Λειτουργικά Συστήματα
Project 2

Τσόγκας Παναγιώτης, Νίκας Ιωάννης Ιάσων & Τσιαντός Δημήτριος

Απρίλιος 2025

Περιεχόμενα

1 Ο αλγόριθμος **3**
1.1 Shortest Job First 3
1.2 Παραλλαγή Shortest Job First 4

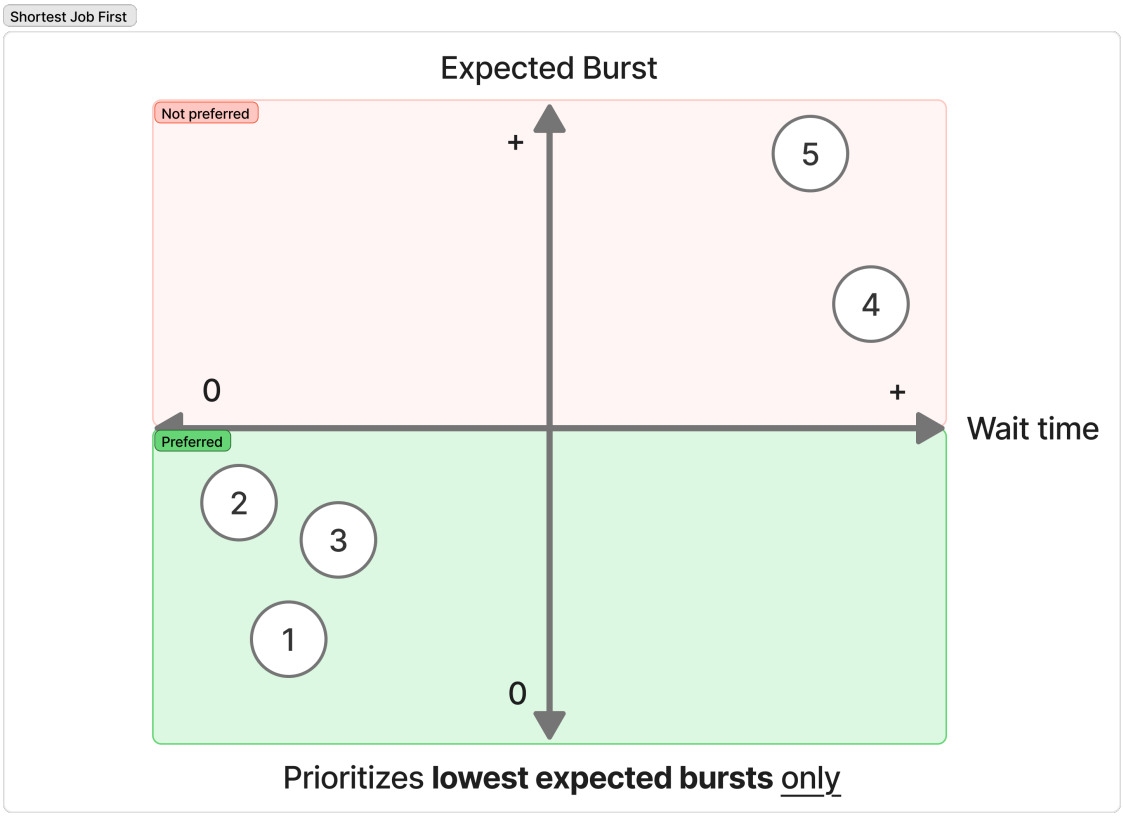
2 Profiling **5**
2.1 Πρώτη δοκιμή: 1 Non-Interactive &25 Interactive 5
2.2 Δεύτερη δοκιμή: 5 Non-Interactive &5 Interactive 9
2.3 Τρίτη δοκιμή: 4 Non-Interactive 10
2.4 Τέταρτη δοκιμή: 4 Interactive 11

3 Συμπεράσματα **11**

1 Ο αλγόριθμος

1.1 Shortest Job First

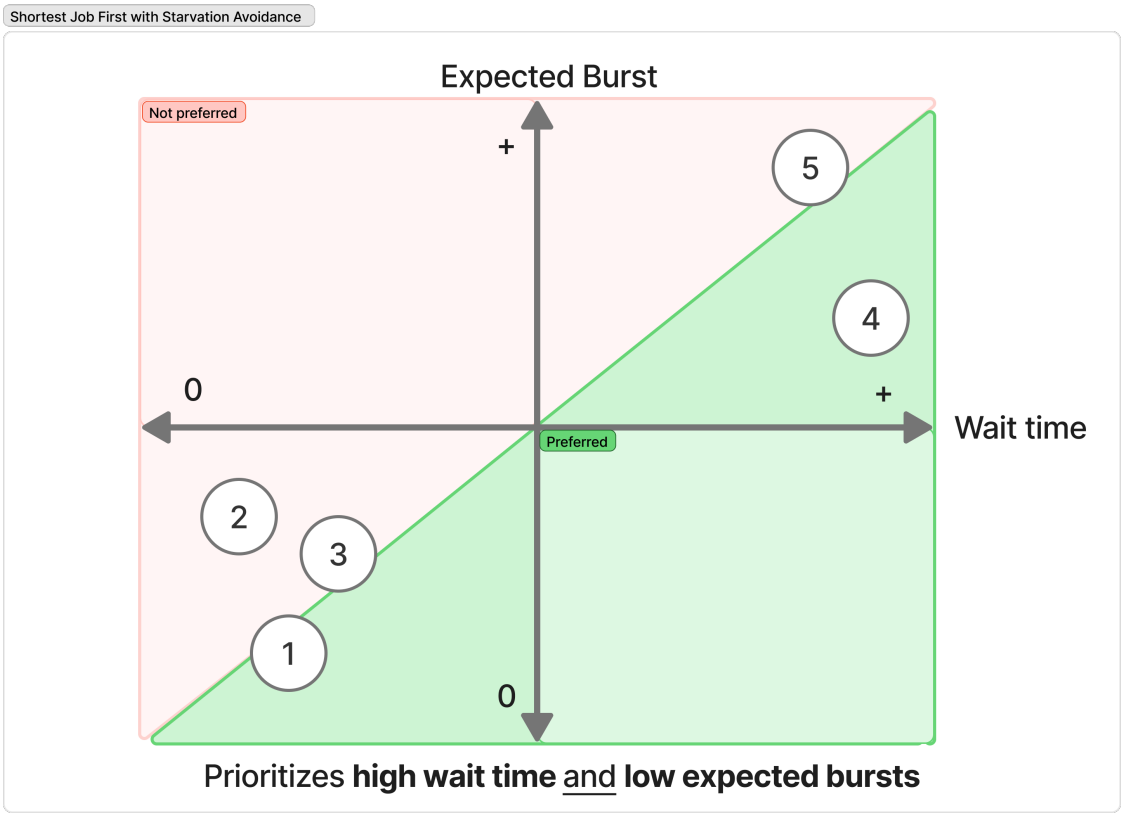
Ο αλγόριθμος χρονοδρομολόγησης Shortest Job First (SJF) επιλέγει τη διεργασία με τη μικρότερη προβλεπόμενη διάρκεια εκτέλεσης CPU – expected burst time. Ωστόσο, η προσέγγιση αυτή μπορεί να οδηγήσει στο πρόβλημα της λιμοκτονίας για διεργασίες με μεγάλη διάρκεια εκτέλεσης, όταν υπάρχει συνεχής ροή μικρών διεργασιών στο σύστημα, όπως αποτυπώνεται στο σχήμα 3.



Σχήμα 1: Γράφημα χρόνου αναμονής και αναμενόμενου χρόνου εκτέλεσης της κάθε διεργασίας. Ο αλγόριθμος Shortest Job First κρίνει μόνο με βάση τον αναμενόμενο χρόνο εκτέλεσης, άρα οι διεργασίες 1, 2 και 3 θα προτιμούνται συχνότερα, ενώ οι 4 και 5 θα λιμοκτωνούν.

1.2 Παραλλαγή Shortest Job First

Προκειμένου να λύσουμε το πρόβλημα αυτό, λαμβάνουμε υπ' όψιν και τον χρόνο αναμονής της κάθε διεργασίας στην ουρά έτοιμων διεργασιών, έτσι ώστε αν κάποια διεργασία περιμένει πολλή ώρα, να της δοθεί ο επεξεργαστής. Θα αναφερόμαστε σε αυτή την νέα παραλλαγή του αλγορίθμου ως Shortest Job First με αποφυγή λιμοκτονίας.

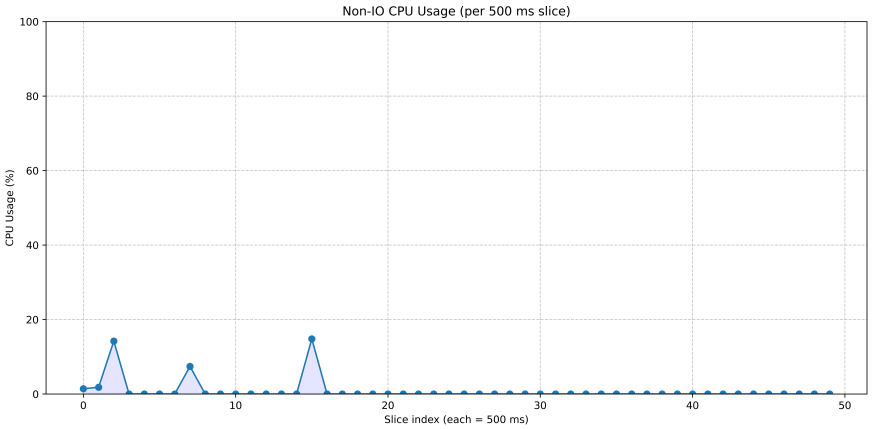


Σχήμα 2: Οι διεγρασίες 1 και 4 θα εκτελεστούν πρώτες. Η 4 έχει μεγάλο χρόνο αναμονής, άρα κινδυνεύει να λιμοκτονήσει και η 1 έχει πολύ μικρό expected burst και αξίζει η επιλογή της. Ο αλγόριθμος προτιμάει την διεργασία που απέχει την μεγαλύτερη απόσταση από το κέντρο και βρίσκεται στο πράσινο κομμάτι, αφού δίνουμε το ίδιο βάρος στις δύο παραμέτρους.

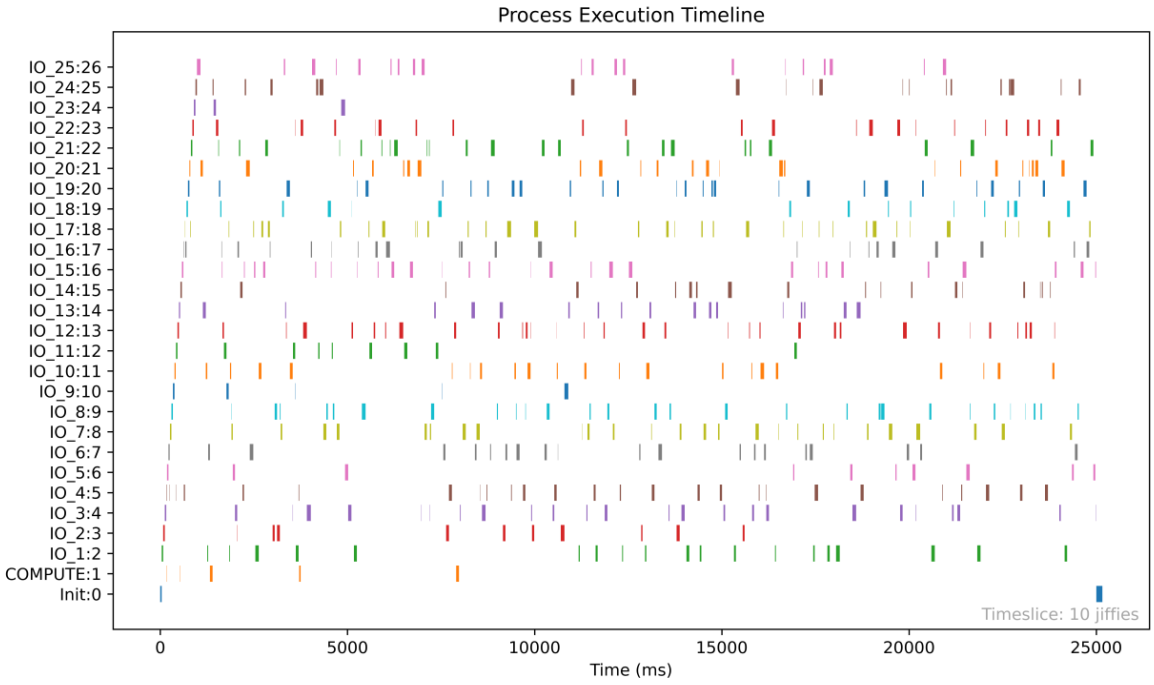
2 Profiling

2.1 Πρώτη δοκιμή: 1 Non-Interactive & 25 Interactive

Για να αναδειχθεί η λιμοκτονία των διεργασιών στο σύστημα με τον SJF, ως πρώτη δοκιμή επιλέγεται μία εκτέλεση με **ένα non interactive process και πολλά interactive**.



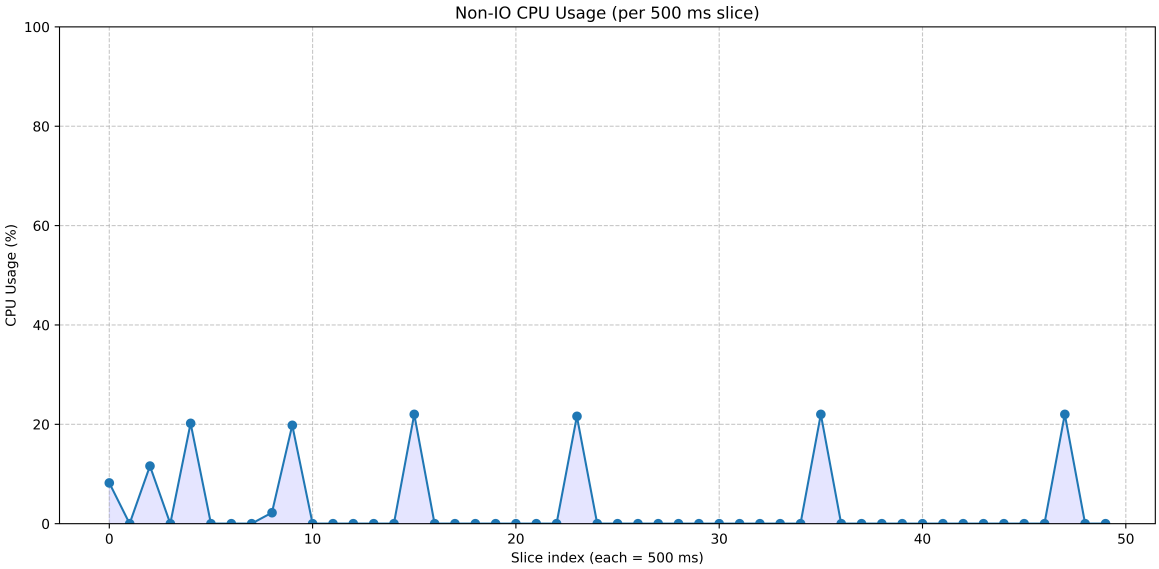
Σχήμα 3: Ποσοστό χρήσης του επεξεργαστή από την non interactive διεργασία όταν τρέχουν και άλλες 25 interactive διεργασίες. Η non interactive λιμοκτονεί.



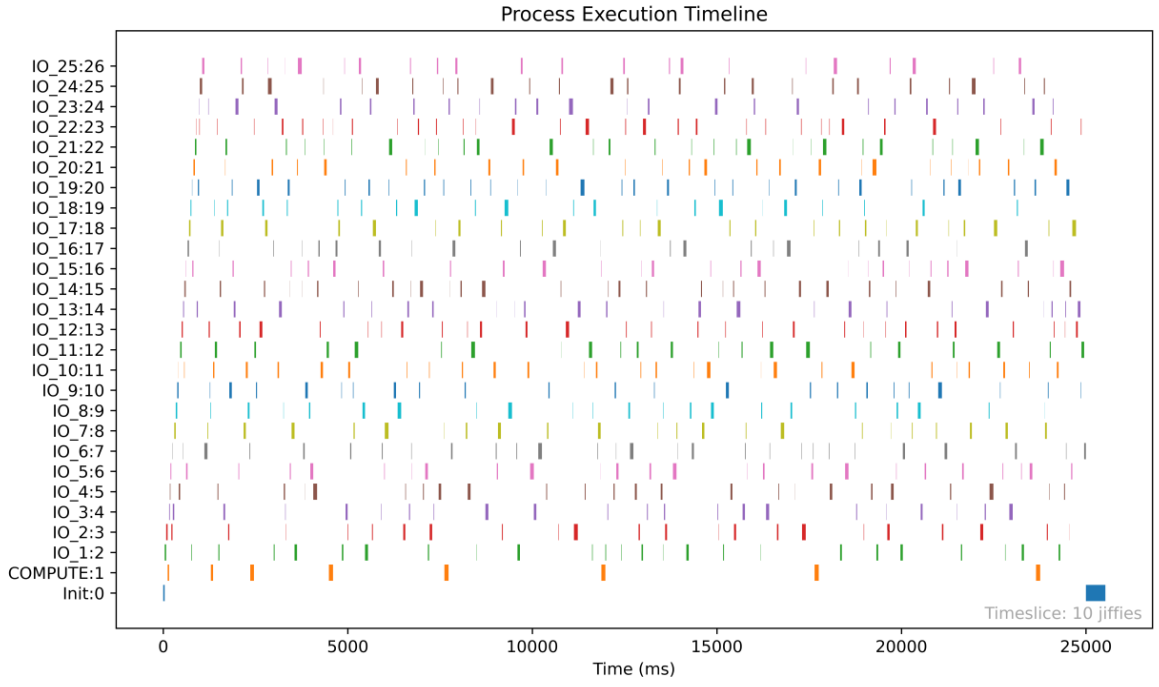
Σχήμα 4: Το γράφημα Gantt εκτέλεσης 25 interactive και μίας non-interactive διεργασίας (COMPUTE:1 - με πορτοκαλί χρώμα).

Να σημειωθεί πως ύστερα από δοκιμές παρατηρείται λιμοκτονία στην διεργασία 'Compute' όταν υπάρχουν περίπου άλλες 20 διαδραστικές διεργασίες. Τα νούμερα μπορεί να διαφέρουν από παράγοντες όπως μέγεθος timeslice και τυχαιότητα των interactive διεργασιών.

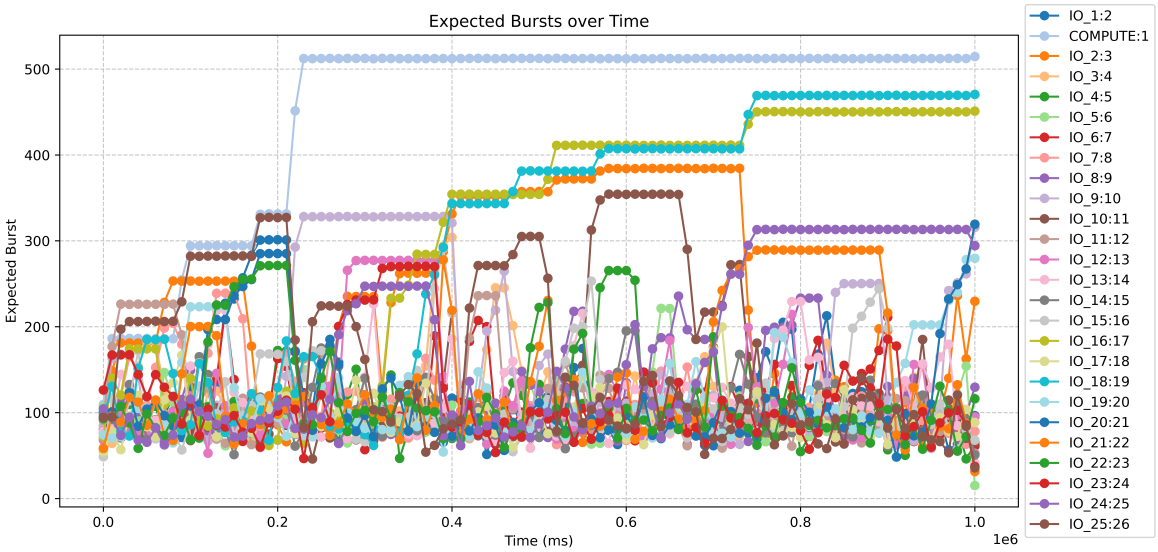
Σε αντίθεση με τον απλό SJF, παρατηρούμε πως στην παραλλαγή με αποφυγή λιμοκτονίας η non-interactive διεργασία εκτελείται για περισσότερο χρόνο καθώς περιμένει πολλή ώρα χωρίς τον επεξεργαστή και τελικά επιλέγεται (Σχήματα 5 και 6).



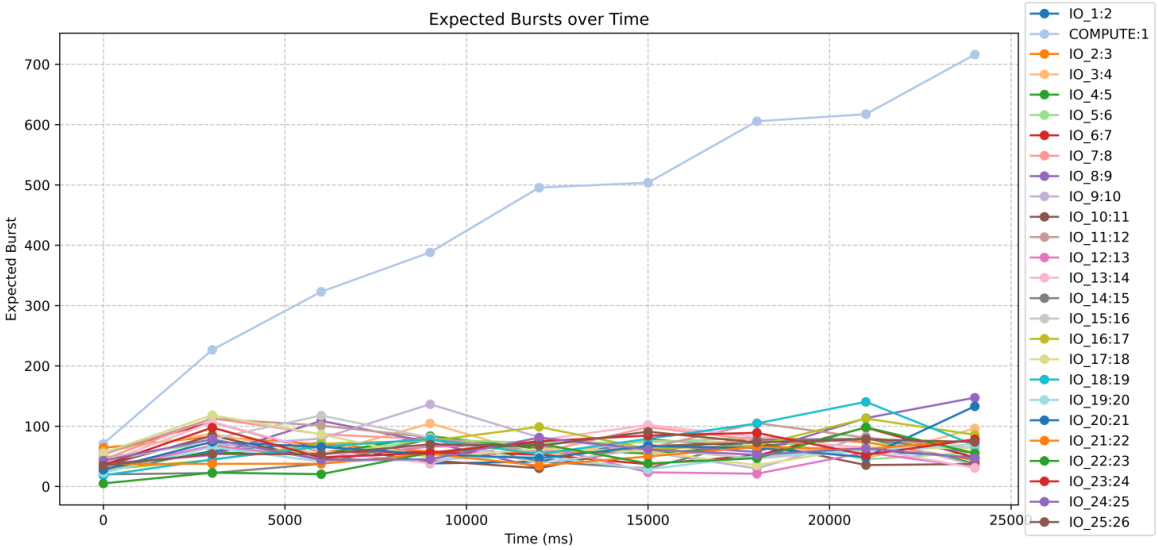
Σχήμα 5: Ποσοστό χρήσης του επεξεργαστή από την non interactive. Η non interactive δεν λιμοκτωνεί με τον νέο αλγόριθμο.



Σχήμα 6: Gantt chart εκτέλεσης παραλλαγής του Shortest Job First αλγορίθμου με 25 interactive και ένα non interactive process



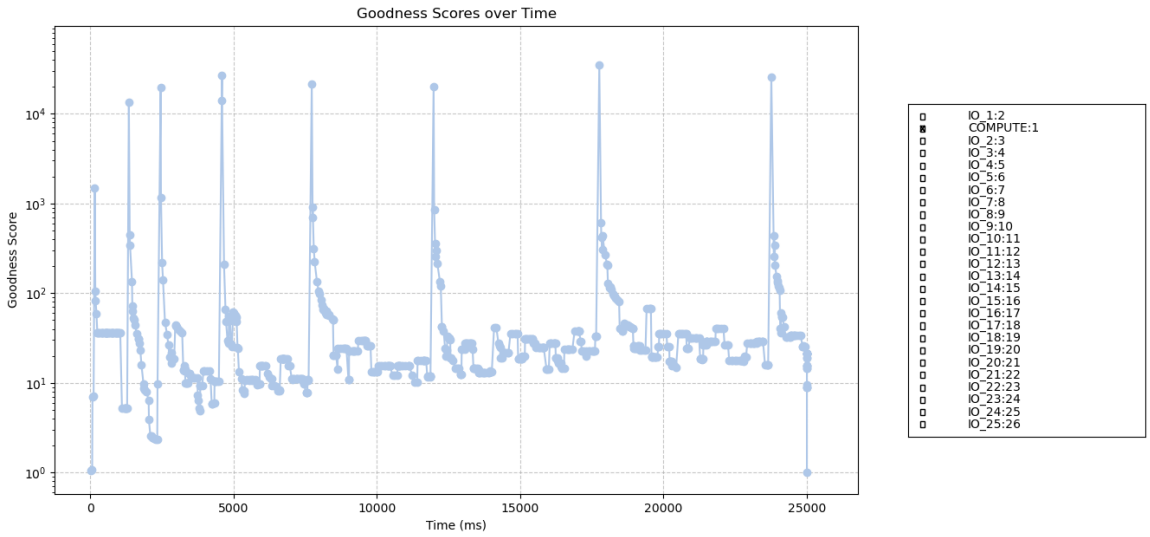
Σχήμα 7: Αναμενόμενος χρόνος εκτέλεσης της κάθε διεργασίας στον χρόνο σε απλό SJF. Η διεργασία Compute εκτελείται στην αρχή και φτάνει σε μία μεγάλη τιμή.



Σχήμα 8: Αναμενόμενος χρόνος εκτέλεσης της κάθε διεργασίας στον χρόνο στην παραλλαγή του SJF (σε intervals των 3000ms με χρήση μέσου όρου).

Από το Σχήμα 7 παρατηρούμε ότι στον κλασικό αλγόριθμο η αναμενόμενη διάρκεια εκτέλεσης (Expected Burst) της διεργασίας υπολογισμού 'Compute' αυξάνεται προοδευτικά, καθώς οι τιμές των υπόλοιπων διεργασιών ανανεώνονται σε κάθε κύκλο αναμονής (sleep). Ως αποτέλεσμα, όσο περνά ο χρόνος, η διεργασία υπολογισμού καταλαμβάνει ολοένα και λιγότερο χρόνο στη CPU.

Η σημαντική διαφορά που παρατηρείται μεταξύ των σχημάτων 7 και 8 για την πρόβλεψη της διεργασίας 'Compute', οφείλεται στο γεγονός ότι, μετά από ένα συγκεκριμένο χρονικό σημείο λόγω της συχνότητας των καταστάσεων sleep, η διεργασία υπολογισμού διατηρεί σταθερά την υψηλότερη τιμή αναμενόμενου χρόνου εκτέλεσης. Αυτό έχει ως αποτέλεσμα να μην επιλέγεται ξανά από τον χρονοδρομολογητή, καθώς δεν έχει ποτέ τον μικρότερο αναμενόμενο χρόνο εκτέλεσης σε σχέση με τις υπόλοιπες διεργασίες. Αυτό το ενδεχόμενο αποτρέπεται στην χρονοδρομολόγηση με βάση το goodness score και έτσι η διεργασία έχει την ευκαιρία να αυξήσει την αναμενόμενη διάρκεια εκτέλεσης της.

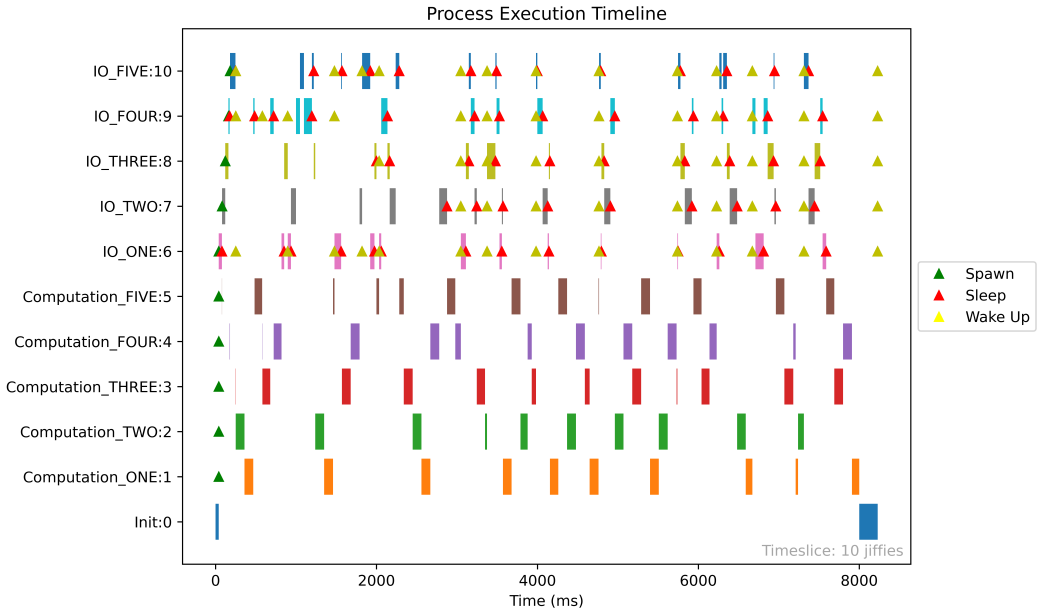


Με τη βοήθεια του παραπάνω σχήματος παρατηρούμε ότι, παρά τη συνεχή αύξηση του αναμενόμενου χρόνου εκτέλεσης της διεργασίας 'Compute', η τροποποιημένη εκδοχή του αλγορίθμου Shortest Job First, με την προσθήκη του παράγοντα WaitingInReadyQueue, οδηγεί σε διαφορετική συμπεριφορά κατά την επιλογή διεργασιών.

Συγκεκριμένα, η νέα συνθήκη επιλογής, μέσω του δείκτη goodness, δεν παραμένει σταθερά υψηλή για τη διεργασία υπολογισμού, αντιθέτως, σε ορισμένες περιπτώσεις μειώνεται, επιτρέποντάς της να αποκτήσει πρόσβαση στον επεξεργαστή. Αυτό δίνει στη διεργασία compute την ευκαιρία να εκτελεστεί, παρά την αρνητική επίδραση του αυξανόμενου expected burst, όπως φαίνεται στο Σχήμα 5.

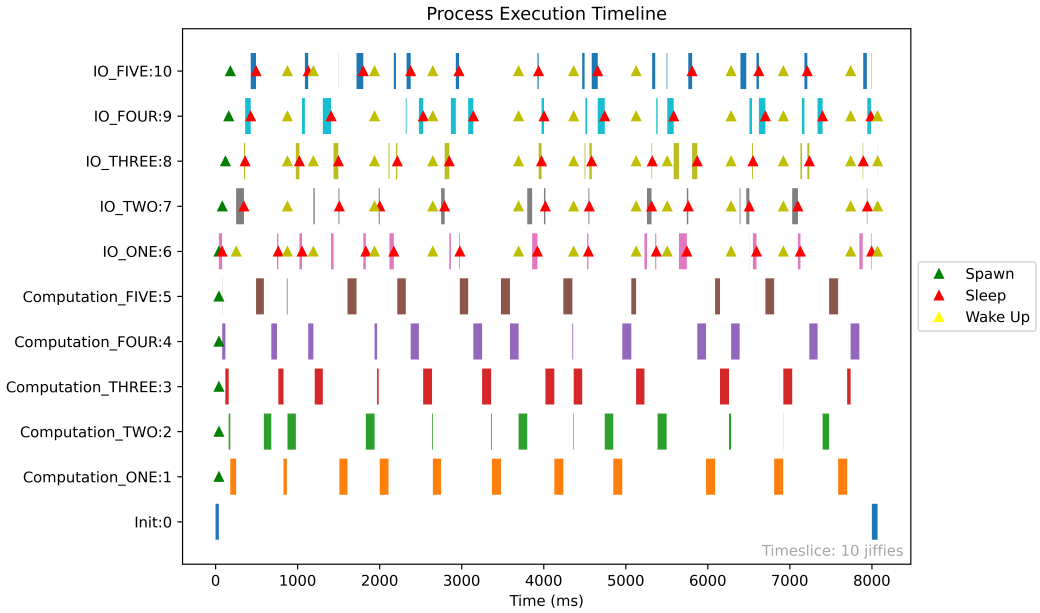
Επιπλέον, είναι εμφανή στο γράφημα τα σημεία όπου η διεργασία compute αποδεσμεύει την CPU, όπως υποδεικνύεται από τις αιχμές (spikes) στις τιμές του δείκτη.

2.2 Δεύτερη δοκιμή: 5 Non-Interactive & 5 Interactive



Σχήμα 9: Γράφημα Gantt εκτέλεσης SJF.

Στην συγκεκριμένη εκτέλεση στο Σχήμα 9 με τον κλασικό Shoretst Job First και config το οποίο έχει **πέντε Interactive και πέντε Non-Interactive** διεργασίες, παρατηρούμε ότι μετά απο λίγη ώρα οι Non-Interactive έχουν την μικρότερη προτεραιότητα και τρέχουν μόνο οταν δεν δεσμεύει κάποια διαδραστική διεργασία τον επεξεργαστή.

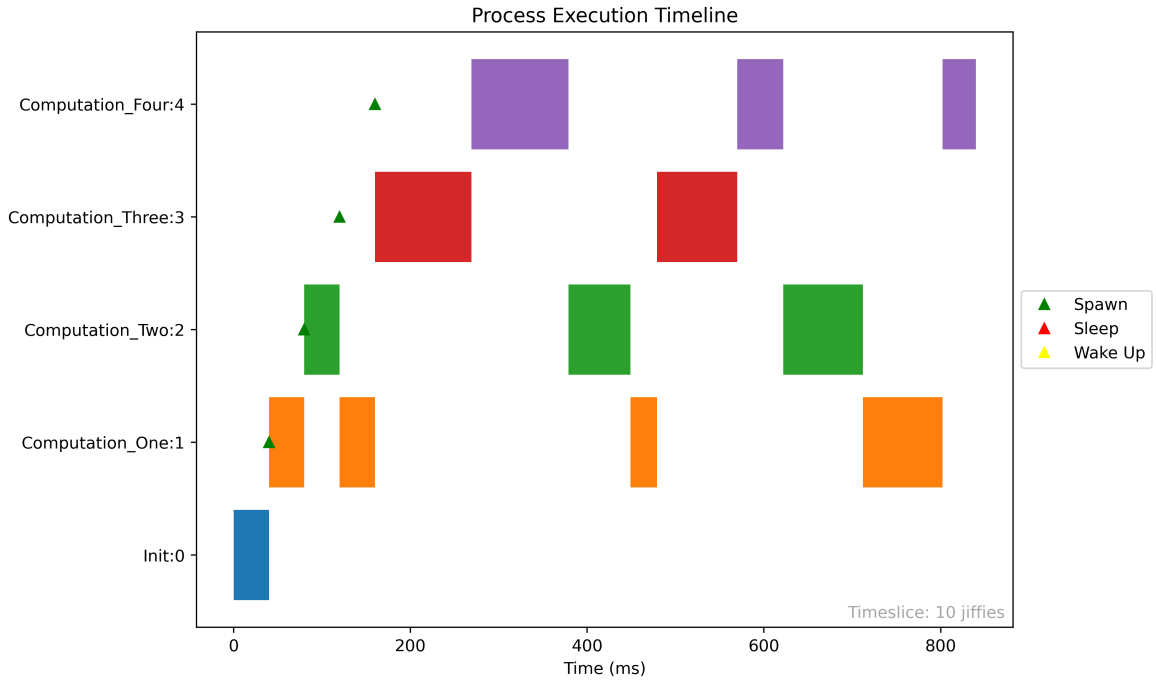


Σχήμα 10: Γράφημα Gantt εκτέλεσης SJF με αποφυγή λιμοκτονίας.

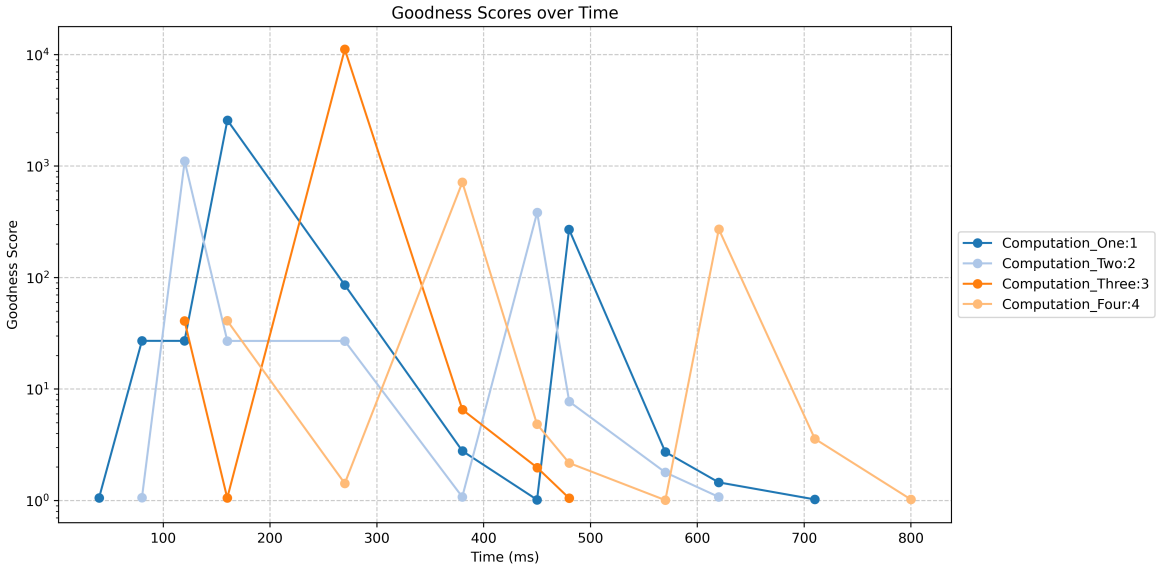
Στο ίδιο σενάριο με την χρήση της παραλλαγής του SJF Σχήμα 10, μπορούμε να δούμε ότι όλες οι διεργασίες έχουν ευκαιρία να τρέξουν οποιαδήποτε στιγμή.

2.3 Τρίτη δοκιμή: 4 Non-Interactive

Στην εκτέλεση με **μόνο τέσσερις μη διαδραστικές διεργασίες** παρατηρούμε ότι και οι δύο αλγόριθμοι ουσιαστικά συμπεριφέρονται σαν Round Robin, δίνοτας ο ένας στον άλλον την CPU ανά συγκεκριμένα χρονικά διαστήματα.

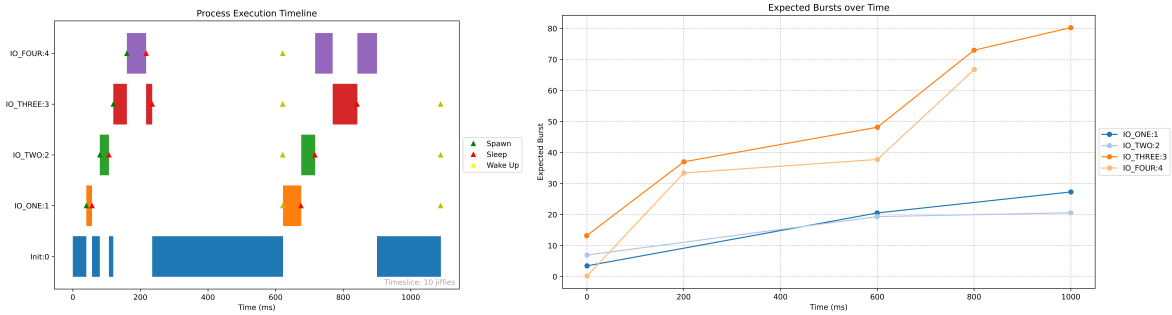


Σχήμα 11: Γράφημα Gantt εκτέλεσης παραλλαγής του αλγορίθμου μόνο για μη διαδραστικές διεργασίες.

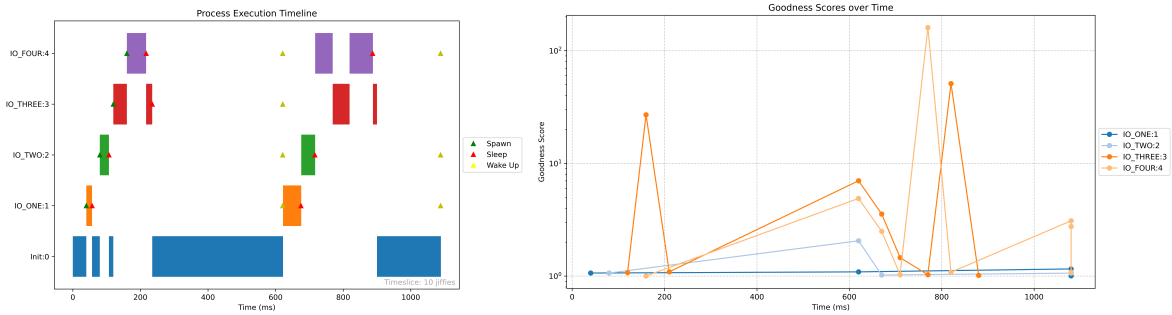


Σχήμα 12: Goodness score των τεσσάρων διεργασιών. Ο χρονοδρομολογητής επιλέγει την διεργασία με το μικρότερο goodness σε κάθε επανάληψη.

2.4 Τέταρτη δοκιμή: 4 Interactive



Σχήμα 13: Gantt chart και expected bursts σε απλό SJF.



Σχήμα 14: Gantt chart και Goodness scores στην παραλλαγή του SJF.

3 Συμπεράσματα

Ο αλγόριθμος SJF είναι αρκετά απλός στην υλοποίηση και παρέχει βέλτιστο μέσο χρόνο αναμονής, καθώς πάντα επιλέγει τη διεργασία με το μικρότερο αναμενόμενο χρόνο εκτέλεσης. Μπορεί, όμως, σε περίπτωση που έχουμε πολλές interactive διεργασίες να οδηγήσουν τις non-interactive σε λιμοκτονία.

Από την άλλη πλευρά, η παραλλαγή του αλγορίθμου, μπορεί να επιλύει το πρόβλημα της λιμοκτονίας αλλά επιβαρύνει επιπλέον το σύστημα με πράξεις σε κάθε κλήση του χρονοδρομολογητή. Πολλές φορές επιλέγονται, επίσης, non-interactive διεργασίες εις βάρος των interactive.

Η επιλογή του κατάλληλου αλγορίθμου θα πρέπει να γίνει με βάση την ανοχή στις καθυστερήσεις των διαδραστικών διεργασιών και τον αριθμό των διαδραστικών διεργασιών που αναμένεται να εκτελεστούν. Σε συστήματα με λίγες διαδραστικές διεργασίες δεν υπάρχει λιμοκτονία και ο χρόνος εκτέλεσης των υπόλοιπων διεργασιών δεν επηρεάζονται κατά πολύ, άρα η καλύτερη επιλογή είναι ο SJF. Σε συστήματα με πολλές διαδραστικές διεργασίες ο αλγόριθμος με αποφυγή λιμοκτονίας θα δίνει καλύτερο χρόνο εκτέλεσης στις μη διαδραστικές διεργασίες.