Πολυτεχνείο Κρήτης - Σχολή Μηχανικών Παραγωγής & Διοίκησης
Πρόγραμμα Μεταπτυχιακών Σπουδών στην «Επιχειρησιακή Έρευνα»

Μεταπτυχιακή Διατριβή

Εργασία που υπεβλήθη για τη μερική ικανοποίηση των απαιτήσεων για την απόκτηση μεταπτυχιακού διπλώματος ειδίκευσης

Ανάπτυξη εξελικτικού αλγορίθμου και βελτιστοποίηση παραμέτρων λειτουργίας του για προβλήματα χρονοπρογραμματισμού εργασιών

Εισηγητής:
Νικόλαος Μακρυμανωλάκης

Τριμελής Επιτροπή:
Επικ. Καθ. Ιωάννης Μαρινάκης (επιβλέπων)
Καθ. Γεώργιος Σταυρουλάκης
Επικ. Καθ. Ευστράτιος Ιωαννίδης

Χανιά Ιούνιος 2015

Η ολοκλήρωση της συγκεκριμένης επιστημονικής εργασίας για «Ανάπτυξη εξελικτικού αλγορίθμου και βελτιστοποίηση παραμέτρων λειτουργίας του για προβλήματα χρονοπρογραμματισμού εργασιών», συγχρηματοδοτήθηκε μέσω του Έργου «Υποτροφίες ΙΚΥ» από πόρους του ΕΠ «Εκπαίδευση και Δια Βίου Μάθηση», του Ευρωπαϊκού Κοινωνικού Ταμείου (ΕΚΤ) του ΕΣΠΑ, 2007-2013.

Περίληψη

Στην εργασία αυτή δημιουργούμε ένα νέο εξελικτικό αλγόριθμος κατάλληλο για επίλυση προβλημάτων συνδυαστικής βελτιστοποίησης αναδιάταξης εργασιών / σημείων, όπως τα προβλήματα χρονοπρογραμματισμού εργασιών, το πρόβλημα του περιπλανώμενου πωλητή και άλλα. Εστιάζουμε τη μελέτη μας στο πρόβλημα χρονοπρογραμματισμού εργασιών flow-shop, όπου ένα πλήθος n εργασιών πρέπει να τύχει επεξεργασίας από ένα πλήθος m μηχανών, με την ίδια σειρά για όλες τις εργασίες, και αναζητούμε το μικρότερο χρόνο ολοκλήρωσης. Ο αλγόριθμος συνδυάζει τεχνικές από αυτές που εφαρμόζονται στη διαδικασία τοπικής αναζήτησης. Καθώς ο αλγόριθμος είναι δεκτικός παραμετροποίησης για ορισμένα στοιχεία που αφορούν στη λειτουργία τους, ακολουθείται μία συστηματική διαδικασία εξόρυξης δεδομένων, με τη βοήθεια της οποίας αξιοποιούνται δεδομένα μετρήσεων από έναν αριθμό εκτελέσεων του αλγορίθμου σε πραγματικά προβλήματα, και αναζητούνται μοντέλα που θα εκτιμήσουν την κατάλληλη παραμετροποίηση του αλγορίθμου για κάθε μέγεθος προβλήματος m x n. Στο τέλος παρατίθενται τα αποτελέσματα με βάση τις προτεινόμενες παραμετροποιήσεις από τα μοντέλα εξόρυξης δεδομένων και γίνεται επαλήθευση της ποιότητας των αποτελεσμάτων. Τα αποτελέσματα δείχνουν ότι η διαδικασία που ακολουθήθηκε δημιουργεί νέες προοπτικές στη βελτίωση της αποδοτικότητας εξελικτικών αλγορίθμων για προβλήματα συνδυαστικής βελτιστοποίησης αλλά και σε άλλες εφαρμογές όπου η παραμετροποίηση αλγορίθμων είναι σημαντικό στοιχείο για την αύξηση της αποτελεσματικότητάς τους.

Abstract

In this thesis we develop a new evolutionary algorithm, suitable for solving combinatorial optimization scheduling problems, such as jobs scheduling problems, the travelling salesperson problem, etc. We focus our study on flow-shop scheduling problem, where a number of n jobs has to be processed by a number of m machines, at the same sequence for every job, and we seek for smallest completion time. The algorithm combines various techniques used in local search. As various elements of the algorithm can be tuned, we follow a systematic data mining procedure and we utilize data from a number of executions in real problems, in order to seek models for the suitable parameterization for every $m \; x \; n$ problem size. Finally we present our results using the model suggested parameters and we verify the quality of our results. The results show that the procedure we follow creates new promises on the improvement in efficiency for evolutionary algorithms in combinatorial optimization problems, and also in other applications where the fine-tuning of algorithms is important element in order to increase their efficiency.

Περιεχόμενα

"Time is money."

Benjamin Franklin

Κεφάλαιο 1:
Εισαγωγή

Βελτιστοποίηση είναι η διαδικασία με την οποία επιδιώκεται το καλύτερο δυνατό αποτέλεσμα σε μία ενέργεια, κάτω από συγκεκριμένες και δεδομένες συνθήκες. Η χρήση της βελτιστοποίησης γίνεται κατά τη διαδικασία λήψης αποφάσεων σε πληθώρα περιπτώσεων, όπως στο σχεδιασμό, στην κατασκευή, στα οικονομικά, στη λειτουργία συστημάτων και στόχο έχει να υποβοηθήσει τη διαδικασία λήψης απόφασης. Στόχος είναι είτε η ελαχιστοποίηση του *κόστους* ή η μεγιστοποίηση του *οφέλους*, ό,τι κι εάν σημαίνει «κόστος» ή «όφελος» σε κάθε συγκεκριμένη εφαρμογή (λ.χ. χρηματικό, χρονικό, απόστασης, ποιοτικό κ.λπ.).

Η *Συνδυαστική Βελτιστοποίηση* αναφέρεται σε προβλήματα ή καταστάσεις όπου αναζητείται μία βέλτιστη διακριτή λύση μεταξύ πεπερασμένου ή άπειρου αριθμού επιλογών. Κάθε δυνατή διακριτή λύση, όπως προκύπτει από τα χαρακτηριστικά του προβλήματος και τους περιορισμούς που αυτό θέτει, έχει μία αξία ως προς το κόστος ή το όφελος που δίνει στο πρόβλημα. Με βάση αυτό το κόστος ή το όφελος επιλέγεται αυτή που εντοπίζεται να δίνει το καλύτερο αποτέλεσμα.

Τα προβλήματα χρονοπρογραμματισμού εργασιών σε μηχανές (machine scheduling) αναφέρονται στη βελτιστοποίηση χρονοπρογραμματισμού μιας σειράς εργασιών (jobs) οι οποίες πρέπει να επεξεργαστούν από διάφορες μηχανές (machines)[1]. Στόχος της διαδικασίας είναι η βελτιστοποίηση μιας κρίσιμης παραμέτρου, όπως λχ η ελαχιστοποίηση του συνολικού χρόνου επεξεργασίας, η ελαχιστοποίηση του χρόνου αναμονής των μηχανών κ.ά. Τα σχετικά προβλήματα μπορεί να είναι απεριόριστα (unconstrained), δηλαδή χωρίς περιορισμούς, ή με περιορισμούς (constrained) που να

[1] Στην πράξη, οι έννοιες τόσο των εργασιών όσο και των μηχανών μπορούν να αφορούν έννοιες εκτός του «βιομηχανικού» χώρου, όπως λ.χ. στην εξυπηρέτηση πελατών ή στο χρονοπρογραμματισμό εργασιών στην κεντρική μονάδα επεξεργασίας ενός ηλεκτρονικού υπολογιστή. Άλλωστε, τόσο αυτό όσο και πολλά άλλα προβλήματα της συνδυαστικής βελτιστοποίησης, έχουν συμβολικό χαρακτήρα ως προς το περιεχόμενό τους.

απαιτούν ότι η λύση που θα δοθεί θα πρέπει να τους ικανοποιεί στο σύνολό τους (λ.χ. καμία μηχανή να μην παραμείνει ανενεργή περισσότερο από Χ χρόνο).

Από τη δεκαετία του 1950 και μετά υπάρχει εκτεταμένη ερευνητική δραστηριότητα στην περιοχή αυτή, σε εφαρμογές που εκτείνονται πέραν των υπολογιστών και της βιομηχανίας, όπως μεταφορές, νοσοκομεία, αγροτική παραγωγή κ.λπ., διατηρώντας όμως το στόχο της βέλτιστης διάθεσης των πόρων.

Η συγκεκριμένη κατηγορία προβλημάτων, όπως και πολλά άλλα προβλήματα συνδυαστικής βελτιστοποίησης, χαρακτηρίζονται ως NP-hard (Non-deterministic Polynomial time – hard) – (Anderson et al. 1997). Πρόκειται για προβλήματα στα οποία η λύση τους δεν μπορεί να βρεθεί σε πολυωνυμικό χρόνο, δηλαδή η επίλυσή τους είναι εκθετικής πολυπλοκότητας, με αποτέλεσμα να αυξάνει δραματικά ο χρόνος με την αύξηση του μεγέθους του προβλήματος. Παράδειγμα προβλήματος που λύνεται σε πολυωνυμικό χρόνο είναι η εύρεση της ελάχιστης διαδρομής σε ένα γράφο. Παράδειγμα προβλήματος που δε λύνεται σε πολυωνυμικό χρόνο είναι η εύρεση της μακρύτερης διαδρομής σε ένα γράφο. Από το παραπάνω ενδεικτικό παράδειγμα βλέπουμε ότι τα NP-hard προβλήματα δεν είναι απαραίτητα εντελώς διαφορετικής λογικής από τα P προβλήματα, αλλά η φύση τους είναι τέτοια που τα καθιστά διαφορετικά στη διαδικασία επίλυσής τους. Πάντως, αν και δεν είναι πολυωνυμικά προσδιορισμένος ο χρόνος εύρεσης της βέλτιστης λύσης στα *NP-hard προβλήματα*, εντούτοις συνήθως υπάρχει πολυωνυμικά προσδιορισμένος χρόνος για να δούμε εάν μία λύση είναι εντός του συνόλου των εφικτών και να προσδιορίσουμε την αξία της. Η ιδιότητα αυτή θα αποδειχθεί ιδιαίτερη χρήσιμη, όπως θα διαπιστώσουμε στην πορεία.

Κατά τον Yannarakis (1997), τα προβλήματα όπως το flow-shop καθώς και άλλα προβλήματα που αντιμετωπίζουμε με αλγόριθμους τοπικής αναζήτησης, ανήκουν σε μία κατηγορία που ονομάζει PLS, δηλαδή polynomial-time local search. Σύμφωνα με τον ερευνητή, ένα πρόβλημα Π τοπικής αναζήτησης εάν υπάρχουν τρεις αλγόριθμοι A_Π, B_Π και Γ_Π πολυωνυμικού χρόνου, με τα ακόλουθα χαρακτηριστικά:

1. Ο A_Π να ελέγχει εάν μία λύση s ανήκει στο χώρο λύσεων του Π.
2. Ο B_Π να υπολογίζει το κόστος της λύσης s.
3. Ο Γ_Π να μπορεί να εντοπίσει για κάθε λύση s μία γειτονική λύση.

Το πρόβλημα flow-shop, όπως θα διαπιστώσουμε και στη συνέχεια, διαθέτει τέτοιους αλγορίθμους A_{Π}, B_{Π} και Γ_{Π} σε πολυωνυμικό χρόνο, τους οποίους άλλωστε θα χρησιμοποιήσουμε για την υλοποίηση του προτεινόμενου αλγορίθμου.

Εκεί, για παράδειγμα, θα διαπιστώσουμε ότι ένα πρόβλημα 10 εργασιών σε 5 μηχανές απαιτεί το διπλάσιο χρόνο από ένα πρόβλημα 5 εργασιών σε 5 μηχανές για να διαπιστώσουμε την αξία μιας λύσης τους. Όμως, τα δεδομένα δεν είναι πολυωνυμικά ως προς το μέγεθος του χώρου λύσεων.

Σε ένα πρόβλημα «μικρού» μεγέθους, η εξαντλητική λύση είναι εφικτή. Για παράδειγμα, εάν έχουμε 2 εργασίες, οι πιθανές λύσεις είναι δύο (2!) οπότε σχετικά εύκολα υπολογίζουμε την αξία κάθε μίας λύσης και επιλέγουμε αυτή που δίνει καλύτερο αποτέλεσμα με βάση το ζητούμενο. Στις 3 εργασίες, οι πιθανές λύσεις γίνονται 6 (δηλαδή 3!), ενώ στις 4 εργασίες οι πιθανές λύσεις γίνονται 24 (δηλαδή 4!). Όμως ήδη, απλά διπλασιάζοντας τις εργασίες από 2 σε 4, έχουμε 12πλασιάσει τις πιθανές λύσεις. Εάν προσθέσουμε ελάχιστες ακόμα εργασίες, ο αριθμός των πιθανών λύσεων θα «εκτοξευτεί» και θα καταστεί απαγορευτικός για εξαντλητική αξιολόγηση[2].

Για τη λύση σε τέτοιου είδους, μεγέθους και πολυπλοκότητας προβλήματα, επιστρατεύονται κυρίως αλγόριθμοι *τοπικής αναζήτησης (local search)*, οι οποίοι εφαρμόζουν διάφορες στρατηγικές αναζήτησης, με την προσδοκία να φτάσουμε σε μία περιοχή «καλών λύσεων» κι εκεί να αναζητήσουμε την καλύτερη λύση από αυτές. Πρόκειται για *αλγόριθμους καθολικής βελτιστοποίησης (global optimization algorithms)*, οι οποίοι κοιτούν το σύνολο του χώρου λύσεων, ενώ με συγκεκριμένες διαδικασίες, διαφορετικές σε κάθε αλγόριθμο, προσπαθούν να αποτρέψουν τον εγκλωβισμό τους σε τοπικά ελάχιστα.

Τις τελευταίες δεκαετίες έχει αναπτυχθεί πλήθος αλγορίθμων *ευρετικού* ή *μεθευρετικού* χαρακτήρα για προβλήματα συνδυαστικής βελτιστοποίησης. Ως *ευρετικός (heuristic)* θεωρείται ένας αλγόριθμος ο οποίος αξιοποιεί σε μεγάλο βαθμό τα ιδιαίτερα χαρακτηριστικά ενός προβλήματος, και με βάση αυτά προσπαθεί μία καλή λύση. Η

[2] Είναι χαρακτηριστικό ότι σε ένα πρόβλημα με 20 εργασίες, δηλαδή 2.432.902.008.176.640.000 λύσεις (20!) με έναν κώδικα σε C που υπολογίζει σε ένα δευτερόλεπτο την αξία 1.000.000 λύσεων, θα χρειαζόμασταν περίπου 77 χιλιάδες χρόνια υπολογισμών για να εντοπίσουμε την καλύτερη λύση, αξιολογώντας όλες τις πιθανές λύσεις! Τα δε προβλήματα με 20 εργασίες, θεωρούνται «μικρά» στο πλαίσιο της έρευνας που διεξάγεται, καθιστώντας εμφανές ότι η εξαντλητική μελέτη είναι ανεφάρμοστη.

διαδικασία είναι σε μεγάλο βαθμό ντετερμινιστική η δε λύση που προσφέρει ο αλγόριθμος (συνήθως ίδια σε κάθε εκτέλεση) επιδέχεται βελτίωσης.

Ως *μεθευρετικός (metaheuristic)* θεωρείται ένας αλγόριθμος που εστιάζει στη διαδικασία αναζήτησης στο χώρο λύσεων, κατά κανόνα χωρίς να ενσωματώνει[3] σε αυτήν ιδιαίτερη γνώση από τα χαρακτηριστικά του προβλήματος (Anderson et al. 1997). Το ίδιο το πρόβλημα θεωρείται για τον αλγόριθμο ένα «μαύρο κουτί», και κάθε λύση που εντοπίζει ο αλγόριθμος αξιολογείται από το «μαύρο κουτί» ως προς την αξία που έχει για το στόχο βελτιστοποίησης (λ.χ. χρόνος εκτέλεσης). Συνεπώς, οι μεθευρετικές τεχνικές μπορούν να χρησιμοποιηθούν –συχνά με ελαχιστότατες μετατροπές- σε πλήθος διαφορετικών προβλημάτων συνδυαστικής βελτιστοποίησης, καθιστώντας τις πιο καθολικά εργαλεία σε σχέση με τις ευρετικές αλγορίθμους. Επίσης, οι μεθευρετικοί αλγόριθμοι ενσωματώνουν «νοημοσύνη», συχνά μέσω στοχαστικών στοιχείων που διαθέτουν, που τους επιτρέπει να κινούνται σε διάφορες περιοχές του χώρου των εφικτών λύσεων, ξεπερνώντας τοπικά ελάχιστα και οδηγούμενοι προς τις περιοχές με τις ολικά καλύτερες λύσεις. Φυσικά, το ότι οι μεθευρετικοί αλγόριθμοι δεν «γνωρίζουν» τα χαρακτηριστικά του προβλήματος που καλούνται να επιλύσουν, δε σημαίνει ότι δεν είναι δεκτικοί ρυθμίσεων (fine tuning) που θα τους καταστήσει πιο αποτελεσματικούς στη λειτουργία τους επί συγκεκριμένων προβλημάτων. Όπως θα δούμε στη συνέχεια, οι ρυθμίσεις αυτές διαδραματίζουν καθοριστικό ρόλο στην απόδοση του αλγορίθμου, και ως προς την εύρεση καλύτερης λύσης αλλά και ως προς το χρόνο εκτέλεσης. Διότι, είναι εύλογο, ότι ενδιαφερόμαστε μεν για την καλύτερη λύση, αλλά στο συντομότερο δυνατό χρόνο. Οπότε η ισορροπία μεταξύ «βελτίωσης λύσης / χρόνου εκτέλεσης» είναι σχεδόν πάντα ζητούμενη (Taha, 2014).

Στη βιβλιογραφία υπάρχει ένα πλήθος μεθευρετικών προσεγγίσεων. Ενδεικτικά να αναφέρουμε αλγόριθμους εμπνευσμένους από τη φύση, όπως λ.χ. τους γενετικούς (genetic), τα νευρωνικά δίκτυα (neural networks), την προσομοιωμένη ανόπτιση (simulated annealing), την αποικία μυρμηγκιών (ant colony), το σμήνος πουλιών (particle swarm optimization) κ.ά., αλλά και αλγόριθμοι που αποτελούν ξεχωριστές ιδέες, όπως λ.χ. η περιορισμένη αναζήτηση (tabu search, Glover et al. (1993, 1997)), ο GRASP κ.ά.

[3] Ακριβέστερα, οι τεχνικές των μεθευρετικών αλγορίθμων μπορούν να χρησιμοποιηθούν σε πληθώρα προβλημάτων, ανεξάρτητα από τη φύση των προβλημάτων αυτών. Όμως, σε συγκεκριμένες υλοποιήσεις, οι δημιουργοί των αλγορίθμων μπορεί να ενσωματώσουν σε αυτούς στοιχεία από το πρόβλημα, καθιστώντας τον αλγόριθμο αποτελεσματικό σε συγκεκριμένη κατηγορία προβλημάτων. Αυτό δεν αλλάζει τη λογική ότι οι μεθευρετικές τεχνικές μπορούν να εφαρμοστούν σε πληθώρα εφαρμογών.

Η ιδέα πίσω από έναν αλγόριθμο δίνει τη βασική κατεύθυνση στην υλοποίησή του. Όμως, κάθε υλοποίηση μπορεί να έχει τα δικά της χαρακτηριστικά που την καθιστούν μοναδική, ως προς τις επιλογές που κάνει ο σχεδιαστής της. Επίσης, είναι ενδιαφέρον ότι οι παραπάνω αλγοριθμικές ιδέες μπορούν να συνδυαστούν μεταξύ τους, δημιουργώντας νέες προσεγγίσεις. Ο τελικός κριτής κάθε προσέγγισης είναι φυσικά το αποτέλεσμα, στην επίλυση διαφόρων προβλημάτων. Όμως, παράλληλα, η φαντασία στις υλοποιήσεις μπορεί να ανοίξει και νέους δρόμους στην έρευνα, δημιουργώντας νέες κατηγορίες αλγορίθμων ή βελτιώνοντας την απόδοση γνωστών αλγοριθμικών προσεγγίσεων.

1.1 Το αντικείμενο της εργασίας

Έχοντας προηγηθεί μία εισαγωγή σε βασικές θεωρητικές έννοιες, προχωρούμε σε μία σύντομη αναφορά στα όσα διαπραγματεύεται η συγκεκριμένη εργασία.

Εστιάζοντας σε μία συγκεκριμένη κατηγορία προβλημάτων χρονοπρογραμματισμού, αναπτύσσουμε έναν μεθευρετικό αλγόριθμο για την επίλυση τέτοιων προβλημάτων. Ο αλγόριθμος αυτός συνδυάζει στοιχεία από διάφορες αλγοριθμικές ιδέες, ώστε να αυξήσει την αποτελεσματικότητά του. Η αποτελεσματικότητα του αλγορίθμου δοκιμάζεται σε μία σειρά από *δοκιμαστικά σετ (test sets)*, τα οποία χρησιμοποιούνται ευρέως από τους ερευνητές για τη συγκεκριμένη κατηγορία προβλημάτων χρονοπρογραμματισμού. Αυτό μας επιτρέπει να συγκρίνουμε την αποδοτικότητα του αλγορίθμου μας με άλλες προσεγγίσεις που έχουν υλοποιηθεί και δημοσιευτεί.

Το αμέσως επόμενο ζήτημα που προκύπτει είναι η παραμετροποίηση του αλγορίθμου. Ο συγκεκριμένος αλγόριθμος –αλλά και η πλειονότητα αντίστοιχων αλγορίθμων- απαιτεί μία σειρά από *ρυθμίσεις (fine tuning)*, ώστε να καταστεί πιο αποτελεσματικός στα διάφορα είδη προβλημάτων που τον εφαρμόζουμε. Στο πλαίσιο της εργασίας αυτής επιχειρείται μία συστηματική και εξαντλητική διαδικασία ρύθμισης του αλγορίθμου, με στοιχεία από χιλιάδες εκτελέσεις του, τα οποία επεξεργαζόμαστε μέσω μοντέλων *εξόρυξης δεδομένων (data mining)*. Το αποτέλεσμα αυτής της διαδικασίας είναι άκρως διαφωτιστικό ως προς τις κατάλληλες ρυθμίσεις για κάθε τύπο προβλήματος. Η διαφορά στην απόδοση του αλγορίθμου πριν και μετά τις ακριβείς ρυθμίσεις, είναι πραγματικά εντυπωσιακή.

Στο σημείο αυτό να σημειώσουμε ότι η διαδικασία του fine tuning μπορεί να χρησιμοποιηθεί σε οποιοδήποτε αλγόριθμο, ακόμα και κάποιον που δεν υλοποιούμε εμείς. Άρα, η όλη διαδικασία έχει μεγάλη αξία, καθώς είναι συχνό φαινόμενο στη βιβλιογραφία η ρύθμιση των αλγορίθμων να γίνεται εμπειρικά ή στο πλαίσιο «παρατηρήσεων με το μάτι». Οι παρατηρήσεις αυτές είναι μεν ενδεικτικές της κατεύθυνσης που πρέπει να ακολουθήσουν οι ρυθμίσεις, αλλά ενδεχομένως απέχουν πολύ από το να θεωρηθούν ότι οδηγούν τον αλγόριθμο στα πραγματικά όριά του. Είναι σύνηθες αλγόριθμοι να αποδίδουν καλά σε κάποιες κατηγορίες προβλημάτων και όχι τόσο καλά σε κάποιες άλλες. Ο ισχυρισμός ότι αυτό δεν οφείλεται σε αδυναμία του αλγορίθμου αλλά σε ελλιπή διαδικασία fine tuning αυτού, δεν πρέπει να απέχει πολύ από την πραγματικότητα.

Τέλος, έχοντας υλοποιήσει τον αλγόριθμο και έχοντας «αποσπάσει» με το data mining τις βέλτιστες παραμέτρους λειτουργίας του, προχωράμε στις δοκιμές επί των test sets. Εκεί συγκρίνουμε την απόδοση του αλγορίθμου με βάση τις μέχρι σήμερα γνωστές καλύτερες λύσεις. Με μία σειρά από δοκιμές, επιδιώκουμε να δείξουμε ότι οι παραμετροποιήσεις που λάβαμε από την εξόρυξη δεδομένων υπερέχει έναντι άλλων παραμετροποιήσεων που θα μπορούσαμε να επιλέξουμε.

Ολοκληρώνουμε την εργασία με την εξαγωγή συμπερασμάτων καθώς και με αναφορά σε θέματα για περαιτέρω μελέτη, εστιάζοντας τόσο στον ίδιο τον αλγόριθμο που υλοποιήσαμε όσο και στη διαδικασία τής παραμετροποίησής του με τη διαδικασία της εξόρυξης δεδομένων και τις προοπτικές που αυτή ανοίγει.

"Everything requires time.
It is the only truly universal condition.
All work takes place in time and uses up time.
Yet most people take for granted this unique,
irreplaceable, and necessary resource.
Nothing else, perhaps,
distinguishes effective executives as much
as their tender loving care of time."

Peter F. Drucker

Κεφάλαιο 2:
Τα προβλήματα χρονοπρογραμματισμού

Στο κεφάλαιο αυτό θα παρουσιάσουμε το συγκεκριμένο πρόβλημα χρονοπρογραμματισμού, το οποίο διερευνούμε στο πλαίσιο αυτής της εργασίας. Καθώς το πρόβλημα αυτό είναι μέρος μιας ευρύτερης οικογένειας προβλημάτων χρονοπρογραμματισμού, τα οποία και μελετώνται στη βιβλιογραφία, κρίνεται σκόπιμο να προηγηθεί μία ευρύτερη αναφορά στη φύση και στις κατηγορίες των προβλημάτων αυτών. Η αναφορά αυτή, μεταξύ άλλων, θα μας τροφοδοτήσει και με τα απαραίτητα στοιχεία μοντελοποίησης του προβλήματος, τα οποία θα χρησιμοποιήσουμε στη συνέχεια της μελέτης μας.

2.1 Μοντέλα χρονοπρογραμματισμού

Σε γενικές γραμμές, ένα μοντέλο χρονοπρογραμματισμού εργασιών ακολουθεί το σενάριο ότι υπάρχει m αριθμός μηχανών, οι οποίες χρησιμοποιούνται για την επεξεργασία n αριθμού εργασιών. Ο προγραμματισμός (schedule) καθορίζει για κάθε

μηχανή i και για κάθε εργασία j, ένα ή περισσότερα παράθυρα χρόνου στα οποία η εργασία j εκτελείται στη μηχανή i.

Ένας προγραμματισμός είναι εφικτός όταν δεν υπάρχουν επικαλύψεις στα παράθυρα χρόνου που αναφέρονται στην ίδια εργασία ή στην ίδια μηχανή. Δηλαδή, μία εργασία δεν μπορεί να εκτελείται ταυτόχρονα από δύο ή περισσότερες μηχανές και μία μηχανή δεν μπορεί να εκτελεί ταυτόχρονα δύο ή περισσότερες εργασίες. Επίσης, κάθε πρόβλημα μπορεί να θέτει και άλλους περιορισμούς εφικτότητας, τους οποίους πρέπει να ικανοποιεί ο προτεινόμενος προγραμματισμός. Για παράδειγμα, μπορεί να απαιτείται από το πρόβλημα κάποιες εργασίες να εκτελούνται απαραιτήτως μετά από συγκεκριμένες άλλες εργασίες, ή κάποιες εργασίες να έχουν ολοκληρώσει την επεξεργασία τους από κάποιες μηχανές εντός συγκεκριμένου συνολικού διαστήματος, ή να μην επιτρέπεται οι μηχανές να μείνουν ανενεργές περισσότερο από κάποιο Χ χρονικό διάστημα κ.ο.κ.

Ως προς το περιβάλλον των μηχανών, καταρχήν σε κάθε περίπτωση θεωρούμε ότι όλες οι μηχανές είναι διαθέσιμες για επεξεργασία εργασιών τη χρονική στιγμή t_0. Υπάρχουν προβλήματα *ενός σταδίου (single-stage)* όπου κάθε εργασία απαιτεί επεξεργασία από μία μόνο μηχανή και προβλήματα *πολλαπλών σταδίων (multiple-stage)* όπου κάθε εργασία απαιτεί επεξεργασία από περισσότερες μηχανές. Επίσης, υπάρχουν προβλήματα με μία μηχανή και προβλήματα με περισσότερες μηχανές που εκτελούν την ίδια εργασία. Στα single-stage προβλήματα, εάν έχουμε περισσότερες από μία μηχανές αυτές θεωρείται ότι λειτουργούν παράλληλα και έχουν την ίδια λειτουργία. Βέβαια κι εδώ μπορούμε να θεωρήσουμε τρεις υποπεριπτώσεις, τις *πανομοιότυπες παράλληλες μηχανές (identical parallel machines)* όπου οι μηχανές είναι πανομοιότυπες ως προς τα χαρακτηριστικά τους, τις *παρόμοιες παράλληλες μηχανές (uniform parallel machines)* όπου οι μηχανές έχουν ίδια λειτουργία αλλά διαφορετική ταχύτητα και τις *ασυσχέτιστες παράλληλες μηχανές (unrelated parallel machines)* όπου οι μηχανές έχουν διαφορετική ταχύτητα και λειτουργία, ανάλογα με την εργασία που εκτελούν (Anderson et al. 1997). Στα multi-stage προβλήματα, όπου πλέον είναι βέβαιο ότι έχουμε περισσότερες από μία μηχανές, διακρίνουμε πάλι τρεις κύριες κατηγορίες. Στα προβλήματα τύπου *job-shop*, κάθε εργασία πρέπει να περάσει με συγκεκριμένη και προκαθορισμένη σειρά από τις μηχανές, όπου η σειρά μπορεί να είναι διαφορετική από εργασία σε εργασία. Στα προβλήματα τύπου *open-shop* κάθε εργασία πρέπει να περάσει από όλες τις μηχανές, αλλά η σειρά με την οποία κάθε εργασία θα περάσει από

τις μηχανές δεν είναι προκαθορισμένη και αποτελεί μέρος της λύσης. Τέλος, στα προβλήματα τύπου *flow-shop*, κάθε εργασία πρέπει να περάσει από όλες τις μηχανές, με την ίδια σειρά. Με βάση τα παραπάνω καθίσταται εμφανές ότι εάν σε ένα πρόβλημα job-shop η σειρά εξυπηρέτησης από τις μηχανές είναι ίδια για όλες τις εργασίες, τότε έχουμε ένα πρόβλημα flow-shop. Επίσης, ένα πρόβλημα open-shop είναι ουσιαστικά ένα πρόβλημα job-shop, χωρίς συγκεκριμένα δεδομένα ως προς την υποχρεωτική σειρά με την οποία κάθε εργασία πρέπει να περάσει από τις μηχανές (Anderson et al. 1997).

Ως προς τις εργασίες, για κάθε εργασία j μας δίνεται ο χρόνος που απαιτεί στη μηχανή i, και συμβολίζεται με p_{ij}. Από εκεί και πέρα μπορεί να δίνονται και άλλοι περιορισμοί, όπως λ.χ. η εργασία j να έχει τελειώσει την επεξεργασία της πριν από χρόνο d_j, να υπάρχει χρόνος προετοιμασίας της μηχανής i για να δεχθεί την εργασία k μετά την εκτέλεση της εργασίας j (χρόνος t_{ijk}), η σειρά εκτέλεσης των εργασιών στις μηχανές (αν είναι ελεύθερη, ίδια για όλες τις εργασίες ή διαφορετική ανά εργασία) κ.ο.κ. Επίσης, υπάρχει και κατηγορία *προεκτοπιστικών (preemptive)* προβλημάτων, στα οποία επιτρέπεται η διακοπή μιας εργασίας και συνέχισή της αργότερα, με ή χωρίς χρονική ποινή (Anderson et al. 1997).

Από τα προαναφερθέντα είναι εμφανές ότι υπάρχει πληθώρα προβλημάτων χρονοπρογραμματισμού εργασιών, όπως αυτά μπορούν να προκύψουν από το συνδυασμό των παραπάνω παραμέτρων, των οποίων μάλιστα η αναφορά στην παράγραφο αυτή δεν είναι εξαντλητική.

2.2 Κριτήρια βελτιστοποίησης

Σε κάθε πρόβλημα μπορεί να μας ενδιαφέρει η βελτιστοποίηση (συνήθως ελαχιστοποίηση) συγκεκριμένου μεγέθους. Ορισμένα από τα κριτήρια που χρησιμοποιούνται συνήθως είναι (Anderson et al. 1997):

- Ελαχιστοποίηση του μέγιστου χρόνου ολοκλήρωσης $C_{max}=max\ C_j$ (ο χρόνος που ολοκληρώνεται και η τελευταία εργασία).
- Ελαχιστοποίηση της μέγιστης βραδύτητας (lateness) στο χρόνο εκτέλεσης μιας εργασίας, σε σχέση με το χρόνο που αυτή απαιτεί $L_{max}=max\ (C_j-d_j)$, όπου d_j ο χρόνος εκτέλεσης που απαιτεί η εργασία j.

- Ελαχιστοποίηση της σταθμισμένης απόκλισης (tardiness) στο χρόνο υλοποίησης των εργασιών $\Sigma_j(w_j)max\{C_j\text{-}d_j, 0\}$, όπου w_j το βάρος της εργασίας j.
- Ελαχιστοποίηση του σταθμισμένου συνολικού χρόνου εκτέλεσης των εργασιών $\Sigma_j(w_j)C_j$, όπου w_j όπως παραπάνω.
- Ελαχιστοποίηση του σταθμισμένου αριθμού των καθυστερημένων εργασιών $\Sigma_j(w_j)\begin{Bmatrix}1\ \alpha\nu\ C_j > d_j\\0\end{Bmatrix}$

Να σημειώσουμε στο σημείο αυτό ότι ο λόγος ύπαρξης διαφόρων κριτηρίων βελτιστοποίησης και δεν αρκούμαστε λ.χ. απλά στη βελτιστοποίηση ως προς το συνολικό χρόνο, είναι διότι ανάλογα με το σύστημα επεξεργασίας, συχνά μας ενδιαφέρει περισσότερο το έγκαιρο της ολοκλήρωσης μιας εργασίας, ή το να μην υπάρχουν αποκλίσεις από τους προγραμματισμένους χρόνους, ή το να μην υπάρχουν ανενεργοί χρόνοι στις μηχανές. Συνεπώς, βλέπουμε ότι υπάρχουν πολλές πτυχές που μπορούν να βελτιστοποιηθούν και κάθε φορά μας ενδιαφέρει μία ή περισσότερες από αυτές.

2.3 Συμβολική αναπαράσταση τύπου προβλήματος

Όπως αναφέρεται στο Anderson et al. (1997), από τους Graham et al. το 1979, προτάθηκε μία συμβολική αναπαράσταση του τύπου του προβλήματος με τρία περιγραφικά πεδία $\alpha|\beta|\gamma$. Στην αναπαράσταση αυτή το «α» αναπαριστά το περιβάλλον των μηχανών, το «β» τα χαρακτηριστικά των εργασιών και το «γ» το κριτήριο βελτιστοποίησης. Με δεδομένο ότι, όπως είδαμε και παραπάνω, μπορούν να δημιουργηθεί μεγάλος αριθμός προβλημάτων συνδυάζοντας τα διάφορα χαρακτηριστικά, η συμβολική αυτή αναπαράσταση εξυπηρετεί το να καθίσταται κάθε φορά σαφές το είδος και τα χαρακτηριστικά του προβλήματος που επεξεργαζόμαστε.

Το πρώτο πεδίο «α» αποτελείται από δύο στοιχεία, τα α_1 και α_2, τα οποία εξηγούνται παρακάτω (όπου **O** το κενό).

Το α_1 παίρνει τιμές από το σύνολο $\{\boldsymbol{O}, P, Q, R, F, O, J\}$ που σημαίνουν:

- **O**: Μία μηχανή
- P: Πανομοιότυπες παράλληλες μηχανές (identical parallel machines)
- Q: Παρόμοιες παράλληλες μηχανές (uniform parallel machines)
- R: Ασυσχέτιστες παράλληλες μηχανές (unrelated parallel machines)

- F: Flow-shop
- O: Open-shop
- J: Job-shop

Το α_2 παίρνει τιμές από το σύνολο $\{\boldsymbol{O}, m\}$ που σημαίνουν:

- $\boldsymbol{O}$: Ο αριθμός των μηχανών είναι απροσδιόριστος
- m: Υπάρχει συγκεκριμένος αριθμός m μηχανών

Το δεύτερο πεδίο «β» μπορεί να αποτελείται κατά μέγιστο από τέσσερα στοιχεία, $\beta \subseteq \{\beta_1, \beta_2, \beta_3, \beta_4\}$.

Το $\beta_1 \in \{\boldsymbol{O}, r_j\}$ όπου:

- $\boldsymbol{O}$: Δεν υπάρχει συγκεκριμένος χρόνος διαθεσιμότητας (release) των εργασιών
- r_j: Οι εργασίες έχουν συγκεκριμένο χρόνο διαθεσιμότητας.

Το $\beta_2 \in \{\boldsymbol{O}, d_j\}$ όπου:

- $\boldsymbol{O}$: Δεν υπάρχει συγκεκριμένος χρόνος ολοκλήρωσης (deadline) των εργασιών
- d_j: Οι εργασίες έχουν συγκεκριμένο χρόνο ολοκλήρωσης.

Το $\beta_3 \in \{\boldsymbol{O}, t_{jk}, t_j, s_{fg}, s_f\}$ όπου:

- $\boldsymbol{O}$: Δεν υπάρχουν χρόνοι προετοιμασίας (setup) των εργασιών
- t_{jk}: Υπάρχουν γενικοί χρόνοι προετοιμασίας εργασιών
- t_j: Υπάρχουν χρόνοι προετοιμασίας των εργασιών, ανεξάρτητοι από τη διαδοχή
- s_{fg}: Υπάρχουν χρόνοι προετοιμασίας οικογένειας εργασιών
- s_f: Υπάρχουν χρόνοι προετοιμασίας οικογένειας εργασιών, ανεξάρτητοι από τη διαδοχή

Το $\beta_4 \in \{\boldsymbol{O}, prec\}$ όπου:

- $\boldsymbol{O}$: Δεν υπάρχουν περιορισμοί αλληλουχίας των εργασιών
- $prec$: Οι εργασίες έχουν συγκεκριμένους περιορισμούς αλληλουχίας

Τέλος, το τρίτο στοιχείο «γ» ορίζει το κριτήριο βελτιστοποίησης, δηλαδή κάποιο από αυτά που είδαμε στην παράγραφο 2.2.

Για παράδειγμα, το $J||C_{max}$ είναι ένα πρόβλημα job-shop με στόχο την ελαχιστοποίηση του μέγιστου χρόνου ολοκλήρωσης εργασιών (αναγκαστικά πρόκειται για πρόβλημα multi-stage αφού είναι job-shop). Σε άλλο παράδειγμα το $P5|r_j|\Sigma C_j$ είναι ένα πρόβλημα με 5 πανομοιότυπες μηχανές που λειτουργούν παράλληλα, με εργασίες που καθίστανται διαθέσιμες σε συγκεκριμένο χρόνο και έχουμε στόχο την ελαχιστοποίηση του συνολικού χρόνου εκτέλεσης.

2.4 Το πρόβλημα χρονοπρογραμματισμού flow-shop

Το πρόβλημα που θα μας απασχολήσει στο πλαίσιο αυτής της εργασίας, είναι τύπου $Fm||C_{max}$, σύμφωνα με την αναπαράσταση της παραγράφου 2.3. Πρόκειται, δηλαδή για ένα πρόβλημα flow-shop, όπου όλες οι εργασίες πρέπει να περάσουν με την ίδια σειρά, από όλες τις μηχανές. Θα διερευνήσουμε διάφορα προβλήματα, με ποικίλους αριθμούς μηχανών m, εξ ου και παραμένει αδιευκρίνιστη η παράμετρος m στον τύπο. Τέλος, εστιάζουμε στο συνολικό χρόνο επεξεργασίας C_{max}, δηλαδή μας ενδιαφέρει η ελαχιστοποίηση του συνολικού χρόνου εντός του οποίου θα ολοκληρωθεί και η τελευταία εργασία στην τελευταία μηχανή. Στη βιβλιογραφία αυτός ο συνολικός χρόνος επεξεργασίας αναφέρεται ως *makespan*. Κάθε μία από τις n εργασίες από το σύνολο εργασιών $J=\{1, 2, \dots, n\}$ με $n>1$, θα περάσει με τη σειρά $1, 2, \dots, m$ από τις m μηχανές. Από τη στιγμή που η εργασία j εισαχθεί στη μηχανή i για επεξεργασία, δεν μπορεί να διακοπεί μέχρι να ολοκληρώσει την επεξεργασία της στη μηχανή αυτή. Ο χρόνος επεξεργασίας p_{ij} της εργασίας j στη μηχανή i είναι εκ των προτέρων γνωστός.

Όταν ολοκληρωθεί η επεξεργασία σε μία μηχανή, η εργασία την απελευθερώνει και η μηχανή μπορεί να δεχθεί την επόμενη εργασία, ενώ η εργασία που ολοκληρώθηκε μπορεί να προχωρήσει στην επεξεργασία της στην επόμενη μηχανή, εφόσον βέβαια είναι διαθέσιμη και δεν απασχολείται με άλλη εργασία (Marinakis et al., 2013). Υπάρχει μία παραλλαγή του προβλήματος flow-shop, το flow-shop με blocking, όπου όταν μία εργασία ολοκληρώσει την επεξεργασία της σε μία μηχανή, δεν την αποδεσμεύει για επόμενη εργασία, εφόσον η επόμενη μηχανή που χρειάζεται η εργασία δεν είναι διαθέσιμη για να τη δεχθεί. Ουσιαστικά μιλάμε για μία κατάσταση όπου δεν υπάρχει δυνατότητα buffering, δηλαδή χώρου αναμονής μεταξύ των εργασιών, κι αυτές πρέπει να βρίσκονται αποκλειστικά εντός κάποιας μηχανής μέχρι την ολοκλήρωση όλων των επεξεργασιών σε αυτές. Είναι εύλογο ότι ο συνολικός χρόνος επεξεργασίας στο

πρόβλημα flow-shop με blocking είναι τουλάχιστον ίσος ή μεγαλύτερος από αυτόν του προβλήματος χωρίς blocking, καθώς οι όποιες καθυστερήσεις στο να εισαχθεί μία εργασία σε επόμενη μηχανή, αυξάνουν το χρόνο παραμονής της στην προηγούμενη, κι άρα μπλοκάρουν τη δυνατότητα η προηγούμενη μηχανή να δεχθεί επόμενη εργασία μέχρι να φύγει η εργασία που ήδη έχουν, έστω κι εάν αυτή παραμένει ανενεργή.

Καθώς όλες οι εργασίες πρέπει να περάσουν από όλες τις μηχανές, μία εφικτή λύση πρέπει να περιλαμβάνει μία διάταξη των n εργασιών του τύπου $\pi=(\pi(1), \pi(2), \dots, \pi(n))$ στο σύνολο J των εργασιών. Το $\pi(1)$ συμβολίζει την εργασία που θα εκτελεστεί 1η, το $\pi(2)$ την εργασία που θα εκτελεστεί 2η κ.ο.κ. μέχρι την εργασία $\pi(n)$ που θα εκτελεστεί nστη. Το σύνολο Π όλων των δυνατών διατάξεων είναι προφανές ότι περιλαμβάνει $n!$ στοιχεία.

Ο μέγιστος χρόνος ολοκλήρωσης C_{max} για κάθε διάταξη π, δίνεται από τη σχέση:

$$C_{max}(\pi) = \max_{1\le t_1\le t_2\le\cdots\le t_{n-1}\le t_n}\left(\sum_{j=1}^{t_1} p_{\pi(j)1} + \sum_{j=t_1}^{t_2} p_{\pi(j)2} + \dots + \sum_{j=t_{n-1}}^{t_n} p_{\pi(j)m}\right) \quad (1)$$

Η βέλτιστη διάταξη π* θα ικανοποιεί τη σχέση:

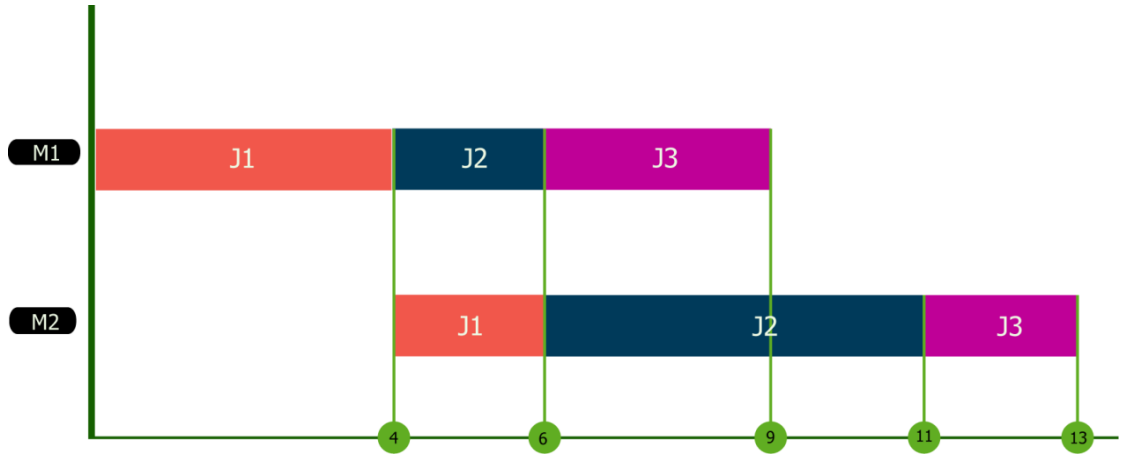
$$C_{max}(\pi^*) = min\, C_{max}(\pi),\ \ \forall \pi \in \Pi \quad (2)$$

Ας δούμε όλα τα παραπάνω σε ένα μικρό παράδειγμα, 3 εργασιών και 2 μηχανών.

	ΕΡΓΑΣΙΕΣ		
ΜΗΧΑΝΕΣ	J1	J2	J3
M1	4	2	3
M2	2	5	2

Πίνακας 1: Μήτρα χρόνου επεξεργασίας

Όπως φαίνεται στο Γράφημα 1, η διάταξη $J1$-$J2$-$J3$ με βάση τους χρόνους του Πίνακα 1, ολοκληρώνεται σε 13 μονάδες χρόνου.



Γράφημα 1: Διάγραμμα Gantt για τη διάταξη J1-J2-J3.

2.5 Προσεγγίσεις δημιουργίας ή βελτίωσης λύσεων

Για την επίλυση ενός προβλήματος όπως το permutation flow-shop, όπου η λύση είναι μία προτεινόμενη σειρά εργασιών του τύπου $\pi=(\pi(1), \pi(2), \dots, \pi(n))$ στο σύνολο J των εργασιών, από το σύνολο Π όλων των δυνατών διατάξεων, υπάρχουν δύο κύριες προσεγγίσεις: Η δημιουργία και η βελτίωση λύσεων (Ancau, 2012).

Στην προσέγγιση της *δημιουργίας λύσης (constructive approach),* ξεκινάμε από μία κενή σειρά εργασιών και προσθέτουμε σε αυτή εργασίες με συγκεκριμένη αλγοριθμική διαδικασία. Τοποθετώντας και την τελευταία εργασία στη σειρά π, έχουμε τη βέλτιστη λύση που μπορεί να δώσει ο συγκεκριμένος αλγόριθμος που χρησιμοποιούμε, και ολοκληρώνεται η διαδικασία του. Σε όλη την πορεία της δημιουργίας της λύσης, μέχρι να φτάσουμε στο στάδιο που τοποθετούμε στη σειρά της και την τελευταία εργασία, δεν έχουμε εφικτή λύση για το πρόβλημα, αφού από αυτή απουσιάζουν μία ή περισσότερες εργασίες.

Στην προσέγγιση της *βελτίωσης λύσης (improvement approach)*, ξεκινάμε με μία εφικτή λύση π από το σύνολο Π, την οποία με συγκεκριμένες διαδικασίες αλλάζουμε μερικώς, με στόχο να οδηγηθούμε σε μία καλύτερη λύση. Σε όλη την πορεία αναζήτησης καλύτερης λύσης, έχουμε πάντα στη διάθεσή μας εφικτές λύσεις. Άρα, εάν τερματίσουμε τη διαδικασία σε οποιοδήποτε σημείο της, μπορούμε σε κάθε περίπτωση να έχουμε την καλύτερη λύση που έχουμε εντοπίσει μέχρι εκείνη τη στιγμή.

Εάν το πρόβλημά μας έχει περιορισμούς, μπορούμε να επιλέξουμε την προσέγγιση να έχουμε μεν μία διάταξη που να περιλαμβάνει όλες τις διαδικασίες, έστω κι εάν αυτή

παραβιάζει κάποιους περιορισμούς και σταδιακά να μεταμορφώνουμε τη διάταξη αυτή ώστε να μην παραβιάζει κανέναν περιορισμό. Φυσικά, είναι δυνατή και η προσέγγιση να έχουμε σε όλη τη διάρκεια αναζήτησης λύσης, αποκλειστικά εφικτές διατάξεις, δηλαδή διατάξεις που δεν παραβιάζουν κανέναν περιορισμό. Η επιλογή, στις περιπτώσεις αυτές, ανήκει στο σχεδιαστή του αλγορίθμου.

2.6 Πολυπλοκότητα και προσπάθειες επίλυσης

Στη βιβλιογραφία υπάρχουν πλήθος προσπάθειες επιτυχούς επίλυσης του προβλήματος για περιορισμένο αριθμό μηχανών (Anderson et al. 1997). Έτσι το 1954 ο Johnson πρότεινε έναν αλγόριθμο για πρόβλημα $F2||C_{max}$, δηλαδή δύο μηχανών, ο οποίος υπό συνθήκες μπορεί να χρησιμοποιηθεί και σε πρόβλημα τριών μηχανών, $F3||C_{max}$. Ο αλγόριθμος αυτός έχει υπολογισμένη πολυπλοκότητα $O(n\ logn)$. Ο αλγόριθμος του Johnson χρησιμοποιήθηκε από τους Lageweg et al. (1978) και Potts (1980) για branch-and-bound αλγορίθμους, οι οποίοι επιλύουν με τον αλγόριθμο του Johnson υποπροβλήματα δύο μηχανών.

Έρευνα σε αλγορίθμους branch-and-bound είχαμε κυρίως μέχρι τη δεκαετία του 1980 (Nowicki et al., 2004). Η απόδοση των αλγορίθμων αυτή δεν είναι ικανοποιητική, καθώς αντιμετωπίζουν δυσκολίες στην επίλυση προβλημάτων 15 εργασιών και 4 μηχανών και, φυσικά, μεγαλυτέρων αυτών. Πλέον, ήδη από την δεκαετία του 1990 είχε γίνει εμφανές ότι οι branch-and-bound προσεγγίσεις είχαν φτάσει στα όριά τους, καθώς σε μονοεπεξεργαστικά συστήματα συχνά αποτύγχαναν να βρουν λύση σε προβλήματα 50 εργασιών και 20 μηχανών, ακόμα και μετά από ημέρες επεξεργασίας (Nowicki et al., 2004). Η παράλληλη επεξεργασία μπορεί να βοηθήσει τέτοιες προσεγγίσεις, αλλά με κόστος. Είναι χαρακτηριστικό το παράδειγμα της εργασίας των Kouki, Jemni και Ladhari (2011), που για την εύρεση της βέλτιστης λύσης σε πρόβλημα με 50 εργασίες και 10 μηχανές, με αλγόριθμο τύπου branch-and-bound, χρειάστηκαν 1 ώρα και 55 λεπτά, σε σύστημα παράλληλης επεξεργασίας με 50 επεξεργαστές. Το κόστος εύρεσης της βέλτιστης λύσης είναι σημαντικό, αν αναλογιστούμε ότι μπορούμε να οδηγηθούμε με άλλους αλγόριθμους στην ίδια λύση, με μόνον έναν επεξεργαστή και σε χρόνο μερικών δευτερολέπτων! Βέβαια, από την άλλη πλευρά, ένας branch-and-bound αλγόριθμος μας εγγυάται το βέλτιστο της λύσης, κάτι που δεν μπορούν να μας εγγυηθούν άλλες προσεγγίσεις τοπικής αναζήτησης.

Γενικά, έχει δειχθεί ότι το πρόβλημα $F3||C_{max}$ είναι ισχυρά NP-hard (Garey et al. (1976)), κάτι που φυσικά ισχύει και για οποιοδήποτε με μεγαλύτερο αριθμό μηχανών. Συνεπώς, ιδιαίτερα για προβλήματα 4 και περισσοτέρων μηχανών, οι ερευνητικές προσπάθειες εστιάζουν στον εντοπισμό μιας καλύτερης λύσης, αφού η με σιγουριά εύρεση της απόλυτα βέλτιστης λύση, είναι δυσχερής. Μία εξαιρετική εργασία που παρουσιάζει το μέγεθος του προβλήματος στην αναζήτηση εντός του χώρου λύσεων ενός flow-shop προβλήματος, είναι αυτή των Nowicki και Smutnicki (2004).

2.7 Ο αλγόριθμος NEH

Το 1982, οι Nawaz, Enscore και Ham πρότειναν ένα ευρετικό αλγόριθμο δημιουργίας λύσης για το πρόβλημα flow-shop. Ο αλγόριθμος αυτός συναντάται συχνά στη βιβλιογραφία ως NEH, από τα αρχικά των ονομάτων των τριών ερευνητών που τον δημιούργησαν.

Τα βήματα του αλγορίθμου είναι τα ακόλουθα:

Βήμα 1: Για κάθε εργασία i υπολογίζουμε το συνολικό χρόνο επεξεργασίας που απαιτεί από όλες τις μηχανές m, δηλαδή το $T_i = \sum_{j=1}^{m} t_{ij}$, όπου t_{ij} ο χρόνος που απαιτεί η εργασία i στη μηχανή j.

Βήμα 2: Ταξινομούμε τις εργασίες με φθίνουσα σειρά ως προς το T_i.

Βήμα 3: Παίρνουμε τις δύο πρώτες εργασίες της λίστας του βήματος 2 και βρίσκουμε τη διάταξη αυτών των δύο εργασιών που δίνει το μικρότερο makespan. Στη συνέχεια του αλγορίθμου δεν αλλάζουμε τη σχετική θέση αυτών των δύο εργασιών. Θέτουμε $i=3$.

Βήμα 4: Επιλέγουμε από τη λίστα του βήματος 2 την εργασία στη θέση i και βρίσκουμε τη θέση της στην προηγούμενη διάταξη, μέσα από τις i δυνατές θέσεις, που δίνει το μικρότερο makespan για το σύνολο των i εργασιών. Δεν αλλάζουμε τη σχετική θέση των υπολοίπων εργασιών.

Βήμα 5: Εάν $i=n$ τότε ο αλγόριθμος τερματίζει. Διαφορετικά θέτουμε $i=i+1$ και πηγαίνουμε ξανά στο βήμα 4.

Ο αλγόριθμος ΝΕΗ δημιουργεί μία συγκεκριμένη λύση για κάθε πρόβλημα, για την οποία δεν υπάρχει καμία βεβαιότητα ότι είναι η βέλτιστη. Όμως μετρήσεις σε διάφορα σετ προβλημάτων έχουν δείξει ότι ο συγκεκριμένος αλγόριθμος δίνει λύσεις «αρκετά κοντά» στις βέλτιστες ή τουλάχιστον σε αυτές που μπορούμε να εντοπίσουμε ως βέλτιστες για μεγάλα προβλήματα. Έτσι, αποτελεί μία εξαιρετική ευρετική διαδικασία για να έχουμε μία καλή λύση σε ένα πρόβλημα flow-shop.

Έχουν προταθεί αρκετές παραλλαγές στον αλγόριθμο ΝΕΗ, οι οποίες αποδίδουν καλύτερα σε κάποια σετ δεδομένων και χειρότερα σε κάποια άλλα (ενδεικτικά αναφέρουμε τις εργασίες των Gao και Chen το 2011, του Taillard το 1990 κ.ά.) Επίσης, έχουν προταθεί αλγόριθμοι δημιουργίας λύσης με διαφορετική προσέγγιση από αυτή του ΝΕΗ, με καλύτερα μεν αποτελέσματα, αλλά απαιτώντας περισσότερο χρόνο (ενδεικτικά να αναφέρουμε την εργασία του Ancau το 2012, με πολυπλοκότητα n^3 έναντι πολυπλοκότητας n^2 του αλγορίθμου ΝΕΗ).

Το αρνητικό των ευρετικών προσεγγίσεων είναι ότι προτείνουν πάντα μία συγκεκριμένη λύση, και ο ίδιος ο αλγόριθμος δεν μπορεί να τη βελτιώσει. Επίσης, η λειτουργία τους είναι απολύτως συνδεδεμένη με το πρόβλημα που λύνουν, καθώς ο ίδιος αλγόριθμος δεν μπορεί να επιλύσει ένα άλλο πρόβλημα συνδυαστικής βελτιστοποίησης. Τέλος, καθώς η λύση χρησιμοποιεί χαρακτηριστικά του προβλήματος, συνήθως αποτελεί ένα καλό τοπικό ελάχιστο, οπότε η διερεύνηση στη γειτονιά της λύσης δεν οδηγεί σε σημαντική βελτίωση (Ancau, 2012).

Από την άλλη, οι ευρετικές προσεγγίσεις μας παρέχουν ένα ικανοποιητικό σημείο εκκίνησης για αναζήτηση μιας καλύτερης λύσης. Αντί, λοιπόν, σε έναν αλγόριθμο βελτίωσης λύσης να ξεκινήσουμε από ένα τυχαίο σημείο, που μπορεί να είναι πολύ μακριά από μία καλή λύση, με τη χρήση ενός ευρετικού αλγορίθμου έχουμε μία πολύ καλή αρχική λύση για να την εξελίξουμε (Nowicki et al., 2004). Έτσι, εξαιρετικοί μεθευρετικοί αλγόριθμοι χρησιμοποιούν αλγόριθμους όπως το ΝΕΗ για αρχική λύση, ξεκινώντας την αναζήτησή τους από ένα πολύ καλό αρχικό σημείο (ενδεικτικά Grabowski et al. (2004), Grabowski et al. (2005), Nowicki et al. (1994)).

2.8 Το σετ προβλημάτων του Taillard

Ολοκληρώνουμε την αναφορά μας σε ζητήματα που αφορούν το πρόβλημα που διαπραγματευόμαστε, με το σετ προβλημάτων του Taillard.

Προκειμένου να μπορούν να συγκριθούν οι αλγόριθμοι που προτείνονται από τους διαφόρους ερευνητές, χρειαζόμαστε ένα κοινό σύνολο προβλημάτων, ώστε κάθε αλγόριθμος να δοκιμάζεται σε αυτά. Με τον τρόπο αυτό μπορούμε να συγκρίνουμε τους αλγορίθμους ως προς την ικανότητά τους να προσεγγίζουν καλές λύσεις, αλλά ως προς το χρόνο που χρειάζονται για να το επιτύχουν.

Με αυτό το σκεπτικό ο Eric Taillard το 1993 παρουσίασε ένα σύνολο 120 προβλημάτων 12 διαφορετικών διαστάσεων (10 προβλήματα ανά διάσταση), ώστε να δοκιμάζονται σε αυτά οι αλγόριθμοι χρονοπρογραμματισμού. Το σύνολο αυτό των προβλημάτων συναντάται στη βιβλιογραφία ως *σετ Taillard*.

Σε κάθε ένα πρόβλημα δίνεται ο αριθμός εργασιών, ο αριθμός μηχανών καθώς και ο χρόνος που απαιτεί κάθε εργασία σε κάθε μηχανή. Οι 12 ομάδες προβλημάτων έχουν διαστάσεις (εργασίες x μηχανές): 20 x 5, 20 x 10, 20 x 20, 50 x 5, 50 x 10, 50 x 20, 100 x 5, 100 x 10, 100 x 20, 200 x 10, 200 x 20, 500 x 20.

Στη συνέχεια της εργασίας θα χρησιμοποιούμε το συμβολισμό nxm (λ.χ. 20x5), αναφερόμενοι σε ένα πρόβλημα n εργασιών που πρέπει να τύχουν επεξεργασία από m μηχανές.

"I made this letter longer than usual

because I lack the time to make it shorter."

Pascal, Provincial Letters XVI

Κεφάλαιο 3:

Ο εξελικτικός αλγόριθμος

Έχοντας παρουσιάσει τα στοιχεία του προβλήματος, στο κεφάλαιο αυτό προχωράμε στην παρουσίαση του αλγορίθμου που αναπτύσσουμε στο πλαίσιο της εργασίας.

Η προσέγγιση που ακολουθούμε ως προς την εύρεση της καλύτερης λύσης, στο μεθευρετικό μέρος του αλγορίθμου, είναι αυτή της βελτίωσης της λύσης και όχι της δημιουργίας της. Άρα, είναι σημαντικό να σημειωθεί ότι σε κάθε φάση του αλγορίθμου, έχουμε στη διάθεσή μας εφικτές λύσεις κι όχι ανέφικτες.

3.1 Αλγόριθμοι μίας λύσης ή πολλαπλών λύσεων

Υπάρχουν αλγοριθμικές ιδέες που βασίζονται σε πλήθος λύσεων, με την έννοια ότι το πλήθος «ενσωματώνει τη νοημοσύνη» για να κινηθεί συνολικά προς μία καλή περιοχή και άρα μία καλή λύση. Για παράδειγμα, οι γενετικοί αλγόριθμοι, βασίζονται σε ένα πλήθος λύσεων, όπου με τη μεταξύ τους διασταύρωση, οδηγούμαστε σε γενιές λύσεων με καλύτερα χαρακτηριστικά, δηλαδή πλησιέστερα σε καλύτερες λύσεις. Αντίστοιχα πλήθη λύσεων χρησιμοποιούν και άλλοι αλγόριθμοι, όπως λ.χ. ο αλγόριθμος αποικίας μυρμηγκιών, αλγόριθμοι βασισμένοι σε χαρακτηριστικά μελισσών, αλγόριθμοι βασισμένοι σε χαρακτηριστικά σμήνους πουλιών και άλλοι.

Από την άλλη πλευρά, υπάρχουν αλγοριθμικές ιδέες που γενικά βασίζονται στην εξέλιξη και βελτίωση μίας και μόνης αρχικής λύσης. Για να είμαστε ακριβέστεροι, η λογική τους είναι τέτοια που εφαρμόζονται κάθε φορά σε μία λύση και δεν λαμβάνουν υπόψη τους την ύπαρξη άλλων λύσεων. Για παράδειγμα, ο αλγόριθμος της προσομοιωμένης ανόπτισης, ασχολείται κάθε φορά με μία λύση, αποφασίζοντας το πώς αυτή θα μετεξελιχθεί. Αντίστοιχα λειτουργεί και ο αλγόριθμος περιορισμένης αναζήτησης, ο αλγόριθμος αποδοχής κατωφλιού και άλλοι.

Αν και τα παραπάνω ισχύουν σε γενικές γραμμές, εντούτοις η «γοητεία» στο χώρο των εξελικτικών αλγορίθμων, είναι ότι τίποτα δεν αποτελεί απαράβατο κανόνα. Το ότι κάτι γίνεται «γενικά» και «συνήθως» δεν σημαίνει σε καμία περίπτωση ότι πρέπει να ακολουθείται πάντα. Κάθε αλγόριθμος μπορεί να ακολουθήσει το δικό του σκεπτικό, τις δικές του προσεγγίσεις και βέβαια να αξιολογηθεί γι' αυτές από τα αποτελέσματά του. Έτσι, μπορούμε να έχουμε υλοποιήσεις όπου «παραδοσιακοί» αλγόριθμοι μιας λύσης να αξιοποιούν πολλές λύσεις και γενικότερα αλγορίθμους που ενσωματώνουν επιμέρους ιδέες από διαφορετικές τεχνικές τοπικής αναζήτησης.

3.2 Η έννοια της γειτονιάς

Στους αλγόριθμους τοπικής αναζήτησης, η έννοια της *γειτονιάς (neighborhood)* είναι σημαντικότατη. Βρισκόμαστε σε κάθε φάση με μία εφικτή λύση –ή περισσότερες όπου επιδιώκεται το ίδιο για κάθε λύση, οπότε είναι το ίδιο ζήτημα- από την οποία θέλουμε να «μετακινηθούμε» σε μία άλλη λύση, ευελπιστώντας ότι σταδιακά θα φτάσουμε σε κάποια καλύτερη λύση από αυτή που έχουμε τώρα. Η έννοια της γειτονιάς προσδιορίζει τη διαδικασία με την οποία από μία λύση π_1 μπορούμε να οδηγηθούμε σε μία άλλη λύση π_2. Από την π_1, εφαρμόζοντας τη διαδικασία μετακίνησης στη γειτονιά, δεν μπορούμε να οδηγηθούμε μόνο στην π_2, αλλά σε πλήθος n άλλων λύσεων. Τότε λέμε ότι οι λύσεις $\{\pi_1, \pi_2, \dots, \pi_n\} \in N$, δηλαδή στο σύνολο λύσεων της γειτονιάς N.

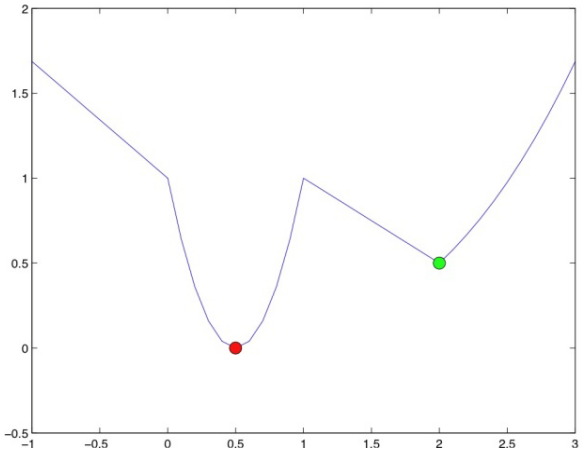
Δεν υπάρχει μία διαδικασία με την οποία διαμορφώνεται η γειτονιά σε κάθε πρόβλημα (Aarts et al., 1997). Αποτελεί κομμάτι του σχεδιασμού του αλγορίθμου, και μάλιστα με σημαντική συμβολή στην ευρωστία του.

Για παράδειγμα, έστω σε ένα πρόβλημα flow-shop η λύση που έχουμε μία δεδομένη στιγμή είναι η $\pi_1=(1, 2, 3, 4, 5)$. Αν έχουμε ορίσει ως διαδικασία για τη δημιουργία γειτονιάς το να τοποθετηθεί κάθε εργασία ως πρώτη, τότε η γειτονιά N της λύσης π_1 αποτελείται από τις λύσεις: $N=\{(1, 2, 3, 4, 5), (2, 1, 3, 4, 5), (3, 1, 2, 4, 5), (4, 1, 2, 3, 5), (5, 1, 2, 3, 4)\}$. Συνεπώς, από την π_1 πηγαίνουμε στις λύσεις της γειτονιάς, τις ελέγχουμε κι αποφασίζουμε το πώς θα κινηθούμε με βάση τα χαρακτηριστικά του αλγορίθμου. Αν λ.χ. η κίνηση που θα επιλέξουμε είναι η $(3, 1, 2, 4, 5)$, τότε με εκκίνηση αυτήν τη λύση θα επισκεφτούμε τις λύσεις που ανήκουν στη γειτονιά της,

που διαμορφώνεται με την ίδια διαδικασία, δηλαδή έχει τη γειτονιά: $N=\{(3, 1, 2, 4, 5), (1, 3, 2, 4, 5), (2, 3, 1, 4, 5), (4, 3, 1, 2, 5), (5, 3, 1, 2, 4)\}$. κ.ο.κ.

3.3 Απεγκλωβισμός από τοπικά ελάχιστα

Επιθυμία μας είναι, κινούμενοι από λύση σε λύση και από γειτονιά σε γειτονιά, να βελτιώνουμε διαρκώς τη λύση, με ιδανικό στόχο το απόλυτο βέλτιστο. Όμως, εάν ακολουθήσουμε διαρκώς καθοδική πορεία, τότε πολύ σύντομα θα εγκλωβιστούμε σε ένα ελάχιστο, κατά πάσα πιθανότητα τοπικό και όχι ολικό. Αν στο παράδειγμα της προηγούμενης παραγράφου η λύση που θα επιλέξουμε από τη γειτονιά N είναι αυτή που βελτιώνει περισσότερο τη λύση που ήδη έχουμε (τον *καλύτερο γείτονα - best neighbor*) κι εν συνεχεία από τη νέα γειτονιά επίσης τον καλύτερο γείτονα, τότε, εάν το πρόβλημά μας είναι φραγμένο, σύντομα θα οδηγηθούμε σε μία γειτονιά που κανένα μέλος της δεν θα μπορεί να μας δώσει καλύτερη λύση. Άρα μοιραία θα περιοριστούμε σε αυτή τη λύση, η οποία όμως μπορεί να είναι μακριά από το ολικό βέλτιστο.



Εικόνα 1:Για να απεγκλωβιστούμε από το τοπικό ελάχιστο που βρίσκεται στα δεξιά, θα πρέπει σταδιακά να αποδεχθούμε και χειρότερες λύσεις, ώστε να μπορέσουμε να φτάσουμε αριστερότερα, στην περιοχή με ένα ακόμα καλύτερο βέλτιστο, ίσως και το ολικό.

Για να είναι αποτελεσματικός ένας αλγόριθμος τοπικής αναζήτησης, θα πρέπει να μπορεί να κινείται ευρύτερα, και να απεγκλωβίζεται από τοπικά ελάχιστα. Ένας τρόπος για να το επιτύχει αυτό είναι εάν αποδέχεται προς διερεύνηση και λύσεις που είναι χειρότερες από αυτές που ήδη έχει. Αυτές οι λύσεις θα του επιτρέψουν να «ανέβει τις ανηφοριές» ενός τοπικού ελάχιστου και να οδηγηθεί σε πιο ελπιδοφόρες περιοχές. Φυσικά, τίποτα δεν αποκλείει ένα από τα ελάχιστα που έχει ήδη επισκεφτεί και «εγκατέλειψε» να είναι το ολικό ή έστω ένα καλύτερο τοπικό ελάχιστο από αυτά που

ήδη έχει εξερευνήσει. Στην Εικόνα 1 βλέπουμε ένα παράδειγμα τοπικού ελαχίστου (δεξί σημείο). Εάν ο αλγόριθμος φτάσει σε αυτό, θα πρέπει να έχει τη δυνατότητα να αποδεχθεί και χειρότερες τιμές, ώστε να κινηθεί προς ένα καλύτερο ελάχιστο (αριστερό σημείο).

3.4 Παρουσίαση του εξελικτικού αλγορίθμου

Στην ενότητα αυτή θα παρουσιάσουμε τα βασικά χαρακτηριστικά του προτεινόμενου αλγορίθμου. Σε κάθε συστατικό του αλγορίθμου θα παρατίθεται το πρόβλημα που έπρεπε να αντιμετωπιστεί, το σκεπτικό που αναπτύχθηκε και η λύση που τελικά επιλέχθηκε.

3.4.1 Αριθμός λύσεων και επανασύνδεση διαδρομής

Σε κάθε επανάληψη αξιολογείται μία λύση. Όμως, ο αλγόριθμος είναι παραμετρικός ως προς το πλήθος των λύσεων που χειρίζεται συνολικά. Αν λ.χ. ο χρήστης επιλέξει να τον εκτελέσει με 10 λύσεις, είναι σαν να κάνει 10 διαφορετικές εκτελέσεις του αλγορίθμου με μία λύση τη φορά.

Αν και ο αριθμός των λύσεων επηρεάζει σχεδόν γραμμικά το χρόνο εκτέλεσης, εντούτοις παρουσιάζει δύο κύρια πλεονεκτήματα. Το πρώτο έχει να κάνει με το χρόνο προετοιμασίας του προβλήματος. Η ανάγνωση του αρχείου, οι αρχικοποιήσεις, η προετοιμασία της αρχικής λύσης και οι άλλες διαδικασίες λαμβάνουν χώρα και χρειάζονται πρακτικά τον ίδιο χρόνο είτε πρόκειται για επίλυση με μία λύση είτε για επίλυση με δεκάδες ή εκατοντάδες λύσεις. Το δεύτερο έχει να κάνει με το «άπλωμα» των λύσεων στο χώρο των εφικτών λύσεων του προβλήματος. Έχοντας 10 λύσεις, που είναι σαν να λύνουμε ουσιαστικά το πρόβλημα 10 φορές, έχουμε αυξημένες πιθανότητες κάποια από αυτές τις λύσεις να βρεθεί κοντά σε ένα καλύτερο ελάχιστο, πιο σύντομα από ότι εάν το διερευνούσαμε μόνο με μία λύση. Αυτό σημαίνει ότι ο χρήστης στο τέλος της κάθε εκτέλεσης θα πάρει ένα καλύτερο αποτέλεσμα, από ότι με μία λύση.

Όμως, ο αλγόριθμος δεν αρκείται απλά στο να εκτελείται ανεξάρτητα με κάθε μία λύση. Στο τέλος κάθε σετ, δηλαδή κάθε προσπάθειας εξέλιξης όλων των λύσεων, επιχειρείται η *επανασύνδεση (path re-linking)* των λύσεων μεταξύ τους. Η επανασύνδεση αποτελεί κατά κανόνα μία συμπληρωματική τεχνική που χρησιμοποιείται σε αρκετούς

αλγορίθμους (για παράδειγμα βλ. Marinakis et al. 2013, Nowicki et al. 2004) και η οποία έχοντας δύο λύσεις, κάνει τις απαραίτητες μεταβολές σε μία από αυτές, ώστε να ταυτιστεί με την άλλη. Το σκεπτικό πίσω από το path re-linking είναι ότι μία καλή λύση μπορεί να εμφανιστεί κατά τη «νοητή διαδρομή» μεταξύ δύο λύσεων.

Για παράδειγμα, εάν έχουμε μία λύση $\pi_1=(1, 2, 3, 4, 5)$ και μία άλλη $\pi_2=(2, 1, 5, 4, 3)$, τότε εφαρμόζοντας διασύνδεση της 2ης λύσης με την πρώτη, θα κάνουμε αλλαγές στην π_2 ώστε να ταυτιστεί με την π_1. Μία πρώτη αλλαγή θα μπορούσε να είναι η αντιμετάθεση των δύο πρώτων στοιχείων, οπότε η π_2 θα γινόταν $(1, 2, 5, 4\ 3)$, οπότε με μία ακόμα αντιμετάθεση του 3ου και του 5ου στοιχείου λαμβάνουμε την π_1.

Ο αλγόριθμος της συνάρτησης του path re-linking είναι:

```
Σάρωσε θέση προς θέση τις δύο λύσεις
      Εάν βρεις σημείο διαφοράς
            Βρες τη θέση του σημείου αυτού στη 2η λύση
            Κάνε ανταλλαγή των θέσεων
            Εάν βρήκες καλύτερη λύση, κράτησέ την
      Τέλος
Τέλος
```

Καθώς, όπως είπαμε, το path re-linking γίνεται στο τέλος κάθε επαναληπτικού σετ, επιλέχθηκε στον προτεινόμενο αλγόριθμο όταν πρόκειται για άρτιο σετ, κάθε λύση επανασυνδέεται με τη βέλτιστη λύση που έχει εντοπιστεί μέχρι εκείνη τη στιγμή από τον αλγόριθμο. όταν πρόκειται για περιττό σετ, κάθε λύση επανασυνδέεται με την επόμενή της στη σειρά, από τη λίστα λύσεων που διερευνούμε. Με τον τρόπο αυτό διερευνούμε πιο συχνά την περιοχή της τρέχουσας βέλτιστης λύσης, προσδοκώντας ότι εκεί είναι πιθανότερο να βρίσκεται μία ακόμα καλύτερη λύση. Όμως, δεν περιοριζόμαστε μόνο εκεί, καθώς μας ενδιαφέρει η διερεύνηση και άλλων περιοχών.

3.4.2 Η έννοια της γειτονιάς

Έχοντας μία λύση, το ενδιαφέρον μας είναι να μετακινηθούμε σε μία άλλη λύση, στο πλαίσιο της γειτονιάς της, σύμφωνα με όσα είδαμε στο κεφάλαιο 2.

Στον προτεινόμενο αλγόριθμο η έννοια της γειτονιάς προσδιορίζεται με τη μετακίνηση μιας τυχαίας εργασίας, από μία θέση της λύσης, και εισαγωγή της σε μία άλλη, επίσης

τυχαία, θέση της λύσης. Δηλαδή εάν η λύση μας είναι η $\pi=(5, 4, 2, 1, 3)$ και επιλέξουμε να μετακινήσουμε τη λύση από τη θέση 5 στη θέση 2, τότε η λύση που θα προκύψει θα είναι $\pi'=(5, 3, 4, 2, 1)$. Πολλές ερευνητικές εργασίες υποστηρίζουν ότι η εισαγωγή μίας εργασίας σε μία άλλη θέση δίνει καλύτερα αποτελέσματα από την αμοιβαία ανταλλαγή θέση δύο εργασιών, οπότε επιλέξαμε κι εμείς την προσέγγιση της εισαγωγής σε άλλη θέση (Nowicki et al. 2004).

Σε κάθε επαναληπτικό σετ, κάθε λύση υπόκειται σε ένα ορισμένο από το χρήστη αριθμό αλλαγών με την παραπάνω διαδικασία. Στόχος είναι να διερευνηθεί η περιοχή της λύσης εκκίνησης (γειτονιά της) και να τοποθετηθούμε σε ένα άλλο σημείο αυτής, το οποίο θα αποτελέσει το σημείο / λύση εκκίνησης για το επόμενο επαναληπτικό σετ. Το κριτήριο αποδοχής της επόμενης λύσης περιγράφεται στην παράγραφο 3.4.3.

3.4.3 Κριτήριο αποδοχής της νέας λύσης

Έχοντας δημιουργήσει τη λύση π', όπως είδαμε στην προηγούμενη παράγραφο, πρέπει να δούμε πώς θα την αξιοποιήσουμε. Εάν η π' είναι καλύτερη από την π, τότε αποδεχόμαστε την π' ως λύση για τη συνέχεια. Εάν επιπρόσθετα η π' είναι καλύτερη κι από την μέχρι στιγμή καλύτερη λύση που έχουμε επισκεφτεί, την π^*, τότε θέτουμε $\pi^*=\pi'$.

Τα παραπάνω, όμως, δημιουργούν μία διαρκώς καθοδική πορεία, η οποία όπως προαναφέραμε σύντομα θα μας οδηγήσει σε κάποιο ελάχιστο (πιθανά τοπικό) και θα μας εγκλωβίσει σε αυτό. Για να αποφύγουμε μία τέτοια κατάσταση, αποδεχόμαστε την π' ακόμα και στην περίπτωση που είναι χειρότερη από την π, αλλά όχι πολύ χειρότερη. Συγκεκριμένα, την αποδεχόμαστε εφόσον η αξία της (εν προκειμένω το makespan της) είναι μεγαλύτερη της αξίας της π, προσαυξημένης κατά τον αριθμό του σετ επανάληψης που βρισκόμαστε. Για παράδειγμα, εάν βρισκόμαστε στο πρώτο από τα 5 σετ επανάληψης, αποδεχόμαστε την π' εάν έχει αξία το πολύ μέχρι 5 μονάδες περισσότερες από την π. Καθώς προχωράει η εκτέλεση του αλγορίθμου και μειώνεται ο αριθμός των σετ επαναλήψεων που απομένουν, ο αλγόριθμος γίνεται όλο και πιο λιγότερος δεκτικός σε αποδοχές χειρότερων λύσεων ($\pi+4, \pi+3, \pi+2, \pi+1$).

Η παραπάνω υλοποίηση βασίζεται στην προσέγγιση του *κατωφλιού αποδοχής (threshold accepting)* σύμφωνα με την οποία αποδεχόμαστε μία χειρότερη λύση, εφόσον αυτή δεν ξεπερνά κάποιο όριο (Aarts et al. 1997, Μαρινάκης et al. 2010). Μάλιστα, το όριο αυτό μειώνεται στην πορεία του αλγορίθμου, με τη λογική ότι αρχικά είμαστε πιο δεκτικοί σε χειρότερες λύσεις για να μετακινηθούμε ταχύτερα και σε άλλες περιοχές του συνόλου λύσεων, ενώ στη συνέχεια, θεωρώντας ότι έχουμε καλύψει μεγάλο μέρος αυτού, είμαστε πιο αυστηροί στην επιλογή μας, αποδεχόμενοι μόνο λύσεις που είναι ελάχιστα χειρότερες από την τρέχουσα. Φυσικά, πάντα δεχόμαστε τις καλύτερες λύσεις.

Η προσέγγιση του κατωφλιού αποδοχής κινείται στη λογική των αλγορίθμων της *προσομοιωμένης ανόπτισης (simulated annealing)* - (Aarts et al. 1997, Μαρινάκης et al. 2010). Εκεί η πιθανότητα αποδοχής μίας χειρότερης λύσης εξαρτάται αφενός από το πόσο χειρότερη είναι η λύση αυτή από την τρέχουσα και αφετέρου από τη χρονική φάση που βρίσκεται η εκτέλεση του αλγορίθμου (όσο πιο αρχική η φάση, τόσο μεγαλύτερη η πιθανότητα αποδοχής της χειρότερης λύσης). Η διαφορά του κατωφλιού αποδοχής σε σχέση με την προσομοιωμένη ανόπτιση, είναι ως προς την πιθανότητα αποδοχής μιας λύσης, η οποία εξαρτάται αποκλειστικά από το πόσο χειρότερη είναι από την τρέχουσα. Εάν είναι πάνω από το κατώφλι, η πιθανότητα αποδοχής της είναι 0, ενώ εάν είναι κάτω από το κατώφλι, η πιθανότητα αποδοχής της είναι 1. Αρχικές δοκιμές έδωσαν καλύτερα αποτελέσματα με το κατώφλι αποδοχής σε σχέση με την προσομοιωμένη ανόπτιση, οπότε τελικά επιλέχθηκε η συγκεκριμένη προσέγγιση για τον προτεινόμενο αλγόριθμο. Όμως, αυτό είναι ένα σημείο που μπορεί να εξεταστεί περαιτέρω, στο πλαίσιο επιπλέον βημάτων βελτιστοποίησης της λειτουργίας του αλγορίθμου.

Σε κάθε λύση η διαδικασία δημιουργίας και ελέγχου αποδοχής λύσεων επαναλαμβάνεται για έναν αριθμό φορών, που προσδιορίζεται από το χρήστη. Η δε διαδικασία αυτή επαναλαμβάνεται σε κάθε σετ επανάληψης, για κάθε λύση. Να σημειωθεί ότι όταν δημιουργείται μία καινούρια λύση και ελέγχεται εάν θα την αποδεχθούμε ή όχι, η επόμενη επανάληψη θα γίνει είτε με την αρχική λύση, εάν επιλέξουμε να μην αποδεχθούμε αυτή που δημιουργήθηκε, είτε με τη λύση που δημιουργήσαμε και αποδεχθήκαμε.

Συνολικά, ο αλγόριθμος για όλα τα παραπάνω έχει ως εξής:

```
Για κάθε λύση π από τη λίστα
      Για ένα αριθμό επαναλήψεων
            Δημιούργησε νέα λύση π' με βάση τη λύση π
                  Αν π' καλύτερη από την π+κατώφλι, τότε π=π'
                  Αν π' καλύτερη από την π*, τότε π*=π'
            Τέλος
      Τέλος
Τέλος
```

Εάν χρησιμοποιηθεί μικρός αριθμός επαναλήψεων με εκκίνηση μία συγκεκριμένη λύση, τότε η περιοχή της λύσης εκκίνησης δεν θα διερευνηθεί επαρκώς. Αν αντιθέτως χρησιμοποιήσουμε υπερβολικά μεγάλο αριθμό επαναλήψεων, τότε θα απομακρυνθούμε από την περιοχή της λύσης εκκίνησης, διερευνώντας μία άλλη περιοχή αρκετά μακρύτερα. Στόχος είναι το να διερευνούμε επαρκώς την περιοχή μιας λύσης, χωρίς όμως να απομακρυνόμαστε ιδιαίτερα από αυτή, τουλάχιστον σε κάθε βήμα. Την ανάγκη για διερεύνηση και άλλων περιοχών του χώρου λύσεων καλύπτουν οι υπόλοιπες λύσεις που είναι διάσπαρτες σε αυτόν καθώς και η διαδικασία του path re-linking που είναι ενσωματωμένη στον προτεινόμενο αλγόριθμο.

3.4.4 «Βελτίωση» λύσης

Με την ολοκλήρωση κάθε σετ επαναλήψεων, έχουμε μία λίστα από λύσεις, σε διάφορα σημεία του χώρου λύσεων. Η καλύτερη λύση που έχουμε επισκεφτεί μέχρι εκείνη τη στιγμή κρατείται πάντα σε ξεχωριστή θέση, μπορεί δε να αποτελεί κάποια στιγμή μέρος της λίστας των λύσεων, μπορεί όμως και όχι (είναι συμπτωματικό, ανάλογα με το εάν η θέση που κάποτε είχε στη λίστα, έχει αντικατασταθεί με άλλη λύση – μη λησμονούμε ότι μπορούμε να δεχθούμε και χειρότερη λύση από αυτή που ήδη έχουμε, άρα μία χειρότερη λύση μπορεί να πάρει θέση μίας καλύτερης στη λίστα των λύσεων).

Στο τέλος κάθε σετ επαναλήψεων, εφαρμόζουμε μία διαδικασία με διπλή επιδίωξη. Η πρώτη είναι να επιτύχουμε μία βελτίωση της κάθε λύσης από τη λίστα, αν κι εφόσον αυτή μπορεί να επιτευχθεί μέσω της διαδικασίας αυτής. Η δεύτερη είναι να δημιουργήσουμε μία «μικρή αναταραχή», ένα μικρό ανακάτεμα στις λύσεις, με το σκεπτικό ότι «έχουμε δει τι αποτέλεσμα δίνουν ως έχουν, ας δοκιμάσουμε και λίγο παραδίπλα».

Η διαδικασία που ακολουθούμε σε κάθε λύση είναι ευρετικού τύπου, δηλαδή αιτιοκρατική, χωρίς κανένα στοχαστικό στοιχείο. Αυτό που κάνουμε είναι κάθε φορά

χωρίζουμε τη λύση σε δύο στοιχεία: Το τελευταίο στοιχείο της λύσης και όλα τα υπόλοιπα. Αυτό που εξετάζουμε είναι εάν η αξία της λύσης βελτιώνεται τοποθετώντας το τελευταίο στοιχείο ως πρώτο. Εάν βελτιώνεται κρατάμε τη βελτιωμένη εκδοχή ως λύση. Φυσικά, αν η κίνηση αυτή βελτιώνει και την καλύτερη λύση που έχουμε μέχρι εκείνη τη στιγμή, αποθηκεύουμε την εκδοχή ως βέλτιστη λύση. Ο αλγόριθμος «βελτίωσης» εφαρμόζεται σε κάθε τμήμα της λύσης. Δηλαδή, αρχικά ελέγχουμε τα δύο πρώτα στοιχεία της λύσης αν τυχών δίνει καλύτερο αποτέλεσμα η αντιμετάθεσή τους. Μετά κρατάμε τα δύο πρώτα στοιχεία κι ελέγχουμε εάν είναι καλύτερα να τοποθετήσουμε το τρίτο στοιχείο πριν από τα δύο πρώτα. Ακολούθως κρατάμε τα τρία πρώτα στοιχεία κι ελέγχουμε εάν είναι καλύτερα να τοποθετήσουμε το τέταρτο στοιχείο πριν από τα τρία πρώτα κ.ο.κ.

Η διαδικασία «βελτίωσης» είναι σύντομη, και οι δοκιμές μας έδειξαν ότι βελτιώνει σε χρήσιμο βαθμό τις λύσεις που εισάγονται σε αυτήν, οπότε αποφασίστηκε η παραμονή της στον προτεινόμενο αλγόριθμο.

Ο αλγόριθμος της διαδικασίας αυτής έχει ως εξής:

```
N=1
Επανέλαβε
      Κράτα σταθερά Ν στοιχεία και βάλε το Ν+1 πριν τα Ν στοιχεία
      Αν η νέα λύση είναι καλύτερη κράτα την
      N=N+1
Έως N=ΑΡΙΘΜΟΣ ΣΤΟΙΧΕΙΩΝ ΛΥΣΗΣ
```

3.4.5 Αρχικές λύσεις

Όπως ήδη αναφέραμε, ο αλγόριθμός μας δεν ακολουθεί την προσέγγιση της δημιουργίας λύσεων αλλά της βελτίωσης λύσεων. Έτσι, σε κάθε φάση υπάρχει μία –ή περισσότερες- εφικτές λύσεις, οι οποίες βήμα – βήμα βελτιώνονται. Και φυσικά προκύπτει το ζήτημα πώς θα δημιουργηθούν οι αρχικές λύσεις, ώστε στην πορεία να βελτιώνονται.

Καταρχήν οποιαδήποτε εφικτή λύση μπορεί να αποτελέσει αφετηρία για τον αλγόριθμό μας. Μία λύση που τοποθετεί σε τυχαίες θέσεις τις εργασίες –προφανώς ακριβώς μία φορά την κάθε εργασία- είναι εφικτή και άρα κατάλληλη για τον αλγόριθμό μας. Βέβαια, η αξία μίας τέτοιας τυχαίας λύσης μπορεί να είναι κοντά ή μακριά μιας πραγματικά καλής τιμής, αυτής του ολικού ή ενός καλού τοπικού ελάχιστου.

Ακολουθώντας την πρακτική πολλών ερευνητών, χρησιμοποιούμε τον αλγόριθμο ΝΕΗ για να δημιουργήσουμε μία αρχική λύση. Η λύση του ΝΕΗ αποτελεί την πρώτη λύση στη λίστα των λύσεων -και μοναδική εάν δεν χρησιμοποιούμε περισσότερες λύσεις- ενώ αφού περάσει από τη διαδικασία «βελτίωσης» της προηγούμενης παραγράφου, καταχωρείται και ως συνολικά καλύτερη λύση, μέχρι εκείνη τη στιγμή.

Εάν, τώρα, ο χρήστης έχει επιλέξει περισσότερες λύσεις, τότε κάθε μία από αυτές δημιουργείται από την προηγούμενή της στη λίστα, με τυχαίες ανατοποθετήσεις εργασιών σε αυτή σε ποσοστό ίσο με το 25% του αριθμού των εργασιών του προβλήματος. Δηλαδή εάν το πρόβλημά μας περιλαμβάνει 20 εργασίες, τότε κάθε νέα λύση προκύπτει με μετακινήσεις 5 εργασιών. Εάν το πρόβλημα περιλαμβάνει 100 εργασίες, γίνονται 25 μετακινήσεις κ.ο.κ.

Ο λόγος που επιλέξαμε ένα ποσοστό μεταβολής και όχι έναν συγκεκριμένο αριθμό μετακινήσεων είναι γιατί θέλουμε κάθε νέα λύση να είναι κατά συγκεκριμένο ποσοστό διαφορετική από την προηγούμενη, και άρα να καλύψουμε ένα ευρύ μέρος του χώρου λύσεων. Εάν επιλέγαμε η νέα λύση να δημιουργείται με λ.χ. 5 ανατοποθετήσεις, αυτές θα ήταν αρκετές για ένα πρόβλημα 20 εργασιών, αλλά ελάχιστες για ένα πρόβλημα 500 εργασιών (δεν θα απομακρυνόταν αρκετά από τη λύση εκκίνησης στην τελευταία περίπτωση).

3.4.6 Ολική λειτουργία του αλγορίθμου

Η διαδικασία που περιγράφηκε παραπάνω, εκτελείται για έναν αριθμό επαναληπτικών σετ. Όπως αναφέραμε, το επαναληπτικό σετ που εκτελείται κάθε φορά επηρεάζει το ύψος του κατωφλιού αποδοχής: Τις πρώτες φορές ο αλγόριθμος είναι δεκτικός σε λύσεις που είναι περισσότερο χειρότερες από τη βέλτιστη, ενώ σταδιακά αυτή η ανοχή μειώνεται. Η βέλτιστη λύση εξελίσσεται όσο βρίσκεται στη λίστα των λύσεων, καθώς μπορεί να βγει από αυτές εάν ο αποδεχθεί μία χειρότερη λύση στο πλαίσιο της κίνησής του στη γειτονιά της. Όμως, συνδυάζεται με τις υπόλοιπες στη διαδικασία του path re-linking, οπότε διαδραματίζει κι αυτή ρόλο στην προσπάθεια ανεύρεσης μίας καλύτερης λύσης από την τρέχουσα.

Στο τέλος της διαδικασίας ο χρήστης λαμβάνει την καλύτερη λύση που εντοπίστηκε από την εκτέλεση του αλγορίθμου.

Συνολικά ο ψευδοκώδικας του αλγορίθμου έχει ως εξής:

```
Δημιούργησε αρχικές λύσεις
Καλύτερη λύση η π*
Επανέλαβε για τον ορισμένο αριθμό επαναληπτικών σετ
      Επανέλαβε για κάθε λύση π από τη λίστα
            Επανέλαβε για ορισμένο αριθμό επαναλήψεων
                  Δημιούργησε νέα λύση π' από την π
                  Αν π' < (π + κατώφλι) τότε π=π'
                  Αν π' < π* τότε π*=π'
            Τέλος
      Τέλος
      Αν άρτιο επαναληπτικό σετ
            κάνε path re-linking κάθε λύσης με την π*
      Διαφορετικά
            κάνε path re-linking κάθε λύσης με την επόμενή της
      Τέλος Αν
      Εφάρμοσε διαδικασία βελτίωσης σε κάθε λύση της λίστας
      Προχώρα στο επόμενο επαναληπτικό σετ
Τέλος
Επέστρεψε την π*
```

3.5 Υλοποίηση του αλγορίθμου σε κώδικα

Στην προηγούμενη ενότητα παρουσιάσαμε τα βασικά συστατικά του προτεινόμενου εξελικτικού αλγορίθμου για το πρόβλημα flow-shop. Στην υλοποίηση του κώδικα κάποιες από τις επιλογές τις οποίες αναφέραμε ενσωματώθηκαν σταθερά στον κώδικα, ενώ σε κάποιες άλλες δόθηκε η δυνατότητα να τις ορίσει ο χρήστης. Η διαδικασία αυτή μας οδηγεί σταδιακά στο έργο της βελτιστοποίησης του αλγορίθμου που θα δούμε στη συνέχεια. Όμως, μέχρι τότε κρίνεται απαραίτητο να δούμε ορισμένα στοιχεία για το πώς ο αλγόριθμος έχει υλοποιηθεί σε κώδικα, καθώς και τις παραμέτρους που επιλέξαμε να μπορεί να καθορίσει ο χρήστης.

3.5.1 Αντικειμενική συνάρτηση flow-shop

Η αντικειμενική συνάρτηση του προβλήματος είναι αυτή που για πάθε λύση π επιστρέφει μία τιμή που αναφέρεται στο makespan αυτής, δηλαδή στο χρόνο C_{max} όπου ολοκληρώνεται και η τελευταία εργασία στην τελευταία μηχανή. Αν και ο τρόπος που υπολογίζεται το makespan δεν αποτελεί μέρος του προτεινόμενου αλγορίθμου, αλλά του προβλήματος, εντούτοις για λόγους πληρότητας θα αναφερθούμε και σε αυτή τη διαδικασία. Θα χρησιμοποιήσουμε ένα απλό παράδειγμα, λίγων εργασιών και μηχανών, ώστε να γίνει κατανοητή η διαδικασία μέσω αυτού του παραδείγματος. Έστω

λοιπόν ότι σε ένα πρόβλημα 5 εργασιών και 5 μηχανών (5x5) μας δίνεται ο Πίνακας 2 των χρόνων για το πρόβλημα.

	ΕΡΓΑΣΙΕΣ					
	α/α	0	1	2	3	4
ΜΗΧΑΝΕΣ	0	54	83	15	71	77
	1	79	3	11	99	56
	2	16	89	49	15	89
	3	66	58	31	68	78
	4	58	56	20	85	53

Πίνακας 2: Μήτρα χρόνου επεξεργασίας

Κάθε στήλη μας δίνει το χρόνο που απαιτεί η συγκεκριμένη εργασία (0-4) σε κάθε μηχανή (0-4). Να σημειωθεί ότι για λόγους διευκόλυνσης στην κωδικοποίηση η πρώτη εργασία (ή μηχανή) έχει τον αριθμό 0, η δεύτερη το 1 κ.ο.κ. Δηλαδή το στοιχείο (2,1) σημαίνει ότι η εργασία 1 (που είναι η δεύτερη) χρειάζεται στη μηχανή 2 (που είναι η τρίτη), 89 χρονικές μονάδες.

Έστω ότι θέλουμε να δούμε το makespan της λύσης $\pi=(0, 1, 2, 3, 4)$. Αυτό σημαίνει ότι πρώτη θα εισαχθεί στη μηχανή 0 η εργασία 0, μετά την ολοκλήρωσή της η εργασία 0 θα πάει στη μηχανή 1 και η μηχανή 0 θα γίνει διαθέσιμη για την εργασία 1 κ.ο.κ. Αυτό σημαίνει ότι η εργασία 0 ολοκληρώνεται από όλες τις μηχανές ακριβώς στο χρόνο που αυτή απαιτεί, δηλαδή μετά από 54 χρονικές μονάδες στο μηχανή 0, μετά από 54+79=133 μονάδες στη μηχανή 1, μετά από 54+79+16=149 μονάδες στη μηχανή 2 κ.ο.κ. Επίσης, η μηχανή 0 είναι διαθέσιμη σε όλες τις εργασίες, αμέσως μόλις τελειώσει η προηγούμενη εργασία. Δηλαδή είναι αμέσως διαθέσιμη στην εργασία 0, διαθέσιμη στην εργασία 1 μετά από 54 μονάδες και άρα ολοκληρώνει την 1 μετά από 54+83=137 μονάδες, διαθέσιμη στην εργασία 2 μετά από 137 μονάδες και την ολοκληρώνει μετά από 137+15=152 μονάδες κ.ο.κ. Με τον τρόπο αυτό συμπληρώνουμε άμεσα τη γραμμή της μηχανής 0 και τη στήλη της εργασίας 0 στη μήτρα υπολογισμού makespan του Πίνακα 3.

Η πρώτη κενή θέση προς συμπλήρωση είναι αυτή της εργασίας 1 στη μηχανή 1. Η μηχανή 1 είναι διαθέσιμη για την εργασία 1 την $133^{η}$ χρονική στιγμή, αλλά η εργασία 1 είναι έτοιμη να μπει στη μηχανή 1 μετά την $137^{η}$ χρονική στιγμή που ολοκληρώνεται η επεξεργασία της στη μηχανή 0. Άρα, η επεξεργασία της εργασίας 1 στη μηχανή 1 ξεκινά την $137^{η}$ χρονική στιγμή και ολοκληρώνεται σε χρόνο 137+3=140. Μετά η εργασία 1

πηγαίνει στη μηχανή 2. Η εργασία 1 είναι έτοιμη να μπει στη μηχανή 2 την 140η χρονική στιγμή, αλλά η μηχανή 2 δεν μπορεί να τη δεχθεί πριν την 149η χρονική στιγμή, μέχρι δηλαδή να τελειώσει με την εργασία 0. Συνεπώς η εργασία 1 μπαίνει στη μηχανή 2 την 149η χρονική στιγμή και ολοκληρώνεται σε χρόνο 149+89=238.

	ΕΡΓΑΣΙΕΣ					
	α/α	0	1	2	3	4
ΜΗΧΑΝΕΣ	0	54	137	152	223	300
	1	133	140	163	322	378
	2	149	238	287	337	467
	3	215	296	327	405	545
	4	273	352	372	490	598

Πίνακας 3: Μήτρα υπολογισμού makespan στο πρόβλημα flow-shop

Από την παραπάνω περιγραφή προκύπτει ότι για να συμπληρώσουμε ένα κελί, πέραν αυτά της πρώτης γραμμής και πρώτης στήλης, μας ενδιαφέρουν το κελί που βρίσκεται στην προηγούμενη γραμμή και το κελί που βρίσκεται στην προηγούμενη στήλη του κελιού που συμπληρώνουμε. Από αυτά τα δύο κελιά παίρνουμε το μεγαλύτερο, που μας λέει τι τελειώνει πιο αργά, η προηγούμενη εργασία σε αυτή τη μηχανή ή η τρέχουσα εργασία στην προηγούμενη μηχανή (καθώς για να ξεκινήσει η επεξεργασία σε μία μηχανή, πρέπει και η μηχανή και η εργασία να είναι διαθέσιμες). Σε αυτή τη χρονική στιγμή, που μας λέει πότε μπορεί να ξεκινήσει η τρέχουσα εργασία στην τρέχουσα μηχανή, προσθέτουμε το χρόνο που χρειάζεται στη μηχανή αυτή, για να τοποθετήσουμε στο κελί το χρόνο που αυτή η εργασία θα ολοκληρώσει σε αυτήν τη μηχανή.

Με αυτήν την απλή, επαναληπτική διαδικασία, συμπληρώνεται ο πίνακας της κάθε λύσης. Το makespan, δηλαδή ο χρόνος ολοκλήρωσης της τελευταίας εργασίας από την τελευταία μηχανή, είναι το τελευταίο κελί του πίνακα, δηλαδή στο παράδειγμά μας το makespan της λύσης π είναι 598 χρονικές μονάδες.

3.5.2 Αντικειμενική συνάρτηση flow-shop με blocking

Ας δούμε, τώρα, πώς υπολογίζεται το makespan στην περίπτωση του flow-shop με blocking. Θυμίζουμε ότι στην περίπτωση του blocking, όταν μία εργασία ολοκληρώσει την επεξεργασία της σε μία μηχανή, εάν δεν είναι διαθέσιμη η επόμενη μηχανή,

παραμένει στη μηχανή που ολοκλήρωσε, καθιστώντας τη μηχανή μη διαθέσιμη για την επόμενη εργασία.

Ακολουθώντας το ίδιο παράδειγμα της προηγούμενης παραγράφου, και εδώ η πρώτη εργασία εκτελείται ανεμπόδιστα, χωρίς καθυστερήσεις μεταξύ των μηχανών στους χρόνους που αυτή χρειάζεται σε κάθε μηχανή. Έτσι, η πρώτη στήλη του πίνακα υπολογισμού (Πίνακας 4) συμπληρώνεται όπως και στο απλό flow-shop.

Η εργασία 1 θα μπει στη μηχανή 0 μόλις ολοκληρώσει από αυτήν η εργασία 0, δηλαδή την 54η χρονική στιγμή και θα ολοκληρώσει από αυτήν σε χρόνο τουλάχιστον ίσο με 54+83=137. Λέμε τουλάχιστον γιατί δεν θα βγει από τη μηχανή 0 εάν η μηχανή 1, η επόμενη δηλαδή μηχανή, δεν απελευθερωθεί από την εργασία 0. Εδώ βλέπουμε ότι η εργασία 1 τελειώνει από τη μηχανή 0 την 137η στιγμή, δηλαδή μετά την 133η χρονική στιγμή που η μηχανή 1 καθίσταται διαθέσιμη από την εργασία 0. Άρα, η εργασία 1 θα ολοκληρώσει από τη μηχανή 0 τη στιγμή 137. Προχωράει στη μηχανή 1 όπου θα ολοκληρώσει σε χρόνο τουλάχιστον 137+3=140. Όμως, την 140η χρονική στιγμή η μηχανή 2 δεν είναι διαθέσιμη, και συνεπώς δεν μπορεί να εισέλθει σε αυτή. Έτσι παραμένει στη μηχανή 1 μέχρι τη χρονική στιγμή 149, που θα ολοκληρωθεί η επεξεργασία της προηγούμενης εργασίας στη μηχανή 2, ώστε η εργασία 1 να εισέλθει σε αυτήν. Άρα, η εργασία 1 θα εγκαταλείψει τη μηχανή 1 (και άρα θα την κάνει διαθέσιμη σε άλλες εργασίες) την 149η χρονική στιγμή. Στη μηχανή 2 θα παραμείνει τουλάχιστον μέχρι τη στιγμή 149+89=238. Καθώς η μηχανή 3 γίνεται διαθέσιμη τη χρονική στιγμή 215, δεν θα καθυστερήσει περισσότερο στη μηχανή 2 και τη στιγμή 238 θα εισέλθει στη μηχανή 3.

	ΕΡΓΑΣΙΕΣ					
	α/α	0	1	2	3	4
ΜΗΧΑΝΕΣ	0	54	137	152	238	337
	1	133	149	238	337	393
	2	149	238	296	352	482
	3	215	296	352	420	560
	4	273	352	372	505	613

Πίνακας 4: Μήτρα υπολογισμού makespan στο πρόβλημα flow-shop με blocking

Από την παραπάνω περιγραφή προκύπτει ότι για τη συμπλήρωση του κάθε κελιού, κοιτάμε το κελί στην προηγούμενη γραμμή του, και προσθέτουμε σε αυτό το χρόνο που απαιτείται στη μηχανή που βρίσκεται. Αν όμως το άθροισμα των χρόνων αυτών

είναι μικρότερο από το χρόνο που τελειώνει η προηγούμενη εργασία στην επόμενη μηχανή (κελί επόμενης γραμμής και προηγούμενης στήλης), τότε στο κελί συμπληρώνεται αυτός του κελιού της επόμενης γραμμής και προηγούμενης στήλης.

Με αυτήν την απλή, επαναληπτική διαδικασία, συμπληρώνεται ο πίνακας της κάθε λύσης. Το makespan, δηλαδή ο χρόνος ολοκλήρωσης της τελευταίας εργασίας από την τελευταία μηχανή, είναι το τελευταίο κελί του πίνακα, δηλαδή στο παράδειγμά μας το makespan της λύσης π για το flow-shop με blocking είναι 613 χρονικές μονάδες.

3.5.3 Soft-coded και hard-coded επιλογές

Κατά την παρουσίαση του αλγορίθμου, έγινε αναφορά σε μία σειρά από μεγέθη ή διαδικασίες. Για παράδειγμα τα επαναληπτικά σετ, ο αριθμός των λύσεων, το ποσοστό «ανακατέματος» κάθε νέας λύσης, το ύψος του κατωφλιού αποδοχής κ.ά. Στον κώδικα που αναπτύχθηκε, κάποια από αυτά τα μεγέθη επιλέχθηκε να είναι hard-coded, δηλαδή σταθερά μέσα στον κώδικα, χωρίς δυνατότητα να τα αλλάξει κάποιος που δεν έχει πρόσβαση στον πηγαίο κώδικα και κάποια άλλα να είναι soft-coded, δηλαδή να έχει δυνατότητα ο χρήστης να τα επιλέξει σε κάθε εκτέλεση του κώδικα.

Ιδανικά θα μπορούσαμε να αφήσουμε όλες ανεξαιρέτως τις επιλογές στο χρήστη. Όμως, αυτή η ελευθερία σε μία σειρά από αλληλοεπηρεαζόμενες παραμέτρους, πολύ πιθανά θα οδηγούσε σε υπερβολική δυσκολία να εντοπίσει ο χρήστης τις κατάλληλες παραμέτρους για κάθε σημείο του αλγορίθμου. Αν από την άλλη πλευρά όλες οι παράμετροι του κώδικα ήταν hard-coded, ο χρήστης δεν θα μπορούσε να δοκιμάσει παραμετροποίηση που θα οδηγούσε σε καλύτερα αποτελέσματα θυσιάζοντας το χρόνο ή το αντίστροφο.

Είναι εμφανές από τη λογική του αλγορίθμου, ότι όσο περισσότερες λύσεις χρησιμοποιήσουμε[4], τόσο αυξάνεται η πιθανότητα να οδηγηθούμε σε καλύτερη λύση. Όμως, περισσότερες επαναλήψεις και περισσότερες λύσεις, σημαίνει και μεγαλύτερο χρόνο εκτέλεσης. Ζητούμενο δεν είναι πάντα η καλύτερη δυνατή λύση, αλλά η καλύτερη λύση εντός κάποιων αποδεκτών ορίων χρόνου, ανάλογα με τη φύση του ζητήματος. Εάν, για παράδειγμα, έχουμε ένα συγκεκριμένο και σταθερό πρόβλημα, η

[4] Η αύξηση του αριθμού των επαναλήψεων, πέρα από ένα σημείο, οδηγεί σε υπερβολική απομάκρυνση από τη λύση εκκίνησης, με αποτέλεσμα να μην διερευνάται επαρκώς η γειτονιά τής συγκεκριμένης λύσης. Άρα, η εύρεση του βέλτιστου αριθμού επαναλήψεων για κάθε μέγεθος προβλήματος, είναι από τα ζητούμενα κατά τη βελτιστοποίηση της παραμετροποίησης του αλγορίθμου.

επίλυση του οποίου θα οδηγήσει σε ρύθμιση μιας γραμμής παραγωγής που θα παραμείνει ίδια για μήνες ή χρόνια, τότε ναι αξίζει να αφιερώσουμε και λεπτά και ώρες ή και ημέρες για να βρούμε την καλύτερη δυνατή λύση. Εάν όμως πρέπει να λύνουμε διαρκώς διαφορετικά προβλήματα, για να προγραμματίσουμε την παραγωγή του επόμενου λεπτού, τότε η επίλυσή τους δεν πρέπει να διαρκεί περισσότερο από μερικά δευτερόλεπτα, αν όχι και λιγότερο!

Στην παράγραφο αυτή θα παρουσιάσουμε όλες τις παραμέτρους που επηρεάζουν τη λειτουργία του αλγορίθμου, είτε πρόκειται για soft-coded είτε για hard-coded στον κώδικά μας. Κι αυτό για να μπορεί κάποιος να υλοποιήσει τον αλγόριθμο σε κώδικα που θα κάνει διαφορετικές επιλογές στις παραμέτρους αυτές.

Soft-coded επιλογές

Ο πίνακας χρόνων του προβλήματος (υποχρεωτική). Είναι πάντα η πρώτη παράμετρος του προβλήματος και ορίζει ένα αρχείο κειμένου που περιέχει τον αριθμό των εργασιών, τον αριθμό των μηχανών και το χρόνο που απαιτεί κάθε εργασία σε κάθε μηχανή. Είναι η τυπική μορφή με την οποία είναι διαθέσιμο το σετ προβλημάτων του Taillard.

Απλό flow-shop ή με blocking (προαιρετική – default 0). Είναι η δεύτερη επιλογή. Εάν 0 τότε επιλύεται το απλό flow-shop πρόβλημα. Εάν 1 τότε επιλύεται ως flow-shop με blocking. Θα μπορούσαν στον ίδιο κώδικα να προστεθούν και άλλες αντικειμενικές συναρτήσεις, ώστε να επιλύονται άλλα προβλήματα (λ.χ. Travelling Salesman Problem) που χρησιμοποιούν ίδιας μορφής λύσεις.

Αριθμός επαναλήψεων (προαιρετική – default 1000). Η τρίτη επιλογή και αναφέρεται στον αριθμό των επαναλήψεων για δημιουργία νέων λύσεων σε κάθε επαναληπτικό σετ. Αναφέρεται σε κάθε λύση και δεν μπορεί να είναι λιγότερες από 1 (το 1 σημαίνει μία μόνο αλλαγή στη λύση εκκίνησης – αν είναι καλύτερη ή έστω εντός κατωφλιού την αποδεχόμαστε, διαφορετικά παραμένουμε με τη λύση εκκίνησης που προφανώς δεν θα μας οδηγήσει κάπου καλύτερα).

Αριθμός επαναληπτικών σετ (προαιρετική – default 5). Η τέταρτη επιλογή αναφέρεται στον αριθμό των επαναληπτικών σετ και δεν μπορεί να είναι λιγότερα από 1.

Αριθμός λύσεων (προαιρετική – default 5). Η πέμπτη επιλογή και αναφέρεται στον αριθμό των λύσεων που θα βελτιώνονται ταυτόχρονα. Δεν μπορεί να είναι λιγότερες από 1.

Hard-coded επιλογές

Κατώφλι αποδοχής αντί προσομοιωμένης ανόπτισης. Ο ίδιος αλγόριθμος θα μπορούσε να χρησιμοποιεί προσομοιωμένη ανόπτιση αντί για κατώφλι αποδοχής. Μάλιστα, με ευκολία μπορεί να αλλαχθεί ο κώδικας ώστε η επιλογή να γίνεται από τον χρήστη. Επιλέχθηκε το κατώφλι αποδοχής, καθώς στις αρχικές δοκιμές μας έδωσε καλύτερα αποτελέσματα από την προσομοιωμένη ανόπτιση.

Ύψος κατωφλιού αποδοχής. Το ύψος του κατωφλιού αποδοχής ορίστηκε στο ύψος του αριθμού των σετ επαναλήψεων που απομένουν. Εάν ο χρήστης έχει ζητήσει 5 σετ επαναλήψεων, το πρώτο σετ θα εκτελεστεί με κατώφλι αποδοχής 5, το δεύτερο με κατώφλι αποδοχής 4 μέχρι το τελευταίο που θα έχει κατώφλι αποδοχής 1. Αρχικές δοκιμές με διπλασιασμό του κατωφλιού αποδοχής, έδωσαν χειρότερα αποτελέσματα από την προσέγγιση που επιλέχθηκε τελικά. Όμως, θα μπορούσε να αφεθεί ως επιλογή στον ίδιο τον χρήστη και βέβαια να αποτελέσει αντικείμενο βελτιστοποίησης των παραμέτρων του αλγορίθμου.

Path re-linking. Στον κώδικα το path re-linking γίνεται στο ένα σετ επαναλήψεων μεταξύ της καλύτερης λύσης και κάθε λύσης από τη λίστα και στο άλλο (εναλλάξ) μεταξύ μίας λύσης από τη λίστα και της επόμενής της. Θα μπορούσε να αφεθεί ως επιλογή στο χρήστη είτε το με τι θα γίνεται το path re-linking είτε και η έντασή του (λ.χ. εντατικό path re-linking μεταξύ όλων των λύσεων μεταξύ τους). Η τελική επιλογή στον κώδικα έγινε διαισθητικά.

«Βελτίωση» λύσης. Θα μπορούσε να δοθεί η επιλογή στο χρήστη να απενεργοποιήσει τη διαδικασία αυτή. Δεν έχουν γίνει σχετικές δοκιμές.

«Ανακάτεμα» λύσης κατά τη δημιουργία της. Το ανακάτεμα γίνεται σε σταθερό ποσοστό 25% επί του μεγέθους της λύσης. Θα μπορούσε να δίνεται επιλογή στον ίδιο το χρήστη να επιλέγει το ποσοστό αυτό.

Με βάση τα παραπάνω, διακρίνουμε πρακτικά 8 επιλογές που θα μπορούσαν να είχαν δοθεί όλες στο χρήστη. Από αυτές επιλέχθηκε, άλλες φορές μετά από κάποιες δοκιμές και άλλες φορές διαισθητικά, οι πέντε να είναι hard-coded χωρίς δυνατότητα αλλαγής από το χρήστη και οι τρεις να είναι soft-coded με δυνατότητα επιλογής από το χρήστη. Οι επιλογές του αρχείου και του τύπου flow-shop (blocking ή μη) δεν συγκαταλέγονται στις επιλογές που ρυθμίζουν την απόδοση του αλγορίθμου, αλλά είναι απαραίτητα στοιχεία.

Το πλήθος των 3 soft-coded επιλογών θα μας απασχολήσει στο επόμενο κεφάλαιο, όπου θα ασχοληθούμε με το πώς θα κάνουμε τις κατάλληλες επιλογές αυτών των παραμέτρων για να επιτύχουμε καλύτερα αποτελέσματα με τον αλγόριθμό μας.

3.5.4 Ο κώδικας

Η υλοποίηση του αλγορίθμου έγινε σε standard C99, με compiler τον GNU έκδοση 3.4.3 κάτω από Windows 7, 32bit. Ο κώδικας εκτελέστηκε σε έναν πυρήνα, σε υπολογιστή Intel core 2 duo 2GHz. Να σημειωθεί ότι πολλές από τις εκτελέσεις, ειδικά αυτές που συμπεριελάμβαναν πληθώρα εκτελέσεων με διαφορετικούς συνδυασμούς ρυθμίσεων και διήρκεσαν ώρες ή και αρκετά 24ωρα, έγιναν με απασχολημένο τον υπολογιστή, οπότε οι χρόνοι είναι δυνητικά αρνητικά επηρεασμένοι.

Οι χρόνοι των εκτελέσεων έχουν συγκριτική αξία, αλλά όχι απόλυτη, καθώς σε άλλο σύστημα θα ήταν διαφορετικοί. Ενδεικτικά να αναφέρουμε ότι σε λειτουργικό σύστημα linux 64bit στον ίδιο υπολογιστή από πλευράς hardware, οι χρόνοι εκτέλεσης ήταν περίπου 30% καλύτεροι.

"You can't make up for lost time.

You can only do better in the future."

Ashley Ormon

Κεφάλαιο 4:

Ακριβής προσαρμογή του αλγορίθμου

Πολλοί ερευνητές έχουν διαπιστώσει τη διαφοροποίηση στη συμπεριφορά των αλγορίθμων τους, ανάλογα με το μέγεθος και τα χαρακτηριστικά των προβλημάτων που καλούνται να επιλύσουν. Είναι χαρακτηριστική η εργασία των Watson et al. (1999), στην οποία παρουσιάζονται οι διαφοροποιήσεις στη συμπεριφορά διαφόρων αλγορίθμων ως προς το μέγεθος και άλλα χαρακτηριστικά των προβλημάτων στα οποία τους εφαρμόζουμε.

Έχοντας υλοποιήσει τον κώδικα του αλγορίθμου, προχωράμε στην εκτέλεση του κώδικα επί του σετ προβλημάτων του Taillard. Και κάνουμε κάποιες σποραδικές διαπιστώσεις.

Για παράδειγμα, στο πρόβλημα 71, των 100 εργασιών και των 10 μηχανών, με 500 επαναλήψεις, 8 λύσεις και 5 επαναληπτικά σετ λαμβάνουμε καλύτερη λύση με makespan 5822 σε 1,213 δευτερόλεπτα, με 1000 επαναλήψεις, 5 λύσεις και 5 επαναληπτικά σετ καλύτερη λύση με makespan 5801 σε 1,3 δευτερόλεπτα, με 4000 επαναλήψεις, 1 λύση και 5 επαναληπτικά σετ καλύτερη λύση με makespan 5800 σε 0,713 δευτερόλεπτα και τέλος με 72000 επαναλήψεις 1 λύση και 5 επαναληπτικά σετ καλύτερη λύση με makespan 5785 στα 9,07 δευτερόλεπτα. Αντίστοιχα και βεβαίως πολύ περισσότερα αποτελέσματα λαμβάνουμε από την εκτέλεση όλων των προβλημάτων από το σύνολο του Taillard.

Και μοιραία προκύπτουν δύο καίρια ερωτήματα:

1. Με ποιους συνδυασμούς παραμέτρων λαμβάνουμε καλύτερες λύσεις σε συντομότερο χρόνο;
2. Οι καλύτεροι συνδυασμοί είναι ίδιοι για όλα τα προβλήματα του Taillard;

Η απάντηση στη δεύτερη ερώτηση δίνεται εύκολα, με μερικές εκτελέσεις προβλημάτων διαφόρων διαστάσεων. Από μερικές εκατοντάδες εκτελέσεις γίνεται εμφανές ότι οι παράμετροι που μπορεί να επιλέξει ο χρήστης επηρεάζουν διαφορετικά τα «μικρά προβλήματα» (αυτά με λίγες εργασίες) και διαφορετικά τα «μεγάλα» των πολλών εργασιών. Αυτό μας οδηγεί στην αίσθηση ότι η απάντηση στο πρώτο ερώτημα δεν θα είναι μονοσήμαντη για όλα τα μεγέθη των προβλημάτων, αλλά θα εξαρτάται από το μέγεθος αυτών.

Οι τρεις επιλογές για παραμετροποίηση που αφέθηκαν στο χρήστη, δημιουργούν ένα ευρύ σύνολο συνδυασμών, στο οποίο πρέπει να εξεταστούν τα διαφόρων διαστάσεων προβλήματα του Taillard. Στόχος είναι να εντοπίσουμε τους συνδυασμούς παραμέτρων εκείνους που μας οδηγούν σε καλύτερες λύσεις, δηλαδή σε λύσεις με το μικρότερο makespan αλλά ταυτόχρονα και στο συντομότερο δυνατό χρόνο. Να επαναλάβουμε ότι όσο περισσότερες επαναλήψεις κάνουμε (μέχρι κάποιο όριο), όσο περισσότερες λύσεις χρησιμοποιήσουμε (χωρίς όριο), τόσο αυξάνεται η πιθανότητα να οδηγηθούμε σε καλύτερη λύση. Όμως, περισσότερες επαναλήψεις και περισσότερες λύσεις, σημαίνει και μεγαλύτερο χρόνο εκτέλεσης. Εάν μπορούμε να οδηγηθούμε σε αντίστοιχο αποτέλεσμα αλλά σε λιγότερο χρόνο, προφανώς αυτό είναι ακόμα καλύτερο και άρα το ζητούμενο! Εάν ο χρόνος εκτέλεσης δεν ήταν ζητούμενο, τότε η αύξηση του αριθμού λύσεων στο άπειρο θα βελτίωνε διαρκώς τα αποτελέσματα, αλλά θα καθιστούσε και άπειρο το χρόνο εκτέλεσης του αλγορίθμου!

Το κεφάλαιο αυτό είναι αφιερωμένο στην κρίσιμη διαδικασία της ακριβούς προσαρμογής (fine tuning) του αλγορίθμου, ώστε να βρούμε τις βέλτιστες παραμέτρους λειτουργίας ανά κατηγορία προβλήματος. Πρόκειται για μία διαδικασία που πραγματικά μπορεί να «εκτοξεύσει» την απόδοση του αλγορίθμου ή να την «καταστρέψει» εντελώς! Είναι ενδεικτικό ότι από τις εκτελέσεις του κώδικα, λ.χ. στο τεστ 112 (500x20) λάβαμε λύσεις με makespan περίπου 26870 σε διαφορετικές εκτελέσεις των 73, 101, 147, 205, 288, 515, 588, 816 και 8525 δευτερολέπτων. Γιατί να περιμένουμε περίπου 2,5 ώρες (που είναι τα 8525 δευτερόλεπτα) για να λάβουμε μία λύση, όταν μπορούμε να λάβουμε λύση του αυτού επιπέδου σε 1-2 λεπτά (που

είναι τα 73 δευτερόλεπτα); Επίσης, σε 5182 δευτερόλεπτα λάβαμε λύση με makespan 26940. Γιατί λοιπόν να περιμένουμε 1,5 ώρες για να λάβουμε μία χειρότερη λύση από αυτήν που μπορούμε να λάβουμε σε μόλις ένα λεπτό εκτέλεσης; Οι διαφορές στις παραπάνω λύσεις, σε επίπεδο makespan και χρόνου, οφείλονται στις διαφορετικές παραμέτρους εκτέλεσης του κώδικα. Συνεπώς, η επιλογή των κατάλληλων παραμέτρων είναι καίρια και άκρως ουσιώδης για τον αλγόριθμο.

Το fine tuning του αλγορίθμου γίνεται σε βάθος, με τη χρήση μοντέλων εξόρυξης δεδομένων (data mining), την οποία και θα περιγράψουμε στη συνέχεια του κεφαλαίου αυτού. Πρόκειται για μία πρωτοποριακή προσέγγιση βελτιστοποίησης παραμέτρων λειτουργίας λογισμικού, η οποία μπορεί να εφαρμοστεί και σε άλλους αλγορίθμους που επιδέχονται παραμετροποίηση.

4.1 Η έννοια της εξόρυξης δεδομένων

Η *εξόρυξη δεδομένων (data mining)*[5] είναι διαδικασία με την οποία αναζητούνται ενδιαφέροντα και χρήσιμα μοτίβο, διατάξεις και σχέσεις σε μεγάλο όγκο δεδομένων. Αξιοποιώντας δεδομένα, συνήθως σε μεγάλες ποσότητες, τα αναλύει και προσπαθεί να εντοπίσει ιδιαίτερα χαρακτηριστικά και να τα μοντελοποιήσει. Τα μοντέλα που δημιουργούνται μπορούν να δώσουν απαντήσεις σε καταστάσεις παρόμοιες με αυτές που έχουμε ως δεδομένες, εφαρμόζοντας τη γνώση από την ανάλυση των πραγματικών δεδομένων που έχουμε στη διάθεσή μας (Witten et al. 2011).

Με τη χρήση των μεθόδων και των τεχνικών του data mining, εξερευνούνται και αναλύονται σετ δεδομένων, με αυτόματο ή –συνηθέστερα- ημιαυτόματο τρόπο, με σκοπό να εντοπίσουμε άγνωστους ή κριμένους κανόνες, συσχετίσεις, τάσεις και άλλες χρήσιμες πληροφορίες που διέπουν τα δεδομένα μας. Η διαδικασία της εξόρυξης δεδομένων οδηγεί σε εξόρυξη πληροφορίας και γνώσης από τα δεδομένα. Πρόκειται για μία διαδικασία περιγραφής και πρόβλεψης. Η περιγραφή αφορά την επεξεργασία υπαρχόντων δεδομένων, ενώ η πρόβλεψη αφορά την εφαρμογή των ιδιοτήτων που προέκυψαν από την επεξεργασία των υπαρχόντων δεδομένων σε νέες καταστάσεις, των οποίων το αποτέλεσμα δεν είναι εκ των προτέρων γνωστό. (Tuffery 2011). Με άλλα λόγια, προσπαθούμε από τα δεδομένα και τις μετρήσεις που ήδη έχουμε, να εκτιμήσουμε το αποτέλεσμα άλλων καταστάσεων, για τις οποίες δεν μπορούμε να

[5] Ο όρος αποδίδεται στα ελληνικά και ως «εξόρυξη γνώσης». Εκτιμούμε ότι ο όρος «εξόρυξη δεδομένων» είναι ακριβέστερος και αντιπροσωπευτικότερος και γι' αυτό τον χρησιμοποιούμε στην παρούσα εργασία, έναντι του «εξόρυξη γνώσης».

έχουμε μετρήσεις, δεν θέλουμε ή δεν είναι ευχερές να έχουμε αναλυτικές μετρήσεις για κάθε περίπτωση.

Καθώς οι διαδικασίες του data mining εμπλέκουν την επεξεργασία δεδομένων, ο τομέας γνωρίζει ανάπτυξη από τα τέλη της δεκαετίας του 1980 και μετά, πολύ δε μεγαλύτερη από τη δεκαετία του 1990, όπου είναι διαθέσιμα όλο και μεγαλύτερα συστήματα βάσεων δεδομένων και ψηφιοποιημένα δεδομένα (North 2012).

Οι εφαρμογές της εξόρυξης δεδομένων εκτείνονται σε ευρύ φάσμα, από μοντέλα για πρόβλεψη καιρικών φαινομένων μέχρι προβλέψεις για χρηματιστηριακούς δείκτες, εφαρμογές διατήρησης πελατών, δημιουργία νέων προϊόντων ή υπηρεσιών και πραγματικά η λίστα μπορεί να πάει πάρα πολύ μακριά.

4.2 Η διαδικασία της εξόρυξης δεδομένων

Στα τέλη της δεκαετίας του 1990, εταιρίες που διέθεταν δεδομένα όπως η Daimler-Benz και η ασφαλιστική εταιρία OHRA, εταιρίες πληροφορικής όπως η NCR Corp., και εταιρίες λογισμικού στατιστικής όπως η SPSS Inc., συνεργάστηκαν για τη δημιουργία ενός προτύπου για την προσέγγιση της διαδικασίας του data mining. Το αποτέλεσμα της δουλειάς τους ήταν το μοντέλο CRISP-DM (CRoss-Industry Standard Process for Data Mining). Αν και το CRISP-DM επηρεάστηκε από τα προϊόντα που διέθεταν οι συμμετέχουσες εταιρίες, εντούτοις ως μοντέλο είναι ανεξάρτητο από οποιοδήποτε συγκεκριμένο εργαλείο (North 2012).

Παρότι στους στόχους της παρούσης εργασίας δεν αποτελεί η θεωρητική τεκμηρίωση και ανάλυση της εξόρυξης δεδομένων, εντούτοις η σύντομη αναφορά των έξι βημάτων του μοντέλου CRISP-DM θα μας επιτρέψει να καταλάβουμε τι εμπεριέχει μία διαδικασία εξόρυξης δεδομένων, ανεξάρτητα από το μέγεθος, το αντικείμενο και το πεδίο εφαρμογής της.

Βήμα 1 - Κατανόηση Αντικειμένου (Business Understanding). Στο βήμα αυτό γίνεται κατανόηση του τι ακριβώς θέλουμε να μάθουμε από τα δεδομένα μας, πώς μπορούμε να βοηθηθούμε από αυτά.

Βήμα 2 – Κατανόηση Δεδομένων (Data Understanding). Στο βήμα αυτό γίνεται κατανόηση της φύσης και του είδους των δεδομένων που έχουμε στη διάθεσή μας.

Μπορεί να έχουμε περισσότερα δεδομένα από όσα μας είναι χρήσιμα (πεδία που δεν προσφέρουν κάτι στην ανάλυσή μας). Μπορεί να μην έχουμε κάποια άλλα δεδομένα που θεωρούμε ότι θα μπορούσαν να μας είναι χρήσιμα. Σε κάθε περίπτωση πρέπει να μελετήσουμε τα δεδομένα που έχουμε, να δούμε εάν και κατά πόσο είναι πλήρη ή έχουν ελλείψεις (εγγραφές που πρέπει να συμπληρωθούν) κι εάν αυτά τα δεδομένα είναι επαρκώς σχετικά με τα ερωτήματα που πρέπει να απαντήσουμε με βάση το πρώτο βήμα.

Βήμα 3 – Προετοιμασία Δεδομένων (Data Preperation). Στο βήμα αυτό γίνεται η προετοιμασία των δεδομένων, ώστε να είναι κατάλληλα προς επεξεργασία. Τα δεδομένα μας μπορεί να προέρχονται από διάφορες πηγές, να έχουν διαφορετικές μορφές ή μορφοποίηση που δεν είναι κατάλληλη για το σύστημα που θα κάνει την επεξεργασία. Η προετοιμασία συμπεριλαμβάνει και την αφαίρεση δεδομένων που δεν είναι χρήσιμα για την επεξεργασία μας, όπως αυτά προέκυψαν από το προηγούμενο βήμα.

Βήμα 4 – Μοντελοποίηση (Modeling). Με απλά λόγια, σε αυτό το βήμα έχουμε μία αναπαράσταση των πραγματικών δεδομένων που έχουμε σε μία μορφή όπου αποτυπώνονται οι συσχετίσεις, οι τάσεις, τα μοτίβο που εντοπίζονται στα δεδομένα. Εδώ γίνεται η παραγωγή του μοντέλου με βάση τα δεδομένα εκπαίδευσης που έχουμε στη διάθεσή μας. Υπάρχουν διάφορα μοντέλα που μπορούμε να χρησιμοποιήσουμε και επιλέγουμε αυτά που είναι κατάλληλα και έχουν νόημα για τα δεδομένα που διαθέτουμε και τις απαντήσεις που ζητάμε.

Βήμα 5 – Αξιολόγηση (Evaluation). Το μοντέλο αναπαριστά τα πραγματικά δεδομένα και κάνει κάποιες εκτιμήσεις για τις συσχετίσεις που εντοπίζει σε αυτά. Αυτό που μας ενδιαφέρει είναι να δούμε πόσο ακριβές είναι σε αυτήν την αναπαράσταση το μοντέλο. Δηλαδή, εάν το μοντέλο μάς πει ότι στην περίπτωση που το πεδίο Α έχει την τιμή 7 και το πεδίο Β έχει την τιμή 12, το πεδίο-στόχος θα πρέπει να έχει την τιμή 23, όπως αυτή έχει προκύψει από τις συσχετίσεις και την επεξεργασία που έχει κάνει. Εάν εμείς στα πραγματικά μας δεδομένα έχουμε μέτρηση με τιμές 7 και 12 στα πεδία Α και Β αντίστοιχα, τι πραγματική τιμή έχουμε στο πεδίο-στόχος; Αν είναι 23, τότε το μοντέλο εκτιμά σωστά στη συγκεκριμένη περίπτωση. Αν δεν είναι 23, το μοντέλο δεν εκτιμά σωστά στη συγκεκριμένη περίπτωση. Όμως ακόμα και στην περίπτωση της «αστοχίας», μας ενδιαφέρει και το μέγεθος αυτής (απόκλιση), γιατί είναι διαφορετικό

για την ακρίβεια του μοντέλου μας να εκτιμήσει για το πεδίο-στόχος την τιμή 22 (απόκλιση 1) και διαφορετικό να εκτιμήσει την τιμή 46 (απόκλιση 23). Συνεπώς, στο βήμα αυτό ελέγχουμε το μοντέλο μας ως προς τα ίδια τα δεδομένα με τα οποία δημιουργήθηκε για να δούμε τα ποσοστά ακρίβειας που επιτυγχάνει, τις αποκλείσεις που έχει, το ύψος του στατιστικού λάθους κ.λπ. Στο βήμα αυτό γίνονται προσαρμογές στις παραμέτρους του μοντέλου (βήμα 4) ώστε να αυξηθεί η ακρίβεια του μοντέλου. Η διαδικασία αυτή είναι κατά κανόνα διαδικασία «δοκιμής και λάθους» (trial and error).

Βήμα 6 – Εφαρμογή (Deployment). Έχοντας οριστικοποιήσει τη μορφή του μοντέλου που έχει τα καλύτερα ποσοστά επιτυχίας στα πραγματικά δεδομένα, μπορούμε να προχωρήσουμε στην εφαρμογή του, δηλαδή στην εισαγωγή στο μοντέλο δεδομένων που δεν γνωρίζουμε την απάντηση και επιθυμούμε να μας την εκτιμήσει το μοντέλο. Συνήθως οι απαντήσεις δίνονται με κάποιο ποσοστό εμπιστοσύνης, οπότε ο αποφασίζων καλείται να αποφασίσει ποιες από τις απαντήσεις ικανοποιούν το επιθυμητό κατώφλι σιγουριάς που αυτός θέτει. Ο αποφασίζων μπορεί να κινηθεί από την πλήρη υιοθέτηση των αποτελεσμάτων του μοντέλου μέχρι την πλήρη απόρριψή τους, ανάλογα με τις εκτιμήσεις που έχει για το πόσο τα αποτελέσματα του μοντέλου ανταποκρίνονται στο είδος της απόφασης που ενδιαφέρεται να λάβει.

Είναι σημαντικό να τονίσουμε ότι ως εργαλείο λήψης απόφασης (decision making took) η διαδικασία της εξόρυξης δεδομένων δεν υποκαθιστά τον αποφασίζοντα αλλά τον βοηθά. Η ερμηνεία των αποτελεσμάτων του μοντέλου συχνά εμπίπτει στον υποκειμενισμό του αποφασίζοντα και στη δική του φιλοσοφία. Όμως, η κατάσταση αυτή δε διαφέρει σε σχέση με αυτό που συμβαίνει και σε άλλα εργαλεία λήψης αποφάσεων, όπως λ.χ. την πολυκριτήρια ανάλυση.

4.3 Η εξόρυξη δεδομένων (βήματα 1 και 2)

Εκτελώντας τον αλγόριθμό μας στο σετ προβλημάτων του Taillard, κάνοντας διάφορες επιλογές στις τρεις παραμέτρους που μπορεί να ορίσει ο χρήστης, λαμβάνουμε αποτελέσματα για το makespan που προκύπτει σε κάθε εκτέλεση καθώς και για το χρόνο που αυτή απαιτεί (όλες οι μετρήσεις μας έγιναν με το πρόβλημα flow-shop χωρίς blocking, αλλά η ίδια διαδικασία μπορεί να εφαρμοστεί και στο πρόβλημα με blocking, δίνοντας τα δικά της αποτελέσματα). Άρα, σε κάθε εκτέλεση έχουμε τις ακόλουθες παραμέτρους:

- Αριθμός εργασιών του προβλήματος (από το μέγεθος του προβλήματος)
- Αριθμός μηχανών του προβλήματος (από το μέγεθος του προβλήματος)
- Αριθμός επαναλήψεων (επιλογή χρήστη)
- Αριθμός επαναληπτικών σετ (επιλογή χρήστη)
- Αριθμός λύσεων (επιλογή χρήστη)

Μετά από κάθε εκτέλεση έχουμε επιπλέον δύο στοιχεία που είναι:

- Το makespan που προέκυψε
- Ο χρόνος που χρειάστηκε η εκτέλεση

Κατ' αρχήν ως προς τον αριθμό των εκτελέσεων, είναι προφανές ότι για κάθε πρόβλημα του Taillard εάν το εκτελέσουμε με λ.χ. 10 διαφορετικές τιμές για τις επαναλήψεις, 5 διαφορετικές τιμές για τα επαναληπτικά σετ 8 διαφορετικές τιμές για τον αριθμό των λύσεων, θα χρειαστούμε 10x5x8=400 εκτελέσεις. Κάποιες από τις εκτελέσεις αυτές θα ολοκληρωθούν σχετικά σύντομα (αυτές που έχουν λίγες επαναλήψεις, λίγα επαναληπτικά σετ και λίγες λύσεις) ενώ αργότερα από όλες θα ολοκληρωθεί η εκτέλεση με το μέγιστο αριθμό επαναλήψεων, επαναληπτικών σετ και λύσεων.

Ιδανικά πρέπει να εκτελεστούν όλα τα προβλήματα του σετ Taillard και να συλλέξουμε από όλα δεδομένα εκτέλεσης. Όμως, με δεδομένο ότι τα προβλήματα αυτά είναι 12 δεκάδες, άρα 12 διαφορετικά μεγέθη από πλευράς εργασιών Χ μηχανές, μπορούμε να θεωρήσουμε ότι αρκεί να εκτελέσουμε κάποιον αριθμό προβλημάτων από κάθε δεκάδα, ώστε να έχουμε αντιπροσωπευτικά στοιχεία από αυτές.

Κάνοντας δοκιμαστικές εκτελέσεις σε προβλήματα διαφόρων μεγεθών, διαπιστώσαμε ότι ο αριθμός των επαναληπτικών σετ άνω του 5 δεν βελτίωνε ουσιωδώς την ποιότητα των λύσεων. Έτσι, αποφασίστηκε να βγάλουμε τη συγκεκριμένη παράμετρο από τη ρύθμιση του αλγορίθμου, και να τη θέσουμε σταθερά, σε όλες τις εκτελέσεις, ίση με 5. Αυτό έγινε για να γίνει διαχειρίσιμος ο αριθμός των εκτελέσεων, καθώς όπως θα δούμε ακόμα κι χωρίς αυτήν την παράμετρο, ο αριθμός των εκτελέσεων και ο συνολικά απαιτούμενος χρόνος γι' αυτές ήταν αρκετά μεγάλος.

Ως προς τον αριθμό των επαναλήψεων, αυτό που θέλουμε να εντοπίσουμε είναι εάν είναι προτιμότερο σε πρόβλημα συγκεκριμένων διαστάσεων να έχει πολλές ή λίγες

επαναλήψεις. Θέτοντας αρχικά κάτω όριο τις 1000 επαναλήψεις και μέγιστο όριο τις 100.000 επαναλήψεις, με σταθερό βήμα 5.000, κάναμε τις εξής διαπιστώσεις. Στα προβλήματα που οι περισσότερες επαναλήψεις ήταν αποτελεσματικότερες, δεν υπήρχαν σημαντικές διαφοροποιήσεις λ.χ. μεταξύ 70.000 και 100.000 επαναλήψεων, αν και στο διάστημα αυτό έγιναν εκατοντάδες εκτελέσεις (με διαφορετικό αριθμό λύσεων). Στα προβλήματα, όμως, που οι λιγότερες επαναλήψεις ήταν αποτελεσματικές, το βήμα των 5.000 μας απέκρυπτε πληροφορία, καθώς σε πολλά προβλήματα 1000, 2000 ή περίπου τόσες επαναλήψεις ήταν επαρκώς αποτελεσματικές.

Μία αντίστοιχη εικόνα είχαμε και για τον αριθμό των λύσεων. Ενώ αρχικά τον προσεγγίσαμε με ελάχιστες λύσεις τις 5, μέγιστες τις 100 και βήμα 5, διαπιστώσαμε ότι σε πολλά προβλήματα η αποτελεσματική επίλυση εμφάνιζε διαφορά μεταξύ 1, 2 ή 3 λύσεων, ενώ στα προβλήματα που ήθελαν περισσότερες λύσεις, οι τιμές 70 ή 90 δεν είχαν ουσιαστική διαφορά.

Με αυτές τις παρατηρήσεις, καταλήξαμε σε εκτελέσεις με αριθμό επαναλήψεων και λύσεων σύμφωνα με την ακολουθία Fibonacci. Έτσι, για τον αριθμό των επαναλήψεων η ακολουθία έχει ως πρώτο όρο το 500 και ως δεύτερο το 1000, ενώ εκτείνεται μέχρι τις 116.500 επαναλήψεις δηλαδή έχει 12 στοιχεία. Αντίστοιχα για τον αριθμό των λύσεων η ακολουθία έχει ως πρώτο όρο το 1 και ως δεύτερο το 2, εκτείνεται δε μέχρι το 144, έχοντας δηλαδή 11 στοιχεία.

Συνοψίζοντας τα παραπάνω, η παραγωγή των δεδομένων γίνεται με 132 εκτελέσεις κάθε προβλήματος, όπου η συντομότερη εκτελείται με 500 επαναλήψεις και 1 λύση και η πλέον χρονοβόρα με 116.500 επαναλήψεις και 144 λύσεις. Όπως προαναφέραμε όλες οι εκτελέσεις γίνονται με 5 επαναληπτικά σετ[6].

Ολοκληρώνοντας τις εκτελέσεις, έχουμε χιλιάδες εγγραφές από μετρήσεις, όπου η κάθε μία περιλαμβάνει τα ακόλουθα έξι πεδία:

[6] Ενδεικτικά στα μικρά προβλήματα (20x5) η συντομότερη εκτέλεση χρειάστηκε 0,009 δευτερόλεπτα ενώ η πλέον χρονοβόρα 263,854 δευτερόλεπτα, ενώ συνολικά και οι 132 εκτελέσεις του προβλήματος 1768 δευτερόλεπτα, δηλαδή 29,5 λεπτά της ώρας. Στα μεγάλα προβλήματα (500x20), η συντομότερη εκτέλεση χρειάστηκε 26 δευτερόλεπτα, η πλέον χρονοβόρα 19.420 δευτερόλεπτα (δηλαδή 5,4 ώρες), ενώ συνολικά οι 132 εκτελέσεις του προβλήματος 155.729 δευτερόλεπτα (δηλαδή 43,26 ώρες).

- Αριθμός εργασιών του προβλήματος (από το μέγεθος του προβλήματος)
- Αριθμός μηχανών του προβλήματος (από το μέγεθος του προβλήματος)
- Αριθμός επαναλήψεων (επιλογή χρήστη)
- Αριθμός λύσεων (επιλογή χρήστη)
- Το makespan που προέκυψε
- Ο χρόνος που χρειάστηκε η εκτέλεση

Στο σημείο αυτό έχουμε ολοκληρώσει τα δύο πρώτα βήματα του CRISP-DM, καθώς γνωρίζουμε τι θέλουμε (τις παραμέτρους που είναι πιο αποτελεσματικές σε κάθε μέγεθος προβλήματος) και έχουμε κατανοήσει και παράξει τα πρωτογενή δεδομένα.

4.4 Προετοιμασία των δεδομένων (βήμα 3)

Από τα πρωτογενή δεδομένα, όπως παρήχθησαν από τις εκτελέσεις, πρέπει να διαμορφώσουμε δεδομένα κατάλληλα για το μοντέλο του data mining.

Ας ξεκινήσουμε με το makespan κάθε εκτέλεσης. Από τη βιβλιογραφία γνωρίζουμε το βέλτιστο makespan κάθε προβλήματος, τουλάχιστον αυτό που έχει εντοπιστεί μέχρι στιγμής ως βέλτιστο (Best Known Solution - BKS), διότι τίποτα δεν αποκλείει να βρεθεί στο μέλλον κάποιο καλύτερο (μικρότερο). Εάν σε ένα πρόβλημα το BKS είναι C^*_{max} και μία εκτέλεση έδωσε C_{max}, τότε η απόκλιση $\Delta = C_{max} - C^*_{max}$ είναι ένα μέτρο για το πόσο καλή είναι αυτή η λύση που προέκυψε από τον αλγόριθμό μας. Το μέτρο της διαφοράς δεν είναι απόλυτο, αλλά σχετικό. Κι αυτό γιατί σε κάποια προβλήματα του Taillard η χειρότερη λύση που λαμβάνουμε από εκτέλεση απέχει λ.χ. 200 από τη BKS, ενώ σε άλλα απέχει μόλις 5 ή 10. Άρα, δεν μπορούμε να πούμε ότι το 10 απόσταση από τη BKS είναι καλό ή κακό αποτέλεσμα.

Αξιολογώντας σχετικά τα αποτελέσματα, σε κάθε πρόβλημα του Taillard δημιουργούμε τρεις ομάδες εκτελέσεων, με όσο το δυνατό ίσο αριθμό εκτελέσεων, όπου στην $1^η$ ομάδα βρίσκεται το 1/3 των εκτελέσεων με τη μικρότερη διαφορά του makespan σε σχέση με τη BKS, στην $3^η$ ομάδα το 1/3 των εκτελέσεων με τη χειρότερη διαφορά από τη BKS και στη $2^η$ ομάδα το 1/3 των εκτελέσεων με τις ενδιάμεσες τιμές της διαφοράς. Η πρώτη ομάδα έχει, δηλαδή τις πολύ καλές λύσεις (Very Good – VG), η δεύτερη τις καλές λύσεις (Good – G) και η τρίτη τις κακές (Bad – BAD).

Προχωράμε τώρα στο χρόνο εκτέλεσης. Κι εδώ τα αποτελέσματα μπορούν να αξιολογηθούν σχετικά, καθώς λ.χ. τα 5 δευτερόλεπτα σε ένα πρόβλημα είναι αργός χρόνος, ενώ σε άλλο εξαιρετικά γρήγορος. Άρα και πάλι περιοριζόμαστε στο να εξετάσουμε κάθε πρόβλημα του Taillard ξεχωριστά. Στην περίπτωση του χρόνου δεν έχουμε απόλυτα καλύτερη τιμή από τη βιβλιογραφία για να συγκρίνουμε. Οπότε το τι είναι γρήγορο και το τι αργό θα πρέπει να το δούμε συγκριτικά με τα αποτελέσματα των διαφόρων εκτελέσεων στο ίδιο πρόβλημα. Στην περίπτωση του χρόνου, και καθώς οι συνδυασμοί των παραμέτρων δεν είναι γραμμικοί, αλλά ακολουθούν την ακολουθία Fibonacci, παρατηρούνται περισσότερες λύσεις με μικρούς χρόνους, ενώ κάποιες, αυτές με τις πολλές επαναλήψεις και τις πολλές λύσεις, να εμφανίζουν χρόνους σε κάποιες περιπτώσεις εκατοντάδων, χιλιάδων ή και δεκάδων χιλιάδων δευτερολέπτων. Εδώ, λοιπόν, επειδή αφενός η διαφορά της ταχύτερης εκτέλεσης με την βραδύτερη είναι μεγάλη, αφετέρου δε η πλειονότητα των εκτελέσεων είναι πλησίον της ταχύτερης εκτέλεσης, χρησιμοποιούμε διαφορετικό κριτήριο για την ομαδοποίηση. Πάντως κι εδώ ο στόχος είναι να δημιουργήσουμε τρεις όσο το δυνατό ισάριθμες ομάδες, με βάση το χρόνο εκτέλεσης. Στα περισσότερα προβλήματα, η ταχύτερη ομάδα (Very Fast – VF) προσδιορίζεται εάν η εκτέλεση είναι ταχύτερη από την ταχύτερη εκτέλεση συν το 1/100 της διαφοράς βραδύτερης με ταχύτερης εκτέλεσης. Η βραδύτερη ομάδα (SIow – SLOW) προσδιορίζεται εάν έχει χρόνο μεγαλύτερο από αυτόν της ταχύτερης εκτέλεσης συν το 1/10 της διαφοράς βραδύτερης με ταχύτερη εκτέλεση και τα υπόλοιπα αποτελούν τη γρήγορη ομάδα (Fast – F).

Άρα, από κάθε εκτέλεση, πέραν των υπολοίπων χαρακτηριστικών που είδαμε στην προηγούμενη παράγραφο, έχουμε επιπλέον δημιουργήσει τους χαρακτηρισμούς VG ή G ή BAD που αναφέρονται στην προσέγγιση της BKS και τους χαρακτηρισμούς VF ή F ή SLOW που αναφέρονται στο χρόνο εκτέλεσης. Συνδέοντας τους παραπάνω χαρακτηρισμούς σε ένα πεδίο, δημιουργούνται σε κάθε λύση όλοι ή κάποιοι από τους εννέα δυνατούς συνδυασμούς VGVF, VGF, VGSLOW, GVF, GF, GSLOW, BADVF, BADF και BADSLOW.

Είναι προφανές ότι το βασικό μας ενδιαφέρον, οι ποιοτικότερες λύσεις, είναι αυτές με το χαρακτηρισμό VGVF (Very Good Very Fast - πολύ καλές και πολύ γρήγορες) ή έστω με τον χαρακτηρισμό VGF (πολύ καλές και γρήγορες). Επίσης προφανώς θέλουμε να αποφύγουμε τις λύσεις BADSLOW (κακές και αργές) ακόμα και τις λύσεις BADF (κακές

και γρήγορες) ή GSLOW (καλές και αργές), καθώς κυριαρχούνται τόσο σε makespan όσο και σε χρόνο από τις λύσεις VGVF ή VGF.

Να σημειώσουμε εδώ ότι εάν σε κάποια προβλήματα δεν είχαμε έναν αριθμό έστω και 10 λύσεων με χαρακτηριστικό VGVF, παρεμβαίναμε στην παραμετροποίηση του χρόνου (καθώς το κριτήριο του makespan ήταν πιο συγκεκριμένο ως προς την κατηγοριοποίηση), αυξάνοντας τις λύσεις με χαρακτηρισμό VF, ώστε να προκύψει ένας αριθμός λύσεων με χαρακτηρισμό VGVF. Η ύπαρξη τέτοιων λύσεων είναι ουσιώδης για την εκπαίδευση του μοντέλου μας και γι' αυτό δεν πρέπει να απουσιάζουν τέτοιες λύσεις.

Η τελευταία ενέργεια που κάνουμε στα δεδομένα είναι να αφαιρέσουμε αλληλουχίες εκτελέσεων που παρουσιάζουν ασυνέπεια στην ποιότητα της λύσης (makespan σε σχέση με τη BKS) εξαιτίας της στοχαστικότητας του αλγορίθμου. Σε ένα σετ, μπορεί να έχουμε σε 500 επαναλήψεις και 1 λύση, αποτέλεσμα που να χαρακτηρίζεται VGVF. Αμέσως μετά μπορεί να δούμε με 500 επαναλήψεις και 2 λύσεις αποτέλεσμα BADVF και μετά με 500 επαναλήψεις και 4 λύσεις, αποτέλεσμα GF. Ως προς το χρόνο, τα πράγματα είναι αναμενόμενα, και καθώς αυξάνουν οι επαναλήψεις/λύσεις, αυξάνεται και ο χρόνος (πηγαίνουμε ομαλά από VF σε F και μετά τα SLOW λύσεις). Όμως, ως προς το makespan μπορεί να συναντήσουμε το φαινόμενο όπου εκτέλεση με λιγότερες λύσεις να δώσει καλύτερο makespan από μία εκτέλεση με περισσότερες λύσεις. Αυτό οφείλεται στο γεγονός ότι ο αλγόριθμος περιέχει στοχαστικά στοιχεία, και δυνητικά μπορεί κάποιος με μία λύση να πέσει κατ' ευθείαν στο απόλυτο ελάχιστο. Όμως, η ύπαρξη τέτοιων ασυνεπειών (ας πούμε VG λύσεις με λιγότερους πόρους και BAD με περισσότερους) μπερδεύει την εκπαίδευση του μοντέλου μας. Έτσι, κάναμε επιλεκτική αφαίρεση εγγραφών ώστε να διατηρείται η πορεία BAD->G->VG στις μετρήσεις, ανάλογα με την αύξηση των πόρων, δηλαδή του αριθμού των λύσεων.

7	20	5	500	13	BADVF
8	20	5	500	21	BADVF
9	20	5	500	34	VGVF
10	20	5	500	55	BADVF
11	20	5	500	89	VGVF
12	20	5	500	144	VGVF
13	20	5	1000	1	BADVF
14	20	5	1000	2	BADVF
15	20	5	1000	3	BADVF
16	20	5	1000	5	BADVF
17	20	5	1000	8	BADVF
18	20	5	1000	13	GVF
19	20	5	1000	21	BADVF
20	20	5	1000	34	VGVF
21	20	5	1000	55	VGVF

Εικόνα 2: Παράδειγμα ασυνεπειών στα δεδομένα στις γραμμές 9 και 19.

Στην Εικόνα 2 παρουσιάζονται δύο τέτοια παραδείγματα ασυνεπειών. Στη γραμμή 9 έχουμε μία λύση VG με 34 λύσεις και 500 επαναλήψεις, ενώ στη γραμμή 10 έχουμε BAD λύση αν και αυξήσαμε τις λύσεις στις 55. Αφαιρώντας μία από τις δύο γραμμές, έχουμε ομαλή συνέχεια στην ποιότητα των λύσεων για να εκπαιδευτεί καλύτερα το μοντέλο μας. Το ίδιο συμβαίνει και στη γραμμή 19. Ενώ με 1000 επαναλήψεις και 13 λύσεις είχαμε επιτύχει G λύση, εντούτοις αυξάνοντας τις λύσεις σε 21, η λύση χειροτερεύει. Και πάλι η διαγραφή μίας εκ των δύο γραμμών, βοηθάει στην ομαλότητα των μετρήσεων για καλύτερη εκπαίδευση του μοντέλου μας. Η αφαίρεση έγινε με κριτήριο να αφαιρεθούν όσο το δυνατόν λιγότερες εγγραφές, καθώς σε κάποιες ομάδες μετρήσεων χρειάστηκαν περισσότερες της μίας διαγραφές.

Να σημειώσουμε στο σημείο αυτό ότι οι παραπάνω διαγραφές των μετρήσεων, δεν αποτελεί μη επιτρεπτή παρέμβαση στα δεδομένα. Ο λόγος είναι γιατί είχαμε μία μόνο εκτέλεση σε κάθε συνδυασμό, ενώ η στοχαστική φύση του αλγορίθμου θα επέβαλε πλήθος εκτελέσεων ανά συνδυασμό, ώστε να χρησιμοποιηθεί ο μέσος όρος. Όμως, καθώς ήδη έγιναν 8469 μοναδικές εκτελέσεις, η εκτέλεση λ.χ. 5 ή 10 φορές με κάθε συνδυασμό θα απαιτούσε εντυπωσιακά υψηλό χρόνο για να αφιερωθεί μόνο στις μετρήσεις[7]. Έτσι, η παρέμβαση αυτή στα δεδομένα, ουσιαστικά προσφέρει την ομαλοποίηση που θα προσέφερε ο πολύ μεγαλύτερος αριθμός εκτελέσεων.

Η τοποθέτηση σε ένα φύλλο excel όλων των εγγραφών που περιλαμβάνουν

- Αριθμός εργασιών του προβλήματος (από το μέγεθος του προβλήματος)
- Αριθμός μηχανών του προβλήματος (από το μέγεθος του προβλήματος)
- Αριθμός επαναλήψεων (επιλογή χρήστη)
- Αριθμός λύσεων (επιλογή χρήστη)
- Τον χαρακτηρισμό της λύσης (εκ των 9 δυνατών)

ολοκληρώνει το βήμα 3 της προετοιμασίας των δεδομένων.

[7] Να σημειωθεί ότι οι 8470 μετρήσεις χρειάστηκαν 1.026.813 δευτερόλεπτα καθαρής επεξεργασίας, ή 285,5 ώρες. Για 10 εκτελέσεις ανά συνδυασμό, θα χρειαζόμασταν 120 ημέρες επεξεργασίας ενώ για 5 εκτελέσεις, 60 ημέρες επεξεργασίας.

4.5 Το λογισμικό RapidMiner

Η περαιτέρω επεξεργασία για την εξόρυξη γνώσης από τα δεδομένα μας, γίνεται με το λογισμικό RapidMiner, για το οποίο θα αναφέρουμε κάποια στοιχεία στην παράγραφο αυτή[8].

Η ανάπτυξη του RapidMiner ξεκίνησε το 2001 στη μονάδα Τεχνητής Νοημοσύνης του Πολυτεχνείου του Dortmund (Dortmund University of Technology) στη Γερμανία. Οι Ralf Klinkenberd, Ingo Mierswa και Simon Fischer ξεκίνησαν την ανάπτυξη ενός ισχυρού και ευέλικτου περιβάλλοντος για data mining, κάτω από την ονομασία YALE. Το αρχικό open-source προϊόν σύντομα έγινε ιδιαίτερα δημοφιλές και το 2006 οι Klinkenberg και Mierswa δημιούργησαν μία εταιρία για την υποστήριξη του προϊόντος τους. Το λογισμικό αναπτύχθηκε εκ νέου και επανακυκλοφόρησε ως RapidMiner, με μεγαλύτερες δυνατότητες στο χειρισμό μεγάλο όγκου δεδομένων, ώστε να ικανοποιεί τις ανάγκες μεγάλων οργανισμών, συνέχισε δε την εξέλιξή του. Μέχρι το 2010 η πλατφόρμα είχε περισσότερες από 500.000 εγκαταστάσεις σε περισσότερες από 50 χώρες. Το 2012 διέθετε περισσότερους από 1.000 εκπαιδευμένους ειδικούς και αναπτυσσόταν με ρυθμό 10.000 νέες εγκαταστάσεις το μήνα.

Έχει λάβει διακρίσεις και θετικά σχόλια από εταιρίες όπως η Gartner Research, ενώ χρησιμοποιείται από εταιρίες όπως οι PayPal, Deloitte, ebay, CISCO, Miele, Volkswagen, Hitachi κ.ά. Πλέον διαθέτει εκδόσεις client (RapidMiner Studio), server και cloud. Η εταιρία εδρεύει στο Cambridge στην Πολιτεία της Μασαχουσέτης των ΗΠΑ, καθώς στο διάστημα της λειτουργίας της έχει λάβει χρηματοδότηση με ιδιωτικά κεφάλαια για την ανάπτυξή της.

Το RapidMiner έχει διαθέσιμους στο χρήστη εκατοντάδες τελεστές, κάθε ένας εκ των οποίων εκτελεί συγκεκριμένη λειτουργία στη διαμόρφωση ενός μοντέλου. Κάποιοι τελεστές εκτελούν λειτουργίες όπως η ανάγνωση και η εξαγωγή δεδομένων, ο καθορισμός ρόλων στα δεδομένα και γενικά φροντίζουν για την εισαγωγή και εξαγωγή στοιχείων. Άλλοι τελεστές υλοποιούν την επεξεργασία, το κύριο μέρος του μοντέλου, ενώ υπάρχουν και τελεστές που αξιολογούν τη λειτουργία του μοντέλου, δηλαδή ελέγχουν το πώς συμπεριφέρεται το μοντέλο αν εφαρμοστεί πάνω στα πραγματικά δεδομένα που έχουμε στη διάθεσή μας και γνωρίζουμε τα αποτελέσματά τους. Έτσι,

[8] Τα στοιχεία προέρχονται από την ιστοσελίδα του RapidMiner και συγκεκριμένα από τη διεύθυνση: https://rapidminer.com/about-us/press-kit/

δεν περιοριζόμαστε στη δημιουργία απλά και μόνο ενός μοντέλου, άγνωστης αποτελεσματικότητας, αλλά σε ένα μοντέλο που έχει δοκιμαστεί και ρυθμιστεί, ώστε να αποδίδει το δυνατόν καλύτερα στο σετ δεδομένων που έχουμε από την πραγματικότητα και άρα έχει τις καλύτερες δυνατές προδιαγραφές για να αποδώσει σωστά σε δεδομένα με μη γνωστά αποτελέσματα.

Σε κάθε τελεστή υπάρχει δυνατότητα παραμετροποίησής του, ανάλογα με το είδος του και τις ρυθμίσεις που μπορεί να δεχθεί. Για παράδειγμα, ο τελεστής δημιουργίας νευρωνικού δικτύου μπορεί να παραμετροποιηθεί ως προς του κύκλους εκπαίδευσης (training cycles), το ρυθμό εκμάθησης (learning rate), το momentum κ.ά. παραμέτρους. Σημαντικό στοιχείο είναι ότι το ίδιο το RapidMiner περιλαμβάνει και τελεστή βελτιστοποίησης των παραμέτρων των υπολοίπων τελεστών. Αυτό σημαίνει ότι μπορούμε να δημιουργούμε και να αξιολογούμε ξανά και ξανά το ίδιο μοντέλο με διαφορετική παραμετροποίηση στον τελεστή του μοντέλου (λ.χ. στο νευρωνικό δίκτυο), ώστε να βρούμε αυτήν που δίνει τα καλύτερα αποτελέσματα. Η βελτιστοποίηση γίνεται εκτελώντας με διαφόρους συνδυασμούς παραμέτρων, με μικρά βήματα αλλαγών, ώστε να λάβουμε το συνδυασμό που δίνει τα καλύτερα αποτελέσματα.

Έχοντας παρουσιάσει τα βασικά χαρακτηριστικά του λογισμικού RapidMiner, συνεχίζουμε με τη μοντελοποίηση του data mining, χρησιμοποιώντας τα δεδομένα μας. Να σημειώσουμε ότι για τις ανάγκες της παρούσης εργασίας, χρησιμοποιήθηκε η έκδοση RapidMiner Studio 6.4.0 σε Win32bit, με τη ακαδημαϊκή άδεια χρήσης λογισμικού που μας παραχωρήθηκε από την κατασκευάστρια εταιρία.

4.6 Μοντελοποίηση (βήμα 4)

Στο βήμα αυτό θα δούμε στην πράξη όσα περιγράψαμε στις προηγούμενες παραγράφους και αφορούν στη δημιουργία και έλεγχο του μοντέλου εξόρυξης δεδομένων.

Ξεκινάμε με τα δεδομένα από τις εκτελέσεις μας που βρίσκονται σε ένα φύλλο Excel. Έγιναν 8469 εκτελέσεις του αλγορίθμου, με διαφορετικούς συνδυασμούς αριθμού επαναλήψεων και αριθμού λύσεων, σε όλη τη γκάμα των προβλημάτων του Taillard. Από αυτές, μετά τη διαδικασία της ομαλοποίησης που προηγήθηκε στο βήμα 3, έμειναν

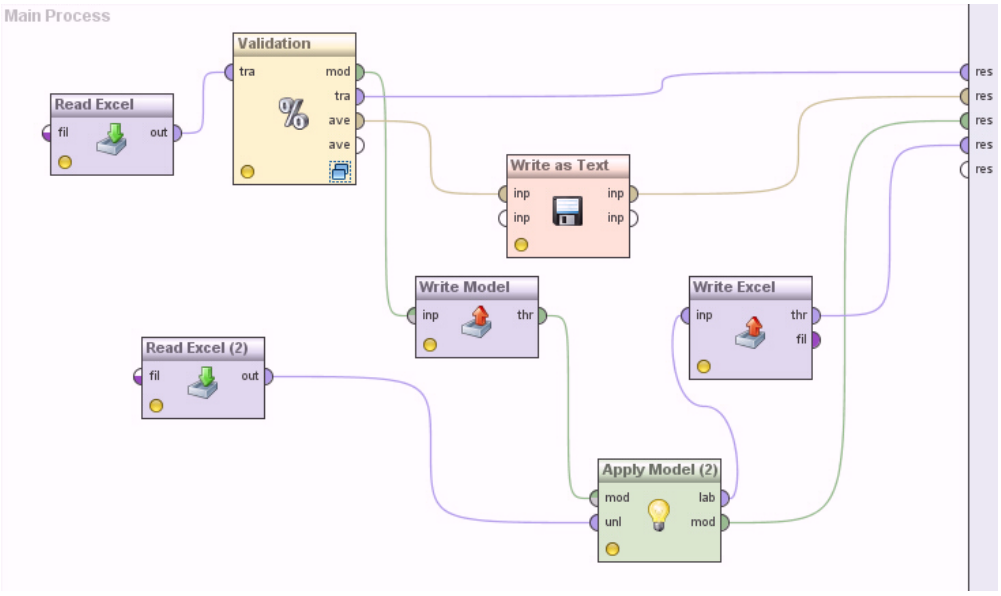
7662 εγγραφές που τοποθετήθηκαν σε ένα φύλο Excel της μορφής που φαίνεται στην Εικόνα 3.

	A	B	C	D	E	F
1	JOBS	MACH	ITER	SOLS	CONC	SET
2	20	5	500	1	BADVF	003.init
3	20	5	500	2	BADVF	003.init
4	20	5	500	3	BADVF	003.init
5	20	5	500	5	BADVF	003.init
6	20	5	500	8	BADVF	003.init
7	20	5	500	13	BADVF	003.init
8	20	5	500	21	BADVF	003.init
9	20	5	500	34	VGVF	003.init
10	20	5	500	89	VGVF	003.init
11	20	5	500	144	VGVF	003.init
12	20	5	1000	1	BADVF	003.init
13	20	5	1000	2	BADVF	003.init
14	20	5	1000	3	BADVF	003.init
15	20	5	1000	5	BADVF	003.init
16	20	5	1000	8	BADVF	003.init
17	20	5	1000	21	BADVF	003.init
18	20	5	1000	34	VGVF	003.init
19	20	5	1000	55	VGVF	003.init
20	20	5	1000	89	VGVF	003.init
21	20	5	1000	144	VGVF	003.init
22	20	5	1500	1	BADVF	003.init
23	20	5	1500	2	BADVF	003.init

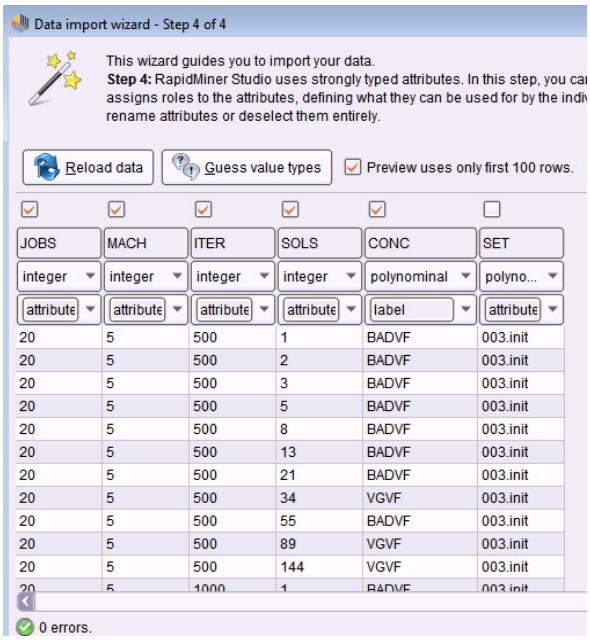
Εικόνα 3: Μέρος των δεδομένων εκτέλεσης του αλγορίθμου, έτοιμα για τη δημιουργία του μοντέλου.

Το μοντέλο μας στο RapidMiner έχει τη μορφή που εμφανίζεται στην Εικόνα 4.

Αρχικά με τη διεργασία Read Excel διαβάζουμε τα δεδομένα από το αρχείο Excel. Όπως φαίνεται στην Εικόνα 5, δεν συμπεριλαμβάνουμε την τελευταία στήλη (η οποία υπάρχει για λόγους ταυτοποίησης του προβλήματος Taillard που αφορά). Επίσης, είναι σημαντικό να ορίσουμε τη στήλη CONC με το χαρακτηριστικό LABEL. Με αυτό δηλώνουμε στο RapidMiner ότι αυτό είναι το πεδίο-στόχος, πάνω στο οποίο θέλουμε να εκπαιδεύσει το μοντέλο και για το οποίο θα ζητήσουμε προσδιορισμό τιμών. Το πεδίο αυτό λαμβάνει τιμές polynominal (πλήθος διακριτών μη αριθμητικών τιμών).



Εικόνα 4: Οι διαδικασίες για το μοντέλο μας στο λογισμικό RapidMiner.

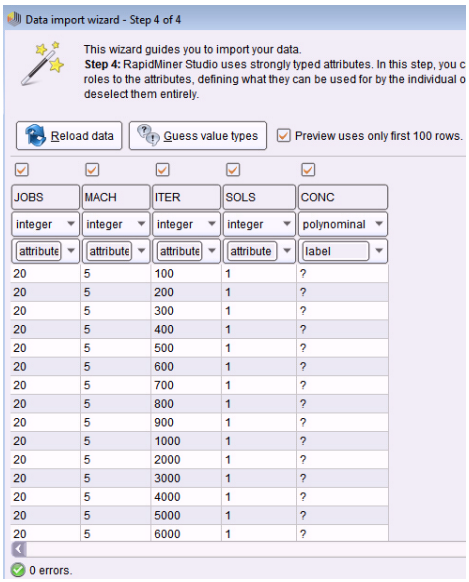


Εικόνα 5: Εισαγωγή δεδομένων εκπαίδευσης και χαρακτηρισμός πεδίων στο λογισμικό RapidMiner.

Η επόμενη διεργασία είναι η Validation. Αυτή αφενός εμπεριέχει το μοντέλο, άρα εντός αυτής θα εκπαιδευτεί και θα δημιουργηθεί το μοντέλο, και αφετέρου ελέγχει το μοντέλο που θα προκύψει για την αποτελεσματικότητά του ως προς τα δεδομένα της εκπαίδευσης. Δηλαδή, εφαρμόζει το μοντέλο στα δεδομένα εκπαίδευσης και ελέγχει την τιμή που το μοντέλο δίνει στο πεδίο CONC. Αν η τιμή είναι η ίδια με την πραγματική που έχει το πεδίο CONC στην εγγραφή που ελέγχει κάθε φορά, σημειώνει επιτυχία. Διαφορετικά, σημειώνει αποτυχία, καταγράφοντας παράλληλα την τιμή που εκτίμησε το μοντέλο. Αυτό είναι ιδιαίτερα σημαντικό γιατί είναι μικρότερο το πρόβλημα εάν εκτιμήσει μία τιμή VGVF ως VGF, αλλά μεγαλύτερο –για την ακρίβεια του μοντέλου- αν την εκτιμήσει ως BADSLOW. Ο έλεγχος έγινε χωρίζοντας το σύνολο των εγγραφών σε 10 ίσα μέρη και κάνοντας τη διαδικασία εκπαίδευσης και ελέγχου 10 φορές. Η δειγματοληψία έγινε με τη διαδικασία stratified sampling.

Η διεργασία Write as Text είναι βοηθητική και απλά γράφει τα αποτελέσματα αξιολόγησης του μοντέλου σε αρχείο, ώστε να τα έχουμε διαθέσιμα (θα μπορούσε να απουσιάζει από το μοντέλο).

Η διεργασία Write Model αποθηκεύει το εκπαιδευμένο μοντέλο σε αρχείο, ώστε να μπορούμε να κάνουμε με αυτό μελλοντικές εκτελέσεις, χωρίς να χρειάζεται να το εκπαιδεύσουμε εκ νέου (θα μπορούσε να απουσιάζει από το μοντέλο).



Εικόνα 6: Εισαγωγή δεδομένων για απαντήσεις και χαρακτηρισμός πεδίων στο λογισμικό RapidMiner.

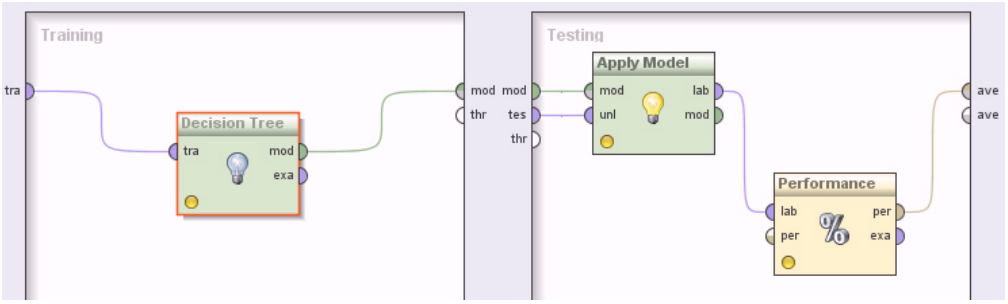
Η διεργασία Read Excel (2) διαβάζει το φύλλο Excel με τα δεδομένα που θέλουμε να συμπληρώσουμε, δηλαδή αυτά για τα οποία ζητάμε απαντήσεις από το μοντέλο. Όπως φαίνεται στην Εικόνα 6, από τα δεδομένα αυτά απουσιάζει το περιεχόμενο στο πεδίο CONC (δηλωμένο ως LABEL και polynominal), καθώς είναι αυτό ακριβώς το περιεχόμενο που θέλουμε να μας προσδιορίσει το μοντέλο. Δίνουμε τις διαστάσεις του προβλήματος (σύμφωνα με τα διάφορα μεγέθη του σετ Taillard, από 20x5 μέχρι 500x20), και για διαφόρους συνδυασμούς αριθμού επαναλήψεων και αριθμού λύσεων, ζητάμε να μας προσδιορίσει τι εκτιμά για την ποιότητα της λύσης (λ.χ. GVF, VGSLOW κ.λπ.)

Η διεργασία Apply Model εφαρμόζει το μοντέλο (όπως αυτό βγαίνει από το Validation) πάνω στα δεδομένα της διεργασίας Read Excel (2), δηλαδή σε αυτά που θέλουμε απαντήσεις. Το αποτέλεσμα οδηγείται στη διεργασία Write Excel, ώστε να έχουμε τα αποτελέσματα σε αρχείο Excel για να τα επεξεργαστούμε περαιτέρω, ενώ το οδηγούμε και στα αποτελέσματα του RapidMiner.

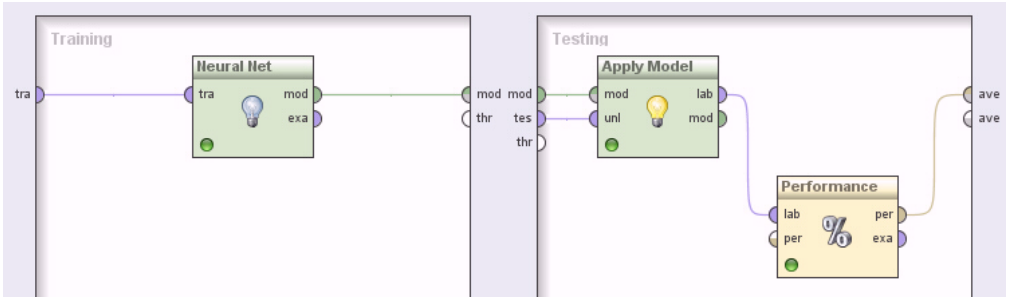
Αφήσαμε για το τέλος την αναφορά στο ουσιαστικότερο κομμάτι της διαδικασίας, στον «πυρήνα» του μοντέλου, δηλαδή στο είδος του μοντέλου που θα εφαρμόσουμε για να εκπαιδεύσουμε με τα δεδομένα μας. Η φύση των δεδομένων προέκρινε δύο ως τις επικρατέστερες προσεγγίσεις, αυτή του Δέντρου Απόφασης (Decision Tree) και των Νευρωνικών Δικτύων (Neural Networks). Το RapidMiner παρέχει και τις δύο αυτές

διαδικασίες, οπότε τις εφαρμόσαμε και τις δύο για να μπορούμε να παραλληλίσουμε και να συγκρίνουμε τα αποτελέσματα που δίνουν.

Ο «πυρήνας» του μοντέλου βρίσκεται στο εσωτερικό της διεργασίας Validation, που είναι ενφωλιασμένη (nested). Όπως φαίνεται στην Εικόνα 7, στο εσωτερικό της τοποθετούμε τη διεργασία που κάνει την εκπαίδευση, καθώς και τις διεργασίες που κάνουν τη δοκιμή. Στην Εικόνα 7 φαίνεται η δημιουργία του μοντέλου με Δέντρο Απόφασης, ενώ στην Εικόνα 8 η δημιουργία του μοντέλου με Νευρωνικό Δίκτυο.



Εικόνα 7: Δημιουργία μοντέλου με Δέντρο Απόφασης (Decision Tree).



Εικόνα 8: Δημιουργία μοντέλου με Νευρωνικό Δίκτυο (Neural Network).

Το μοντέλο που εκπαιδεύεται με τα δεδομένα εισόδου εφαρμόζεται στα ίδια δεδομένα μέσω της διεργασίας Apply Model. Η διεργασία Performance μας επιτρέπει να ορίσουμε τις παραμέτρους με τις οποίες θα ελέγξουμε την απόδοση του μοντέλου, πάνω στα δεδομένα εκπαίδευσης (ακόμα δεν ασχολούμαστε με τα δεδομένα για τα οποία εμείς επιθυμούμε απαντήσεις).

Οι διεργασίες Decision Tree και Neural Net δέχονται διάφορες παραμέτρους που επηρεάζουν τη λειτουργία τους και την αποτελεσματικότητά τους σε διάφορα σετ δεδομένων. Το RapidMiner διαθέτει τη διεργασία Optimize Parameters μέσω της οποίας μπορούμε να εκτελέσουμε ξανά και ξανά την ίδια εκπαίδευση με διαφορετικές

παραμέτρους, ώστε να δούμε ποια δίνει τα αποτελέσματα που είναι πλησιέστερα σε αυτά των δεδομένων εκπαίδευσης.

Χρησιμοποιώντας τη διεργασία Optimize Parameters, καταλήξαμε στην ακόλουθη παραμετροποίηση για τη διεργασία Decision Tree:

- Criterion: Information Gain
- Maximal Depth: 11
- Apply Pruning: ON
- Confidence: 0,25
- Apply Prepruning: ON
- Minimal Gain: 0,0
- Minimal Leaf Size: 5
- Minimal Size for Split: 11
- Number of prepruning alternatives: 100

Αντίστοιχα, Χρησιμοποιώντας τη διεργασία Optimize Parameters, καταλήξαμε στην ακόλουθη παραμετροποίηση για τη διεργασία Neural Net:

- Learning Rate: 0,18
- Momentum: 0,1
- Decay: OFF
- Shuffle: ON
- Normalize: ON
- Error epsilon: 1,0E-5

Για τη Neural Net εφαρμόστηκαν 20.000 κύκλοι εκπαίδευσης.

Να σημειωθεί ότι η βελτιστοποίηση της παραμετροποίησης στις παραμέτρους του Δέντρου Απόφασης και του Νευρωνικού Δικτύου, οδήγησε σε συνολική αύξηση της απόδοσης του μοντέλου περίπου κατά 80%, αφού οι πρώτες δοκιμές έδιναν μέσο ποσοστό επιτυχίας της τάξης του 35%, ενώ καταλήξαμε σε ποσοστό της τάξης του 65%. Η βελτιστοποίηση των παραμέτρων θα μπορούσε να ήταν και πιο διεξοδική, αλλά ο παράγων χρόνος είναι σημαντικός, καθώς μία και μόνη δημιουργία μοντέλου Νευρωνικού Δικτύου με 20.000 κύκλους εκπαίδευσης, απαιτεί 6 ώρες. Συνεπώς, αν η

ανεύρεση των καλύτερων παραμέτρων απαιτεί 20, 30 ή 50 διαφορετικούς συνδυασμούς παραμέτρων, γίνεται κατανοητό ότι ο χρόνος που απαιτείται για πιο ενδελεχή βελτιστοποίηση της παραμετροποίησης, είναι εξαιρετικά μεγάλος (χωρίς από ένα σημείο και μετά να συνεισφέρει σημαντικά στη βελτίωση της απόδοσης του μοντέλου).

4.7 Αξιολόγηση αποτελεσμάτων (βήμα 5)

Στην ενότητα αυτή θα ασχοληθούμε με την αξιολόγηση των αποτελεσμάτων, όπως προέκυψαν από τα δύο μοντέλα μας, ξεκινώντας από την παρουσίαση των αποτελεσμάτων.

4.7.1 Αποτελέσματα μοντέλου Δέντρου Απόφασης

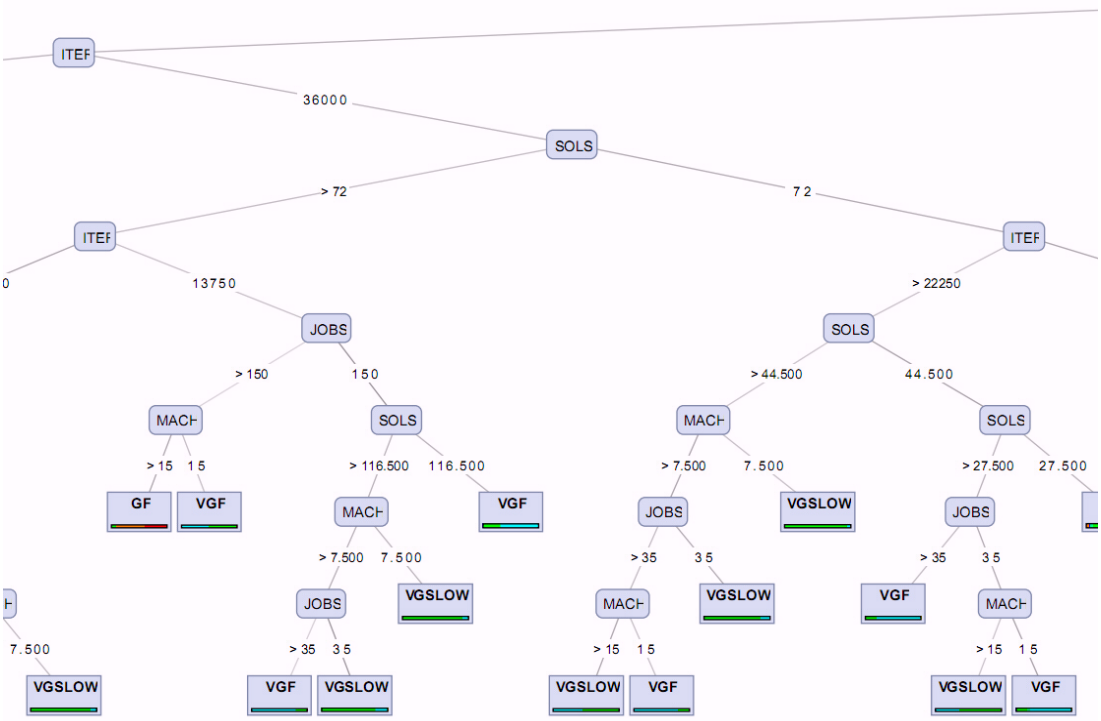
Η συνολική ακρίβεια του μοντέλου με Δέντρο Απόφασης είναι 64,89% (±1,18%), η διασπορά τους ανά κατηγορία εμφανίζεται στον Πίνακα 5.

Ο Πίνακας 5 στις στήλες μας δίνει την ακρίβεια με την οποία έγινε η αντιστοίχηση των πραγματικών αποτελεσμάτων από το μοντέλο. Για παράδειγμα, οι εκτελέσεις που ήταν χαρακτηρισμένες ως BADVF, σε ποσοστό 70,29% εκτιμήθηκαν ως BADVF και από το μοντέλο (1133 εκτελέσεις). Υπήρξαν, όμως, και εκτελέσεις που αν και BADVF στην πραγματικότητα, το μοντέλο της εκτίμησε ως VGVF (263), VGF (34), BADF (11), GVF (169) και GF (2).

	true BADVF	true VGVF	true VGF	true BADF	true VGSLOW	true BADSLOW	true GVF	true GF	true GSLOW	class precision
pred. BADVF	1133	287	28	30	0	0	339	12	0	61.95%
pred. VGVF	263	577	140	13	2	0	206	5	0	47.84%
pred. VGF	34	99	1523	178	193	2	30	177	3	68.02%
pred. BADF	11	1	22	74	1	0	6	18	1	55.22%
pred. VGSLOW	0	0	97	15	1115	13	0	2	11	88.99%
pred. BADSLOW	0	0	0	0	0	0	0	0	0	0.00%
pred. GVF	169	142	14	5	0	0	430	34	0	54.16%
pred. GF	2	4	33	18	4	0	17	119	7	58.33%
pred. GSLOW	0	0	0	0	1	0	0	1	1	33.33%
class recall	70.29%	51.98%	82.01%	22.22%	84.73%	0.00%	41.83%	32.34%	4.35%	

Πίνακας 5: Αποτελέσματα μοντέλου Δέντρου Απόφασης

Στις γραμμές βλέπουμε την ακρίβεια των προβλέψεων. Για παράδειγμα, όταν το μοντέλο προβλέπει εγγραφή ως BADVF, κατά 61,95% είναι πράγματι BADVF (1133 εκτελέσεις). Όμως, με βάση τις εκτελέσεις, προέβλεψε ως BADVF και εκτελέσεις που ήταν VGVF (287), VGF (28), BADF (30), GVF (339) και GF (12).



Εικόνα 9: Στιγμιότυπο Δέντρου Απόφασης.

Το ίδιο το Δέντρο Απόφασης είναι αδύνατο να απεικονιστεί στις διαστάσεις μιας σελίδας. Ενδεικτικά στην Εικόνα 9 παρουσιάζεται ένα στιγμιότυπο αυτού.

Κάθε κλαδί του δέντρου αποτελεί ένα σημείο απόφασης για μία μεταβλητή. Ακολουθώντας τα κλαδιά, ανάλογα με τις τιμές που έχουμε στις τέσσερις μεταβλητές απόφασης που χρησιμοποιούμε, καταλήγουμε στο χαρακτηρισμό για το συγκεκριμένο συνδυασμό μεταβλητών.

4.7.2 Αποτελέσματα μοντέλου Νευρωνικού Δικτύου

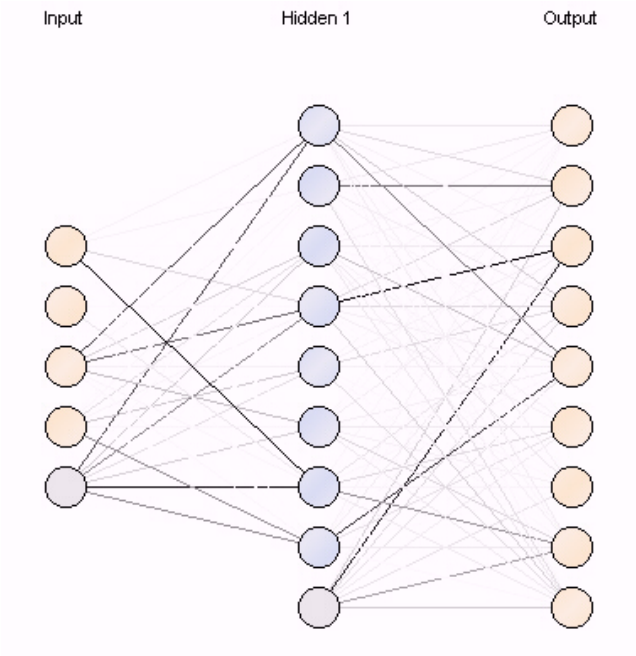
Η συνολική ακρίβεια του μοντέλου με Νευρωνικά Δίκτυα είναι 63,36% (±1,56%), η διασπορά τους ανά κατηγορία εμφανίζεται στον Πίνακα 6.

Η ανάγνωση του Πίνακα 6 γίνεται με τον ίδιο τρόπο που αναλύσαμε στον Πίνακα 5.

	true BADVF	true VGVF	true VGF	true BADF	true VGSLOW	true BADSLOW	true GVF	true GF	true GSLOW	class precision
pred. BADVF	1242	356	43	52	0	0	379	19	0	59.40%
pred. VGVF	196	414	113	13	0	0	165	7	0	45.59%
pred. VGF	34	182	1575	207	198	4	72	214	5	63.23%
pred. BADF	7	0	5	16	0	0	0	5	1	47.06%
pred. VGSLOW	0	0	88	6	1114	11	0	5	8	90.42%
pred. BADSLOW	0	0	0	0	0	0	0	0	0	0.00%
pred. GVF	125	157	22	4	1	0	401	27	0	54.41%
pred. GF	8	1	11	33	3	0	11	88	4	55.35%
pred. GSLOW	0	0	0	2	0	0	0	3	5	50.00%
class recall	77.05%	37.30%	84.81%	4.80%	84.65%	0.00%	39.01%	23.91%	21.74%	

Πίνακας 6: Αποτελέσματα μοντέλου Νευρωνικού Δικτύου.

Το Νευρωνικό Δίκτυο που δημιουργείται έχει τη δομή της Εικόνας 10. Οι κόμβοι στην έξοδο (Output) είναι οι 9 διαφορετικές κλάσεις, ενώ έχουν καταγραφεί αναλυτικά οι τιμές σιγμοειδών για το κρυφό επίπεδο.



Εικόνα 10: Δομή του Νευρωνικού Δικτύου.

4.8 Εφαρμογή (βήμα 6)

Φτάνουμε στο τελευταίο βήμα της εξόρυξης δεδομένων, με την εφαρμογή του μοντέλου στα δεδομένα που επιθυμούμε να μάθουμε την απάντηση που εκτιμά το μοντέλο. Όπως είδαμε στην ενότητα 4.6, όταν ολοκληρωθεί η εκπαίδευση του

μοντέλου, μετά εισάγονται σε αυτό οι συνδυασμοί για τους οποίους θέλουμε εκτίμηση και λαμβάνουμε τα αποτελέσματα του μοντέλου σε ένα φύλλο Excel. Τα αποτελέσματα έχουν τη μορφή της Εικόνας 11.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	JOBS	MACH	ITER	SOLS	CONC	confidence (BADVF)	confidence (VGVF)	confidence (VGF)	confidence (BADF)	confidence (VGSLOW)	confidence (BADSLOW	confidence (GVF)	confidence (GF)	confidence (GSLOW)	prediction (CONC)
2	20	5	100	1		,833	,099	,000	,000	,000	,000	,068	,000	,000	BADVF
3	20	5	200	1		,829	,096	,000	,000	,000	,000	,075	,000	,000	BADVF
4	20	5	300	1		,823	,095	,000	,000	,000	,000	,082	,000	,000	BADVF
5	20	5	400	1		,817	,094	,000	,000	,000	,000	,089	,000	,000	BADVF
6	20	5	500	1		,810	,093	,000	,000	,000	,000	,096	,000	,000	BADVF
7	20	5	600	1		,803	,093	,000	,000	,000	,000	,103	,000	,000	BADVF
8	20	5	700	1		,796	,094	,000	,000	,000	,000	,110	,000	,000	BADVF
9	20	5	800	1		,789	,095	,000	,000	,000	,000	,116	,000	,000	BADVF
10	20	5	900	1		,781	,096	,000	,001	,000	,000	,122	,000	,000	BADVF
11	20	5	1000	1		,774	,098	,000	,001	,000	,000	,128	,000	,000	BADVF
12	20	5	2000	1		,710	,121	,000	,004	,000	,000	,165	,000	,000	BADVF
13	20	5	3000	1		,674	,140	,000	,008	,000	,000	,177	,000	,000	BADVF
14	20	5	4000	1		,659	,151	,001	,010	,000	,000	,180	,000	,000	BADVF
15	20	5	5000	1		,652	,156	,001	,011	,000	,000	,180	,000	,000	BADVF
16	20	5	6000	1		,650	,158	,001	,012	,000	,000	,180	,000	,000	BADVF
17	20	5	7000	1		,649	,159	,002	,012	,000	,000	,179	,000	,000	BADVF
18	20	5	8000	1		,649	,160	,002	,012	,000	,000	,177	,000	,000	BADVF
19	20	5	9000	1		,650	,160	,002	,013	,000	,000	,176	,000	,000	BADVF
20	20	5	10000	1		,652	,160	,002	,013	,000	,000	,173	,000	,000	BADVF
21	20	5	15000	1		,679	,158	,003	,016	,000	,000	,144	,000	,000	BADVF
22	20	5	20000	1		,736	,142	,007	,029	,000	,000	,087	,000	,000	BADVF
23	20	5	25000	1		,776	,110	,008	,062	,000	,000	,045	,000	,000	BADVF
24	20	5	30000	1		,765	,080	,016	,121	,000	,000	,019	,000	,000	BADVF
25	20	5	35000	1		,715	,067	,036	,172	,000	,000	,011	,000	,000	BADVF
26	20	5	40000	1		,669	,063	,056	,203	,000	,000	,009	,000	,000	BADVF
27	20	5	45000	1		,629	,061	,075	,226	,000	,000	,009	,000	,000	BADVF
28	20	5	50000	1		,587	,060	,096	,248	,000	,000	,009	,000	,000	BADVF
29	20	5	55000	1		,542	,057	,121	,270	,000	,000	,010	,000	,000	BADVF

Εικόνα 11: Αποτελέσματα Μοντέλου.

Η στήλη CONC είναι κενή, καθώς έτσι την εισάγαμε. Στις στήλες A, B, C και D υπάρχουν οι 4 μεταβλητές, ενώ στη στήλη O αναφέρεται η εκτίμηση για το συγκεκριμένο συνδυασμό (το ζητούμενο δηλαδή). Στις στήλες F έως N αναφέρονται τα ποσοστά εμπιστοσύνης για κάθε κλάση. Για παράδειγμα, από τη γραμμή 2, για 20 εργασίες και 5 μηχανές, αν χρησιμοποιήσουμε 100 επαναλήψεις και μία λύση, με εμπιστοσύνη 0,833 θα λάβουμε μία λύση BADVF (κακή, αλλά γρήγορη). Όμως, εκτιμάται με εμπιστοσύνη 0,099 ότι η λύση μπορεί να είναι VGVF (πολύ καλή και πολύ γρήγορη) και με εμπιστοσύνη 0,068 ότι η λύση μπορεί να είναι GVF (καλή και πολύ γρήγορη). Αντίστοιχα είναι τα αποτελέσματα και στις υπόλοιπες 6200 περίπου εγγραφές που ζητήσαμε να μας εκτιμήσει το μοντέλο.

Εμάς μας ενδιαφέρουν οι λύσεις με χαρακτηριστικό VGVF, οπότε σε κάθε ομάδα προβλημάτων του Taillard (δυνητικά και σε άλλα μεγέθη προβλημάτων) θα αναζητήσουμε τους συνδυασμούς με την υψηλότερη εμπιστοσύνη στο VGVF. Όμως, εφόσον η εμπιστοσύνη στο VGVF δεν είναι 1 (που σημαίνει ότι το μοντέλο είναι απόλυτα βέβαιο ότι ο συνδυασμός αυτός λειτουργεί ως VGVF και μόνο), δεν αρκεί

αυτό. Θα πρέπει να κοιτάξουμε που βρίσκονται και οι τιμές εμπιστοσύνης των υπολοίπων συνδυασμών. Έτσι, θα προτιμήσουμε μία λύση όπου όλες οι τιμές εμπιστοσύνης είναι στις «θετικές κατηγορίες» (VGVF, GVF, VGF, GF) από μία λύση που μπορεί να έχει υψηλότερη εμπιστοσύνη στο VGVF, αλλά να έχει και τιμές σε «αρνητικές κατηγορίες» (BADSLOW, GSLOW, VGSLOW, BADF, BADVF). Κατά τη μελέτη των αποτελεσμάτων σε κάθε ομάδα προβλημάτων του Taillard θα αναπτύσσουμε το σκεπτικό επιλογής κάθε συνδυασμού.

Καθώς έχουμε αποτελέσματα και από το Νευρωνικό Δίκτυο και από το Δέντρο Απόφασης, θα αναζητούμε τις προτάσεις και των δύο μοντέλων, καταγράφοντας πού υπάρχει ομοφωνία και πού –και σε τι βαθμό- διαφωνία. Η όλη διαδικασία γίνεται χειροκίνητα, μέσω της διαδικασίας των φίλτρων του Excel, για κάθε ομάδα προβλημάτων του Taillard. Η διαδικασία αυτή επιβεβαιώνει, όπως θα διαπιστωθεί και στη συνέχεια, ότι η εφαρμογή του μοντέλου ενσωματώνει και τον υποκειμενισμό του αποφασίζοντα, καθώς σε ελάχιστες περιπτώσεις οι προτάσεις από τα μοντέλα είναι «απόλυτες»

4.8.1 Προβλήματα 20x5

Το Δέντρο Απόφασης δίνει συνδυασμό VGVF με μεγαλύτερη εμπιστοσύνη VGVF στο 0,693. Όμως, οι συγκεκριμένοι συνδυασμοί έχουν τιμή εμπιστοσύνης στο BADVF ύψους 0,140. Άρα, μόνο 0,860 εμπιστοσύνη είναι σε «θετικές» κατηγορίες. Με μικρότερη εμπιστοσύνη VGVF (0,636), υπάρχουν συνδυασμοί με όλα τα υπόλοιπα σε «θετικές» κατηγορίες (VGF – 0,274 και GVF – 0,91), οπότε καθίστανται προτιμότεροι. Είναι οι συνδυασμοί που εμφανίζονται στον Πίνακα 7.

ΕΠΑΝΑΛΗΨΕΙΣ	ΛΥΣΕΙΣ
800	75
900	75
1000	75
800	90
900	90
1000	90
800	100
900	100
1000	100

Πίνακας 7: Καλύτεροι συνδυασμοί από Δέντρο Απόφασης για πρόβλημα 20x5

Στο Νευρωνικό Δίκτυο, οι συνδυασμοί με τη μεγαλύτερη εμπιστοσύνη σε «θετικές» κατηγορίες (max 0,98) είναι της κατηγορίας VGF. Εστιάζοντας αποκλειστικά σε συνδυασμούς κατηγορίας VGVF, με μέγιστο το 0,952 και ελάχιστο 0,928 για τα 10 πρώτα, έχουμε τους συνδυασμούς του Πίνακα 8.

ΕΠΑΝΑΛΗΨΕΙΣ	ΛΥΣΕΙΣ
900	75
1000	60
800	75
900	60
800	90
700	75
800	60
800	100
700	90
700	60

Πίνακας 8: Καλύτεροι συνδυασμοί από Νευρωνικό Δίκτυο για πρόβλημα 20x5

Άρα, οι συνδυασμοί 800/75 και 900/75 είναι οι καταλληλότεροι για την κατηγορία αυτή (850/75).

4.8.2 Προβλήματα 20x10

Από το Δέντρο Απόφασης έχουμε συνδυασμούς με εμπιστοσύνη 1 σε «θετικές» κατηγορίες, όπως φαίνονται στον Πίνακα 9.

Στο Νευρωνικό Δίκτυο, οι συνδυασμοί με τη μεγαλύτερη εμπιστοσύνη σε «θετικές» κατηγορίες (max 0,98) είναι της κατηγορίας VGF. Εστιάζοντας αποκλειστικά σε συνδυασμούς κατηγορίας VGVF, με μέγιστο το 0,925 και ελάχιστο 0,898 για τα 10 πρώτα, έχουμε τους συνδυασμούς του Πίνακα 10.

Καθώς οι συνδυασμοί 1000/75 και 1000/90 εμφανίζονται συνολικά σε υψηλότερες και στους δύο πίνακες, είναι οι προτιμότεροι της κατηγορίας (1000/83).

ΕΠΑΝΑΛΗΨΕΙΣ	ΛΥΣΕΙΣ
100	120
200	120
300	120
400	120
500	120
600	120
700	120
2000	45
2000	60
800	75
900	75
1000	75
800	90
900	90
1000	90
800	100
900	100
1000	100
3000	35

Πίνακας 9: Καλύτεροι συνδυασμοί από Δέντρο Απόφασης για πρόβλημα 20x10

ΕΠΑΝΑΛΗΨΕΙΣ	ΛΥΣΕΙΣ
1000	75
1000	90
900	75
2000	45
1000	60
900	90
800	75
900	60
1000	75
1000	90

Πίνακας 10: Καλύτεροι συνδυασμοί από Νευρωνικό Δίκτυο για πρόβλημα 20x10

4.8.3 Προβλήματα 20x20

Από το Δέντρο Απόφασης, οι συνδυασμοί με την υψηλότερη εμπιστοσύνη (1) σε θετικές κατηγορίες, έχουν εμπιστοσύνη VGVF μόλις 0,5 (9000/8, 10000/8, 3000/35). Αναζητώντας μεγαλύτερη εμπιστοσύνη στην κατηγορία VGVF φτάνουμε στα επίπεδα συνολικής «θετικής» εμπιστοσύνης (0,976) για τους συνδυασμούς 2000/25 και 2000/35.

Στο Νευρωνικό Δίκτυο οι συνδυασμοί με την υψηλότερη εμπιστοσύνη σε «θετικές» κατηγορίες, εμφανίζονται στον Πίνακα 11 (μέγιστη 0,972).

ΕΠΑΝΑΛΗΨΕΙΣ	ΛΥΣΕΙΣ
2000	60
3000	35
1000	75
1000	60
2000	45
900	75
1000	90
900	60
900	90
800	75

Πίνακας 11: Καλύτεροι συνδυασμοί από Νευρωνικό Δίκτυο για πρόβλημα 20x20

Το 3000/35 είναι ο κοινός συνδυασμός της κατηγορίας, παρά το γεγονός ότι υπάρχουν αποκλίσεις μεταξύ των αποτελεσμάτων από τα δύο μοντέλα.

4.8.4 Προβλήματα 50x5

Στο Δέντρο Απόφασης, οι συνδυασμοί με τη μεγαλύτερη θετική εμπιστοσύνη είναι VGF (με εμπιστοσύνη VGVF μικρότερη από 0,065). Εστιάζοντας στην κατηγορία VGVF, οι καλύτεροι συνδυασμοί είναι 4000/25 και 5000/25 (θετική εμπιστοσύνη 0,889), ενώ κοντά είναι και οι συνδυασμοί 800/45, 900/45, 1000/45, 800/60, 900/60, 1000/60 (με 0,875) και οι συνδυασμοί 3000/25 και 3000/35 (με θετική εμπιστοσύνη 0,872).

Στο Νευρωνικό Δίκτυο οι συνδυασμοί με τη μεγαλύτερη θετική εμπιστοσύνη είναι VGF (0,984) και είναι οι 6000/45, 5000/45, 7000/45 και 8000/45, με χαμηλή όμως εμπιστοσύνη VGVF). Ως VGVF ο συνδυασμός 2000/60 έχει τη μεγαλύτερη θετική εμπιστοσύνη, ενώ ακολουθούν οι 1000/90, 1000/75, 1000/100, 900/75, 900/90 και 1000/60.

Δεν έχουμε σύγκλιση των δύο μοντέλων, οπότε ένας συνδυασμός μεταξύ των 4000/25 και 2000/60, όπως ο 3000/45 φαίνεται ως μία κατάλληλη ενδιάμεση επιλογή.

4.8.5 Προβλήματα 50x10

Στα προβλήματα 50x10, εμφανίζεται διαφοροποίηση στα δύο μοντέλα. Κοινή εκτίμηση φαίνεται στο συνδυασμό 30000/4, όπου στο Δέντρο Απόφασης εκτιμάται ως VGVF (με 1 θετική εμπιστοσύνη) και στο Νευρωνικό Δίκτυο GVF, με θετική εμπιστοσύνη 0,931 και εμπιστοσύνη VGVF στο 0,192.

4.8.6 Προβλήματα 50x20

Στα προβλήματα 50x20, εμφανίζεται διαφοροποίηση στα δύο μοντέλα. Το Δέντρο Απόφασης προκρίνει ως «θετικούς» συνδυασμούς με 1 έως 8 λύσεις, ενώ το Νευρωνικό Δίκτυο έχει προτάσεις μεγαλύτερη γκάμα, αλλά με την εμπιστοσύνη τους να είναι κυρίως στις περιοχές VGF, GF και GVF. Μία κάποια σύγκληση βρίσκουμε στους συνδυασμούς 50000/4 και 10000/8, οπότε θα προκρίνουμε τη μέση λύση του συνδυασμού 30000/6.

4.8.7 Προβλήματα 100x5

Και στα προβλήματα 100x5 εμφανίζεται διαφοροποίηση στα δύο μοντέλα. Σύγκληση εντοπίζουμε στο συνδυασμό 55000/2, με 1 θετική εμπιστοσύνη στο Δέντρο Απόφασης και χαρακτηρισμό VGVF και 0,97 θετική εμπιστοσύνη στο Νευρωνικό Δίκτυο και χαρακτηρισμό VGF.

4.8.8 Προβλήματα 100x10

Και στα προβλήματα 100x10 εμφανίζεται διαφοροποίηση στα δύο μοντέλα. Σύγκληση εντοπίζουμε στο συνδυασμό 55000/2, με 1 θετική εμπιστοσύνη στο Δέντρο Απόφασης και χαρακτηρισμό VGVF και 0,958 θετική εμπιστοσύνη στο Νευρωνικό Δίκτυο και χαρακτηρισμό VGF.

4.8.9 Προβλήματα 100x20

Και στα προβλήματα 100x20 εμφανίζεται διαφοροποίηση στα δύο μοντέλα. Σύγκληση εντοπίζουμε στο συνδυασμό 55000/2, με 1 θετική εμπιστοσύνη στο Δέντρο Απόφασης και χαρακτηρισμό VGVF και 0,921 θετική εμπιστοσύνη στο Νευρωνικό Δίκτυο και χαρακτηρισμό VGF.

Να σημειωθεί ότι και στα τρία σετ προβλημάτων με 100 εργασίες υπάρχουν και άλλοι συνδυασμοί με ικανοποιητική εμπιστοσύνη, ειδικά στο Δέντρο Απόφασης, που έχει

πολλούς συνδυασμούς με εμπιστοσύνη 1. Στόχος μας ήταν να βρούμε το συνδυασμό που υποδεικνύεται με μεγάλη εμπιστοσύνη και καλό χαρακτηρισμό και στα δύο μοντέλα, έστω κι εάν σε κάθε μοντέλο προτείνονται αρκετοί άλλοι συνδυασμοί με – κατά την εκτίμηση του μοντέλου- δυνατότητα για καλά και γρήγορα αποτελέσματα.

4.8.10 Προβλήματα 200x10

Στα προβλήματα 200x10 εμφανίζεται διαφοροποίηση στα δύο μοντέλα. Σύγκληση εντοπίζουμε στο συνδυασμό 90000/1, με 1 θετική εμπιστοσύνη στο Δέντρο Απόφασης και χαρακτηρισμό VGVF και 0,998 θετική εμπιστοσύνη στο Νευρωνικό Δίκτυο και χαρακτηρισμό VGF.

4.8.11 Προβλήματα 200x20

Και στα προβλήματα 200x20 εμφανίζεται διαφοροποίηση στα δύο μοντέλα. Σύγκληση εντοπίζουμε στο συνδυασμό 90000/1, με 1 θετική εμπιστοσύνη στο Δέντρο Απόφασης και χαρακτηρισμό VGVF και 0,997 θετική εμπιστοσύνη στο Νευρωνικό Δίκτυο και χαρακτηρισμό VGF.

4.8.12 Προβλήματα 500x20

Στα προβλήματα 500x20 εμφανίζεται διαφοροποίηση στα δύο μοντέλα. Σύγκληση εντοπίζουμε στο συνδυασμό 90000/2, με 1 θετική εμπιστοσύνη στο Δέντρο Απόφασης και χαρακτηρισμό VGVF και 0,978 θετική εμπιστοσύνη στο Νευρωνικό Δίκτυο και χαρακτηρισμό VGF.

4.8.13 Τελικοί συνδυασμοί

Με βάση τις αναλύσεις στις παραπάνω παραγράφους, συγκεντρώνουμε εδώ (Πίνακας 12) τους προτεινόμενους συνδυασμούς επαναλήψεων και λύσεων για κάθε διάσταση προβλήματος από το σετ προβλημάτων του Taillard.

Φυσικά θα μπορούσαμε να επιλέξουμε και άλλους συνδυασμούς με ικανοποιητική εμπιστοσύνη, σύμφωνα με τα αποτελέσματα των μοντέλων. Όμως, για να έχουμε ένα σετ αποτελεσμάτων επιλέξαμε ένα συνδυασμό ανά μέγεθος προβλήματος. Λήψη μετρήσεων με άλλους συνδυασμούς παραμέτρων που τα μοντέλα προτείνουν ως «καλούς» θα είχε ενδιαφέρον για να δούμε εάν οδηγούν σε καλύτερα, χειρότερα ή παρόμοια αποτελέσματα με τις τελικές μας επιλογές.

ΔΙΑΣΤΑΣΗ	ΕΠΑΝΑΛΗΨΕΙΣ	ΛΥΣΕΙΣ
20x5	850	75
20x10	1000	83
20x20	3000	35
50x5	3000	45
50x10	30000	4
50x20	30000	6
100x5	55000	2
100x10	55000	2
100x20	55000	2
200x10	90000	1
200x20	90000	1
500x20	90000	2

Πίνακας 12: Καλύτεροι συνδυασμοί ανά διάσταση προβλήματος με βάση τα αποτελέσματα των μοντέλων εξόρυξης δεδομένων

4.9 Ολοκληρώνοντας με την εξόρυξη δεδομένων

Τα αποτελέσματα που παρατίθενται στον Πίνακα 12, επιβεβαιώνουν τόσο την «αίσθηση» που αναφέραμε στην αρχή του κεφαλαίου ότι διαφορετική παραμετροποίηση απαιτείται στα «μικρά» προβλήματα και διαφορετική στα «μεγάλα». Είναι εμφανές από τον Πίνακα 12, ότι τα προβλήματα με λίγες εργασίες απαιτούν λιγότερες επαναλήψεις αλλά μεγαλύτερο αριθμό λύσεων, ενώ τα προβλήματα με πολλές εργασίες το αντίστροφο, δηλαδή πολλές επαναλήψεις και λίγες εργασίες.

Το άλλο ενδιαφέρον συμπέρασμα από τον Πίνακα 12 είναι η μη «ομαλή» διαφοροποίηση των παραμέτρων ανά κατηγορία προβλήματος. Έτσι βλέπουμε περιπτώσεις που αυξάνοντας τη διάσταση του προβλήματος να αυξάνονται αντί να μειώνονται (κατά τη γενική τάση) οι λύσεις που προτείνουν τα μοντέλα. Επίσης, σε άλλα προβλήματα με ίδιο αριθμό εργασιών οι επαναλήψεις μένουν σταθερές (λ.χ. προβλήματα με 100 εργασίες), ενώ σε άλλα μεταβάλλονται και μάλιστα σημαντικά (λ.χ. στα προβλήματα με 50 εργασίες). Τέτοια συμπεριφορά θα ήταν αδύνατο να εντοπιστεί με στατιστικές διαδικασίες, όπως λ.χ. την παλινδρόμηση.

Επίσης, αν και ο αριθμός λύσεων του κάθε προβλήματος επηρεάζεται από τον αριθμό των εργασιών (για παράδειγμα ένα πρόβλημα με 20 εργασίες έχει 20! εφικτές λύσεις, ανεξαρτήτως του αριθμού μηχανών και ένα πρόβλημα με 50 εργασίες έχει 50! εφικτές λύσεις, επίσης ανεξαρτήτως του αριθμού των μηχανών), εντούτοις ο αριθμός των μηχανών επηρεάζει σημαντικά την πολυπλοκότητα του προβλήματος. Καθώς ο

αλγόριθμος βλέπει ως «μαύρο κουτί» την αντικειμενική συνάρτηση και ασχολείται μόνο με το μήκος της λύσης (αν λ.χ. είναι 20 ή 50 εργασιών) ο τρόπος που επηρεάζουν οι μηχανές τη συμπεριφορά του αλγορίθμου δεν είναι εμφανής. Αυτό προκύπτει μόνο ως αποτέλεσμα από τις εκτελέσεις, αλλά και από τις προτάσεις των μοντέλων, όπου σε προβλήματα ίδιου αριθμού εργασιών και διαφορετικού αριθμού μηχανών, προτείνουν άλλο συνδυασμό παραμέτρων.

Προτιμήσαμε να εφαρμόσουμε και τα δύο μοντέλα, το Δέντρο Απόφασης και τα Νευρωνικά Δίκτυα, ώστε να ισχυροποιήσουμε τις επιλογές μας. Αν και συχνά τα δύο μοντέλα είχαν διαφοροποιήσεις στις προτάσεις τους, εντούτοις θα πρέπει να επισημανθεί ότι ουσιαστικά οι διαφοροποιήσεις αυτές ήταν οριακές. Μπορεί λ.χ. ένας συνδυασμός να έγινε εμπιστοσύνη 1 στο ένα μοντέλο, αλλά ο ίδιος συνδυασμός στο άλλο μοντέλο έδινε εμπιστοσύνη της τάξης του 0,97, δηλαδή εξαιρετικά υψηλή. Άρα, η αναφορές που κάναμε ότι τα δύο μοντέλα είχαν διαφοροποιήσεις ως προς τις προτάσεις τους, είχαν καθαρά οριακό χαρακτήρα και όχι ουσιαστικό.

Για κάποια σετ προβλημάτων, κυρίως τα μεγαλύτερα, το μοντέλο των Νευρωνικών Δικτύων δεν μας έδωσε λύσεις με χαρακτηρισμό VGVF. Αυτό σχετίζεται με τον τρόπο που λειτουργεί η εκπαίδευση του συγκεκριμένου μοντέλου, αλλά και με τον αριθμό των διαθέσιμων εκτελέσεων (ήταν σημαντικά περισσότερες στα προβλήματα μικρότερων διαστάσεων). Όμως αυτό δεν μας οδηγεί σε αδιέξοδο για επιλογή των καλύτερων συνδυασμών, καθώς μας ενδιαφέρει συνολικά η εμπιστοσύνη τους να βρίσκεται σε «θετικές» εκτιμήσεις (να είναι γρήγορες και καλές). Με αυτήν την παραδοχή εντοπίσαμε τις ποιοτικές λύσεις και από το Νευρωνικό Δίκτυο, και τις ταιριάξαμε με αυτές από το Δέντρο Απόφασης, το οποίο σημειωτέον έδωσε πάντα VGVF προτάσεις και μάλιστα με «θετική» εμπιστοσύνη στο 1.

Αντίστοιχη διαδικασία εξόρυξης δεδομένων θα μπορούσε να εφαρμοστεί και για τα επαναληπτικά σετ καθώς και για τις υπόλοιπες παραμέτρους του αλγορίθμου. Όμως, πλέον έχει γίνει περισσότερο από εμφανές το πόσο μία και μόνη προσθήκη παραμέτρους, θα αύξανε τις απαιτούμενες εκτελέσεις και άρα το χρόνο για λήψη των αποτελεσμάτων. Άλλωστε, σε κάθε εφαρμογή εξόρυξης δεδομένων γίνονται κάποιες παραχωρήσεις ως προς το τι θεωρείται σταθερό, διότι η πλήρης παραμετροποίηση των πάντων, θα οδηγούσε σε πρακτικά αδιέξοδα.

Συνολικά, η διαδικασία της εξόρυξης δεδομένων ήταν επιτυχής, καθώς δεν θα μπορούσαμε με άλλο τρόπο να εντοπίσουμε τους καλύτερους συνδυασμούς παραμέτρων. Αυτό που μένει τώρα να δούμε είναι εάν πράγματι αυτοί οι συνδυασμοί «δικαιώνουν» την επιλογή τους, δηλαδή το αντικείμενο του επόμενου κεφαλαίου.

"Time is the only commodity that is irreplaceable:

invest it, share it, spend it...but never waste it."

Tracy Sherwood

Κεφάλαιο 5:

Εκτέλεση του αλγορίθμου - Αποτελέσματα

Στο κεφάλαιο αυτό θα χρησιμοποιήσουμε τους συνδυασμούς παραμέτρων του Πίνακα 12 από το προηγούμενο κεφάλαιο και θα εκτελέσουμε με τις ανάλογες παραμέτρους τα προβλήματα από το σετ του Taillard. Κάθε πρόβλημα εκτελείται 10 φορές, σύμφωνα με τις επιταγές της σχετικής βιβλιογραφίας, για να περιορίσουμε την επίδραση της στοχαστικότητας, και καταγράφεται η καλύτερη και η μέση τιμή του αποτελέσματος (makespan) καθώς και ο μέσος χρόνος. Στη συνέχεια προχωράμε σε αξιολόγηση των αποτελεσμάτων αυτών.

5.1 Παράθεση αποτελεσμάτων

Κάθε πρόβλημα από το σετ του Taillard εκτελέστηκε 10 φορές, με τον κώδικα που υλοποιεί το προτεινόμενο αλγόριθμο και τις παραμέτρους του Πίνακα 12, ανάλογα με το μέγεθος του προβλήματος. Όλα τα προβλήματα εκτελέστηκαν με 5 επαναληπτικά σετ, καθώς όπως ήδη αναφέρθηκε, η συγκεκριμένη παράμετρος διατηρήθηκε σταθερή κι εκτός πεδίου έρευνας από τη διαδικασία της εξόρυξης δεδομένων. Ο χρόνος κάθε μίας από τις 10 εκτελέσεις δεν παρουσίασε αξιόλογες αποκλίσεις, με δεδομένο ότι η εκτέλεση έγινε σε πολυδιεργασιακό λειτουργικό σύστημα, οπότε αναφέρουμε τη μέση τιμή όλων των εκτελέσεων κάθε προβλήματος, ως ένδειξη για τον χρόνο που απαιτήθηκε σε κάθε εκτέλεση.

Τα αποτελέσματα, που παρουσιάζονται αναλυτικά στους Πίνακες 13 έως 24, αναφέρουν το ελάχιστο, μέγιστο και μέσο makespan, το μέσο χρόνο, τη Best Known Solution (BKS), την απόκλιση του ελάχιστου, μέγιστου και μέσου makespan από τη BKS, τη λύση NEH και την απόκλιση της λύσης NEH από τη BKS και τέλος το ποσοστό της BKS

που αποτελεί η διαφορά μεταξύ μέγιστου και ελάχιστου makespan σε κάθε εκτέλεση. Να σημειωθεί ότι η απόκλιση είναι ποσοστιαία και υπολογίζεται από τον τύπο:

$$\omega = RC_{max}(\pi) = 100\frac{C_{max}(\pi) - C_{max}(\pi^*)}{C_{max}(\pi^*)}$$

όπου C_{max} το makespan, π η τρέχουσα λύση και π^* η βέλτιστη λύση (Nowicki et al., 2006).

α/α	BKS	MIN MS	MAX MS	AVER MS	TIME (sec)	NEH	ω NEH	ω MIN	ω MAX	ω AVER	ω MAX-MIN
1	1278	1278	1278	1278	1,044	1286	0,626	0,000	0,000	0,000	0,000
2	1359	1359	1359	1359	0,991	1365	0,442	0,000	0,000	0,000	0,000
3	1081	1081	1100	1087	1,015	1159	7,216	0,000	1,758	0,555	1,758
4	1293	1293	1298	1294,8	0,987	1325	2,475	0,000	0,387	0,139	0,387
5	1235	1235	1237	1235,5	1,058	1305	5,668	0,000	0,162	0,040	0,162
6	1195	1195	1210	1197	0,998	1228	2,762	0,000	1,255	0,167	1,255
7	1239	1239	1251	1245,5	0,956	1278	3,148	0,000	0,969	0,525	0,969
8	1206	1206	1206	1206	0,994	1223	1,410	0,000	0,000	0,000	0,000
9	1230	1230	1251	1232,1	1,002	1291	4,959	0,000	1,707	0,171	1,707
10	1108	1108	1108	1108	1,007	1151	3,881	0,000	0,000	0,000	0,000

Πίνακας 13: Εκτελέσεις προβλημάτων 1-10, διαστάσεων 20x5 με 850 επαναλήψεις και 75 λύσεις.

α/α	BKS	MIN MS	MAX MS	AVER MS	TIME (sec)	NEH	ω NEH	ω MIN	ω MAX	ω AVER	ω MAX-MIN
11	1582	1583	1589	1584,4	2,754	1680	6,195	0,063	0,442	0,152	0,379
12	1659	1660	1676	1668,9	2,711	1729	4,219	0,060	1,025	0,597	0,964
13	1496	1496	1513	1504,5	2,740	1557	4,078	0,000	1,136	0,568	1,136
14	1377	1378	1388	1381,8	2,702	1439	4,503	0,073	0,799	0,349	0,726
15	1419	1419	1427	1424,3	2,705	1502	5,849	0,000	0,564	0,374	0,564
16	1397	1400	1409	1404	2,718	1453	4,009	0,215	0,859	0,501	0,644
17	1484	1484	1489	1485,5	2,777	1562	5,256	0,000	0,337	0,101	0,337
18	1538	1545	1556	1549,5	2,755	1609	4,616	0,455	1,170	0,748	0,715
19	1593	1594	1612	1603,1	2,663	1647	3,390	0,063	1,193	0,634	1,130
20	1591	1591	1608	1603,1	2,741	1653	3,897	0,000	1,069	0,761	1,069

Πίνακας 14: Εκτελέσεις προβλημάτων 11-20, διαστάσεων 20x10 με 1000 επαναλήψεις και 83 λύσεις.

α/α	BKS	MIN MS	MAX MS	AVER MS	TIME (sec)	NEH	ω NEH	ω MIN	ω MAX	ω AVER	ω MAX-MIN
21	2297	2303	2311	2307,5	6,932	2410	4,919	0,261	0,609	0,457	0,348
22	2099	2101	2112	2108,6	6,891	2150	2,430	0,095	0,619	0,457	0,524
23	2326	2332	2343	2336,5	6,931	2411	3,654	0,258	0,731	0,451	0,473
24	2223	2223	2233	2227,1	6,875	2262	1,754	0,000	0,450	0,184	0,450
25	2291	2296	2305	2300,5	6,933	2397	4,627	0,218	0,611	0,415	0,393
26	2226	2226	2240	2233,4	6,932	2349	5,526	0,000	0,629	0,332	0,629
27	2273	2277	2295	2283,2	7,014	2362	3,916	0,176	0,968	0,449	0,792
28	2200	2205	2212	2209,3	6,883	2249	2,227	0,227	0,545	0,423	0,318
29	2237	2240	2246	2242,7	6,893	2320	3,710	0,134	0,402	0,255	0,268
30	2178	2179	2191	2184,7	6,883	2277	4,545	0,046	0,597	0,308	0,551

Πίνακας 15: Εκτελέσεις προβλημάτων 21-30, διαστάσεων 20x20 με 3000 επαναλήψεις και 35 λύσεις.

α/α	BKS	MIN MS	MAX MS	AVER MS	TIME (sec)	NEH	ω NEH	ω MIN	ω MAX	ω AVER	ω MAX-MIN
31	2724	2724	2724	2724	4,647	2733	0,330	0,000	0,000	0,000	0,000
32	2834	2838	2838	2838	4,468	2843	0,318	0,141	0,141	0,141	0,000
33	2621	2621	2622	2621,3	4,446	2640	0,725	0,000	0,038	0,011	0,038
34	2751	2753	2765	2759,5	4,468	2782	1,127	0,073	0,509	0,309	0,436
35	2863	2863	2864	2863,6	4,686	2868	0,175	0,000	0,035	0,021	0,035
36	2829	2829	2831	2829,7	4,451	2850	0,742	0,000	0,071	0,025	0,071
37	2725	2725	2746	2730,2	4,606	2758	1,211	0,000	0,771	0,191	0,771
38	2683	2683	2699	2688,2	4,513	2721	1,416	0,000	0,596	0,194	0,596
39	2552	2552	2561	2558,1	4,491	2576	0,940	0,000	0,353	0,239	0,353
40	2782	2782	2786	2782,9	4,378	2790	0,288	0,000	0,144	0,032	0,144

Πίνακας 16: Εκτελέσεις προβλημάτων 31-40, διαστάσεων 50x5 με 3000 επαναλήψεις και 45 λύσεις.

α/α	BKS	MIN MS	MAX MS	AVER MS	TIME (sec)	NEH	ω NEH	ω MIN	ω MAX	ω AVER	ω MAX-MIN
41	2991	3025	3050	3034,2	7,594	3135	4,814	1,137	1,973	1,444	0,836
42	2867	2911	2923	2914,8	7,776	3032	5,755	1,535	1,953	1,667	0,419
43	2839	2869	2904	2882,6	7,793	2986	5,178	1,057	2,290	1,536	1,233
44	3063	3071	3071	3071	7,657	3198	4,407	0,261	0,261	0,261	0,000
45	2976	3006	3025	3015,5	7,631	3160	6,183	1,008	1,647	1,327	0,638
46	3006	3041	3083	3069,9	7,787	3178	5,722	1,164	2,562	2,126	1,397
47	3093	3115	3145	3127,9	7,845	3277	5,949	0,711	1,681	1,128	0,970
48	3037	3042	3053	3049,8	7,831	3123	2,832	0,165	0,527	0,421	0,362
49	2897	2911	2932	2919,3	7,815	3002	3,624	0,483	1,208	0,770	0,725
50	3065	3110	3131	3128,9	7,786	3257	6,264	1,468	2,153	2,085	0,685

Πίνακας 17: Εκτελέσεις προβλημάτων 41-50, διαστάσεων 50x10 με 30000 επαναλήψεις και 4 λύσεις.

α/α	BKS	MIN MS	MAX MS	AVER MS	TIME (sec)	NEH	ω NEH	ω MIN	ω MAX	ω AVER	ω MAX-MIN
51	3850	3899	3930	3914,5	25,046	4082	6,026	1,273	2,078	1,597	0,805
52	3704	3747	3781	3761,8	25,028	3921	5,859	1,161	2,079	1,560	0,918
53	3640	3717	3758	3739	25,144	3927	7,885	2,115	3,242	2,720	1,126
54	3720	3780	3807	3792,9	25,193	3969	6,694	1,613	2,339	1,960	0,726
55	3610	3658	3699	3675,7	25,324	3835	6,233	1,330	2,465	1,820	1,136
56	3681	3726	3773	3750,5	25,103	3914	6,330	1,222	2,499	1,888	1,277
57	3704	3758	3788	3770,4	25,544	3952	6,695	1,458	2,268	1,793	0,810
58	3691	3773	3796	3782,6	25,463	3938	6,692	2,222	2,845	2,482	0,623
59	3743	3803	3833	3816,6	25,508	3952	5,584	1,603	2,404	1,966	0,801
60	3756	3804	3825	3818,3	25,586	4079	8,600	1,278	1,837	1,659	0,559

Πίνακας 18: Εκτελέσεις προβλημάτων 51-60, διαστάσεων 50x20 με 30000 επαναλήψεις και 6 λύσεις.

α/α	BKS	MIN MS	MAX MS	AVER MS	TIME (sec)	NEH	ω NEH	ω MIN	ω MAX	ω AVER	ω MAX-MIN
61	5493	5493	5514	5505,9	6,843	5519	0,473	0,000	0,382	0,235	0,382
62	5268	5284	5289	5284,5	6,937	5348	1,519	0,304	0,399	0,313	0,095
63	5175	5175	5175	5175	6,887	5219	0,850	0,000	0,000	0,000	0,000
64	5014	5023	5023	5023	6,929	5023	0,179	0,179	0,179	0,179	0,000
65	5250	5255	5255	5255	6,483	5266	0,305	0,095	0,095	0,095	0,000
66	5135	5135	5137	5135,3	6,483	5139	0,078	0,000	0,039	0,006	0,039
67	5246	5246	5259	5253,8	6,548	5259	0,248	0,000	0,248	0,149	0,248
68	5094	5094	5098	5094,6	6,301	5120	0,510	0,000	0,079	0,012	0,079
69	5448	5448	5454	5449,8	6,382	5489	0,753	0,000	0,110	0,033	0,110
70	5322	5324	5340	5328,4	6,455	5341	0,357	0,038	0,338	0,120	0,301

Πίνακας 19: Εκτελέσεις προβλημάτων 61-70, διαστάσεων 100x5 με 55000 επαναλήψεις και 2 λύσεις.

α/α	BKS	MIN MS	MAX MS	AVER MS	TIME (sec)	NEH	ω NEH	ω MIN	ω MAX	ω AVER	ω MAX-MIN
71	5770	5782	5800	5790,3	13,268	5846	1,317	0,208	0,520	0,352	0,312
72	5349	5362	5378	5364,6	12,934	5453	1,944	0,243	0,542	0,292	0,299
73	5676	5690	5712	5694	13,447	5824	2,607	0,247	0,634	0,317	0,388
74	5781	5814	5833	5824,8	13,112	5929	2,560	0,571	0,899	0,758	0,329
75	5467	5493	5525	5513,1	13,323	5679	3,878	0,476	1,061	0,843	0,585
76	5303	5310	5328	5318,9	13,256	5375	1,358	0,132	0,471	0,300	0,339
77	5595	5621	5663	5633,3	13,444	5704	1,948	0,465	1,215	0,685	0,751
78	5617	5637	5691	5666,2	13,476	5760	2,546	0,356	1,317	0,876	0,961
79	5871	5901	5944	5909,2	13,533	6032	2,742	0,511	1,243	0,651	0,732
80	5845	5860	5903	5894,3	13,569	5918	1,249	0,257	0,992	0,843	0,736

Πίνακας 20: Εκτελέσεις προβλημάτων 71-80, διαστάσεων 100x10 με 55000 επαναλήψεις και 2 λύσεις.

α/α	BKS	MIN MS	MAX MS	AVER MS	TIME (sec)	NEH	ω NEH	ω MIN	ω MAX	ω AVER	ω MAX-MIN
81	6202	6304	6379	6346,1	28,521	6541	5,466	1,645	2,854	2,323	1,209
82	6183	6284	6327	6303,6	28,630	6523	5,499	1,634	2,329	1,951	0,695
83	6271	6383	6442	6415,2	28,394	6639	5,868	1,786	2,727	2,299	0,941
84	6269	6369	6408	6384,3	28,306	6557	4,594	1,595	2,217	1,839	0,622
85	6314	6466	6496	6479,1	28,508	6695	6,034	2,407	2,882	2,615	0,475
86	6364	6487	6512	6496,6	28,518	6664	4,714	1,933	2,326	2,084	0,393
87	6268	6367	6403	6391,2	28,573	6632	5,807	1,579	2,154	1,966	0,574
88	6401	6534	6591	6549,8	28,407	6739	5,280	2,078	2,968	2,325	0,890
89	6275	6385	6420	6401,9	28,528	6677	6,406	1,753	2,311	2,022	0,558
90	6434	6553	6585	6569,9	28,770	6677	3,777	1,850	2,347	2,112	0,497

Πίνακας 21: Εκτελέσεις προβλημάτων 81-90, διαστάσεων 100x20 με 55000 επαναλήψεις και 2 λύσεις.

α/α	BKS	MIN MS	MAX MS	AVER MS	TIME (sec)	NEH	ω NEH	ω MIN	ω MAX	ω AVER	ω MAX-MIN
81	6202	6304	6379	6346,1	28,521	6541	5,466	1,645	2,854	2,323	1,209
82	6183	6284	6327	6303,6	28,630	6523	5,499	1,634	2,329	1,951	0,695
83	6271	6383	6442	6415,2	28,394	6639	5,868	1,786	2,727	2,299	0,941
84	6269	6369	6408	6384,3	28,306	6557	4,594	1,595	2,217	1,839	0,622
85	6314	6466	6496	6479,1	28,508	6695	6,034	2,407	2,882	2,615	0,475
86	6364	6487	6512	6496,6	28,518	6664	4,714	1,933	2,326	2,084	0,393
87	6268	6367	6403	6391,2	28,573	6632	5,807	1,579	2,154	1,966	0,574
88	6401	6534	6591	6549,8	28,407	6739	5,280	2,078	2,968	2,325	0,890
89	6275	6385	6420	6401,9	28,528	6677	6,406	1,753	2,311	2,022	0,558
90	6434	6553	6585	6569,9	28,770	6677	3,777	1,850	2,347	2,112	0,497

Πίνακας 21: Εκτελέσεις προβλημάτων 81-90, διαστάσεων 100x20 με 55000 επαναλήψεις και 2 λύσεις.

α/α	BKS	MIN MS	MAX MS	AVER MS	TIME (sec)	NEH	ω NEH	ω MIN	ω MAX	ω AVER	ω MAX-MIN
91	10862	10892	10921	10904,6	21,846	10942	0,737	0,276	0,543	0,392	0,267
92	10480	10539	10571	10552,1	21,181	10716	2,252	0,563	0,868	0,688	0,305
93	10922	11017	11017	11017	21,869	11025	0,943	0,870	0,870	0,870	0,000
94	10889	10893	10939	10914,5	21,680	11057	1,543	0,037	0,459	0,234	0,422
95	10524	10575	10575	10575	21,172	10645	1,150	0,485	0,485	0,485	0,000
96	10329	10337	10378	10366,6	21,258	10458	1,249	0,077	0,474	0,364	0,397
97	10854	10882	10943	10908,1	21,320	10989	1,244	0,258	0,820	0,498	0,562
98	10730	10798	10828	10806,9	21,510	10829	0,923	0,634	0,913	0,717	0,280
99	10438	10473	10486	10476,1	21,246	10574	1,303	0,335	0,460	0,365	0,125
100	10675	10727	10758	10733,2	21,616	10807	1,237	0,487	0,778	0,545	0,290

Πίνακας 22: Εκτελέσεις προβλημάτων 91-100, διαστάσεων 200x10 με 90000 επαναλήψεις και 1 λύση.

α/α	BKS	MIN MS	MAX MS	AVER MS	TIME (sec)	NEH	ω NEH	ω MIN	ω MAX	ω AVER	ω MAX-MIN
101	11195	11368	11454	11410,5	47,123	11594	3,564	1,545	2,314	1,925	0,768
102	11203	11410	11492	11448,2	45,748	11675	4,213	1,848	2,580	2,189	0,732
103	11281	11543	11623	11574,3	46,828	11852	5,062	2,322	3,032	2,600	0,709
104	11275	11437	11561	11510,2	46,329	11803	4,683	1,437	2,537	2,086	1,100
105	11259	11390	11475	11430,6	46,108	11685	3,784	1,164	1,918	1,524	0,755
106	11176	11370	11470	11405,8	46,716	11629	4,053	1,736	2,631	2,056	0,895
107	11360	11498	11590	11543,5	46,362	11833	4,164	1,215	2,025	1,615	0,810
108	11334	11516	11590	11556,6	46,573	11913	5,109	1,606	2,259	1,964	0,653
109	11192	11417	11525	11467,6	46,187	11673	4,298	2,010	2,975	2,462	0,965
110	11288	11479	11603	11543,9	46,627	11869	5,147	1,692	2,791	2,267	1,099

Πίνακας 23: Εκτελέσεις προβλημάτων 101-110, διαστάσεων 200x20 με 90000 επαναλήψεις και 1 λύση.

α/α	BKS	MIN MS	MAX MS	AVER MS	TIME (sec)	NEH	ω NEH	ω MIN	ω MAX	ω AVER	ω MAX-MIN
111	26059	26297	26369	26325,8	230,023	26670	2,345	0,913	1,190	1,024	0,276
112	26520	26824	26909	26865,1	228,576	27232	2,685	1,146	1,467	1,301	0,321
113	26371	26682	26803	26760,3	228,488	27060	2,613	1,179	1,638	1,476	0,459
114	26456	26677	26747	26721,9	228,557	27055	2,264	0,835	1,100	1,005	0,265
115	26334	26506	26594	26546,3	229,109	26727	1,492	0,653	0,987	0,806	0,334
116	26477	26666	26731	26704	229,427	26992	1,945	0,714	0,959	0,857	0,245
117	26389	26608	26658	26624,1	228,770	26797	1,546	0,830	1,019	0,891	0,189
118	26560	26796	26872	26827,9	228,678	27138	2,176	0,889	1,175	1,009	0,286
119	26005	26260	26381	26328,4	228,880	26631	2,407	0,981	1,446	1,244	0,465
120	26457	26658	26737	26701,3	227,699	26984	1,992	0,760	1,058	0,923	0,299

Πίνακας 24: Εκτελέσεις προβλημάτων 111-120, διαστάσεων 500x20 με 90000 επαναλήψεις και 2 λύσεις.

5.2 Σχολιασμός αποτελεσμάτων

Στην ενότητα αυτή θα αναλύσουμε τα αποτελέσματα που παραθέσαμε στην προηγούμενη ενότητα, ως προς την ποιότητα των λύσεων. Στόχος μας να διαπιστώσουμε εάν πράγματι τα αποτελέσματα αυτά είναι πολύ καλά και πολύ γρήγορα, όπως επιδιώκαμε.

Ξεκινώντας από το χρόνο, όλες οι εκτελέσεις λαμβάνουν τη διάκριση VF (Very Fast), σύμφωνα με τις περίπου 8500 εκτελέσεις που κάναμε στο κεφάλαιο 4. Σαφέστατα δεν πρόκειται για τις απόλυτα πιο σύντομες εκτελέσεις κάθε προβλήματος, αλλά δεν ήταν εξ αρχής αυτό το ζητούμενο. Το ζητούμενο ήταν να λάβουμε πολύ καλά αποτελέσματα, ως προς το makespan, σε σύντομο χρόνο. Μπορεί να είχαμε λ.χ. εκτελέσεις που έδωσαν αντίστοιχα αποτελέσματα στο κεφάλαιο 4 ως προς το makespan, στο ένα δέκατο του χρόνου των εκτελέσεων με τις τελικές ρυθμίσεις. Όμως, ο αλγόριθμός μας

έχει στοχαστικά στοιχεία. Μπορεί η πρώτη τυχαία σειρά εργασιών να πιάσει –ή και να ξεπεράσει- τη BKS, σε χρόνο ενός εκατοστού του δευτερολέπτου. Αυτό όμως δεν έχει συνέχεια και συνέπεια. Οι τελικές ρυθμίσεις, στόχευαν στο να έχουμε αποτελέσματα που να είναι συνεπώς καλά, σε κάθε εκτέλεση.

Ως προς το makespan, η μικρότερη τιμή σε κάθε απόκλιση είναι εξαιρετικό αποτέλεσμα ως προς την απόκλιση που έχει από τη BKS κάθε προβλήματος. Σε 30 από τα 120 προβλήματα ισοφάρισε την BKS (0% απόκλιση), ενώ συνολικά στα 120 προβλήματα η μέση απόκλιση ήταν 0,657%, η δε μέγιστη στο 2,407%. Να σημειωθεί ότι απόκλιση άνω του 2% είχαμε μόλις σε 6 προβλήματα, ενώ απόκλιση άνω του 1% είχαμε σε 38 προβλήματα. Δηλαδή σε 82 από τα 120 προβλήματα, η απόκλιση ήταν το πολύ 1%. Με βάση τα 56 προβλήματα τα οποία είχαμε εκτελέσει στο 4^{ο} κεφάλαιο, στα 55 οι καλύτερες λύσεις θα λάμβαναν το χαρακτηρισμό VGVF και σε ένα το χαρακτηρισμό GVF.

Κάθε 10άδα εκτελέσεων, φυσικά έδωσε και μία μέγιστη τιμή makespan, τη χειρότερη. Η τιμή αυτή μας ενδιαφέρει γιατί από αυτή βλέπουμε πόσο συνεπή είναι τα αποτελέσματα. Σε 6 προβλήματα η μέγιστη τιμή ήταν ίση με την BKS, δηλαδή ο αλγόριθμος έβρισκε σε κάθε εκτέλεση το βέλτιστο. Στο σύνολο των προβλημάτων η μέση απόκλιση του μέγιστου makespan από την BKS ήταν 1,197%, η δε μέγιστη στο 3,241%. Να σημειωθεί ότι απόκλιση άνω του 3% είχαμε μόλις σε 2 προβλήματα, ενώ απόκλιση άνω του 2% είχαμε σε 31 προβλήματα. Δηλαδή σε 89 από τα 120 προβλήματα, η μέγιστη απόκλιση που πήραμε από εκτέλεση, ήταν το πολύ 2%. Με βάση τα 56 προβλήματα που είχαμε εκτελέσει στο 4^{ο} κεφάλαιο, το μέγιστο makespan 16 προβλημάτων θα λάμβανε το χαρακτηρισμό VGVF, σε 26 προβλήματα θα ήταν GVF και σε 13 προβλήματα θα ήταν BADVF.

Εκ πρώτοις η διαπίστωση ότι σε 13 από τα 56 προβλήματα θα λαμβάναμε λύση BADVF (για τη χειρότερη πάντα εκτέλεση), δεν είναι πολύ καλό στοιχείο. Όμως, με μία προσεκτική ματιά στα προβλήματα που εντοπίζουμε τις BADVF λύσεις, θα δούμε ότι με εξαίρεση δύο προβλήματα που η απόκλιση είναι περίπου στο 1,7%, στις υπόλοιπες περιπτώσεις η απόκλιση από τη BKS είναι στο 1% ή και μικρότερη, αφορά δε κατά πλειονότητα μικρών διαστάσεων προβλήματα. Πρόκειται δηλαδή για εκτελέσεις που ακόμα και μία ελάχιστη απόκλιση από τη BKS (ακόμα και μερικές μονάδες), αρκεί για να χαρακτηριστεί η λύση ως «κακή», χωρίς όμως πρακτικά να έχει σημαντική απόκλιση.

Χαρακτηριστική είναι η περίπτωση του προβλήματος 5, όπου η BKS είναι 1235 και εκτέλεση με το μέγιστο makespan στο 1237, δηλαδή μόλις 2 μονάδες μεγαλύτερη ή 0,16%. Αυτή όμως η απόκλιση, με βάση τα αποτελέσματα που λάβαμε από τις αρχικές εκτελέσεις, κατατάσσει τη λύση με το 1237 στις «κακές».

Συμπερασματικά, λοιπόν, μπορούμε να πούμε ότι ο αλγόριθμος όχι μόνο έδωσε πολύ καλές βέλτιστες λύσεις, αλλά και οι κακές λύσεις του, είναι καλές και κοντά στις καλύτερες. Είναι χαρακτηριστικό ότι η μέγιστη απόκλιση ως προς τη BKS της διαφοράς μέγιστης και ελάχιστης λύσης σε πρόβλημα είναι 1,75%, ενώ μόλις σε 14 προβλήματα αυτή η απόκλιση είναι άνω του 1%.

Με αυτήν την εικόνα ως προς τις καλύτερες και χειρότερες λύσεις, η κατάσταση στη μέση λύση ανά πρόβλημα διαπιστώνεται επίσης πολύ καλή. Η μέση απόκλιση είναι στο 0,904%, ενώ η μέγιστη στο 2,719%, με μόλις 17% προβλήματα να ξεπερνούν το 2%. 38 από τα 56 προβλήματα έχουν μέση λύση με χαρακτηρισμό VGVF, 14 μέση λύση με χαρακτηρισμό GVF και 3 με χαρακτηρισμό BADVF. Και στα 3 όμως αυτά προβλήματα με BADVF, η απόκλιση από τη BKS είναι 0,55% ή μικρότερη.

Στον Πίνακα 25 παρουσιάζονται οι μέσες αποκλίσεις από τη BKS για την ελάχιστη, μέγιστη και μέση λύση, σε κάθε δεκάδα ομοειδών προβλημάτων.

SET	ω MIN	ω MAX	ω AVER	TIME AVER
20x5	0,000	0,624	0,160	1,005
20x10	0,093	0,859	0,478	2,727
20x20	0,142	0,616	0,373	6,916
50x5	0,021	0,266	0,116	4,515
50x10	0,899	1,625	1,277	7,751
50x20	1,527	2,406	1,944	25,294
100x5	0,062	0,187	0,114	6,625
100x10	0,346	0,890	0,592	13,336
100x20	1,826	2,511	2,154	28,515
200x10	0,402	0,667	0,516	21,470
200x20	1,657	2,506	2,069	46,460
500x20	0,890	1,204	1,054	228,821
ΣΥΝΟΛΟ	0,657	1,197	0,905	32,786

Πίνακας 25: Αποκλίσεις ελάχιστης, μέγιστης και μέσης λύσης ανά ομάδα προβλημάτων.

Όπως παρατηρούμε στον Πίνακα 25, οι σημαντικότερες αποκλίσεις παρουσιάζονται στα σετ 50x20, 100x20 και 200x20. Παρατηρούμε ότι πρόκειται για προβλήματα με 20 μηχανές[9]. Όμως, τα συγκεκριμένα 30 προβλήματα είναι τα δυσκολότερα από τα δυσκολότερα του σετ του Taillard, όπως επιβεβαιώνεται και από τη βιβλιογραφία (Nowicki et al., 2006), οπότε δεν αποτελεί έκπληξη το γεγονός ότι σε αυτά τα 30 προβλήματα παρουσιάζονται οι μεγαλύτερες αποκλίσεις.

5.3 Σύγκριση με άλλη παραμετροποίηση

Τα αποτελέσματα που είδαμε στην προηγούμενη ενότητα είναι καλά. Τα αποτελέσματα αυτά βασίστηκαν στην εύρεση της κατάλληλης παραμετροποίησης για κάθε ομάδα προβλημάτων του Taillard, ανάλογα με τη διάστασή της.

Εάν δεν κάναμε παραμετροποίηση ανά ομάδα προβλημάτων –με βάση τη διαδικασία του data mining- αλλά χρησιμοποιούσαμε σταθερή παραμετροποίηση σε όλα τα προβλήματα, τι αποτελέσματα θα λαμβάναμε; Για να απαντήσουμε στο ερώτημα αυτό αναζητήσαμε τις παραμέτρους με τα καλύτερα χαρακτηριστικά με βάση τα μοντέλα του Δέντρου Αποφάσεων και του Νευρωνικού Δικτύου. Καταλήξαμε ότι η παραμετροποίηση 25000 επαναλήψεις και 2 λύσεις εμφανίζει από τις υψηλότερες τιμές ως καλός συνδυασμός και στα δύο μοντέλα. Έτσι, εκτελέσαμε όλα τα προβλήματα του Taillard και με το συνδυασμό αυτό.

Στον Πίνακα 26 καταγράφουμε τι διαφορά στις αποκλίσεις μεταξύ των εκτελέσεων της ενότητας 5.2 και των εκτελέσεων με παραμετροποίηση 25000/2, για τη μέγιστη, ελάχιστη και μέση τιμή του makespan. Θετική τιμή στις αποκλίσεις σημαίνει ότι η παραμετροποίηση της παραγράφου 5.2 έδωσε καλύτερα αποτελέσματα, ενώ αρνητική τιμή σημαίνει ότι η παραμετροποίηση 25000/2 έδωσε καλύτερα αποτελέσματα[10].

Στον Πίνακα 26 διαπιστώνουμε ότι μόνο σε 8 προβλήματα από τα 120, η παραμετροποίηση 25000/2 έδωσε καλύτερα αποτελέσματα (μικρότερο ελάχιστο makespan) από την παραμετροποίηση που επιλέξαμε ως καταλληλότερη για κάθε ομάδα προβλημάτων. Ως προς το μέσο makespan, σε 12 προβλήματα η παραμετροποίηση 25000/2 έδωσε μικρότερο, ενώ ως προς το maximum makespan, η παραμετροποίηση 25000/2 έδωσε μικρότερο σε 24 από τα 120 προβλήματα.

[9] Αυξημένη απόκλιση σε σχέση με τα 20x5 και 20x10, εμφανίζει και το 20x20, κι αυτό με 20 μηχανές (είναι 50% μεγαλύτερη από το 20x10), αλλά σε απόλυτο μέγεθος δεν είναι μεγάλη.
[10] Για ευκολία στην ανάγνωση, οι αρνητικές τιμές στον Πίνακα 26 είναι σκιασμένες.

α/α	ω MAX	ω MIN	ω AVER	α/α	ω MAX	ω MIN	ω AVER	α/α	ω MAX	ω MIN	ω AVER
1	0,000	0,000	0,000	41	0,435	0,334	0,308	81	0,048	0,500	0,353
2	0,000	0,000	0,000	42	0,000	0,000	0,042	82	0,307	0,226	0,301
3	0,000	1,758	1,203	43	0,000	0,211	0,324	83	0,223	0,367	0,277
4	-0,232	0,000	-0,108	44	0,000	0,000	0,000	84	0,271	0,000	0,212
5	0,567	0,729	0,688	45	0,067	0,168	0,091	85	0,206	-0,032	0,239
6	-1,255	0,000	-0,167	46	0,000	0,665	0,170	86	0,063	0,157	0,148
7	0,000	0,969	0,444	47	-0,162	0,744	0,339	87	0,558	0,463	0,450
8	0,249	0,000	0,050	48	0,033	0,165	0,020	88	0,281	0,234	0,441
9	0,163	1,870	1,699	49	0,000	0,035	0,128	89	0,382	0,494	0,406
10	0,000	0,000	0,000	50	0,000	0,685	0,069	90	0,528	0,389	0,465
11	0,632	0,190	0,468	51	0,130	0,208	0,184	91	-0,009	0,129	0,074
12	0,241	0,723	0,380	52	0,540	1,053	0,891	92	0,191	0,076	0,140
13	0,401	0,267	0,114	53	0,192	0,742	0,473	93	0,000	0,000	0,000
14	0,290	0,073	0,203	54	0,188	0,403	0,309	94	0,000	0,000	0,118
15	1,198	0,775	0,923	55	0,222	-0,055	0,030	95	0,000	0,000	0,000
16	-0,573	-0,215	-0,444	56	-0,081	0,435	0,193	96	0,029	0,368	0,105
17	2,089	2,022	2,217	57	0,189	0,108	0,292	97	-0,083	0,240	0,100
18	1,040	-0,065	0,592	58	0,867	0,704	0,650	98	0,000	0,000	0,051
19	0,314	-0,063	0,082	59	-0,027	-0,027	0,102	99	-0,077	0,000	-0,001
20	0,566	0,880	0,409	60	0,453	0,426	0,248	100	0,000	0,000	0,174
21	0,522	0,305	0,292	61	-0,164	0,036	-0,053	101	-0,054	0,339	0,083
22	0,524	-0,095	0,324	62	0,000	0,000	0,028	102	0,116	0,509	0,325
23	1,247	0,258	0,503	63	0,541	0,000	0,108	103	-0,089	0,204	0,182
24	0,000	0,450	0,265	64	0,000	0,000	0,000	104	0,035	0,532	0,296
25	2,052	1,746	2,003	65	0,000	0,000	0,000	105	-0,036	0,275	0,099
26	1,572	0,270	0,773	66	-0,039	0,000	-0,006	106	-0,116	0,546	0,370
27	0,440	0,440	0,607	67	0,000	0,000	0,004	107	0,167	0,704	0,397
28	0,636	0,318	0,323	68	0,000	0,000	0,004	108	0,203	0,362	0,274
29	1,162	0,089	0,434	69	0,000	0,000	0,029	109	-0,098	0,465	0,232
30	0,413	0,184	0,363	70	-0,225	0,075	-0,008	110	-0,009	0,682	0,309
31	0,184	0,184	0,184	71	0,000	0,156	0,109	111	0,161	0,157	0,253
32	0,000	-0,071	-0,014	72	-0,224	0,000	-0,022	112	0,204	0,332	0,255
33	-0,038	0,000	-0,011	73	0,088	0,000	0,099	113	0,258	0,383	0,257
34	-0,109	0,327	0,091	74	0,086	0,208	0,118	114	0,333	0,359	0,270
35	0,000	0,000	0,007	75	-0,018	0,366	0,104	115	0,095	0,258	0,219
36	0,000	0,000	-0,011	76	-0,038	0,000	0,017	116	0,555	0,476	0,438
37	0,183	0,000	0,572	77	0,107	0,465	0,424	117	0,019	0,034	0,093
38	-0,335	0,000	-0,075	78	0,089	0,498	0,331	118	0,297	0,282	0,290
39	0,000	0,353	0,114	79	-0,273	0,204	0,167	119	0,077	0,296	0,166
40	0,108	0,252	0,219	80	0,000	0,359	0,074	120	0,185	0,265	0,222

Πίνακας 26: Αποκλίσεις «βέλτιστης» παραμετροποίησης με παραμετροποίηση 25000/2.

Είναι σημαντικό να παρατηρήσουμε στον Πίνακα 26 ότι οι καλύτερες επιδόσεις του συνδυασμού 25000/2, δεν βρίσκονται μαζεμένες σε κάποια ομάδα προβλημάτων, αλλά είναι διάσπαρτες. Αυτό σημαίνει ότι το 25000/2 δεν αποτελεί ξεκάθαρα καλύτερη επιλογή από τους συνδυασμούς που επιλέξαμε ειδικά για κάθε ομάδα, με βάση την εξόρυξη δεδομένων. Στο ελάχιστο makespan η επιλογή 25000/2 είναι κατά μέσο όρο 0,29% χειρότερη από τις επιμέρους ρυθμίσεις, με μέγιστο ποσοστό απόκλισης το 2,022%. Αντίστοιχη εικόνα έχουμε και στη μέση τιμή, με 0,259% και 2,217% αντίστοιχα καθώς και στη μέγιστη τιμή με 0,187% και 2,089% αντίστοιχα.

Στον Πίνακα 27 βλέπουμε τους χρόνους των δύο διαφορετικών παραμετροποιήσεων. Η παραμετροποίηση 25000/2 είναι από 50% έως 200% πιο γρήγορη σε όλα τα προβλήματα.

SET	ΕΠΙΜΕΡΟΥΣ ΠΑΡΑΜΕΤΡΟΙ	ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗ 25000/2
20x5	1,005	0,738
20x10	2,727	1,503
20x20	6,916	3,178
50x5	4,515	1,593
50x10	7,751	3,170
50x20	25,294	6,958
100x5	6,625	3,077
100x10	13,336	6,153
100x20	28,515	13,258
200x10	21,470	12,587
200x20	46,460	26,541
500x20	228,821	82,129
ΣΥΝΟΛΟ	32,786	13,407

Πίνακας 27: Μέσοι χρόνοι εκτέλεσης ανά ομάδα προβλημάτων.

5.4 Δοκιμασία ακριβούς μικρο-προσαρμογής

Εκτελέσαμε τα προβλήματα με συγκεκριμένη παραμετροποίηση για κάθε ομάδα, την οποία εκτιμήσαμε ως καταλληλότερη με βάση τα μοντέλα από την εξόρυξη δεδομένων. Είναι όμως πράγματι καλύτερη η συγκεκριμένη παραμετροποίηση σε σχέση με μικρές διαφοροποιήσεις αυτής;

Για να το δοκιμάσουμε αυτό, εστιάσαμε σε ένα συγκεκριμένο πρόβλημα, το 51^{o}, το οποίο είναι 50x20 και ανήκει σε μία από τις τρεις δεκάδες με τα δύσκολα προβλήματα, σύμφωνα με τη βιβλιογραφία (Nowicki et al., 2004). Σε αυτό το πρόβλημα επιλέξαμε

την παραμετροποίηση 30000/6, δηλαδή 30000 επαναλήψεις με 6 λύσεις. Πώς συγκρίνεται τα αποτελέσματα αυτής της παραμετροποίησης σε σχέση με γειτονικές της, δηλαδή τις πολύ κοντινές 31000/6 και 29000/6 αλλά και τις λίγο πιο μακρινές 25000/6 και 35000/6; Δηλαδή αν μεταβάλουμε τις επαναλήψεις κατά +/- 1000 και κατά +/- 5000, πόσο επηρεάζει αυτό την ποιότητα των λύσεων;

Εκτελέσαμε το πρόβλημα 51 από 102 φορές με τους συνδυασμούς 30000/6, 25000/6, 29000/6, 31000/6 και 35000/6 . Τα αποτελέσματα εμφανίζονται στον Πίνακα 28 και δείχνουν ότι η παραμετροποίηση 30000/6 δίνει καλύτερα αποτελέσματα, σε σχέση με τις άλλες τέσσερις, και ως προς το καλύτερο makespan, και ως προς χειρότερο makespan αλλά και ως προς τη μέση τιμή στο σύνολο των εκτελέσεων. Άρα, από αυτήν την ενδελεχή δοκιμασία στο συγκεκριμένο πρόβλημα, φαίνεται ότι η παραμετροποίηση που προέκυψε από τα μοντέλα μας είναι καλύτερη από ελαφρά διαφοροποιημένες παραμετροποιήσεις στην περιοχή της.

	25000/6	29000/6	30000/6	31000/6	35000/6
MIN	3896	3897	3893	3897	3894
MAX	3941	3938	3931	3934	3932
AVERAGE	3917,637	3915,461	3913,608	3914,598	3914,716

Πίνακας 28: Σύγκριση όμορων παραμετροποιήσεων για το πρόβλημα 51.

Παρατηρώντας την τιμή του μέσου, που έχει και τη μεγαλύτερη σημασία, διαπιστώνουμε ότι μειώνεται έντονα από τις 25000 προς τις 30000 επαναλήψεις, όπου εντοπίζουμε ελάχιστο και μετά αυξάνει αλλά κινείται σταθεροποιητικά προς τις 35000 επαναλήψεις. Ο λόγος που παρατηρούμε αυτή τη σταθεροποιητική κίνηση όσο αυξάνουμε τις επαναλήψεις είναι ότι με περισσότερες επαναλήψεις δεν βλάπτουμε ουσιωδώς την ποιότητα της λύσης. Όμως, πέραν του ότι χάνουμε σε χρόνο εκτέλεσης, με τις περισσότερες επαναλήψεις απομακρυνόμαστε από την περιοχή της λύσης με την οποία κάθε φορά ξεκινάμε την επαναληπτική διαδικασία. Έτσι, μπορεί μεν να οδηγηθούμε σε κάποια άλλη περιοχή με μικρότερο makespan, αλλά δεν έχουμε διερευνήσει επαρκώς την περιοχή της λύσης που ξεκινήσαμε. Συνεπώς, η αύξηση του αριθμού των επαναλήψεων από ένα σημείο και μετά θα αλλάξει τη συμπεριφορά του αλγορίθμου. Βέβαια, όπως προαναφέραμε, πρώτα από όλα θα αυξηθεί ο χρόνος, οπότε θα βγουν οι εκτελέσεις από την περιοχή των πολύ γρήγορων λύσεων (VF) ή όποιο άλλο όριο χρόνου τεθεί.

Για το λόγο αυτό και δεν επιχειρήσαμε εκτελέσεις με μικρότερο και μεγαλύτερο αριθμό λύσεων. Η χρήση περισσότερων λύσεων δεν θα βλάψει την ποιότητα του αποτελέσματος, ενδεχομένως δε να το βελτιώσει. Όμως, θα αυξήσει το χρόνο εκτέλεσης. Και πρέπει πάντα να έχουμε υπόψη μας ότι αυτό που ζητήσαμε από τα μοντέλα κατά την εξόρυξη δεδομένων, δεν είναι οι συνδυασμοί για τις απολύτως καλύτερες λύσεις, τις οποίες εύλογα θα παρήγαγαν με άπειρο αριθμό επαναλήψεων και άπειρο αριθμό λύσεων, αλλά τους συνδυασμούς που μας δίνουν πολύ καλές λύσεις σε πολύ καλό χρόνο και όπου παραπάνω επαναλήψεις και παραπάνω λύσεις, δεν θα προσέθεταν μεγάλη ωφέλεια, σε σχέση που τον επιπλέον χρόνο που αυτές θα απαιτούσαν. Αντίστοιχα, με λιγότερες επαναλήψεις και λιγότερες λύσεις θα είχαμε μεν καλύτερο χρόνο εκτέλεσης, αλλά όχι λύσεις στο επίπεδο ποιότητας που επιθυμούμε.

Να σημειώσουμε ότι οι εκτελέσεις αυτές έδωσαν για το πρόβλημα 51 μικρότερη τιμή ελάχιστου makespan από αυτή που έχουμε καταγεγραμμένη στον Πίνακα 18 (3899) όπως και μεγαλύτερη τιμή από το 3930 ως χειρότερη λύση. Τέτοιες διαφοροποιήσεις είναι αναμενόμενες από τον αλγόριθμό μας, αφού ο Πίνακας 18 (και οι υπόλοιποι με τα αναλυτικά αποτελέσματα) έχει δημιουργηθεί με 10 εκτελέσεις, ενώ για την ενδελεχή μελέτη του micro-tuning, κάναμε 102 εκτελέσεις σε κάθε συνδυασμό. Από τις εκτελέσεις αυτές, με βάση και τη στοχαστική φύση του αλγορίθμου, ήταν αναμενόμενο να προκύψουν οριακά καλύτερες και οριακά χειρότερες ακραίες λύσεις. Ελαφρά διαφοροποιημένη εμφανίζεται και η μέση τιμή (3913 έναντι 3914), κάτι που επίσης ερμηνεύεται από το μεγαλύτερο αριθμό εκτελέσεων με τον οποίο προέκυψε ο Πίνακας 28 σε σχέση με τον Πίνακα 18.

5.5 Ολοκληρώνοντας με τα αποτελέσματα

Έχοντας κατασκευάσει τον αλγόριθμο, και παρατηρώντας ότι η παραμετροποίησή του επηρεάζει σημαντικά την αποδοτικότητά του, αναζητήσαμε διαδικασία για να εντοπίσουμε την καλύτερη παραμετροποίηση. Γι’ αυτό καταφύγαμε στην εξόρυξη δεδομένων, κάνοντας τη διαδικασία που περιγράψαμε στο κεφάλαιο 4, προσπαθώντας από πολλαπλές εκτελέσεις να εντοπίσουμε τους καλύτερους συνδυασμούς για κάθε ομάδα προβλημάτων.

Στην αρχή του κεφαλαίου 4 θέσαμε δύο καίρια ερωτήματα, με τα οποία ξεκινήσαμε τη διαδικασία του data mining και τα οποία και θυμίζουμε:

1. Με ποιους συνδυασμούς παραμέτρων λαμβάνουμε καλύτερες λύσεις σε συντομότερο χρόνο;
2. Οι καλύτεροι συνδυασμοί είναι ίδιοι για όλα τα προβλήματα του Taillard;

Ως προς το δεύτερο ερώτημα, οι αρχικές μας ενδείξεις επιβεβαιώθηκαν και με τις μετέπειτα μετρήσεις, καθώς δεν υπάρχει καμία ένδειξη που να επιβεβαιώνει ότι ένας μοναδικός συνδυασμός αριθμού επαναλήψεων και αριθμού λύσεων θα αποδίδει καλύτερα σε **όλα** τα προβλήματα από οποιονδήποτε άλλο συνδυασμό. Τα προβλήματα με περισσότερες εργασίες χρειάζονται περισσότερες επαναλήψεις για να διερευνηθεί επαρκώς η γειτονιά της λύσης εκκίνησης. Από την άλλη τα προβλήματα με λιγότερες εργασίες και με λιγότερες επαναλήψεις έχουμε επαρκέστερη κάλυψη της γειτονιάς, όμως η χρήση πολλαπλών λύσεων ευνοεί τον ταχύτερο και αποτελεσματικότερο απεγκλωβισμό από τοπικά ελάχιστα και εύρεση ενός καλού ελαχίστου (που σε πολλά από αυτά φαίνεται να είναι το ολικό βέλτιστο). Καθώς στα μικρά προβλήματα η αφετηρία μας (από τον ΝΕΗ) είναι συχνά πολύ κοντά στη βέλτιστη λύση, ο εγκλωβισμός του προτεινόμενου αλγόριθμου σε κάποιο τοπικό ελάχιστο είναι συχνός.

Ως προς το πρώτο ερώτημα, ο στόχος φαίνεται να επετεύχθη, τουλάχιστον με την έννοια ότι βρήκαμε παραμέτρους για να λαμβάνουμε από κάθε ομάδα προβλημάτων πολύ καλές λύσεις, σε σύντομο χρόνο και με ικανοποιητική συνέπεια. Το τελευταίο, το ζήτημα της συνέπειας δηλαδή, είναι πολύ σημαντικό, καθώς δεν μας αρκεί να βρούμε σε μία εκτέλεση μία καλή λύση, αλλά θα πρέπει σε όλες τις εκτελέσεις να λαμβάνουμε καλές λύσεις, ή τουλάχιστον να μη λαμβάνουμε κακές. Η ανάλυση των αποτελεσμάτων έδειξε ότι ακόμα και οι χειρότερες λύσεις που λάβαμε με τις παραμέτρους που χρησιμοποιήσαμε σε κάθε ομάδα προβλημάτων, ήταν ικανοποιητικά καλές, με την έννοια ότι είτε δεν είχαν σημαντικές αποκλίσεις από τις καλύτερες λύσεις, είτε δεν είχαν σημαντικές αποκλίσεις από τη BKS.

Επιλέγοντας ένα μέσο συνδυασμό, τον 25000/2 για όλα τα προβλήματα, διαπιστώσαμε ότι με συνέπεια δεν απέδωσε πουθενά καλύτερα από τις εξατομικευμένες παραμέτρους. Έδωσε μόνο περιστασιακά και σκόρπια καλύτερες λύσεις που δικαιολογούνται από τη στοχαστικότητα του αλγορίθμου, από την καλή επιλογή του μέσου συνδυασμού (υποδείχθηκε από τα μοντέλα μας ως υποσχόμενος) και από το γεγονός ότι ο αλγόριθμός μας εν γένει είναι αποδοτικός.

Φυσικά υπάρχει και η παράμετρος του χρόνου. Ο συνδυασμός 25000/2 είναι από 50% έως 200% ταχύτερος από τις εξατομικευμένες παραμέτρους. Δεν είναι σημαντικό αυτό; Προφανώς και είναι, αλλά όσο σημαντικό κρίνεται από κάθε εγκατάσταση. Όπως προαναφέραμε, εάν η εκτέλεση γίνεται για να ρυθμιστεί μία γραμμή παραγωγής για τους επόμενους 12 μήνες, τότε μπορούν να θυσιαστούν λεπτά, ώρες ή και ημέρες εκτέλεσης για να βρούμε την καλύτερη δυνατή λύση σε ένα πρόβλημα. Αν όμως η εκτέλεση γίνεται για τη ρύθμιση των μηχανών το επόμενο λεπτό, προφανώς ο χρόνος εκτέλεσης του αλγορίθμου πρέπει να είναι της τάξης το πολύ μερικών δευτερολέπτων. Άρα, ένας συνδυασμός όπως ο 25000/2 είναι ταχύς, δεν θα απογοητεύσει με τα αποτελέσματά του, αλλά υπάρχουν συνδυασμοί που δίνουν καλύτερα αποτελέσματα χωρίς να κοστίζουν πολύ περισσότερο σε χρόνο.

Από την άλλη πλευρά, ο ένας και μοναδικός συνδυασμός, μπορεί να αποβεί καταστροφικός. Για παράδειγμα, εκτός του 25000/2 κάναμε εκτελέσεις με το συνδυασμό 1000/75, που όπως φαίνεται από τον Πίνακα 12, είναι πολύ κοντά στο συνδυασμό 850/75 με τον οποίο εκτελέσαμε τα προβλήματα 20x5. Τι έδειξαν οι εκτελέσεις με τον συνδυασμό 1000/75; Καταρχήν είχαν από 30% έως 100% μεγαλύτερο χρόνο εκτέλεσης σε σχέση με τις εξατομικευμένες παραμέτρους. Ως προς την ποιότητα των λύσεων, με εξαίρεση τα προβλήματα διάστασης 20x5, όπου σε 4 προβλήματα έδωσε καλύτερη μέγιστη τιμή για το makespan και σε 3 καλύτερη μέση τιμή για το makespan (σε σχέση με τις εξατομικευμένες παραμέτρους), στα υπόλοιπα προβλήματα οι καλύτερες λύσεις ήταν σποραδικές και σπάνιες. Άρα, ακόμα κι εάν βασιστούμε σε έναν συνδυασμό για όλα τα προβλήματα, και γι' αυτόν τον συνδυασμό πρέπει να κάνουμε ενδελεχή μελέτη των δεδομένων, ώστε να τον επιλέξουμε. Αλλιώς κινδυνεύουμε να χρησιμοποιήσουμε ένα συνδυασμό που δεν θα φέρει συνολικά καλά αποτελέσματα.

Αντίστοιχη επιβεβαίωση ως προς την καταλληλότητα των παραμέτρων από την εξόρυξη δεδομένων λάβαμε και από τον έλεγχο μικρο-προσαρμογής. Η παραμετροποίηση από την εξόρυξη δεδομένων, για ένα συγκεκριμένο πρόβλημα που επιλέξαμε να κάνουμε τη δοκιμασία, αποδείχθηκε συνολικά καλύτερη και ταχύτερη από ελαφρά διαφοροποιημένες γειτονικές της παραμετροποιήσεις. Μάλιστα, διαπιστώσαμε πώς η αύξηση των επαναλήψεων μειώνει μέχρι ένα σημείο τη μέση λύση (που είναι επιθυμητό), αλλά από ένα σημείο και πέρα λειτουργεί από σταθεροποιητικά έως και

αρνητικά στην ποιότητα της λύσης, και εξηγήσαμε το γιατί (πέραν βεβαίως της αύξησης του χρόνου εκτέλεσης που σημαίνει η αύξηση των επαναλήψεων).

Ο στόχος που είχαμε από την εξόρυξη δεδομένων ήταν πολύ καλές λύσεις, σε πολύ καλό χρόνο. Η έννοια του «πολύ καλού χρόνου» είχε συγκριτικό χαρακτήρα μεταξύ των χρόνων που απαίτησαν οι διαφορετικές παράμετροι. Έτσι, λ.χ. ο πολύ καλός χρόνος για τα προβλήματα 20x5 ήταν χρόνος κάτω από 1 δευτερόλεπτο. Θα μπορούσαμε να θέταμε εξ αρχής στόχο ως προς το μέγιστο χρόνο που μπορούμε να διαθέσουμε σε κάθε διάσταση προβλήματος (λ.χ. στα προβλήματα 50x5 maximum 5 δευτερόλεπτα). Στην περίπτωση αυτή θα διεξαγάγαμε την ανάλυση της εξόρυξης δεδομένων αναζητώντας τους συνδυασμούς με καλύτερες λύσεις, εντός του ορίου του χρόνου. Πάλι, όμως, η εξόρυξη δεδομένων θα μας βοηθούσε, υποδεικνύοντας τους συνδυασμούς που έχουν βάσιμη εμπιστοσύνη ως προς την παραγωγή καλών αποτελεσμάτων και με συνέπεια, εντός βέβαια του χρονικού πλαισίου.

Ακόμα, όμως, και στην περίπτωση που ο χρόνος δεν ήταν πρόβλημα, και πάλι η εξόρυξη δεδομένων θα μας καθοδηγούσε στις παραμέτρους που δίνουν καλύτερες λύσεις. Είχαμε, σε μεγάλα προβλήματα, εκτελέσεις της τάξης των 20.000 δευτερολέπτων, που όμως δεν έδωσαν καλύτερες λύσεις σε σχέση με εκτελέσεις των 500 ή των 1000 δευτερολέπτων. Άρα, ακόμα και χωρίς την παράμετρο του χρόνου, η παραμετροποίηση μέσω του data mining θα μας καθοδηγούσε ως προς τις παραμέτρους που θα εξυπηρετούσαν καλύτερα του επιδιωκόμενο.

Πριν ολοκληρώσουμε το σχολιασμό των αποτελεσμάτων, είναι σημαντικό να αναφερθούμε στην έννοια της απομάκρυνσης από τη λύση εκκίνησης. Ένα πρόβλημα 20x20 έχει 20! λύσεις ενώ ένα πρόβλημα 500x20 έχει 500! λύσεις. Και τα δύο προβλήματα έχουν την πολυπλοκότητα των 20 μηχανών. Επίσης, αφιερώνοντας 7 δευτερόλεπτα για την επίλυση του πρώτου και 229 δευτερόλεπτα για την επίλυση του δεύτερου, διαπιστώνουμε ότι για το δεύτερο αφιερώσαμε 33 φορές περισσότερο χρόνο. Όμως, το μέγεθος του δεύτερου προβλήματος είναι $500!/20!=5x10^{1115}$ μεγαλύτερο από το πρώτο. Άρα, κατ' αναλογία, για να διερευνήσουμε το ίδιο ποσοστό του χώρου λύσεων στο πρόβλημα 500x20, όσο διερευνήσαμε σε 7 δευτερόλεπτα στο πρόβλημα 20x20, θα έπρεπε να αφιερώσουμε $3,5x10^{1116}$ δευτερόλεπτα, δηλαδή έναν ασύλληπτα μεγάλο χρόνο. Αυτό λοιπόν που κάνουμε στα μεγάλα προβλήματα, ακόμα και εκατονταπλασιάζοντας τον αριθμό των επαναλήψεων, είναι να εξετάζουμε ένα

πολύ-πολύ μικρό κλάσμα της γειτονιάς μίας λύσης σε σχέση με το κλάσμα που εξετάζουμε στα μικρά προβλήματα. Το μέγεθος όμως των προβλημάτων είναι τέτοιο, που σε αυτό το μικρό κλάσμα, βρίσκονται λύσεις με διάφορα makespan, συχνά μικρότερα της τρέχουσας, τα οποία και εντοπίζουμε. Στα μικρά προβλήματα η διαφοροποίηση των τιμών δεν είναι μεγάλη, ειδικά όταν ξεκινάμε με μία καλή αρχική λύση, οπότε γι' αυτό και δεν παρατηρούμε βελτίωση ακόμα κι εάν απομακρυνθούμε από την περιοχή της λύσης εκκίνησης.

Συνολικά, λοιπόν, η διαδικασία της εξόρυξης των δεδομένων μας καθοδήγησε επαρκώς και με ικανοποιητικότατα αποτελέσματα ως προς το στόχο που είχαμε θέσει. Με διαφορετικούς στόχους θα αξιοποιούσαμε επίσης κατάλληλα τα αποτελέσματα της εξόρυξης δεδομένων, επιτυγχάνοντας και πάλι το στόχο για καλύτερη παραμετροποίηση του αλγορίθμου. Οι μετρήσεις που έγιναν εκ των υστέρων επιβεβαίωσαν ότι η διαδικασία της εξόρυξης δεδομένων συνέβαλε θετικά στο να λάβουμε καλύτερα αποτελέσματα, σε καλύτερο χρόνο, από τον ίδιο αλγόριθμο, ενώ θα λαμβάναμε χειρότερα αποτελέσματα με λιγότερο κατάλληλη παραμετροποίηση.

"You will never "find" time for anything. If you want time, you must make it."

Charles Bruxton

Κεφάλαιο 6:

Συμπερασματικές παρατηρήσεις

Στο κεφάλαιο αυτό θα καταγράψουμε ορισμένα συμπεράσματα από όλη την εργασία. Τα συμπεράσματα αυτά είναι χρήσιμα για να καταγραφεί το τι έγινε στο πλαίσιο αυτής της εργασίας, τις προοπτικές που δημιουργεί και τα σχετικά ζητήματα που μπορούν να διερευνηθούν στο μέλλον.

6.1 Ο αλγόριθμος

Υπάρχουν δεκάδες ιδέες πάνω στις οποίες βασίζονται εξελικτικοί αλγόριθμοι για ολική βελτιστοποίηση. Και φυσικά υπάρχουν εκατοντάδες υλοποιήσεις αυτών των αλγορίθμων, με μικρότερες ή μεγαλύτερες διαφοροποιήσεις, καθώς και υλοποιήσεις που αξιοποιούν κάποια από τα χαρακτηριστικά των προβλημάτων στα οποία κυρίως εστιάζουν.

Η δική μας προσέγγιση εστίασε στα ακόλουθα:

- Συνδυασμός ιδεών
- Απλότητα
- Δυνατότητα χρήσης σε κάθε πρόβλημα αναδιάταξης σημείων
- Ταχύτητα εκτέλεσης

Εκτιμούμε ότι το μείγμα της προσέγγισής μας απέδωσε, καθώς ο αλγόριθμος αποδείχθηκε και εύρωστος και ταχύς. Σε πολλά προβλήματα, η εύρεση μίας πολύ καλής λύσης μπορεί να έρθει σε δέκατα του δευτερολέπτου, στοιχείο που καθιστά τον προτεινόμενο αλγόριθμο ιδανικό για εγκαταστάσεις just-in-time (JIT), όπου οι αποφάσεις πρέπει να λαμβάνονται άμεσα, σε πραγματικό χρόνο. Επιπρόσθετα, ο αλγόριθμος είναι απλούστατος στην υλοποίησή του σε κώδικα.

Αξίζει να επιμείνουμε λίγο περισσότερο στη δυνατότητά του να χρησιμοποιηθεί σε οποιοδήποτε πρόβλημα η λύση του προέρχεται με αναδιάταξη εργασιών/σημείων κ.λπ. Με μία απλή αλλαγή της αντικειμενικής συνάρτησης, δηλαδή της συνάρτησης που υπολογίζει την αξία της κάθε λύσης, μπορεί να χρησιμοποιηθεί σε προβλήματα flow-shop με blocking, job-shop, open-shop, ακόμα και σε προβλήματα τύπου travelling salesman problem (TSP) ή vehicle routing problem (VRP). Σε όλα αυτά τα προβλήματα η λύση προέρχεται από αναδιάταξη σημείων / εργασιών, όπως αυτή που κάνει ο αλγόριθμός μας. Φυσικά, για προβλήματα πέραν του flow-shop δεν μπορεί να χρησιμοποιηθεί ο αλγόριθμος ΝΕΗ, που είναι κατάλληλος μόνο για τα συγκεκριμένα προβλήματα. Όμως, υπάρχουν άλλοι αντίστοιχοι ευρετικού τύπου αλγόριθμοι για τα διάφορα προβλήματα, που θα μπορούσαν να μας προσφέρουν μία ικανοποιητική αρχική λύση.

Στη βιβλιογραφία υπάρχουν εξελικτικοί αλγόριθμοι με δυνατότητα να προσεγγίσουν τη BKS σε μεγάλο αριθμό προβλημάτων, μεγαλύτερο του 30 στα 120 που πέτυχε ο δικός μας. Όμως, με τη μελέτη αυτών των αλγορίθμων διαπιστώνουμε ότι πρόκειται για υλοποιήσεις που αξιοποιούν στοιχεία από το είδος του προβλήματος (Jin et al. 2009). Για παράδειγμα ο εξαιρετικός αλγόριθμος των Nowicki και Smutnicki (1994) χρησιμοποιεί την έννοια της κρίσιμης διαδρομής και των κρίσιμων ομάδων, ενώ αντίστοιχες έννοιες χρησιμοποιεί και ο επίσης εξαιρετικός αλγόριθμος των Grabowski και Wodecki (2004). Οι συγκεκριμένοι αλγόριθμοι επιτυγχάνουν λύσεις εξαιρετικής ποιότητας, ακόμα και στα προβλήματα μεγάλων διαστάσεων, αλλά είναι κατάλληλοι μόνον για προβλήματα flow-shop, αφού στηρίζουν την αποτελεσματικότητά τους σε χαρακτηριστικά του ίδιου του προβλήματος (καταγράφουν διαδρομές στον πίνακα λύσης). Ακόμα και για την παραλλαγή flow-shop με blocking χρειάζεται ένας άλλος αλγόριθμος, ενδεχομένως πολύ διαφορετικός από τον αλγόριθμο του προβλήματος flow-shop. Είναι σαφές ότι όσο πιο εξειδικευμένος είναι ένας αλγόριθμος, ως προς τα προβλήματα που μπορεί θα αντιμετωπίσει, τόσο καλύτερα συμπεριφέρεται στα προβλήματα αυτά, αλλά είναι εντελώς ακατάλληλος για άλλου τύπου προβλήματα. Εμείς επιδιώξαμε να δημιουργήσουμε έναν αλγόριθμο κατάλληλο για κάθε πρόβλημα αναδιάταξης εργασιών/σημείων, έστω κι εάν για λόγους διαχείρισης των μετρήσεων περιορίσαμε τη μελέτη του στο πρόβλημα flow-shop.

Το άλλο χαρακτηριστικό του αλγορίθμου είναι η δυνατότητα που παρέχει στο χρήστη για παραμετροποίηση. Μπορεί να επιλέξει μικρό ή μεγάλο σμήνος λύσεων, σε

προβλήματα που ευνοούνται από αυτό, ή περισσότερες επαναλήψεις στην παραγωγή λύσεων σε άλλα προβλήματα. Κατά προσέγγιση, οι επιλογές 1000 επαναλήψεις 4 λύσεις, 500 επαναλήψεις 8 λύσεις και 2000 επαναλήψεις 2 λύσεις, είναι περίπου ισοδύναμες σε χρόνο εκτέλεσης[11]. Όμως, με την εκτέλεση διαπιστώνουμε ότι έχουν διαφορετική συμπεριφορά ως προς τα αποτελέσματά τους. Για παράδειγμα, στο πρόβλημα 81, με το συνδυασμό 500/8 λαμβάνουμε makespan όχι μικρότερο του 6510, με τον συνδυασμό 1000/4 λαμβάνουμε makespan κάπου στο 6470, ενώ με το συνδυασμό 2000/2 επιτυγχάνουμε αποτελέσματα κοντά στο 6450, με όλες τις εκτελέσεις να είναι από 1,5 έως 2 δευτερόλεπτα. Συνεπώς, η δυνατότητα που παρέχει ο αλγόριθμος στον χρήστη να παραμετροποιήσει την εκτέλεσή του, βοηθάει στο να επιτύχουμε καλύτερα αποτελέσματα, ανάλογα με το είδος του προβλήματος που έχουμε να αντιμετωπίσουμε.

Ένα άλλο χαρακτηριστικό του αλγορίθμου μας είναι ότι θα μπορούσε να ενσωματώσει στο κομμάτι της τοπικής αναζήτησης και άλλα στοιχεία, είτε επιπρόσθετα είτε σε αντικατάσταση υπαρχόντων. Για παράδειγμα, αντί του κατωφλιού αποδοχής θα μπορούσε να χρησιμοποιηθεί ο προσομοιωμένη ανόπτιση. Επίσης θα μπορούσε να χρησιμοποιηθεί το tabu search είτε αντί του κατωφλιού αποδοχής, είτε σε συνδυασμό με το κατώφλι υποδοχής ή την προσομοιωμένη ανόπτιση. Αυτές είναι μερικές μόνο από τις ιδέες που θα μπορούσε να συνδυάσει κάποιος στον αλγόριθμό μας, πάντα διατηρώντας το χαρακτηριστικό του γενικού αλγορίθμου για προβλήματα αναδιάταξης.

6.2 Η εξόρυξη δεδομένων

Κατά την ανάλυση του αλγορίθμου παρουσιάσαμε την ανάγκη που προέκυψε για λήψη αποφάσεων ως προς συγκεκριμένες παραμέτρους και επιλογές. Εξηγήσαμε την ανάγκη και πώς η εξόρυξη δεδομένων μάς βοήθησε ώστε να κάνουμε τις κατάλληλες επιλογές σε κάθε περίπτωση που αναζητήσαμε σχετικές απαντήσεις.

Σε έναν εξελικτικό αλγόριθμο, μία μικρή αλλαγή μπορεί να προκαλέσει μεγάλη διαφορά, θετική ή αρνητική. Κατά τη μελέτη δεκάδων εξελικτικών αλγορίθμων από αυτούς που έχουν προταθεί, διαπιστώνουμε ότι οι δημιουργοί τους προχωρούν σε διάφορες επιλογές, με τρόπο λίγο έως πολύ αυθαίρετο ή τουλάχιστον χωρίς να δηλώνεται κάποια

[11] Υπάρχει ένας σταθερός χρόνος που απαιτείται για την αρχικοποίηση του προβλήματος (λ.χ. δημιουργία λύσης με ΝΕΗ, παραγωγή αρχικών λύσεων κ.λπ.) καθώς και ο χρόνος που απαιτείται για τη διενέργεια του path-relinking, ο οποίος είναι μεγαλύτερος όσο αυξάνουν οι λύσεις. Πάντως, σε γενικές γραμμές, μπορούμε να θεωρήσουμε ότι υπάρχει μία γραμμική σχέση μεταξύ λύσεων – επαναλήψεων και συνολικού χρόνου εκτέλεσης.

συστηματική διαδικασία ελέγχου η οποία τους οδήγησε στη μία έναντι της άλλης επιλογής.

Η συστηματική διαδικασία ελέγχου και απόφασης που ακολουθήσαμε, βασισμένη σε τεχνικές και αρχές της εξόρυξης δεδομένων, μας επιτρέπει να κάνουμε μία σειρά από διαπιστώσεις.

Η πρώτη έχει να κάνει με το ότι η παραμετροποίηση ενός εξελικτικού αλγορίθμου, είναι ένα πλήρες έργο εξόρυξης δεδομένων, και η όλη διαδικασία εμπεριέχει όλα τα στάδια/βήματα ενός ολοκληρωμένου έργου data mining, σύμφωνα με το μοντέλο CRISP-DM. Μία διαδικασία που ξεκινά από την κατανόηση του αντικειμένου (τη στόχευση της παραμετροποίησης) και των δεδομένων, την προετοιμασία των δεδομένων και τη μοντελοποίηση, την αξιολόγηση της ποιότητας του μοντέλου και τέλος την εφαρμογή του πάνω στα πραγματικά μας ερωτήματα, στα θέματα που θέλουμε αποφάσεις για τον αλγόριθμό μας.

Η δεύτερη διαπίστωση αφορά στο ότι η όλη διαδικασία παραμετροποίησης διεξάγεται μέσω πληθώρας μετρήσεων, δηλαδή πρωτογενών δεδομένων, τα οποία εμπεριέχουν γνώση που όμως γίνεται ορατή και χρήσιμη μόνο μετά από τη συγκεκριμένη διαδικασία της εξόρυξης δεδομένων. Από τις πρώτες σποραδικές εκτελέσεις του κώδικα, ήταν αντιληπτό ότι τα μικρά προβλήματα ήθελαν λιγότερες επαναλήψεις από τα μεγαλύτερα. Όμως, η αίσθηση αυτή, δεν ήταν αρκετή για να μας δώσει τις συγκεκριμένες και πλέον κατάλληλες τιμές για τις παραμέτρους μας. Αυτή η γνώση των κατάλληλων τιμών προέκυψε μετά από χιλιάδες μετρήσεις και μετά από την κατάλληλη δημιουργία, επεξεργασία και εφαρμογή των μοντέλων εξόρυξης δεδομένων που χρησιμοποιήσαμε.

Η τρίτη διαπίστωση είναι ότι η εξόρυξη δεδομένων μπορεί να μας καθοδηγήσει όχι μόνο σε συγκεκριμένες επιλογές, αλλά και σε ομάδες επιλογών. Το λογισμικό του data mining μπορεί να δώσει ομάδες παραμέτρων με συγκεκριμένα χαρακτηριστικά, όπως λ.χ. παραμετροποίηση για εξαιρετικά γρήγορες λύσεις, ή παραμετροποίηση για αναποτελεσματικές λύσεις. Η διαδικασία του clustering μπορεί να μας προσφέρει αρκετές εναλλακτικές για την παραμετροποίηση, και μάλιστα με τρόπο που δεν θα ήταν ευχερής με άλλη διαδικασία, μετά από τόσες χιλιάδες εκτελέσεις.

Η τέταρτη διαπίστωση είναι ότι η διαδικασία του data mining θα μπορούσε να απαντήσει και σε άλλα ερωτήματα. Για παράδειγμα θα μπορούσε να μας δώσει τις καλύτερες παραμέτρους ώστε ο χρόνος εκτέλεσης να μην ξεπερνά τα Χ δευτερόλεπτα. Ή τις καλύτερες παραμέτρους για πρόβλημα λ.χ. 100 εργασιών σε διαφορετικό αριθμό μηχανών (λ.χ. και 5 και 10 και 20), ή για πρόβλημα 10 μηχανών αλλά διαφορετικού αριθμού εργασιών. Τα ίδια δεδομένα, η ίδια λογική επεξεργασίας, μπορεί να δώσουν απάντηση σε διαφορετικά ερωτήματα, αλλάζοντας το πρόβλημα στο 1ο βήμα του CRISP-DM.

Η πέμπτη διαπίστωση είναι ότι η διαδικασία του data mining είναι στενά συνυφασμένη με το πρόβλημα που μελετάμε. Για παράδειγμα, αυτές οι παράμετροι ισχύουν για το πρόβλημα $Fm||C_{max}$ και μόνον. Εάν μας ενδιαφέρουν οι παράμετροι για ένα άλλο πρόβλημα, ακόμα και για το πολύ συγγενές flow-shop με blocking, θα πρέπει να ακολουθήσουμε την ίδια διαδικασία εξόρυξης δεδομένων, με μετρήσεις που αφορούν το flow-shop με blocking.

Η έκτη διαπίστωση είναι ότι τα μοντέλα του data mining μπορούν να δώσουν εκτιμήσεις ακόμα και για μεγέθη προβλημάτων που δεν έχουμε αντιμετωπίσει. Για παράδειγμα, εάν προκύψει ένα πρόβλημα μεγέθους 500x5, για το οποίο δεν έχουμε καθόλου δεδομένα εκτέλεσης, το μοντέλο του Δέντρου Απόφασης προτείνει ως συνδυασμό παραμέτρων, για VGVF λύση, το 40000/2. Με παρόμοιο τρόπο, έχοντας το μοντέλο, μπορούμε να λάβουμε ενδείξεις για τους καταλληλότερους συνδυασμούς παραμετροποίησης ακόμα και για προβλήματα άγνωστων διαστάσεων. Βέβαια, όσο απομακρυνόμαστε από διαστάσεις για τις οποίες δεν έχουμε δεδομένα, μειώνεται η ακρίβεια του μοντέλου και η δυνατότητά του να ανταποκριθεί αποτελεσματικά. Για παράδειγμα, εάν ζητήσουμε παραμετροποίηση για ένα πρόβλημα 20x40, ενδεχομένως οι προτάσεις που θα λάβουμε από το μοντέλο να μην είναι τόσο ικανοποιητικές όσο αυτές που λάβαμε στα υπόλοιπα προβλήματα, καθώς δεν έχουμε καμία εκτέλεση με 40 μηχανές και άρα τα μοντέλα πρέπει να εκτιμήσουν τη συμπεριφορά των παραμέτρων σε ένα τέτοιο πρόβλημα. Η εκτίμηση αυτή θα γίνει με βάση τη βαρύτητα που έχουν προσδιορίσει για τον αριθμό των μηχανών, από τα προβλήματα που περιλαμβάνουν οι εκτελέσεις μας.

Η έβδομη διαπίστωση είναι ότι η διαδικασία της εξόρυξης δεδομένων δεν είναι ισοδύναμη με την απλή εκτέλεση των προβλημάτων, αλλά εξάγει από αυτή πολύ

περισσότερη γνώση. Για παράδειγμα, μπορεί σε μία ομάδα προβλημάτων να επιλέξαμε ως καταλληλότερη την παραμετροποίηση 30000/6. Στις εκτελέσεις, όμως, δεν εκτελέσαμε ποτέ τα προβλήματα με τη συγκεκριμένη παραμετροποίηση. Εκτελέσαμε με παραπλήσιες παραμετροποιήσεις αλλά και με άλλες αρκετά διαφοροποιημένες. Τα μοντέλα μας υπέδειξαν την 30000/6 μετά από συνυπολογισμό των αποτελεσμάτων των εκτελέσεων τόσο στην ίδια την ομάδα που είναι το πρόβλημα, όσο και σε άλλες ομάδες με διαφορετικό αριθμό εργασιών και μηχανών. Έκαναν δηλαδή μία εκτίμηση, ότι με βάση τα όσα διαπιστώνουν σε έναν περιορισμένο αριθμό δοκιμαστικών παραμετροποιήσεων, επί προβλημάτων διαφόρων μεγεθών, για το συγκεκριμένο πρόβλημα ο ιδανικότερος συνδυασμός είναι ο 30000/6. Αυτό μας εξοικονομεί πολύ χρόνο, καθώς αν έπρεπε να εκτελέσουμε κάθε σετ με όλους τους δυνατούς συνδυασμούς επαναλήψεων/λύσεων, ώστε να βρούμε αυτόν που θα έδινε το καλύτερο αποτέλεσμα (και μάλιστα θα έπρεπε να το εκτελέσουμε πολλές φορές ώστε να μειώσουμε την επίδραση της στοχαστικότητας), θα συζητούσαμε για εκατοντάδες χιλιάδες εκτελέσεις, και ενδεχομένως αρκετούς μήνες εκτελέσεων! Αυτά όμως εξοικονομήθηκαν με τη χρήση των μοντέλων εξόρυξης δεδομένων.

Να επισημάνουμε ότι ο μεγάλος αριθμός εκτελέσεων για τη δημιουργία του μοντέλου που εμείς χρησιμοποιήσαμε, δεν είναι απαραίτητο να ακολουθείται σε κάθε περίπτωση. Θα μπορούσαμε να περιορίσουμε τις εκτελέσεις μας σε 1-2 προβλήματα ανά ομάδα (δεκάδα) και να λάβουμε μία ικανοποιητική ένδειξη για τις κατάλληλες παραμέτρους. Όλα είναι θέμα μιας της ισορροπίας μεταξύ του χρόνου που θέλουμε να αφιερώσουμε για τη δημιουργία του μοντέλου και της ακρίβειας που αυτό επιθυμούμε να έχει. Άλλωστε, το εάν θα είναι καλό ή κακό το μοντέλο, εν πολλοίς θα το επιβεβαιώσει η διαδικασία του αξιολόγησης (evaluation) του μοντέλου, αφού ένα κακό μοντέλο δεν θα δίνει ικανοποιητικό αριθμό ορθών εκτιμήσεων πάνω στα πραγματικά δεδομένα (οπότε κατά συνέπεια δεν παρέχει τα εχέγγυα να το κάνει και σε άγνωστα δεδομένα).

Πάντως, έχοντας ως ένδειξη τα αποτελέσματα του μοντέλου για τα προβλήματα διαφόρων διαστάσεων, θα μπορούσαμε σε δεύτερη φάση να επανέλθουμε με μετρήσεις εστιασμένες στην περιοχή των τιμών παραμέτρων που τα μοντέλα μας επέδειξαν ως καταλληλότερες. Για παράδειγμα, αφού στα «μεγάλα» προβλήματα φαίνεται να μην είναι αποτελεσματική η παραμετροποίηση με λίγες επαναλήψεις και πολλές λύσεις (το τελευταίο λόγο μεγάλης αύξησης του χρόνου), θα μπορούσαμε να εστιάσουμε σε εκτελέσεις για μετρήσεις με παραμέτρους κοντά στις προτεινόμενες τιμές (μία

διαδικασία fine-tuning). Αντίστοιχα θα μπορούσαμε να κινηθούμε σε όλα τα μεγέθη προβλημάτων, βελτιώνοντας ακόμα περισσότερο τα μοντέλα μας. Οι μετρήσεις που εμείς χρησιμοποιήσαμε κινούνται σε μεγάλο εύρος συνδυασμών παραμέτρων για όλα τα προβλήματα, γιατί αρχικά κινούμασταν «στα τυφλά» ως προς το πού βρίσκονται οι κατάλληλοι παράμετροι. Τώρα που το γνωρίζουμε θα μπορούσαμε να εστιάσουμε περισσότερο σε αυτές τις περιοχές.

6.3 Οι προοπτικές για μελλοντική έρευνα

Η εφαρμογή διαδικασιών data mining για βελτιστοποίηση αλγορίθμων, και ειδικότερα στο χώρο της βελτιστοποίησης, είναι ένα πεδίο που πρακτικά δεν έχει διερευνηθεί. Οι δημιουργοί αλγορίθμων στις δημοσιεύσεις τους αναφέρονται σε αποφάσεις που έλαβαν κατά το σχεδιασμό των αλγορίθμων τους, χωρίς όμως να αναφέρουν εάν ακολούθησαν κάποια συστηματική διαδικασία ελέγχου για τις αποφάσεις αυτές.

Η διαδικασία που εφαρμόσαμε μας επέτρεψε να λάβουμε το καλύτερο από έναν αλγόριθμο που δημιουργήσαμε. Ακολουθώντας αντίστοιχες διαδικασίες, μπορούμε να βελτιστοποιήσουμε νέους αλλά και υπάρχοντες αλγορίθμους, τόσο στη γενικότητά τους όσο και σε συγκεκριμένα προβλήματα.

Είναι συχνό το φαινόμενο να αφιερώνονται ανθρωποέτη στη δημιουργία ενός καλού και αποτελεσματικού αλγορίθμου βελτιστοποίησης, που βασίζεται σε μελέτη του προβλήματος βελτιστοποίησης που αυτός καλείται να επιλύσει. Βελτιστοποιώντας τη διαδικασία παραμετροποίησης, μπορούμε να βελτιώσουμε την αποτελεσματικότητα κάθε αλγορίθμου με στοχαστικά στοιχεία, λαμβάνοντας το καλύτερο που αυτός μπορεί να αποδώσει.

Περαιτέρω έρευνα μπορεί να γίνει πάνω στον ίδιο τον προτεινόμενο αλγόριθμο, σε σημεία που αναφέρθηκαν στο πλαίσιο της εργασίας αυτής. Όμως, μπορεί να γίνει και σε υπάρχοντες αλγορίθμους, από τους εκατοντάδες που έχουν δημοσιευτεί. Το έργο αυτό μπορεί να οδηγήσει σε ακόμα καλύτερα αποτελέσματα, χρησιμοποιώντας ήδη υπάρχουσες ιδέες, σε συνδυασμό με τη βελτιστοποίηση που μπορεί να προσφέρει η εξόρυξη δεδομένων στην παραμετροποίησή τους.

Για παράδειγμα, εάν εστιάσουμε στον αλγόριθμο tabu search, σε κάθε υλοποίηση υπάρχουν ερωτήματα για το μέγεθος της λίστας tabu, τη διαδικασία εντατικοποίησης

(intensification) της αναζήτησης, την αναζήτηση σε άλλες περιοχές (diversification) κ.ο.κ. (Hertz et al. 1997, Μαρινάκης et al. 2010, Glover et al. 1997, Glover et al. 1993). Η διαδικασία της παραμετροποίησης των παραπάνω με βάση το μέγεθος και τα χαρακτηριστικά του προβλήματος δεν είναι κάτι που εντοπίζεται συχνά στη βιβλιογραφία, καθώς οι περισσότεροι ερευνητές εντοπίζουν μία τιμή ως καταλληλότερη και την εφαρμόζουν σε όλα τα προβλήματα. Όμως, αν ο αλγόριθμος έχει διαφοροποιημένη συμπεριφορά σε διαφορετικά μεγέθη προβλημάτων, η ακριβής προσαρμογή του με μία διαδικασία εξόρυξης δεδομένων όπως αυτή που προτείνεται στην παρούσα εργασία, θα μπορούσε να βελτιώσει σημαντικά την αποτελεσματικότητά του. Φυσικά, αντίστοιχα ζητήματα υπάρχουν σε κάθε αλγόριθμο, όπως λ.χ. σε αυτόν της προσομοιωμένης ανόπτισης (λ.χ. προγραμματισμός διαδικασίας ψύξης) – (Aarts et al. 1997, Μαρινάκης et al. 2010), σε γενετικούς αλγορίθμους (λ.χ. συχνότητα μετάλλαξης, μέγεθος πληθυσμού) – (Muhlenbein 1997, Μαρινάκης et al. 2010) κ.ά.

Ένα νέο πεδίο έρευνας στη «βελτιστοποίηση της βελτιστοποίησης», φαίνεται να είναι αρκετά υποσχόμενο…

Βιβλιογραφία

Aarts, E. H. L., Korst, J. H. M., van Laarhoven, P., J., M. (1997) "Simulated Annealing" in (Aarts, 1997) pp. 91-120.

Aarts, E., Lenstra, J. K. (editors - 1997) "Local Search in Combinatorial Optimization", John Wiley & Sons.

Ancau, M. (2012) "On solving flow-shop scheduling problems", in Proceedings of the Romanian Academy, Series A, Vol 13, Number 1/2012, pp. 71-79.

Anderson, E. J., Glass, C. A., Potts, C. N. (1997) "Machine Scheduling" in (Aarts, 1997) pp. 361-414.

Berry, M. J. A., Linoff, G. (2004) "Data mining techniques for marketing, sales and customer relationship management", Wiley.

Chakraborty, U. K. (editor - 2009) "Computational Intelligence in Flow Shop and Job Shop Scheduling", Springer – Verlag.

Emmons, H., Vairaktarakis, G. (2013) "Flowshop Scheduling – Theoretical Results, Algorithms and Applications", Springer – Verlag.

Gao, J., Chen, R. (2011) "An NEH-based heuristic algorithm for distributed permutation flowshop scheduling problems" in Scientific Research and Essays Vol 6(14) pp. 3094-3100.

Garey, M. R., Johnson, D. S., Sethi, R. (1976) "The complexity of flow-shop and job-shop scheduling", in Mathematics of Operations Research 1, 117-129.

Graham, R. L., Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G. (1979) "Optimization and approximation in deterministic sequencing and scheduling survey" in Discrete Optimization, Annals of Discrete Mathematics 5, North-Holland, Amsterdam, 287-326.

Glover, F., Laguna, M. (1997) "Tabu Search", Kluwer Academic Publishers.

Glover, F., Taillard, E., de Werra, D. (1993) "A User's Guide to Tabu Search" in Annals of Operations Research 41 (1993) 3-28.

Grabowski, J., Wodecki, M. (2004) "A Very Fast Tabu Search Algorithm for the Permutation Flow Shop Problem with Makespan Criterion" in Computers & Operations Research 31 (2004) 1891-1909.

Grabowski, J., Pempera, J. (2004) "The Permutation Flow Shop Problem with Blocking. A Tabu Search Approach" in Omega 35 (2007) 302-311.

Hertz, A., Taillard, E., de Werra, D. (1997) "Tabu Search" in (Aarts, 1997) pp. 121-136.

Jarboui, B., Eddaly, M., Siarry, P., Rebai, A. (2009) "An Estimation of Distribution Algorithm for Minimizing the Makespan in Blocking Flowshop Scheduling Problems" in (Chakraborty, 2009) pp. 151-168.

Jin, F., Song, S., Wu, C. (2009) "Structural Property and Meta-Heuristic for the Flow-Shop Scheduling Problem" in Comput. Intel. in Flow Shop and Job Shop Sched., SCI 230 pp. 1-20.

Johnson, D. S., McGeoch, L. A. (1997) "The Traveling Salesman Problem: A Case Study" in (Aarts, 1997) pp. 215-310.

Johnson, S. M. (1954) "Optimal two- and three-stage production schedules with setup times included", Naval Research Logistics Quarterly 1, 61-68.

Kouki, S., Jemni, M., Ladhari, T. (2011), "Solving the Permutation Flow Shop Problem with Makespan Criterion using Grids", in International Journal of Grid and Distributed Computing, vol. 4, no 2, June 2011.

Lageweg, B. J., Lenstra, J. K., Rinnooy Kan, A. H. G. (1978) "A general bounding scheme for the permutation flow-shop problem" in Operations Research 26, 53-67.

Marinakis, Y., Marinaki, M. (2013) "Particle swarm optimization with expanding neighborhood topology for the permutation flowshop scheduling problem", Soft Comput (2013) 17:1159–1173, Springer-Verlag.

Mokotoff, E. (2009) "Multi-Objective Simulated Annealing for Permutation Flow-Shop Problems" in (Chakraborty, 2009) pp. 101-150.

Muhlenbein, H. (1997) "Genetic Algorithms" in (Aarts, 1997) pp. 137-171.

Nawaz, M., Enscore, E. E. jr., Ham, I. (1982) "A Heuristic Algorithm for the m-Machine, n-Job Flow-Shop Sequencing Problem" in OMEGA The international journal of management science vol. 11, no. 1, pp. 91-95 (1983).

North, M. (2012) "Data Mining for the Masses" Global Text Project (CCA 3.0 License).

Nowicki, E., Smutnicki, C. (1994) "A Fast Tabu Search Algorithm for the Permutation Flow-Shop Problem" in European Journal of Operational Research 91 (1996) 160-175.

Nowicki, E., Smutnicki, C. (2004) "Some aspects of scatter search in the flow-shop problem" in European Journal of Operational Research 169 (2006) 654-666.

Potts, C. N. (1980) "An adaptive branching rule for the permutation flow-shop problem" in European Journal of Operational Research 5, 19-25.

Ruiz, R., Vallada, E., Fernandez-Martinez, C. (2009) "Scheduling in Flow-Shops with No-Idle Machines" in (Chakraborty, 2009) pp. 21-52.

Taha, H. A. (2014) "Εισαγωγή στην Επιχειρησιακή Έρευνα" (9η έκδοση - μεταφρ.: Μάργαρης, Α. Ι., επιμέλεια: Κατσαβούνης, Σ.) εκδόσεις Τζιόλα.

Taillard, E. (1990) "Some efficient heuristic methods for the flow shop sequencing problem", European Journal of Operational Research 47 (1990) 65-74.

Taillard, E. (1993) "Benchmarks for basic scheduling problems" in European Journal of Operational Research, vol. 64, pp. 278-285.

Tovey, C. A. (1997) "Local Improvement on Discrete Structures" in (Aarts, 1997) pp. 57-89.

Tuffery, S. (2011) "Data Mining and Statistics in Decision Making" (translated by Riesco R.) Wiley.

Wang, J. (editor – 2008) "Encyclopedia of data warehousing and data mining" (2nd ed.) Information Science Reference.

Watson, J-P, Barbulescu, L., Howe, A. E., Whitley, L. D. (1999) "Algorithm Performance and Problem Structure for Flow-shop Scheduling", American Association for Artificial Intelligence.

Witten, I. H., Eibe, F., Hall, M. A. (2011) "Data Mining: Practical Machine Learning Tools and Techniques" (3rd ed.), Elsevier.

Yannakakis, M. (1997) "Computational Complexity" in (Aarts, 1997) pp. 19-55.

Μαρινάκης, Ι., Μαρινάκη, Μ. (2010) "Σημειώσεις Μεταπτυχιακού Μαθήματος: Εξελικτικοί Αλγόριθμοι και Βελτιστοποίηση Συστημάτων Μεγάλης Κλίμακας", Πολυτεχνείο Κρήτης.