοδική**, γιατί καθώς ο χρόνος περνά, η αλυσίδα βρίσκεται στις καταστάσεις 1, 2

και μετά πάλι στις 1,2 (περίοδος = 2). Θα δούμε ότι η μέθοδος Jacobi δεν συγκλίνει σε τέτοιες περιπτώσεις.



Σχήμα 3: Περιοδική αλυσίδα Markov με 2 καταστάσεις

Ξεκινάμε με αρχικές τιμές $P_1 = P_2 = \frac{1}{2} = 0.5$. Οι εξισώσεις Chapman-Kolmogorov είναι:

$$2P_1 = 1P_2 \Rightarrow P_1 = \frac{1}{2}P_2 = 0.25$$
$$1P_2 = 2P_1 \Rightarrow P_2 = 2P_1 = 1$$

Επιπλέον, πρέπει να χρησιμοποιήσουμε και την εξίσωση κανονικοποίησης $P_1 + P_2 = 1$, κι έτσι θα έχουμε:

$$P_1 = \frac{0.25}{1.25} = \frac{1}{5} = 0.2$$

$$P_2 = \frac{1}{1.25} = \frac{4}{5} = 0.8$$

Αυτές οι τιμές χρησιμοποιούνται στην επόμενη επανάληψη:

$$P_1 = \frac{1}{2} * P_2 = \frac{1}{2} * \frac{4}{5} = \frac{2}{5} = 0.4$$

$$P_2 = 2 * P_1 = 2 * \frac{1}{5} = \frac{2}{5} = 0.4$$

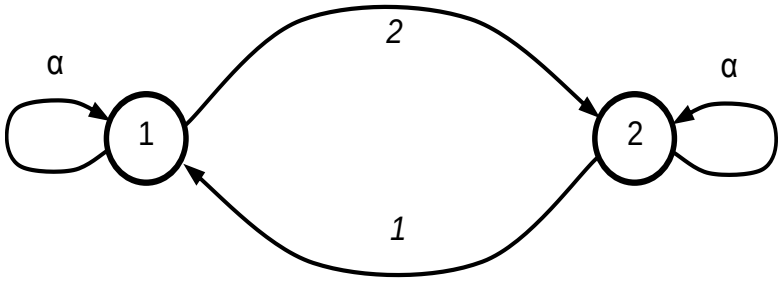
Με την κανονικοποίηση, γίνονται:

$$P_1 = \frac{0.4}{0.8} = \frac{1}{2} = 0.5$$

$$P_2 = \frac{0.4}{0.8} = \frac{1}{2} = 0.5$$

Καταλήγουμε δηλαδή στις πιθανότητες που είχαμε αρχικά και αν συνεχίσουμε παρατηρούμε ότι οι επαναλήψεις δίνουν περιοδικές λύσεις και η μέθοδος δεν συγκλίνει.

Πρέπει λοιπόν να έχουμε μια αλυσίδα Markov που να είναι απεριοδική και αυτό το επιτυγχάνουμε αν θεωρήσουμε μια επιπλέον μετάβαση στην ίδια κατάσταση με έναν ρυθμό έστω α=0.1.



Σχήμα 4: Απεριοδική αλυσίδα Markov με 2 καταστάσεις

Οι νέες εξισώσεις Chapman- Kolmogorov είναι:

$(2+\alpha)P_1 = 1P_2 + \alpha P_1$

$(1+\alpha)P_2 = 2P_1 + \alpha P_2$

Σε αυτές τις εξισώσεις, αν απλοποιήσουμε όλους τους όρους που έχουν α, τότε προκύπτουν οι αρχικές. Άρα, έχουμε πρακτικά το ίδιο σύστημα εξισώσεων, όμως τώρα η μέθοδος συγκλίνει.

Για αρχικές πιθανότητες $P_1 = P_2 = \frac{1}{2} = 0.5$,οι εξισώσεις γίνονται:

$$P_1 = \frac{1}{2+\alpha} * (1 * P_2 + \alpha * P_1) \Rightarrow P_1 = \frac{1}{2+0.1} * \left(\frac{1}{2} + 0.1 * \frac{1}{2}\right) = 0.275$$

$$P_2 = \frac{1}{(1+\alpha)} * (2 * P_1 + \alpha * P_2) \Rightarrow P_2 = \frac{1}{(1+0.1)} * (2 * 0.5 + 0.1 * 0.5) = 0.95455$$

Με την κανονικοποίηση, από την εξίσωση $P_1 + P_2 = 1$, γίνονται:

$$P_1 = \frac{0.275}{1.22955} = 0.22366$$

$$P_2 = \frac{0.95455}{1.22955} = 0.77634$$

Ομοίως, συνεχίζοντας τις επαναλήψεις και κάνοντας κανονικοποίηση, βρίσκουμε τα εξής:

Πίνακας 1: Πιθανότητες P_1 και P_2 σε κάθε επανάληψη

P_1	P_2
0.5	0.5
0.22366	0.77634
0.45558	0.54442
0.25152	0.74848
0.42406	0.57594
0.27299	0.72701
0.40140	0.59860
0.28939	0.71061
0.38497	0.61503
0.30183	0.69817
0.37297	0.62703
0.31122	0.68878
0.36417	0.63583
0.31828	0.68172
0.35769	0.64231
0.32358	0.67642
0.35290	0.64710
0.32755	0.67245
0.34936	0.65064
0.33051	0.66949
0.34674	0.65326
0.33272	0.66728

Άρα, τελικά:

$P_1=0.33272 \approx \frac{1}{3}$

$P_2=0.66728 \approx \frac{2}{3}$

Η μέθοδος αυτή ονομάζεται μέθοδος Διαδοχικής Υπερχαλάρωσης (SOR) και, εκτός από την Jacobi, εφαρμόζεται και σε άλλες επαναληπτικές μεθόδους.

Για την εφαρμογή της μεθόδου στο επαναληπτικό σχήμα Jacobi ξεκινάμε από την αρχική εξίσωση Chapman-Kolmogorov:

$$R(y') * P(y') = \sum_{\forall \text{γεφικτη}} R(y, y') * P(y)$$

Προσθέτουμε σε κάθε μέλος της εξίσωσης το γινόμενο της πιθανότητας που έχει η κατάσταση y’ με την παράμετρο υπερχαλάρωσης α:

$$(R(y')+\alpha)*P(y') = \sum_{\forall \text{γεφικτη}} R(y,y')*P(y) + P(y')*\alpha$$

Στο δεξί μέλος, αντικαθιστούμε τις άγνωστες πιθανότητες P(y) με τις εκτιμήσεις τους Q(y) που βρέθηκαν στην προηγούμενη επανάληψη και προκύπτει:

$$(R(y')+\alpha)*P(y') = \sum_{\forall \text{γεφικτη}} R(y,y')*Q(y) + Q(y')*\alpha$$

Στο όριο, έχουμε P=Q, οπότε η προηγούμενη εξίσωση είναι ίδια. Τώρα πλέον ο αλγόριθμος συγκλίνει.

2.4.3 Μέθοδος Gauss- Seidel

Η μέθοδος Jacobi απαιτεί να κρατάμε όλες τις συνιστώσες του P_t μέχρις ότου ολοκληρωθεί ο υπολογισμός του P_{t+1}. Μια πολύ πιο φυσική ιδέα, που απαιτεί το μισό της προηγούμενης μνήμης, είναι να αρχίσουμε να χρησιμοποιούμε κάθε συνιστώσα του νέου διανύσματος P_{t+1}, αμέσως μετά τον υπολογισμό της. Έτσι, το P_{t+1} παίρνει τη θέση του P_t, συνιστώσα προς συνιστώσα, και το Q(y) μπορεί να καταστραφεί ταυτόχρονα με τη δημιουργία του P_{t+1}. Συνεπώς, σε κάθε εξίσωση χρησιμοποιούνται οι πιο πρόσφατες τιμές που έχουν επιτευχθεί μέχρι στιγμής από τις προηγούμενες εξισώσεις (όπου προηγούμενη της πρώτης εξίσωσης είναι, κυκλικά, η τελευταία).

Σε αυτή την περίπτωση μας αρκεί ένα διάνυσμα που θα αναπαριστά το P_{t+1}. Η μέθοδος Gauss- Seidel όμως μπορεί να γίνει πολύ καλύτερη με την εισαγωγή ενός συντελεστή υπερχαλάρωσης, όπως και στην περίπτωση της μεθόδου Jacobi. Η σύγκλιση είναι ταχύτερη αν προχωρήσουμε πέρα από τη διόρθωση Gauss- Seidel P_{t+1}- P_t. Μιλώντας μη αυστηρά, η συνηθισμένη μέθοδος συγκλίνει μονοτόνως και οι προσεγγίσεις P_t παραμένουν από την ίδια μεριά της πραγματικής λύσης. Έτσι, εισάγοντας έναν συντελεστή υπερχαλάρωσης α, πάμε κοντύτερα στη λύση. Μετά από δοκιμές, θέτουμε α=0.1, αφού πρόκειται για μια τιμή με την οποία ο αλγόριθμος συγκλίνει. Θα μπορούσαμε επίσης να χρησιμοποιήσουμε α=0.2 ή α=0.3, τιμές για τις οποίες επίσης ο αλγόριθμος συγκλίνει. Ο πίνακας μεταβάσεων δεν είναι στοχαστικός. Παρ’ όλα αυτά απλές μεταβάσεις τον αποδίδουν ως στοχαστικό.

Για τη συγκεκριμένη μέθοδο, αρκεί ένα μόνο διάνυσμα για να εκφράσει το P_{t+1}, αφού για την παλιά τιμή P_t αρκεί μια μεταβλητή, η οποία κάθε φορά ενημερώνεται με την τελευταία τιμή του P_t.

Τώρα, ξεκινώντας από την εξίσωση P=AP, θα πρέπει να αποθηκεύονται όλες οι εφικτές μεταβάσεις y->y' και οι αντίστοιχοι ρυθμοί για κάθε τελική κατάσταση y'. Όπως και για την μέθοδο Jacobi, αρχικά έχουμε την εξίσωση:

$$R(y') * P(y') = \sum_{\substack{\forall \text{γεφικτη} \\ \text{απο} \\ \text{τρεχουσα} \\ \text{επαναληψη}}} R(y, y') * P(y) + \sum_{\substack{\forall \text{γεφικτη} \\ \text{απο} \\ \text{προηγουμενη} \\ \text{επαναληψη}}} R(y, y') * Q(y)$$

όπου το πρώτο άθροισμα έχει τις πιθανότητες που έχουν ήδη εκτιμηθεί στην τρέχουσα επανάληψη και το δεύτερο έχει εκτιμήσεις από την προηγούμενη επανάληψη.

Εφαρμόζοντας επιπλέον και την μέθοδο υπερχαλάρωσης, το επαναληπτικό σχήμα Gauss-Seidel διαμορφώνεται ως εξής:

$$\left(R(y') + \alpha\right) * P(y') = \sum_{\substack{\forall \text{γεφικτη} \\ \text{απο} \\ \text{τρεχουσα} \\ \text{επαναληψη}}} R(y, y') * Q(y) + \sum_{\substack{\forall \text{γεφικτη} \\ \text{απο} \\ \text{προηγουμενη} \\ \text{επαναληψη}}} R(y, y') * Q(y) + Q(y') * \alpha$$

Τώρα πλέον ο αλγόριθμος συγκλίνει.

3 ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ

Η λογική του αλγορίθμου είναι η εξής:

1. Βρίσκει τις εφικτές καταστάσεις της γραμμής παραγωγής (μέγεθος διανύσματος P(y)).
2. Βρίσκει τις εφικτές μεταβάσεις της γραμμής παραγωγής (πίνακας μεταβάσεων R(y,y')).
3. Λύνει τις εξισώσεις Chapman- Kolmogorov.
4. Υπολογίζει την παραγωγικότητα και το μέσο απόθεμα της γραμμής παραγωγής.

Ουσιαστικά, έχουμε κάνει 2 αλγόριθμους, διότι οι εξισώσεις Chapman-Kolmogorov λύνονται με δύο επαναληπτικές μεθόδους: την μέθοδο Jacobi και την μέθοδο Gauss- Seidel.

Αρχικά, ο χρήστης του προγράμματος δίνει ως δεδομένα τον αριθμό των μηχανών k, την χωρητικότητα C κάθε αποθήκης και τους ρυθμούς παραγωγής μ, βλάβης p και επισκευής r κάθε μηχανής. Έτσι, αφού βρεθεί το σύνολο των πιθανών καταστάσεων, ελέγχεται ποιες από αυτές είναι εφικτές με βάση τις συνθήκες εφικτότητας που έχουμε περιγράψει.

3.1 Μεταβάσεις

Στη συνέχεια, για κάθε μία εφικτή κατάσταση εξετάζονται τα ακόλουθα γεγονότα:

1. Βλάβη της μηχανής M_i.
2. Επισκευή της μηχανής M_i.
3. Παραγωγή από τη μηχανή M_i.

3.1.1 Βλάβη της μηχανής M_i

Εξετάζουμε την περίπτωση που μια λειτουργική μηχανή παθαίνει βλάβη, οπότε η κατάσταση της μηχανής από 1 γίνεται 0. Για τη νέα κατάσταση που προκύπτει, δεδομένου ότι οι χωρητικότητες των αποθηκών και οι καταστάσεις των υπόλοιπων μηχανών δεν μεταβάλλονται, χρησιμοποιούμε τις συνθήκες εφικτότητας και βρίσκουμε αν η νέα κατάσταση (και κατ' επέκταση η μετάβαση από την αρχική στην τελική κατάσταση) είναι εφικτή. Αν είναι εφικτή, βάζουμε στην αντίστοιχη θέση του πίνακα που αποθηκεύει τους ρυθμούς μετάβασης τον ρυθμό βλάβης της συγκεκριμένης μηχανής.

3.1.2 Επισκευή της μηχανής M_i

Εξετάζουμε την περίπτωση που μια χαλασμένη μηχανή επισκευάζεται, οπότε η κατάσταση της μηχανής από 0 γίνεται 1. Για τη νέα κατάσταση που προκύπτει, δεδομένου ότι οι χωρητικότητες των αποθηκών και οι καταστάσεις των υπόλοιπων μηχανών δεν μεταβάλλονται, χρησιμοποιούμε τις συνθήκες εφικτότητας και βρίσκουμε αν η νέα κατάσταση (και κατ' επέκταση η μετάβαση από την αρχική στην τελική κατάσταση) είναι εφικτή. Αν είναι εφικτή, βάζουμε στην αντίστοιχη θέση του πίνακα που αποθηκεύει τους ρυθμούς μετάβασης τον ρυθμό επισκευής της συγκεκριμένης μηχανής.

3.1.3 Παραγωγή από τη μηχανή M_i

Εξετάζουμε την περίπτωση που μια λειτουργική μηχανή παράγει ένα κομμάτι, οπότε στην επόμενη αποθήκη B_{i-1} προστίθεται ένα κομμάτι και από την προηγούμενη αποθήκη B_i αφαιρείται ένα κομμάτι. Για τη νέα κατάσταση που προκύπτει, δεδομένου ότι οι χωρητικότητες των υπόλοιπων αποθηκών και οι καταστάσεις των μηχανών δεν μεταβάλλονται, χρησιμοποιούμε τις συνθήκες εφικτότητας και βρίσκουμε αν η νέα κατάσταση (και κατ' επέκταση η μετάβαση από την αρχική στην τελική κατάσταση) είναι εφικτή. Αν είναι εφικτή, βάζουμε στην αντίστοιχη θέση του πίνακα που αποθηκεύει τους ρυθμούς μετάβασης τον ρυθμό παραγωγής της συγκεκριμένης μηχανής.

3.2 Κωδικοποίηση- Αποκωδικοποίηση

Για να βρούμε όλες τις πιθανές καταστάσεις του συστήματος, χρειάζεται να κωδικοποιήσουμε και να αποκωδικοποιήσουμε τις καταστάσεις. Με άλλα λόγια, πρέπει να αντιστοιχίσουμε κάθε πιθανή κατάσταση της γραμμής παραγωγής με έναν αριθμό, ο οποίος στο εξής θα υποδηλώνει την συγκεκριμένη κατάσταση της γραμμής παραγωγής. Αυτό είναι απαραίτητο, γιατί ο υπολογιστής θα πρέπει να αποθηκεύει τις καταστάσεις ως $1^η$, $2^η$, $3^η$,κτλ, ώστε να αντιστοιχούν σε γραμμές και στήλες πινάκων και διανύσματα. Επειδή κάθε μηχανή έχει 2 καταστάσεις (0, 1) και κάθε αποθήκη έχει C+3 καταστάσεις (0,..., C+2), το συνολικό πλήθος των δυνατών συνδυασμών καταστάσεων είναι το γινόμενο του πλήθους καταστάσεων κάθε στοιχείου.

Το πλήθος των πιθανών καταστάσεων βρίσκεται από τον αναδρομικό αλγόριθμο:

$$w(1) = 1$$
$$w(2) = w(1) * m(1)$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$n = w(2*k) = w(2*k-1) * m(2*k-1)$$

όπου m(2i) = C_i+3 , αν πρόκειται για αποθήκη (2i=2,4,6,…) και m(2i+1)=2 ,αν πρόκειται για μηχανή (2i+1=1,3,5,…), δεδομένου ότι υπάρχουν 2*k-1 θέσεις (μηχανές και αποθήκες) στη γραμμή παραγωγής.

3.2.1 Κωδικοποίηση

Η μέγιστη τιμή που μπορεί να πάρει κάθε κατάσταση μηχανής είναι 1, ενώ η μέγιστη τιμή που μπορεί να πάρει κάθε κατάσταση αποθήκης είναι C_i+2. Έτσι, χρησιμοποιώντας τον αλγόριθμο της κωδικοποίησης, προκύπτουν όλες οι πιθανές καταστάσεις της γραμμής παραγωγής, κάθε μία από τις οποίες είναι ένας συνδυασμός (μηχανές- αποθήκες) και αντιστοιχεί σε έναν συγκεκριμένο αριθμό. Δηλαδή, το διάνυσμα x= (α_k, η_{k-1},α_{k-1}, … , α_2, η_1, α_1) κωδικοποιείται σε έναν ακέραιο αριθμό. Το διάνυσμα x αποθηκεύει τους αριθμούς που αντιστοιχούν στις καταστάσεις των μηχανών και των αποθηκών για κάθε μια κατάσταση, συνεπώς είναι αδύνατη η επεξεργασία τους στη συνέχεια του αλγορίθμου προκειμένου να εφαρμοστούν οι επαναληπτικές μέθοδοι επίλυσης. Ο αλγόριθμος της κωδικοποίησης έχει την εξής λογική:

Πρόγραμμα encode

Για i=1,..., k
  x(i)=0
ΤέλοςΓια
y=0
Τύπωσε y, και όλα τα (x(i), i=1,...,k) *// σχόλιο: αυτός είναι ο πρώτος συνδυασμός*

Για y=1,..., n
  ipointer=1
  Κάνε τα ακόλουθα, while x(ipointer)=m(ipointer)
    x(ipointer)=0
    ipointer=ipointer+1 *// σχόλιο: για κάθε x=m έχουμε «ένα το κρατούμενο» και θέτουμε x=0 (restart)*
  Τέλος while x(ipointer)=m(ipointer)
  x(ipointer)=x(ipointer)+1
  Τύπωσε y, και όλα τα (x(i), i=1,...,k) *// σχόλιο: αυτός είναι ο νέος συνδυασμός*
ΤέλοςΓια

Τέλος encode

Για παράδειγμα, για μία γραμμή παραγωγής με 2 μηχανές και 1 αποθήκη μηδενικής χωρητικότητας (μέγιστο πλήθος κομματιών μεταξύ των μηχανών = 2), μετά την κωδικοποίηση, βρίσκουμε:

M_2- B_1-M_1	y
000	0
001	1
010	2
011	3
020	4
021	5
100	6
101	7
110	8
111	9
120	10
121	11

Δηλαδή, το πλήθος των καταστάσεων των μηχανών M_2 και M_1 είναι m(1)=m(3)=2 και το πλήθος των καταστάσεων της αποθήκης είναι m(2)=3. Σύμφωνα με τον αλγόριθμο, τα βάρη υπολογίζονται ως εξής:

$$w(1)=1$$
$$w(2)= m(1) = 2$$
$$w(3)=w(2)*m(2) = 2*3=6$$

Έτσι, υπολογίζεται ο δείκτης που αντιστοιχεί σε κάθε κατάσταση, δηλαδή ξεκινάει από το συνδυασμό 0 0 0 (θέσεις 3, 2, 1) και τυπώνει 0*w(3)+0*w(2)+0*w(1) = 0 (δείκτης συνδυασμού = 0) μετά προσθέτει μία μονάδα και προκύπτει ο συνδυασμός 0 0 1 για τον οποίο ο δείκτης είναι 0*w(3)+0*w(2)+1*w(1) = 1 μετά άλλη μία μονάδα και προκύπτει ο συνδυασμός 0 1 0 για τον οποίο ο δείκτης είναι 0*w(3)+1*w(2)+0*w(1) = 2 ... κοκ μέχρι τον τελευταίο συνδυασμό που πρέπει να τον σχηματίζει αυτόματα. Εδώ ο τελευταίος συνδυασμός είναι 1 2 1 και έχει δείκτη 1*w(3)+2*w(2)+1*w(1) = 11.

Θα πρέπει να σημειωθεί ότι δεν είναι όλες οι καταστάσεις y εφικτές. Τις εξετάζουμε μία προς μία και απορρίπτουμε ως μη εφικτές τις καταστάσεις για τις οποίες ισχύουν τα ακόλουθα:

- Η μηχανή M_i είναι χαλασμένη (κατάσταση 0) και το απόθεμα της προηγούμενης αποθήκης B_i είναι 0 (άδεια αποθήκη).
- Η μηχανή M_i είναι χαλασμένη (κατάσταση 0) και το απόθεμα της επόμενης αποθήκης B_{i-1} είναι C_{i-1}+2 (γεμάτη αποθήκη).
- Η μηχανή M_i είναι χαλασμένη (κατάσταση 0), το απόθεμα της επόμενης αποθήκης B_{i-1} είναι C_{i-1}+1 και η επόμενη μηχανή M_{i-1} είναι μπλοκαρισμένη. Δηλαδή, η μηχανή M_i είναι μπλοκαρισμένη.
- Το απόθεμα της αποθήκης B_i είναι C_i+2 και το απόθεμα της αποθήκης B_{i-1} είναι C_{i-1}+2.

Έτσι, για παράδειγμα, η κατάσταση y=0 αντιστοιχεί στην 0-0-0, η οποία δεν είναι εφικτή, αφού η μηχανή M_1 είναι χαλασμένη και η αποθήκη B_1 είναι άδεια.

3.2.2 Αποκωδικοποίηση

Η αποκωδικοποίηση είναι το αντίστροφο της κωδικοποίησης, δηλαδή αποκωδικοποιούμε έναν αριθμό y για να δούμε σε ποια πιθανή κατάσταση του συστήματος αντιστοιχεί. Αυτό είναι απαραίτητο έτσι ώστε να γνωρίζουμε σε ποιες καταστάσεις αποθηκών και μηχανών (διάνυσμα x) αντιστοιχεί κάθε κατάσταση y. Έτσι, μπορούμε να διακρίνουμε τις εφικτές από τις μη εφικτές καταστάσεις και να φτιάξουμε τον πίνακα μεταβάσεων R(y,y'). Ο αλγόριθμος της αποκωδικοποίησης έχει την εξής λογική:

Πρόγραμμα decode

Για y=0,..., n
  yrem=y
// σχόλιο: Για παράδειγμα, αν y=235. Πρέπει να βρούμε ότι x(3)=2, x(2)=3 και x(1)=5. Αυτό γίνεται ως εξής:
  Για i=k,k-1,...,1
    x(i)=yrem/w(i) *//σχόλιο: Για παράδειγμα αν έχουμε y=235, διαιρούμε έναν ακέραιο yrem με το w και το αποτέλεσμα πρέπει να είναι ακέραιος, π. χ. 235/100=2*
    yrem=yrem-x(i)*w(i) *//σχόλιο: εδώ «κόβουμε» το x(3) από το yrem, π.χ. . 235-2*100=, και αυτό είναι το καινούριο yrem.*
Συνεχίζουμε ομοίως και παίρνουμε x(2)=3 και x(1)=5
ΤέλοςΓια
  Τύπωσε y, και όλα τα (x(i), i=1,...,k) *// σχόλιο: αυτός είναι ο νέος συνδυασμός*
ΤέλοςΓια

Τέλος decode

Για παράδειγμα, για μία γραμμή παραγωγής με 2 μηχανές και 1 αποθήκη μηδενικής χωρητικότητας (μέγιστο πλήθος κομματιών μεταξύ των μηχανών = 2), μετά την αποκωδικοποίηση, βρίσκουμε:

y	M_2-B_1-M_1
0	000
1	001
2	010
3	011
4	020
5	021
6	100
7	101
8	110
9	111
10	120
11	121

Δηλαδή, για κάθε έναν δείκτη αποκωδικοποιείται η κατάσταση. Έτσι, ο δείκτης 0 αντιστοιχεί στην μη εφικτή κατάσταση 0 0 0, ο δείκτης 1 στην 0 0 1 ... κοκ μέχρι τον τελευταίο δείκτη με αριθμό 11 που αντιστοιχεί στην κατάσταση 1 2 1.

3.3 Έλεγχος σύγκλισης

Μια καλή αρχική εκτίμηση μπορεί να επιταχύνει σημαντικά τους υπολογισμούς. Οι επαναλήψεις τερματίζονται όταν η τιμή που προκύπτει για κάθε πιθανότητα είναι πολύ κοντά στην προηγούμενη τιμή. Επειδή ο πίνακας μεταβάσεων δεν μεταβάλλεται στην επαναληπτική διαδικασία, οι επαναληπτικές μέθοδοι δεν υπόκεινται σε συσσώρευση λαθών εξαιτίας των στρογγυλοποιήσεων. Το κύριο μειονέκτημα των επαναληπτικών μεθόδων είναι ότι η σύγκλιση δεν είναι πάντα εγγυημένη.

Η σύγκλιση είναι ένα πολύ σημαντικό θέμα για τις επαναληπτικές μεθόδους. Μια προσέγγιση μπορεί να αναφέρεται στην επιλογή κατάλληλων τεχνικών για σύγκλιση αλλά δεν υπάρχουν γενικοί αλγόριθμοι για την επιλογή μιας τέτοιας τεχνικής. Επειδή η επιθυμητή λύση (διάνυσμα) δεν είναι γνωστή, μια εκτίμηση του λάθους πρέπει να χρησιμοποιηθεί για να προσδιορίσει τη σύγκλιση. Ένα ανεκτό επίπεδο πρέπει να οριστεί για να παρέχει ένα μέτρο του πόσο κοντά είναι η λύση της παρούσας επανάληψης στην επιθυμητή. Αν το σφάλμα σύγκλισης είναι πολύ μικρό, η σύγκλιση μπορεί να είναι πολύ αργή ή να μη γίνει καθόλου. Αν το σφάλμα σύγκλισης είναι πολύ μεγάλο, μπορεί να παραβιαστούν οι απαιτήσεις ακρίβειας ή η σύγκλιση να θεωρηθεί λάθος. Το αποτέλεσμα της σύγκλισης μπορεί να είναι μία σημαντική βελτίωση της μεθόδου Jacobi και της μεθόδου Gauss-Seidel. ([5],[6])

Για κάθε μια μέθοδο, σε κάθε επανάληψη ο αλγόριθμος συγκρίνει την πιθανότητα κάθε κατάστασης P με την τιμή στην προηγούμενη επανάληψη Q.

Έτσι, για την μέθοδο Jacobi, η απόκλιση κάθε πιθανότητας μετράται με το σχετικό σφάλμα:

$$\text{Απόκλιση} = \frac{|\,P(y) - Q(y)\,|}{Q(y)}$$

Επειδή η πιθανότητα της προηγούμενης επανάληψης μπορεί να είναι 0 και να οδηγήσει τον αλγόριθμο σε overflow, μπορούμε να το αποφύγουμε βάζοντας στον παρονομαστή την αρχική τιμή του Q(y):

$$\text{Απόκλιση} = \frac{|\,P(y) - Q(y)\,|}{\frac{1}{\text{Πλήθος\_εφικτών\_καταστάσεων}}} =$$

$$= \text{Πλήθος\_εφικτών\_καταστάσεων*}\,|\,P(y) - Q(y)\,|$$

Στη μέθοδο Gauss-Seidel, χρησιμοποιούμε μόνο ένα διάνυσμα P που εκφράζει την τιμή της πιθανότητας στην παρούσα επανάληψη και μία ακόμα μεταβλητή OLD_P που εκφράζει την τιμή της πιθανότητας στην προηγούμενη επανάληψη. Οπότε, το σχετικό σφάλμα που μετράει την απόκλιση κάθε πιθανότητας θα είναι:

$$\text{Απόκλιση} = \frac{|\,P(y) - OLD\_P\,|}{OLD\_P}$$

Επειδή η πιθανότητα της αρχικής κατάστασης σε κάποια επανάληψη να είναι 0 και να οδηγήσει τον αλγόριθμο σε overflow, μπορούμε να το αποφύγουμε όπως προηγουμένως:

$$\text{Απόκλιση} = \frac{|\,P(y) - OLD\_P\,|}{\frac{1}{\text{Πλήθος\_εφικτών\_καταστάσεων}}} =$$

$$= \text{Πλήθος\_εφικτών\_καταστάσεων*}\,|\,P(y) - OLD\_P\,|$$

Αν η απόκλιση για κάποια κατάσταση y είναι μεγαλύτερη από την μέγιστη απόκλιση (η οποία έχει αρχικοποιηθεί ίση με 0 για κάθε επανάληψη), τότε στην μέγιστη απόκλιση εκχωρείται η τιμή της απόκλισης. Επίσης, στις πιθανότητες των προηγούμενων επαναλήψεων Q και OLD_P εκχωρούνται οι πιθανότητες της τρέχουσας επανάληψης. Στο τέλος της επανάληψης γίνεται ο έλεγχος σύγκλισης, κατά τον οποίο ελέγχεται αν η μέγιστη απόκλιση είναι μικρότερη από το σφάλμα σύγκλισης, οπότε σταματάνε οι επαναλήψεις, διαφορετικά ο αλγόριθμος συνεχίζει μέχρι η μέγιστη απόκλιση να γίνει μικρότερη από την επιθυμητή τιμή (0.01).

3.4 Υπολογισμός μέτρων απόδοσης

Τα μέτρα απόδοσης που υπολογίζει ο αλγόριθμος είναι:

1. Μέσος ρυθμός παραγωγής
2. Μέση στάθμη αποθήκης B_i
3. Μέση στάθμη όλων των αποθηκών της γραμμής παραγωγής

3.4.1 Μέσος ρυθμός παραγωγής

Η παραγωγικότητα ΤΗ (throughput) μιας γραμμής παραγωγής προσεγγίζεται με το μέσο ρυθμό παραγωγής της τελευταίας μηχανής, δηλαδή προκύπτει από το γινόμενο του ρυθμού παραγωγής της τελευταίας μηχανής M_1 με το άθροισμα των πιθανοτήτων για τις οποίες η τελευταία μηχανή M_1 της γραμμής παραγωγής είναι λειτουργική (παράγει) και η προηγούμενή της αποθήκη B_1 έχει τουλάχιστον 1 κομμάτι. Δηλαδή, παίρνουμε υπόψιν μας τις καταστάσεις από τις οποίες η γραμμή παραγωγής μπορεί άμεσα να παράγει (στην επόμενη κατάσταση). Έτσι, για κάθε εφικτή κατάσταση y, όπου η M_1 είναι στην κατάσταση 1 και η στάθμη της B_1 είναι μεγαλύτερη του 1, προσθέτω σε μια μεταβλητή SUM την πιθανότητα αυτής της κατάστασης. Το αποτέλεσμα του SUM το πολλαπλασιάζω με τον ρυθμό παραγωγής της M_1. Οπότε, η παραγωγικότητα δίνεται από τον τύπο:

$$TH = \mu_1 * \sum_{\forall \text{εφικτη}} P(M_1 = 1 \cap B_1 > 0)$$

3.4.2 Μέση στάθμη αποθήκης B_i

Η μέση στάθμη κάθε αποθήκης Bi προσεγγίζεται από το μέσο απόθεμα κάθε αποθήκης Bi, το οποίο ισούται με το γινόμενο του αποθέματος της αποθήκης επί την πιθανότητα η γραμμή παραγωγής να βρίσκεται σε κατάσταση κατά την οποία η αποθήκη Bi έχει το συγκεκριμένο απόθεμα. Έτσι, ισχύει ο τύπος:

$$\overline{B_i} = \sum_{\forall \text{εφικτή}} B_i * P(y)$$

3.4.3 Μέση στάθμη όλων των αποθηκών της γραμμής παραγωγής

Η μέση στάθμη όλων των αποθηκών της γραμμής παραγωγής Β προσεγγίζεται από το μέσο απόθεμα της γραμμής παραγωγής, το οποίο ισούται με το άθροισμα της μέσης στάθμης κάθε αποθήκης. Έτσι, έχουμε τον τύπο:

$$N = \overline{B} = \sum_{i=1}^{k-1} \overline{B}_i$$

4 ΑΡΙΘΜΗΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

Μετά την ανάλυση της γραμμής παραγωγής με k μηχανές και k-1 αποθήκες, έγιναν πειράματα με βάση τους αλγορίθμους που αναπτύχθηκαν με τις μεθόδους Jacobi και Gauss- Seidel. Τα αποτελέσματα της παραγωγικότητας και του μέσου αποθέματος στα οποία κατέληξαν ο αλγόριθμος με τη μέθοδο Jacobi, ο αλγόριθμος με τη μέθοδο Gauss- Seidel και ο αλγόριθμος της προσομοίωσης συγκρίνονται μεταξύ τους ως προς την ακρίβειά τους και την ταχύτητα εκτέλεσης των αλγορίθμων. Σε κάθε πείραμα μεταβάλαμε μια παράμετρο που αφορά ρυθμό κατά 10 % και 20% και την παράμετρο που αφορά την χωρητικότητα κατά 1 και 2 κρατώντας όλες τις υπόλοιπες σταθερές. Έτσι, μπορούμε να διαπιστώσουμε το πώς μεταβάλλεται το μέσο απόθεμα και η παραγωγικότητα με τη μεταβολή κάθε μιας παραμέτρου ξεχωριστά (ρυθμός παραγωγής, ρυθμός βλάβης, ρυθμός επισκευής, χωρητικότητα). Οι μεταβολές γίνονται ταυτόχρονα για όλες τις μηχανές αν πρόκειται για ρυθμούς ή ταυτόχρονα για όλες τις αποθήκες αν πρόκειται για χωρητικότητα, αλλά για διαφορετικές παραμέτρους κάθε φορά. Τα πειράματα έγιναν σε ηλεκτρονικό υπολογιστή με μνήμη RAM 2 GB.

4.1 Σύστημα 4 μηχανών

4.1.1 Αρχικές τιμές συστήματος

Διεξάγαμε πειράματα για μία γραμμή παραγωγής με 4 μηχανές.

$$M_4 \rightarrow B_3 \rightarrow M_3 \rightarrow B_2 \rightarrow M_2 \rightarrow B_1 \rightarrow M_1$$

Δώσαμε αρχικές τιμές ως εξής:

Ρυθμοί παραγωγής	Ρυθμοί βλάβης	Ρυθμοί επισκευής
$\mu_4 = 1$	$p_4 = 0.05$	$r_4 = 0.1$
$\mu_3 = 0.8$	$p_3 = 0.05$	$r_3 = 0.1$
$\mu_2 = 1.2$	$p_2 = 0.05$	$r_2 = 0.1$
$\mu_1 = 1$	$p_1 = 0.05$	$r_1 = 0.1$

Για τις αποθήκες πήραμε ως αρχική τιμή $C_i = 2$ κομμάτια.

Για αυτές τις τιμές, παίρνουμε τα εξής αποτελέσματα:

Πίνακας 2: Αποτελέσματα των μεθόδων για τις αρχικές τιμές

	Jacobi	**Gauss-Seidel**	**Προσομοίωση**
TH	0.330511	0.327017	0.326830492
N	5.346562	5.305130	5.21690623
Επαναλήψεις	36	29	-
Χρόνος (sec)	1.000000	1.000000	1.265625

4.1.2 Μεταβολή ρυθμού παραγωγής μ_i

Τώρα, μεταβάλουμε τον ρυθμό παραγωγής μ_i των μηχανών.

Για αύξηση των μ_i κατά 10%:

Ρυθμοί παραγωγής	Ρυθμοί βλάβης	Ρυθμοί επισκευής
$\mu_4 = 1.1$	$p_4 = 0.05$	$r_4 = 0.1$
$\mu_3 = 0.88$	$p_3 = 0.05$	$r_3 = 0.1$
$\mu_2 = 1.32$	$p_2 = 0.05$	$r_2 = 0.1$
$\mu_1 = 1.1$	$p_1 = 0.05$	$r_1 = 0.1$

Για αυτές τις τιμές, παίρνουμε τα εξής αποτελέσματα:

Πίνακας 3: Αποτελέσματα των μεθόδων για αύξηση των μ_i κατά 10%

	Jacobi	**Gauss-Seidel**	**Προσομοίωση**
TH	0.359861	0.356366	0.355913683
N	5.342030	5.306060	5.21478642
Επαναλήψεις	37	30	-
Χρόνος (sec)	1.000000	0.000000	1.25

Για αύξηση των μ_i κατά 20%:

Ρυθμοί παραγωγής	Ρυθμοί βλάβης	Ρυθμοί επισκευής
$\mu_4 = 1.2$	$p_4 = 0.05$	$r_4 = 0.1$
$\mu_3 = 0.96$	$p_3 = 0.05$	$r_3 = 0.1$
$\mu_2 = 1.44$	$p_2 = 0.05$	$r_2 = 0.1$
$\mu_1 = 1.2$	$p_1 = 0.05$	$r_1 = 0.1$

Για αυτές τις τιμές, παίρνουμε τα εξής αποτελέσματα:

Πίνακας 4: Αποτελέσματα των μεθόδων για αύξηση των μ_i κατά 20%

	Jacobi	**Gauss-Seidel**	**Προσομοίωση**
TH	0.389396	0.385541	0.384465306
N	5.340437	5.305637	5.21553908
Επαναλήψεις	37	30	-
Χρόνος (sec)	1.000000	0.000000	1.25

Για μείωση των μi κατά 10%:

Ρυθμοί παραγωγής	Ρυθμοί βλάβης	Ρυθμοί επισκευής
$\mu_4 = 0.9$	$p_4 = 0.05$	$r_4 = 0.1$
$\mu_3 = 0.72$	$p_3 = 0.05$	$r_3 = 0.1$
$\mu_2 = 1.08$	$p_2 = 0.05$	$r_2 = 0.1$
$\mu_1 = 0.9$	$p_1 = 0.05$	$r_1 = 0.1$

Για αυτές τις τιμές, παίρνουμε τα εξής αποτελέσματα:

Πίνακας 5: Αποτελέσματα των μεθόδων για μείωση των μ_i κατά 10%

	Jacobi	**Gauss-Seidel**	**Προσομοίωση**
TH	0.300657	0.297535	0.296783711
N	5.349190	5.306166	5.22118059
Επαναλήψεις	36	29	-
Χρόνος (sec)	1.000000	1.000000	1.28125

Για μείωση των μ_i κατά 20%:

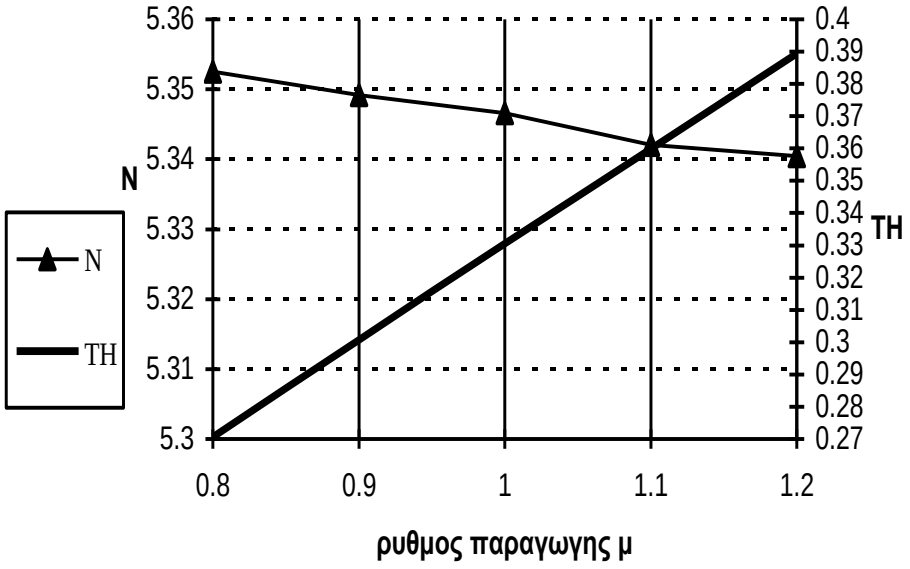
Ρυθμοί παραγωγής	Ρυθμοί βλάβης	Ρυθμοί επισκευής
$\mu_4 = 0.8$	$p_4 = 0.05$	$r_4 = 0.1$
$\mu_3 = 0.64$	$p_3 = 0.05$	$r_3 = 0.1$
$\mu_2 = 0.96$	$p_2 = 0.05$	$r_2 = 0.1$
$\mu_1 = 0.8$	$p_1 = 0.05$	$r_1 = 0.1$

Για αυτές τις τιμές, παίρνουμε τα εξής αποτελέσματα:

Πίνακας 6: Αποτελέσματα των μεθόδων για μείωση των μ_i κατά 20%

	Jacobi	**Gauss-Seidel**	**Προσομοίωση**
TH	0.270611	0.267848	0.267320385
N	5.352555	5.307739	5.22452307
Επαναλήψεις	36	29	-
Χρόνος (sec)	1.000000	0.000000	1.28125

Κάνουμε τώρα το διάγραμμα που δείχνει πως μεταβάλλεται το μέσο απόθεμα N και η παραγωγικότητα TH συναρτήσει του ρυθμού παραγωγής μ_i.



Σχήμα 5: Διάγραμμα μεταβολής του N και του TH συναρτήσει του μ_i

Από το διάγραμμα παρατηρούμε ότι καθώς αυξάνεται ο ρυθμός (πιθανότητα) παραγωγής μ_i των μηχανών, η παραγωγικότητα αυξάνεται και το μέσο απόθεμα μειώνεται.

4.1.3 Μεταβολή ρυθμού βλάβης p_i

Τώρα, μεταβάλουμε τον ρυθμό βλάβης p_i των μηχανών.

Για αύξηση των pi κατά 10%:

Ρυθμοί παραγωγής	Ρυθμοί βλάβης	Ρυθμοί επισκευής
$\mu_4 = 1$	$p_4 = 0.055$	$r_4 = 0.1$
$\mu_3 = 0.8$	$p_3 = 0.055$	$r_3 = 0.1$
$\mu_2 = 1.2$	$p_2 = 0.055$	$r_2 = 0.1$
$\mu_1 = 1$	$p_1 = 0.055$	$r_1 = 0.1$

Για αυτές τις τιμές, παίρνουμε τα εξής αποτελέσματα:

Πίνακας 7: Αποτελέσματα των μεθόδων για αύξηση των p_i κατά 10%

	Jacobi	**Gauss-Seidel**	**Προσομοίωση**
TH	0.315995	0.312685	0.312292971
N	5.347373	5.306437	5.21703436
Επαναλήψεις	36	29	-
Χρόνος (sec)	1.000000	0.000000	1.28125

Για αύξηση των p_i κατά 20%:

Ρυθμοί παραγωγής	Ρυθμοί βλάβης	Ρυθμοί επισκευής
$\mu_4 = 1$	$p_4 = 0.06$	$r_4 = 0.1$
$\mu_3 = 0.8$	$p_3 = 0.06$	$r_3 = 0.1$
$\mu_2 = 1.2$	$p_2 = 0.06$	$r_2 = 0.1$
$\mu_1 = 1$	$p_1 = 0.06$	$r_1 = 0.1$

Για αυτές τις τιμές, παίρνουμε τα εξής αποτελέσματα:

Πίνακας 8: Αποτελέσματα των μεθόδων για αύξηση των p_i κατά 20%

	Jacobi	**Gauss-Seidel**	**Προσομοίωση**
TH	0.302896	0.299748	0.299503858
N	5.348192	5.307516	5.22219747
Επαναλήψεις	36	29	-
Χρόνος (sec)	1.000000	0.000000	1.28125

Για μείωση των p_i κατά 10%:

Ρυθμοί παραγωγής	Ρυθμοί βλάβης	Ρυθμοί επισκευής
$\mu_4 = 1$	$p_4 = 0.045$	$r_4 = 0.1$
$\mu_3 = 0.8$	$p_3 = 0.045$	$r_3 = 0.1$
$\mu_2 = 1.2$	$p_2 = 0.045$	$r_2 = 0.1$
$\mu_1 = 1$	$p_1 = 0.045$	$r_1 = 0.1$

Για αυτές τις τιμές, παίρνουμε τα εξής αποτελέσματα:

Πίνακας 9: Αποτελέσματα των μεθόδων για μείωση των p_i κατά 10%

	Jacobi	**Gauss-Seidel**	**Προσομοίωση**
TH	0.346372	0.343029	0.342272612
N	5.343127	5.305081	5.21926831
Επαναλήψεις	37	30	-
Χρόνος (sec)	1.000000	0.000000	1.25

Για μείωση των p_i κατά 20%:

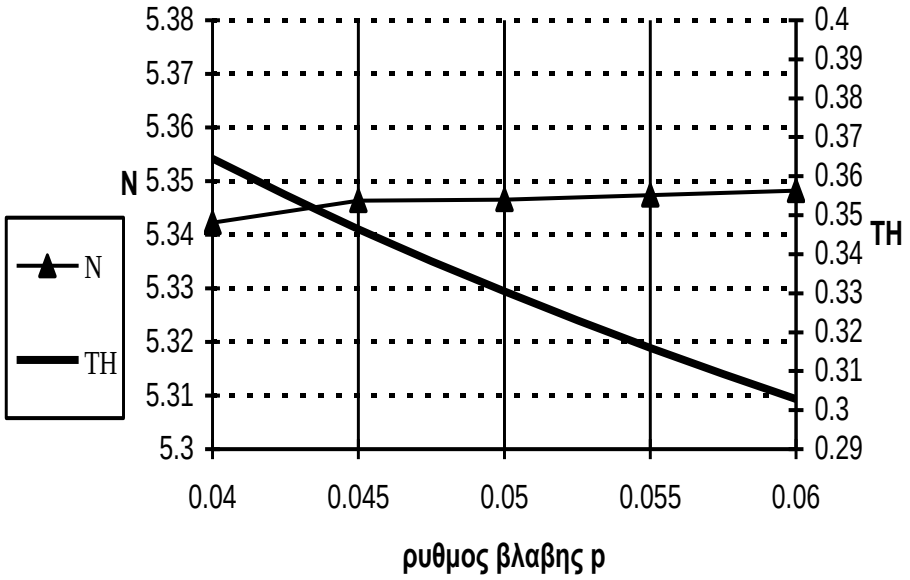
Ρυθμοί παραγωγής	Ρυθμοί βλάβης	Ρυθμοί επισκευής
$\mu_4 = 1$	$p_4 = 0.04$	$r_4 = 0.1$
$\mu_3 = 0.8$	$p_3 = 0.04$	$r_3 = 0.1$
$\mu_2 = 1.2$	$p_2 = 0.04$	$r_2 = 0.1$
$\mu_1 = 1$	$p_1 = 0.04$	$r_1 = 0.1$

Για αυτές τις τιμές, παίρνουμε τα εξής αποτελέσματα:

Πίνακας 10: Αποτελέσματα των μεθόδων για μείωση των p_i κατά 20%

	Jacobi	**Gauss-Seidel**	**Προσομοίωση**
TH	0.364520	0.360954	0.360330725
N	5.342226	5.303088	5.22409192
Επαναλήψεις	37	30	-
Χρόνος (sec)	1.000000	0.000000	1.25

Κάνουμε τώρα το διάγραμμα που δείχνει πως μεταβάλλεται το μέσο απόθεμα N και η παραγωγικότητα TH συναρτήσει του ρυθμού βλάβης p_i.



Σχήμα 6: Διάγραμμα μεταβολής του N και του TH συναρτήσει του p_i

Από το διάγραμμα παρατηρούμε ότι, καθώς ο ρυθμός (πιθανότητα) βλάβης p_i των μηχανών αυξάνεται, η παραγωγικότητα μειώνεται και το μέσο απόθεμα αυξάνεται.

4.1.4 Μεταβολή ρυθμού επισκευής r_i

Τώρα, μεταβάλουμε τον ρυθμό επισκευής r_i των μηχανών.

Για αύξηση των r_i κατά 10%:

Ρυθμοί παραγωγής	Ρυθμοί βλάβης	Ρυθμοί επισκευής
$\mu_4 = 1$	$p_4 = 0.05$	$r_4 = 0.11$
$\mu_3 = 0.8$	$p_3 = 0.05$	$r_3 = 0.11$
$\mu_2 = 1.2$	$p_2 = 0.05$	$r_2 = 0.11$
$\mu_1 = 1$	$p_1 = 0.05$	$r_1 = 0.11$

Για αυτές τις τιμές, παίρνουμε τα εξής αποτελέσματα:

Πίνακας 11: Αποτελέσματα των μεθόδων για αύξηση των r_i κατά 10%

	Jacobi	**Gauss-Seidel**	**Προσομοίωση**
TH	0.348167	0.344633	0.344405228
N	5.347095	5.305207	5.22074352
Επαναλήψεις	36	29	-
Χρόνος (sec)	1.000000	0.000000	1.265625

Για αύξηση των r_i κατά 20%:

Ρυθμοί παραγωγής	Ρυθμοί βλάβης	Ρυθμοί επισκευής
$\mu_4 = 1$	$p_4 = 0.05$	$r_4 = 0.12$
$\mu_3 = 0.8$	$p_3 = 0.05$	$r_3 = 0.12$
$\mu_2 = 1.2$	$p_2 = 0.05$	$r_2 = 0.12$
$\mu_1 = 1$	$p_1 = 0.05$	$r_1 = 0.12$

Για αυτές τις τιμές, παίρνουμε τα εξής αποτελέσματα:

Πίνακας 12: Αποτελέσματα των μεθόδων για αύξηση των r_i κατά 20%

	Jacobi	**Gauss-Seidel**	**Προσομοίωση**
TH	0.364360	0.360784	0.360502477
N	5.347526	5.305102	5.2240491
Επαναλήψεις	36	29	-
Χρόνος (sec)	1.000000	0.000000	1.28125

Για μείωση των r_i κατά 10%:

Ρυθμοί παραγωγής	Ρυθμοί βλάβης	Ρυθμοί επισκευής
$\mu_4 = 1$	$p_4 = 0.05$	$r_4 = 0.09$
$\mu_3 = 0.8$	$p_3 = 0.05$	$r_3 = 0.09$
$\mu_2 = 1.2$	$p_2 = 0.05$	$r_2 = 0.09$
$\mu_1 = 1$	$p_1 = 0.05$	$r_1 = 0.09$

Για αυτές τις τιμές, παίρνουμε τα εξής αποτελέσματα:

Πίνακας 13: Αποτελέσματα των μεθόδων για μείωση των r_i κατά 10%

	Jacobi	**Gauss-Seidel**	**Προσομοίωση**
TH	0.310879	0.307777	0.307581174
N	5.343500	5.306724	5.21251885
Επαναλήψεις	37	30	-
Χρόνος (sec)	1.000000	0.000000	1.265625

Για μείωση των r_i κατά 20%:

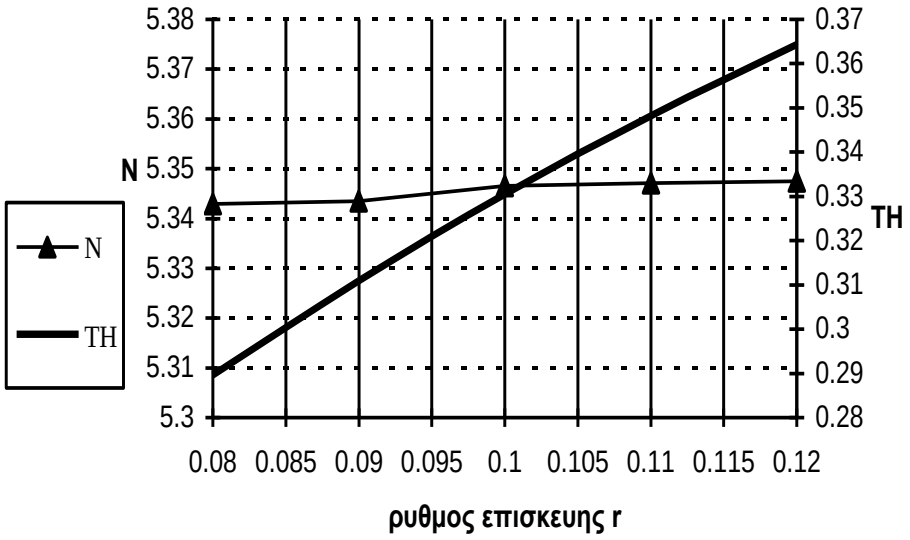
Ρυθμοί παραγωγής	Ρυθμοί βλάβης	Ρυθμοί επισκευής
$\mu_4 = 1$	$p_4 = 0.05$	$r_4 = 0.08$
$\mu_3 = 0.8$	$p_3 = 0.05$	$r_3 = 0.08$
$\mu_2 = 1.2$	$p_2 = 0.05$	$r_2 = 0.08$
$\mu_1 = 1$	$p_1 = 0.05$	$r_1 = 0.08$

Για αυτές τις τιμές, παίρνουμε τα εξής αποτελέσματα:

Πίνακας 14: Αποτελέσματα των μεθόδων για μείωση των r_i κατά 20%

	Jacobi	**Gauss-Seidel**	**Προσομοίωση**
TH	0.289677	0.286610	0.286427964
N	5.342897	5.306308	5.20748952
Επαναλήψεις	37	30	-
Χρόνος (sec)	1.000000	0.000000	1.265625

Κάνουμε τώρα το διάγραμμα που δείχνει πως μεταβάλλεται το μέσο απόθεμα N και η παραγωγικότητα TH συναρτήσει του ρυθμού επισκευής r_i.



Σχήμα 7: Διάγραμμα μεταβολής του N και του TH συναρτήσει του r_i

Από το διάγραμμα παρατηρούμε ότι, καθώς ο ρυθμός (πιθανότητα) επισκευής r_i των μηχανών αυξάνεται, η παραγωγικότητα αυξάνεται και το μέσο απόθεμα επίσης αυξάνεται.

4.1.5 Μεταβολή χωρητικότητας C_i

Τώρα, μεταβάλουμε την χωρητικότητα C_i των αποθηκών.

Για αύξηση των C_i κατά 1:

Ρυθμοί παραγωγής	Ρυθμοί βλάβης	Ρυθμοί επισκευής
$\mu_4 = 1$	$p_4 = 0.05$	$r_4 = 0.1$
$\mu_3 = 0.8$	$p_3 = 0.05$	$r_3 = 0.1$
$\mu_2 = 1.2$	$p_2 = 0.05$	$r_2 = 0.1$
$\mu_1 = 1$	$p_1 = 0.05$	$r_1 = 0.1$

Για αυτές τις τιμές, παίρνουμε τα εξής αποτελέσματα:

Πίνακας 15: Αποτελέσματα των μεθόδων για αύξηση των C_i κατά 1

	Jacobi	**Gauss-Seidel**	**Προσομοίωση**
TH	0.351364	0.347254	0.346400423
N	6.762785	6.693879	6.60950183
Επαναλήψεις	44	36	-
Χρόνος (sec)	1.000000	1.000000	1.25

Για αύξηση των C_i κατά 2:

Ρυθμοί παραγωγής	Ρυθμοί βλάβης	Ρυθμοί επισκευής
$\mu_4 = 1$	$p_4 = 0.05$	$r_4 = 0.1$
$\mu_3 = 0.8$	$p_3 = 0.05$	$r_3 = 0.1$
$\mu_2 = 1.2$	$p_2 = 0.05$	$r_2 = 0.1$
$\mu_1 = 1$	$p_1 = 0.05$	$r_1 = 0.1$

Για αυτές τις τιμές, παίρνουμε τα εξής αποτελέσματα:

Πίνακας 16: Αποτελέσματα των μεθόδων για αύξηση των C_i κατά 2

	Jacobi	**Gauss-Seidel**	**Προσομοίωση**
TH	0.369177	0.364233	0.362430039
N	8.184300	8.077167	7.98459439
Επαναλήψεις	50	42	-
Χρόνος (sec)	1.000000	1.000000	1.25

Για μείωση των C_i κατά 1:

Ρυθμοί παραγωγής	Ρυθμοί βλάβης	Ρυθμοί επισκευής
$\mu_4 = 1$	$p_4 = 0.05$	$r_4 = 0.1$
$\mu_3 = 0.8$	$p_3 = 0.05$	$r_3 = 0.1$
$\mu_2 = 1.2$	$p_2 = 0.05$	$r_2 = 0.1$
$\mu_1 = 1$	$p_1 = 0.05$	$r_1 = 0.1$

Για αυτές τις τιμές, παίρνουμε τα εξής αποτελέσματα:

Πίνακας 17: Αποτελέσματα των μεθόδων για μείωση των C_i κατά 1

	Jacobi	**Gauss-Seidel**	**Προσομοίωση**
TH	0.303950	0.301352	0.301096123
N	3.926218	3.909150	3.80325515
Επαναλήψεις	27	23	-
Χρόνος (sec)	0.000000	0.000000	1.28125

Για μείωση των C_i κατά 2:

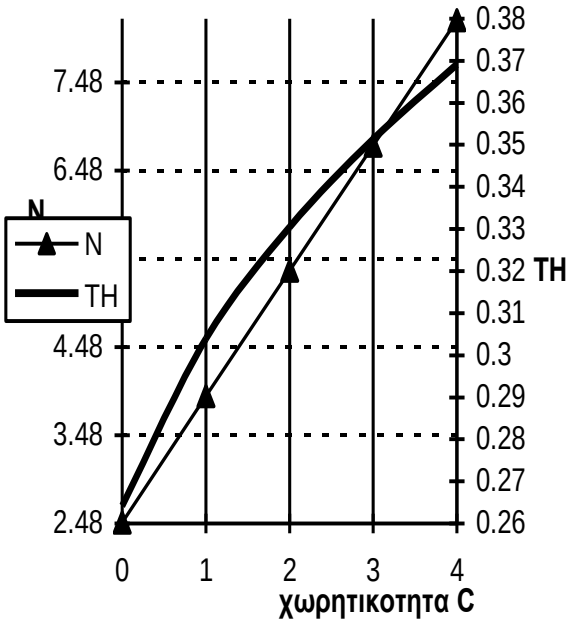
Ρυθμοί παραγωγής	Ρυθμοί βλάβης	Ρυθμοί επισκευής
$\mu_4 = 1$	$p_4 = 0.05$	$r_4 = 0.1$
$\mu_3 = 0.8$	$p_3 = 0.05$	$r_3 = 0.1$
$\mu_2 = 1.2$	$p_2 = 0.05$	$r_2 = 0.1$
$\mu_1 = 1$	$p_1 = 0.05$	$r_1 = 0.1$

Για αυτές τις τιμές, παίρνουμε τα εξής αποτελέσματα:

Πίνακας 18: Αποτελέσματα των μεθόδων για μείωση των C_i κατά 2

	Jacobi	**Gauss-Seidel**	**Προσομοίωση**
TH	0.264251	0.263778	0.261898931
N	2.492937	2.494154	2.34594354
Επαναλήψεις	20	21	-
Χρόνος (sec)	0.000000	0.000000	1.3125

Κάνουμε τώρα το διάγραμμα που δείχνει πως μεταβάλλεται το μέσο απόθεμα N και η παραγωγικότητα TH συναρτήσει της χωρητικότητας C_i των αποθηκών.



Σχήμα 8: Διάγραμμα μεταβολής του N και του TH συναρτήσει του C_i

Από το διάγραμμα παρατηρούμε ότι, καθώς η χωρητικότητα των αποθηκών αυξανεται, τόσο η παραγωγικότητα όσο και το μέσο απόθεμα αυξάνονται.

Από τα αποτελέσματα που έδωσαν τα πειράματα, προκύπτει ότι οι αλγόριθμοι ανάλυσης της παραγωγής που αναπτύξαμε στην παρούσα διπλωματική εργασία με τις μεθόδους Jacobi και Gauss- Seidel είναι σαφώς ταχύτεροι από τον αλγόριθμο της προσομοίωσης, αφού απαιτούν μόνο μερικά κλάσματα του δευτερολέπτου για να δώσουν το αποτέλεσμα. Βέβαια, υστερούν σε ακρίβεια, κάτι το οποίο οφείλεται στο γεγονός ότι υπάρχουν πολλές πιθανότητες που είναι μικρές και ο υπολογιστής όταν τις αποθηκεύει αποκόπτει σημαντικά ψηφία τους. Μεταξύ των δύο μεθόδων που χρησιμοποιήσαμε, η μέθοδος Gauss- Seidel συγκλίνει ταχύτερα, κάτι το οποίο φαίνεται από τον αριθμό των επαναλήψεων οι οποίες απαιτούνται.

5 ΣΥΜΠΕΡΑΣΜΑΤΑ- ΕΠΕΚΤΑΣΕΙΣ

Στην παρούσα διπλωματική εργασία ασχοληθήκαμε με την ανάλυση των γραμμών παραγωγής που υπόκεινται σε βλάβες και πιο συγκεκριμένα, με την αριθμητική επίλυση αλυσίδων Markov μεγάλων διαστάσεων για βιομηχανικές γραμμές παραγωγής. Αναπτύχθηκαν 2 αλγόριθμοι, οι οποίοι επιλύουν τις εξισώσεις Chapman-Kolmogorov με αυτόματο υπολογισμό των πινάκων που απαιτούνται. Αυτό επιτυγχάνεται με τη χρήση προσεγγιστικών μοντέλων Markov. Τα αποτελέσματα δεν είναι πάντα ακριβή, αλλά είναι ένας τρόπος να επιλυθούν μεγάλα συστήματα εξισώσεων που απαιτούν πράξεις πολύ μεγάλων πινάκων. Η χρήση αριθμητικών μεθόδων έχει ιδιαίτερη σημασία για την επίλυση τέτοιων συστημάτων, γι' αυτό χρησιμοποιήσαμε τις μεθόδους Jacobi και Gauss-Seidel. Στο τέλος των 2 αλγορίθμων υπολογίζουμε την παραγωγικότητα και το μέσο απόθεμα της γραμμής παραγωγής.

Για ένα σύστημα με 4 μηχανές και 3 αποθήκες, κάνουμε πειράματα μεταβάλλοντας τους ρυθμούς παραγωγής, βλάβης και επισκευής των μηχανών καθώς και την χωρητικότητα των αποθηκών και συγκρίνουμε τις 2 μεθόδους μεταξύ τους και με την προσομοίωση ως προς την ακρίβεια των αποτελεσμάτων και την ταχύτητα εκτέλεσης των αλγορίθμων. Προκύπτει, λοιπόν, ότι οι 2 αλγόριθμοι είναι σαφώς ταχύτεροι από τον αλγόριθμο της προσομοίωσης (και κυρίως ο αλγόριθμος που χρησιμοποιεί τη μέθοδο Gauss- Seidel), ενώ η απόκλιση των αποτελεσμάτων στα οποία καταλήγουν σε σχέση με τα αποτελέσματα της προσομοίωσης είναι εντός ανεκτών ορίων. Σημειώνουμε ότι ο αλγόριθμος στρογγυλοποιεί τον χρόνο εκτέλεσης του και δεν βγάζει αποτελέσματα με ακρίβεια δεκαδικών ψηφίων. Παρ' όλα αυτά είναι προφανές ότι είναι πολύ ταχύτερος από την προσομοίωση.

Συνεπώς, βασιζόμενοι σε αυτούς τους 2 αλγορίθμους, μπορούμε να κάνουμε ανάλυση μεγάλων γραμμών παραγωγής, όπου η προσομοίωση δεν ενδείκνυται. Ένα από τα σημεία που θα μπορούσαν να μελετηθούν περεταίρω είναι η χρήση τεχνικών εξοικονόμησης μνήμης RAM κατά την εκτέλεση των αλγορίθμων, έτσι ώστε να είναι δυνατή η ανάλυση μεγάλων γραμμών παραγωγής χωρίς να απαιτείται μεγάλος χώρος από τη μνήμη του υπολογιστή. Με αυτό τον τρόπο μπορούν να αποφευχθούν προβλήματα έλλειψης μνήμης όπου σε κάποιο σημείο του ο αλγόριθμος έχει χρησιμοποιήσει όλη τη μνήμη του υπολογιστή και δεν μπορεί να συνεχίσει να «τρέχει» ώστε να καταλήξει σε αποτέλεσμα. Για να αποφευχθούν τέτοια προβλήματα προτείνεται οι αλγόριθμοι να «τρέχουν» σε υπολογιστές με όσο το δυνατόν μεγαλύτερη μνήμη RAM. Τέλος, θα μπορούσαν να διερευνηθούν οι λόγοι που οδηγούν σε αποκλίσεις όσον αφορά τα αποτελέσματα για την παραγωγικότητα και για το μέσο απόθεμα του αναλυτικού μοντέλου από εκείνο της προσομοίωσης.

6 ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Μ. Βιδάλης, *Εκτίμηση απόδοσης και βέλτιστη κατανομή αποθηκευτικών χώρων σε γραμμές παραγωγής*, Διδακτορική Διατριβή, Τμήμα Μαθηματικών, Πανεπιστήμιο Αιγαίου, 1998

[2] Β. Κουϊκόγλου, *Προσομοίωση*, Σημειώσεις μαθήματος, Πολυτεχνείο Κρήτης, 2006

[3] W. J. Stewart, Numerical Methods for Computing Stationary Distributions of Finite Irreducible Markov Chains, Chapter 3, Advances in Computational Probability, W. Grassmann, Kluwer, 1999

[4] Γ. Φίλη, *Δίκτυα Παραγωγής*, Σημειώσεις μαθήματος, Πολυτεχνείο Κρήτης, 2003

[5] D. Mitra and P.Tsoucas (1988), Relaxations for the Numerical Solutions of Some Stochastic Problems, Commun. Statist.-Stochastic Models, Vol. 4 (3), pp. 387-419.

[6] D. Radev, V. Denchev and E. Rashkova, Steady-State Solutions of Markov Chains, The 7th Balkan Conference on Operational Research "BACOR 05", Constanta, Romania, May 2005.

[7] D. Gross and B. Gu, R. M. Soland (1993), Iterative Solution Methods for Obtaining the Steady-State Probability Distributions of Markovian Multi-Echelon Repairable Item Inventory Systems, Computers Ops Res. Vol. 20, No 8, pp. 817-828, 1993

[8] I. Adan. *Numerical Solution of Equilibrium Equations: Iterative Methods*. Available: http://www.win.tue.nl/~iadan/algoritme/notes2.pdf

[9] R. Mehmood, Serial Disk-Based Analysis of Large Stochastic Models, Validation of Stochastic Systems, LNCS 2925, pp. 230-255,2004

[10] R. B. Cooper and D. Gross (1991), On the Convergence of Jacobi and Gauss-Seidel Iteration for Steady-State Probabilities of Finite-State Continuous-Time Markov Chains, Commun. Statist.-Stochastic Models, Vol. 7 (1), pp. 185-189

[11] M. E. H Petering, J. Seo and C. Lee, Performance Analysis of a Multiple Vehicle Tandem System with Inter-Vehicle Buffers and Blocking, Computers in Industry, Vol. 58, pp. 3-11, 2007. Available: http://sciencedirect.com

[12] Β. Κουϊκόγλου, *Αξιοπιστία Συστημάτων, Συντήρηση, Αντικατάσταση*, Σημειώσεις μαθήματος, Πολυτεχνείο Κρήτης, 2007

[13] Γ. Φίλη, *Στοχαστικές Διαδικασίες*, Σημειώσεις μαθήματος, Πολυτεχνείο Κρήτης, 2006

[14] G. Strang, *Γραμμική Άλγεβρα και Εφαρμογές,* Πανεπιστημιακές Εκδόσεις Κρήτης, 2005

ΠΑΡΑΡΤΗΜΑ

Κώδικας του αλγορίθμου με τη μέθοδο Jacobi Over- Relaxation

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <string.h>

time_t start,end;
double dif;
int *w, *m, *x,*ci,y;
int k;
int n;
double *o,*f,*r;
double epsilon, sfalma;

int decodeb();
void init();


int main()
{
      int i;
      FILE* file_out;

      init();
      free(ci);
      time(&start);

      w[0]=1;
      for(i=1;i<2*k;i++)
      {
         w[i]=w[i-1]*(m[i-1]+1);
      }
      n=w[2*k-2]*(m[2*k-2]+1);
      decodeb();

      printf("\n");
      system("PAUSE");
}

//Sinartisi gia to diavasma twn dedomenwn apo ena arxeio txt.
//Ta dedomena pou dinei o xristis einai: o arithmos twn apothikwn k,
//oi rithmoi paragwgis,vlavis kai episkevis twn mixanwn
//kai i xwritikotita twn apothikwn

void init()
{
    FILE* file_in;
    char c;
    int i;
    file_in = fopen("dedomena.txt", "r");

    while(1)
    {
```

```
    fscanf(file_in, "%c", &c);
    if(c<'0' || c>'9')
    {
        while(c != '\n')
        {
            fscanf(file_in, "%c", &c);
        }
    }
    else
    {
        ungetc(c, file_in);
        break;
    }
}

fscanf(file_in, "%d",&k);

m=(int*)malloc((2*k-1)*sizeof(int));
ci=(int*)malloc((2*k-1)*sizeof(int));
x=(int*)malloc((2*k-1)*sizeof(int));
w=(int*)malloc((2*k-1)*sizeof(int));
o=(double*)malloc((2*k-1)*sizeof(double)); // o=rithmos paragwgis
f=(double*)malloc((2*k-1)*sizeof(double)); // f=rithmos vlavis
r=(double*)malloc((2*k-1)*sizeof(double)); //r=rithmos episkevis

while(1)
{
    fscanf(file_in, "%c", &c);
    if(c<'0' || c>'9')
    {
        while(c != '\n')
        {
            fscanf(file_in, "%c", &c);
        }
    }
    else
    {
        ungetc(c, file_in);
        break;
    }
}

for (i=0;i<2*k-1;i=i+2)
{
   fscanf(file_in, "%lf",&o[i]);
    fscanf(file_in, "%lf",&f[i]);
    fscanf(file_in, "%lf",&r[i]);
}

while(1)
{
    fscanf(file_in, "%c", &c);
    if(c<'0' || c>'9')
    {
        while(c != '\n')
        {
            fscanf(file_in, "%c", &c);
        }
    }
    else
    {
        ungetc(c, file_in);
        break;
    }
}
for (i=1;i<2*k-1;i=i+2)
{
   fscanf(file_in, "%d",&ci[i]);
```

```
        m[i]=ci[i]+2;
     }

     for(i=0; i<2*k; i=i+2)
           m[i]=1;


       fclose(file_in);

       file_in = fopen("apoklisi.txt", "r");
    while(1)
    {
        fscanf(file_in, "%c", &c);
        if(c<'0' || c>'9')
        {
            while(c != '\n')
            {
                fscanf(file_in, "%c", &c);
            }
        }
        else
        {
            ungetc(c, file_in);
            break;
        }
    }
    fscanf(file_in, "%lf", &sfalma);
    fscanf(file_in, "%lf", &epsilon);
    fclose(file_in);
}

int decodeb()
{
    int fores,t,u,v,g,q,pointer,km,metr,ip,meas,po,poi,poib;
    int pointer_b,re,rv,rp,pp,ipi,pros,prosb,prosc,yp,prosw,pr,apoth,ligo;
    int sum_efiktes,i,j,yrem,*efiktes,*sum,**efiktoi_syn;
    int *mallon_efiktoi_syn, ,**efiktoi_syn_t;
    double *palia_p,*nea_p,**metavasi,max_apokl,apokl,one,sum_nea_p;
    double *a,sump,gi,th,*nb,sumn,sum_d,sumpb,tempp,dif;
    FILE* fout;

     sum=(int*)malloc(n*sizeof(int));
     efiktoi_syn=(int**)malloc(n*sizeof(int*));
     for (t=0;t<n;t++)
     {
         efiktoi_syn[t]=(int*)malloc((2*k-1)*sizeof(int));
     }
     sum_efiktes=0;

     //apokwdikopoiisi
     for (y=0;y<n;y++)
     {
             yrem=y;
        for(i=2*k-2;i>=0;i--)
        {
            x[i]=yrem/w[i];
            yrem=yrem-x[i]*w[i];
        }

        //evresi efiktwn kai mi efiktwn katastasewn
        u=0;
        v=0;
        g=0;
        for (i=2*k-2;i>=0;i=i-2)
        {
           if ((i<2*k-2) && (i>0))
           {
              if ((x[i+1]==m[i+1]) && (x[i-1]==m[i-1]))
```

```
      {
         u=u+1;
      }
      if (x[i-1]==m[i-1])
      {

         if (x[i]==0)
         {
            u=u+1;
         }
      }
      if (x[i+1]==0)
      {

         if (x[i]==0)
         {
            u=u+1;
         }
      }
      if (x[i]==0)
      {
         if (f[i]==0)
         {
            u=u+1;
         }
      }
      if ((x[i]==0) && (x[i-1]==m[i-1]-1) && (x[i-3]==m[i-3]))
      {
            u=u+1;
       }
   }
   if (i==2*k-2)
   {
      if ((x[i-1]==m[i-1]) && (x[i]==0))
      {
         u=u+1;
      }
      if ((x[i-1]==m[i-1]) && (x[i-3]==m[i-3]))
      {
         u=u+1;
      }
      if ((x[i]==0) && (f[i]==0))
      {
         u=u+1;
      }
      if ((x[i]==0) && (x[i-1]==m[i-1]-1) && (x[i-3]==m[i-3]))
      {
            u=u+1;
       }
   }
   if (i==0)
   {
      if ((x[i]==0) && (x[i+1]==0))
      {
         u=u+1;
      }
      if ((x[i]==0) && (f[i]==0))
      {
         u=u+1;
      }
   }
 }
 /*if (u!=0)
    Mi efikti katastasi*/
if (u==0)
 {
    //Efikti katastasi
    sum[sum_efiktes]=y;
```

```
       for (i=2*k-2;i>=0;i--)
       {
           efiktoi_syn[sum_efiktes][i]=x[i];
       }
       sum_efiktes=sum_efiktes+1;
    }
}
free(x);
free(w);
efiktes=(int*)malloc(sum_efiktes*sizeof(int));
efiktoi_syn_t=(int**)malloc(sum_efiktes*sizeof(int*));
for (i=0;i<sum_efiktes;i++)
{
    efiktoi_syn_t[i]=(int*)malloc((2*k-1)*sizeof(int));
}
for (i=0;i<sum_efiktes;i++)
{
  for (t=2*k-2;t>=0;t--)
  {
      efiktoi_syn_t[i][t]=efiktoi_syn[i][t];
  }
}
for (i=0;i<n;i++)
{
  free(efiktoi_syn[i]);
}
free(efiktoi_syn);
for (i=0;i<sum_efiktes;i++)
{
  efiktes[i]=sum[i];
}
free(sum);
mallon_efiktoi_syn=(int*)malloc((2*k-1)*sizeof(int));
one=1;
metavasi=(double**)malloc(sum_efiktes*sizeof(double*));
for (i=0;i<sum_efiktes;i++)
{
  metavasi[i]=(double*)malloc(sum_efiktes*sizeof(double));
}

//evresi efiktwn metavasewn kai rithmwn metavasis
for (i=0;i<sum_efiktes;i++)
{
    pointer=2*k-2;
    km=k;
    ligo=0;
    do{


         if (efiktoi_syn_t[i][pointer]==0)
         {
            //EPISKEVI
            efiktoi_syn_t[i][pointer]=1;
            for (t=2*k-2;t>=0;t--)
            {
               mallon_efiktoi_syn[t]=efiktoi_syn_t[i][t];
            }
           efiktoi_syn_t[i][pointer]=0;
            metr=0;
            poi=0;
            do{
                po=0;
                for (t=2*k-2;t>=0;t--)
                {
                    if (mallon_efiktoi_syn[t]==efiktoi_syn_t[metr][t])
                    {
                           po=po+1;
                     }
```

```
     }
     if (po==2*k-1)
     {
         poi=poi+1;
         break;
     }
     metr=metr+1;

} while (metr<sum_efiktes);
if ((poi!=0) && (r[ligo]!=0))
{
    //Efikti metavasi
    metavasi[i][metr]=r[ligo];
 }
//else mi efikti metavasi

if (efiktoi_syn_t[i][pointer]==1)
{
   //VLAVI
   efiktoi_syn_t[i][pointer]=0;
   for (t=2*k-2;t>=0;t--)
   {
          mallon_efiktoi_syn[t]=efiktoi_syn_t[i][t];
   }
   efiktoi_syn_t[i][pointer]=1;
   metr=0;
   poi=0;
   do{
       po=0;
       for (t=2*k-2;t>=0;t--)
       {
          if (mallon_efiktoi_syn[t]==efiktoi_syn_t[metr][t])
          {
             po=po+1;
          }
        }
        if (po==2*k-1)
        {
          poi=poi+1;
          break;
        }
        metr=metr+1;

} while (metr<sum_efiktes);
if ((poi!=0) && (f[ligo]!=0))
{
    //Efikti metavasi
    metavasi[i][metr]=f[ligo];
}
/*else
    Mi efikti metavasi*/

// PARAGWGI
if (pointer!=0)
{
 efiktoi_syn_t[i][pointer-1]=efiktoi_syn_t[i][pointer-1]+1;
}
else if (pointer==0)
{
    efiktoi_syn_t[i][pointer-1]=0;
}
if (pointer!=2*k-2)
{
   efiktoi_syn_t[i][pointer+1]=efiktoi_syn_t[i][pointer+1]-1;
}
else if (pointer==2*k-2)
{
     efiktoi_syn_t[i][pointer+1]=m[pointer-1];
```

```
                }
                for (t=2*k-2;t>=0;t--)
                {
                    mallon_efiktoi_syn[t]=efiktoi_syn_t[i][t];
                }
                efiktoi_syn_t[i][pointer-1]=efiktoi_syn_t[i][pointer-1]-1;
                efiktoi_syn_t[i][pointer+1]=efiktoi_syn_t[i][pointer+1]+1;
                poib=0;
                metr=0;
                do{
                    meas=0;
                    for (t=2*k-2;t>=0;t--)
                    {
                        if (mallon_efiktoi_syn[t]==efiktoi_syn_t[metr][t])
                        {
                           meas=meas+1;
                        }
                    }
                    if (meas==2*k-1)
                    {
                       poib=poib+1;
                       break;
                    }
                    metr=metr+1;

                } while (metr<sum_efiktes);


                if ((poib!=0) && (o[ligo]!=0))
                {
                    //Efikti metavasi
                    metavasi[i][metr]=o[ligo];
                }
                /*else
                    Mi efikti metavasi */
            }
               pointer=pointer-2;
            km=km-1;
            ligo=ligo+2;

      } while (pointer>=0);
}
free(o);
free(f);
free(r);
free(m);
free(mallon_efiktoi_syn);

//Kataskevi tou pinaka A
a=(double*)malloc(sum_efiktes*sizeof(double));
for (i=0;i<sum_efiktes;i++)
{
   a[i]=0;
}
for (i=0;i<sum_efiktes;i++)
{
   for (j=0;j<sum_efiktes;j++)
   {
          a[i]=a[i]+metavasi[i][j];
   }
}
for (i=0;i<sum_efiktes;i++)
{
   a[i]=a[i]+epsilon;
}
palia_p=(double*)malloc(sum_efiktes*sizeof(double));
nea_p=(double*)malloc(sum_efiktes*sizeof(double));
```

```
//Arxikopoiisi tou pinaka Q (palia_p[i])
   for (i=0;i<sum_efiktes;i++)
   {
          palia_p[i]=one/sum_efiktes;
   }
   fores=0;

   //Methodos Jacobi over-relaxation
   do
   {
          fores=fores+1;
          for (j=0;j<sum_efiktes;j++)
          {
                 nea_p[j]=0;
          }
          for (i=0;i<sum_efiktes;i++)
          {
                 for (j=0;j<sum_efiktes;j++)
                 {
                        nea_p[j]=nea_p[j]+palia_p[i]*metavasi[i][j];
                 }
          }
          for (j=0;j<sum_efiktes;j++)
          {
                 nea_p[j]=nea_p[j]+palia_p[j]*epsilon;
          }
          for (j=0;j<sum_efiktes;j++)
          {
                 nea_p[j]=nea_p[j]/a[j];
          }
          sump=0;
          for (i=0;i<sum_efiktes;i++)
          {
                        sump=sump+nea_p[i];
          }
          for (i=0;i<sum_efiktes;i++)
          {
                        nea_p[i]=nea_p[i]/sump;
          }

          //Elegxos sygklisis
          max_apokl=0;
          for (i=0;i<sum_efiktes;i++)
          {
                 if (palia_p[i]>0)
                 {
                        apokl=fabs(palia_p[i]-nea_p[i])/palia_p[i];
                 }
                 else
                 {
                        apokl=sum_efiktes*fabs(palia_p[i]-nea_p[i]);
                 }
                 if (apokl>max_apokl)
                 {
                        max_apokl=apokl;
                 }
                 palia_p[i]=nea_p[i];
          }
} while (max_apokl>sfalma);
time(&end);
dif = difftime (end,start);
sum_d=0;
for (i=0;i<sum_efiktes;i++)
{
   if ((efiktoi_syn_t[i][0]==1) && (efiktoi_syn_t[i][1]>0))
   {
              sum_d=sum_d+palia_p[i];
   }
```

```
    }
    fout = fopen("apotelesmata.txt", "w");
    for (i=0;i<sum_efiktes;i++)
    {
      fprintf(fout, "y=%ld\n",efiktes[i]);
      for (t=2*k-2;t>=0;t--)
      {
                  fprintf(fout, "%ld",efiktoi_syn_t[i][t]);
      }
      fprintf(fout, "\n");
      fprintf(fout, "P(y=%ld)=%lf\n",efiktes[i],palia_p[i]);
      fprintf(fout, "\n");
    }
    th=sum_d*o[2*k-2];  //ypologismos TH
    fprintf(fout, "\n");
    fprintf(fout, "\n");
    fprintf(fout, "TH=%lf\n",th);
    nb=(double*)malloc((2*k-1)*sizeof(double));
    for (i=2*k-2;i>=0;i--)
    {
      nb[i]=0;
    }
    apoth=k;
    sumn=0;
    for (i=2*k-3;i>0;i=i-2)
    {
      sum_nea_p=0;
      apoth=apoth-1;
      for (ipi=0;ipi<sum_efiktes;ipi++)
      {
            nb[i]=nb[i]+efiktoi_syn_t[ipi][i]*palia_p[ipi];
      }
      sumn=sumn+nb[i];
      fprintf(fout, "n(%ld)=%lf\n",apoth,nb[i]);  //Ypologismos Bi
    }
    fprintf(fout, "n=%lf\n",sumn);  //Ypologismos N
    fprintf(fout, "epanalipseis:%d\n",fores);
    fprintf(fout, "xronos:%lf\n",dif);
    fclose(fout);
    free(efiktes);
    for (t=0;t<sum_efiktes;t++)
    {
      free(efiktoi_syn_t[t]);
    }
    free(efiktoi_syn_t);
    free(palia_p);
    free(nea_p);
    free(a);
    free(nb);
    for (i=0;i<sum_efiktes;i++)
    {
      free(metavasi[i]);
    }
    free(metavasi);
}
```

Κώδικας του αλγορίθμου με τη μέθοδο Gauss-Seidel Over-Relaxation

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <string.h>

time_t start,end;
double dif;
int *w, *m, *x,*ci,y;
int k;
int n;
double *o,*f,*r;
double epsilon, sfalma;

int decodeb();
void init();


int main()
{
      int i;
      FILE* file_out;

      init();
      free(ci);
      time(&start);

      w[0]=1;
      for(i=1;i<2*k;i++)
      {
         w[i]=w[i-1]*(m[i-1]+1);
      }
      n=w[2*k-2]*(m[2*k-2]+1);
      decodeb();

      printf("\n");
      system("PAUSE");
}

//Sinartisi gia to diavasma twn dedomenwn apo ena arxeio txt.
//Ta dedomena pou dinei o xristis einai: o arithmos twn apothikwn k,
//oi rithmoi paragwgis,vlavis kai episkevis twn mixanwn
//kai i xwritikotita twn apothikwn
void init()
{
    FILE* file_in;
    char c;
    int i;
    file_in = fopen("dedomena.txt", "r");

    while(1)
    {
        fscanf(file_in, "%c", &c);
        if(c<'0' || c>'9')
        {
            while(c != '\n')
            {
                fscanf(file_in, "%c", &c);
            }
```

```
    }
    else
    {
        ungetc(c, file_in);
        break;
    }
}

fscanf(file_in, "%d",&k);

m=(int*)malloc((2*k-1)*sizeof(int));
ci=(int*)malloc((2*k-1)*sizeof(int));
x=(int*)malloc((2*k-1)*sizeof(int));
w=(int*)malloc((2*k-1)*sizeof(int));
o=(double*)malloc((2*k-1)*sizeof(double)); // o=rithmos paragwgis
f=(double*)malloc((2*k-1)*sizeof(double)); // f=rithmos vlavis
r=(double*)malloc((2*k-1)*sizeof(double)); //r=rithmos episkevis

while(1)
{
    fscanf(file_in, "%c", &c);
    if(c<'0' || c>'9')
    {
        while(c != '\n')
        {
            fscanf(file_in, "%c", &c);
        }
    }
    else
    {
        ungetc(c, file_in);
        break;
    }
}

for (i=0;i<2*k-1;i=i+2)
{
    fscanf(file_in, "%lf",&o[i]);
    fscanf(file_in, "%lf",&f[i]);
    fscanf(file_in, "%lf",&r[i]);
}

while(1)
{
    fscanf(file_in, "%c", &c);
    if(c<'0' || c>'9')
    {
        while(c != '\n')
        {
            fscanf(file_in, "%c", &c);
        }
    }
    else
    {
        ungetc(c, file_in);
        break;
    }
}
for (i=1;i<2*k-1;i=i+2)
{
   fscanf(file_in, "%d",&ci[i]);
   m[i]=ci[i]+2;
}

for(i=0; i<2*k; i=i+2)
       m[i]=1;
```

```
    fclose(file_in);

    file_in = fopen("apoklisi.txt", "r");
    while(1)
    {
        fscanf(file_in, "%c", &c);
        if(c<'0' || c>'9')
        {
            while(c != '\n')
            {
                fscanf(file_in, "%c", &c);
            }
        }
        else
        {
            ungetc(c, file_in);
            break;
        }
    }
    fscanf(file_in, "%lf", &sfalma);
    fscanf(file_in, "%lf", &epsilon);
    fclose(file_in);
}

int decodeb()
{
    int fores,t,u,v,g,q,pointer,km,metr,ip,mix,mix_b,meas,po,poi;
    int poib,re,rv,rp,pp,ipi,pros,prosb,prosc,yp,prosw,pr,apoth,ligo;
     int sum_efiktes,i,j,yrem,*efiktes,**efiktoi_syn_t;
     int *mallon_efiktoi_syn,*sum,**efiktoi_syn;
     double old_p,*p,**metavasi,max_apokl,apokl,one,sum_nea_p
     double *a,sump,gi,th,*nb,sumn,sum_d,sumpb,tempp,dif;
     FILE* fout;

     sum=(int*)malloc(n*sizeof(int));
     efiktoi_syn=(int**)malloc(n*sizeof(int*));
     for (t=0;t<n;t++)
     {
         efiktoi_syn[t]=(int*)malloc((2*k-1)*sizeof(int));
     }
     sum_efiktes=0;
     //apokwdikopoiisi
     for (y=0;y<n;y++)
     {
        yrem=y;
        for(i=2*k-2;i>=0;i--)
        {
            x[i]=yrem/w[i];
            yrem=yrem-x[i]*w[i];
        }
        //evresi efiktwn kai mi efiktwn katastasewn
        u=0;
        v=0;
        g=0;
        for (i=2*k-2;i>=0;i=i-2)
        {
           if ((i<2*k-2) && (i>0))
           {
              if ((x[i+1]==m[i+1]) && (x[i-1]==m[i-1]))
              {
                 u=u+1;
              }
              if (x[i-1]==m[i-1])
              {

                 if (x[i]==0)
                 {
                    u=u+1;
```

```
            }
         }
         if (x[i+1]==0)
         {

            if (x[i]==0)
            {
               u=u+1;
            }
         }
         if (x[i]==0)
         {
            if (f[i]==0)
            {
               u=u+1;
            }
         }
         if ((x[i]==0) && (x[i-1]==m[i-1]-1) && (x[i-3]==m[i-3]))
         {
               u=u+1;
          }
      }
      if (i==2*k-2)
      {
         if ((x[i-1]==m[i-1]) && (x[i]==0))
         {
            u=u+1;
         }
         if ((x[i-1]==m[i-1]) && (x[i-3]==m[i-3]))
         {
            u=u+1;
         }
         if ((x[i]==0) && (f[i]==0))
         {
            u=u+1;
         }
         if ((x[i]==0) && (x[i-1]==m[i-1]-1) && (x[i-3]==m[i-3]))
         {
               u=u+1;
          }
      }
      if (i==0)
      {
         if ((x[i]==0) && (x[i+1]==0))
         {
            u=u+1;
         }
         if ((x[i]==0) && (f[i]==0))
         {
            u=u+1;
         }
      }
   }
   /*if (u!=0)
      Mi efikti katastasi*/
   if (u==0)
   {
      //Efikti katastasi
      sum[sum_efiktes]=y;
      for (i=2*k-2;i>=0;i--)
      {
          efiktoi_syn[sum_efiktes][i]=x[i];
      }
      sum_efiktes=sum_efiktes+1;
   }
}
free(x);
free(w);
```

```
efiktes=(int*)malloc(sum_efiktes*sizeof(int));
efiktoi_syn_t=(int**)malloc(sum_efiktes*sizeof(int*));
for (i=0;i<sum_efiktes;i++)
{
    efiktoi_syn_t[i]=(int*)malloc((2*k-1)*sizeof(int));
}
for (i=0;i<sum_efiktes;i++)
{
  for (t=2*k-2;t>=0;t--)
  {
        efiktoi_syn_t[i][t]=efiktoi_syn[i][t];
  }
}
for (i=0;i<n;i++)
{
  free(efiktoi_syn[i]);
}
free(efiktoi_syn);
for (i=0;i<sum_efiktes;i++)
{
  efiktes[i]=sum[i];
}
free(sum);
mallon_efiktoi_syn=(int*)malloc((2*k-1)*sizeof(int));
one=1;
metavasi=(double**)malloc(sum_efiktes*sizeof(double*));
for (i=0;i<sum_efiktes;i++)
{
  metavasi[i]=(double*)malloc(sum_efiktes*sizeof(double));
}
//evresi efiktwn metavasewn kai rithmwn metavasis
for (i=0;i<sum_efiktes;i++)
{

    pointer=2*k-2;
    km=k;
    ligo=0;

    do{


         if (efiktoi_syn_t[i][pointer]==0)
         {
            //EPISKEVI
            efiktoi_syn_t[i][pointer]=1;
            for (t=2*k-2;t>=0;t--)
            {
               mallon_efiktoi_syn[t]=efiktoi_syn_t[i][t];
            }
            efiktoi_syn_t[i][pointer]=0;
            metr=0;
            poi=0;
            do{
                po=0;
                for (t=2*k-2;t>=0;t--)
                {
                    if (mallon_efiktoi_syn[t]==efiktoi_syn_t[metr][t])
                    {
                       po=po+1;
                    }
                }
                if (po==2*k-1)
                {
                   poi=poi+1;
                   break;
                }
                metr=metr+1;
```

```
   } while (metr<sum_efiktes);
   if ((poi!=0) && (r[ligo]!=0))
   {
       //Efikti metavasi
       metavasi[i][metr]=r[ligo];
   }
}
if (efiktoi_syn_t[i][pointer]==1)
{

   //VLAVI
   efiktoi_syn_t[i][pointer]=0;
   for (t=2*k-2;t>=0;t--)
   {
      mallon_efiktoi_syn[t]=efiktoi_syn_t[i][t];
   }
   efiktoi_syn_t[i][pointer]=1;
   metr=0;
   poi=0;
   do{
       po=0;
       for (t=2*k-2;t>=0;t--)
       {
           if (mallon_efiktoi_syn[t]==efiktoi_syn_t[metr][t])
           {
              po=po+1;
           }

       }
       if (po==2*k-1)
       {
          poi=poi+1;
          break;
       }
       metr=metr+1;

   } while (metr<sum_efiktes);
   if ((poi!=0) && (f[ligo]!=0))
   {
       //Efikti metavasi
       metavasi[i][metr]=f[ligo];
   }
   /*else
      Mi efikti metavasi*/


   //PARAGWGI
   if (pointer!=0)
   {
      efiktoi_syn_t[i][pointer-1]=efiktoi_syn_t[i][pointer-1]+1;
   }
   else if (pointer==0)
   {
       efiktoi_syn_t[i][pointer-1]=0;
   }
   if (pointer!=2*k-2)
   {
      efiktoi_syn_t[i][pointer+1]=efiktoi_syn_t[i][pointer+1]-1;
   }
   else if (pointer==2*k-2)
   {
        efiktoi_syn_t[i][pointer+1]=m[pointer-1];
   }
   for (t=2*k-2;t>=0;t--)
   {
       mallon_efiktoi_syn[t]=efiktoi_syn_t[i][t];
   }
   efiktoi_syn_t[i][pointer-1]=efiktoi_syn_t[i][pointer-1]-1;
```

```
            efiktoi_syn_t[i][pointer+1]=efiktoi_syn_t[i][pointer+1]+1;
            poib=0;
            metr=0;
            do{
                meas=0;
                for (t=2*k-2;t>=0;t--)
                {
                    if (mallon_efiktoi_syn[t]==efiktoi_syn_t[metr][t])
                    {
                       meas=meas+1;
                    }
                }
                if (meas==2*k-1)
                {
                   poib=poib+1;
                   break;
                }
                metr=metr+1;

             } while (metr<sum_efiktes);


             if ((poib!=0) && (o[ligo]!=0))
             {
                //Efikti metavasi
                metavasi[i][metr]=o[ligo];
             }
             /*else
                      Mi efikti metavasi*/

            pointer=pointer-2;
             km=km-1;
             ligo=ligo+2;

      } while (pointer>=0);
}
free(o);
free(f);
free(r);
free(m);
free(mallon_efiktoi_syn);
//Kataskevi tou pinaka A
a=(double*)malloc(sum_efiktes*sizeof(double));
for (i=0;i<sum_efiktes;i++)
{
   a[i]=0;
}
for (i=0;i<sum_efiktes;i++)
{
    for (j=0;j<sum_efiktes;j++)
    {
          a[i]=a[i]+metavasi[i][j];
    }
 }
 for (i=0;i<sum_efiktes;i++)
 {
   a[i]=a[i]+epsilon;
 }
 p=(double*)malloc(sum_efiktes*sizeof(double));
 //Arxikopoiisi tou pinaka p[i]
 for (i=0;i<sum_efiktes;i++)
 {
   p[i]=one/sum_efiktes;
 }
 fores=0;
 //Methodos Gauss-Seidel over-relaxation
 do
 {
```

```
	fores=fores+1;
	max_apokl=0;
	sump=0;
	for (j=0;j<sum_efiktes;j++)
	{
		old_p=p[j];
		p[j]=epsilon*p[j];
		for (i=0;i<sum_efiktes;i++)
		{
			p[j]=p[j]+p[i]*metavasi[i][j];
		}
		p[j]=p[j]/a[j];
		sump=sump+p[j];
		//Elegxos sygklisis
		if (old_p>0)
		{
			apokl=fabs(old_p-p[j])/old_p;
		}
		else
		{
			apokl=fabs(old_p-p[j])*sum_efiktes;
		}
		if (apokl>max_apokl)
		{
			max_apokl=apokl;
		}
	}
	for (j=0;j<sum_efiktes;j++)
	{
		p[j]=p[j]/sump;
	}
} while (max_apokl>sfalma);
time(&end);
dif = difftime (end,start);
sum_d=0;
for (i=0;i<sum_efiktes;i++)
{
	if ((efiktoi_syn_t[i][0]==1) && (efiktoi_syn_t[i][1]>0))
	{
			sum_d=sum_d+p[i];
	}
}
fout = fopen("apotelesmata.txt", "w");
for (i=0;i<sum_efiktes;i++)
{
	fprintf(fout, "y=%ld\n",efiktes[i]);
	for (t=2*k-2;t>=0;t--)
	{
		fprintf(fout, "%ld",efiktoi_syn_t[i][t]);
	}
	fprintf(fout, "\n");
	fprintf(fout, "P(y=%ld)=%lf\n",efiktes[i],p[i]);
	fprintf(fout, "\n");
}
th=sum_d*o[2*k-2];  //ypologismos TH
fprintf(fout, "\n");
fprintf(fout, "\n");
fprintf(fout, "TH=%lf\n",th);
nb=(double*)malloc((2*k-1)*sizeof(double));
for (i=2*k-2;i>=0;i--)
{
	nb[i]=0;
}
apoth=k;
sumn=0;
for (i=2*k-3;i>0;i=i-2)
{
	sum_nea_p=0;
```

```
			apoth=apoth-1;
			for (ipi=0;ipi<sum_efiktes;ipi++)
			{
				nb[i]=nb[i]+efiktoi_syn_t[ipi][i]*p[ipi];
			}
			sumn=sumn+nb[i];
			fprintf(fout, "n(%ld)=%lf\n",apoth,nb[i]);  //Ypologismos Bi
		}
		fprintf(fout, "n=%lf\n",sumn);  //Ypologismos N
		fprintf(fout, "epanalipseis:%d\n",fores);
		fprintf(fout, "xronos:%lf\n",dif);
		fclose(fout);
		free(efiktes);
		for (t=0;t<sum_efiktes;t++)
		{
			free(efiktoi_syn_t[t]);
		}
		free(efiktoi_syn_t);
		free(p);
		free(a);
		free(nb);
		for (i=0;i<sum_efiktes;i++)
		{
			free(metavasi[i]);
		}
		free(metavasi);
}
```