ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΠΑΡΑΓΩΓΗΣ

ΕΝΑΣ ΕΞΕΛΙΚΤΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ ΓΙΑ ΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΠΑΡΑΓΩΓΗΣ ΣΕ ΚΛΩΣΤΟΥΦΑΝΤΟΥΡΓΙΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΔΗΜΗΤΡΟΠΟΥΛΟΣ ΧΡΗΣΤΟΣ

ΕΠΙΒΛΕΠΩΝ: **ΒΑΣΙΛΗΣ Σ. ΚΟΥΙΚΟΓΛΟΥ**

ΧΑΝΙΑ, ΟΚΤΩΒΡΙΟΣ 2009

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1. Εισαγωγή

Στην παρούσα διπλωματική εργασία εξετάζουμε ένα πρόβλημα προγραμματισμού παραγωγής σε ένα εργοστάσιο. Ο προγραμματισμός αναφέρεται στο σχέδιο εκτέλεσης ενός πλήθους δραστηριοτήτων οι οποίες δεσμεύουν πόρους (χρήμα, χρόνο, μηχανές, συστήματα μεταφοράς, ανθρώπινο δυναμικό κλπ) με σκοπό τη βελτιστοποίηση κάποιων κριτηρίων (αύξηση παραγωγικότητας, έγκαιρη παράδοση, ελαχιστοποίηση κόστους παραγωγής). Ένα τέτοιο πρόβλημα είναι η εύρεση της σειράς εκτέλεσης παραγγελιών ώστε να ελαχιστοποιείται ο χρόνος παράδοσης της τελευταίας παραγγελίας.

Για συστήματα μέχρι δύο μηχανών η ακριβής λύση υπολογίζεται μέσω του αλγορίθμου Johnson. Γιά μεγαλύτερης πολυπλοκότητας συστήματα το πρόβλημα του προγραμματισμού περιπλέκεται λόγω του πλήθους των περιορισμών που συνδέουν πόρους και δραστηριότητες και η εύρεση ενός ολικού βέλτιστου είναι αδύνατη ή πολύ χρονοβόρα. Σε αυτές τις περιπτώσεις εφαρμόζουμε διάφορες μεθόδους βελτιστοποίησης προκειμένου να προσεγγίσουμε την ολική βέλτιστη λύση κάποιου κριτηρίου όπως:

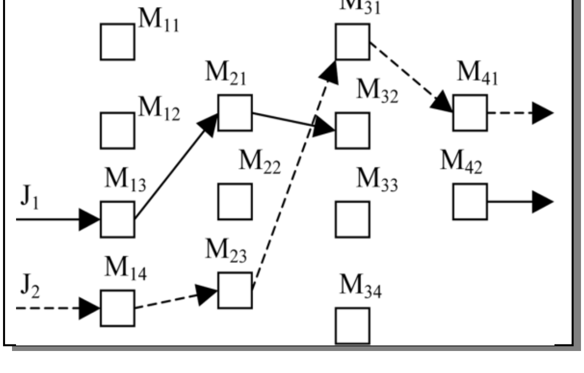
Αναζήτηση Tabu (Tabu Search)
Προσομοιωμένη Ανόπτηση (Simulated Annealing)
Δυναμικός Προγραμματισμός (Dynamic Programming)
Αλγόριθμος Διακλάδωσης και φράγματος (Branch and Bound)
Γενετικός Αλγόριθμος (Genetic Algorithm)

1.1 Αντικείμενο της διπλωματικής εργασίας

Εξετάζουμε ένα ευέλικτο κατάστημα ροής (Flexible Flow Shop). Ένα τέτοιο σύστημα παραγωγής περιγράφεται από τα κάτωθι στοιχεία:

Έστω ένα σύνολο N από εργασίες, N={1,2,3,…,n} οι οποίες πρέπει να περάσουν από m στάδια. Στο στάδιο i, i = 1, …, m, υπάρχει ένας αριθμός από παράλληλες μηχανές Mij = 1,…,mi, οι οποίες μπορούν να εκτελέσουν την συγκεκριμένη εργασία. Κάθε εργασία πρέπει να περάσει από όλα τα στάδια και σε κάθε στάδιο υφίσταται κατεργασία από μία μόνο μηχανή.

Εικόνα 1. Σχηματική περιγραφή ευέλικτου καταστήματος ροής



Ονομάζεται κατάστημα ροής διότι τα κομμάτια πρέπει να επισκεφθούν όλα τα στάδια της γραμμής παραγωγής και ευέλικτο επειδή σε κάθε στάδιο μπορούν να υπάρχουν παραπάνω από μία μηχανές. Το στοιχείο αυτό αποτρέπει το μπλοκάρισμα της παραγωγικής διαδικασίας και ταυτόχρονα αυξάνει το ρυθμό παραγωγής. Επιπλέον για κάθε εργασία ο χρόνος μιάς κατεργασίας εξαρτάται από τη μηχανή που την εκτελεί. Υπάρχουν επίσης χρόνοι προετοιμασίας (setup times) ανάλογα τη μηχανή και το κομμάτι που επεξεργάζεται κάθε φορά.

Στην εργασία αυτή μελετήθηκε ένα τμήμα εργοστασίου στο οποίο υπάρχουν δύο στάδια παραγωγής σε σειρά. Σκοπός είναι η εύρεση ενός προγράμματος παραγωγής (εκτέλεσης παραγγελιών) που να ελαχιστοποιεί το χρόνο περάτωσης της τελευταίας παραγγελίας.

Για την βελτιστοποίηση χρησιμοποιήθηκε ένας γενετικός αλγόριθμος.

Τέτοιου είδους συστήματα παραγωγής βρίσκουν εφαρμογή στις βιομηχανίες υφασμάτων [60], στην κεραμοποιία [62], στη δημιουργία ηλεκτρονικών πλακετών [59], [61], στη βιομηχανία κατασκευής λαμαρίνων, στην αυτοκινητοβιομηχανία, στην χαρτοβιομηχανία, καθώς και αλλού.

1.2 Επισκόπηση της βιβλιογραφίας

Ο χρονοπρογραμματισμός παραγωγικών διαδικασιών προσελκύει το ενδιαφέρον πολλών ερευνητών τα τελευταία πενήντα χρόνια. Πλήθος εργασιών που αναφέρονται τόσο σε δίκτυα παραγωγής ή καταστήματα εργασιών (jobshop) όσο και ειδικότερα σε γραμμές παραγωγής ή καταστήματα ροής (flowshop) έχουν γραφτεί κατά τη διάρκεια αυτών των χρόνων. Μέσα από διάφορες τεχνικές οι ερευνητές δίνουν απαντήσεις τόσο σε προβλήματα ανάθεσης αλλά και εύρεσης βέλτιστης αλληλουχίας εργασιών. Μία περιγραφή και ανάλυση τέτοιων προβλημάτων μπορεί να βρει κανείς στα βιβλία [1], [8], [24].

Πολλές μέθοδοι έχουν αναπτυχθεί προκειμένου να δώσουν απαντήσεις στα παραπάνω ερωτήματα τόσο για απλές αντικειμενικές συναρτήσεις όσο και για πολυκριτήριες συναρτήσεις [53].

Για καταστήματα δύο μηχανών, ή και τριών αλλά με ορισμένα χαρακτηριστικά, το θεώρημα του Johnson δίνει μία διαδικασία εύρεσης βέλτιστης λύσης. Μέθοδοι Ακέραιου Γραμμικού Προγραμματισμού έχουν δοκιμαστεί αλλά δεν είναι ιδιαίτερα αποτελεσματικές όταν αυξάνεται ο αριθμός των μηχανών [35].

Μια μέθοδος Διακλάδωσης και Φράγματος για καταστήματα ροής με εξαρτημένους χρόνους προετοιμασίας των μηχανών προτάθηκε από τους Rios Mercado και Bard [26]. Ενδιαφέρον παρουσιάζει και η εργασία των McKay κ.α [63] που αναφέρεται σε περιπτώσεις πραγματικών συστημάτων παραγωγής. Εκεί θα δούμε ότι πολλές τεχνικές που έχουν εφαρμοστεί σε θεωρητικά μοντέλα αποτυγχάνουν να βρουν μια καλή λύση σε πραγματικά συστήματα. Το γεγονός αυτό είχε επισημανθεί και πιό πριν από τους Ledbetter και Cox [58].

Ανάλογα με την αποτελεσματικότητα κάθε μεθόδου, βρίσκεται μία σχεδόν βέλτιστη λύση αλλά πολλές φορές ο χρόνος που χρειάζεται ο κάθε αλγόριθμος ποικίλει απο μέθοδο σε μέθοδο [19], [29], [30], [36].

Οι λεγόμενες ευρετικές μέθοδοι έχουν εφαρμοστεί σε προβλήματα προγραμματισμού παραγωγής [9], [18], [40], [41], [42], [44], [45], [47]. Παρόλα αυτά ακόμη και οι πιό καλές ευρετικές μέθοδοι [31] αποτυγχάνουν να προσεγγίσουν βέλτιστες λύσεις σε πιό σύνθετα συστήματα που συναντώνται συνήθως στη πράξη [54], [55], [56], [57], [58], [59], [60]. Τον τελευταίο καιρό πολλοί ερευνητές επικεντρώνονται ολοένα και περισσότερο σε αλγορίθμους Τεχνητής Νοημοσύνης όπως Προσομοιωμένη Ανόπτηση [32], [34], [61], Αναζήτηση Tabu [11], Γενετικοί Αλγόριθμοι [2], [3], [5], [38], Ασαφής Λογική [13], [14], [15], [16], [17], Βελτιστοποίηση Αποικίας Μυρμηγκιών [28], [43], Επαναλαμβανόμενη Τοπική Αναζήτηση [39], Αλγόριθμοι Διαφορικής Εξέλιξης [33].

Ειδικότερα για την εφαρμογή των Γενετικών Αλγορίθμων σε προβλήματα χρονοπρογραμματισμού αξίζει να αναφέρουμε μερικές εργασίες.

Σε μία από αυτές, η αντικειμενική συνάρτηση προς βελτιστοποίηση αποτελείται από πολλά κριτήρια με τη μορφή σταθμισμένου μέσου. Ο Γενετικός Αλγόριθμος χρησιμοποιεί πολλούς πληθυσμούς πιθανών λύσεων όπου κάθε πληθυσμός έχει και από ένα κριτήριο βελτιστοποίησης. Αυτή η τεχνική εφαρμόζεται σε ένα σύστημα παραγωγής αποτελούμενο από παράλληλες μηχανές [64].

Σε μια άλλη εργασία εξετάζεται η περίπτωση ενός καταστήματος ροής και πιό συγκεκριμένα η εύρεση μιας αλληλουχίας εργασιών με σκοπό την ελαχιστοποίηση του συνολικού χρόνου περάτωσης των (makespan). Σε αυτό το πρόβλημα εφαρμόζεται ένας Γενετικός Αλγόριθμος και συγκρίνεται με άλλες μεθόδους βελτιστοποίησης όπως Τοπική Γειτονική Αναζήτηση (Local Neighbourhood Search) και Προσομοιωμένη Ανόπτηση (Simulated Anealing). Τα αποτελέσματα των συγκρίσεων αυτών έχουν ιδιαίτερο ενδιαφέρον καθώς δείχνουν ότι για συστήματα παραγωγής με μικρό αριθμό εργασιών η μέθοδος της Προσομοιωμένης Ανόπτησης επιτυγχάνει καλύτερα αποτελέσματα σε σύγκριση με τις άλλες δύο μεθόδους. Αντίθετα όσο μεγαλώνει η τάξη του προβλήματος η υπεροχή του Γενετικού Αλγορίθμου αρχίζει και διαφαίνεται τόσο από άποψη υπολογιστικού φόρτου όσο και από την συντόμευση του χρόνου παραγωγής που επιτυγχάνεται [65].

Πολύ καλά αποτελέσματα δίνουν επίσης και οι γενετικοί αλγόριθμοι που συνδυάζονται με άλλες μεθόδους όπως Τοπική Αναζήτηση ή Προσομοιωμένη Ανόπτηση. Οι συνδυασμοί αυτοί ονομάζονται Υβριδικοί Γενετικοί Αλγόριθμοι [66].

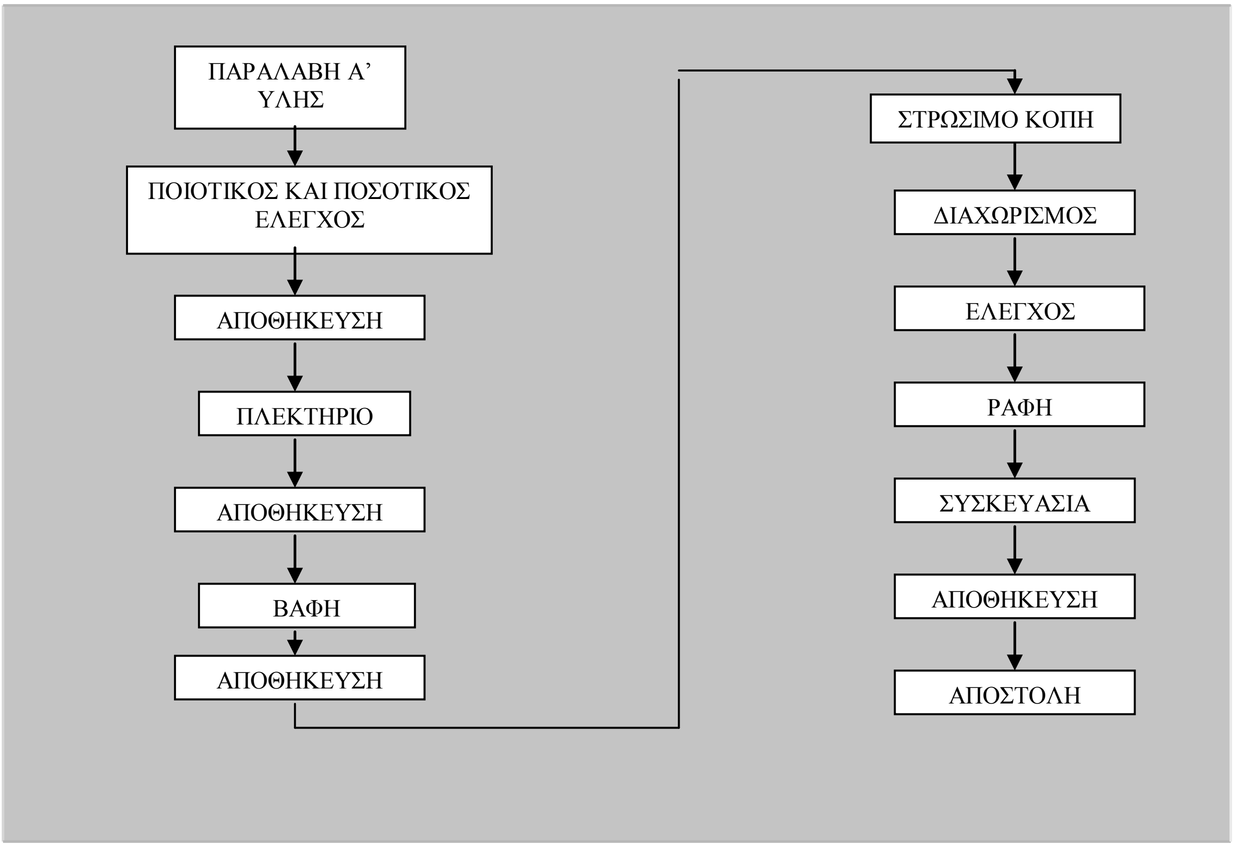
Τέλος, αναφέρουμε μια εργασία που μελετάει μια εργοστασιακή μονάδα παραγωγής τηλεοράσεων. Ένα τέτοιο σύστημα μοντελοποιείται με τη μορφή ενός υβριδικού καταστήματος ροής έξι σταδίων, όπου σε κάθε στάδιο έχουμε παράλληλες μηχανές με διαφορετικά τεχνικά χαρακτηριστικά. Εκεί οι ερευνητές προτείνουν έναν Γενετικό Αλγόριθμο που καθορίζει τόσο την ανάθεση των εργασιών αλλά και την σειρά με την οποία αυτές θα πρέπει να εισαχθούν στο σύστημα [67].

2. Περιγραφή και προσομοίωση του συστήματος

Στην παρούσα εργασία εξετάζεται η περίπτωση ενός πραγματικού συστήματος παραγωγής που ανήκει στο κλάδο της βιομηχανίας υφασμάτων. Τα δεδομένα που χρησιμοποιούνται είναι πραγματικά και έχουν συλλεγεί από εργοστάσιο της Βόρειας Ελλάδας.

2.1 Διαδικασία βιομηχανικής παραγωγής ενδυμάτων

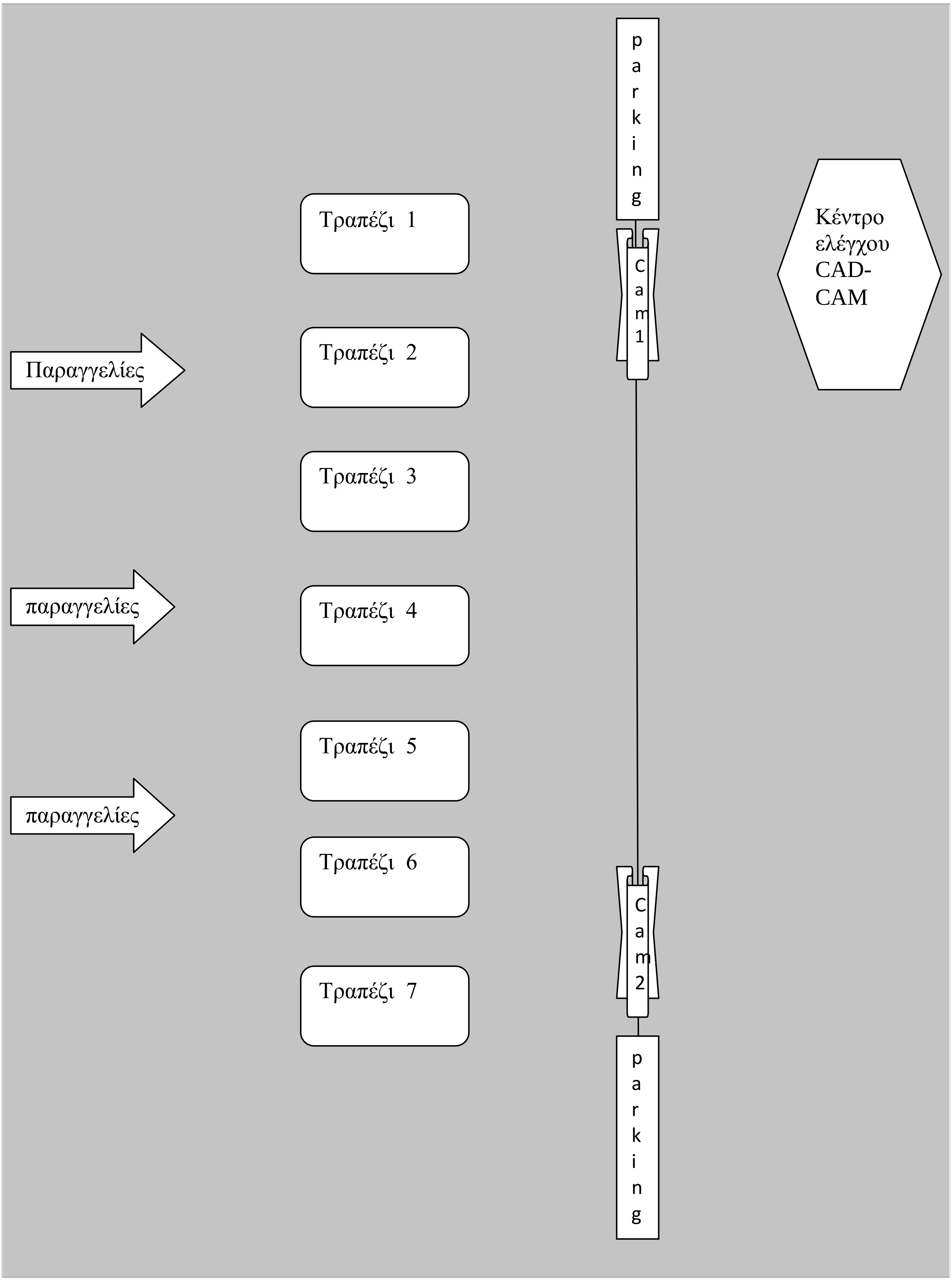
Εικόνα 2. Σχηματική αναπαράσταση τμημάτων εργοστασίου



Κλωστοϋφαντουργία είναι η τέχνη της κατασκευής νημάτων, υφασμάτων από τα νήματα αυτά και, τέλος, προϊόντων από τα υφάσματα. Η διαδικασία της μετατροπής της πρώτης ύλης σε τελικό προϊόν είναι μεγάλη και διέρχεται από πολλά στάδια. Στην εικόνα 2 δίνεται μια σχηματική αναπαράσταση της γραμμής παραγωγής από την πρώτη ύλη που είναι τα νήματα μέχρι το τελικό προιόν που είναι το ρούχο. Στο τμήμα της παραγωγής με το οποίο θα ασχοληθούμε γίνεται στρώσιμο και κοπή του υφάσματος και περιλαμβάνει δύο στάδια:
Το πρώτο στάδιο αναφέρεται στο στρώσιμο των υφασμάτων σε μεγάλα τραπέζια. Για κάθε στρώσιμο απαιτείται δέσμευση ανθρώπινων πόρων.
Μόλις ολοκληρωθεί το στρώσιμο ξεκινά η κοπή από ρομποτικούς μηχανισμούς, που στο εξής θα ονομάζονται CAM. Τα CAM εκτελούν κοπές με βάση κάποιο σχέδιο το

οποίο φορτώνουν αυτόματα. Η ρύθμιση των μηχανημάτων CAM δεσμεύει ανθρώπινο δυναμικό.

Εικόνα 3. Σχηματική αναπαράσταση τμήματος στρωσίματος-κοπής



Μία παραγγελία μπορεί να απαιτεί στρώσιμο σε περισσότερα του ενός τραπέζια και επίσης πολλά προγράμματα κοπής (π.χ, κορμός, μανίκια, κ.ο.κ).
Στην εικόνα 3 δίνεται μία σχηματική αναπαράσταση του τμήματος της παραγωγής που εξετάζουμε. Πιο συγκεκριμένα, η πρώτη εργασία γίνεται από τα συνεργεία στρωσίματος. Οι εργάτες παίρνουν τα υφάσματα και τα απλώνουν σε μεγάλα τραπέζια σε πολλές στρώσεις. Κάθε φορά που τελειώνει το στρώσιμο σε κάποιο τραπέζι ενεργοποιούνται οι χειριστές των μηχανημάτων CAM. Η δουλειά των χειριστών είναι να περάσουν μέσω ηλεκτρονικού προγράμματος CAD το πρόγραμμα παραγωγής στη μηχανή που είναι διαθέσιμη. Αυτό απαιτεί κάποιο χρόνο προετοιμασίας του CAM (setup time). Ο χρόνος αυτός μπορεί να σχετίζεται και με το ποιά παραγγελία εργαζόταν προηγουμένως το συγκεκριμένο CAM. Ετσι π.χ αν οι παραγγελίες είναι τελείως διαφορετικές θα χρειαστεί να αλλάξουμε το κοπτικό εργαλείο του CAM ή να φορτωθεί κάποιο άλλο πρόγραμμα κοπής. Αφού περαστεί το πρόγραμμα παραγωγής, η μηχανή CAM κατευθύνεται πάνω από το συγκεκριμένο τραπέζι και είναι έτοιμη να κόψει.

<u>Αξίζει να σημειωθεί ότι η κίνηση των μηχανημάτων CAM υπόκειται σε κάποιους περιορισμούς θέσης</u>. Κοιτώντας προσεκτικότερα την εικόνα 3 βλέπουμε ότι τα CAM δεν μπορούν να κινηθούν ανεξέλεγκτα καθώς έχουν έναν διάδρομο κίνησης πάνω στον οποίο εξυπηρετούνται και τα δύο. Έτσι π.χ αν το CAM1 δουλεύει στο τραπέζι 4 το CAM2 <u>δεν</u> μπορεί να εξυπηρετήσει τα τραπέζια 1,2,3 και φυσικά το τραπέζι 4. Τα τραπέζια στα οποία μπορεί να πάει το CAM2 είναι τα 5,6,7. Βέβαια αν πρέπει για κάποιο λόγο (π.χ αν υποστεί βλάβη το CAM1) το CAM2 να πάει σε ένα από τα τραπέζια 1,2,3 θα πρέπει να περιμένει να τελειώσει το CAM1 και να πάει στο πάρκινγκ του. Αυτή η ιδιομορφία στην κίνηση των CAM συνδιαζόμενη με τη προβληματική της ανάθεσης αναγκάζει τα συνεργεία στρωσίματος να είναι επιλεκτικά όσον αφορά τα τραπέζια που θα πρέπει να στρώνουν. Επιπλέον τα τραπέζια όταν στρωθούν έχουν και αυτά με τη σειρά τους προτίμηση σε ποιό CAM θα απευθυνθούν. Τέλος, τα CAM, όταν υπάρχουν πολλά στρωμένα τραπέζια μεροληπτούν όσον αφορά ποιά τραπέζια θα ψάξουν πρώτα για να κόψουν. Αυτές οι προτιμήσεις δηλώνονται στο αρχείο δεδομένων του προγράμματος προσομοίωσης και βοηθούν ουτωσόστε η διαδικασία στρωσίματος – κοπής να γίνεται χωρίς καθυστερήσεις και όσο το δυνατόν πιο βέλτιστα.

Ας επιστρέψουμε τώρα στο πρόβλημα παραγωγής που εξετάζουμε. Στη παράγραφο 1.1 δώσαμε έναν γενικό ορισμό ενός ευέλικτου καταστήματος ροής. Το σύστημα παραγωγής που αναλύουμε ανήκει σε αυτή τη κατηγορία όπου όμως αντί για εργασίες έχουμε παραγγελίες και κάθε παραγγελία έχει ένα πλήθος από εργασίες που πρέπει να γίνουν προκειμένου η παραγγελία να διεκπαιρεωθεί και να σταλεί στο επόμενο τμήμα παραγωγής. Ας χρησιμοποιήσουμε ένα παράδειγμα για την ευκολότερη κατανόηση των ανωτέρω:
Έστω ότι ένας πελάτης μας έχει παραγγείλει χίλια τεμάχια από μία συγκεκριμένη κατηγορία ενδύματος (π.χ πουκάμισα). Τα μηχανήματα CAM έχουν δυνατότητα κοπής πεντακοσίων πατρόν το καθένα. Άρα η παραγγελία σπάει σε δύο εργασίες. Πεντακόσια πατρόν θα στρωθούν\κοπούν στο τραπέζι 1 και τα υπόλοιπα

πεντακόσια στο τραπέζι 2. Επιπλέον μια τρίτη εργασία θα πρέπει να γίνει και αυτή θα είναι το στρώσιμο\κοπή των μανικιών (εικόνα4).

Εικόνα 4. Επιμέρους εργασίες της παραγγελίας

	παραγγελία	
	Στάδιο 1	στάδιο2
Εργασία 1η	στρώσιμο 500 τεμ	κοπή 500 τεμ
Εργασία 2η	στρώσιμο 500 τεμ	κοπή 500 τεμ
Εργασία 3η	στρώσιμο μανικιών	κοπή μανικιών

Ας αναδιατυπώσουμε λοιπόν τον ορισμό ώστε να ταιριάζει στο πρόβλημά μας:
Έστω ένα σύνολο N από παραγγελίες N={1,2,…,n} οι οποίες δίνονται τη χρονική στιγμή τ=0 (στατικό πρόβλημα).
Κάθε παραγγελία πρέπει να περάσει από m=2 στάδια.
Σε κάθε στάδιο υπάρχει ένα πλήθος από παράλληλες μηχανές οι οποίες μπορούν να εκτελέσουν τις εργασίες της εκάστοτε παραγγελίας. Στο πρόβλημα μας έχουμε στο πρώτο στάδιο ορισμένα συνεργεία στρωσίματος και επτά τραπέζια και στο δεύτερο στάδιο 2 μηχανές CAM και δύο συνεργεία λειτουργίας των CAM.
Κάθε εργασία πρέπει να περάσει από όλα τα στάδια και σε κάθε στάδιο δεσμεύει μια μόνο “μηχανή” (τραπέζι, CAM) και ένα συνεργείο.
Σκοπός μας είναι να βρούμε τη σειρά εκτέλεσης των παραγγελιών ούτος ώστε το πρόγραμμα παραγωγής μας να υλοποιείται σε όσο το δυνατόν λιγότερο χρονικό διάστημα. Το μέτρο απόδοσης δηλαδή που υιοθετήθηκε είναι η χρονική στιγμή ολοκλήρωσης και της τελευταίας παραγγελίας (makespan). Υπενθυμίζουμε εδώ πάλι ότι μία παραγγελία ολοκληρώνεται μόλις ολοκληρωθούν όλες οι εργασίες που την απαρτίζουν.
Από τα παραπάνω γίνεται φανερό ότι χρειαζόμαστε ένα εργαλείο για την εκτίμηση της απόδοσης του συστήματος όταν εφαρμόζεται ένα συγκεκριμένο πρόγραμμα παραγωγής. Αυτό το εργαλείο είναι η προσομοίωση.

2.2 Προσομοίωση

Η προσομοίωση είναι μία πειραματική μέθοδος με τη χρήση της οποίας γίνεται προσπάθεια αναπαράστασης της λειτουργίας ενός συστήματος παραγωγής χωρίς να είναι απαραίτητη η δημιουργία ή και λειτουργία του πραγματικού συστήματος.
Για να μελετηθεί συστηματικά η λειτουργία ενός συστήματος παραγωγής χρειάζεται να καθοριστούν οι κανόνες που το διέπουν και καθορίζουν την πορεία του. 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 τη περιγραφή του συστήματος κάθε στιγμή. 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 (η περισσοτέρων) πειράματος το οποίο καταγράφει την κατάσταση του συστήματος σε διαδοχικές χρονικές στιγμές αποτυπώνοντας ένα σενάριο εξέλιξης του συστήματος στο χρόνο. Με τη προσομοίωση μπορεί κανείς να αξιολογήσει την αποτελεσματικότητα η απόδοση ενός συστήματος πριν αυτό κατασκευαστεί με σκοπό τη βέλτιστη σχεδίαση του.

Το μέτρο απόδοσης που υιοθετήθηκε εδώ είναι η χρονική στιγμή ολοκλήρωσης και της τελευταίας εργασίας. Το συγκεκριμένο μέτρο απόδοσης ζητήθηκε από την εταιρία της οποίας τη γραμμή παραγωγής αναλύουμε.

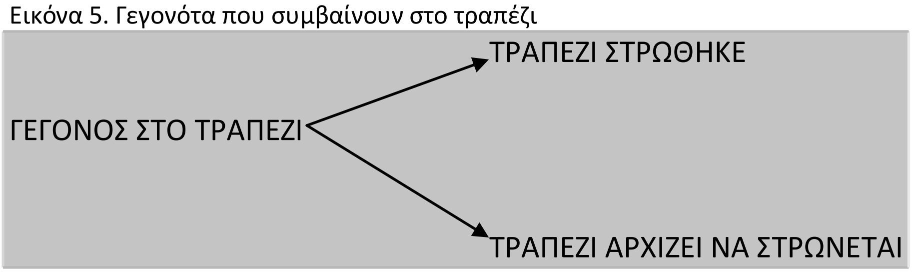
Το μοντέλο του συστήματος που αναλύουμε είναι συνάρτηση γνωστών παραμέτρων και επομένως το σύστημα μας είναι αιτιοκρατικό.

Κάθε στιγμή υπάρχει ένα πλήθος διαφορετικών γεγονότων τα οποία συναγωνίζονται για το ποιό θα συμβεί ενωρίτερα. Τέτοια γεγονότα είναι η αρχή και το τέλος κάποιας εργασίας. Τη στιγμή τ λαμβάνει χώρα ένα γεγονός και η κατάσταση του συστήματος μεταβάλλεται βηματικά. Λόγω της μεταβολής αυτής καθένα από τα γεγονότα που ευρίσκονται σε ανταγωνισμό εκείνη τη στιγμή επαναπροσδιορίζει το μελλοντικό χρόνο εμφάνισης του. Το επόμενο γεγονός του συστήματος είναι αυτό που θα συμβεί στο συντομότερο χρόνο και προκύπτει αλγοριθμικά μέσα από τη ρουτίνα επόμενου γεγονότος.

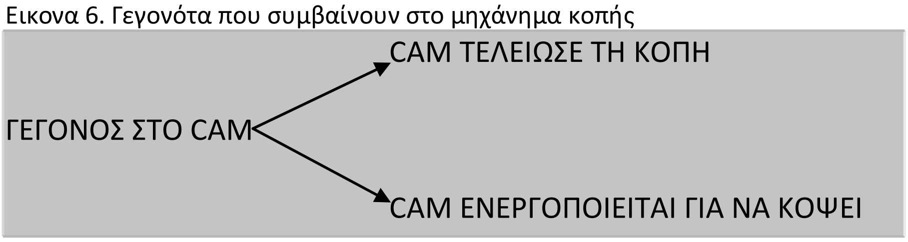
Για το τμήμα παραγωγής που εξετάζουμε τα γεγονότα αποτυπώνονται σχηματικά μέσα από τις εικόνες 5 και 6.

Στη συνέχεια, περιγράφονται τα γεγονότα και οι βασικές μεταβλητές που αλλάζουν σε κάθε γεγονός. Για πιό λεπτομερή περιγραφή ο αναγνώστης μπορεί να ανατρέξει στο παράρτημα όπου παρουσιάζεται μια υλοποίηση της προσομοίωσης του συγκεκριμένου συστήματος στη γλώσσα προγραμματισμού C.

Εικόνα 5. Γεγονότα που συμβαίνουν στο τραπέζι



Εικονα 6. Γεγονότα που συμβαίνουν στο μηχάνημα κοπής



Α. ΓΕΓΟΝΟΣ ΣΤΟ ΤΡΑΠΕΖΙ

Α1. ΤΡΑΠΕΖΙ ΣΤΡΩΘΗΚΕ

Ανανέωσε κατάσταση τραπεζίου που στρώθηκε (δήλωσε δηλαδή το τραπέζι ως στρωμένο)
Έλεγχος αν το CAM μπορεί να κόψει
Αν ναι: ανανέωσε κατάσταση CAM (το CAM δηλώνεται ως απασχολημένο), ανανέωσε χρόνο επόμενου γεγονότος στο CAM, ανανέωσε χρόνο επόμενου γεγονότος στο τραπέζι.
Αν όχι: Ο χρόνος επόμενου γεγονότος στο τραπέζι γίνεται άπειρος (το τραπέζι είναι μπλοκαρισμένο) και θα αλλάξει μόλις απελευθερωθεί κάποιο CAM.

Εφόσον στρώθηκε ένα τραπέζι απελευθερώθηκε ένα συνεργείο στρωσίματος.
Αν υπάρχουν παραγγελίες/εργασίες που δεν έχουν γίνει:
Αναζήτηση όλων των διαθέσιμων τραπεζιών με βάση λίστα προτίμησης και εύρεση τραπεζιού που θα στρώσει το συνεργείο στρωσίματος.
Ανανέωση χρόνου επόμενου γεγονότος στο τραπέζι που βρέθηκε.
Ανανέωση ελεύθερων συνεργείων στρωσίματος.
Ανανέωση ελεύθερων τραπεζιών.

Εφόσον βρήκαμε τραπέζι για στρώσιμο θα πρέπει να ενεργοποιήσουμε το CAM:
Αναζήτηση CAM που θα αναλάβει το τραπέζι που στρώνεται με βάση λίστα προτίμησης.
Στην αναζήτηση του CAM λαμβάνουμε υπ' όψιν το περιορισμό θέσης και κίνησης.

Α2. ΤΡΑΠΕΖΙ ΑΡΧΙΖΕΙ ΝΑ ΣΤΡΩΝΕΤΑΙ

Αν δεν υπάρχουν συνεργεία στρωσίματος αλλά ούτε παραγγελίες ή εργασίες παραγγελιών τότε:
Ο χρόνος επόμενου γεγονότος στο τραπέζι γίνεται άπειρος.
Αλλιώς:
Ανανέωση χρόνου επόμενου γεγονότος στο τραπέζι.
Ανανέωση συνεργείων στρωσίματος.
Εύρεση CAM με βάση λίστα προτεραιότητας και περιορισμούς θέσης/κίνησης που θα δεσμευτεί για το τραπέζι που στρώνεται.
Ανανέωση χρόνου επόμενου γεγονότος στο CAM

Β. ΓΕΓΟΝΟΣ ΣΤΟ CAM

Β1. CAM C ΤΕΛΕΙΩΣΕ ΤΗ ΚΟΠΗ (C=1, 2)

Εδώ το CAM C τελείωσε τη κοπή σε κάποιο τραπέζι. Άρα μπορούμε να πάμε αυτό το CAM (το C) σε κάποιο άλλο τραπέζι που είναι στρωμένο. Καθώς αναζητούμε τραπέζι, προσέχουμε το περιορισμό θέσης/κίνησης.
Αν υπάρχει τραπέζι στρωμένο που περιμένει και δεν βρίσκεται κανένα CAM στο τραπέζι και το CAM C μπορεί να πάει ανανέωσε:

Χρόνο επόμενου γεγονότος στο CAM (TEC[C]=CLOCK + SETUP)
Κατάσταση CAM (SC[C]=2)
Είδος επόμενου γεγονότος στο CAM (NEC[C]=2)
Ανανέωσε τη θέση του CAM C (CT[T]=C)

Αν όχι:
Ο χρόνος επόμενου γεγονότος στο CAM γίνεται άπειρος (TEC[C]=BIG)
Ανανέωσε τη κατάσταση του CAM (SC[C]=0)

Έπειτα η προσοχή μας στρέφεται στο άλλο CAM και ειδικότερα αν είναι ελεύθερο για χρήση. Επιπλέον πρέπει να είναι ελεύθερος για χρήση και ένας χειριστής CAM.
Αν ναι: Αναζητούμε κάποιο τραπέζι από τη λίστα προτίμησης (prt[]). Αφού βρούμε το τραπέζι ελέγχουμε αν είναι στρωμένο, αν δεν υπάρχει άλλο CAM πάνω από το τραπέζι και αν μπορεί να πάει το cam c.
Αν ναι ανανέωσε:
Κατάσταση CAM (SC[C]=2)
Θέση cam (CT[T])
Χρόνο επόμενου γεγονότος στο CAM (TEC[C]=CLOCK + SETUP)
Είδος επόμενου γεγονότος στο CAM (NEC[C])
Θέση CAM (TC[C])
Βγες από τη ρουτίνα

Αν όχι:
Ανανεώνουμε τον αριθμό των ελεύθερων CAM και χειριστών CAM αφού δεν μπορούμε να τα χρησιμοποιήσουμε.

<u>B2. To CAM C ΕΝΕΡΓΟΠΟΙΕΙΤΑΙ ΓΙΑ ΝΑ ΚΟΨΕΙ</u>
Το CAM πηγαίνει πάνω από κάποιο τραπέζι T. Δύο ενδεχόμενα μπορούν να συμβούν. Η το τραπέζι να μην είναι στρωμένο η να είναι στρωμένο και να περιμένει για να κοπεί.
Αν το τραπέζι δεν είναι στρωμένο:
Ανανέωσε κατάσταση CAM (SC[C]=3)
Ανανέωσε χρόνο επόμενου γεγονότος στο CAM (TEC[C]=BIG)

Αν το τραπέζι είναι στρωμένο:
Ανανέωσε κατάσταση CAM (SC[C]=1).
Ανανέωσε χρόνο επόμενου γεγονότος στο CAM (TEC[C]=CLOCK+TJC[][]).
Ανανέωσε είδος επόμενου γεγονότος στο CAM (NEC[C]=1).
Ανανέωσε χρόνο επόμενου γεγονότος στο τραπέζι (TET[T]=TEC-TCLEAR).
Ανανέωσε είδος επόμενου γεγονότος στο τραπέζι (NET[T]=2).

Ο αλγόριθμος προσομοίωσης ξεκινά θεωρώντας μία ακολουθία παραγγελιών με συγκεκριμένη σειρά. Για παράδειγμα ένα πρόγραμμα 10 παραγγελιών είναι

1	3	5	7	8	9	10	4	6	2

Δηλαδή πρώτα μπαίνει στο σύστημα παραγωγής η 1η παραγγελία η οποία είναι η 1, μετά η 2η παραγγελία η οποία είναι η 3, μετά η 3η παραγγελία η οποία είναι η 5 και ούτω καθεξής. Κάθε παραγγελία έχει κάποιον αριθμό από εργασίες οι οποίες πρέπει να εκτελεστούν όπως ακριβώς αναφέραμε στη παράγραφο 2.1. Οι παραγγελίες όταν μπαίνουν στο σύστημα δεν διακόπτονται και αυτό σημαίνει ότι η παραγγελία που προηγείται στο πρόγραμμα δεσμεύει σε κάθε στάδιο όσους πόρους χρειάζεται και έχει προτεραιότητα έναντι της επόμενης παραγγελίας.
Με την είσοδο της πρώτης εργασίας της πρώτης παραγγελίας το ρολόι του συστήματος αρχίζει να «τρέχει» μέχρις ότου τελειώσουμε με την τελευταία εργασία της τελευταίας παραγγελίας. Εκείνη τη στιγμή το χρονόμετρο σταματάει και η ένδειξη που γράφει είναι ο συνολικός χρόνος ολοκλήρωσης του συγκεκριμένου προγράμματος παραγωγής. Ο χρόνος αυτός ονομάζεται στην αγγλική βιβλιογραφία ως makespan και αυτός είναι ο όρος που θα χρησιμοποιείται από εδώ και στο εξής για να περιγράψουμε το κριτήριο απόδοσης του συστήματος μας. Παρόλα αυτά υπάρχουν και άλλα μέτρα απόδοσης του συστήματος. Θα αναφερθούμε σε αυτά στη παράγραφο 2.2.2 όπου θα συζητήσουμε για τα μέτρα απόδοσης των συστημάτων προγραμματισμού παραγωγής. Προς το παρόν ας δώσουμε σε μορφή ψευδογλώσσας τη ρουτίνα εύρεσης επόμενου γεγονότος καθώς επίσης και τη ρουτίνα που καλεί κάθε γεγονός:

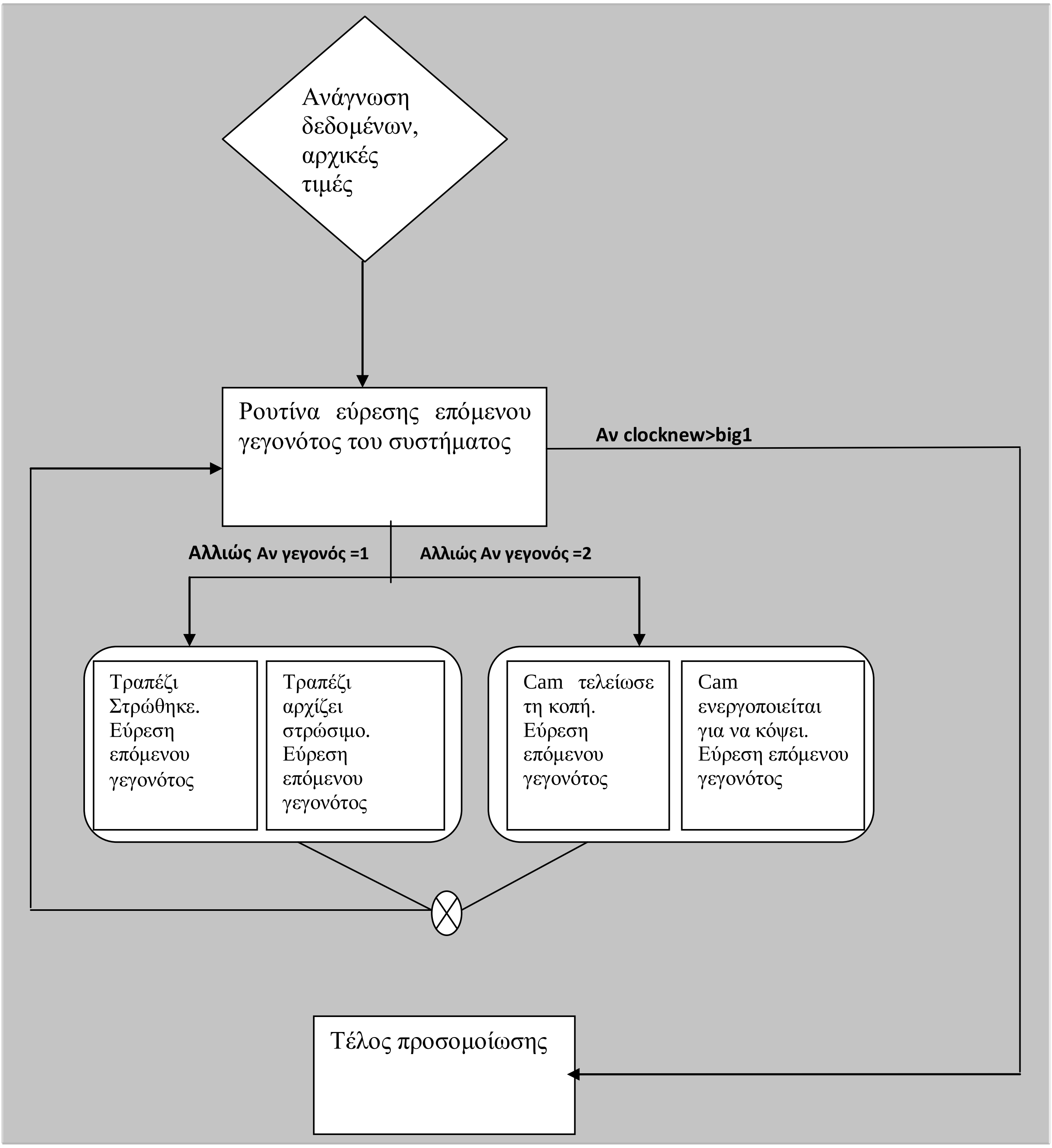
ΡΟΥΤΙΝΑ ΕΥΡΕΣΗΣ ΕΠΟΜΕΝΟΥ ΓΕΓΟΝΟΤΟΣ

```
Καινούργιο_ρολόι = big
ΓΕΓΟΝΟΣ=0
M=-1

Για(i=1; i<=NT; i++)
{
        Αν ( χρόνος_επόμενου_γεγονότος_στο_τραπέζι[i] < καινούργιο_ρολόι)
        {
        Καινούργιο_ρολόι =  χρόνος_επόμενου_γεγονότος_στο_τραπέζι[i]
        ΓΕΓΟΝΟΣ=1
        M=i
        }
}

Για (j=1; j<=NC; j++)
{
        Αν ( χρόνος_επόμενου_γεγονότος_στο_CAM[j]  <  καινούργιο_ρολοι)
        {
        Καινούργιο_ρολόι =  χρόνος_επόμενου_γεγονότος_στο_CAM[j]
        ΓΕΓΟΝΟΣ=2
        M=j
        }


}
```

ΡΟΥΤΙΝΑ ΚΛΗΣΗΣ ΓΕΓΟΝΟΤΩΝ

```
Αν ( καινούργιο_ρολόι> big1 )
Πήγαινε στο 2 (Τέλος προσομοίωσης)

Ρολόι_προσομοίωσης = καινούργιο_ρολόι

Αν (ΓΕΓΟΝΟΣ==1)
{
        Αν (επόμενο_γεγονός_στο_τραπέζι[Μ]==1)
        ΤΡΑΠΕΖΙ_ΣΤΡΩΘΗΚΕ(Μ)
        Αλλιώς  Αν (επόμενο_γεγονος_στο_τραπέζι[Μ]==2)
        ΤΡΑΠΕΖΙ_ΑΡΧΙΖΕΙ_ΣΤΡΩΣΙΜΟ(Μ)
}

Αλλιώς  Αν (ΓΕΓΟΝΟΣ==2)
{
        Αν (Επόμενο_Γεγονός_στο_Cam[M]==1)
        CAM_ΤΕΛΕΙΩΣΕ(Μ)
        Αλλιώς  Αν (Επόμενο_Γεγονός_στο_Cam[M]==2)
        CAM_ΕΝΕΡΓΟΠΟΙΕΙΤΑΙ(Μ)

}


Πήγαινε στη ρουτίνα επόμενου γεγονότος

2. ΤΕΛΟΣ ΠΡΟΣΟΜΟΙΩΣΗΣ: Εκτύπωση αποτελεσμάτων
```

Εικόνα 7. Διαγραμματική απεικόνιση ροής αλγορίθμου προσομοίωσης



Εδώ να σημειώσουμε πως έχουμε δύο τύπους γεγονότων, γεγονός στο τραπέζι και γεγονός στο cam, τα οποία υποδιαιρούνται σε άλλα δύο γεγονότα όπως δείξαμε στις εικόνες 5,6. Μόλις ο αλγόριθμος τελειώσει με τη ρουτίνα εύρεσης επόμενου γεγονότος θα τρέξει μία από τις τέσσερις διαφορετικές περιπτώσεις:

A1. Τραπέζι στρώθηκε

A2. Τραπέζι αρχίζει στρώσιμο

B1. CAM τελείωσε κοπή

B2. CAM ενεργοποιείται για να κόψει

με βάση ποιά θα έχει τον ενωρίτερο χρόνο πραγματοποίησης. Έπειτα θα γυρίσει πάλι στη ρουτίνα εύρεσης επόμενου γεγονότος η οποία ρουτίνα θα του υποδείξει σε ποιά περίπτωση γεγονότος θα πάει και ούτω καθεξής. Η επαναληπτική αυτή διαδικασία θα διακοπεί μόλις τελειώσουμε όλες τις παραγγελίες. Μόλις συμβεί αυτό το καινούργιο ρολόι του συστήματος (clocknew) θα πάρει μια πολύ μεγάλη τιμή (big > big1). Μόλις συμβεί αυτό ο αλγόριθμος θα εξέλθει από το βρόγχο και θα έχει τελειώσει η προσομοίωση του συγκεκριμένου προγράμματος παραγωγής.
Η χρονική στιγμή ολοκλήρωσης και της τελευταίας παραγγελίας (makespan) έχει αποθηκευτεί στη μεταβλητή που ονομάζεται ρολόι συστήματος (clock).

2.2.1 Αρχείο δεδομένων εισόδου

Σε αυτή τη παράγραφο θα δώσουμε λεπτομέρειες όσον αφορά την είσοδο του αλγορίθμου προσομοίωσης για το ευέλικτο κατάστημα ροής που αναλύουμε.
Θυμίζουμε ότι έχουμε ένα ευέλικτο κατάστημα ροής με 2 στάδια και σε κάθε στάδιο από δύο μηχανές χωρίς ενδιάμεσο αποθηκευτικό χώρο. Οι χρόνοι προετοιμασίας των μηχανών είναι ανεξάρτητοι και γνωστοί. Οι παραγγελίες δίδονται όλες τη χρονική στιγμή μηδέν. Οι διάρκειες εκτέλεσης των εργασιών αποτελούν πραγματικά δεδομένα και έχουν παρθεί από βιοτεχνία η οποία εντάσσεται στο κλάδο της βιομηχανίας υφασμάτων.
Η είσοδος του προγράμματος προσομοίωσης χρησιμοποιεί αυτό το αρχείο προκειμένου να υπολογίσει ένα μέτρο της απόδοσης του συστήματος. Το αρχείο αυτό χρησιμοποιείται και από το γενετικό αλγόριθμο που αναπτύξαμε προκειμένου να υπολογιστεί πόσο «καλό» είναι ένα πρόγραμμα παραγωγής. Ας επιστρέψουμε στον αλγόριθμο προσομοίωσης αναφέροντας ότι τα παρακάτω στοιχεία που θα εισάγουμε μπορούν να αλλάξουν για να περιγράψουν το ίδιο μοντέλο συστήματος αλλά με διαφορετικά δεδομένα.

Η πρώτη πληροφορία που περιέχει το αρχείο δεδομένων είναι το πλήθος των παραγγελιών καθώς και ο χρόνος που μπορεί να ξεκινήσει κάποιο στρώσιμο όταν το CAM φτάνει προς το τέλος της κοπής.

10	5

Δηλαδή υπάρχουν 10 παραγγελίες και η εργασία στρώσης υφάσματος στο τραπέζι μπορεί να ξεκινήσει πέντε λεπτά προτού τελειώσει το CAM τη κοπή του προηγούμενου υφάσματος πάνω από το συγκεκριμένο τραπέζι.

Έπειτα δίδεται το πρόγραμμα παραγωγής που θέλουμε να υπολογίσουμε κάποιο μέτρο απόδοσης του.

9	2	7	10	1	5	6	8	3	4

Ξεκινά να στρώνεται πρώτα η παραγγελία 9 μετά η 2 μετά η 7 και ούτω καθεξής.

Κατόπιν έχουμε τις θέσεις εργασίας.
Έχουμε αναφερθεί ήδη στη παράγραφο 2.1 ότι έχουμε τραπέζια που απλώνουμε τα υφάσματα σε πολλές στρώσεις, κάποιους εργάτες που στρώνουν καθώς επίσης και κάποιους εργάτες που χειρίζονται τα CAM

7	4	2

Δηλαδή έχουμε 7 τραπέζια 4 εργάτες που στρώνουν και δύο εργάτες που χειρίζονται τα CAM. Τα μηχανήματα CAM είναι πάντα δύο.

Έπειτα δίδουμε τις παραγγελίες και ότι σχετίζεται με αυτές. Ενδεικτικά αναφέρουμε για τη πρώτη παραγγελία, 3 στρώσεις-κοπές και χρόνους που φαίνονται παρακάτω:

3	
99.00	56.00
99.00	56.00
99.00	73.00

Η επόμενη πληροφορία στο αρχείο δεδομένων έχει σχέση με τους χρόνους προετοιμασίας CAM. Ο υπεύθυνος της εταιρίας μας ενημέρωσε πως είναι ανεξάρτητοι και δε σχετίζονται με το ποιά εργασία είχε γίνει προηγουμένως.
Εδώ υποθέτουμε ότι εφόσον έχουμε 7 τραπέζια και δυο θέσεις πάρκινγκ οι δυνατές θέσεις στις οποίες μπορεί να βρίσκεται το CAM είναι εννέα. Άρα έχουμε ένα πίνακα χρόνων με διάσταση 9Χ9:

5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0
5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0
5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0
5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0
5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0
5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0
5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0
5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0
5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0

Τα υπόλοιπα στοιχεία έχουν σχέση με τη προβληματική της ανάθεσης. Στα προβλήματα προγραμματισμού παραγωγής αναδύονται δυο προβλήματα. Το πρώτο έχει σχέση με την αλληλουχία των εργασιών/παραγγελιών και το δεύτερο με το πρόβλημα της ανάθεσης των εργασιών στις μηχανές. Ένας ακόμη λόγος χρησιμότητας των παρακάτω πληροφοριών έχει σχέση και με το περιορισμό θέσης/ κίνησης των CAM. Πιό συγκεκριμένα έχουμε τη σειρά προτίμησης των τραπεζιών:

1 7 2 6 3 5 4

Τα τραπέζια που είναι στην άκρη (1,7) προτιμώνται γιατί όταν ένα CAM είναι εκεί, το άλλο είναι πιο ελεύθερο να κινηθεί.

Μετά για κάθε τραπέζι θα βάλουμε τις προτιμήσεις του, όταν δηλαδή έχει στρωθεί σε ποιό CAM θα απευθύνεται:

1 2
1 2
1 2
1 2
2 1
2 1
2 1

Δηλαδή για τα τραπέζια 1-4 προτιμάται το CAM1 (αν είναι ελεύθερο) ενώ για τα υπόλοιπα το CAM2 (αν είναι ελεύθερο).

Τέλος για κάθε CAM θα βάλουμε τις προτιμήσεις του σε τραπέζια που θα ψάχνει πρώτα:

1 2 3 4 5 6 7
7 6 5 4 3 2 1

Δηλαδή για το CAM1 προτιμώνται τα πρώτα τραπέζια άρα η σειρά ειναι 1,2,3,4,5,6,7 ενώ για το CAM2 τα τελευταία, και η σειρά προτίμησης είναι 7,6,5,4,3,2,1.

2.2.2 Μέτρα απόδοσης προβλημάτων προγραμματισμού παραγωγής

Όπως ήδη έχουμε αναφέρει, το ζητούμενο είναι η εύρεση της σειράς (ακολουθίας) εκτέλεσης των παραγγελιών ώστε να ικανοποιείται κάποιο μέτρο απόδοσης. Πριν ορίσουμε κάποια μέτρα απόδοσης τέτοιων συστημάτων ας περιγράψουμε κάποιες βασικές μεταβλητές οι οποίες περιγράφουν και καθορίζουν το πρόβλημα (στη παράγραφο 2.2.1 κάναμε αναφορά και μιλήσαμε συγκεκριμένα για το πρόβλημα παραγωγής που εξετάζουμε. Εδώ γενικεύουμε περιληπτικά).
Έστω λοιπόν ένα κατάστημα εργασιών (δίκτυο παραγωγής) το οποίο αποτελείται από m μηχανές, οι οποίες εκτελούν κάποιες εργασίες. Οι εργασίες αυτές μπορεί να αναφέρονται σε παραγγελίες ή πρώτες ύλες που θα υποστούν κατεργασία. Οι παραγγελίες μπορεί να είναι όλες διαθέσιμες τη χρονική στιγμή τ=0 ή να έρχονται στο σύστημα σε μεταγενέστερους χρόνους. Ορίζουμε τις ακόλουθες ποσότητες:

p_i = διάρκεια εκτέλεσης εργασίας i (processing time)
r_i = χρόνος άφιξης εργασίας i στο σύστημα (ready time)
d_i = Προθεσμία, μέγιστος επιτρεπτός χρόνος παράδοσης (due date)
$a_i = d_i - r_i$

Για τις ποσότητες αυτές μπορεί να έχουμε πλήρη πληροφορία ή μερική μέσω πιθανοτήτων. Στη πρώτη περίπτωση έχουμε αιτιοκρατικούς χρόνους ενώ στη δεύτερη στοχαστικούς.
Οι μηχανές του συστήματος επιτελούν κάποιες λειτουργίες/εργασίες. Έστω ότι το σύστημα έχει m μηχανές συνολικά σε όλα τα στάδια του. Συμβολίζουμε με g_i το πλήθος όλων των υποεργασιών των εργασιών.

Κάθε εργασία αποτελείται από gi υποεργασίες:

$$
\begin{array}{ll}
m_{i,1} & p_{i,1} \\
m_{i,2} & p_{i,2} \\
. & . \\
. & . \\
. & . \\
m_{i,gi} & p_{i,gi}
\end{array}
$$

$m_{i,j}$ = Η μηχανή i εκτελεί την υποεργασία j.
$p_{i,j}$ = ο χρόνος που θα χρειαστεί η μηχανή $m_{i,j}$ για να ολοκληρώσει την υποεργασία.
$p_i = \sum p_{i,j}$ Ο συνολικός χρόνος εκτέλεσης της εργασίας.

$W_{i,j}$= Χρόνος αναμονής υποεργασίας j της εργασίας i στην ουρά πριν ξεκινήσει η κατεργασία της.

$W_i = \sum w_{i,j}$ = Συνολικός χρόνος αναμονής εργασίας i

<u>Τώρα είμαστε έτοιμοι να ορίσουμε κάποια συνήθη μέτρα απόδοσης</u>:

Παρακάτω θα γενικεύσουμε αλλά οι ίδιοι τύποι ισχύουν και για το πρόβλημα προγραμματισμού παραγωγής που εξετάζουμε.

C_i , Χρόνος συμπλήρωσης (παράδοσης) εργασίας i. Έιναι ο χρόνος ολοκλήρωσης και της τελευταίας υποεργασίας της εργασίας (completion time). Επομένως

$$C_i = r_i + W_{i,j} + p_{i,1} + W_{i,2} + p_{i,2} + \dots + W_{i,gi} + p_{i,gi} = r_i + p_i + W_i$$

F_i , Χρόνος ροής, συνολικός χρόνος αναμονής εργασίας i στο σύστημα (flow-time)

$$F_i = W_{i,1} + p_{i,1} + W_{i,2} + p_{i,2} + \dots + W_{i,gi} + p_{i,gi} = p_i + W_i = C_i - r_i$$

L_i , Θετική η αρνητική απόκλιση από τη προθεσμία παράδοσης , βραδύτητα (lateness)

$L_i = C_i - d_i = F_i - d_i$

T_i , καθυστέρηση στη παράδοση (Tardiness)
$T_i = \max \{ L_i , 0 \}$

E_i , Ενωρίτητα (earliness)
$E_i = \max \{ -L_i , 0 \}$

Οι ποσότητες L_i , T_i , E_i , αποτελούν τρεις διαφορετικούς τρόπους για να περιγράψουμε το χρόνο ολοκλήρωσης της εργασίας σε σχέση με τον επιθυμητό χρόνο ολοκλήρωσης.

Όταν μας δίνεται ένα πρόγραμμα παραγωγής:

Fmax , χρόνος ολοκλήρωσης προγράμματος παραγωγής. Στην ουσία το Fmax είναι το διάγραμμα Gantt των παραγγελιών ή αλλιώς makespan.
$Fmax = \max \{ F(1) , F(2) ,\dots, F(N) \}$.

Tmax , καθυστέρηση στη παράδοση των παραγγελιών
$Tmax = \max \{F(i) - D(i) , 0\}$

Fmean , Μέσος συνολικός χρόνος αναμονής εργασιών στο σύστημα
$Fmean = 1/N \sum_i F(i)$

Tmean , Μέση καθυστέρηση με προθεσμίες
$Tmean = 1/N [\sum_i \max_i \{ F(i) - D(i) , 0 \}]$

Πιό σύνθετα μέτρα απόδοσης μπορούν να προκύψουν από τα παραπάνω με τη μορφή σταθμισμένου κριτηρίου. Παρόλα αυτά υπάρχουν κάποιες ποσότητες οι οποίες δεν μπορούν να αποτελέσουν ένα αντικειμενικό κριτήριο μέτρησης της απόδοσης ενός συστήματος όπως παραδείγματος χάριν η ποσότητα E_{mean} ή E_{max}.
Επιπλέον ένα πρόγραμμα παραγωγής το οποίο επιτυγχάνει ελάχιστη κατά μέσο όρο απόκλιση από τη προθεσμία παράδοσης (min E_{mean}) δεν συνεπάγεται ότι θα έχει ελάχιστη κατά μέσο όρο καθυστέρηση στη παράδοση (min T_{mean}). Αυτό ίσως να συμβαίνει σε πολύ συγκεκριμένες περιπτώσεις. Σε κάθε περίπτωση το μέτρο απόδοσης που θα εφαρμόσουμε εξαρτάται από το πρόβλημα που επιλύουμε αλλά και το τι επιθυμεί η εταιρία σαν αντικείμενο βελτιστοποίησης.

3. Ένας Γενετικός Αλγόριθμος για την αναζήτηση βέλτιστης λύσης

3.1 Γενική ιδέα πίσω από τους Γενετικούς Αλγορίθμους

Μέχρι στιγμής έχουμε παρουσιάσει το πρόβλημα τόσο σε γενική μορφή αλλά και πιό συγκεκριμένα μέσω της παραγράφου 2.2 όπου κάναμε λόγο για ένα τμήμα παραγωγής μιας συγκεκριμένης βιομηχανικής μονάδας. Ορίσαμε το πρόβλημα σαν ένα πρόβλημα προγραμματισμού παραγωγής και ειδικότερα για ένα ευέλικτο κατάστημα ροής 2 σταδίων με δύο μηχανές σε κάθε στάδιο. Γενικεύοντας θα λέγαμε ότι σε πολλά προβλήματα που σχετίζονται με το προγραμματισμό της παραγωγικής διαδικασίας αλλά και σε άλλους κλάδους όπως π.χ ο σχεδιασμός κινητήρων αεροπλάνων, κατασκευή γεφυρών, σχεδιασμός αγωγών αερίου, σε προβλήματα ρομποτικής κίνησης, στην οικονομία, στο σχεδιασμό λειτουργία και έλεγχο των συστημάτων ηλεκτρικής ενέργειας, χρησιμοποιούνται οι γενετικοί αλγόριθμοι. Πριν προχωρήσουμε σε αλγοριθμική υλοποίηση του γενετικού αλγορίθμου ας συζητήσουμε ποιά ιδέα προσομοιώνει.

Ο θεμελιωτής της θεωρίας των ειδών στη βιολογία ήταν ο Charles Darwin. Στο βιβλίο του , *"The origin of species"* , χρησιμοποιήθηκε ο όρος *"descent with modifications" (απόγονοι με τροποποιήσεις)* προκειμένου να μελετήσει την εξέλιξη των έμβιων όντων αλλά και γενικότερα τη μετατροπή/διαφοροποίηση που αυτά παρουσίαζαν κατά τη διάρκεια χιλιάδων ετών δημιουργώντας κάποιο «καλύτερο» κομμάτι του εαυτού τους. Η κινητήρια δύναμη σε όλα αυτά είναι η μάχη για επιβίωση του είδους, που αναγκάζει κάθε είδος να μετασχηματίζεται ανάλογα με το πόσο εχθρικό ή φιλικό είναι ένα περιβάλλον και να περνάει με κάποιο τρόπο αυτή του τη πληροφορία στους απογόνους του. Ενδιαφέρουσα είναι και η περιγραφή του στο βιβλίο:

«As many more individuals of each species are born than can possibly survive; and as, consequently, there is a frequently recurring struggle for existence, it follows that any being, if it vary however slightly in any manner profitable to itself, under the complex and sometimes varying conditions of life, will have a better chance of surviving, and thus be naturally selected.»

Απέναντι λοιπόν στην επίθεση από το εξωτερικό περιβάλλον, τη μόνη εναλλακτική που έχει κάθε οργανισμός είναι να μετασχηματίζεται, να αλλάζει, ελπίζοντας ότι αυτές του οι αλλαγές θα είναι όσο το δυνατόν πιο κερδοφόρες για αυτόν προκειμένου να έχει όσο το δυνατόν αυξημένες πιθανότητες επιβίωσης για να μπορεί να δημιουργήσει και αυτός με τη σειρά του απογόνους.

Λίγο μετά ο βρετανός Φιλόσοφος Herbet Spencer (1820-1903) χρησιμοποιεί την έκφραση *"survival of fittest"* για να αναφερθεί σε αυτό που ο Charles Darwin ονόμαζε *"descent with modifications"*. Σύμφωνα με τους μελετητές του Δαρβίνου Desmond και Moore, το δεύτερο μισό του 19ου αιώνα υπήρξε περίοδος μεγάλων ιδεολογικών ζυμώσεων αλλά και αλλαγών σε όλους τους τομείς της κοινωνικής ζωής στην Αγγλία. Το προοδευτικότερο στρώμα της βικτωριανής και

μεταβικτωριανής Αγγλίας προσπαθούσε να προωθήσει την αναμόρφωση του στρατεύματος, της εκπαίδευσης, του ευρύτερου δημόσιου τομέα.
Στο πλαίσιο αυτό η θεωρία του Δαρβίνου, διαβάστηκε με μεγαλύτερο ζήλο και έδωσε τροφή στην ιδέα της εξέχουσας θέσης που είχε ο άνθρωπος στη φύση ως το πιο προσαρμοστικό αλλά και πιό εξελισσόμενο είδος. Καθώς ο Spencer ήταν από τους ένθερμους υποστηριχτές των παραπάνω χρησιμοποίησε την λέξη «εξέλιξη» αντί της «μεταμόρφωσης» (transformation) που έδειχνε να προτιμά ο Δαρβίνος αλλά και οι συνάδελφοί του (για να περιγράψουν την ανάπτυξη μίας ομάδας οργανισμών) στα επιστημονικά κείμενά του.
Ο Spencer, ο οποίος δεν έβλεπε και μεγάλη διαφορά μεταξύ της ανάπτυξης ενός ατόμου και της ανάπτυξης μιάς ομάδας οργανισμών, χρησιμοποίησε παντού τη λέξη εξέλιξη. Φαίνεται δε ότι είχε πολύ μεγαλύτερο κοινό από τον Δαρβίνο και έτσι ο όρος επιβλήθηκε.

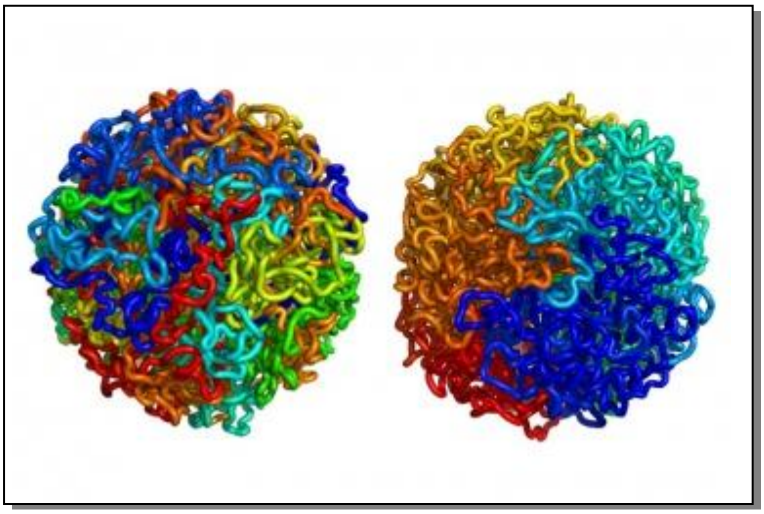
Αφήνοντας μια και καλή τα ιστορικά δεδομένα επιστρέφουμε λοιπόν στο όλο παιχνίδι των τροποποιήσεων που αναγκάζεται να μπει κάθε έμβιος οργανισμός προκειμένου να δημιουργήσει τα εργαλεία εκείνα που θα του δώσουν αυξημένες δυνατότητες για επιβίωση έναντι των υποψήφιων θηρευτών του.
Μέσα από όλη αυτή την «εξελικτική διαδικασία» τείνουν να επιζούν οι οργανισμοί εκείνοι οι οποίοι είναι καλύτεροι. Βέβαια όλα αυτά διαδραματίζονται στο πέρασμα εκατομμυρίων χρόνων και υπάρχει μια μορφή ανατροφοδότησης αλλά και αλληλεπίδρασης μεταξύ των ειδών και του περιβάλλοντος τόσο σε μικροσκοπικό όσο και σε μακροσκοπικό επίπεδο. Αυτή τη δυνατότητα δεν μπορεί να προσομοιώσει επακριβώς ο γενετικός αλγόριθμος που τρέχει μόλις σε μερικά δευτερόλεπτα.

Ένας εξελικτικός αλγόριθμος προσπαθεί να μιμηθεί (η αλήθεια είναι πως το κάνει σε ένα πολύ απλοϊκό επίπεδο) αυτή την διαδικασία. Αρχίζει με ένα αρχικό πληθυσμό λύσεων του προβλήματος και εν συνεχεία μέσα από κάποιες επαναληπτικές διαδικασίες τροποποιεί / μετασχηματίζει τα «καλά» άτομα του πληθυσμού όπως ακριβώς γίνεται στη φύση με την διασταύρωση και τη μετάλλαξη ελπίζοντας ότι μετά από κάποιες επαναλήψεις κάποιο μέλος κάποιου πληθυσμού να έχει επισκεφτεί ένα όσο το δυνατόν καλό βέλτιστο σημείο του χώρου λύσεων του προβλήματος.
Υπάρχουν εξελικτικοί αλγόριθμοι οι οποίοι αυξομειώνουν το μέγεθος του πληθυσμού ενώ κάποιοι άλλοι το κρατούν σταθερό. Άλλοι εξελικτικοί αλγόριθμοι τρέχουν παράλληλα δύο ή και περισσότερους πληθυσμούς μέσω παράλληλου προγραμματισμού. Σε αυτές τις περιπτώσεις χρησιμοποιούνται και άλλοι γενετικοί τελεστές όπως η μετανάστευση (migration).
Η αλήθεια είναι πως δεν υπάρχει ξεκάθαρη διαχωριστική γραμμή ανάμεσα στους γενετικούς και εξελικτικούς αλγορίθμους. Από την άλλη το σίγουρο είναι ότι ένας γενετικός αλγόριθμος έχει οπωσδήποτε έναν πληθυσμό υποψήφιων λύσεων, μία διαδικασία επιλογής καλών λύσεων καθώς και κάποιους γενετικούς τελεστές, συνήθως είναι η διασταύρωση και η μετάλλαξη.
Η συγκράτηση ενός πληθυσμού λύσεων πολύ απλά σημαίνει ότι ο αλγόριθμος έχει τη δυνατότητα να ψάχνει παράλληλα διαφορετικές περιοχές του χώρου εφικτών λύσεων της συνάρτησης. Οι γενετικοί τελεστές είναι ένας τρόπος για να μεταβούμε

ταυτόχρονα από μια περιοχή που ψάξαμε σε κάποια άλλη, πολύ μακριά της ή πολύ κοντά της, ελπίζοντας έτσι να ανακαλύψουμε ένα καλύτερο σημείο.
Πριν εφαρμοστούν οι γενετικοί τελεστές στο πληθυσμό θα πρέπει να αξιολογηθούν όλα τα άτομα του πληθυσμού και να περάσουν τα καλύτερα. Αυτή η διαδικασία ονομάζεται διαδικασία επιλογής και μιμείται τη πίεση που ασκούν οι διάφοροι κίνδυνοι του περιβάλλοντος σε κάποιον οργανισμό. Στην ουσία αποτελεί ένα είδος φίλτρου όπου στο τέλος έχουν αφαιρεθεί από το πληθυσμό με μεγάλη πιθανότητα οι χειρότερες λύσεις.

Μια πρόσφατη εξέλιξη στις έρευνες γύρω από το ανθρώπινο χρωμόσωμα που αξίζει να αναφερθεί έρχεται από το Ιατρικό τμήμα του πανεπιστημίου της Μασαχουσέτης (http://web.mit.edu/press/2009/3d-genome.html).
Ομάδα επιστημόνων κατάφερε με τη βοήθεια υπολογιστή να βρει και να αναπαραστήσει τη γεωμετρία του DNA. Να αναφέρουμε ότι αυτή η διπλή έλικα μήκους δύο μέτρων που έρχεται στο μυαλό μας και αναφέρεται στο DNA θα πρέπει να διπλώσει και να χωρέσει στο πυρήνα ενός κυττάρου διαμέτρου περίπου ενός εκατοστού του χιλιοστού. Κάθε κύτταρο σχηματίζει περίπου τρία δισεκατομμύρια ζεύγη βάσεων. Η πυκνότητα πληροφορίας που βρίσκεται στο πυρήνα ενός κυττάρου είναι τρισεκατομμύρια φορές μεγαλύτερη από αυτή που βρίσκεται σε ένα chip. Οι ερευνητές έβγαλαν δυο ενδιαφέροντα ευρήματα. Το πρώτο λέει ότι το ανθρώπινο γονιδίωμα είναι οργανωμένο σε δυο ξεχωριστά διαμερίσματα και το δεύτερο ότι υιοθετεί μια οργάνωση γνωστή από τα μαθηματικά ως fractal (εικόνα 8).

Εικόνα 8



Όπως εξήγησε ο επικεφαλής των ερευνητών Τζομπ Ντέκερ, *«βλέπουμε τώρα με αυτή την τρισδιάστατη απεικόνιση ότι γονίδια που είναι πολύ απομακρυσμένα στη γραμμική αλληλουχία του γονιδιώματος βρίσκονται ουσιαστικώς το ένα δίπλα στο άλλο σε αυτή την πακεταρισμένη δομή.*
Με απλά λόγια, ένα γονίδιο δεν έρχεται σε επαφή μόνο με τα γειτονικά του στην αλυσίδα, αλλά και με πολλά άλλα, που αν το γονιδίωμα είχε γραμμική μορφή θα ήταν πολύ απομακρυσμένα από αυτό.

Οι Γενετικοί Αλγόριθμοι (ΓΑ) δανείζονται κάποιες λέξεις από τη βιολογία. Θα αναφερόμαστε σε άτομα/μέλη κάποιου πληθυσμού (π.χ προγράμματα παραγωγής) χαρακτηρίζοντάς τα ως χρωμοσώματα. Βέβαια στη φύση κάθε κύτταρο ενός οργανισμού που ανήκει σε κάποιο είδος έχει μέσα του κάποιο συγκεκριμένο αριθμό από χρωμοσώματα. Για παράδειγμα ο άνθρωπος έχει 46 χρωμοσώματα.
Κάθε χρωμόσωμα στο γενετικό αλγόριθμο είναι και μία δόκιμη λύση. Στο πρόβλημά μας ένα χρωμόσωμα κάποιου πληθυσμού εκφράζει ένα πρόγραμμα παραγωγής. Ένα χρωμόσωμα αποτελείται από κάποια στοιχεία που ονομάζονται γονίδια. Το πλήθος των γονιδίων είναι το μήκος του χρωμοσώματος. Στο πρόβλημά μας κάθε χρωμόσωμα έχει μήκος όσες ακριβώς είναι και οι παραγγελίες. Στην εικόνα 9 φαίνεται ένας πληθυσμός δέκα ατόμων/χρωμοσωμάτων, κάθε χρωμόσωμα έχει και μια τιμή. Αυτή η τιμή, η οποία περιγράφει το χρόνο ολοκλήρωσης και της τελευταίας παραγγελίας, μας επιτρέπει να αξιολογούμε τα χρωμοσώματα μεταξύ τους. Μικρότερες τιμές αντιστοιχούν σε καλύτερες δόκιμες λύσεις.

Εικόνα 9 . Παράδειγμα ενός πληθυσμού

i=0	9 2 7 10 1 5 6 8 3 4	1951
i=1	6 8 10 4 7 2 9 5 3 1	2025
i=2	9 6 8 3 2 1 7 4 10 5	1935
i=3	2 9 6 1 5 10 7 3 8 4	1940
i=4	3 1 7 4 2 5 6 9 8 10	2049
i=5	4 6 7 1 5 9 3 8 2 10	1939
i=6	3 1 8 2 10 7 6 5 4 9	2034
i=7	1 6 5 9 8 3 2 10 7 4	1925
i=8	6 3 4 7 8 5 9 1 2 10	2022
i=9	6 1 2 7 9 4 3 5 8 10	2031
i=10	1 6 5 9 8 3 2 10 7 4	1925

Τα βασικά βήματα ενός γενετικού αλγορίθμου είναι τα εξής:

1 . Δημιουργία ενός αρχικού πληθυσμού Π(τ=0)
Εδώ δημιουργούμε μέσω γεννήτριας τυχαίων αριθμών έναν αρχικό πληθυσμό χρωμοσωμάτων. Εναλλακτικά ο αρχικός πληθυσμός μας θα μπορούσε να δημιουργηθεί από κάποια τεχνική βελτιστοποίησης ευελπιστώντας έτσι να αρχίσουμε από μια αρκετά καλή περιοχή.
Ο πληθυσμός αποτελείται από ένα πλήθος εναλλακτικών προγραμμάτων παραγωγής.

2 . Αξιολόγηση Πληθυσμού/Συνάρτηση καταλληλότητας
Η αξιολόγηση του πληθυσμού γίνεται μέσω του προγράμματος προσομοίωσης που αναφέραμε στο κεφάλαιο 2. Έτσι σε κάθε πρόγραμμα παραγωγής (χρωμόσωμα) αντιστοιχούμε και την τιμή που έχει η αντικειμενική συνάρτηση βελτιστοποίησης. Να αναφέρουμε εδώ ότι η υλοποίηση του χρωμοσώματος στη γλώσσα C έγινε μέσα από δομή δεδομένων όπου αποθηκεύεται το πρόγραμμα παράγωγης καθώς επίσης και μια τιμή που δείχνει πόσο «υγιές» είναι ένα πρόγραμμα παραγωγής.

3. Όσο δεν ισχύει το κριτήριο τερματισμού δημιουργούμε μια νέα γενεά (τ+1) και συνεχίζουμε την εκτέλεση του γενετικού αλγορίθμου.

$\tau \leftarrow \tau + 1$

4 . Επίλεξε Π(τ)

Η διαδικασία της επιλογής είναι μια σημαντική ενέργεια στους γενετικούς αλγορίθμους. Στην ουσία αυτή η συνάρτηση αποφασίζει ποιά προγράμματα παραγωγής θα υπερισχύσουν και θα ζήσουν και έναντι ποιών. Εδώ επιθυμούμε να περνούν οι καλές λύσεις και οι χειρότερες να εξουδετερώνονται με μεγάλη πιθανότητα. Όσα προγράμματα παραγωγής επιζήσουν θα περάσουν από την επίδραση κάποιων γενετικών τελεστών. Αυτό θα γίνει στο βήμα 5.

5 . Δημιουργία νέων λύσεων από Π(τ)

Εδώ λαμβάνουν δράση οι γενετικοί τελεστές. Η βοήθειά τους είναι σημαντική καθώς αυξάνουν την αναζήτηση και την προεκτείνουν σε όσο το δυνατόν περισσότερα σημεία του χώρου. Οι γενετικοί τελεστές που χρησιμοποιούνται συνήθως είναι η διασταύρωση και η μετάλλαξη. Γενικότερα όμως οι τελεστές που θα χρησιμοποιήσουμε σε κάποιον γενετικό αλγόριθμο έχουν σχέση με το είδος του προβλήματος καθώς και με την αναπαράσταση της λύσης.

Στο παρόν πρόβλημα παραγωγής που εξετάζουμε ***έχει χρησιμοποιηθεί και ένας πρωτότυπος τελεστής ο οποίος τροποποιεί τυχαία κάποια άτομα με βάση πληροφορίες που παίρνει από ολόκληρο το πληθυσμό***. Όλοι οι γενετικοί τελεστές έχουν κάποια πιθανότητα εμφάνισης στα άτομα του πληθυσμού.

6 . Αξιολόγηση Πληθυσμού/Συνάρτηση καταλληλότητας

7. Τέλος

Ο αλγόριθμος περατώνεται μετά από κάποιο αριθμό γενεών. Μόλις πάψει να ισχύει το κριτήριο τερματισμού του βήματος 3 ο αλγόριθμος τελειώνει. Σε όλη τη διάρκεια που τρέχει ο γενετικός αλγόριθμος χρησιμοποιούμε μια μεταβλητή που αποθηκεύει κάθε φορά το καλύτερο πρόγραμμα παραγωγής. Αυτό το πρόγραμμα παραγωγής είναι το καλύτερο όλων των γενεών και αυτή είναι η λύση του προβλήματος.

Το πρώτο πράγμα που θα γίνει στην αρχή είναι να δημιουργηθεί ένας αρχικός πληθυσμός από προγράμματα παραγωγής και θα γίνει η αξιολόγηση του πληθυσμού. Στη συνέχεια ο αλγόριθμος θα μπει σε μία επαναληπτική διαδικασία που περιλαμβάνει τα βήματα 3 μέχρι 6. Κατά τη διάρκεια αυτών των επαναλήψεων ο αλγόριθμος θα εξερευνεί το τοπίο λύσεων του προβλήματος προσπαθώντας να φτάσει στο πιό «χαμηλό» σημείο του που είναι και το βέλτιστο makespan.

Εδώ δεν έχουμε αναφέρει ακόμη μία ρουτίνα του γενετικού αλγορίθμου που διορθώνει το πληθυσμό. Η ρουτίνα αυτή περιγράφεται παρακάτω περιληπτικά αλλά θα ασχοληθούμε με αυτή στη παράγραφο 3.2 όπου περιγράφουμε αναλυτικά όλες τις λεπτομέρειες του γενετικού αλγορίθμου που δημιουργήσαμε.

Στη συνέχεια θα δώσουμε σε μορφή ψευδογλώσσας τη βασική μορφή του Γ.Α που υλοποιήσαμε.

Ο ΓΕΝΕΤΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ ΣΕ ΜΟΡΦΗ ΨΕΥΔΟΓΛΩΣΣΑΣ

```
While (κριτήριο ελέγχου)
{
        If (αύξηση makespan)
    {
      Επαναλήψεις χωρίς βελτίωση = επαναλήψεις χωρίς βελτίωση+1
          If (επανάληψη χωρίς βελτίωση == 20)
       {
          Κατάταξη πληθυσμού()
          Διόρθωση πληθυσμού()
          Υπολογισμός Συνάρτησης καταλληλότητας()
          Ανανέωση κριτηρίου ελέγχου
          Επαναλήψεις χωρίς βελτίωση=0
        }

     }
    Διαδικασία επιλογής()
    Γενετικοί τελεστές()
    Υπολογισμός συνάρτησης καταλληλότητας()
    Ανανέωσε βέλτιστο makespan
    Συνέχισε
}
```

Ο γενετικός αλγόριθμος που σχεδιάστηκε, ενισχύθηκε με ένα μπλοκ εντολών που τον βοηθά να ανακάμπτει από τα τοπικά βέλτιστα. Το συγκεκριμένο κομμάτι του κώδικα έχει ως εξής:

```
If (αύξηση makespan)
    {
        Επαναλήψεις χωρίς βελτιωση=επαναληψεις χωρίς βελτιωση+1
          If (επανάληψη χωρίς βελτιωση==20)
       {
          Κατάταξη πληθυσμού()
          Διόρθωση πληθυσμού()
          Υπολογισμός συνάρτησης καταλληλότητας ()
          Ανανέωση κριτηρίου ελέγχου
          Επαναλήψεις χωρίς βελτίωση=0
        }
    }
```

Χρησιμοποιούμε ένα μετρητή, ο οποίος μετράει το πλήθος των φορών που δε προκύπτει βελτίωση σε σχέση με τη προηγούμενη επανάληψη. Όταν ο μετρητής αυτός φτάσει κάποια συγκεκριμένη τιμή (π.χ 20) ενεργοποιείται η διόρθωση του πληθυσμού αφού πρώτα έχει γίνει κατάταξη πληθυσμού από το καλύτερο προς το χειρότερο.

3.2 Τα βασικά χαρακτηριστικά του Γενετικού Αλγορίθμου

3.2.1 Διαδικασία επιλογής

Η επιλογή είναι μία διαδικασία που καθορίζουμε πόσες φορές μία λύση θα επιλεγεί για αναπαραγωγή και συνεπώς τον αριθμό των απογόνων που θα δώσει. Κατά τη διαδικασία αυτή, αρχικά οι υποψήφιες λύσεις αντιγράφονται σε μια δεξαμενή ζευγαρώματος (mating pool). Σε αυτή αντιγράφονται μέλη του αρχικού πληθυσμού με πιθανότητα ανάλογη της καταλληλότητας τους. Κάποιοι γονείς με πολύ χαμηλή τιμή στη συνάρτηση καταλληλότητας ενδέχεται να επιλεγούν για αναπαραγωγή περισσότερες φορές (θυμίζουμε πως έχουμε ένα πρόβλημα ελαχιστοποίησης) από κάποιους άλλους γονείς με υψηλότερη τιμή. Για την επιλογή των χρωμοσωμάτων που θα αντιγραφούν στη δεξαμενή ζευγαρώματος χρησιμοποιούνται αρκετές τεχνικές όπως η τεχνική της ρουλέτας (roulette wheel selection), στοχαστική δειγματοληψία με αντικατάσταση (stochastic sampling with replacement) , τεχνική τουρνουά η πρωταθλήματος (Tournament selection).

Στο γενετικό αλγόριθμο χρησιμοποιήθηκε η διαδικασία επιλογής τύπου τουρνουά δύο ατόμων (Binary Tournament Selection).Η ιδέα της είναι αρκετά απλή:

Από τον πληθυσμό Π(τ) διαλέγουμε τυχαία δύο προγράμματα παραγωγής. Όποιο πρόγραμμα παραγωγής έχει καλύτερο makespan επιλέγεται και καταλαμβάνει μια θέση στον καινούργιο πληθυσμό Π(τ+1).

Η παραπάνω διαδικασία επαναλαμβάνεται POPSIZE φορές (POPSIZE = μέγεθος πληθυσμού = πλήθος δόκιμων λύσεων).

Εδώ επιθυμούμε οι καλές λύσεις να περνάνε στον επόμενο πληθυσμό.

Έστω ότι ένας πληθυσμός προγραμμάτων παραγωγής με POPSIZE=10 τη χρονική στιγμή τ είναι:

Εικόνα 10. Παράδειγμα πληθυσμού Γ.Α

	Λύσεις Π.Π	makespan
i=0	9 5 3 8 4 1 7 10 2 6	1977
i=1	9 5 6 1 8 3 2 10 7 4	1901
i=2	9 7 4 3 2 8 1 10 5 6	2024
i=3	9 3 6 8 5 2 1 10 7 4	2010
i=4	2 5 6 10 9 3 8 1 7 4	1924
i=5	9 5 2 3 8 1 6 10 7 4	1921
i=6	9 5 6 2 10 4 8 3 7 1	1885
i=7	8 5 3 4 7 10 9 1 2 6	1977
i=8	9 5 6 10 7 1 8 3 2 4	1921
i=9	9 5 2 10 7 1 8 3 6 4	1921
i=10	9 5 6 2 10 4 8 3 7 1	1885

→ 9 5 6 1 8 3 2 10 7 4 1901

Καλούμε τη γεννήτρια τυχαίων αριθμών γιά να μας βγάλει δύο αριθμούς από μηδέν μέχρι POPSIZE (τον αριθμό των ατόμων που αποτελούν το πληθυσμό). Έστω ότι μας βγάζει τον αριθμό 1 και 7. Οι δύο αυτοί αριθμοί δείχνουν κάποια συγκεκριμένα χρωμοσώματα (εικόνα 10).

Το χρωμόσωμα με τον αριθμό 1 είναι το

i=1	9 5 6 1 8 3 2 10 7 4	1901

ενώ το χρωμόσωμα με τον αριθμό 7 είναι το

i=7	8 5 3 4 7 10 9 1 2 6	1977

Ένα από τα δύο χρωμοσώματα θα επιζήσει και θα είναι αυτό που θα έχει χαμηλότερο makespan. Άρα το χρωμόσωμα i=1 θα επιζήσει και θα καταλάβει μία θέση στη πισινά ζευγαρώματος. Εφόσον ο πληθυσμός μας αποτελείται από δέκα άτομα θα χρειαστεί να κάνουμε τα παραπάνω άλλες εννέα φορές.

Τα προγράμματα παραγωγής τα οποία θα έχουν τη δυνατότητα να δημιουργήσουν νέες λύσεις μέσω των γενετικών τελεστών αναμένουμε να είναι αυτά τα οποία θα είναι τα πιό γρήγορα. Εδώ πρέπει να ειπωθεί ότι κάποιο πρόγραμμα παραγωγής μπορεί να αντιγραφεί περισσότερες από μία φορές (συνήθως αυτό γίνεται στα καλύτερα χρωμοσώματα) ενώ το χειρότερο με μεγάλη πιθανότητα=99,9%=100-(1/POP SIZE) χάνεται.

Πράγματι, μετά τη διαδικασία της επιλογής ο πληθυσμός είναι:

	Λύσεις Π.Π	makespan
i=0	9 5 6 10 7 1 8 3 2 4	1921
i=1	9 5 2 10 7 1 8 3 6 4	1921
i=2	9 5 6 1 8 3 2 10 7 4	1901
i=3	8 5 3 4 7 10 9 1 2 6	1977
i=4	9 5 2 10 7 1 8 3 6 4	1921
i=5	9 5 2 3 8 1 6 10 7 4	1921
i=6	9 5 6 10 7 1 8 3 2 4	1921
i=7	9 5 6 10 7 1 8 3 2 4	1921
i=8	9 5 2 3 8 1 6 10 7 4	1921
i=9	8 5 3 4 7 10 9 1 2 6	1977
i=10	9 5 6 2 10 4 8 3 7 1	1885

Στη τελευταία θέση της δομής δεδομένων αποθηκεύεται το καλύτερο πρόγραμμα παραγωγής του εκάστοτε πληθυσμού.

3.2.2 Διόρθωση Πληθυσμού

Όπως έχουμε αναφέρει ο γενετικός αλγόριθμος ξεκινά δημιουργώντας έναν αρχικό πληθυσμό. Ο πληθυσμός αυτός αποτελείται από POPSIZE προγράμματα παραγωγής. Στη συνέχεια επιλέγουμε τα καλύτερα μέλη του πληθυσμού τα οποία περνάνε μέσα από κάποιους γενετικούς τελεστές και δημιουργούνται νέες λύσεις.
Π.χ ένας αρχικός πληθυσμός είναι ο παρακάτω (εικόνα 11)

Εικόνα 11. Αρχικός πληθυσμός προγραμμάτων παραγωγής

i=0	9 2 7 10 1 5 6 8 3 4	1951
i=1	6 8 10 4 7 2 9 5 3 1	2025
i=2	9 6 8 3 2 1 7 4 10 5	1935
i=3	2 9 6 1 5 10 7 3 8 4	1940
i=4	3 1 7 4 2 5 6 9 8 10	2049
i=5	4 6 7 1 5 9 3 8 2 10	1939
i=6	3 1 8 2 10 7 6 5 4 9	2034
i=7	1 6 5 9 8 3 2 10 7 4	1925
i=8	6 3 4 7 8 5 9 1 2 10	2022
i=9	6 1 2 7 9 4 3 5 8 10	2031
i=10	1 6 5 9 8 3 2 10 7 4	1925

Η διαδικασία της επιλογής δίνει ζωή σε κάποια προγράμματα παραγωγής και τερματίζει κάποια άλλα. Κάποια καλά προγράμματα παραγωγής αντιγράφονται παραπάνω από μία φορές ενώ τα χειρότερα με πολύ μεγάλη πιθανότητα τερματίζονται.

Εικόνα 12 . Πισίνα ζευγαρώματος (mating pool)

i=0	9 2 7 10 1 5 6 8 3 4	1951
i=1	2 9 6 1 5 10 7 3 8 4	1940
i=2	1 6 5 9 8 3 2 10 7 4	1925
i=3	2 9 6 1 5 10 7 3 8 4	1940
i=4	6 3 4 7 8 5 9 1 2 10	2022
i=5	4 6 7 1 5 9 3 8 2 10	1939
i=6	9 2 7 10 1 5 6 8 3 4	1951
i=7	2 9 6 1 5 10 7 3 8 4	1940
i=8	9 6 8 3 2 1 7 4 10 5	1935
i=9	2 9 6 1 5 10 7 3 8 4	1940
i=10	1 6 5 9 8 3 2 10 7 4	1925

Ο παραπάνω πληθυσμός είναι η λεγόμενη πισίνα ζευγαρώματος.
Σε αυτά τα άτομα θα εφαρμοστούν κάποιοι γενετικοί τελεστές και θα προκύψει ο κάτωθι πληθυσμός (εικόνα 13):

Εικόνα 13. Ο πληθυσμός μετά την εφαρμογή γενετικών τελεστών

i=0	9 2 7 10 1 5 6 8 3 4	1951
i=1	2 9 6 1 5 10 7 3 8 4	1940
i=2	1 5 6 9 8 3 2 10 7 4	1909
i=3	2 9 6 3 5 10 7 1 8 4	1940
i=4	6 3 4 7 8 5 9 1 2 10	2022
i=5	4 6 7 1 5 9 3 8 2 10	1939
i=6	9 2 7 10 1 5 6 8 3 4	1951
i=7	2 5 6 1 9 10 7 3 8 4	1905
i=8	9 8 5 1 6 4 7 2 3 10	1936
i=9	2 9 6 1 5 10 7 3 8 4	1940
i=10	2 5 6 1 9 10 7 3 8 4	1905

Θυμίζουμε ότι στη τελευταία θέση της δομής δεδομένων (i=10) αποθηκεύουμε το καλύτερο πρόγραμμα παραγωγής του πληθυσμού. Το καλύτερο χρωμόσωμα του αρχικού μας πληθυσμού είχε makespan 1925 ενώ μόλις ολοκληρώθηκε μια επανάληψη το καλύτερο χρωμόσωμα του πληθυσμού έχει makespan 1905. Πρόεκυψε δηλαδή μια βελτίωση. Όσο προκύπτει βελτίωση σε κάθε επανάληψη δεν δημιουργείται κανένα πρόβλημα. Σιγά σιγά λόγω της πίεσης που ασκεί η διαδικασία της επιλογής θα φτάσουμε σε κάποιο πληθυσμό ο οποίος θα έχει πολλές αντιγραφές πολύ καλών προγραμμάτων γιατί σε αυτά επιτρέπεται να ζήσουν.

Στο τέλος θα μείνουμε με 1 ή το πολύ 2 πολύ καλά προγράμματα παραγωγής, δηλαδή ο αλγόριθμος θα «κολλήσει» σε ένα «βαθύ» τοπικό ελάχιστο και καθώς θα προσπαθεί να «σκαρφαλώσει» για να προσεγγίσει άλλα σημεία θα αυξάνεται ταυτόχρονα και το makespan. Το makespan λειτουργεί σαν «βαρίδι» και αναγκάζει το πληθυσμό να πέφτει πάλι μέσα στην ίδια τρύπα. Ο πληθυσμός μας λοιπόν έχει αποτύχει στο στόχο του καθώς όλα τα άτομα του πληθυσμού γίνανε κλώνοι ενός πολύ καλού (προσπαθώντας να επιβιώσουν…) σε σχέση με αυτά αλλά όχι και τόσο καλού σε σχέση με κάποια άλλα άτομα που δεν έχουν ακόμη δημιουργηθεί. Αυτός ο πληθυσμός λοιπόν είναι εν μέρει άχρηστος και το μόνο που μας μένει να κάνουμε είναι η μερική εξόντωση του. Αυτή είναι και η διόρθωση του πληθυσμού με πολύ απλά λόγια. Η ρουτίνα αυτή βοηθά ούτως ώστε ο αλγόριθμος να ανακάμπτει από τα τοπικά βέλτιστα.

Η διόρθωση του πληθυσμού έχει ως εξής:

Κρατάμε το 10% του πληθυσμού από τα καλύτερα χρωμοσώματα, ένα 50% του πληθυσμού προκύπτει από μετάλλαξη των 10% πρώτων χρωμοσωμάτων, και τέλος το 40% των χρωμοσωμάτων που απομένουν δημιουργούνται τυχαία από γεννήτρια τυχαίων αριθμών.

Εικόνα 14. Κατάταξη πληθυσμού

i=0	3 5 9 4 2 7 6 8 1 10	1747
i=1	3 5 6 2 4 9 7 10 1 8	1764
i=2	3 5 6 2 4 9 7 8 1 10	1764
i=3	3 5 9 1 8 7 6 4 2 10	1768
i=4	3 7 5 8 9 2 4 6 1 10	1938
i=5	3 6 9 4 8 7 5 2 1 10	1941
i=6	3 6 2 4 5 7 9 8 10 1	1945
i=7	3 9 4 1 5 7 6 8 2 10	1946
i=8	3 1 7 5 6 8 2 4 9 10	2027
i=9	3 10 8 5 7 4 2 6 9 1	2041
i=10	3 5 9 4 2 7 6 8 1 10	1747

Εικόνα 15. Διόρθωση πληθυσμού

i=0	3 5 9 4 2 7 6 8 1 10	1747
i=1	3 5 4 2 9 7 6 8 1 10	1763
i=2	3 5 9 2 7 6 8 1 4 10	1764
i=3	3 5 9 4 2 7 6 10 8 1	1749
i=4	5 9 4 2 7 6 8 1 10 3	1977
i=5	3 5 9 2 7 4 6 8 1 10	1758
i=6	3 5 9 4 2 7 6 8 1 10	1747
i=7	2 7 5 3 1 9 8 4 10 6	2033
i=8	3 2 5 1 7 9 4 6 10 8	1954
i=9	1 5 3 6 9 2 4 10 8 7	1982
i=10	3 5 9 4 2 7 6 8 1 10	1747

Μόλις ολοκληρωθεί η διόρθωση του πληθυσμού υπολογίζεται η τιμή της αντικειμενικής συνάρτησης των καινούργιων λύσεων που έχουν προκύψει.

Αξίζει να αναφερθεί ότι τέτοιου είδους μπλοκ εντολών (regeneration schemes) είναι ευρέως χρησιμοποιούμενα στους γενετικούς αλγορίθμους προκειμένου να αντιμετωπιστεί ένα σημαντικό μειονέκτημα τους: η προσκόλληση σε ένα τοπικό βέλτιστο του πεδίου λύσεων της συνάρτησης [38].

Τέτοιου είδους σφάλματα γίνονται σταδιακά κατά τη διάρκεια των επαναλήψεων του γενετικού αλγορίθμου και σχετίζονται με τη πίεση που ασκεί η διαδικασία της επιλογής πάνω στις υποψήφιες λύσεις. Μία διαδικασία επιλογής η οποία επιλέγει για αναπαραγωγή μονάχα τις καλύτερες λύσεις καταλήγει σε κάποιο Π(τ) που αποτελείται από μια ή το πολύ δύο λύσεις. Έτσι δεν υπάρχει η απαραίτητη ποικιλομορφία που θα οδηγήσει σταδιακά τον γενετικό αλγόριθμο σε κάποια προσέγγιση ολικού βέλτιστου.

Απεναντίας διαδικασίες επιλογής με χαλαρή πίεση έχουν το μειονέκτημα της αργής σύγκλισης.

Η ισορρόπηση των δύο αυτών κριτηρίων οδηγεί σε πολύ καλύτερα αποτελέσματα.

3.2.3 Γενετικοί Τελεστές

Οι γενετικοί τελεστές συμβάλουν στη δημιουργία νέων προγραμμάτων παραγωγής αξιοποιώντας ταυτόχρονα τη πληροφορία που έχει αποκτηθεί και βρίσκεται σε κάθε μέλος του πληθυσμού. Ο Γ.Α λοιπόν μετά από κάποιες επαναλήψεις αρχίζει και μαθαίνει στα άτομα του πληθυσμού να ταχτοποιούν τις παραγγελίες με βάση ποιά θα γίνει πρώτη, ποιά θα γίνει δεύτερη κ.ο.κ έτσι ώστε να επιτύχουν όσο το δυνατό μικρότερο makespan. Οι γενετικοί τελεστές θα πρέπει να προσέξουν να επιδρούν πάνω στα προγράμματα παραγωγής όσο το δυνατόν λιγότερο καθολικά, δηλαδή να επιφέρουν μικρές αλλαγές έτσι ώστε να μην αχρηστεύονται όλες αυτές οι επαναλήψεις του Γ.Α.

Ένας γενετικός τελεστής είναι η διασταύρωση (crossover) και για κάθε είδος προβλήματος πρέπει να εξευρεθεί με προσεκτικό τρόπο ποιό είδος διασταύρωσης ταιριάζει σε αυτόν. Η διασταύρωση χρειάζεται συνήθως δύο άτομα τα οποία προβαίνουν σε ανταλλαγή πληροφορίας. Ένας άλλος τελεστής είναι η μετάλλαξη (mutation) μέσα στην οποία αλλάζουμε θέση κάποιο στοιχείο του χρωμοσώματος. Επιπλέον δημιουργήθηκε και ένας νέος τελεστής ο οποίος ονομάζεται επικοινωνία (communication), είναι παρόμοιος με τη διασταύρωση αλλά παίρνει πληροφορίες από πολλά άτομα του πληθυσμού. Ας ξεκινήσουμε τη περιγραφή των γενετικών τελεστών ξεκινώντας από το τελευταίο που αναφέραμε.

Επικοινωνία πληθυσμού (communication)

Η συνάρτηση αυτή εφαρμόζεται στα χρωμοσώματα με βάση κάποια πιθανότητα όπως όλοι οι γενετικοί τελεστές εξάλλου. Χρησιμοποιεί πληροφορίες από περισσότερα μέλη του πληθυσμού σε σχέση με την απλή διασταύρωση δύο χρωμοσωμάτων. Παρακάτω θα χρησιμοποιήσουμε ένα παράδειγμα για την ευκολότερη περιγραφή αυτού του τελεστή.

Έστω ότι έχουμε ένα πληθυσμό αποτελούμενο από δέκα προγράμματα παραγωγής. Ο πληθυσμός αυτός αφού περάσει από τη διαδικασία της επιλογής γίνεται:

i=0	9 2 7 10 1 5 6 8 3 4	1951
i=1	2 9 6 1 5 10 7 3 8 4	1940
i=2	1 6 5 9 8 3 2 10 7 4	1925
i=3	2 9 6 1 5 10 7 3 8 4	1940
i=4	6 3 4 7 8 5 9 1 2 10	2022
i=5	4 6 7 1 5 9 3 8 2 10	1939
i=6	9 2 7 10 1 5 6 8 3 4	1951
i=7	2 9 6 1 5 10 7 3 8 4	1940
i=8	9 6 8 3 2 1 7 4 10 5	1935
i=9	2 9 6 1 5 10 7 3 8 4	1940
i=10	1 6 5 9 8 3 2 10 7 4	1925

Ο παραπάνω πληθυσμός είναι αυτός που αποκαλούμε πισίνα ζευγαρώματος (mating pool) και πάνω σε αυτό το πληθυσμό θα εφαρμοστούν οι διάφοροι γενετικοί τελεστές. Ας επικεντρωθούμε όμως εμείς στο τελεστή communication.

Αυτός λοιπόν έχει κάποια πιθανότητα εμφάνισης, έστω ότι η πιθανότητα επικοινωνίας (Pcommunication) είναι 0.15. Γιά κάθε άτομο του πληθυσμού καλούμε τη γεννήτρια τυχαίων αριθμών για να μας δώσει μία πιθανότητα η οποία θα συγκριθεί με το 0.15 (πιθανότητα επικοινωνίας) και αν είναι μικρότερη, το άτομο αυτό θα επιλεγεί και θα εφαρμοστεί επάνω του αυτή η συνάρτηση. Έστω λοιπόν ότι επιλέγεται το i=8 μέλος:

i=8 9 6 8 3 2 1 7 4 10 5 1935

Καλούμε πάλι τη γεννήτρια και μας βγάζει τυχαία το πλήθος των ατόμων που θα αλληλεπιδράσει το μέλος i=8, αυτή τη μεταβλητή τη λέμε friends (αυτή την ονομασία έχουμε και στο κώδικα του Γ.Α)
Η γεννήτρια έβγαλε:
friends=3
Μέχρι στιγμής ξέρουμε ότι το μέλος i=8 θα «συνομιλήσει» με τρία άτομα του πληθυσμού και θα αλλάξει τη σειρά κάποιων παραγγελιών στο πρόγραμμα παραγωγής του. Δεν γνωρίζουμε όμως ποιά είναι αυτά τα τρία. Απλούστατα αυτοί οι τρεις «φίλοι» προκύπτουν τυχαία από γεννήτρια.

<u>Γιά τη πρώτη φορά</u>

Καλούμε λοιπόν τη γεννήτρια όπου αυτή τη φορά μας βγάζει ένα αριθμό από 0 μέχρι το μέγεθος του πληθυσμού:

Friend : i=0 9 2 7 10 1 5 6 8 3 4

Τυχαία διαλέγουμε κάποια θέση στο πάνω χρωμόσωμα. Αυτό το λέμε point.
Point=4
Στην 4^η θέση στο χρωμόσωμα i=0 υπάρχει η παραγγελία 10, η οποία τώρα θα πάει στη 4^η θέση στο χρωμόσωμα i=8 ενώ η παραγγελία που ήταν στη 4^η θέση μετατίθεται στη θέση 9 όπου ήταν πριν η παραγγελία 10. Το χρωμόσωμα αυτό (i=8) θα γίνει:

i=8 9 6 8 10 2 1 7 4 3 5

Άρα λοιπόν το μέλος 8 «συνομίλησε» με το μέλος i=0 (το οποίο δεν αλλάζει καθόλου) και αντέγραψε από αυτό τη παραγγελία που βρίσκεται στην 4^η θέση.
Μας απομένουν άλλα δύο άτομα στα οποία θα κάνουμε την ίδια διαδικασία.

<u>Για τη δεύτερη φορά</u>

Friend : i=4 6 3 4 7 8 5 9 1 2 10

Point= 8
Στην 8^η θέση στο χρωμόσωμα i=4 υπάρχει η παραγγελία 1, η οποία τώρα θα πάει στην 8^η θέση στο χρωμόσωμα i=8 και το χρωμόσωμα αυτό θα γίνει :

i=8 | 9 6 8 10 2 4 7 1 3 5

Άρα λοιπόν το μέλος 8 «συνομίλησε» και με το μέλος i=4 (το οποίο δεν αλλάζει καθόλου) και αντέγραψε από αυτό τη παραγγελία που βρίσκεται στην $8^{η}$ θέση.
Μας απομένει άλλη μία «επικοινωνία»

Για τη τρίτη φορά

Friend : i=9   2 9 6 1 5 10 7 3 8 4

Point = 2
Στην $2^{η}$ θέση στο χρωμόσωμα i=9 υπάρχει η παραγγελία 9, η οποία τώρα θα πάει στην $2^{η}$ θέση στο χρωμόσωμα i=8 και το χρωμόσωμα αυτό θα γίνει:

i=8 | 6 9 8 10 2 4 7 1 3 5

Άρα λοιπόν το μέλος 8 «συνομίλησε» και με το μέλος i=9 (το οποίο δεν αλλάζει καθόλου) και αντέγραψε από αυτό τη παραγγελία που βρίσκεται στην $2^{η}$ θέση.

Συνολικά δηλαδή το χρωμόσωμα i=8 «επικοινώνησε» με τρία άλλα χρωμοσώματα και αντέγραψε τις θέσεις που βρίσκονται κάποιες παραγγελίες των άλλων χρωμοσωμάτων στο δικό του χρωμόσωμα.

Όλα τα μέλη του πληθυσμού εξετάζονται αν θα υποβληθούν σε αυτή τη διαδικασία με κάποια πιθανότητα, παρόμοια με αυτή της πιθανότητας διασταύρωσης ή μετάλλαξης. Η συνάρτηση αυτή είναι αποτελεσματική σε μικρότερες πιθανότητες συγκρινόμενη με τη πιθανότητα μετάλλαξης, έτσι ώστε να εισάγει μια απαραίτητη ποικιλομορφία αλλά ταυτόχρονα να μην καταστρέφει το γενετικό υλικό που έχει δημιουργηθεί στο πληθυσμό μέχρι εκείνη τη στιγμή.
Όσο μεγαλώνει η τάξη του προβλήματος (π.χ. περισσότερες παραγγελίες) χρειάζεται να αυξήσουμε σημαντικά το πληθυσμό των υποψήφιων λύσεων (POPSIZE) για να οδηγηθούμε σε ένα ικανοποιητικό βέλτιστο, κάτι που οδηγεί σε καθυστέρηση του αλγορίθμου. Ειδικά να αναφέρουμε ότι δοκιμάστηκε η περίπτωση των 60 παραγγελιών, χωρίς τη χρήση της ο πληθυσμός έπρεπε να μεγαλώσει σημαντικά για να δώσει τη καλύτερη τιμή και από δώδεκα τα μέλη του πληθυσμού γίναμε 25. Σε μια άλλη περίπτωση των 90 παραγγελιών το βέλτιστο σημείο που βρίσκει ο Γ.Α επιτυγχάνεται σε πληθυσμό δέκα ατόμων ενώ χωρίς τη συνάρτηση ο πληθυσμός έπρεπε να είναι κάπου στο είκοσι.
Στην ουσία αυτό που κάνει ένας μεγάλος πληθυσμός στον γενετικό αλγόριθμο είναι να αναζητά περισσότερες και διαφορετικές αλληλουχίες παραγγελιών. Κάτι παρόμοιο προσπαθεί να κάνει και ο τελεστής αυτός παίρνοντας πληροφορίες από μέλη του πληθυσμού.
Η χρησιμοποίηση της συνάρτησης αυτής μειώνει σημαντικά το μέγεθος του πληθυσμού του γενετικού αλγορίθμου όσο αυξάνεται η τάξη του προβλήματος.

Διασταύρωση (crossover)

Αυτό που προσπαθεί να καταφέρει η διασταύρωση είναι να δημιουργεί άλλα προγράμματα παραγωγής έτσι ώστε να μπορεί να συνεχίσει την αναζήτηση ο γενετικός αλγόριθμος. Η διασταύρωση λαμβάνει χώρα με μία πιθανότητα διασταύρωσης (PXOVER). Για κάθε άτομο που είναι μέλος της πισίνας ζευγαρώματος (mating pool) καλείται η γεννήτρια και δίνει μια πιθανότητα. Αυτή η πιθανότητα συγκρίνεται με τη πιθανότητα διασταύρωσης και αν είναι μικρότερη, τότε το συγκεκριμένο πρόγραμμα παραγωγής είναι διαθέσιμο για διασταύρωση. Έπειτα πρέπει να βρεθεί κάποιο άλλο άτομο για να γίνει η διασταύρωση. Αυτό το βρίσκουμε όπως το πρώτο. Έτσι λοιπόν τα δύο άτομα προβαίνουν σε ανταλλαγή πληροφορίας (ανταλλαγή γενετικού υλικού). Υπάρχουν πολλών ειδών τελεστές διασταύρωσης και σε κάθε περίπτωση ανάλογα με το είδος του προβλήματος κάποιοι αποδίδουν καλύτερα από κάποιους άλλους.

Εδώ χρησιμοποιήθηκε η διασταύρωση δύο σημείων (two point crossover) μεταξύ δυο μελών και το αποτέλεσμα έχει δύο απογόνους/προγράμματα παραγωγής.

Έστω ότι έχουμε δύο μέλη τα οποία έχουν επιλεγεί για διασταύρωση όπως ακριβώς περιγράψαμε παραπάνω.

Μέσω της γεννήτριας τυχαίων αριθμών επιλέγουμε τυχαία δύο σημεία

point 1 point 2

2	3	5	1	8	6	9	7	10	4

1	10	9	7	6	3	5	2	8	4

Έστω λοιπόν ότι επιλέγουμε το 2^{o} και το 8^{o} σημείο. Η πρώτη λύση που θα σχηματιστεί από τη διασταύρωση θα έχει τις ίδιες παραγγελίες μέχρι το σημείο 1 (point 1), και από το σημείο 2 (point 2) μέχρι το τέλος με το γονέα 1. Τις ενδιάμεσες παραγγελίες θα τις πάρει από το δεύτερο γονέα ως εξής:

Ψάχνουμε από το τέλος τον δεύτερο γονέα και όσες παραγγελίες επιτρέπονται εκχωρούνται στην 3^{η}, 4^{η}, 5^{η}, 6^{η}, 7^{η} θέση αντίστοιχα.

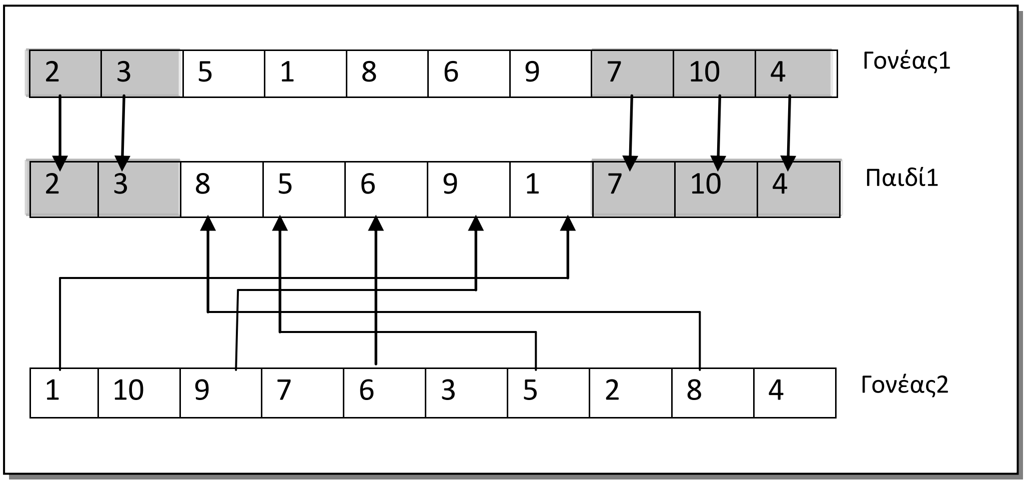
Με αυτή τη διαδικασία ο πρώτος απόγονος είναι ο (εικόνα 10):

2	3	8	5	6	9	1	7	10	4

Αντίστοιχα ο δεύτερος απόγονος είναι:

1	10	7	9	6	5	3	2	8	4

Εικόνα 16. Σχηματική περιγραφή διασταύρωσης



Προτιμήθηκε ο two-point crossover διότι έχει καλύτερη ικανότητα στο να κρατάει ακέραια μικρά κομμάτια μπλοκς. Στο πρόβλημα του χρονικού προγραμματισμού έχει αποδεδειχθεί πως κάτι τέτοιο οδηγεί σε καλύτερα αποτελέσματα.

Μετάλλαξη(mutation)

Η μετάλλαξη που χρησιμοποιείται στον αλγόριθμο είναι η απλή αντιμετάθεση δύο θέσεων.

Σύμφωνα με αυτήν όλα τα άτομα του πληθυσμού ελέγχονται αν θα μπουν στη διαδικασία της μετάλλαξης με βάση κάποια πιθανότητα μετάλλαξης και εάν ναι εκτελούνται τα παρακάτω:

Με τη βοήθεια της γεννήτριας βγάζουμε δύο τυχαίους αριθμούς στο διάστημα από 1 έως το σύνολο των παραγγελιών του προβλήματος. Σε αυτές τις δύο θέσεις αντιστοιχούν δύο παραγγελίες οι οποίες υποβάλλονται σε απλή αντιμετάθεση (εικόνα 17).

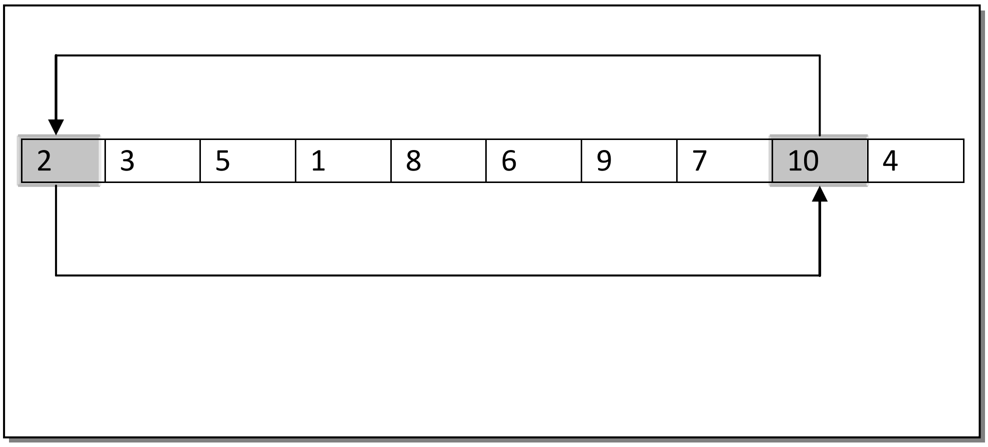
Π.χ. έστω η λύση:

2	3	5	1	8	6	9	7	10	4

Η λύση αυτή μετά τη διαδικασία της μετάλλαξης γίνεται:

10	3	5	1	8	6	9	7	2	4

Εικόνα 17. Σχηματική περιγραφή μετάλλαξης



4. Σύγκριση Γενετικού Αλγορίθμου με αλγόριθμο Τυχαίας Αναζήτησης

Στο παρόν κεφάλαιο θα προβούμε σε σύγκριση μεταξύ του Γενετικού Αλγορίθμου (ΓΑ) , και ενός αλγορίθμου Τυχαίας Αναζήτησης (ΤΑ) με την έννοια της τυχαίας δημιουργίας νέων λύσεων (προγραμμάτων παραγωγής) σε κάθε επανάληψη. Η σύγκριση αυτή θα διερευνήσει την αποτελεσματικότητα των γενετικών αλγορίθμων στον χρονικό προγραμματισμό εργασιών.
Ο υπολογιστής που χρησιμοποιήσαμε είναι το μοντέλο HP Pavilion dv6500 Notebook PC με επεξεργαστή Intel Core 2 Duo 2.40 GHZ και μνήμη RAM 2 GB.
Και οι δύο αλγόριθμοι χρησιμοποιούν την ίδια γεννήτρια τυχαίων αριθμών με τον ίδιο αρχικό σπόρο.
Η σύγκριση μεταξύ των δύο αλγορίθμων, ΓΑ και ΤΑ, θα γίνει στα επίπεδα των 10, 20, 30, 40 και 50 παραγγελιών που πρέπει να διεκπεραιωθούν σε όσο το δυνατόν λιγότερο χρονικό διάστημα. Οι παραγγελίες αυτές είναι γνωστές τη χρονική στιγμή τ=0 και από εκεί και πέρα δεν επιτρέπονται άλλες παραγγελίες να μπουν στο σύστημα παραγωγής. Παράλληλα, στα επίπεδα των παραγγελιών έχουν γίνει εσωτερικές ρυθμίσεις στον γενετικό αλγόριθμο όσον αφορά τις μεταβλητές:

1. Μέγεθος Πληθυσμού (POPSIZE)
2. Κριτήριο Τερματισμού (MAXGENS)
3. Πιθανότητα Διασταύρωσης (PXOVER)
4. Πιθανότητα Μετάλλαξης (PMUTATION)
5. Πιθανότητα Επικοινωνίας (PCOMMUNICATION)

έτσι ώστε να αποδίδει όσο το δυνατόν πιο βέλτιστα για τα δεδομένα ύψη των παραγγελιών.

<u>Επιπρόσθετα έχει δημιουργηθεί και ακόμη ένας αλγόριθμος ο οποίος βρίσκει όλες τις δυνατές αλληλουχίες των παραγγελιών</u> και υπολογίζει το makespan για κάθε πρόγραμμα παραγωγής. Ο αλγόριθμος αυτός εξερευνά όλα τα προγράμματα παραγωγής που είναι δυνατόν να υπάρξουν και έχει ως έξοδο το βέλτιστο πρόγραμμα παραγωγής, δηλαδή ένα ολικό βέλτιστο. Βεβαίως, ο αλγόριθμος αυτός δοκιμάστηκε μονάχα για δέκα παραγγελίες, μιας και η χρήση του σε μεγαλύτερα προβλήματα είναι απαγορευτική λόγω της παραγοντικής αύξησης των δυνατών συνδυασμών.
Αξίζει να αναφέρουμε ότι για δέκα παραγγελίες οι δυνατοί συνδυασμοί είναι 10!=3.628.800 ενώ για 20 παραγγελίες έχουμε 2.432.902.008.176.640.000 συνδυασμούς.
Για το ύψος των 10 παραγγελιών η λύση που βρέθηκε από το γενετικό αλγόριθμο είναι και η ολική βέλτιστη λύση του προβλήματος.

10 παραγγελίες με τρείς εργασίες η κάθε μία (10 Χ 3)

ΤΥΧΑΙΟΣ ΑΛΓΟΡΙΘΜΟΣ			ΓΕΝΕΤΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ	
MAKESPAN (Min)	CPU TIME (Sec)	ITERATIONS	MAKESPAN (Min)	CPU TIME (Sec)
1735	0,61	30.000	1731	2,23
1735	1,12	50.000		
1734	1,34	60.000		
1734	1,78	80.000		
1734	2,25	100.000		
1731	2,56	107.000		

Βέλτιστο makespan: 1731
Χείριστο makespan: 2099
<u>Ρυθμίσεις Γ.Α</u>
POPSIZE=10
MAXGENS=10
PXOVER=0.85
PMUTATION=0.25
PCOMMUNICATION=0.15
10Χ3=Δέκα παραγγελίες με τρεις εργασίες η κάθε μία

Όπως είπαμε, μόνο για τη περίπτωση των δέκα παραγγελιών μπορούμε να είμαστε σίγουροι ότι έχουμε βρει την ολική βέλτιστη λύση του προβλήματος. Εδώ και οι δύο αλγόριθμοι τα καταφέρνουν μια χαρά. Τόσο ο Γενετικός Αλγόριθμος όσο και ο αλγόριθμος Τυχαίας Αναζήτησης βρίσκουν το ελάχιστο makespan σε σχεδόν ίδιο χρόνο. Εδώ να αναφέρουμε ότι τα προγράμματα παραγωγής που καταφέρνουν να έχουν makespan ίσο με 1731 είναι πολλά. Ένα από αυτά είναι το

Εικόνα 18

10	5	4	6	2	7	8	3	1	9

Αυτό που έχει ενδιαφέρον είναι ότι οι πρώτες πέντε παραγγελίες πρέπει να είναι αυτές που έχουν επισημανθεί στην εικόνα 18 αν θέλουμε να επιτύχουμε το ελάχιστο makespan. Για τις υπόλοιπες παραγγελίες που μένουν δεν μας ενδιαφέρει σε ποιά σειρά θα μπουν. Αυτό σημαίνει ότι υπάρχουν 5! βέλτιστες λύσεις και μάλλον για αυτό το λόγο ο ΤΑ σε αυτή τη περίπτωση δεν τα πάει και άσχημα. Η περίπτωση των δέκα παραγγελιών έχει ενδιαφέρον γιατί είναι η μοναδική ανάμεσα σε αυτές που εξετάσαμε που σημειώνεται μια σχετική ισορροπία ανάμεσα στους δύο αλγορίθμους. Σε όλες τις άλλες περιπτώσεις η υπεροχή του γενετικού αλγόριθμου είναι προφανής τόσο από άποψη υπολογιστικού φόρτου όσο και από επιτυγχανόμενο makespan.

20 παραγγελίες (20Χ3)

ΑΛΓΟΡΙΘΜΟΣ ΤΥΧΑΙΑΣ ΑΝΑΖΗΤΗΣΗΣ			ΓΕΝΕΤΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ	
MAKESPAN	CPU TIME	ITERATIONS	MAKESPAN	CPU TIME
3946	2,5	50.000	3918	2,5
3931	4,39	100.000		
3931	8,72	200.000		
3931	13,14	300.000		
3927	22	500.000		
3927	87	2.000.000		
3920	130	3.000.000		
3920	219	5.000.000		

Βέλτιστο makespan: 3918
Χείριστο makespan: 4390

Ρυθμίσεις Γ.Α
POPSIZE=10
MAXGENS=10
PXOVER=0.85
PMUTATION=0.25
PCOMMUNICATION=0.09

Αναγκάσαμε τους δυο αλγόριθμους να τρέξουν στον ίδιο χρόνο (2,5 δευτερόλεπτα). Το πρόγραμμα παραγωγής του ΓΑ είναι γρηγορότερο από εκείνο του ΤΑ κατά μισή ώρα.
Ο αλγόριθμος Τυχαίας Αναζήτησης δυσκολεύεται να προσεγγίσει το βέλτιστο σημείο που βρίσκει ο Γενετικός Αλγόριθμος. Χρειάζεται να δοκιμάσει πέντε εκατομμύρια προγράμματα παραγωγής (κάτι που του κοστίζει τρεισήμισι λεπτά) για να προσεγγίσει αυτό που ανακαλύπτει ο Γενετικός Αλγόριθμος μέσα σε δυόμιση δευτερόλεπτα.

30 παραγγελίες (30Χ3)

ΑΛΓΟΡΙΘΜΟΣ ΤΥΧΑΙΑΣ ΑΝΑΖΗΤΗΣΗΣ			ΓΕΝΕΤΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ	
MAKESPAN	CPU TIME	ITERATIONS	MAKESPAN	CPU TIME
6229	1,29	20.000	6197	2,7
6222	3,24	50.000		
6222	6,65	100.000		
6217	13,04	200.000		
6208	25,97	400.000		
6208	52	800.000		
6202	130	2.000.000		
6198	1800	30.000.000		

Βέλτιστο makespan: 6197
Χείριστο makespan: 6671

<u>Ρυθμίσεις Γ.Α</u>
POPSIZE=10
MAXGENS=10
PXOVER=0.85
PMUTATION=0.25
PCOMMUNICATION=0.15

Από αυτό το επίπεδο παραγγελιών και κάτω τα πράγματα αρχίζουν και γίνονται πιό ξεκάθαρα. Ας αναφέρουμε ενδεικτικά ότι το βέλτιστο makespan που βρίσκει ο Γενετικός Αλγόριθμος μέσα σε δυόμιση δευτερόλεπτα, προσεγγίζεται από τον αλγόριθμο Τυχαίας Αναζήτησης μετά από τριάντα εκατομμύρια προσομοιώσεις προγραμμάτων παραγωγής και του κοστίζει χρονικά μισή ώρα.

40 παραγγελίες (40x3)

ΑΛΓΟΡΙΘΜΟΣ ΤΥΧΑΙΑΣ ΑΝΑΖΗΤΗΣΗΣ			ΓΕΝΕΤΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ	
MAKESPAN	CPU TIME	ITERATIONS	MAKESPAN	CPU TIME
8497	8,58	100.000	8477	2,8
8497	17	200.000		
8489	43	500.000		
8489	68	800.000		
8485	66	1.000.000		
8485	172	2.000.000		

Βέλτιστο makespan: 8477
Χείριστο makespan: 8958
<u>Ρυθμίσεις Γ.Α</u>
POPSIZE=10
MAXGENS=10
PXOVER=0.85
PMUTATION=0.25
PCOMMUNICATION=0.10

Ένα θετικό στοιχείο του Γενετικού Αλγορίθμου είναι ότι όσο και να αυξάνεται το πλήθος των παραγγελιών (τουλάχιστον μέχρι εκατό παραγγελίες), ο χρόνος που χρειάζεται για να υπολογίσει μία καλή βέλτιστη λύση είναι γύρω στα 2,5 με 3 δευτερόλεπτα. Αυτό το καταφέρνει υπολογίζοντας πολύ λιγότερα προγράμματα παραγωγής σε σύγκριση με τον αλγόριθμο Τυχαίας Αναζήτησης ο οποίος λόγω της τυχαιότητας του πηγαίνει τελείως στα «τυφλά». Απεναντίας, ο Γ.Α κάνει χρήση της πληροφορίας που έχει αποκτηθεί από το παρελθόν του (τα προγράμματα παραγωγής των προηγούμενων γενεών). Αυτή του η συμπεριφορά αποδεικνύεται πολύ χρήσιμη όσο αυξάνονται οι παραγγελίες και αυτό συμβαίνει διότι:
Όταν αυξάνονται οι παραγγελίες αυξάνεται παραγοντικά και ο εφικτός χώρος λύσεων του προβλήματος. Αυτό μας αναγκάζει να έχουμε δυο επιλογές για την αναζήτηση ενός όσο το δυνατόν βέλτιστου σημείου, η να συνεχίσουμε να ψάχνουμε στα «τυφλά» ή να βρούμε με κάποιο τρόπο το μονοπάτι εκείνο που οδηγεί σε συνδυασμό παραγγελιών με το χαμηλότερο makespan έτσι ώστε να αποφύγουμε περιττούς υπολογισμούς προγραμμάτων παραγωγής. Ένας Γενετικός Αλγόριθμος από τη φύση του λειτουργεί με βάση το δεύτερο τρόπο.

50 παραγγελίες (50X3)

ΑΛΓΟΡΙΘΜΟΣ ΤΥΧΑΙΑΣ ΑΝΑΖΗΤΗΣΗΣ			ΓΕΝΕΤΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ	
MAKESPAN	CPU TIME	ITERATIONS	MAKESPAN	CPU TIME
10801	2,18	20.000	10764	2,9
10789	10,8	100.000		
10778	22	200.000		
10778	37	300.000		
10778	53	500.000		
10778	90	800.000		
10778	108	1.000.000		

Βέλτιστο makespan: 10764
Χείριστο makespan: 11245
<u>Ρυθμίσεις Γ.Α</u>
POPSIZE=12
MAXGENS=10
PXOVER=0.85
PMUTATION=0.25
PCOMMUNICATION=0.15

Εδώ να σημειώσουμε ότι στον ίδιο χρόνο λειτουργίας των δύο αλγορίθμων, ο Γ.Α βρίσκει πρόγραμμα παραγωγής γρηγορότερο πάνω από μισή ώρα σε σχέση με τον αλγόριθμο Τυχαίας Αναζήτησης.

Η χρήση του Γενετικού Αλγορίθμου που προτείνεται αποβαίνει ιδιαίτερα αποτελεσματική στον προγραμματισμό της συγκεκριμένης παραγωγικής διαδικασίας, όταν μάλιστα οι περισσότερες εταιρίες αυτού του κλάδου βρίσκουν κάποιο «καλό» πρόγραμμα παραγωγής συνήθως με το «χέρι» ή με εμπειρικές μεθόδους. Η χρησιμοποίηση του προτεινόμενου Γενετικού Αλγορίθμου θα έχει ως αποτέλεσμα την μείωση του απαιτούμενου χρόνου ολοκλήρωσης των παραγγελιών αλλά και την αύξηση της παραγωγικής διαδικασίας. Ενδεικτικά να αναφέρουμε ότι για τις πενήντα παραγγελίες η διαφορά μεταξύ χειρότερου και καλύτερου makespan είναι οχτώ ώρες.

5. Συμπεράσματα

Στην εργασία αυτή αναπτύχθηκε ένας γενετικός αλγόριθμος για την εύρεση του βέλτιστου προγράμματος εκτέλεσης παραγγελιών σε μία γραμμή παραγωγής κλωστοϋφαντουργίας με 2 στάδια παραγωγής και παράλληλες μηχανές σε κάθε στάδιο.

Στον γενετικό αλγόριθμο, εκτός από τους κλασικούς τελεστές μετάλλαξης και διασταύρωσης, αναπτύχθηκε ένας ακόμη πρωτότυπος τελεστής ο οποίος επέτρεψε την επίλυση προβλημάτων με πολλές παραγγελίες χρησιμοποιώντας μικρό πλήθος δόκιμων λύσεων.

Τα πειραματικά αποτελέσματα έδειξαν ότι ο γενετικός αλγόριθμος υπερτερεί ως προς την ταχύτητα έναντι ενός αλγορίθμου πλήρους αναζήτησης επιτυγχάνοντας ίδια ή παραπλήσια ποιότητα λύσεων, και επίσης υπερτερεί έναντι ενός αλγορίθμου τυχαίας αναζήτησης τόσο ως προς τη ταχύτητα όσο και ως προς την ποιότητα των λύσεων.

Αντικείμενα μελλοντικής έρευνας θα μπορούσαν να είναι η ανάπτυξη παρόμοιων αλγορίθμων για προβλήματα με περισσότερα στάδια παραγωγής.

6. Βιβλιογραφία

[1] R. Conway, W. Maxwell and L. Miller. *Theory of scheduling*. AddisonWesley,1967.
[2] Chen, C.L., Neppalli, R.V., Aljaber, N. “Genetic algorithms applied to the continuous flow shop problem,” *Computers and Industrial Engineering* 30: (4), 1996, 919-929.
[3] Cleveland, G.A., Smith, S.F. “Using genetic algorithms to schedule flow shop,” Proceedings of 3rd Conference on Genetic Algorithms, Schaffer, D.(ed.), San Mateo: Morgan Kaufmann Publishing, 1989, 160-169.
[4] J. Carlier and E. Neron. An exact method for solving the multiprocessor FlowShop. *RAIRO – Recherche Operationnelle*, 34(1) :125,2000.
[5] Eitler, O., Toklu, B., Atak, M., Wilson,J. “A genetic algorithm for flowshop scheduling problems,” J. Oper. Res. Soc., 55, 2004, 830-835.
[6] L. Davis. JobShop scheduling with genetic algorithms. In *International Conference on Genetic Algorithms and their Applications*, pages 136140.Lawrence Erlbaum Associates, 1985.
[7] A. Doye, O. Engin and C. Ozkan. A new artificial immune system approach to solve permutation flowshop scheduling problem. In *Turkish Symposium on Artificial Intelligence and Neural Networks –* TAINN'03, volume 1, 2003.
[8] Βασίλης Σ. Κουικόγλου. Προγραμματισμός Παραγωγής
[9] Framinan, J.M, Ruiz-Usano, R., Leisten, R. “Sequencing CONWIP flow-shops: analysis and heuristic,” Int. J. Prod. Res., 39, 2001, 2735-2749.
[10] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A.H.G. Rinnooy Kan.Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of Discrete Mathematics*, volume 5, pages 287326. 1979.
[11] Grabowski, J., Wodecki, M. “A very fast tabu search algorithm for the permutation flowshop problem with makespan criterion,” Comput. Oper. Res., 31, 2004, 1891-1909.
[12] M.J.M. Heijligers and J.A.G. Jess. Highlevel synthesis scheduling andallocation using genetic algorithms based on constructive topological scheduling techniques. In *2nd IEEE International Conference on Evolutionary Computation ICEC'95*, pages 6166, 1995.
[13] Hong, T.P., Chuang, T.N. “Fuzzy scheduling on two-machine flow shop,” Journal of Intelligent & Fuzzy Systems, 6: (4), 1998, 471-481.
[14] Hong, T.P., Chuang, T.N. “Fuzzy CDS scheduling for flow shops with more than two machines,” Journal of Intelligent & Fuzzy Systems, 6: (4), 1998, 471-481.
[15] Hong, T.P., Chuang, T.N. “Fuzzy Palmer scheduling for flow shops with more than two machines,” Journal of Information Science and Engineering, Vol.15, 1999, 397-406.
[16] Hong, T.P., Wang, T.T. “A heuristic Palmer-based fuzzy flexible flow-shop scheduling algorithm,” Proceedings of the IEEE International Conference on Fuzzy Systems, Vol. 3, 1999, 1493 1497.
[17] Hong, T.P., Huang, C.M., Yu, K.M. “LPT scheduling for fuzzy tasks,” Fuzzy Sets and Systems, Vol. 97, 1998, 277-286.
[18] M. Haouari and T. Ladhari. A branch and bound based local search for the flowshop problem. *Journal of the Operational Research Society*, 54(10): 10761084, 2003.
[19] Ignall, E., Schrage, L.E. “Application of the branch and bound technique to some flow shops scheduling problems,” Operations Research, Vol. 13: (3), 1965, 400-412.
[20] H. Ishibuchi, T. Yoshida, and T. Murata. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transaction on Evolutionary Computation, 7(2): 204223, 2003.*
[21] W.H. Kohler and K. Steiglitz. Exact, approximate, and guaranteed accuracy algorithms for the flowshop problem n/2/F/F. *Journal ACM*, 22(1): 106114, 1975.
[22] CY. Lee, L. Lei, and M. Pinedo. Current trends in deterministic scheduling. *Annals of Operations Research*, 70: 141, 1997.
[23] E. Nowicki and C. Smutnicki. A fast tabu search algorithm for the flow shop problem. *European Journal of Operational Research*. 91: 160175, 1996.
[24] M. Pinedo. *Scheduling: Theory, Algorithms and Systems*. PrenticeHall, 1995.

[25] S. Parthasarathy and C. Rajendran. Scheduling to minimize mean tardiness
and weighted mean tardiness in flowshop and flowlinebased manufacturing cell. *Computers and Industrial Engineering*, 34(2): 531546, 1998.
[26] R. Z. Rios Mercado and J. F. Bard. Computational experience with a branch and cut
algorithm for flowshop scheduling with setups. *Computers and Operations Research*, 25(5): 351366, 1998.
[27] R. Z. RiosMercado and J. F. Bard. A branchandbound algorithm for permutation flow shops with sequencedependent setup times. *IIE Transactions*, 31(8): 721731, 1999.
[28] Liu, B., Wang, L., Jin, Y-H. "An effective PSObased memetic algorithm for flow shop scheduling,
"IEEE T. Syst. Man. CY. B.," 37, 2007, 18-27.
[29] S. Sayin and S. Karabati. A bicriteria approach to the twomachine flow shop scheduling problem. *European Journal of Operational Research*, (113): 435449, 1999.
[30] T. Stutzle. An ant approach for the flow shop problem. In *Proceedings of
the 6th European Congress on Intelligent & Soft Computing (EUFIT'98)*, volume 3, pages 15601564. VerlagMainz, 1998.
[31] Nawaz, M., Enscore Jr. E., Ham, I. "A heuristic algorithm for the m-machine, n-job flow-shop
sequencing problem," Omega-Int. J. Manage. S., 11, 1983, 91-95.
[32] Ogbu, F.A., Smith, D.K. "The application of the simulated annealing algorithm to the solution of the n/m/Cmax flowshop problem," Comput. Oper. Res., 17, 1990, 243-253.
[33] Onwubolu, G.C., Davendra, D. "Scheduling flowshopsusing differential evolution algorithm," Eur. J.Oper. Res., 171, 2006, 674-692.
[34] Osman, I., Potts, C. "Simulated annealing for permutation flow shop scheduling," OMEGA, 17,
1989, 551-557.
[35] Pan, C.H. "A study of integer programming formulations for scheduling problems,"
International Journal of System Science, 28: (1), 1997, 33-41
[36] KC, Ying and CJ. Liao. An ant colony system for permutation flowshop sequencing. *Computers and Operational Research*, 31(5): 791801, 2004.
[37] Z. Yong and N. Sannomiya. An improvement of genetic algorithms by search space reductions in solving largescale flowshop problems. *EEJ Transactions on Electronics, Information and Systems*, 121C(6): 10101016, 2001.
[38] Ruiz, R., Maroto, C., Alcaraz, J. "Two new robust genetic algorithms for the flowshop scheduling
problem," Omega-Int. J. Manage. S., 34, 2006, 461- 476.
[39] Stutzle, T. "Applying iterated local search to the permutation flowshop problem," AIDA-98-04, TU Darmstadt, FG Intellektik, 1998.
[40] Taillard, E. "Some efficient heuristic methods for the flow shop sequencing problem," Eur. J. Oper. Res., 47, 1990, 65-74.
[41] Taillard, E. "Benchmarks for basic scheduling problems," Eur. J. Oper. Res., 64, 1993, 278-285.
[42] Wang, C.G., Chu, C.B., Proth, J.M. "Efficient heuristic and optimal approaches for N/2/F/SIGMAC- I scheduling problems," International Journal of Production Economics, 44: (3), 1996, 225-237.
[43] Ying, K.C., Liao, C.J. "An ant colony system for permutation flow-shop sequencing," Comput. Oper. Res., 31, 2004, 791-801.
[44] Hong, T.P., Wang, C.L., Wang, S.L. "A heuristic Gupta-based flexible flow-shop scheduling
algorithm," Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Vol.
1, 2000, 319-322.
[45] Lai, T.C. "A note on heuristics of flow-shop scheduling," Operations Research, 44: (6), 1996,
648-652
[46] Lomnicki, Z. "A branch and bound algorithm for the exact solution of three machine scheduling
problem," Operational Research Quarterly, 16: (1), 1965, 89-100.
[47] Zobolas, G.I., Tarantilis, C.D., Ioannou, G. "Minimizing makespan in Permutation Flow Shop
scheduling problems using a hybrid metaheuristic algorithm," Computers and Operations Research,
2008, doi:10.1016/j.cor.2008.01.007.
[48] McMahon, C.B., Burton, P.G. "Flow-shop scheduling with the branch and bound method,"
Operations Research, 15: (3), 1967, 473-481
[49] Pan, C.H., Chen, J.S. "Scheduling alternative operations in two-machine flow-shops," Journal of
the Operational Research Society, 48: (5), 1997, 533-540.
[50]R. Z. RiosMercado and J. F. Bard. A branchandbound algorithm for permutation flow shops with sequencedependent setup times. *IIE Transactions*, 31(8): 721731, 1999.

[51]W.H. Kohler and K. Steiglitz. Exact, approximate, and guaranteed accuracy algorithms for the flowshop problem n/2/F/F. *Journal ACM*, 22(1): 106114, 1975.
[52] Lee, G.C., Kim, Y.D., Choi, S. W. “Bottleneckfocused scheduling for a hybrid flow-shop,” Int. J. Prod. Res., 42, 2004, 165-181.
[53] Ravendran, C. “Heuristic for scheduling in flowshop with multiple objectives,” Eur. J. Oper. Res., 82, 1995, 540-555.
[54] Allaxverdi, A., Gupta, J.N.D., Aldowaisan, T., 1999 . A review of scheduling research involving setup considerations. OMEGA, The international Journal of Management Science 27 ,219-239.
[55] Dudek , R.A ., Panwalkar, S.S ., Smith , M.L., 1992. The lessons of flowshop scheduling research. Operations Research 40, 7-13
[56] Linn, R ., Zhang,. W., 1999 . A Hybrid flowshop scheduling: A survey. Computers and Industrial Engineering 37, 57-61
[57] Mac Carthy, B.L., Liu, J ., 1993. Adressing the gap in scheduling research: A review of optimization and heuristic methods in production scheduling. International Journal of production Research. 31 (1) , 59-79
[58]Ledbetter, W.N., Cox , J.F., 1997. Operations research in production management. An investigation of past and present utilization. Production and Inventory Management 18 (3), 84-91.
[59]Schaller, J., Gupta, J.N.D., Vakharia, A.J., 2000. Scheduling a flowline manufacturing cell with sequence dependent family setup times. European Journal of Operational Research 125, 324-339.
[60]Gendreau, M., Laporte, G., Guimarães, E.M., 2001. A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times. European Journal of Operational Research 133, 183-189.
[61]Kim, D.W., Kim, K.H., Jang, W., Chen, F.F., 2002. Unrelated parallel machine scheduling with setup times using simulated annealing. Robotics and Computer Integrated Manufacturing 18, 223-231.
[62]Andrés et al. (2005a) addressed the problem of product grouping in the tile industry and stated that the problem can be modeled as a three-stage hybrid flow shop with separate and sequence-dependent setup times. They pointed out that the objective for such a problem is to minimize the changeover (setup) time in order to reduce the production Time.
[63]McKay, K.N., Pinedo, M. and Webster, S., Practice-focused research issues for scheduling systems. Production and Operations Management. v11 i2. 249-258.
[64] Jeffery K.Cochran, Shwu-Min Horng, John W.Fowler. A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines. Computers and Operation Research 30 (2003) 1087-1102
[65] Colin R. Reeves. A genetic algorithm for flowshop sequencing. Computer and Operation Research, Vol. 22, No. 1, pp. 5-13,1995
[66] Tadahiko Murata, Hisao Ishibuchi , Hideo Tanaka. Genetic algorithms for flowshop scheduling problems. Computers ind. Engineering Vol 30, No. 4, pp. 1061-1071 1996
[67] Victor Yaurima, Larisa Burtseva, Andrei Tchernykh. Hybrid Flowshop with unrelated machines, sequence dependent setup time, Computers and Industrial Engineering(2008), doi: 10.1016/j.cie.2008.09.004

7. Παράρτημα

Α. Κώδικας αλγορίθμου Τυχαίας Αναζήτησης

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

double Z,IE;
long int XHI,XALO,LEFTLO,FHI,K;
int
J,O,NITER,M,EV,T,JCC,C,NT,NAVT,NAVMT,NC,NAVC,NAVMC,NMT,NMC,NO,NJ[100],SEQ[100],BEST[100],WORST[100],
PR[10],PRC[10][3],PRT[3][10],TC[3],SC[3],OC[3],OT[10],JC[3],JT[10],CT[10],ST[10],NET[10],NEC[3];
float TJT[100][4],TJC[100][4],SETUP[10][10],TCLEAR;
int NC=2;
double BIG=10000000,BIG1=9999999;
double CLOCK,TEC[3],TET[10],CLOCKNEW,BCLOCK,WCLOCK;

INITIALIZE();
NEXTEV();
TFINISH(int);
TSTART(int);
CFINISH(int);
CARRIVE(int);
eye();

void main (void)
{
 int  i,j,iter,l;
 FILE *ptr1,*ptr2,*ptr3;
 clock_t start, end;
        double cpu_time_used;

 ptr1=fopen("fanco1.txt","r");
 ptr2=fopen("fanco2.txt","w");
 ptr3=fopen("fanco3.txt","w");

 if(ptr1==NULL)
 printf("problem opening fanco1.txt");
 if(ptr2==NULL)
 printf("problem opening fanco2.txt");
 eye();

 printf("how many iterations?");
 scanf("%d",&iter);
 printf("initial seed?");
 scanf("%lf",&IE);
 BCLOCK=BIG;
 WCLOCK=0;
 start = clock();
 for(l=0;l<iter;l++)
 {

        INITIALIZE();
 start:
        NEXTEV();
        if(CLOCKNEW>BIG1)
        goto end3;
        CLOCK=CLOCKNEW;

 if(EV==1)
 {
        if(NET[M]==1)
        TFINISH(M);
        else if(NET[M]==2)
        TSTART(M);
 }

 else
 {
```

```
        if(NEC[M]==1)
        CFINISH(M);
        else if(NEC[M]==2)
        CARRIVE(M);

 }

 goto start;


 end3:

 if(CLOCK<BCLOCK)
 {
 BCLOCK=CLOCK;
 for(i=1;i<=NO;i++)
 BEST[i]=SEQ[i];
 }

 if(CLOCK>WCLOCK)
 {
 WCLOCK=CLOCK;
 for(i=1;i<=NO;i++)
 WORST[i]=SEQ[i];
 }

 randorder();


}
 fprintf(ptr3,"\n\n\nTHE BEST SEQUENCE IS==>");
 for(i=1;i<=NO;i++)
 fprintf(ptr3,"%d ",BEST[i]);
 fprintf(ptr3,"\nWITH MAKESPAN=%lf",BCLOCK);

 end = clock();
 cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
 fprintf(ptr3,"\nCPU TIME=%lf",cpu_time_used);


}
INITIALIZE()
{

 int i,j,MIN,MAX,ITER;

MAX=0;
MIN=MAX;

 CLOCK=0.0;
 NAVT=NT;
 NAVC=NC;
 NAVMC=NMC;
 NAVMT=NMT;
 TC[1]=0;/*POU BRISKONTAI TA CAM,SE POIO TRAPEZI PANW*/
 TC[2]=NT+1;

 for(i=1;i<=NC;i++)/*ARXIKOPOIHSH CAM*/
 {
        SC[i]=0;
        TEC[i]=BIG;
        OC[i]=0;
        JC[i]=0;
 }

 for(i=1;i<=NT;i++)/*ARXIKOPOIHSH TABLES*/
 {
        TET[i]=BIG;
        CT[i]=0;
        ST[i]=0;
        OT[i]=0;
        JT[i]=0;

 }

 J=0; /*STHN ARXH DEN EXOUME FORTWSEI JOB ARA OUTE ORDER*/
 O=0;
```

```
 ITER=1;

again:
 T=PR[ITER];/*TO  TABLE POU THA STRWSOUME TO PERNOUME APO PRIORITI LIST*/
 ST[T]=1;/*MOLIS BRHKAME ENA TABLE SUMFWNA ME PRIORITI LIST*.H KATASTASH TOU GINETAI 1*/



 if(J==0 || J==(NJ[SEQ[O]]))/*AN DEN EXOUME FORTWSEI KANENA JOB H EXOUME TELEIWSEI OLA TA JOBS THS
 SYGKEKRIMENHS PARAGGELIAS*/
 {
        J=1;
        O=O+1;
        JT[T]=1;
 }
 else    /*BRISKOMASTE STHN IDIA PARAGGELIA*/
 {
        J=J+1;
        JT[T]=J;
 }

 OT[T]=SEQ[O];
 TET[T]=CLOCK+ TJT[OT[T]][JT[T]];
 NET[T]=1;/*MOLIS BRIKAME TRAPEZI GIA NA STRWTHEI ARA MPOREI NA MPEI STO TFINISH*/
 NAVMT=NAVMT-1;/*DESMEYOUME ERGATES*/
 NAVT=NAVT-1;

 if(ITER==1)
 {
        C=PRC[T][1];/*TO CAM POU THA KOPSEI PANW APTO TRAPEZI POU STRWTHIKE TO PAIRNOUME APO PRIORITI LIST*/
        SC[C]=2;/*MOLIS BRIKAME TO CAM POU THA KOPSEI SUMFWNA ME PRIORITY LIST. H KATASTASH TOU GINETAI 2*/
        CT[T]=C;
        TEC[C]=CLOCK + SETUP[TC[C]][T];
        NEC[C]=2;/*EIDOS EPOMENOU GEGONOTOS STO CAM*/
        TC[C]=T;
        OC[C]=OT[T];
        JC[C]=JT[T];
        NAVMC=NAVMC-1;
        NAVC=NAVC-1;
        JCC=3-C;
 }
 else if (NAVC>0&&NAVMC>0)/*AMA YPARXOUN CAM */
 {

        if(TC[JCC]<T)
        {
               MIN=TC[JCC];
                MAX=T;
        }
        else if(TC[JCC]>T)
        {
               MIN=T;
               MAX=TC[JCC];
        }
        else
        {
               MIN=T;
               MAX=T;
        }



        if(TC[C]<MIN||TC[C]>MAX)/*KAI MPOROUN NA PANE*/
        {
               SC[JCC]=2;
               CT[T]=JCC;
               TEC[JCC]=CLOCK + SETUP[TC[C]][T];
               NEC[JCC]=2;
               TC[JCC]=T;
               OC[JCC]=OT[T];
               JC[JCC]=JT[T];
               NAVMC=0;
               NAVC=0;
        }

 }
 ITER=ITER+1;
```

```
 if ((O<(NO)||J<(NJ[SEQ[O]]))&&NAVT>0&&NAVMT>0)
         goto again;

 }


NEXTEV()
{

 int i,j;

 CLOCKNEW=BIG;
 EV=0;
 M=-1;
 for(i=1;i<=NT;i++)
 {
         if(TET[i]<CLOCKNEW)
         {
                 CLOCKNEW=TET[i];
                 EV=1;
                 M=i;
         }
 }

 for(j=1;j<=NC;j++)
 {
         if(TEC[j]<CLOCKNEW)
         {
                 CLOCKNEW=TEC[j];
                 EV=2;
                 M=j;
         }
 }



}

TFINISH(int T)/*TRAPEZI STRWTHIKE*/
{
 int i,j,MIN,MAX;
 MAX=0;
 MIN=MAX;


 ST[T]=2;
 C=CT[T];

 if(C>0&&SC[C]==3)/*AN YPARXEI CAM KAI EINAI DIATHESIMO DOULEUEI*/
 {
         SC[C]=1;/*ARXIZEI NA DOULEUEI TO CAM PANW APTO TRAPEZI POU MOLIS STRWTHIKE*/
         TEC[C]=CLOCK + TJC[OT[T]][JT[T]];
         NEC[C]=1;/*NEXT EVENT CAM=CAM FINISH*/
         TET[T]=TEC[C]-TCLEAR;
         NET[T]=2;

 }
 else if(C==0||SC[C]==2)/*DEN YPARXEI CAM H YPARXEI ALLA EINAI APASXOLHMENO*.
 {
         TET[T]=BIG;/*TO TRAPEZI EINAI STRWMENO XWRIS CAM.APROSDIORISTOS O XRONOS POU THA EINAI DIATHESIMOS*/
 }

 NAVMT=NAVMT+1;/*AFOU STRWTHIKE TABLE*/


 for(i=1;i<=NT;i++)/*PSAXNOUME NA BROUME TO TRAPEZI POU THA STRWSOUME*/
 {
         T=PR[i];

         if(ST[T]==0)/*TO TRAPEZI EINAI DIATHESIMO GIA STRWSIMO*/
         {
                 if(O==(NO)&&(J==(NJ[SEQ[O]])))
                 {
                         ST[T]=0;
                         TET[T]=BIG;
                         OT[T]=0;
                         JT[T]=0;
```

```
			}
			else/*NEW JOB STARTS*/
			{

				if(J==(NJ[SEQ[O]]))
				{
					J=1;
					O=O+1;
					JT[T]=1;
				}
				else if(J<(NJ[SEQ[O]]))
				{
					J=J+1;
					JT[T]=J;
				}

				OT[T]=SEQ[O];/*POIA EINAI H PARAGGELIA POU THA  KANEI TWRA*/
				ST[T]=1;
				TET[T]=CLOCK +TJT[OT[T]][JT[T]];
				NET[T]=1;/*EIDOS EPOMENOU GEGONOTOS STO TABLE=TABLEFINISH*/
				NAVMT=NAVMT-1;
				NAVT=NAVT-1;
				CT[T]=0;
				for(j=1;j<=NC;j++)
				{
					C=PRC[T][j];
					if(TC[C]<T)
					{		/*TO C EINAI TO CAM POU RWTAEI AN MPOREI NA PAEI STO T*/
					MIN=TC[C];
					MAX=T;
					}
					else if(TC[C]>T)
					{
					MIN=T;
					MAX=TC[C];
					}
					else
					{
					MIN=T;
					MAX=T;
					}


					if(SC[C]==0&&NAVMC>0&&(TC[3-C]<MIN||TC[3-C]>MAX))
					{

						SC[C]=2;/*DESMEYSH CAM GIA NA PAEI STO T*/
						CT[T]=C;
						TEC[C]=CLOCK+SETUP[TC[C]][T];
						NEC[C]=2;/*TO DESMEUSAME GIA NA PAEI ARA NEXTEVENT=CARRIVE*/
						TC[C]=T;
						OC[C]=OT[T];
						JC[C]=JT[T];

					}
				}




				goto end1;/*AN BREIS DIATHESIMO TRAPEZI KAI YPARXEI JOB NA KANEIS  MHN PSAXNEIS 
ALLO*/

			}




		}



	}
```

```
end1:
 return 0;
}

TSTART(int T)/*TRAPEZI DIATHESIMO GIA STRWSIMO*/
{
 int i,j,MIN,MAX;
 MAX=0;
 MIN=MAX;

 if (NAVMT==0||(O==(NO)&&J==NJ[SEQ[O]]))
 {
         ST[T]=0;
         TET[T]=BIG;
         OT[T]=0;
         JT[T]=0;
         NAVT=NAVT+1;
 }
 else
 {
         if(J==(NJ[SEQ[O]]))
         {
                 J=1;
                 O=O+1;
                 JT[T]=1;
         }
     else if(J<(NJ[SEQ[O]]))
         {
                 J=J+1;
                 JT[T]=J;
         }
         OT[T]=SEQ[O];
         ST[T]=1;
         TET[T]=CLOCK+TJT[OT[T]][JT[T]];
         NET[T]=1;
         NAVMT=NAVMT-1;
         CT[T]=0;
         for(i=1;i<=NC;i++)
         {
                 C=PRC[T][i];/*TO CAM C RWTAEI AN MPOREI NA PAEI STO T*/
                 if(TC[C]<T)
                 {
                 MIN=TC[C];
                 MAX=T;
                 }
                 else if(TC[C]>T)
                 {
                 MIN=T;
                 MAX=TC[C];
                 }
                 else
                 {
                 MIN=T;
                 MAX=T;
                 }


                 if(SC[C]==0&&NAVMC>0&&(TC[3-C]<MIN||TC[3-C]>MAX))
                 {

                         SC[C]=2;
                         CT[T]=C;
                         TEC[C]=CLOCK + SETUP[TC[C]][T];
                         NEC[C]=2;
                         TC[C]=T;
                         OC[C]=OT[T];
                         JC[C]=JT[T];
                 }
         }

 }

}

CFINISH(int C)
{
 int i,j,MIN,MAX;
```

```
 for (i=1;i<=NT;i++)
 {
        T=PRT[C][i];
        if(TC[C]<T)
        {                               /*TO C EINAI POU RWTAEI AN MPOREI NA PAEI STO T*/
                MIN=TC[C];
                MAX=T;
        }
        else if(TC[C]>T)
        {
                MIN=T;
                MAX=TC[C];
        }
        else
        {
        MIN=T;
        MAX=T;
        }

        if((ST[T]>0&&(CT[T]==0))&&(TC[3-C]<MIN||TC[3-C]>MAX))/*NA EINAI STRWMENO TO TABLE,NA MHN YPARXEI CAM
KAI NA MPOREI NA PAEI*/
        {

                SC[C]=2;/*TO CAM DESMEYETAI GIA NA PAEI SE TRAPEZI*/
                CT[T]=C;
                TEC[C]=CLOCK + SETUP[TC[C]][T];
                NEC[C]=2;
                TC[C]=T;
                OC[C]=OT[T];
                JC[C]=JT[T];
                goto end;/*AFOU BRIKE TRAPEZI KAI CAM FEYGEI GIA NEXTEVENT*/
        }
 }
 SC[C]=0;/*AN DE BREI TRAPEZI GIA NA PAEI TO CAM KOIMATAI*/
 TEC[C]=BIG;
 if(NMC<NC)
 {
        JCC=3-C;
        if(SC[JCC]==0)
        {
                for(i=1;i<=NT;i++)
                {
                        T=PRT[JCC][i];
                        if(TC[JCC]<T)
                        {
                                MIN=TC[JCC];
                                MAX=T;
                        }
                        else
                        {
                                MIN=T;
                                MAX=TC[JCC];
                        }

                        if(ST[T]>0&&CT[T]==(0)&&(TC[C]<MIN||TC[C]>MAX))
                        {
                                SC[C]=2;
                                CT[T]=C;
                                TEC[C]=CLOCK + SETUP[TC[C]][T];
                                NEC[C]=2;
                                TC[C]=T;
                                OC[C]=OT[T];
                                JC[C]=JT[T];
                                goto end;
                        }
                }
        }
 }
 NAVC=NAVC+1;
 NAVMC=NAVMC+1;


end:
 return 0;

}
```

```
CARRIVE(int C)
{
 T=TC[C];
 if(ST[T]==1)/*TO TRAPEZI DEN EINAI STRWMENO*/
 {
         SC[C]=3;/*CAM PERIMENEI*/
         TEC[C]=BIG;
 }
 else if(ST[T]==2)
 {
         SC[C]=1;/*CAM KOBEI*/
         TEC[C]=CLOCK+TJC[OC[C]][JC[C]];
         NEC[C]=1;
         TET[T]=TEC[C]-TCLEAR;
         NET[T]=2;
 }



}
randorder()
{
 int I,L,Y;


 for(L=NO;L>=1;L--)
 {
         zrand();
         I=1+Z*L;
         Y=SEQ[L];
         SEQ[L]=SEQ[I];
         SEQ[I]=Y;
 }
}

zrand()
{


         XHI=(IE/65536);
 XALO=(IE-XHI*65536)*24112;
 LEFTLO=XALO/65536;
 FHI=XHI*24112+LEFTLO;
 K=FHI/32768;
 IE=(((XALO-LEFTLO*65536)-2147483647)+(FHI-K*32768)*65536)+K;
 if(IE<0)
         IE=IE+2147483647;
 XHI=IE/65536;
 XALO=(IE-XHI*65536)*26143;
 LEFTLO=XALO/65536;
 FHI=XHI*26143+LEFTLO;
 K=FHI/32768;
 IE=(((XALO-LEFTLO*65536)-2147483647)+(FHI-K*32768)*65536)+K;
 if(IE<0)
         IE=IE+2147483647;
 Z=(2*(IE/256)+1)/16777216;

}

eye()
{
 FILE *ptr1,*ptr2,*ptr3;
 int  i,j;

 ptr1=fopen("fanco1.txt","r");
 ptr2=fopen("fanco2.txt","w");
 ptr3=fopen("fanco3.txt","w");

 if(ptr1==NULL)
 printf("problem opening fanco1.txt");
 if(ptr2==NULL)
 printf("problem opening fanco2.txt");

 fscanf(ptr1,"%d %f",&NO,&TCLEAR);
 fprintf(ptr2,"%d %f",NO,TCLEAR);
```

```
for (i=1;i<=NO;i++)
{
fscanf(ptr1,"%d",&SEQ[i]);
}
fprintf(ptr2,"\n");

for (i=1;i<=10;i++)
{
fprintf(ptr2,"%d ",SEQ[i]);
}

fscanf(ptr1,"%d%d%d",&NT,&NMT,&NMC);
fprintf(ptr2,"\n");
fprintf(ptr2,"%d %d %d",NT,NMT,NMC);

for (i=1;i<=NO;i++)
{
        fscanf(ptr1,"%d",&NJ[i]);
        for(j=1;j<=NJ[i];j++)
        fscanf(ptr1,"%f%f",&TJT[i][j],&TJC[i][j]);
}

fprintf(ptr2,"\n\n");

for (i=1;i<=NO;i++)
{
        fprintf(ptr2,"%d",NJ[i]);
        fprintf(ptr2,"\n");
        for(j=1;j<=NJ[i];j++)
        {       fprintf(ptr2,"\n");
                fprintf(ptr2,"%f %f",TJT[i][j],TJC[i][j]);
        }

        fprintf(ptr2,"\n\n");
}


for(i=0;i<=NT+1;i++)
{
        for(j=0;j<=NT+1;j++)
        fscanf(ptr1,"%f",&SETUP[i][j]);
}

fprintf(ptr2,"\n\n");

for(i=0;i<=NT+1;i++)
{
        for(j=0;j<=NT+1;j++)
        fprintf(ptr2,"%f ",SETUP[i][j]);
        fprintf(ptr2,"\n\n");
}
for(i=1;i<=NT;i++)
fscanf(ptr1,"%d",&PR[i]);
fprintf(ptr2,"\n\n");

for(i=1;i<=7;i++)
fprintf(ptr2,"%d ",PR[i]);


for (i=1;i<=NT;i++)
{
        for(j=1;j<=NC;j++)
        fscanf(ptr1,"%d",&PRC[i][j]);
}

fprintf(ptr2,"\n\n");
for (i=1;i<=NT;i++)
{
        for(j=1;j<=NC;j++)
        fprintf(ptr2,"%d ",PRC[i][j]);

    fprintf(ptr2,"\n");
}

for (i=1;i<=NC;i++)
{
        for(j=1;j<=NT;j++)
```

```
        fscanf(ptr1,"%d",&PRT[i][j]);
 }

 fprintf(ptr2,"\n\n");

 for (i=1;i<=NC;i++)
 {
        for(j=1;j<=NT;j++)
        fprintf(ptr2,"%d ",PRT[i][j]);

        fprintf(ptr2,"\n");
 }


}
```

Β. Κώδικας Γενετικού Αλγορίθμου

```
#include <stdio.h>
#include <time.h>
#define POPSIZE 10
#define MAXGENS 10
#define PXOVER 0.85
#define PMUTATION 0.25
#define PCOMMUNICATION 0.22
double Z,IE;
long int XHI,XALO,LEFTLO,FHI,K;
int parent1,parent2,memb;
int
J,O,NITER,M,EV,T,JCC,C,NT,NAVT,NAVMT,NC,NAVC,NAVMC,NMT,NMC,NO,NJ[100],SEQ[100],BEST[100],WORST[100],PR[10],
PRC[10][3],PRT[3][10],TC[3],SC[3],OC[3],OT[10],JC[3],JT[10],CT[10],ST[10],NET[10],NEC[3];
float TJT[100][4],TJC[100][4],SETUP[10][10],TCLEAR;
int NC=2;
double BIG=10000000,BIG1=9999999;
double CLOCK,TEC[3],TET[10],CLOCKNEW,BCLOCK,WCLOCK,previous_best_makespan;
int regeneration,cur_best;
clock_t start, end;
double cpu_time_used;
FILE *galog;

struct genotype
{
 int SEQ[100];
 double fitness;
};
struct genotype population[POPSIZE+1];
struct genotype newpopulation[POPSIZE+1];
initialize();
communication();
selection();
crossover();
void xover(int one,int two);
mutate();
evaluate();
keep_the_best();
sort_population();
regenerate_population();
randorder();
zrand();

INITIALIZE();
NEXTEV();
TFINISH(int T);
TSTART(int T);
CFINISH(int C);
CARRIVE(int C);
```

```c
eye();

void main(void)
{
 int i,iter_no_improv,mem,regeneration,j,all_bestSEQ[100],all_worstSEQ[100];
 double previous_best_makespan,all_best,all_worst;

 galog=fopen("galog.txt","w");


 regeneration=0;
 previous_best_makespan=1000000;
 iter_no_improv=0;
 all_best=1000000;
 all_worst=1;

 initialize();
 evaluate();
 keep_the_best();


 while(regeneration<=MAXGENS)
 {

	if(population[POPSIZE].fitness>=previous_best_makespan)
	{
		iter_no_improv=iter_no_improv+1;
		if(iter_no_improv==25)
		{

			sort_population();
			regenerate_population();
			evaluate();
			regeneration =regeneration+1;
			iter_no_improv=0;
		}
	}
	previous_best_makespan=population[POPSIZE].fitness;


	selection();
	communication();
	crossover();
	mutate();
	evaluate();
	keep_the_best();

	if(all_best>population[POPSIZE].fitness)
	{
		all_best=population[POPSIZE].fitness;
		for(i=1;i<=NO;i++)
		all_bestSEQ[i]=population[POPSIZE].SEQ[i];
	}

	for(mem=0;mem<POPSIZE;mem++)
	{
		if(all_worst<population[mem].fitness)
			all_worst=population[mem].fitness;
		for(i=1;i<=NO;i++)
			all_worstSEQ[i]=population[mem].SEQ[i];
	}



 }
 fprintf(galog,"\n\n\nTHE BEST SEQUENCE IS==>");
 for(i=1;i<=NO;i++)
 fprintf(galog,"%d ",all_bestSEQ[i]);
 fprintf(galog,"\nWITH MAKESPAN=%lf",all_best);

 fprintf(galog,"\n\n\n THE WORST SEQUENCE IS=>");
 for(i=1;i<=NO;i++)
 fprintf(galog,"%d ",all_worstSEQ[i]);
 fprintf(galog,"\nWITH MAKESPAN=%lf",all_worst);
```

```
  end = clock();
  cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
  fprintf(galog,"\n\nCPU TIME=%lf",cpu_time_used);



}

initialize()
{
  int mem,i;
  FILE *ptr1;
  ptr1=fopen("fanco1.txt","r");

  printf("dwse sporo:apo 1 mexri 12345678\n");
  scanf("%lf",&IE);
  eye();
  start = clock();
  for(i=1;i<=NO;i++)
  population[0].SEQ[i]=SEQ[i];

  for(mem=1;mem<POPSIZE;mem++)
  {
        randorder();
        for(i=1;i<=NO;i++)
        {
        population[mem].SEQ[i]=SEQ[i];
        }
  }
}
communication()
{
  int mem,friends,friendd,i,j,point,point1,tmp;


  for (mem=0;mem<POPSIZE;mem++)
  {
        zrand();
        if(Z<PCOMMUNICATION)
        {

              zrand();
              friends=1+Z*POPSIZE;

              for(j=0;j<friends;j++)
              {
                    zrand();
                    friendd=Z*POPSIZE;

                    zrand();
                    point=1+Z*NO;

                    tmp=population[mem].SEQ[point];
                    for(i=1;i<=NO;i++)
                    {
                          if(population[mem].SEQ[i]==population[friendd].SEQ[point])
                                point1=i;
                    }
                    population[mem].SEQ[point]=population[friendd].SEQ[point];/* ena aplo swap*/
                    population[mem].SEQ[point1]=tmp;
              }
        }
  }
}
randorder()
{
  int I,L,Y;


  for(L=NO;L>=1;L--)
  {
        zrand();
        I=1+Z*L;
        Y=SEQ[L];
        SEQ[L]=SEQ[I];
        SEQ[I]=Y;
  }
```

```
}

zrand()
{


        XHI=(IE/65536);
 XALO=(IE-XHI*65536)*24112;
 LEFTLO=XALO/65536;
 FHI=XHI*24112+LEFTLO;
 K=FHI/32768;
 IE=(((XALO-LEFTLO*65536)-2147483647)+(FHI-K*32768)*65536)+K;
 if(IE<0)
        IE=IE+2147483647;
 XHI=IE/65536;
 XALO=(IE-XHI*65536)*26143;
 LEFTLO=XALO/65536;
 FHI=XHI*26143+LEFTLO;
 K=FHI/32768;
 IE=(((XALO-LEFTLO*65536)-2147483647)+(FHI-K*32768)*65536)+K;
 if(IE<0)
        IE=IE+2147483647;
 Z=(2*(IE/256)+1)/16777216;

}

evaluate()
{
 int  i,j,mem;
 FILE *ptr3;


ptr3=fopen("fanco3.txt","w");

 for(memb=0;memb<=POPSIZE;memb++)
 {



                INITIALIZE();
                start:
                NEXTEV();
                if(CLOCKNEW>BIG1)
                goto end3;
                CLOCK=CLOCKNEW;

                if(EV==1)
                {
                        if(NET[M]==1)
                        TFINISH(M);
                        else if(NET[M]==2)
                        TSTART(M);
                }

                else
                {
                        if(NEC[M]==1)
                        CFINISH(M);
                        else if(NEC[M]==2)
                        CARRIVE(M);
                }
                goto start;
                end3:
                population[memb].fitness=CLOCK;

 }

}
INITIALIZE()
{

 int i,j,MIN,MAX,ITER;

MAX=0;
MIN=MAX;

 CLOCK=0.0;
```

```
 NAVT=NT;
 NAVC=NC;
 NAVMC=NMC;
 NAVMT=NMT;
 TC[1]=0;/*POU BRISKONTAI TA CAM,SE POIO TRAPEZI PANW*/
 TC[2]=NT+1;
 for(i=1;i<=NC;i++)/*ARXIKOPOIHSH CAM*/
 {
        SC[i]=0;
        TEC[i]=BIG;
        OC[i]=0;
        JC[i]=0;
 }

 for(i=1;i<=NT;i++)/*ARXIKOPOIHSH TABLES*/
 {
        TET[i]=BIG;
        CT[i]=0;
        ST[i]=0;
        OT[i]=0;
        JT[i]=0;

 }

 J=0; /*STHN ARXH DEN EXOUME FORTWSEI JOB ARA OUTE ORDER*/
 O=0;
 ITER=1;

again:
 T=PR[ITER];/*TO  TABLE POU THA STRWSOUME TO PERNOUME APO PRIORITI LIST*/
 ST[T]=1;/*MOLIS BRHKAME ENA TABLE SUMFWNA ME PRIORITI LIST*.H KATASTASH TOU GINETAI 1*/



 if(J==0 || J==(NJ[population[memb].SEQ[O]]))/*AN DEN EXOUME FORTWSEI KANENA JOB H EXOUME TELEIWSEI OLA TA
JOBS THS SYGKEKRIMENHS PARAGGELIAS*/
 {
        J=1;
        O=O+1;
        JT[T]=1;
 }
 else   /*BRISKOMASTE STHN IDIA PARAGGELIA*/
 {
        J=J+1;
        JT[T]=J;
 }

 OT[T]=population[memb].SEQ[O];
 TET[T]=CLOCK+ TJT[OT[T]][JT[T]];
 NET[T]=1;/*MOLIS BRIKAME TRAPEZI GIA NA STRWTHEI ARA MPOREI NA MPEI STO TFINISH*/
 NAVMT=NAVMT-1;/*DESMEYOUME ERGATES*/
 NAVT=NAVT-1;

 if(ITER==1)
 {
        C=PRC[T][1];/*TO CAM POU THA KOPSEI PANW APTO TRAPEZI POU STRWTHIKE TO PAIRNOUME APO PRIORITI LIST*/
        SC[C]=2;/*MOLIS BRIKAME TO CAM POU THA KOPSEI SUMFWNA ME PRIORITY LIST. H KATASTASH TOU GINETAI 2*/
        CT[T]=C;
        TEC[C]=CLOCK + SETUP[TC[C]][T];
        NEC[C]=2;/*EIDOS EPOMENOU GEGONOTOS STO CAM*/
        TC[C]=T;
        OC[C]=OT[T];
        JC[C]=JT[T];
        NAVMC=NAVMC-1;
        NAVC=NAVC-1;
        JCC=3-C;
 }
 else if (NAVC>0&&NAVMC>0)/*AMA YPARXOUN CAM */
 {

        if(TC[JCC]<T)
        {
               MIN=TC[JCC];
                MAX=T;
        }
        else if(TC[JCC]>T)
        {
```

```
                MIN=T;
                MAX=TC[JCC];
        }
        else
        {
                MIN=T;
                MAX=T;
        }



        if(TC[C]<MIN||TC[C]>MAX)/*KAI MPOROUN NA PANE*/
        {
                SC[JCC]=2;
                CT[T]=JCC;
                TEC[JCC]=CLOCK + SETUP[TC[C]][T];
                NEC[JCC]=2;
                TC[JCC]=T;
                OC[JCC]=OT[T];
                JC[JCC]=JT[T];
                NAVMC=0;
                NAVC=0;
        }

  }
  ITER=ITER+1;
  if ((O<(NO)||J<(NJ[population[memb].SEQ[O]]))&&NAVT>0&&NAVMT>0)
        goto again;



}
NEXTEV()
{

  int i,j;

  CLOCKNEW=BIG;
  EV=0;
  M=-1;
  for(i=1;i<=NT;i++)
  {
        if(TET[i]<CLOCKNEW)
        {
                CLOCKNEW=TET[i];
                EV=1;
                M=i;
        }
  }

  for(j=1;j<=NC;j++)
  {
        if(TEC[j]<CLOCKNEW)
        {
                CLOCKNEW=TEC[j];
                EV=2;
                M=j;
        }
  }



}
TFINISH(int T)/*TRAPEZI STRWTHIKE*/
{
  int i,j,MIN,MAX;
  MAX=0;
  MIN=MAX;


  ST[T]=2;
  C=CT[T];

  if(C>0&&SC[C]==3)/*AN YPARXEI CAM KAI EINAI DIATHESIMO DOULEUEI*/
  {
        SC[C]=1;/*ARXIZEI NA DOULEUEI TO CAM PANW APTO TRAPEZI POU MOLIS STRWTHIKE*/
        TEC[C]=CLOCK + TJC[OT[T]][JT[T]];
```

```
        NEC[C]=1;/*NEXT EVENT CAM=CAM FINISH*/
        TET[T]=TEC[C]-TCLEAR;
        NET[T]=2;

}
else if(C==0||SC[C]==2)/*DEN YPARXEI CAM H YPARXEI ALLA EINAI APASXOLHMENO*./*ALLAKSA to else if*/
{
        TET[T]=BIG;/*TO TRAPEZI EINAI STRWMENO XWRIS CAM.APROSDIORISTOS O XRONOS POU THA EINAI DIATHESIMOS*/
}

NAVMT=NAVMT+1;/*AFOU STRWTHIKE TABLE*/


for(i=1;i<=NT;i++)/*PSAXNOUME NA BROUME TO TRAPEZI POU THA STRWSOUME*/
{
        T=PR[i];

        if(ST[T]==0)/*TO TRAPEZI EINAI DIATHESIMO GIA STRWSIMO*/
        {
                if(O==(NO)&&(J==(NJ[population[memb].SEQ[O]])))
                {
                        ST[T]=0;
                        TET[T]=BIG;
                        OT[T]=0;
                        JT[T]=0;
                }
                else/*NEW JOB STARTS*/
                {

                        if(J==(NJ[population[memb].SEQ[O]]))
                        {
                                J=1;
                                O=O+1;
                                JT[T]=1;
                        }
                        else if(J<(NJ[population[memb].SEQ[O]]))
                        {
                                J=J+1;
                                JT[T]=J;
                        }

                        OT[T]=population[memb].SEQ[O];/*POIA EINAI H PARAGGELIA POU THA  KANEI TWRA*/
                        ST[T]=1;
                        TET[T]=CLOCK +TJT[OT[T]][JT[T]];
                        NET[T]=1;/*EIDOS EPOMENOU GEGONOTOS STO TABLE=TABLEFINISH*/
                        NAVMT=NAVMT-1;
                        NAVT=NAVT-1;
                        CT[T]=0;
                        for(j=1;j<=NC;j++)
                        {
                                C=PRC[T][j];
                                if(TC[C]<T)
                                {                               /*TO C EINAI TO CAM POU RWTAEI AN MPOREI NA
PAEI STO T*/
                                MIN=TC[C];
                                MAX=T;
                                }
                                else if(TC[C]>T)
                                {
                                MIN=T;
                                MAX=TC[C];
                                }
                                else
                                {
                                MIN=T;
                                MAX=T;
                                }



                                if(SC[C]==0&&NAVMC>0&&(TC[3-C]<MIN||TC[3-C]>MAX))
                                {

                                        SC[C]=2;/*DESMEYSH CAM GIA NA PAEI STO T*/
                                        CT[T]=C;
                                        TEC[C]=CLOCK+SETUP[TC[C]][T];
                                        NEC[C]=2;/*TO DESMEUSAME GIA NA PAEI ARA NEXTEVENT=CARRIVE*/
```

```
                                        TC[C]=T;
                                        OC[C]=OT[T];
                                        JC[C]=JT[T];

                                }
                        }




                        goto end1;/*AN BREIS DIATHESIMO TRAPEZI KAI YPARXEI JOB NA KANEIS  MHN PSAXNEIS
ALLO*/

                }





        }





 }
end1:
 return 0;
}

TSTART(int T)/*TRAPEZI DIATHESIMO GIA STRWSIMO*/
{
 int i,j,MIN,MAX;
 MAX=0;
 MIN=MAX;

 if (NAVMT==0||(O==(NO)&&J==NJ[population[memb].SEQ[O]]))
 {
        ST[T]=0;
        TET[T]=BIG;
        OT[T]=0;
        JT[T]=0;
        NAVT=NAVT+1;
 }
 else
 {
        if(J==(NJ[population[memb].SEQ[O]]))
        {
                J=1;
                O=O+1;
                JT[T]=1;
        }
     else if(J<(NJ[population[memb].SEQ[O]]))
        {
                J=J+1;
                JT[T]=J;
        }
        OT[T]=population[memb].SEQ[O];
        ST[T]=1;
        TET[T]=CLOCK+TJT[OT[T]][JT[T]];
        NET[T]=1;
        NAVMT=NAVMT-1;
        CT[T]=0;
        for(i=1;i<=NC;i++)
        {
                C=PRC[T][i];/*TO CAM C RWTAEI AN MPOREI NA PAEI STO T*/
                if(TC[C]<T)
                {
                MIN=TC[C];
                MAX=T;
                }
                else if(TC[C]>T)
                {
                MIN=T;
                MAX=TC[C];
                }
                else
```

```
                {
                MIN=T;
                MAX=T;
                }


                if(SC[C]==0&&NAVMC>0&&(TC[3-C]<MIN||TC[3-C]>MAX))
                {

                        SC[C]=2;
                        CT[T]=C;
                        TEC[C]=CLOCK + SETUP[TC[C]][T];
                        NEC[C]=2;
                        TC[C]=T;
                        OC[C]=OT[T];
                        JC[C]=JT[T];
                }
        }

 }

}

CFINISH(int C)
{
 int i,j,MIN,MAX;

 for (i=1;i<=NT;i++)
 {
        T=PRT[C][i];
        if(TC[C]<T)
        {                               /*TO C EINAI POU RWTAEI AN MPOREI NA PAEI STO T*/
                MIN=TC[C];
                MAX=T;
        }
        else if(TC[C]>T)
        {
                MIN=T;
                MAX=TC[C];
        }
        else
        {
        MIN=T;
        MAX=T;
        }

        if((ST[T]>0&&(CT[T]==0))&&(TC[3-C]<MIN||TC[3-C]>MAX))/*NA EINAI STRWMENO TO TABLE,NA MHN YPARXEI CAM
KAI NA MPOREI NA PAEI*/
        {

                SC[C]=2;/*TO CAM DESMEYETAI GIA NA PAEI SE TRAPEZI*/
                CT[T]=C;
                TEC[C]=CLOCK + SETUP[TC[C]][T];
                NEC[C]=2;
                TC[C]=T;
                OC[C]=OT[T];
                JC[C]=JT[T];
                goto end;/*AFOU BRIKE TRAPEZI KAI CAM FEYGEI GIA NEXTEVENT*/
        }
 }
 SC[C]=0;/*AN DE BREI TRAPEZI GIA NA PAEI TO CAM KOIMATAI*/
 TEC[C]=BIG;
 if(NMC<NC)
 {
        JCC=3-C;
        if(SC[JCC]==0)
        {
                for(i=1;i<=NT;i++)
                {
                        T=PRT[JCC][i];
                        if(TC[JCC]<T)
                        {
                                MIN=TC[JCC];
                                MAX=T;
                        }
                        else
                        {
```

```
                                MIN=T;
                                MAX=TC[JCC];
                        }

                        if(ST[T]>0&&CT[T]==(0)&&(TC[C]<MIN||TC[C]>MAX))
                        {
                                SC[C]=2;
                                CT[T]=C;
                                TEC[C]=CLOCK + SETUP[TC[C]][T];
                                NEC[C]=2;
                                TC[C]=T;
                                OC[C]=OT[T];
                                JC[C]=JT[T];
                                goto end;
                        }
                }
        }
 }
 NAVC=NAVC+1;
 NAVMC=NAVMC+1;


end:
 return 0;

}

CARRIVE(int C)
{
 T=TC[C];
 if(ST[T]==1)/*TO TRAPEZI DEN EINAI STRWMENO*/
 {
        SC[C]=3;/*CAM PERIMENEI*/
        TEC[C]=BIG;
 }
 else if(ST[T]==2)
 {
        SC[C]=1;/*CAM KOBEI*/
        TEC[C]=CLOCK+TJC[OC[C]][JC[C]];
        NEC[C]=1;
        TET[T]=TEC[C]-TCLEAR;
        NET[T]=2;
 }



}
eye()
{
 FILE *ptr1;
 int  i,j;

 ptr1=fopen("fanco1.txt","r");
 if(ptr1==NULL)
 printf("problem opening fanco1.txt");

 fscanf(ptr1,"%d %f",&NO,&TCLEAR);
 for (i=1;i<=NO;i++)
 {
 fscanf(ptr1,"%d",&SEQ[i]);
 }
 fscanf(ptr1,"%d%d%d",&NT,&NMT,&NMC);
 for (i=1;i<=NO;i++)
 {
        fscanf(ptr1,"%d",&NJ[i]);
        for(j=1;j<=NJ[i];j++)
        fscanf(ptr1,"%f%f",&TJT[i][j],&TJC[i][j]);
 }

 for(i=0;i<=NT+1;i++)
 {
        for(j=0;j<=NT+1;j++)
        fscanf(ptr1,"%f",&SETUP[i][j]);
 }

 for(i=1;i<=NT;i++)
 fscanf(ptr1,"%d",&PR[i]);
```

```
 for (i=1;i<=NT;i++)
 {
         for(j=1;j<=NC;j++)
         fscanf(ptr1,"%d",&PRC[i][j]);
 }
 for (i=1;i<=NC;i++)
 {
         for(j=1;j<=NT;j++)
         fscanf(ptr1,"%d",&PRT[i][j]);
 }
}


keep_the_best()
{
 int mem,cur_best,i,min;
 cur_best=0;
 min=population[0].fitness;
 for(mem=1;mem<POPSIZE;mem++)
 {
         if(population[mem].fitness<min)
         {
                 min=population[mem].fitness;
                 cur_best=mem;
         }
 }
 population[POPSIZE]=population[cur_best];


}

selection()
{
 int mem,parent1,parent2,parent3,parent;




         for(mem=0;mem<POPSIZE;mem++)
         {
                 zrand();
                 parent1=Z*POPSIZE;
                 zrand();
                 parent2=Z*POPSIZE;
                 if(population[parent1].fitness<population[parent2].fitness)
                 newpopulation[mem]=population[parent1];
                 else if(population[parent1].fitness>population[parent2].fitness)
                 newpopulation[mem]=population[parent2];

                 else
                 {

                         newpopulation[mem]=population[parent2];

                 }
         }


 for(mem=0;mem<POPSIZE;mem++)
 population[mem]=newpopulation[mem];


}

crossover()
{
 int i,mem,one;
 int first=0;

 for(mem=0;mem<POPSIZE;mem++)
 {
         zrand();
         if(Z<PXOVER)
         {
                 ++first;
                 if(first%2==0)
```

```
                {
                xover(one,mem);
                }
                else
                one=mem;

        }
 }

}
void xover(int one,int two)
{
        int point1,point2,c,j,i,used_orders[100],temp,tmp[100];

        zrand();
        point1=1+Z*NO;
        zrand();
        point2=1+Z*NO;
        if(point1>point2)
        {
                temp=point1;
                point1=point2;
                point2=temp;
        }
        for(j=1;j<=NO;j++)
        tmp[j]=population[one].SEQ[j];
        for(c=1;c<=2;c++)
        {
                for(i=1;i<=point1;i++)
                        used_orders[population[one].SEQ[i]]=1;
                for(i=point2;i<=NO;i++)
                        used_orders[population[one].SEQ[i]]=1;
                j=NO;

                i=point1+1;
                while(i<point2)
                {
                        if(c==2)
                        {

                                if(used_orders[tmp[j]]!=1)
                                {
                                        population[one].SEQ[i]=tmp[j];
                                        j--;
                                        i++;
                                }
                                else
                                        j--;
                        }
                        else
                        {
                                if(used_orders[population[two].SEQ[j]]!=1)
                                {
                                        population[one].SEQ[i]=population[two].SEQ[j];
                                        j--;
                                        i++;

                                }
                                else
                                j--;
                        }

                }
                temp=one;
                one=two;
                two=temp;

                for(i=1;i<=NO;i++)
                        used_orders[i]=0;
        }
}

mutate()
{
 int mem,point1,point2,temp;

 for(mem=0;mem<POPSIZE;mem++)
```

```
 {
         zrand();
         if(Z<PMUTATION)
         {
                 zrand();
                 point1=1+Z*NO;
                 zrand();
                 point2=1+Z*NO;
                 temp=population[mem].SEQ[point1];
                 population[mem].SEQ[point1]=population[mem].SEQ[point2];
                 population[mem].SEQ[point2]=temp;
         }
 }



}

sort_population()
{
 int i,min,j;

 for(i=0;i<POPSIZE;i++)
 {
         min=i;
         for(j=i+1;j<POPSIZE;j++)
         {
                 if((population[min].fitness)>(population[j].fitness))
                 {
                         min=j;

                 }
         }
         newpopulation[i]=population[i];
         population[i]=population[min];
         population[min]=newpopulation[i];


 }
}

regenerate_population()
{
 int I,L,Y,mem,choose_order,position,i,j,choose_order_position;

 for(mem=0.2*POPSIZE-1;mem<=0.6*POPSIZE;mem++)
 {
         zrand();
         choose_order=1+Z*NO;
         zrand();
         position=1+Z*NO;
         for(i=1;i<=NO;i++)
         {
                 population[mem].SEQ[i]=population[0].SEQ[i];
         }
         for(i=1;i<=NO;i++)
         {
                 if(population[mem].SEQ[i]==choose_order)
                 {
                         choose_order_position=i;
                 }
         }
         if(choose_order_position<position)
         {
                 for(j=choose_order_position;j<position;j++)
                 {
                 population[mem].SEQ[j]=population[0].SEQ[j+1];
                 }
         }
         else
         {
                 for(j=choose_order_position;j>position;j--)
                 {
                         population[mem].SEQ[j]=population[0].SEQ[j-1];
                 }
         }
         population[mem].SEQ[position]=choose_order;
```

```
 }



 for(mem=0.6*POPSIZE+1;mem<POPSIZE;mem++)
 {
        for(i=1;i<=NO;i++)
               SEQ[i]=population[mem].SEQ[i];
        randorder();
        for(i=1;i<=NO;i++)
        population[mem].SEQ[i]=SEQ[i];

 }


}
```

Γ. Κώδικας Εύρεσης όλων των Δυνατών Αλληλουχιών (πλήρης αναζήτηση)

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
int level=-1;
int value[100],N;
void visit(int);
int
J,O,NITER,M,EV,T,JCC,C,NT,NAVT,NAVMT,NC,NAVC,NAVMC,NMT,NMC,NO,NJ[100],SEQ[100],BEST[100],WORST[100],PR[10],
PRC[10][3],PRT[3][10],TC[3],SC[3],OC[3],OT[10],JC[3],JT[10],CT[10],ST[10],NET[10],NEC[3];
float TJT[100][4],TJC[100][4],SETUP[10][10],TCLEAR;
int NC=2;
double BIG=10000000,BIG1=9999999;
double CLOCK,TEC[3],TET[10],CLOCKNEW,BCLOCK,WCLOCK,perm;

FILE *list;
additem();
main()
{
 int  i,j,iter,l;
 FILE *ptr1,*ptr2,*ptr3;
 clock_t start, end;
    double cpu_time_used;

 list=fopen("list.txt","w");
 ptr1=fopen("fanco1.txt","r");
 ptr2=fopen("fanco2.txt","w");
 ptr3=fopen("fanco3.txt","w");

 if(ptr1==NULL)
 printf("problem opening fanco1.txt");
 if(ptr2==NULL)
 printf("problem opening fanco2.txt");
 eye();

 BCLOCK=BIG;
 WCLOCK=0;
 start = clock();



 visit(1);
 fprintf(ptr3,"sunduasmoi=%lf",perm);
 fprintf(ptr3,"\n\n\nTHE BEST SEQUENCE IS==>");
 for(i=1;i<=NO;i++)
 fprintf(ptr3,"%d ",BEST[i]);
 fprintf(ptr3,"\nWITH MAKESPAN=%lf",BCLOCK);

 end = clock();
```

```
 cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
 fprintf(ptr3,"\nCPU TIME=%lf",cpu_time_used);
}

void visit(int k)
{
 int i,j,p;
 N=NO;

 level=level+1;
 value[k]=level;
 if(level==N)
 {
        perm=perm+1;
                for(p=1;p<=NO;p++)
                SEQ[p]=value[p];

                INITIALIZE();
        start:
                NEXTEV();
                if(CLOCKNEW>BIG1)
                goto end3;
                CLOCKNEW=CLOCKNEW;

        if(EV==1)
        {
                if(NET[M]==1)
                TFINISH(M);
                else if(NET[M]==2)
                TSTART(M);
        }

        else
        {
                if(NEC[M]==1)
                CFINISH(M);
                else if(NEC[M]==2)
                CARRIVE(M);

        }

        goto start;


        end3:

        if(CLOCK<BCLOCK)
        {
        BCLOCK=CLOCK;
        for(i=1;i<=NO;i++)
        BEST[i]=SEQ[i];
        }

        if(CLOCK>WCLOCK)
        {
        WCLOCK=CLOCK;
        for(i=1;i<=NO;i++)
        WORST[i]=SEQ[i];
        }




 }

 else
        for( i=1;i<=N;i++)
                if(value[i]==0)
                        visit(i);
 level=level-1;
 value[k]=0;
}
 additem()
{
 int j;
        fprintf(list,"\n........\n");
        for(j=0;j<N;j++)
```

```
        fprintf(list,"%d",value[j]);


}

INITIALIZE()
{

 int i,j,MIN,MAX,ITER;

MAX=0;
MIN=MAX;

 CLOCK=0.0;
 NAVT=NT;
 NAVC=NC;
 NAVMC=NMC;
 NAVMT=NMT;
 TC[1]=0;/*POU BRISKONTAI TA CAM,SE POIO TRAPEZI PANW*/
 TC[2]=NT+1;

 for(i=1;i<=NC;i++)/*ARXIKOPOIHSH CAM*/
 {
        SC[i]=0;
        TEC[i]=BIG;
        OC[i]=0;
        JC[i]=0;
 }

 for(i=1;i<=NT;i++)/*ARXIKOPOIHSH TABLES*/
 {
        TET[i]=BIG;
        CT[i]=0;
        ST[i]=0;
        OT[i]=0;
        JT[i]=0;

 }

 J=0; /*STHN ARXH DEN EXOUME FORTWSEI JOB ARA OUTE ORDER*/
 O=0;
 ITER=1;

again:
 /*do*/



 T=PR[ITER];/*TO  TABLE POU THA STRWSOUME TO PERNOUME APO PRIORITI LIST*/
 ST[T]=1;/*MOLIS BRHKAME ENA TABLE SUMFWNA ME PRIORITI LIST*.H KATASTASH TOU GINETAI 1*/



 if(J==0 || J==(NJ[SEQ[O]]))/*AN DEN EXOUME FORTWSEI KANENA JOB H EXOUME TELEIWSEI OLA TA JOBS THS
SYGKEKRIMENHS PARAGGELIAS*/
 {
        J=1;
        O=O+1;
        JT[T]=1;
 }
 else    /*BRISKOMASTE STHN IDIA PARAGGELIA*/
 {
        J=J+1;
        JT[T]=J;
 }

 OT[T]=SEQ[O];
 TET[T]=CLOCK+ TJT[OT[T]][JT[T]];
 NET[T]=1;/*MOLIS BRIKAME TRAPEZI GIA NA STRWTHEI ARA MPOREI NA MPEI STO TFINISH*/
 NAVMT=NAVMT-1;/*DESMEYOUME ERGATES*/
 NAVT=NAVT-1;

 if(ITER==1)
 {
        C=PRC[T][1];/*TO CAM POU THA KOPSEI PANW APTO TRAPEZI POU STRWTHIKE TO PAIRNOUME APO PRIORITI LIST*/
        SC[C]=2;/*MOLIS BRIKAME TO CAM POU THA KOPSEI SUMFWNA ME PRIORITY LIST. H KATASTASH TOU GINETAI 2*/
        CT[T]=C;
```

```
        TEC[C]=CLOCK + SETUP[TC[C]][T];
        NEC[C]=2;/*EIDOS EPOMENOU GEGONOTOS STO CAM*/
        TC[C]=T;
        OC[C]=OT[T];
        JC[C]=JT[T];
        NAVMC=NAVMC-1;
        NAVC=NAVC-1;
        JCC=3-C;
 }
 else if (NAVC>0&&NAVMC>0)/*AMA YPARXOUN CAM */
 {

        if(TC[JCC]<T)
        {
               MIN=TC[JCC];
                MAX=T;
        }
        else if(TC[JCC]>T)
        {
               MIN=T;
               MAX=TC[JCC];
        }
        else
        {
               MIN=T;
               MAX=T;
        }



        if(TC[C]<MIN||TC[C]>MAX)/*KAI MPOROUN NA PANE*/
        {
               SC[JCC]=2;
               CT[T]=JCC;
               TEC[JCC]=CLOCK + SETUP[TC[C]][T];
               NEC[JCC]=2;
               TC[JCC]=T;
               OC[JCC]=OT[T];
               JC[JCC]=JT[T];
               NAVMC=0;
               NAVC=0;
        }

 }
 ITER=ITER+1;
 if ((O<(NO)||J<(NJ[SEQ[O]]))&&NAVT>0&&NAVMT>0)
        goto again;



}


NEXTEV()
{

 int i,j;

 CLOCKNEW=BIG;
 EV=0;
 M=-1;
 for(i=1;i<=NT;i++)
 {
        if(TET[i]<CLOCKNEW)
        {
               CLOCKNEW=TET[i];
               EV=1;
               M=i;
        }
 }

 for(j=1;j<=NC;j++)
 {
        if(TEC[j]<CLOCKNEW)
        {
               CLOCKNEW=TEC[j];
               EV=2;
```

```
                    M=j;
          }
  }



}

TFINISH(int T)/*TRAPEZI STRWTHIKE*/
{
 int i,j,MIN,MAX;
 MAX=0;
 MIN=MAX;


 ST[T]=2;/*TRAPEZI EFAGE*/
 C=CT[T];

 if(C>0&&SC[C]==3)/*AN YPARXEI CAM KAI EINAI DIATHESIMO DOULEUEI*/
 {
         SC[C]=1;/*ARXIZEI NA DOULEUEI TO CAM PANW APTO TRAPEZI POU MOLIS STRWTHIKE*/
         TEC[C]=CLOCK + TJC[OT[T]][JT[T]];
         NEC[C]=1;/*NEXT EVENT CAM=CAM FINISH*/
         TET[T]=TEC[C]-TCLEAR;
         NET[T]=2;

 }
 else if(C==0||SC[C]==2)/*DEN YPARXEI CAM H YPARXEI ALLA EINAI APASXOLHMENO*./*ALLAKSA to else if*/
 {
         TET[T]=BIG;/*TO TRAPEZI EINAI STRWMENO XWRIS CAM.APROSDIORISTOS O XRONOS POU THA EINAI DIATHESIMOS*/
 }

 NAVMT=NAVMT+1;/*AFOU STRWTHIKE TABLE*/


 for(i=1;i<=NT;i++)/*PSAXNOUME NA BROUME TO TRAPEZI POU THA STRWSOUME*/
 {
         T=PR[i];

         if(ST[T]==0)/*TO TRAPEZI EINAI DIATHESIMO GIA STRWSIMO*/
         {
                 if(O==(NO)&&(J==(NJ[SEQ[O]])))
                 {
                         ST[T]=0;
                         TET[T]=BIG;
                         OT[T]=0;
                         JT[T]=0;
                 }
                 else/*NEW JOB STARTS*/
                 {

                         if(J==(NJ[SEQ[O]]))
                         {
                                 J=1;
                                 O=O+1;
                                 JT[T]=1;
                         }
                         else if(J<(NJ[SEQ[O]]))
                         {
                                 J=J+1;
                                 JT[T]=J;
                         }

                         OT[T]=SEQ[O];/*POIA EINAI H PARAGGELIA POU THA  KANEI TWRA*/
                         ST[T]=1;
                         TET[T]=CLOCK +TJT[OT[T]][JT[T]];
                         NET[T]=1;/*EIDOS EPOMENOU GEGONOTOS STO TABLE=TABLEFINISH*/
                         NAVMT=NAVMT-1;
                         NAVT=NAVT-1;
                         CT[T]=0;
                         for(j=1;j<=NC;j++)
                         {
                                 C=PRC[T][j];
                                 if(TC[C]<T)
                                 {               /*TO C EINAI TO CAM POU RWTAEI AN MPOREI NA PAEI STO T*/
                                 MIN=TC[C];
                                 MAX=T;
```

```
                                    }
                                    else if(TC[C]>T)
                                    {
                                    MIN=T;
                                    MAX=TC[C];
                                    }
                                    else
                                    {
                                    MIN=T;
                                    MAX=T;
                                    }



                                    if(SC[C]==0&&NAVMC>0&&(TC[3-C]<MIN||TC[3-C]>MAX))
                                    {

                                            SC[C]=2;/*DESMEYSH CAM GIA NA PAEI STO T*/
                                            CT[T]=C;
                                            TEC[C]=CLOCK+SETUP[TC[C]][T];
                                            NEC[C]=2;/*TO DESMEUSAME GIA NA PAEI ARA NEXTEVENT=CARRIVE*/
                                            TC[C]=T;
                                            OC[C]=OT[T];
                                            JC[C]=JT[T];

                                    }
                            }




                            goto end1;/*AN BREIS DIATHESIMO TRAPEZI KAI YPARXEI JOB NA KANEIS  MHN PSAXNEIS
ALLO*/

                    }






            }





 }
end1:
 return 0;
}

TSTART(int T)/*TRAPEZI DIATHESIMO GIA STRWSIMO*/
{
 int i,j,MIN,MAX;
 MAX=0;
 MIN=MAX;

 if (NAVMT==0||(O==(NO)&&J==NJ[SEQ[O]]))
 {
         ST[T]=0;
         TET[T]=BIG;
         OT[T]=0;
         JT[T]=0;
         NAVT=NAVT+1;
 }
 else
 {
         if(J==(NJ[SEQ[O]]))
         {
                 J=1;
                 O=O+1;
                 JT[T]=1;
         }
     else if(J<(NJ[SEQ[O]]))
         {
                 J=J+1;
                 JT[T]=J;
```

```
        }
        OT[T]=SEQ[O];
        ST[T]=1;
        TET[T]=CLOCK+TJT[OT[T]][JT[T]];
        NET[T]=1;
        NAVMT=NAVMT-1;
        CT[T]=0;
        for(i=1;i<=NC;i++)
        {
                C=PRC[T][i];/*TO CAM C RWTAEI AN MPOREI NA PAEI STO T*/
                if(TC[C]<T)
                {
                MIN=TC[C];
                MAX=T;
                }
                else if(TC[C]>T)
                {
                MIN=T;
                MAX=TC[C];
                }
                else
                {
                MIN=T;
                MAX=T;
                }


                if(SC[C]==0&&NAVMC>0&&(TC[3-C]<MIN||TC[3-C]>MAX))
                {

                        SC[C]=2;
                        CT[T]=C;
                        TEC[C]=CLOCK + SETUP[TC[C]][T];
                        NEC[C]=2;
                        TC[C]=T;
                        OC[C]=OT[T];
                        JC[C]=JT[T];
                }
        }

 }

}

CFINISH(int C)
{
 int i,j,MIN,MAX;

 for (i=1;i<=NT;i++)
 {
        T=PRT[C][i];
        if(TC[C]<T)
        {                               /*TO C EINAI POU RWTAEI AN MPOREI NA PAEI STO T*/
                MIN=TC[C];
                MAX=T;
        }
        else if(TC[C]>T)
        {
                MIN=T;
                MAX=TC[C];
        }
        else
        {
        MIN=T;
        MAX=T;
        }

        if((ST[T]>0&&(CT[T]==0))&&(TC[3-C]<MIN||TC[3-C]>MAX))/*NA EINAI STRWMENO TO TABLE,NA MHN YPARXEI CAM 
KAI NA MPOREI NA PAEI*/
        {

                SC[C]=2;/*TO CAM DESMEYETAI GIA NA PAEI SE TRAPEZI*/
                CT[T]=C;
                TEC[C]=CLOCK + SETUP[TC[C]][T];
                NEC[C]=2;
                TC[C]=T;
                OC[C]=OT[T];
```

```
                JC[C]=JT[T];
                goto end;/*AFOU BRIKE TRAPEZI KAI CAM FEYGEI GIA NEXTEVENT*/
        }
 }
 SC[C]=0;/*AN DE BREI TRAPEZI GIA NA PAEI TO CAM KOIMATAI*/
 TEC[C]=BIG;
 if(NMC<NC)
 {
        JCC=3-C;
        if(SC[JCC]==0)
        {
                for(i=1;i<=NT;i++)
                {
                        T=PRT[JCC][i];
                        if(TC[JCC]<T)
                        {
                                MIN=TC[JCC];
                                MAX=T;
                        }
                        else
                        {
                                MIN=T;
                                MAX=TC[JCC];
                        }

                        if(ST[T]>0&&CT[T]==(0)&&(TC[C]<MIN||TC[C]>MAX))
                        {
                                SC[C]=2;
                                CT[T]=C;
                                TEC[C]=CLOCK + SETUP[TC[C]][T];
                                NEC[C]=2;
                                TC[C]=T;
                                OC[C]=OT[T];
                                JC[C]=JT[T];
                                goto end;
                        }
                }
        }
 }
 NAVC=NAVC+1;
 NAVMC=NAVMC+1;


end:
 return 0;

}

CARRIVE(int C)
{
 T=TC[C];
 if(ST[T]==1)/*TO TRAPEZI DEN EINAI STRWMENO*/
 {
        SC[C]=3;/*CAM PERIMENEI*/
        TEC[C]=BIG;
 }
 else if(ST[T]==2)
 {
        SC[C]=1;/*CAM KOBEI*/
        TEC[C]=CLOCK+TJC[OC[C]][JC[C]];
        NEC[C]=1;
        TET[T]=TEC[C]-TCLEAR;
        NET[T]=2;
 }



}

eye()
{
 FILE *ptr1,*ptr2,*ptr3;
 int  i,j;


 ptr1=fopen("fanco1.txt","r");
 ptr2=fopen("fanco2.txt","w");
```

```
ptr3=fopen("fanco3.txt","w");

if(ptr1==NULL)
printf("problem opening fanco1.txt");
if(ptr2==NULL)
printf("problem opening fanco2.txt");

fscanf(ptr1,"%d %f",&NO,&TCLEAR);
fprintf(ptr2,"%d %f",NO,TCLEAR);

for (i=1;i<=NO;i++)
{
fscanf(ptr1,"%d",&SEQ[i]);
}
fprintf(ptr2,"\n");

for (i=1;i<=10;i++)
{
fprintf(ptr2,"%d ",SEQ[i]);
}

fscanf(ptr1,"%d%d%d",&NT,&NMT,&NMC);
fprintf(ptr2,"\n");
fprintf(ptr2,"%d %d %d",NT,NMT,NMC);

for (i=1;i<=NO;i++)
{
        fscanf(ptr1,"%d",&NJ[i]);
        for(j=1;j<=NJ[i];j++)
        fscanf(ptr1,"%f%f",&TJT[i][j],&TJC[i][j]);
}

fprintf(ptr2,"\n\n");

for (i=1;i<=NO;i++)
{
        fprintf(ptr2,"%d",NJ[i]);
        fprintf(ptr2,"\n");
        for(j=1;j<=NJ[i];j++)
        {       fprintf(ptr2,"\n");
                fprintf(ptr2,"%f %f",TJT[i][j],TJC[i][j]);
        }

        fprintf(ptr2,"\n\n");
}


for(i=0;i<=NT+1;i++)
{
        for(j=0;j<=NT+1;j++)
        fscanf(ptr1,"%f",&SETUP[i][j]);
}
fprintf(ptr2,"\n\n");

for(i=0;i<=NT+1;i++)
{
        for(j=0;j<=NT+1;j++)
        fprintf(ptr2,"%f ",SETUP[i][j]);
        fprintf(ptr2,"\n\n");
}
for(i=1;i<=NT;i++)
fscanf(ptr1,"%d",&PR[i]);
fprintf(ptr2,"\n\n");

for(i=1;i<=7;i++)
fprintf(ptr2,"%d ",PR[i]);


for (i=1;i<=NT;i++)
{
        for(j=1;j<=NC;j++)
        fscanf(ptr1,"%d",&PRC[i][j]);
}

fprintf(ptr2,"\n\n");
for (i=1;i<=NT;i++)
{
```

```
        for(j=1;j<=NC;j++)
        fprintf(ptr2,"%d ",PRC[i][j]);

     fprintf(ptr2,"\n");
 }

 for (i=1;i<=NC;i++)
 {
        for(j=1;j<=NT;j++)
        fscanf(ptr1,"%d",&PRT[i][j]);
 }

 fprintf(ptr2,"\n\n");

 for (i=1;i<=NC;i++)
 {
        for(j=1;j<=NT;j++)
        fprintf(ptr2,"%d ",PRT[i][j]);

        fprintf(ptr2,"\n");
 }


}
```

Δ. Γεννήτρια Ψευδοτυχαίων Αριθμών

```
zrand()
{


        XHI=(IE/65536);
 XALO=(IE-XHI*65536)*24112;
 LEFTLO=XALO/65536;
 FHI=XHI*24112+LEFTLO;
 K=FHI/32768;
 IE=(((XALO-LEFTLO*65536)-2147483647)+(FHI-K*32768)*65536)+K;
 if(IE<0)
        IE=IE+2147483647;
 XHI=IE/65536;
 XALO=(IE-XHI*65536)*26143;
 LEFTLO=XALO/65536;
 FHI=XHI*26143+LEFTLO;
 K=FHI/32768;
 IE=(((XALO-LEFTLO*65536)-2147483647)+(FHI-K*32768)*65536)+K;
 if(IE<0)
        IE=IE+2147483647;
 Z=(2*(IE/256)+1)/16777216;

}
```