Μεταπτυχιακο Προγραμμα
Στη Λογικη και Θεωρια Αλγοριθμων και Υπολογισμου

$\mu\Pi\lambda\forall$

ΠΡΟΣΕΓΓΙΣΤΙΚΑ ΣΧΗΜΑΤΑ ΓΙΑ ΠΡΟΒΛΗΜΑΤΑ ΧΡΟΝΟΔΡΟΜΟΛΟΓΗΣΗΣ

Διπλωματικη Εργασια

Γεώργιος Ζώης

Επιβλέπων: Σταύρος Γ. Κολλιόπουλος, Επ. Καθηγητής,
Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Ε.Κ.Π.Α.

Αθήνα, Μάρτιος 2008

Περίληψη

Στην εργασία αυτή μελετώνται προσεγγιστικά σχήματα από τη βιβλιογραφία για **NP**-hard προβλήματα χρονοδρομολόγησης. Ένα προσεγγιστικό σχήμα για ένα **NP**-hard πρόβλημα βελτιστοποίησης είναι ένας αλγόριθμος ο οποίος, για κάθε προκαθορισμένη σταθερά $\epsilon > 0$, βρίσκει μία λύση στο πρόβλημα με τιμή μεγαλύτερη κατά το πολύ έναν παράγοντα $1 + \epsilon$ από την τιμή της βέλτιστης λύσης του προβλήματος. Αν η πολυπλοκότητα χρόνου του προσεγγιστικού σχήματος είναι πολυωνυμική ως προς το μέγεθος της εισόδου, τότε το προσεγγιστικό σχήμα καλείται PTAS (Polynomial Time Approximation Scheme). Αν επιπλέον, η πολυπλοκότητα χρόνου είναι πολυωνυμική και ως προς το $1/\epsilon$, τότε το προσεγγιστικό σχήμα καλείται FPTAS (Fully Polynomial Time Approximation Scheme). Ένα τυπικό πρόβλημα χρονοδρομολόγησης αποτελείται από ένα σύνολο διεργασιών, υποκείμενες σε διάφορους περιορισμούς, οι οποίες πρόκειται να εκτελεστούν σε ένα σύνολο διαθέσιμων μηχανών. Δύο συνήθεις στόχοι είναι η ελαχιστοποίηση του makespan (ο απαιτούμενος χρόνος για την ολοκλήρωση όλων των διεργασιών) και η ελαχιστοποίηση του μέσου χρόνου ολοκλήρωσης. Οι περιορισμοί που μας ενδιαφέρουν περιλαμβάνουν χρόνους αποδέσμευσης, προθεσμίες καθώς και preemptive χρονοδρομολογήσεις των διεργασιών. Σε μία preemtive χρονοδρομολόγηση οι διεργασίες μπορούν να διακόπτουν την εκτέλεσή τους, ολοκληρώνοντάς τη σε κάποια επόμενη χρονική στιγμή της χρονοδρομολόγησης. Ακολούθως παραθέτουμε με συνοπτικό τρόπο τα προβλήματα και τα αντίστοιχα αποτελέσματα που μελετάμε.

Μελετάμε το πρόβλημα ελαχιστοποίησης του makespan σε περιβάλλον Παράλληλων Μηχανών. Περιγράφουμε μία γενική τεχνική σχεδιασμού PTAS και την εφαρμόζουμε στο πρόβλημα ελαχιστοποίησης του makespan σε δύο παράλληλες πανομοιότυπες μηχανές [86]. Στη συνέχεια παρουσιάζουμε δύο διαφορετικά PTAS για την εκδοχή του

προβλήματος με γενικό αριθμό παράλληλων πανομοιότυπων μηχανών [39, 86].

Το makespan μίας χρονοδρομολόγησης μετρά το χρόνο στον οποίο ολοκληρώνεται η εκτέλεση όλων των διεργασιών. Ωστόσο, αν θεωρήσουμε ότι οι διεργασίες συναγωνίζονται ανεξάρτητα η μία από την άλλη για τον ίδιο στόχο, τότε ένα πιο ταιριαστό μέτρο της απόδοσής τους είναι ο μέσος χρόνος ολοκλήρωσης της καθεμιάς. Μελετάμε προβλήματα ελαχιστοποίησης του μέσου (βεβαρημένου) χρόνου ολοκλήρωσης σε περιβάλλον Μοναδικής Μηχανής και σε περιβάλλον Παράλληλων Μηχανών. Παρουσιάζουμε ένα FPTAS για το πρόβλημα σε δύο παράλληλες πανομοιότυπες μηχανές [86]. Ακολούθως, θεωρούμε ότι οι διεργασίες διαθέτουν προκαθορισμένους χρόνους αποδέσμευσης και παρουσιάζουμε ένα PTAS για το πρόβλημα σε περιβάλλον Μοναδικής Μηχανής, όπου οι διεργασίες διαθέτουν μοναδιαία βάρη. Παρουσιάζουμε δύο ακόμη PTAS για τις preemptive και non-preemptive εκδοχές του προβλήματος σε γενικό αριθμό παράλληλων πανομοιότυπων μηχανών [2].

Η χρονοδρομολόγηση των διεργασιών για την ελαχιστοποίηση του makespan ή του μέσου χρόνου ολοκλήρωσης δεν θέτει περιορισμούς ως προς το χρόνο ολοκλήρωσης κάθε διεργασίας ατομικά. Αυτό που απαιτείται είναι οι διεργασίες να ολοκληρώνονται στον καλύτερο δυνατό χρόνο με σκοπό την ελαχιστοποίηση της αντίστοιχης αντικειμενικής συνάρτησης. Συσχετίζοντας κάθε διεργασία με μία θετική ακέραια προθεσμία και με μία απώλεια για κάθε αποτυχία ολοκλήρωσης της εκτέλεσής της στον χρόνο αυτό (σε μία δεδομένη χρονοδρομολόγηση) προκύπτει μία σειρά αντικειμενικών συναρτήσεων οι οποίες καλούνται συναρτήσεις απώλειας. Μελετάμε επίσης προβλήματα βελτιστοποίησης διαφόρων συναρτήσεων απώλειας, σε περιβάλλον Μοναδικής Μηχανής. Παρουσιάζουμε ψευδοπολυωνυμικούς αλγορίθμους δυναμικού προγραμματισμού για τα προβλήματα ελαχιστοποίησης του συνολικού βάρους των καθυστερημένων διεργασιών, μεγιστοποίησης της συνολικής βεβαρημένης πρόωρης ολοκλήρωσης, ελαχιστοποίησης της συνολικής βεβαρημένης καθυστέρησης, όταν οι διεργασίες διαθέτουν μία κοινή προθεσμία και ελαχιστοποίησης της συνολικής καθυστέρησης (βλ. [63, 60]). Χρησιμοποιούμε τους παραπάνω αλγορίθμους και σχεδιάζουμε δύο FPTAS για τα προβλήματα ελαχιστοποίησης του συνολικού βάρους των καθυστερημένων διεργασιών και ελαχιστοποίησης της συνολικής καθυστέρησης αντίστοιχα [86, 61].

Ευχαριστίες

Ευχαριστώ τον επιβλέποντα καθηγητή μου, κ. Σταύρο Κολλιόπουλο, για την ουσιώδη καθοδήγηση, τις πολύτιμες υποδείξεις και την επιμονή του στην τελειοποίηση των διατυπώσεων καθόλη τη διάρκεια της συνεργασίας μας. Το ήθος και η ευγένεια που τον διακρίνει αποτελούν πρότυπο για εμένα. Ευχαριστώ επίσης τα μέλη της τριμελούς επιτροπής, καθηγητές κκ. Δ. Θηλυκό και Β. Ζησιμόπουλο. Στον πρώτο οφείλω ένα ιδιαίτερο ευχαριστώ για όλες τις εποικοδομητικές συζητήσεις που είχα μαζί του.

Ευχαριστώ τους καθηγητές κκ. Ε. Ζάχο και Α. Παγουρτζή καθώς και όλη την ομάδα του Corelab για την πολύτιμη βοήθειά τους, ιδιαίτερα κατά τον πρώτο χρόνο της φοίτησής μου στο μΠλ∀. Θα πρέπει να εκφράσω την ευγνωμοσύνη μου και σε όλους τους συντελεστές του μΠλ∀ και ιδιαίτερα στους καθηγητές κκ. Γ. Μοσχοβάκη και Κ. Δημητρακόπουλο για την ευκαιρία σπουδών που μου έδωσαν.

Ευχαριστώ επίσης το φίλο μου Γ. Βαφειάδη για τις υποδείξεις του σε θέματα σχετικά με τη διαχείριση του LaTeX.

Τέλος, ευχαριστώ τους γονείς μου για την υπομονή και τη στήριξή τους καθόλη τη διάρκεια των σπουδών μου.

Γεώργιος Ζώης,

Αθήνα, Μάρτιος 2008

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Αλγορίθμων

Κεφάλαιο 1

Εισαγωγή

Στη σημερινή βιομηχανία, με τα πολλαπλά είδη παραγόμενων προϊόντων, απαιτούνται αρκετά στάδια επεξεργασίας καθώς και διαφορετικός μηχανικός εξοπλισμός για την παραγωγή ενός προϊόντος. Είναι σημαντικό οι αποφάσεις για το εκάστοτε βιομηχανικό σχέδιο, να επικεντρώνονται στην επιτυχή διαχείριση των διαθέσιμων πόρων έτσι ώστε η παραγωγή των προϊόντων να γίνεται με τον αποδοτικότερο δυνατό τρόπο. Για τη λήψη σωστών αποφάσεων είναι απαραίτητος ο χρονοπρογραμματισμός ενός σχεδίου που προωθεί την έγκαιρη διανομή και επιπλέον ελαχιστοποιεί τους αντικειμενικούς στόχους, όπως είναι ο χρόνος που απαιτείται για την παραγωγή ενός προϊόντος.

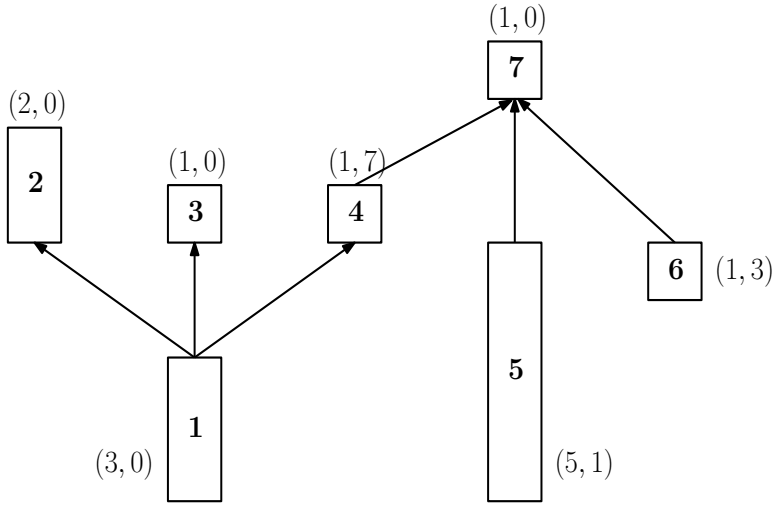
Με βάση αυτούς τους προβληματισμούς, άρχισε να δημιουργείται τις τελευταίες δεκαετίες μία περιοχή μελέτης γνωστή ως *προβλήματα χρονοδρομολόγησης (scheduling problems)*. Πολλά τεχνητά και υπαρκτά προβλήματα χρονοδρομολόγησης έχουν μελετηθεί σε βάθος και έχουν συνθέσει μία πλούσια και ισχυρή θεωρία, ενώ οι αλγοριθμικές τεχνικές που επινοήθηκαν για την επίλυση τους, αλλά και οι μέθοδοι για τον καθορισμό της υπολογιστικής τους πολυπλοκότητας, έχουν επηρεάσει την έρευνα σε περιοχές τόσο της Πληροφορικής (Θεωρητική Πληροφορική, Βάσεις Δεδομένων, Δίκτυα Επικοινωνιών), όσο και της Οικονομίας (Επιχειρησιακή Έρευνα).

1.1 Παράσταση των προβλημάτων χρονοδρομολόγησης

Κάθε πρόβλημα χρονοδρομολόγησης προϋποθέτει ένα σύνολο διεργασιών J, οι οποίες θα πρέπει να εκτελεστούν από ένα σύνολο μηχανών M. Η εκτέλεση κάθε διεργασίας $j \in J$ απαιτεί μία συγκεκριμένη ποσότητα χρόνου p_j η οποία καλείται χρόνος επεξεργασίας και μπορεί να εξαρτάται ή όχι από τη μηχανή που εκτελεί την διεργασία.

Οι διεργασίες ενδεχομένως να υπόκεινται σε κάποιους περιορισμούς ή να διαθέτουν κάποιες

επιπλέον ιδιότητες που πρέπει να ληφθούν υπόψην πριν τη χρονοδρομολόγηση τους. Για παράδειγμα μπορεί να υπάρχει μία δεδομένη μερική διάταξη με βάση την οποία πρέπει να εκτελεστούν *(περιορισμοί προτεραιότητας (precedence constraints))*· οι περιορισμοί προτεραιότητας αναπαρίστανται με τη μορφή ενός άκυκλου κατευθυνόμενου γραφήματος. Οι διεργασίες μπορεί να διακόπτονται και να συνεχίζουν την εκτέλεση τους σε κάποια επόμενη χρονική στιγμή *(preemption)*. Μπορεί ακόμα για κάθε διεργασία να υπάρχει κάποιος προκαθορισμένος χρόνος πριν το πέρας του οποίου να μην επιτρέπεται να αρχίσει η εκτέλεσή της. Ο χρόνος αυτός ονομάζεται *χρόνος αποδέσμευσης (release date)* της διεργασίας. Επίσης κάθε διεργασία μπορεί να έχει μία προκαθορισμένη *προθεσμία (due date)*. Αν, σε μία δεδομένη χρονοδρομολόγηση, μία διεργασία ολοκληρώσει την εκτέλεσή της μετά την προθεσμία της χαρακτηρίζεται ως καθυστερημένη (tardy), ενώ σε αντίθετη περίπτωση ως πρόωρη (early). Στο Σχήμα 1.1 δίνουμε ένα παράδειγμα που εξηγεί κάποιους από τους παραπάνω ορισμούς. Απεικονίζουμε το στιγμιότυπο ενός προβλήματος με επτά διεργασίες. Κάθε διεργασία αναπαρίσταται με ένα ορθογώνιο παραλληλόγραμμο και προσδιορίζεται από τον αριθμό που υπάρχει μέσα σ' αυτό. Ο χρόνος επεξεργασίας και ο χρόνος αποδέσμευσης κάθε διεργασίας δίνονται με τη μορφή ενός διατεταγμένου ζεύγους (p_j, r_j), που τοποθετείται δίπλα σ' αυτή στο σχήμα. Οι περιορισμοί προτεραιότητας αντιστοιχούν σε κατευθυνόμενες ακμές.



Σχήμα 1.1: Ένα στιγμιότυπο προβλήματος με περιορισμούς προτεραιότητας και χρόνους αποδέσμευσης.

Σε ένα πρόβλημα χρονοδρομολόγησης είναι δυνατόν κάθε διεργασία να εμπεριέχει μία ή και περισσότερες αναθέσεις (tasks). Ο όρος "ανάθεση" ερμηνεύεται ως η ενέργεια που συμβάλει στην ολοκλήρωση της διεργασίας. Στα περισσότερα προβλήματα που μελετώνται, οι διεργασίες εμπεριέχουν μία μόνο ανάθεση και επομένως η δέσμευση μιας μόνο μηχανής από το σύνολο M

αρκεί για την εκτέλεση της κάθε διεργασίας. Σε αντίθετη περίπτωση μία διεργασία απαιτεί περισσότερες από μία μηχανές για να εκτελεστεί.

Το περιβάλλον των μηχανών, στα περισσότερα προβλήματα χρονοδρομολόγησης ταυτίζεται με ένα από τα ακόλουθα.

Μοναδική Μηχανή (Single Machine). Μία μόνο μηχανή διατίθεται για την εκτέλεση των διεργασιών. Κάθε διεργασία εμπεριέχει μία ανάθεση και επιπλέον όλες οι διεργασίες εκτελούνται στην ίδια μηχανή.

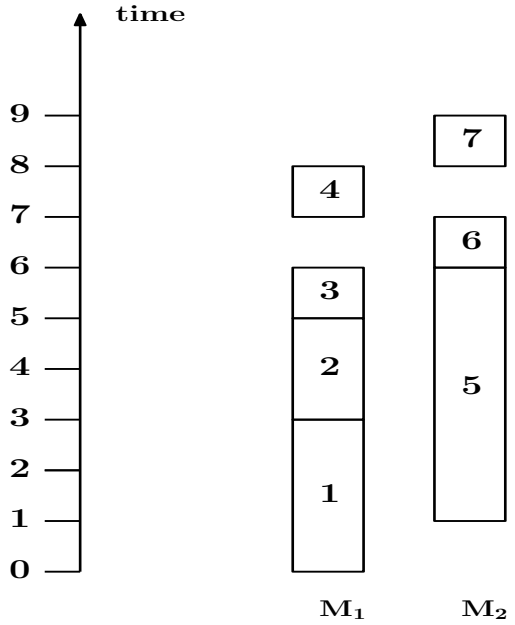
Παράλληλες Μηχανές (Parallel Machines). Πολλαπλές μηχανές διατίθενται για την εκτέλεση των διεργασιών. Οι μηχανές μπορεί να είναι *πανομοιότυπες (identical)*, *μη σχετιζόμενες (unrelated)*, *να διαθέτουν κάποια ταχύτητα εκτέλεσης (uniform)* ή να ειδικεύονται στην εκτέλεση συγκεκριμένων διεργασιών. Ειδικά στην περίπτωση που οι μηχανές είναι μη σχετιζόμενες ο χρόνος επεξεργασίας της κάθε διεργασίας διαφέρει από μηχανή σε μηχανή. Επίσης, η κάθε διεργασία εμπεριέχει μία μόνο ανάθεση.

Flow Shop. Στο πρότυπο αυτό υπάρχει μία αριθμημένη ακολουθία μηχανών $1, 2, 3, \ldots, m$. Κάθε διεργασία εμπεριέχει ακριβώς m αναθέσεις. Η πρώτη ανάθεση για κάθε διεργασία εκτελείται στη μηχανή 1, η δεύτερη στη μηχανή 2 κ.ο.κ. Για την εκτέλεση κάθε διεργασίας χρησιμοποιούνται και οι m μηχανές ενώ οι χρόνοι επεξεργασίας των αναθέσεων σε μία μηχανή διαφέρουν ανάλογα με τη διεργασία στην οποία εμπεριέχεται η κάθε ανάθεση. Σε περίπτωση που κάποιες διεργασίες εμπεριέχουν λιγότερες από m αναθέσεις θεωρούμε μηδενικό το χρόνο επεξεργασίας των αναθέσεων που δεν υπάρχουν. Τέλος, για κάθε διεργασία θα πρέπει η ανάθεση $i-1$ που εκτελείται στη μηχανή $i-1$ να ολοκληρώσει την εκτέλεσή της πριν από την έναρξη εκτέλεσης της ανάθεσης i στη μηχανή i.

Job Shop. Στο πρότυπο αυτό υπάρχει ένα σύνολο αριθμημένων μηχανών και ένα σύνολο αριθμημένων διεργασιών. Κάθε διεργασία εμπεριέχει έναν συγκεκριμένο πλήθος αναθέσεων και κάθε ανάθεση σε μία μηχανή υποδεικνύεται από τρείς δείκτες: ο πρώτος αφορά στον αριθμό της διεργασίας στην οποία ανήκει η ανάθεση, ο δεύτερος στον αριθμό της ίδιας της ανάθεσης και ο τρίτος στον αριθμό της μηχανής στην οποία χρειάζεται να εκτελεστεί η ανάθεση. Σε αντίθεση με το Flow Shop πρότυπο οι αναθέσεις μιας διεργασίας δεν είναι απαραίτητο να εκτελούνται με κατευθυνόμενη σειρά. Τέλος, κάθε διεργασία μπορεί να χρησιμοποιήσει μία μηχανή περισσότερες από μία φορές.

Στο Σχήμα 1.2 απεικονίζεται μία non-preemptive χρονοδρομολόγηση των διεργασιών του Σχήματος 1.1, σε δύο παράλληλες πανομοιότυπες μηχανές. Παρατηρούμε ότι λόγω των αντίστοιχων χρόνων αποδέσμευσης οι διεργασίες 4 και 5 αναγκάζονται να περιμένουν παρότι υπάρχει δια-

θέσιμη μηχανή. Επιπλέον, λόγω του περιορισμού προτεραιότητας για τις διεργασίες 4 και 7, η διεργασία 7 πρέπει να περιμένει μέχρι την ολοκλήρωση της 4.



Σχήμα 1.2: Μία χρονοδρομολόγηση των διεργασιών του Σχήματος 1.1 σε δύο μηχανές.

Κάθε φορά που χρονοδρομολογούμε ένα σύνολο διεργασιών, συνήθως επιθυμούμε είτε να ελαχιστοποιήσουμε, είτε να μεγιστοποιήσουμε την τιμή μίας αντικειμενικής συνάρτησης. Αντιπροσωπευτικές αντικειμενικές συναρτήσεις δίνονται στη συνέχεια.

Makespan. Σε μία δεδομένη χρονοδρομολόγηση, συμβολίζουμε με C_j το *χρόνο ολοκλήρωσης (completion time)* για κάθε διεργασία j, $1 \leq j \leq n$, δηλαδή τη χρονική στιγμή που ολοκληρώνεται η εκτέλεση της j. Το makespan, C_{max}, ορίζεται ως,

$$C_{max} = \max\{C_1, C_2, \ldots, C_n\}.$$

και είναι ο χρόνος ολοκλήρωσης και της τελευταίας εκτελούμενης διεργασίας. Ένα πολύ σημαντικό πρόβλημα που ανακύπτει είναι το πρόβλημα ελαχιστοποίησης του Makespan. Στη χρονοδρομολόγηση του Σχήματος 1.2 οι χρόνοι ολοκλήρωσης $C_1, \ldots, C_7$ είναι ίσοι με $3, 5, 6, 8, 6, 7$ και 9 αντίστοιχα. Το makespan της χρονοδρόμολόγησης ισούται με 9.

Μέσος χρόνος ολοκλήρωσης (Total Completion Time). Ο Μέσος χρόνος ολοκλήρωσης μίας δεδομένης χρονοδρομολόγησης ορίζεται ως,

$$\sum_{j=1}^{n} C_j.$$

Συνήθως αναφερόμαστε στην αντικειμενική αυτή συνάρτηση ως ο *συνολικός χρόνος ολοκλήρωσης*. Στη χρονοδρομολόγηση του Σχήματος 1.2 ο συνολικός χρόνος ολοκλήρωσης ισούται με 44. Αν επιπλέον για κάθε διεργασία j δίνεται μαζί με το χρόνο επεξεργασίας της και κάποιο θετικό ακέραιο βάρος w_j, τότε ορίζουμε το μέσο βεβαρημένο χρόνο ολοκλήρωσης ως,

$$\sum_{j=1}^{n} w_j C_j.$$

Ένα ενδιαφέρον πρόβλημα είναι το πρόβλημα ελαχιστοποίησης του μέσου (βεβαρημένου) χρόνου ολοκλήρωσης.

Καθυστέρηση (Tardiness). Για κάθε διεργασία j, εκτός από το χρόνο επεξεργασίας p_j, δίνεται μία προθεσμία d_j. Επιπλέον, κάθε διεργασία j μπορεί να διαθέτει και κάποιο θετικό ακέραιο βάρος w_j. Η καθυστέρηση T_j μιας διεργασίας j σε μία δεδομένη χρονοδρομολόγηση ορίζεται ως,

$$T_j = \max\{0, C_j - d_j\}.$$

Τρείς αντικειμενικές συναρτήσεις που εμπεριέχουν την καθυστέρηση και συναντώνται συχνά σε προβλήματα χρονοδρομολόγησης είναι οι εξής:

1. $\sum_{j=1}^{n} w_j T_j$
2. $T_{max} = \max_{j=1,\ldots,n} \{T_j\}$.
3. $\sum_{j=1}^{n} w_j U_j$, όπου $U_j = \begin{cases} 1, & \text{αν } T_j > 0, \\ 0, & \text{αλλιώς.} \end{cases}$

Από την πρώτη ανακύπτει το πρόβλημα ελαχιστοποίησης της συνολικής βεβαρημένης καθυστέρησης, από την δεύτερη το πρόβλημα ελαχιστοποίησης της μέγιστης καθυστέρησης, ενώ από την τρίτη έχουμε το πρόβλημα ελαχιστοποίησης του συνολικού βάρους των καθυστερημένων διεργασιών (tardy jobs). Δύο συναφείς με την καθυστέρηση αντικειμενικές συναρτήσεις είναι η *αργοπορία (Lateness)* και η *πρόωρη ολοκλήρωση (earliness)*. Σε μία δεδομένη χρονοδρομολόγηση η αργοπορία μίας διεργασίας j ορίζεται ως $L_j = C_j - d_j$, ενώ η πρόωρη ολοκλήρωση αυτής ορίζεται ως $E_j = \max\{d_j - C_j, 0\}$. Τρία γνωστά προβλήματα είναι τα προβλήματα ελαχιστοποίησης της μέγιστης αργοπορίας $L_{max} = \max_{j=1,\ldots,n}\{L_j\}$, μεγιστοποίησης της συνολικής βεβαρημένης πρόωρης ολοκλήρωσης $\sum_{j=1}^{n} w_j E_j$ και μεγιστοποίησης του συνολικού βάρους των πρόωρων διεργασιών (early jobs) $\sum_{j=1}^{n} w_j U'_j$, όπου $U'_j = 1 - U_j$.

Μπορούμε να ορίσουμε συνοπτικά κάθε πρόβλημα χρονοδρομολόγησης χρησιμοποιώντας την πρότυπη παράσταση τριών πεδίων $\alpha|\beta|\gamma$ (βλ. Graham, Lawler, Lenstra, Kan [33] και Lawler,

Lenstra, Kan, Shmoys [62]), όπου το πεδίο α καθορίζει το περιβάλλον των μηχανών, το πεδίο β καθορίζει τις ιδιότητες των διεργασιών και το πεδίο γ καθορίζει την αντικειμενική συνάρτηση. Αν για παράδειγμα θεωρήσουμε ότι διαθέτουμε m παράλληλες πανομοιότυπες μηχανές, τότε για σταθερό m το πρόβλημα ελαχιστοποίησης του makespan ορίζεται ως $Pm \parallel C_{max}$, αλλιώς συμβολίζεται με $P \parallel C_{max}$. Για το ίδιο περιβάλλον μηχανών τα προβλήματα ελαχιστοποίησης του συνολικού (βεβαρημένου) χρόνου ολοκλήρωσης και της συνολικής καθυστέρησης ορίζονται ως $P \parallel \sum_{j=1}^{n} w_j C_j$ και $P \parallel \sum_{j=1}^{n} T_j$ αντίστοιχα. Αν το περιβάλλον των μηχανών είναι η Μοναδική Μηχανή, τότε το α ισούται με 1.

Στις ενότητες που ακολουθούν χρησιμοποιούμε κάποιες βασικές έννοιες και αποτελέσματα από τη Θεωρία Πολυπλοκότητας. Για την κατανόησή τους ο αναγνώστης παραπέμπεται στο Παράρτημα Α, όπου γίνεται μία σύντομη επισκόπηση της Θεωρίας Πολυπλοκότητας.

1.2 Κανόνες χρονοδρομολόγησης

Η διατύπωση της Θεωρίας της **NP**-πληρότητας και τα επακόλουθα αποτελέσματα αυτής, έδειξαν ότι τα περισσότερα κεντρικά προβλήματα χρονοδρομολόγησης δεν είναι εφικτό να επιλυθούν με αποδοτικό τρόπο, εκτός και αν **P** = **NP**. Οι αλγόριθμοι που παρουσιάστηκαν και παρουσιάζονται κατά καιρούς για προβλήματα χρονοδρομολόγησης, είναι στην πλειοψηφία τους προσεγγιστικοί αλγόριθμοι.

Η πιο διαδεδομένη κλάση αλγορίθμων για την επίλυση προβλημάτων χρονοδρομολόγησης είναι η κλάση των List Scheduling αλγορίθμων: οι διεργασίες ταξινομούνται σε μία λίστα βάσει ενός συγκεκριμένου κανόνα και η επόμενη στη λίστα διεργασία δρομολογείται στην πρώτη μηχανή που είναι διαθέσιμη. Μερικοί γνωστοί κανόνες περιγράφονται στη συνέχεια.

Αυθαίρετη Λίστα (Arbitrary List). Η ταξινόμηση γίνεται με αυθαίρετη μετάθεση των διεργασιών.

Μεγαλύτερος Χρόνος Επεξεργασίας (Longest Processing Time ή LPT). Οι διεργασίες ταξινομούνται σε φθίνουσα σειρά ως προς το χρόνο επεξεργασίας τους. Κάθε στιγμή που μία μηχανή είναι διαθέσιμη, δρομολογείται σ΄ αυτή η διεργασία με το μεγαλύτερο χρόνο επεξεργασίας που βρίσκεται στην τρέχουσα λίστα. Ο αλγόριθμος αυτός αποτελεί μία ευρετική μέθοδο για την επίλυση του προβλήματος $P \parallel C_{max}$ (βλ. [35]). Δρομολογεί πρώτα τις διεργασίες που έχουν μεγάλο χρόνο επεξεργασίας (μεγάλες διεργασίες), έτσι ώστε να αποφύγει την εκτέλεση αυτών προς το τέλος της χρονοδρομολόγησης, γεγονός που μπορεί να προκαλέσει σημαντική αύξηση στο makespan.

Μικρότερος Χρόνος Επεξεργασίας (Shortest Processing Time ή SPT). Οι διεργασίες ταξι-

νομούνται σε αύξουσα σειρά ως προς το χρόνο επεξεργασίας τους και όταν μία μηχανή είναι διαθέσιμη, δρομολογείται σ΄ αυτή η διεργασία με το μικρότερο χρόνο επεξεργασίας που βρίσκεται στην τρέχουσα λίστα. Ο αλγόριθμος αυτός επιλύει βέλτιστα σε πολυωνυμικό χρόνο τα προβλήματα $1 \parallel \sum_{j=1}^{n} C_j$ [92], $P \parallel \sum_{j=1}^{n} C_j$ [19].

Κανόνας του Smith. Πρόκειται για μία παραλλαγή του κανόνα Μικρότερου Χρόνου Επεξεργασίας, όπου οι διεργασίες διαθέτουν βάρη και ταξινομούνται σε αύξουσα σειρά, ως προς το λόγο του χρόνου επεξεργασίας τους p_j προς το αντίστοιχο βάρος τους w_j. Συγκεκριμένα οι διεργασίες ταξινομούνται έτσι ώστε να ισχύει η ακόλουθη ανισότητα,

$$\frac{p_1}{w_1} \leq \frac{p_2}{w_2} \leq \ldots \leq \frac{p_n}{w_n}.$$

Ο αλγόριθμος αυτός βρίσκει μία βέλτιστη λύση για το πρόβλημα $1 \parallel \sum_{j=1}^{n} w_j C_j$ [92]. Αξίζει να σημειωθεί ότι η διάταξη των διεργασιών στον κανόνα του Smith ισοδυναμεί με τη φθίνουσα διάταξη των διεργασιών ως προς το λόγο τους w_j/p_j. Η τελευταία είναι γνωστή ως κανόνας WSPT (Weighted Shortest Processing Time) και συναντάται σε αρκετές εργασίες αντί του κανόνα του Smith.

Μικρότερη Προθεσμία (Earliest Due Date ή EDD). Οι διεργασίες ταξινομούνται σε αύξουσα σειρά ως προς την προθεσμία τους d_j, οπότε εκτελείται πρώτη η διεργασία με τη μικρότερη προθεσμία και τελευταία αυτή με το μεγαλύτερη. Ο αλγόριθμος επιλύει βέλτιστα τα προβλήματα $1 \parallel T_{max}$ και $1 \parallel L_{max}$ (βλ. [46, 92]).

Στη συνέχεια θα δώσουμε ένα παράδειγμα εκτέλεσης ενός List Scheduling αλγορίθμου για το πρόβλημα $P \parallel C_{max}$.

Το πρόβλημα $P \parallel C_{max}$ είναι ένα κεντρικό πρόβλημα της θεωρίας προβλημάτων χρονοδρομολόγησης και ο αλγόριθμος που ακολουθεί παρουσιάστηκε από τον Graham το 1966 [34]. Πρόκειται για ένα List Scheduling αλγόριθμο ο οποίος εφαρμόζει τον κανόνα της Αυθαίρετης Λίστας.

Αλγόριθμος 1 Ένας 2-προσεγγιστικός αλγόριθμος για το πρόβλημα $P \parallel C_{max}$

1. Ταξινόμησε τις διεργασίες με βάση τον κανόνα της Αυθαίρετης Λίστας.
2. Δρομολόγησε τις διεργασίες στις μηχανές αναθέτοντας την επόμενη διεργασία στη μηχανή με το μικρότερο τρέχον φορτίο.

Για μία δεδομένη χρονοδρομολόγηση με τον όρο φορτίο μιας μηχανής M_i εννοούμε την ποσότητα $\sum_{j \in J'} p_j$, όπου J' είναι το σύνολο των διεργασιών που εκτελούνται στη μηχανή M_i.

Ο Αλγόριθμος 1 βασίζεται στα ακόλουθα δύο κάτω φράγματα για την τιμή του ελαχίστου makespan, $\mathrm{OPT}(I, m)$:

1. Μέσος χρόνος εκτέλεσης διεργασιών για μία μηχανή, $\sum_j p_j$ και

2. μέγιστος χρόνος επεξεργασίας, $p_{max} = \max_j\{p_j\}$.

Συνδυάζοντας τα δύο κάτω φράγματα θέτουμε,

$$LB = \max\left\{\frac{1}{m}\sum_j p_j,\ \ p_{max}\right\}.$$

Είναι εύκολο να αποδείξουμε ότι ο Αλγόριθμος 1, για οποιαδήποτε είσοδο I, επιστρέφει μία λύση με τιμή μικρότερη ή ίση από τη διπλάσια τιμή της βέλτιστης λύσης $\mathrm{OPT}(I, m)$.

Θεώρημα 1.1 *Ο Αλγόριθμος 1 πετυχαίνει παράγοντα προσέγγισης* 2 *για το πρόβλημα* $P \parallel C_{max}$.

Απόδειξη. Αρχικά παρατηρούμε ότι το LB είναι επίσης κάτω φράγμα για το $\mathrm{OPT}(I, m)$, $LB \leq \mathrm{OPT}(I, m)$.

Βάσει της χρονοδρομολόγησης του Αλγορίθμου 1, το μικρότερο τρέχον φορτίο L_i, πριν τη δρομολόγηση και της τελευταίας διεργασίας θα είναι,

$$L_i \leq \frac{1}{m}\sum_j p_j \leq LB.$$

Η δρομολόγηση της τελευταίας διεργασίας προσθέτει στο L_i χρόνο p_i, όπου $p_i \leq p_{max} \leq LB$. Συνεπώς έχουμε ότι,

$$L_i + p_j \leq LB + LB \leq 2\,\mathrm{OPT}(I, m).$$

Οπότε αποδεικνύεται ότι ο Αλγόριθμος 1 είναι 2-προσεγγιστικός. □

Από το Θεώρημα 1.1 γίνεται σαφές ότι ο Αλγόριθμος 1 δεν εγγυάται την εύρεση της βέλτιστης λύσης για κάθε στιγμιότυπο του προβλήματος $P \parallel C_{max}$. Συγκεκριμένα πρόκειται για ένα 2-προσεγγιστικό αλγόριθμο πολυωνυμικού χρόνου. Αυτό βέβαια σε καμία περίπτωση δεν αποδεικνύει ότι δεν υπάρχει αλγόριθμος που να βρίσκει τη βέλτιστη λύση στο πρόβλημα $P \parallel C_{max}$. Όπως προέκυψε τελικά από την Θεωρία της **NP**-πληρότητας[1] το πρόβλημα $P \parallel C_{max}$ είναι **NP**-πλήρες πρόβλημα και μάλιστα strongly **NP**-πλήρες [30], γεγονός που καθιστά αδύνατη, εκτός και αν **P** = **NP**, τόσο την αναζήτηση ενός βέλτιστου πολυωνυμικού αλγορίθμου όσο και την αναζήτηση ενός FPTAS.

[1] Ο αλγόριθμος του Graham προηγείται χρονικά της διατύπωσης της θεωρίας της **NP**-πληρότητας και αποτελεί τον πρώτο προσεγγιστικό αλγόριθμο που παρουσιάστηκε για πρόβλημα βελτιστοποίησης.

Σε επόμενη εργασία του ο Graham [35] τροποποίησε τον Αλγόριθμο 1 εφαρμόζοντας τον κανόνα ταξινόμησης LPT. Ως αποτέλεσμα προέκυψε η προσέγγιση της τιμής OPT(I, m) με έναν παράγοντα $4/3$, σαφώς καλύτερο από τον παράγοντα 2 του Αλγορίθμου 1.

Στα επόμενα κεφάλαια θα αναφερθούμε σε αρκετά παραδείγματα προβλημάτων χρονοδρομολόγησης όπου εφαρμόζεται κάποιος από τους παραπάνω κανόνες ταξινόμησης για την επίλυση τους.

1.3 Σκιαγράφηση της μελέτης

Στα επόμενα κεφάλαια μελετώνται προσεγγιστικά σχήματα και βέλτιστοι ψευδοπολυωνυμικοί αλγόριθμοι για προβλήματα χρονοδρομολόγησης που έχουν ως στόχο τη βελτιστοποίηση των εξής αντικειμενικών συναρτήσεων: makespan (C_{max}), μέσος βεβαρημένος χρόνος ολοκλήρωσης ($\sum_{j=1}^{n} w_j C_j$), συνολικό βάρος των καθυστερημένων διεργασιών ($\sum_{j=1}^{n} w_j U_j$), συνολική βεβαρημένη πρόωρη ολοκλήρωση ($\sum_{j=1}^{n} w_j E_j$) και συνολική βεβαρημένη καθυστέρηση ($\sum_{j=1}^{n} w_j T_j$).

Στο Κεφάλαιο 2 παρουσιάζουμε προσεγγιστικά σχήματα πολυωνυμικού χρόνου για προβλήματα που έχουν στόχο την ελαχιστοποίηση του makespan. Αρχικά παρουσιάζουμε ένα PTAS για το πρόβλημα $P2 \parallel C_{max}$, όπου το πλήθος των μηχανών είναι σταθερό και ίσο με 2 [86]. Το αποτέλεσμα αυτό προκύπτει μέσω της εφαρμογής μίας τεχνικής η οποία είναι γνωστή ως τεχνική μετασχηματισμού των δεδομένων της εισόδου (βλ. Ενότητα 2.2). Στη συνέχεια, δίνουμε ένα PTAS για το πρόβλημα $P \parallel C_{max}$ [39], το οποίο στηρίζεται στην μέθοδο προσέγγισης μέσω ενός δυικού προβλήματος, όπως είναι το πρόβλημα Bin Packing (βλ. A.2.1). Επιπλέον, παρουσιάζουμε ένα δεύτερο PTAS για το ίδιο πρόβλημα (βλ. Schuurman και Woeginger [86]), το οποίο προκύπτει επίσης με εφαρμογή της τεχνικής μετασχηματισμού των δεδομένων της εισόδου.

Στο Κεφάλαιο 3 παρουσιάζουμε προσεγγιστικά σχήματα για διάφορες εκδοχές του προβλήματος ελαχιστοποίησης του μέσου βεβαρημένου χρόνου ολοκλήρωσης. Αρχικά δίνουμε ένα FPTAS για το πρόβλημα $P2 \parallel \sum_{j=1}^{n} w_j C_j$ [86], το οποίο βασίζεται στην εφαρμογή μίας τεχνικής που είναι γνωστή ως τεχνική αποκοπής (trimming technique) (βλ. Ενότητα 3.2). Στη συνέχεια θεωρούμε ότι οι διεργασίες ενός στιγμιοτύπου εισόδου, εκτός από το βάρος και το χρόνο επεξεργασίας τους, διαθέτουν επιπλέον και κάποιο χρόνο αποδέσμευσης και μελετάμε τις preemptive και non-preemptive εκδοχές του προβλήματος. Παρουσιάζουμε ένα PTAS για την απλή εκδοχή του προβλήματος όπου το περιβάλλον μήχανής είναι η Μοναδική Μηχανή και τα βάρη των διεργασιών είναι μοναδιαία, $1 \mid r_j \mid \sum_j C_j$ [2] και ακολούθως δίνουμε δύο PTAS για τις πιο γενικές εκδοχές, $P \mid r_j \mid \sum_j w_j C_j$ και $P \mid r_j,\ pmtn \mid \sum_j w_j C_j$ [13, 2]. Η βασική ιδέα για το σχεδιασμό των προσεγγιστικών αυτών σχημάτων είναι η εφαρμογή πολλαπλών μετασχηματισμών στο αρχικό στιγμιότυπο

του προβλήματος έτσι ώστε να είναι εφικτή η αποδοτική επίλυση του είτε με χρήση δυναμικού προγραμματισμού, είτε με εφαρμογή κάποιου κανόνα χρονοδρομολόγησης.

Στο Κεφάλαιο 4 μελετάμε προβλήματα χρονοδρομολόγησης στα οποία οι διεργασίες διαθέτουν προκαθορισμένες προθεσμίες και ο στόχος είναι η ελαχιστοποίηση (ή μεγιστοποίηση) κάποιας συνάρτησης απώλειας. Η έννοια της απώλειας σε μία δεδομένη χρονοδρομολόγηση ερμηνεύεται ως η αποτυχία ολοκλήρωσης της εκτέλεσης μίας διεργασίας στη δεδομένη προθεσμία της. Οι συναρτήσεις απώλειας που μας ενδιαφέρουν είναι το συνολικό βάρος των καθυστερημένων διεργασιών, η συνολική βεβαρημένη πρόωρη ολοκλήρωση και η συνολική βεβαρημένη καθυστέρηση. Αρχικά παρουσιάζουμε αλγορίθμους ψευδοπολυωνυμικού χρόνου για τα προβλήματα $1 \parallel \sum_j w_j U_j$, $1 \parallel \textit{maximize} \sum_j w_j E_j$ και $1 \mid d_j = d \mid \sum_j w_j T_j$ (βλ. Lawler και Moore [63]). Οι αλγόριθμοι αυτοί προκύπτουν μέσω κατάλληλης εφαρμογής ενός δυναμικού προγράμματος σε καθένα από τα αντίστοιχα προβλήματα. Στη συνέχεια παρουσιάζουμε έναν ψευδοπολυωνυμικό αλγόριθμο δυναμικού προγραμματισμού για το πρόβλημα $1 \mid w_j \textit{ agreeable} \mid \sum_j w_j T_j$, όπου τα βάρη των διεργασιών είναι συμβατά, υπό την έννοια ότι αν $p_i < p_j$ τότε $w_i \geq w_j$ (βλ. Lawler [60]). Χρησιμοποιώντας τους παραπάνω αλγόριθμους δίνουμε ένα FPTAS για το πρόβλημα $1 \parallel \sum_j w_j U_j$ [86] και ένα FPTAS για το πρόβλημα $1 \parallel \sum_j T_j$ [61]. Το πρώτο προκύπτει με εφαρμογή της τεχνικής αποκοπής, όπως ακριβώς και το αντίστοιχο αποτέλεσμα για το πρόβλημα $P2 \parallel \sum_{j=1}^{n} w_j C_j$, ενώ το δεύτερο προκύπτει μέσω κατάλληλων μετασχηματισμών των αριθμητικών δεδομένων της εισόδου.

Κεφάλαιο 2

Το πρόβλημα ελαχιστοποίησης του makespan

Το πρόβλημα της ελαχιστοποίησης του makespan αποτελεί ένα κεντρικό πρόβλημα της Θεωρίας Χρονοδρομολόγησης. Όπως είδαμε στο Κεφάλαιο 1 (βλ. Ενότητα 1.1), για δεδομένη αντικειμενική συνάρτηση, όπως είναι το makespan, μπορούμε να κατασκευάσουμε προβλήματα που διαφέρουν μεταξύ τους, είτε ως προς το περιβάλλον των μηχανών, είτε ως προς τις ιδιότητες των διεργασιών. Στις επόμενες ενότητες θα ασχοληθούμε εν γένει με αποτελέσματα που αφορούν την εκδοχή $P \parallel C_{max}$ του προβλήματος ελαχιστοποίησης του makespan, η οποία έχει ίσως διερευνηθεί περισσότερο από κάθε άλλη. Το πρόβλημα ορίζεται τυπικά ως ακολούθως.

Η είσοδος του προβλήματος αποτελείται από n διεργασίες και m παράλληλες, πανομοιότυπες μηχανές. Κάθε διεργασία j έχει έναν θετικό ακέραιο χρόνο επεξεργασίας p_j και όλες οι διεργασίες είναι διαθέσιμες τη χρονική στιγμή μηδέν. Ο στόχος είναι να χρονοδρομολογήσουμε τις διεργασίες στις m μηχανές, έτσι ώστε να ελαχιστοποιήσουμε το χρόνο ολοκλήρωσης της τελευταίας διεργασίας που δρομολογείται. Ο χρόνος αυτός καλείται makespan και συμβολίζεται με C_{max}. Η τιμή της βέλτιστης λύσης ενός στιγμιοτύπου του προβλήματος συμβολίζεται με $\mathrm{OPT}(I, m)$, όπου I είναι το σύνολο των χρόνων επεξεργασίας και m το πλήθος των μηχανών. Υπενθυμίζουμε ότι στην ειδική περίπτωση όπου ο αριθμός των μηχανών είναι σταθερός και ίσος με m το πρόβλημα συμβολίζεται με $Pm \parallel C_{max}$, ενώ η τιμή της βέλτιστης λύσης ενός στιγμιοτύπου του προβλήματος συμβολίζεται με $\mathrm{OPT}(I)$, αφού το m δεν αποτελεί μέρος της εισόδου.

2.1 Επισκόπηση γνωστών αποτελεσμάτων

Η μελέτη της υπολογιστικής πολυπλοκότητας του προβλήματος $P \parallel C_{max}$ απέδειξε ότι πρόκειται για ένα strongly **NP**-hard πρόβλημα [30]. Βάσει του Πορίσματος Α.1, το καλύτερο αποτέλεσμα που θα μπορούσε να περιμένει κανείς είναι ο σχεδιασμός ενός PTAS, εκτός και αν **P** = **NP**.

Το πρώτο PTAS για το πρόβλημα $P \parallel C_{max}$ δόθηκε από τους Hochbaum και Shmoys [39] και στηρίχτηκε σε αρκετά αποτελέσματα προσέγγισης που είχαν προηγηθεί: οι δύο αλγόριθμοι του Graham [34, 35], στους οποίους αναφερθήκαμε στην Ενότητα 1.2, αποτέλεσαν τους πρώτους προσεγγιστικούς αλγορίθμους για το πρόβλημα $P \parallel C_{max}$. Οι Coffman, Garey και Jhonson [25] παρουσίασαν έναν 1.22-προσεγγιστικό αλγόριθμο (MULTIFIT algorithm) στον οποίο συσχέτισαν το $P \parallel C_{max}$ με το συναφές πρόβλημα Bin Packing (βλ. Υποενότητα Α.2.1.2 για ορισμό). Στη συνέχεια οι Friesen [29] και Langston [56], χρησιμοποιώντας πιο περίπλοκες τεχνικές, βελτίωσαν την παραπάνω προσέγγιση σε 1.20 και 72/61 αντίστοιχα. Ένα FPTAS για το πρόβλημα $Pm \parallel C_{max}$ δόθηκε από τον Sahni [81], όμως ο χρόνος εκτέλεσης του ήταν εκθετικός ως προς το m, γεγονός που απορρίπτει την μετατροπή σε ένα προσεγγιστικό σχήμα για την γενική διάσταση του προβλήματος. Το PTAS των Hochbaum και Shmoys [39], προέκυψε τελικά μέσα από κατάλληλο συσχετισμό των προβλημάτων Bin Packing και $P \parallel C_{max}$ (βλ. Ενότητα 2.4.1). Εκ των υστέρων παρουσιάστηκαν PTAS για το ίδιο πρόβλημα, στα οποία δεν χρησιμοποιήθηκε η παραπάνω συσχέτιση. Ένα τέτοιο αποτέλεσμα, οφείλεται στους Schuurman και Woeginger [86] (βλ. Ενότητα 2.4.2).

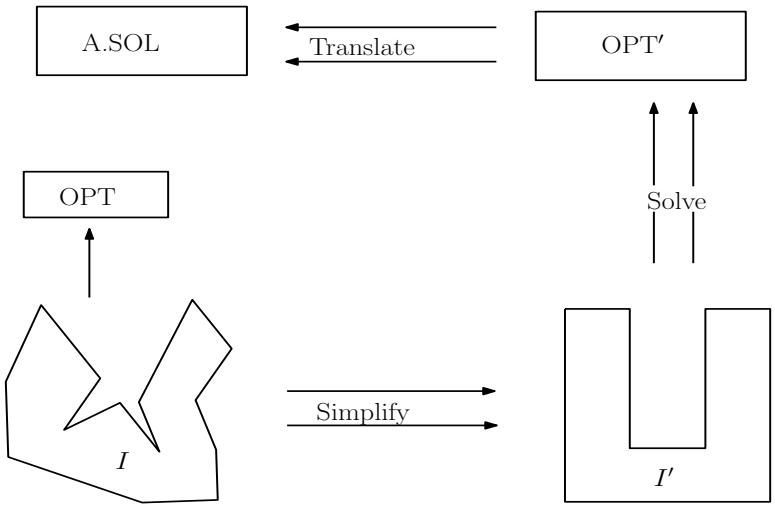
Στην συνέχεια παρουσιάζουμε μία γενική τεχνική σχεδιασμού PTAS, στην οποία στηρίζονται εν γένει τα αποτελέσματα που θα ακολουθήσουν.

2.2 Μία γενική τεχνική σχεδιασμού PTAS

Μία διαδεδομένη τεχνική για την κατασκευή προσεγγιστικών σχημάτων για **NP**-hard προβλήματα βελτιστοποίησης είναι η *τεχνική μετασχηματισμού των δεδομένων της εισόδου*.

Η βασική ιδέα της τεχνικής αυτής είναι να μετασχηματίσουμε ένα "δύσκολο" στιγμιότυπο σε ένα στιγμιότυπο το οποίο μπορεί να επιλυθεί αποδοτικά. Χρησιμοποιώντας τη βέλτιστη λύση αυτού του στιγμιοτύπου, παίρνουμε μία προσεγγιστική λύση για το αρχικό στιγμιότυπο. Τα τρία βήματα που συνθέτουν την τεχνική αυτή απεικονίζονται στο Σχήμα 2.1.

Το Βήμα της απλοποίησης (Simplify) του στιγμιοτύπου I σε ένα στιγμιότυπο I' εξαρτάται από την επιθυμητή ακρίβεια ϵ της προσέγγισης. Όσο πιο κοντά στο 0 είναι το ϵ τόσο μεγαλύτερη είναι η ομοιότητα του στιγμιοτύπου I' με το στιγμιότυπο I. Επιπλέον, ο χρόνος που απαιτείται για την απλοποίηση πρέπει να είναι πολυωνυμικός ως προς το μέγεθος της εισόδου.



Σχήμα 2.1: Η τεχνική μετασχηματισμού των δεδομένων της εισόδου.

Το Βήμα της επίλυσης (Solve) του απλοποιημένου στιγμιοτύπου I' αφορά στην εύρεση μιας βέλτιστης λύσης OPT′ για το I' σε πολυωνυμικό χρόνο.

Στο Βήμα της προσαρμογής (Translate) προσαρμόζουμε τη λύση OPT′ του I' στα αρχικά δεδομένα και παίρνουμε μία προσεγγιστική λύση A.SOL για το στιγμιότυπο I. Η προσαρμογή αυτή εκμεταλλεύεται την ομοιότητα μεταξύ των στιγμιοτύπων I και I' και στην ιδανική περίπτωση το A.SOL θα παραμένει "κοντά" στο OPT′, το οποίο θα βρίσκεται "κοντά" στο OPT. Στην περίπτωση αυτή, έχουμε βρει την επιθυμητή προσέγγιση, δηλαδή ένα PTAS.

Το δυσκολότερο βήμα της παραπάνω τεχνικής είναι ο προσδιορισμός της κατάλληλης απλοποίησης. Για παράδειγμα, αν το στιγμιότυπο I' επιλεχθεί πολύ "κοντά" στο I, τότε το I' μπορεί να παραμείνει **NP**-hard. Αντιθέτως, αν το I' επιλεχθεί πολύ "μακριά" από το I, τότε η επίλυση του I' μπορεί να μη βοηθήσει στην επίλυση του I. Για την απλοποίηση των δεδομένων της εισόδου έχουν προταθεί διάφορες μέθοδοι. Η *στρογγυλοποίηση (rounding)* κάποιων από τους αριθμούς της εισόδου (π.χ. σε τέλειες δυνάμεις του 2) είναι ο πιο απλός τρόπος για να μετασχηματίσουμε τα δεδομένα της εισόδου. Ένας άλλος τρόπος είναι *η συνένωση (merging)* μικρών τμημάτων σε μεγαλύτερα (π.χ. πολλές διεργασίες μικρού χρόνου επεξεργασίας σε μία διεργασία ίσου χρόνου). Μπορούμε ακόμη να κάνουμε *αφαίρεση (cutting)* κάποιων τμημάτων της εισόδου, αλλά και *ευθυγράμμιση (aligning)* των μεγεθών παρόμοιων μεταξύ τους αντικειμένων (π.χ. να αντικαταστήσουμε διαφορετικές διεργασίες που έχουν περίπου ίσους χρόνους επεξεργασίας, με πανομοιότυπα αντίγραφα που έχουν χρόνο επεξεργασίας ίσο με τον μέσο χρόνο των αρχικών).

Η τεχνική μετασχηματισμού των δεδομένων της εισόδου παρουσιάστηκε αρχικά από τους Horowitz και Sahni [43], οι οποίοι την εφάρμοσαν σε προβλήματα διαμέρισης (partition problems). Σε επόμενη εργασία του, ο Sahni [81] εφάρμοσε την τεχνική αυτή για τον σχεδιασμό ενός FPTAS για το πρόβλημα $P2 \parallel C_{max}$. Άλλες εφαρμογές της τεχνικής αυτής βρίσκουμε στην εργασία των

Hochbaum και Shmoys [39] για το πρόβλημα $P \parallel C_{max}$ (βλ. Ενότητα 2.4.1), καθώς και στην εργασία των Fernadez de la Vega και Lueker [23] για το πρόβλημα Bin Packing.

Για περισσότερες αναφορές σε εφαρμογές της παραπάνω τεχνικής ο αναγνώστης παραπέμπεται στην εργασία των Schuurman και Woeginger [86]. Στην ίδια εργασία, γίνεται επίσης και μία προσπάθεια ομαδοποίησης όλων των γνωστών μεθόδων για τον σχεδιασμό PTAS ή FPTAS, σε τρείς συγκεκριμένες τεχνικές.

Στην ενότητα που ακολουθεί παρουσιάζουμε ένα PTAS για το πρόβλημα $P2 \parallel C_{max}$ [86].

2.3 Ελαχιστοποίηση του makespan σε δύο παράλληλες πανομοιότυπες μηχανές

Στο πρόβλημα $P2 \parallel C_{max}$, το πλήθος των μηχανών είναι σταθερό και ίσο με 2 και δεν αποτελεί μέρος της εισόδου του προβλήματος. Πρόκειται δηλαδή για ειδική περίπτωση τόσο του προβλήματος $P \parallel C_{max}$, όσο και του προβλήματος $Pm \parallel C_{max}$ (για $m > 2$). Επιπλέον, το πρόβλημα $P2 \parallel C_{max}$, σε αντίθεση με το $P \parallel C_{max}$, είναι **NP**-hard υπό τη συνήθη έννοια [49] και αυτό καθιστά εφικτό το σχεδιασμό τόσο ενός PTAS, όσο και ενός FPTAS (βλ. Πόρισμα A.1).

Στη συνέχεια παρουσιάζουμε ένα PTAS για το $P2 \parallel C_{max}$ που βασίζεται στην τεχνική μετασχηματισμού των δεδομένων της εισόδου και σχεδιάστηκε από τους Schuurman και Woeginger [86].

Θεωρούμε αρχικά ένα στιγμιότυπο I του προβλήματος και συμβολίζουμε με $\mathrm{OPT}(I)$ την βέλτιστη τιμή του makespan γι' αυτό. Θυμίζουμε από τον Αλγόριθμο 1 ότι το $LB = \max\{\frac{1}{2}\sum_j p_j,\ p_{max}\}$, για $m = 2$, είναι κάτω φράγμα για το $\mathrm{OPT}(I)$ και επομένως ισχύει ότι:

$$LB \ \leq \ \mathrm{OPT}(I). \tag{2.1}$$

Ο σχεδιασμός του PTAS συνοψίζεται στα ακόλουθα τρία στάδια.

Στάδιο Α. Απλοποιούμε το στιγμιότυπο I σε ένα στιγμιότυπο I'. Οι διεργασίες του I διαχωρίζονται βάσει μίας παραμέτρου $\epsilon > 0$, σε *μικρές* και *μεγάλες*:

1. Κάθε *μεγάλη* διεργασία έχει χρόνο επεξεργασίας $p_j > \epsilon LB$ και όλες οι μεγάλες διεργασίες του I, περιέχονται στο I'.

2. Κάθε *μικρή* διεργασία έχει χρόνο επεξεργασίας $p_j \leq \epsilon LB$. Το στιγμιότυπο I' περιέχει ακριβώς $\lfloor S/(\epsilon LB) \rfloor$ διεργασίες, με χρόνο επεξεργασίας ϵLB η καθεμία. Με S συμβολίζουμε το συνολικό χρόνο επεξεργασίας των μικρών διεργασιών στο I.

Η διαφορά μεταξύ των στιγμιοτύπων I, I', είναι προφανές ότι έγκειται στις μικρές διεργασίες

που αυτά περιέχουν. Έτσι, έχει ενδιαφέρον να δούμε πόσο αυτή η διαφορά επηρεάζει την τιμή της βέλτιστης λύσης για το αρχικό στιγμιότυπο I.

Θεωρούμε μία βέλτιστη χρονοδρομολόγηση για το I, με makespan $\mathrm{OPT}(I)$. Με S_1 συμβολίζουμε το συνολικό χρόνο επεξεργασίας (ή φορτίο) των μικρών διεργασιών στην πρώτη μηχανή M_1 και με S_2 το αντίστοιχο φορτίο στη δεύτερη μηχανή M_2. Σε κάθε μηχανή, αφήνουμε ως έχουν τις μεγάλες διεργασίες και αντικαθιστούμε τις μικρές με $\lceil S_i/(\epsilon LB)\rceil$, $i = 1, 2$, διεργασίες μεγέθους[1] ϵLB η καθεμία. Εύκολα παρατηρούμε ότι:

$$\left\lceil \frac{S_i}{\epsilon LB} \right\rceil \epsilon LB - S_i \leq \left(\frac{S_i}{\epsilon LB} + 1 \right) \epsilon LB - S_i = \epsilon LB. \tag{2.2}$$

Από τη (2.2) προκύπτει ότι η αύξηση του φορτίου κάθε μηχανής λόγω της αντικατάστασης των μικρών διεργασιών είναι το πολύ ϵLB. Επιπλέον η χρονοδρομολόγηση αυτή είναι εφικτή για το στιγμιότυπο I'. Συνεπώς θα ισχύει ότι:

$$\mathrm{OPT}(I') \;\leq\; \mathrm{OPT}(I) + \epsilon LB \;\leq\; (1+\epsilon)\,\mathrm{OPT}(I). \tag{2.3}$$

Στάδιο Β. Παρατηρούμε ότι η αντικατάσταση των μικρών διεργασιών του στιγμιοτύπου I δεν προκαλεί αύξηση στο συνολικό χρόνο επεξεργασίας των διεργασιών. Επομένως έχουμε ότι $\sum_j p_j \leq 2LB$, για το στιγμιότυπο I'. Επιπλέον, αφού κάθε διεργασία του I' έχει μέγεθος τουλάχιστον ϵLB, θα υπάρχουν το πολύ $2/\epsilon$ διεργασίες σ'αυτό.

Κάθε μία από τις $2/\epsilon$ διεργασίες μπορεί να ανατεθεί προς εκτέλεση σε μία από τις δύο μηχανές που υπάρχουν και έτσι δημιουργούνται το πολύ $2^{2/\epsilon}$ διαφορετικές εφικτές χρονοδρομολογήσεις των διεργασιών. Το makespan κάθε τέτοιας χρονοδρομολόγησης υπολογίζεται σε χρόνο $O(2/\epsilon)$ και επομένως, το στιγμιότυπο I' επιλύεται σε σταθερό χρόνο (φυσικά πρόκειται για αρκετά μεγάλη σταθερά).

Στάδιο Γ. Θεωρούμε μία βέλτιστη χρονοδρομολόγηση A' για το στιγμιότυπο I' στην οποία ο συνολικός χρόνος επεξεργασίας των νέων μικρών διεργασιών στη μηχανή M_i, είναι S'_i, $i = 1, 2$ και ο αντίστοιχος χρόνος για τις μεγάλες διεργασίες είναι B'_i (Όπου $B'_i = B_i$). Τότε θα ισχύει ότι:

$$S'_1 + S'_2 = \epsilon LB \left\lfloor \frac{S}{\epsilon LB} \right\rfloor > S - \epsilon LB. \tag{2.4}$$

Κατασκευάζουμε μία χρονοδρομολόγηση A για το αρχικό στιγμιότυπο I ως εξής: κάθε μεγάλη διεργασία τοποθετείται στην μηχανή που βρισκόταν στην χρονοδρομολόγηση A'. Για τις μικρές

[1] Ο όρος μέγεθος για μία διεργασία αναφέρεται στον αντίστοιχο χρόνο επεξεργασίας αυτής.

διεργασίες δεσμεύουμε χώρο $S'_1 + 2\epsilon LB$ και S'_2 στις μηχανές M_1 και M_2 αντίστοιχα. Αρχίζουμε να εκτελούμε μικρές διεργασίες στο δεσμευμένο χώρο της μηχανής M_1 μέχρι να βρεθεί κάποια που να μην χωράει σ' αυτόν. Τότε οι μικρές διεργασίες που εκτελέσθηκαν θα έχουν συνολικό μέγεθος τουλάχιστον $S'_1 + \epsilon LB$, ενώ διεργασίες συνολικού μεγέθους το πολύ $S - S'_1 - \epsilon LB$ θα έχουν απομείνει ανεκτέλεστες. Από τη (2.4) προκύπτει ότι οι ανεκτέλεστες αυτές διεργασίες χωράνε να εκτελεστούν στο χώρο S'_2.

Το συνολικό φορτίο L_i κάθε μηχανής στη χρονοδρομολόγηση A (με L'_i συμβολίζουμε το αντίστοιχο φορτίο μηχανής στη χρονοδρομολόγηση A') θα είναι:

$$L_i \leq B'_i + (S'_i + 2\epsilon LB) = L'_i + 2\epsilon LB \leq (1 + 3\epsilon)\,\mathrm{OPT}(I). \tag{2.5}$$

Οι ενδιάμεσες ανισότητες της (2.5) προέκυψαν από τις σχέσεις (2.1) και (2.3). Η (2.5) αποδεικνύει ότι το makespan της χρονοδρομολόγησης A είναι το πολύ $1 + 3\epsilon$ φορές μεγαλύτερο απ' αυτό της βέλτιστης χρονοδρομολόγησης για το αρχικό στιγμιότυπο I. Λόγω του ότι το 3ϵ μπορεί να πλησιάσει αυθαίρετα κοντά στο 0 και η πολυπλοκότητα χρόνου της μεθόδου είναι γραμμική στο n και εκθετική στο $1/\epsilon$ (βλ. Στάδιο Β), έχουμε σχεδιάσει το ζητούμενο PTAS για το πρόβλημα $P2 \parallel C_{max}$.

Αν εφαρμόσουμε το παραπάνω PTAS στην περίπτωση που το πλήθος των μηχανών είναι γενικό, έστω m, τότε η πολυπλοκότητα χρόνου θα γίνει εκθετική ως προς το m, λόγω του Σταδίου Β και επομένως, αφού το m είναι πλέον μέρος της εισόδου, δεν αποτελεί PTAS για το πρόβλημα $P \parallel C_{max}$. Αν δε το m είναι σταθερό και $m \geq 3$ τότε με πολύ μικρές αλλαγές στα Στάδια Α και Γ και για $LB = \max\{\frac{1}{m}\sum_j p_j,\ p_{max}\}$ έχουμε ένα PTAS για το πρόβλημα $Pm \parallel C_{max}$.

Όπως αναφέραμε στην αρχή της ενότητας το πρόβλημα $P2 \parallel C_{max}$ είναι **NP**-hard υπό τη συνήθη έννοια [49], οπότε η εύρεση ενός FPTAS είναι εφικτή. Ο σχεδιασμός ενός τέτοιου FPTAS μπορεί να γίνει με εφαρμογή κατάλληλου μετασχηματισμού σε έναν ακριβή αλγόριθμο ψευδοπολυωνυμικού χρόνου (βλ. Section 5 στο [86]). Ένα πανομοιότυπο αποτέλεσμα παρουσιάζεται στην Ενότητα 3.2 και αφορά στο πρόβλημα $P2 \parallel \sum_j w_j C_j$.

2.4 Ελαχιστοποίηση του makespan σε γενικό αριθμό παράλληλων πανομοιότυπων μηχανών

Στην ενότητα αυτή παρουσιάζουμε δύο διαφορετικά $PTAS$ για το πρόβλημα $P \parallel C_{max}$, τα οποία στηρίζονται εν γένει στην τεχνική μετασχηματισμού των δεδομένων της εισόδου.

Το πρώτο από αυτά παρουσιάζεται στην Ενότητα 2.4.1. Σχεδιάστηκε από τους Hochbaum και

Shmoys [39] και όπως αναφέραμε και στην Ενότητα 2.1, αποτελεί το πρώτο χρονικά $PTAS$ για το πρόβλημα. Η προσέγγιση του προβλήματος $P \parallel C_{max}$ γίνεται μέσω ενός κατά τρόπον δυικού προβλήματος, όπως είναι το πρόβλημα Bin Backing.

Το δεύτερο $PTAS$ παρουσιάζεται στην Ενότητα 2.4.2 και σχεδιάστηκε από τους Schuurman και Woeginger [86]. Σε αντίθεση με το πρώτο $PTAS$, για το σχεδιασμό του δεν αξιοποιείται η δυική σχέση με το πρόβλημα Bin Packing.

2.4.1 Προσέγγιση μέσω ενός δυικού προβλήματος

Ένα συναφές με το πρόβλημα $P \parallel C_{max}$ είναι το πρόβλημα Bin Packing. Εύκολα μπορούμε να παρατηρήσουμε ότι τα δύο προβλήματα έχουν ουσιαστικά το ίδιο πρόβλημα απόφασης. Θα περίμενε κανείς να είναι επίσης εύκολη η μεταφορά ενός αποτελέσματος προσέγγισης από το ένα πρόβλημα στο άλλο, εκτελώντας απλώς μία δυαδική αναζήτηση. Κάτι τέτοιο όμως φαντάζει εξαιρετικά δύσκολο όπως μπορούμε να διαπιστώσουμε στη συνέχεια.

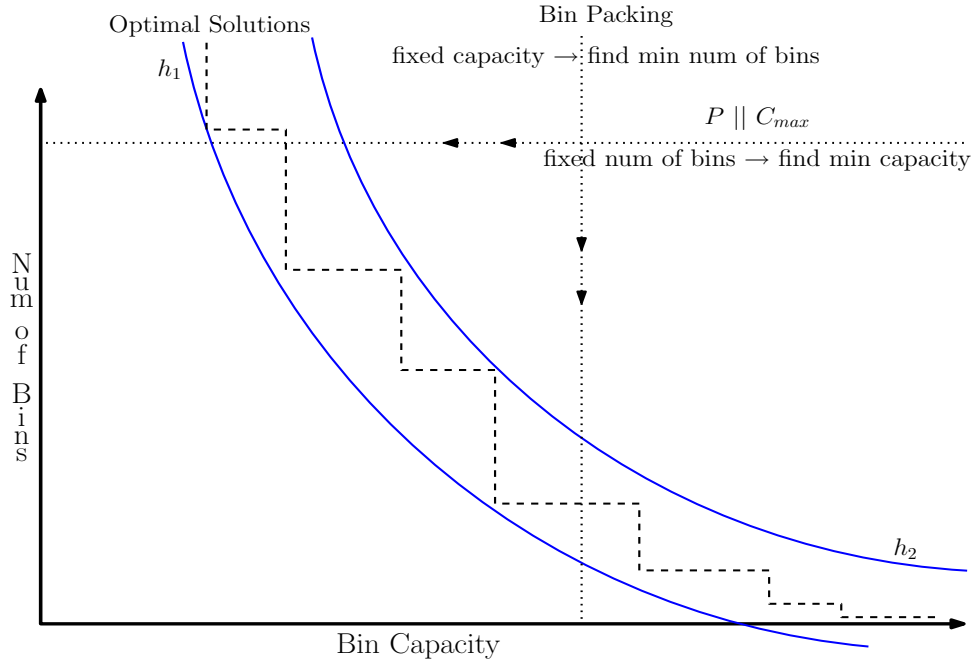
Το πρόβλημα Bin Packing είναι ένα strongly **NP**−hard πρόβλημα [73]. Επίσης από το Θεώρημα Α.3 δεν επιδέχεται $PTAS$, εκτός και αν **P** = **NP**. Το αποτέλεσμα αυτό αφορά σε στιγμιότυπα στα οποία η βέλτιστη λύση είναι μεγέθους 2 ή 3. Έτσι, όταν πρόκειται για "αντιπροσωπευτικά" στιγμιότυπα του προβλήματος όπου η τιμή της βέλτιστης λύσης αυξάνει συναρτήσει του n, τότε μπορούμε να κατασκευάσουμε ένα $PTAS^{\infty}$ (βλ. Ορισμό Α.6). Ένα $PTAS^{\infty}$ για το Bin Packing δόθηκε από τους Fernandez de la Vega και Lueker [23], ενώ οι Karmarkar και Karp [48] παρουσίασαν ένα FPTAS$^{\infty}$ για το ίδιο πρόβλημα. Σε ένα FPTAS$^{\infty}$ η λύση που παίρνουμε, αν πρόκειται για πρόβλημα ελαχιστοποίησης, έχει τιμή το πολύ $(1+\epsilon)\,\mathrm{OPT} + f(1/\epsilon)$, όπου f είναι μία πολυωνυμική συναρτηση και OPT η τιμή της βέλτιστης λύσης του προβλήματος. Βάσει των αποτελεσμάτων αυτών, ανακύπτει μία κρίσιμη διαφορά ανάμεσα στο πρόβλημα $P \parallel C_{max}$ και το πρόβλημα Bin Packing: στο $P \parallel C_{max}$ τα μεγέθη των διεργασιών μπορούν να μετασχηματιστούν (π.χ. στρογγυλοποιώντας προς τα πάνω) έτσι που το OPT να αυξάνεται χωρίς να επηρεάζεται η βασική δομή του προβλήματος. Μπορούμε επομένως να ελαττώσουμε στο 0 την τιμή της σταθεράς-συνάρτησης f, κατασκευάζοντας ένα FPTAS$^{\infty}$ με $f = 0$, δηλαδή ένα FPTAS. Όμως ένα FPTAS για το $P \parallel C_{max}$ δεν μπορεί να υπάρχει, εκτός και αν **P** = **NP**. Η κρίσιμη αυτή διαφορά υποδεικνύει ότι δεν μπορούμε να σχεδιάσουμε ένα προσεγγιστικό σχήμα για το πρόβλημα $P \parallel C_{max}$ χρησιμοποιώντας ως μαύρο κουτί έναν προσεγγιστικό αλγόριθμο για το πρόβλημα Bin Packing.

Η μέθοδος που παρουσιάζουμε στη συνέχεια δεν χρησιμοποιεί ένα προσεγγιστικό αλγόριθμο με τη συμβατική έννοια για το Bin Packing, αλλά μια ενδιαφέρουσα παραλλαγή του. Αποτελεί το πρώτο χρονικά PTAS για το πρόβλημα $P \parallel C_{max}$ και οφείλεται στους Hochbaum και Shmoys [39].

Η κύρια ιδέα της μεθόδου, είναι η δυική σχέση ανάμεσα στα προβλήματα $P \parallel C_{max}$ και Bin

Packing. Συγκεκριμένα, για δεδομένο σύνολο αντικειμένων I, τα αντικείμενα αυτά χωρίζονται σε m ομάδες έτσι ώστε το άθροισμα των μεγεθών των αντικειμένων της κάθε ομάδας να είναι το πολύ c αν και μόνο αν υπάρχει τρόπος τα ίδια αντικείμενα να μοιραστούν σε κάδους χωρητικότητας c χωρίς να χρειαζονται πάνω από m το πλήθος κάδοι. Αντίστοιχα, για δεδομένο στιγμιότυπο I, αν η βέλτιστη λύση του προβλήματος $P \parallel C_{max}$ για m μηχανές (κάδους) είναι $\mathrm{OPT}(I, m)$ και η βέλτιστη λύση του Bin Packing με όριο χωρητικότητας t είναι $\mathrm{OPT}(I, t)$ τότε ισχύει ότι:

$$\mathrm{OPT}(I, m) \;=\; \min\{\, t \mid \mathrm{OPT}(I, t) \leq m \,\}. \tag{2.6}$$



Σχήμα 2.2: *Δυική σχέση ανάμεσα σε Bin Packing και $P \parallel C_{max}$: για δεδομένο στιγμιότυπο I η τιμή της βέλτιστης λύσης για το Bin Packing (αντίστοιχα για το $P \parallel C_{max}$) βρίσκεται σε κάποιο σημείο ενός τμήματος της τεθλασμένης Optimal Solutions το οποίο είναι παράλληλο (αντίστοιχα κάθετο) στον άξονα Num of bins. Οι υπερβολές h_1: Num of Bins = $\sum_j p_j$/Bin Capacity και h_2: Num of Bins = $2\sum_j p_j$/Bin Capacity φράσσουν κάτω και άνω αντίστοιχα τις τιμές των βέλτιστων λύσεων των δύο προβλημάτων.*

Από το Σχήμα 2.2 είναι εμφανές ότι έχουμε δύο αρκετά καλά φράγματα για τις τιμές των βέλτιστων λύσεων των δύο προβλημάτων, συγκεκριμένα τις υπερβολές h_1 και h_2 από κάτω και άνω αντίστοιχα.

Α. Δυαδική αναζήτηση. Θα χρησιμοποιήσουμε τα φράγματα αυτά σαν αρχικά σημεία για να εφαρμόσουμε δυαδική αναζήτηση στο διάστημα $[LB, UB]$ όπου $LB = \max\{\frac{1}{2}\sum_j p_j,\ p_{max}\}$ και $UB = 2LB$. Το τελευταίο συστατικό που χρειαζόμαστε για την δυαδική αναζήτηση είναι μια προσέγγιση του $\mathrm{OPT}(I, t)$. Μία τέτοια προσέγγιση υπολογίζει ο άλγόριθμος $dual_\epsilon(I, t)$, για το

τρέχον στιγμιότυπο I και για χωρητικότητα t που αντιστοιχεί στο makespan αυτού.

Η δυαδική αναζήτηση συνοψίζεται στο ακόλουθο σχήμα.

Αλγόριθμος 2 Ο αλγόριθμος $primal_\epsilon(I, m)$ για το πρόβλημα $P \parallel C_{max}$

1. **Όσο** $UB - LB \leq \epsilon LB$

   1.1 $\delta = \frac{UB+LB}{2}$.

   1.2 **Αν** $dual_\epsilon(I, \delta) > m$ **τότε** $LB = \delta$.

   1.3 **Αλλιώς** $UB = \delta$.

   1.4 $\delta^\star = \delta$.

2. Για $\delta^\star = \lceil LB \rceil$ εκτέλεσε $dual_\epsilon(I, \delta^\star)$

   2.1 **Αν** $dual_\epsilon(I, \delta^\star) \leq m$ **τότε** τέλος.

   2.2 **Αλλιώς** για $\delta^\star = UB$ εκτέλεσε $dual_\epsilon(I, \delta^\star)$.

Στο Βήμα 1, εάν το Bin Packing για χωρητικότητα $(UB+LB)/2$ έχει λύση με λιγότερους κάδους από ότι ζητάμε, τότε συνεχίζουμε να ψάχνουμε για καλύτερη λύση στο διάστημα $[LB, (UB+LB)/2]$. Αλλιώς έχουμε περάσει τη ζητούμενη τιμή και ψάχνουμε στο διάστημα $[(UB + LB)/2, UB]$. Όταν ολοκληρωθεί η δυαδική αναζήτηση, τα άκρα UB, LB θα απέχουν το πολύ ϵLB, άρα και το $\delta^\star$ θα απέχει το πολύ ϵLB από το βέλτιστο. Όμως $LB \leq \mathrm{OPT}(I, m)$, άρα το $\delta^\star$ απέχει από το βέλτιστο το πολύ $\epsilon\,\mathrm{OPT}(I, m)$ και έτσι έχουμε ότι $\delta^\star \leq (1 + \epsilon)\,\mathrm{OPT}(I, m)$. Επομένως, ο αλγόριθμος $primal_\epsilon(I, m)$ είναι ένας $(1 + \epsilon)$-προσεγγιστικός αλγόριθμος για το πρόβλημα $P \parallel C_{max}$.

Β. Δυική Προσέγγιση. Στην δυαδική αναζήτηση χρησιμοποιούμε τον αλγόριθμο $dual_\epsilon(I, t)$ για να υπολογίσουμε προσεγγιστικές λύσεις του προβλήματος Bin Packing. Η καινοτομία του αλγορίθμου αυτού έγκειται στο ότι δεν δίνει πάντοτε εφικτές λύσεις για το πρόβλημα Bin Packing, ενώ χρησιμοποιεί τον βέλτιστο ή και μικρότερο αριθμό κάδων. Συγκεκριμένα για είσοδο (I, t), όπου t είναι η χωρητικότητα κάθε κάδου, ο αλγόριθμος βρίσκει μία ανάθεση των αντικειμένων σε το πολύ $\mathrm{OPT}(I, t)$ κάδους, καθένας από τους οποίους έχει συνολικό φορτίο το πολύ $(1 + \epsilon)t$, ενδεχομένως μεγαλύτερο από την χωρητικότητά του που είναι t.

Πέραν αυτού όμως είναι ένα πολύ καλό παράδειγμα αλγορίθμου που χρησιμοποιεί την τεχνική μετασχηματισμού των δεδομένων της εισόδου.

Η εκτέλεση του αλγορίθμου $dual_\epsilon(I, t)$ συνοψίζεται στο παρακάτω σχήμα (βλ. Αλγόριθμος 3). Με L_i συμβολίζουμε το συνολικό μέγεθος των αντικειμένων-διεργασιών ή αλλιώς το φορτίο του

εκάστοτε κάδου i που επιλέγεται κατά την εφαρμογή του κανόνα της Αυθαίρετης Λίστας (βλ. Ενότητα 1.2) στο Βήμα 5 του αλγορίθμου.

Η στρογγυλοποίηση στο Βήμα 2 γίνεται διαμερίζοντας το διάστημα $(\epsilon t, t]$, όπου ανήκουν τα μεγέθη των διεργασιών που απέμειναν μετά το βήμα 1, σε $k = \lceil \log(1/\epsilon)/\epsilon \rceil$ υποδιαστήματα της μορφής $a_i = [\epsilon(1+\epsilon)^i t, \epsilon(1+\epsilon)^{i+1} t)$, $i = 0, 1, \ldots, k$ και αντικαθιστώντας κάθε διεργασία p_j στο διάστημα a_i από την $p'_j = \epsilon(1+\epsilon)^i t$.

Αλγόριθμος 3 Ο αλγόριθμος $dual_\epsilon(I, t)$ για το πρόβλημα Bin Packing.

---

1. Αφαίρεσε όλες τις *μικρές διεργασίες*, μεγέθους $p_j \leq \epsilon t$.

2. Με στρογγυλοποίηση κατασκεύασε ένα σταθερό αριθμό μεγεθών διεργασιών.

3. Βρές μία βέλτιστη ανάθεση αυτών στους κάδους.

4. Χρησιμοποίησε την ανάθεση αυτή για τα αρχικά μεγέθη των διεργασιών και αύξησε τη χωρητικότητα των κάδων σε $(1+\epsilon)t$.

5. Βάσει του κανόνα Αυθαίρετης Λίστας (βλ. Ενότητα 1.2) δρομολόγησε κάθε διεργασία j μεγέθους $p_j \leq \epsilon t$ στους κάδους

   5.1 **Αν** $p_j + L_i \leq (1+\epsilon)t$ **τότε** εκτέλεσε την j στον κάδο i.

   5.2 **Αλλιώς** άνοιξε νέο κάδο και εκτέλεσε την j.

6. **Επέστρεψε** τον αριθμό των κάδων που χρησιμοποιήθηκαν.

Μετά την στρογγυλοποίηση μπορούμε να αποθηκεύσουμε τις διεργασίες σε ένα διάνυσμα $\vec{n} = (n_1, n_2, \ldots, n_k)$ όπου n_i το πλήθος των διεργασιών μεγέθους $\epsilon(1+\epsilon)^i t$.

Στο Βήμα 3 επιλύουμε το απλοποιημένο αυτό στιγμιότυπο με δυναμικό προγραμματισμό. Ορίζουμε ως $Bins(n_1, n_2, \ldots, n_k)$ τον ελάχιστο αριθμό από κάδους που χρειάζονται για να αναθέσουμε τις διεργασίες που υποδεικνύει το διάνυσμα $\vec{n} = (n_1, n_2, \ldots, n_k)$.

Αρχικά υπολογίζουμε το σύνολο $Q = \{\vec{q} = (q_1, q_2, \ldots, q_k) \mid Bins(q_1, q_2, \ldots, q_k) = 1,\ 0 \leq q_i \leq n_i,\ 1 \leq i \leq k\}$. Αυτό χρειάζεται το πολύ n^k δοκιμές. Κατόπιν, χρησιμοποιούμε την ακόλουθη αναδρομή μέχρι να υπολογίσουμε την ποσότητα $Bins(n_1, n_2, \ldots, n_k)$.

$$Bins(i_1, i_2, \ldots, i_k) = 1 + \min_{\vec{q} \in Q} \{ Bins(i_1 - q_1, i_2 - q_2, \ldots, i_k - q_k) \}, \quad i_j \leq n_j.$$

Εάν εφαρμόσουμε την βέλτιστη ανάθεση που βρήκαμε στο Βήμα 3 στα αρχικά μεγέθη των διεργα-

σιών, τότε οι κάδοι θα έχουν ενδεχομένως φορτίο μεγαλύτερο από t, αλλά σίγουρα όχι μεγαλύτερο από $(1+\epsilon)t$, λόγω της στρογγυλοποίησης. Έτσι στο Βήμα 4 αυξάνουμε τη χωρητικότητα κάθε κάδου από t σε $(1+\epsilon)t$. Επίσης, είμαστε βέβαιοι ότι έχουμε χρησιμοποιήσει το πολύ τον βέλτιστο αριθμό κάδων και επομένως, μένει να αναθέσουμε τις μικρές διεργασίες που αφαιρέθηκαν στο Βήμα 1, διατηρώντας παράλληλα τις συνθήκες αυτές. Αυτό γίνεται στο Βήμα 5 του αλγορίθμου όπου για κάθε μικρή διεργασία j που δρομολογείτε παρουσιάζονται οι εξής δύο περιπτώσεις:

- Η j αυξάνει το φορτίο L_i του κάδου i στον οποίο δρομολογείται σε μέγεθος μικρότερο ή ίσο της χωρητικότητας $(1+\epsilon)t$ αυτού. Τότε η j εκτελείται στον κάδο i και ο κάθε κάδος συνεχίζει να έχει φορτίο μικρότερο ή ίσο της χωρητικότητας του $(1+\epsilon)t$. Επιπλέον, ο αριθμός των κάδων δεν αλλάζει σε σχέση με το Βήμα 3 οπότε και συνεχίζει να είναι το πολύ βέλτιστος.
- Η j αυξάνει το φορτίο L_i του κάδου i στον οποίο δρομολογείται σε μέγεθος μεγαλύτερο της χωρητικότητας $(1+\epsilon)t$ αυτού. Τότε ανοίγεται ένας νέος κάδος και η j εκτελείται σ΄ αυτόν. Έπομένως το φορτίο του κάθε κάδου συνεχίζει να είναι μικρότερο ή ίσο από την χωρητικότητα $(1+\epsilon)t$ αυτού και επιπλέον θα υπάρχουν κάδοι με φορτίο μεγαλύτερο του t, όπως για παράδειγμα ο κάδος i, αφού η διεργασία j έχει μέγεθος το πολύ ϵt. Συνεπώς, κάθε ανάθεση, ακόμα και η βέλτιστη, θα αναγκαζόταν να χρησιμοποιήσει τουλάχιστον έναν ακόμη κάδο.

Στο Βήμα 6 ο αλγόριθμος επιστρέφει έναν ακέραιο ο οποίος αντιστοιχεί στον αριθμό των κάδων που χρησιμοποιήθηκαν μετά και την ολοκλήρωση του Βήματος 5.

Γ. Απόδοση και Πολυπλοκότητα. Όπως είδαμε παραπάνω, το $\delta^\star$ της δυαδικής αναζήτησης είναι το πολύ $(1+\epsilon)\,\mathrm{OPT}(I,m)$. Η μέγιστη υπερχείλιση του $dual_\epsilon(I,\delta^\star)$ είναι $(1+\epsilon)\delta^\star$, οπότε τελικά θα έχουμε μια ανάθεση των διεργασιών με makespan το πολύ

$$(1+\epsilon)\delta^\star = (1+\epsilon)(1+\epsilon)\,\mathrm{OPT}(I,m) \;\leq\; (1+3\epsilon)\,\mathrm{OPT}(I,m).$$

Στη δυαδική αναζήτηση, για να φτάσουμε από ένα διάστημα μήκους $(UB-LB)=LB$ σε ένα διάστημα μήκους ϵLB χρειαζόμαστε $\log(1/\epsilon)$ βήματα. Επιπλέον, το κύριο κόστος του $dual_\epsilon(I,\delta^\star)$ καθορίζεται από τη χρονική πολυπλοκότητα του δυναμικού προγραμματισμού, η οποία είναι $O((n/\epsilon)^k)$. Όμως $k=\lceil\log(1/\epsilon)/\epsilon\rceil$, οπότε η εξάρτηση από το ϵ είναι εκθετική. Επομένως, έχουμε το επιθυμητό PTAS για το πρόβλημα $P\,\|\,C_{max}$.

Είναι εφικτό, κάνοντας κάποιες τροποποιήσεις, να μειώσουμε την εξάρτηση από το n στο παραπάνω PTAS σε γραμμική. Αρχικά παρατηρούμε ότι, σε κάθε επανάληψη της δυαδικής αναζήτησης στον αλγόριθμο $primal_\epsilon(I,m)$, ο καθορισμός των μεγάλων διεργασιών μεγέθους $p_j > \epsilon LB$

και η στρογγυλοποίηση τους στον αλγόριθμο $dual_\epsilon(I,t)$ απαιτεί πολυπλοκότητα χρόνου το πολύ $O(n)$. Επομένως, για να πετύχουμε πολυπλοκότητα χρόνου γραμμική ως προς το n, αλλά εξαρτώμενη από μία αρκετά μεγάλη σταθερά, εκθετική ως προς το $1/\epsilon$, διαμερίζουμε το αρχικό διάστημα $[\epsilon LB, 2LB]$ σε $O(\log 1/\epsilon)/\epsilon)$ υποδιαστήματα όπως και στο Βήμα 2 του αλγορίθμου $dual_\epsilon(I,t)$ και "μαντεύουμε" τιμές για το δ, κατά τη δυαδική αναζήτηση, στο διακριτό πλήθος των υποδιαστημάτων που κατασκευάσαμε. Για κάθε τέτοια τιμή έχουμε έτοιμη την επιθυμητή διαμέριση των μεγεθών των μεγάλων διεργασιών.

Επιπλέον, αντί της εφαρμογής δυναμικού προγραμματισμού για την εύρεση μίας βέλτιστης ανάθεσης των στρογγυλοποιημένων διεργασιών στους κάδους, διαμορφώνουμε το εκάστοτε στιγμιότυπο του προβλήματος Bin Packing (βλ. Βήμα 3 του Αλγορίθμου 3) ως ένα ακέραιο γραμμικό πρόγραμμα με σταθερό πλήθος μεταβλητών, που εξαρτάται εκθετικά από το $1/\epsilon$. Χρησιμοποιώντας κατάλληλα τον αλγόριθμο του Lenstra [64] μπορούμε να λύσουμε βέλτιστα και σε σταθερό χρόνο $C(\epsilon)$ (εκθετικό στο $1/\epsilon$) το αντίστοιχο στιγμιότυπο. Οπότε η συνολική χρονική πολυπλοκότητα του PTAS θα είναι ίση με $n + C(\epsilon)$, όπου $C(\epsilon)$ μία αρκετά μεγάλη σταθερά, εξαρτώμενη από το ϵ. Περισσότερες λεπτομέρειες για το αποτέλεσμα αυτό, υπάρχουν στο [38]. Η ιδέα της διαμόρφωσης ενός τέτοιου ακέραιου γραμμικού προγράμματος και η βέλτιστη επίλυση αυτού χρησιμοποιείται και στο PTAS που περιγράφεται στην ενότητα που ακολουθεί και αφορά επίσης στο πρόβλημα $P \parallel C_{max}$.

2.4.2 Ένα δεύτερο PTAS

Στο τέλος της προηγούμενης ενότητας παρατηρήσαμε ότι μπορούμε να βελτιώσουμε την χρονική πολυπλοκότητα του προσεγγιστικού σχήματος, επιλύοντας το εκάστοτε μετασχηματισμένο στιγμιότυπο του Bin Packing μέσω του αλγορίθμου Lenstra [64]. Στη συνέχεια περιγράφουμε πώς μπορούμε να σχεδιάσουμε ένα δεύτερο PTAS για το πρόβλημα $P \parallel C_{max}$ εφαρμόζοντας τον αλγόριθμο του Lenstra [64] απευθείας σε ένα στιγμιότυπο του $P \parallel C_{max}$, αφού πρώτα το μετατρέψουμε με κατάλληλο τρόπο σε ένα ακέραιο γραμμικό πρόγραμμα με σταθερό πλήθος μεταβλητών. Το αποτέλεσμα αυτό σχεδιάστηκε από τους Schuurman και Woeginger [86] και σε αντίθεση με το PTAS των Hochbaum και Shmoys [39] δεν στηρίζεται στη δυική σχέση του προβλήματος με το Bin Packing.

Όπως προηγουμένως, η βέλτιστη λύση θα βρίσκεται στο διάστημα $[LB, 2LB]$ ή ισοδύναμα:

$$LB \leq \mathrm{OPT}(I,m) \leq 2LB. \tag{2.7}$$

Η κατασκευή του PTAS συνοψίζεται στα τρία στάδια που περιγράφονται στη συνέχεια.

Στάδιο Α. Οι διεργασίες διακρίνονται σε μικρές (χρόνος εκτέλεσης $p_j \leq \epsilon LB$) και μεγάλες. Όπως ακριβώς και στην Ενότητα 2.3, οι μικρές διεργασίες με συνολικό χρόνο επεξεργασίας S αντικαθίστανται από ακριβώς $\lfloor S/(\epsilon LB) \rfloor$ διεργασίες, με χρόνο επεξεργασίας ϵLB η καθεμία. Για κάθε μεγάλη διεργασία j του στιγμιοτύπου I, το στιγμιότυπο I' θα περιέχει τη διεργασία j' με χρόνο εκτέλεσης $p'_j = \epsilon^2 LB \lfloor p_j/(\epsilon^2 LB) \rfloor$, δηλαδή το p'_j προκύπτει με στρογγυλοποίηση του p_j στο αμέσως μικρότερο ακέραιο πολλαπλάσιο του $\epsilon^2 LB$. Παρατηρούμε ότι:

$$p_j \;\leq\; p'_j + \epsilon^2 LB \;\leq\; (1+\epsilon)p'_j. \tag{2.8}$$

Όπως και στην Ενότητα 2.3, μπορεί να δειχθεί ότι $\mathrm{OPT}(I', m) \leq (1+\epsilon)\,\mathrm{OPT}(I, m)$ μόνο που τώρα οι μεγάλες διεργασίες του I' μπορεί να είναι μικρότερες από τις αντίστοιχες στο I. Αυτό όμως λειτουργεί θετικά αφού η αντικατάσταση διεργασιών με διεργασίες μικρότερου μεγέθους μπορεί μόνο να μειώσει το makespan ή να το διατηρήσει ίδιο. Βάσει της (2.7) και για θετικό ακέραιο $E = \lceil 1/\epsilon \rceil$, προκύπτει η ακόλουθη σχέση:

$$OPT(I', m) \;\leq\; 2(1+\epsilon)LB \;\leq\; (2E^2 + 2E)\epsilon^2 LB. \tag{2.9}$$

Στο στιγμιότυπο I' οι χρόνοι επεξεργασίας των στρογγυλοποιημένων μεγάλων διεργασιών ανήκουν στο διάστημα $[\epsilon LB, LB]$ και έτσι μπορούν να γραφούν ως $k \cdot \epsilon^2 LB$, όπου k ακέραιος με $E \leq k \leq E^2$. Παρατηρούμε ότι $\epsilon LB = E \cdot \epsilon^2 LB$ οπότε και ο χρόνος εκτέλεσης των νέων μικρών διεργασιών μπορεί να γραφεί ομοίως ως $k \cdot \epsilon^2 LB$. Για $k = E, \ldots, E^2$, ορίζουμε με n_k το πλήθος των διεργασιών του I' με χρόνο εκτέλεσης $k \cdot \epsilon^2 LB$. Μπορούμε έτσι να αναπαραστήσουμε το στιγμιότυπο I' συλλέγοντας όλα τα δεδομένα στο διάνυσμα $\vec{n} = (n_E, \ldots, n_{E^2})$.

Στάδιο Β. Θα διατυπώσουμε το απλοποιημένο στιγμιότυπο I' ως ένα ακέραιο γραμμικό πρόγραμμα του οποίου το πλήθος των μεταβλητών φράσσεται από μία σταθερά που εξαρτάται από το E και επομένως είναι ανεξάρτητο από το μέγεθος της εισόδου. Στη συνέχεια θα εφαρμόσουμε τον αλγόριθμο του Lenstra [64] για ακέραια γραμμικά προγράμματα με προκαθορισμένη διάσταση για να πάρουμε μία βέλτιστη λύση για το I' σε πολυωνυμικό χρόνο.

Αρχικά δίνουμε κάποιους χαρακτηρισμούς ώστε να περιγράψουμε τις εφικτές χρονοδρομολογήσεις των διεργασιών στο στιγμιότυπο I'. Για οποιαδήποτε μηχανή, το διάνυσμα $\vec{u} = (u_E, \ldots, u_{E^2})$ αντιστοιχεί στις διεργασίες που έχουν δρομολογηθεί προς εκτέλεση στη μηχανή αυτή. Συγκεκριμένα u_k είναι το πλήθος των διεργασιών χρόνου επεξεργασίας $k \cdot \epsilon^2 LB$, για $k = E, \ldots, E^2$. Για κάθε διάνυσμα $\vec{u}$ θέτουμε $C(\vec{u}) = \sum_{k=E}^{E^2} u_k \cdot k$. Παρατηρούμε ότι το φορτίο κάθε μηχανής θα ισούται με $\sum_{k=E}^{E^2} u_k \cdot k \cdot \epsilon^2 LB = C(\vec{u}) \cdot \epsilon^2 LB$. Ορίζουμε ως $\mathcal{U}$ το σύνολο των διανυσμάτων $\vec{u}$ για τα οποία ισχύει ότι $C(\vec{u}) \leq 2E^2 + 2E$, δηλαδή για τα οποία η αντίστοιχη μηχανή έχει φορτίο το πολύ $(2E^2 + 2E) \cdot \epsilon^2 LB$.

Λόγω της ανισότητας (2.10), για να λύσουμε βέλτιστα το I', μας ενδιαφέρουν μόνο διανύσματα του συνόλου $\mathcal{U}$. Επιπλέον, αφού κάθε διεργασία έχει χρόνο εκτέλεσης τουλάχιστον $E \cdot \epsilon^2 LB$, κάθε διάνυσμα θα περιέχει το πολύ $2E+2$ διεργασίες και επομένως ο πληθικός αριθμός του συνόλου $\mathcal{U}$ θα είναι $|\mathcal{U}| \leq (E^2 - E + 1)^{2E+3}$, οπότε και φράσσεται από μία σταθερά που εξαρτάται από το E. Η ιδιότητα αυτή είναι σημαντική αφού το ακέραιο γραμμικό πρόγραμμα που θα κατασκευάσουμε αποτελείται από $2|\mathcal{U}|+1$ μεταβλητές.

Θεωρούμε τώρα μία συγκεκριμένη χρονοδρομολόγηση για το στιγμιότυπο I'. Για κάθε διάνυσμα $\vec{u} \in \mathcal{U}$, ορίζουμε με $x_{\vec{u}}$ τον αριθμό των μηχανών που αντιστοιχούν στο $\vec{u}$. Χρησιμοποιούμε την 0-1 μεταβλητή $y_{\vec{u}}$ ως δείκτη της μεταβλητής $x_{\vec{u}}$. Αν $y_{\vec{u}} = 0$, τότε $x_{\vec{u}} = 0$, αλλιώς $x_{\vec{u}} \geq 1$. Τέλος με τη μεταβλητή z δηλώνουμε το makespan της χρονοδρομολόγησης. Το ακέραιο γραμμικό πρόγραμμα (ILP) θα έχει ως ακολούθως:

ILP. minimize z

subject to $\sum_{\vec{u} \in \mathcal{U}} x_{\vec{u}} = m$

$\sum_{\vec{u} \in \mathcal{U}} x_{\vec{u}} \cdot \vec{u} = \vec{n}$

$y_{\vec{u}} \leq x_{\vec{u}} \leq m \cdot y_{\vec{u}} \qquad \forall\, \vec{u} \in \mathcal{U}$

$C(\vec{u}) \cdot y_{\vec{u}} \leq z \qquad \forall\, \vec{u} \in \mathcal{U}$

$x_{\vec{u}} \geq 0,\ x_{\vec{u}} \in \mathbb{Z} \qquad \forall\, \vec{u} \in \mathcal{U}$

$y_{\vec{u}} \in \{0,1\} \qquad \forall\, \vec{u} \in \mathcal{U}$

$z \geq 0,\ z \in \mathbb{Z}$

Ο στόχος είναι η ελαχιστοποίηση της τιμής του z, οπότε το makespan της επικείμενης χρονοδρομολόγησης θα ισούται με $z \cdot \epsilon^2 LB$, δηλαδή θα είναι πολλαπλάσιο του z. Ο πρώτος περιορισμός δηλώνει ότι ακριβώς m μηχανές πρέπει να χρησιμοποιηθούν. Ο δεύτερος εξασφαλίζει πως όλες οι διεργασίες θα δρομολογηθούν και ο τρίτος συνδέει τη μεταβλητή $x_{\vec{u}}$ με την μεταβλητή-δείκτη αυτής $y_{\vec{u}}$. Συγκεκριμένα κάθε μεταβλητή $x_{\vec{u}}$ θα παίρνει τιμές από 0 έως m. Ο τέταρτος περιορισμός εξασφαλίζει ότι το $z \cdot \epsilon^2 LB$ είναι τουλάχιστον όσο και το makespan της τρέχουσας χρονοδρομολόγησης. Οι υπόλοιποι περιορισμοί είναι περιορισμοί μη-αρνητικότητας και ακέραιοι περιορισμοί. Είναι ξεκάθαρο πως το βέλτιστο makespan $\mathrm{OPT}(I', m)$ θα ισούται με $z^\star \cdot \epsilon^2 LB$, όπου $z^\star$ είναι η βέλτιστη τιμή της αντικειμενικής συνάρτησης του ILP.

Το πλήθος των μεταβλητών του ILP είναι $2|\mathcal{U}|+1$ και δεν εξαρτάται από τις μεταβλητές εισόδου $\{m, n\}$. Η πολυπλοκότητα χρόνου του αλγορίθμου του Lenstra για το ILP είναι εκθετική ως προς το πλήθος των μεταβλητών, αλλά πολυωνυμική ως προς τους λογαρίθμους των συντελεστών των περιορισμών. Οι συντελεστές των περιορισμών είναι το πολύ $\max\{m, n, 2E^2 + 2E\}$)

και η συμμετοχή τους στην χρονική πολυπλοκότητα του αλγορίθμου οφείλεται στο γεγονός ότι ο αλγόριθμος του Lenstra καλεί τον *ελλειψοειδή αλγόριθμο* [52] του οποίου ο χρόνος εκτέλεσης εξαρτάται από τους συντελεστές των περιορισμών. Συνολικά η πολυπλοκότητα χρόνου είναι της τάξης του $O(\log^{O(1)}(m+n))$, όπου η κρυμμένη σταθερά $O(1)$ είναι εκθετικά εξαρτώμενη από το $1/\epsilon$. Συνοψίζοντας μπορούμε να επιλύσουμε το απλοποιημένο στιγμιότυπο I' σε πολυωνυμικό χρόνο.

Στάδιο Γ. Θεωρούμε μία βέλτιστη χρονοδρομολόγηση A' για το στιγμιότυπο I', στην οποία ο συνολικός χρόνος εκτέλεσης των νέων μικρών διεργασιών, σε κάποια μηχανή M_i , είναι S'_i και ο συνολικός χρόνος εκτέλεσης των στρογγυλοποιημένων μεγάλων διεργασιών είναι B'_i.

Κατασκευάζουμε μία χρονοδρομολόγηση A για το αρχικό στιγμιότυπο I ως εξής: σε κάθε μηχανή M_i της A', αντικαθιστούμε κάθε στρογγυλοποιημένη μεγάλη διεργασία με την αντίστοιχη διεργασία της στο στιγμιότυπο I. Λόγω της ανισότητας (2.8), για κάθε μηχανή, θα ισχύει ότι το $B_i \leq (1+\epsilon)B'_i$, όπου B_i το αντίστοιχο του B'_i για το στιγμιότυπο I. Για τις μικρές διεργασίες κάνουμε ό,τι και στην Ενότητα 2.3, δηλαδή δεσμεύουμε χώρο $S'_i + 2\epsilon LB$ σε κάποια μηχανή M_i και εκτελούμε όσες περισσότερες μικρές διεργασίες μπορούμε. Όλες οι ανεκτέλεστες μικρές διεργασίες θα χωράνε να εκτελεστούν στις υπόλοιπες μηχανές.

Η μηχανή M_i, στη χρονοδρομολόγηση A που κατασκευάσαμε, θα έχει φορτίο L_i για το οποίο ισχύει ότι:

$$\begin{aligned} L_i &\leq (1+\epsilon)B'_i + S'_i + 2\epsilon LB \leq (1+\epsilon)\,\mathrm{OPT}(I', m) + 2\epsilon\,\mathrm{OPT}(I, m) \\ &\leq (1+\epsilon)^2 OPT(I, m) + 2\epsilon\,\mathrm{OPT}(I, m) = (1 + 4\epsilon + \epsilon^2)\,\mathrm{OPT}(I, m). \end{aligned}$$

Οι ενδιάμεσες ανισότητες προκύπτουν από τις (2.1) και (2.3). Το $4\epsilon + \epsilon^2$ μπορεί να πλησιάσει αυθαίρετα στο 0 και επιπλέον, λόγω του Σταδίου Β η πολυπλοκότητα χρόνου είναι πολυωνυμική ως προς τα $\{m, n\}$ και εκθετική ως προς το $1/\epsilon$. Επομένως, ο αλγόριθμος που περιγράψαμε αποτελεί ένα PTAS για το πρόβλημα $P \parallel C_{max}$.

2.5 Συμπεράσματα

Στην Ενότητα 2.2 περιγράψαμε μία γενική τεχνική κατασκευής προσεγγιστικών σχημάτων στην οποία βασίζονται πολλά από τα μέχρι τώρα γνωστά αποτελέσματα. Σαφώς δεν πρόκειται για την μοναδική τεχνική που έχει παρουσιασθεί. Για παράδειγμα, για το πρόβλημα $Pm \parallel C_{max}$ υπάρχουν τουλάχιστον τρία PTAS τα οποία βασίζονται σε διαφορετικές τεχνικές (βλ. Schuurman και Woeginger [86]).

Στην Ενότητα 2.3 παρουσιάσαμε ένα PTAS για το πρόβλημα $P2 \parallel C_{max}$ [86] και εξηγήσαμε

τους λόγους για τους οποίους δεν μπορεί να εφαρμοσθεί στην περίπτωση του προβλήματος $P \parallel C_{max}$.

Στις Ενότητα 2.4 παρουσιάσαμε δύο PTAS για το πρόβλημα $P \parallel C_{max}$. Στο πρώτο PTAS [39], το πρόβλημα προσεγγίζεται μέσω ενός κατά τρόπον δυικού του προβλήματος, όπως είναι το πρόβλημα Bin Packing, ενώ στο δεύτερο η προσέγγιση γίνεται με απευθείας εφαρμογή της τεχνικής μετασχηματισμού των δεδομένων της εισόδου σε ένα στιγμιότυπο του $P \parallel C_{max}$ [86].

Η μέθοδος της προσέγγισης μέσω ενός δυικού προβλήματος δόθηκε από τους Hochbaum και Shmoys [39]. Ονομάζεται Dual Approximation Scheme μέθοδος και το αποτέλεσμα της Ενότητας 2.4.1 αποτελεί την πρώτη εφαρμογή της. Πρόκειται για μία μέθοδο η οποία είναι σε θέση να δώσει λύση σε προβλήματα βελτιστοποίησης για τα οποία άλλες μέθοδοι, όπως για παράδειγμα η τεχνική μετασχηματισμού των δεδομένων της εισόδου του προβλήματος, φαντάζει δύσκολο να αποδώσουν το επιθυμητό αποτέλεσμα. Κάτι τέτοιο είναι εφικτό αν προηγουμένως έχει προσδιορισθεί ο κατάλληλος συσχετισμός του προβλήματος με κάποιο κατά τρόπον δυικό του πρόβλημα, το οποίο να μπορούμε να προσεγγίσουμε (βλ. [40] για μία επιπλέον εφαρμογή).

Παρότι στο πρώτο PTAS η προσέγγιση του προβλήματος $P \parallel C_{max}$ έγινε μέσω του προβλήματος Bin Packing και στα δύο PTAS εφαρμόζεται η τεχνική μετασχηματισμού των δεδομένων της εισόδου και διαφέρει ουσιαστικά σε καθένα από αυτά μόνο ως προς τον τρόπο επίλυσης του απλοποιημένου στιγμιοτύπου. Συγκεκριμένα, στο PTAS της Ενότητας 2.4.1 η τεχνική αυτή εφαρμόζεται στο πρόβλημα Bin Packing και το απλοποιημένο στιγμιότυπο επιλύεται με χρήση δυναμικού προγραμματισμού. Αντίστοιχα, στο δεύτερο PTAS (Ενότητα 2.4.2) η τεχνική αυτή εφαρμόζεται απευθείας στο πρόβλημα $P \parallel C_{max}$ και το απλοποιημένο στιγμιότυπο, από το οποίο διαμορφώνεται ένα ακέραιο γραμμικό πρόγραμμα με σταθερό πλήθος μεταβλητών, επιλύεται με χρήση του αλγορίθμου του Lenstra [64] για ακέραια γραμμικά προγράμματα σταθερής διάστασης.

Τα παραπάνω αποτελέσματα δεν φαίνεται να μπορούν να βελτιωθούν περισσότερο και αυτό οφείλεται στο ότι το πρόβλημα $P \parallel C_{max}$ είναι strongly-**NP**-hard. Έτσι, οποιαδήποτε σημαντική βελτίωση θα οδηγούσε στη εύρεση ενός FPTAS για αυτό, το οποίο συνεπάγεται ότι **P** = **NP** (βλ. Ενότητα Α.2.1). Όντας αισιόδοξοι, μπορούμε επομένως να ισχυριστούμε ότι το πρόβλημα $P \parallel C_{max}$ έχει μελετηθεί πλήρως όσον αφορά την προσεγγισιμότητα του.

Παρόλα αυτά υπάρχουν αρκετά προβλήματα χρονοδρομολόγησης, με την ίδια αντικειμενική συνάρτηση C_{max}, για τα οποία παραμένει ανοιχτή τόσο η εύρεση ενός προσεγγιστικού σχήματος, όσο και ο βαθμός προσεγγισιμότητάς τους. Ένα τέτοιο είναι το πρόβλημα $R \parallel C_{max}$ στο οποίο οι μηχανές είναι μη σχετιζόμενες, δηλαδή κάθε διεργασία έχει διαφορετικό χρόνο επεξεργασίας σε κάθε μηχανή:

- Το πρόβλημα $R \parallel C_{max}$ έχει αποδειχθεί ότι δεν μπορεί να προσεγγιστεί με παράγοντα μικρό-

τερο από 3/2 [66], εκτός και αν **P** = **NP**, ενώ μπορεί να προσεγγιστεί με παράγοντα 2 [66]. Παραμένει σημαντικό ανοιχτό ερώτημα αν υπάρχει προσέγγιση με παράγοντα $\rho \in (3/2, 2)$. Τα ίδια ακριβώς άνω και κάτω φράγματα ισχύουν και στην περίπτωση του προβλήματος $R\,|p_{ij} = p_j$ ή $\infty|\,C_{max}$, όπου κάθε διεργασία j δεν μπορεί να εκτελεσθεί σε οποιαδήποτε μηχανή, ενώ στις μηχανές όπου μπορεί να εκτελεστεί ο χρόνος επεξεργασίας της είναι ο ίδιος και ισούται με p_j. Και για το πρόβλημα αυτό παραμένει ανοιχτό το παραπάνω ερώτημα, δηλαδή αν υπάρχει προσέγγιση με παράγοντα $\rho \in (3/2, 2)$.

Κεφάλαιο 3

Το πρόβλημα ελαχιστοποίησης του μέσου χρόνου ολοκλήρωσης

Όπως αναφέραμε στο Κεφάλαιο 2 το makespan μίας χρονοδρομολόγησης μετρά το χρόνο στον οποίο ολοκληρώνεται η εκτέλεση όλων των διεργασιών. Ωστόσο, αν θεωρήσουμε ότι οι διεργασίες συναγωνίζονται ανεξάρτητα η μία από την άλλη για τον ίδιο στόχο, τότε ένα πιο ταιριαστό μέτρο της απόδοσης τους είναι ο μέσος χρόνος ολοκλήρωσης της καθεμιάς. Η παρατήρηση αυτή οδήγησε στη μελέτη προβλημάτων χρονοδρομολόγησης με αντικειμενική συνάρτηση το μέσο (βεβαρημένο) χρόνο ολοκλήρωσης. Στο κεφάλαιο αυτό παρουσιάζουμε προσεγγιστικά σχήματα για το πρόβλημα ελαχιστοποίησης του μέσου (βεβαρημένου) χρόνου ολοκλήρωσης (Minimizing average (weighted) completion time).

Τυπικά το πρόβλημα ορίζεται ως εξής: δίνεται ένα σύνολο από n διεργασίες όπου κάθε διεργασία j έχει ένα χρόνο επεξεργασίας p_j και ένα θετικό βάρος w_j. Ο στόχος είναι να χρονοδρομολογήσουμε τις διεργασίες σε ένα σύνολο από m μηχανές έτσι ώστε να ελαχιστοποιήσουμε την αντικειμενική συνάρτηση $\sum_j w_j C_j$. Θυμίζουμε ότι με C_j συμβολίζουμε το χρόνο ολοκλήρωσης της διεργασίας j, ενώ η αντικειμενική συνάρτηση $\sum_j w_j C_j$ καλείται μέσος βεβαρημένος χρόνος ολοκλήρωσης· αν το $w_j = 1$ για κάθε διεργασία j, τότε καλείται συνολικός χρόνος ολοκλήρωσης $\sum_j C_j$.

Αρχικά μελετάμε την απλή περίπτωση όπου το πλήθος των μηχανών m είναι σταθερό και ίσο με $m = 2$ και δίνουμε ένα FPTAS για το πρόβλημα $P2 \parallel \sum_j w_j C_j$ [86]. Στη συνέχεια διερευνάμε την περίπτωση όπου τα δεδομένα του προβλήματος εμπεριέχουν έναν προκαθορισμένο χρόνο αποδέσμευσης (release date) r_j, για κάθε διεργασία j, πριν το πέρας του οποίου η j δεν επιτρέπεται να αρχίσει την εκτελεσή της. Συγκεκριμένα, μελετάμε τις εκδοχές όπου οι διεργασίες είτε δεν διακόπτονται κατά την εκτέλεσή τους (*non-preemptive schedule*), είτε μπορεί να διακοπούν και

να συνεχίσουν την εκτέλεσή της σε κάποια επόμενη χρονική στιγμή (*preemptive schedule*). Για την πρώτη εκδοχή θεωρούμε ως περιβάλλον μηχανών τη Μοναδική Μηχανή και τις Παράλληλες Μηχανές και παρουσιάζουμε PTAS για τα προβλήματα $1\,|\,r_j|\,\sum_j C_j$ [2], και $P\,|\,r_j|\,\sum_j w_j C_j$ [13, 2]. Για την δεύτερη εκδοχή παρουσιάζουμε ένα PTAS για το πρόβλημα $P\,|\,r_j,\ pmtn|\,\sum_j w_j C_j$ [13, 2] όπου το περιβάλλον μηχανής είναι οι παράλληλες πανομοιότυπες μηχανές και ο συμβολισμός pmtn υποδηλώνει ότι πρόκειται για preemptive χρονοδρομολόγηση.

3.1 Επισκόπηση γνωστών αποτελεσμάτων

Για διάφορες, απλές, εκδοχές του προβλήματος παρουσιάστηκαν βέλτιστοι αλγόριθμοι πολυωνυμικού χρόνου: το πρόβλημα ελαχιστοποίησης του συνολικού χρόνου ολοκλήρωσης σε γενικό πλήθος παράλληλων πανομοιότυπων μηχανών, $P \parallel \sum_j C_j$, μπορεί να λυθεί βέλτιστα σε πολυωνυμικό χρόνο [19] με εφαρμογή του κανόνα Μικρότερου Χρόνου Επεξεργασίας (SPT rule): δρομολόγησε τις διεργασίες σε αύξουσα σειρά ως προς το χρόνο επεξεργασίας τους (βλ. Ενότητα 1.2). Αν επιπλέον, κάθε διεργασία j διαθέτει ένα βάρος w_j και περιοριστούμε σε περιβάλλον Μοναδικής Μηχανής, ο Smith [92] απέδειξε ότι το πρόβλημα $1 \parallel \sum_j w_j C_j$ μπορεί να λυθεί βέλτιστα σε πολυωνυμικό χρόνο εφαρμόζοντας τον ομώνυμο κανόνα: δρομολόγησε τις διεργασίες σε αύξουσα διάταξη ως προς το λόγο $\frac{p_j}{w_j}$ (βλ. Ενότητα 1.2). Όταν το πλήθος των μηχανών είναι σταθερό και ίσο με $m > 1$, $Pm \parallel \sum_j w_j C_j$, το πρόβλημα γίνεται **NP**-hard υπό τη συνήθη έννοια (βλ. [9]). Ο Sahni [81] παρουσίασε ένα FPTAS για το πρόβλημα αυτό· στην Ενότητα 3.2 παρουσιάζουμε ένα FPTAS για $m = 2$, $P2 \parallel \sum_j w_j C_j$, το οποίο οφείλεται στους Schuurman και Woeginger [86] και αποτελεί μία απλοποιημένη εκδοχή του αποτελέσματος του Sahni [81]. Όταν το πλήθος των μηχανών αποτελεί μέρος της εισόδου του προβλήματος, $P \parallel \sum_j w_j C_j$, το πρόβλημα γίνεται strongly **NP**-hard (βλ. [30]). Ένα PTAS για το πρόβλημα $P \parallel \sum_j w_j C_j$, σχεδιάστηκε σχετικά πρόσφατα από τους Skutella και Woeginger [91]: εφαρμόζουμε γεωμετρική στρογγυλοποίηση (βλ. Ενότητα 3.3.1) και ομαδοποιούμε τις διεργασίες βάσει του λόγου $\frac{p_j}{w_j}$ κάθε διεργασίας j. Στη συνέχεια, για κάθε ομάδα διεργασιών, υπολογίζουμε εφικτές προσεγγιστικές χρονοδρομολογήσεις τις οποίες συνενώνουμε σε μία εφαρμόζοντας τον κανόνα του Smith. Αξίζει να σημειωθεί ότι πριν την εμφάνιση του παραπάνω PTAS, το καλύτερο γνωστό αποτέλεσμα για το πρόβλημα $P \parallel \sum_j w_j C_j$ ήταν ένας $\frac{1}{2}(1+\sqrt{2})$-προσεγγιστικός αλγόριθμος των Kawaguchi και Kyan από το 1986 [50], ο οποίος βασίζεται στην εφαρμογή του κανόνα του Smith.

Αν αντί για πανομοιότυπες διαθέτουμε μη σχετιζόμενες παράλληλες μηχανές, τότε κάνοντας χρήση κατάλληλων τεχνικών (matching techniques), μπορούμε να λύσουμε βέλτιστα σε πολυωνυμικό χρόνο το πρόβλημα $R \parallel \sum_j C_j$ (βλ. [9, 42]).

Αν εισάγουμε στην είσοδο του προβλήματος και έναν προκαθορισμένο χρόνο αποδέσμευσης r_j για κάθε διεργασία j, τότε όλα τα παραπάνω προβλήματα, που επιλύονταν βέλτιστα σε πολυωνυμικό χρόνο, γίνονται strongly **NP**-hard και η εφαρμογή των παραπάνω κανόνων δεν είναι σε θέση να αποδώσει μία βέλτιστη λύση για αυτά. Εξαίρεση αποτελεί το πρόβλημα $1\,|\,r_j,\ pmtn|\ \sum_j C_j$, το οποίο μπορεί να λυθεί βέλτιστα σε πολυωνυμικό χρόνο (βλ. [7]) με εφαρμογή του κανόνα *SRPT (shortest remaining processing time)*: λειτουργεί όπως ο κανόνας SPT με τη διαφορά ότι μία εκτελούμενη διεργασία διακόπτει την εκτέλεση της όταν μία διεργασία μικρότερου χρόνου επεξεργασίας αποδεσμεύεται. Επιπλέον, το πρόβλημα $R\,|\,pmtn|\ \sum_j C_j$ παραμένει ανοιχτό ως προς την επιλυσιμότητά του σε πολυωνυμικό χρόνο.

Παρότι συναφές με το πρόβλημα ελαχιστοποίησης του makespan, το πρόβλημα ελαχιστοποίησης του μέσου (βεβαρημένου) χρόνου ολοκλήρωσης άρχισε να μελετάται ως προς την προσεγγισιμότητα του σχετικά πρόσφατα. Συγκεκριμένα, η εκδοχή του προβλήματος $1\,|\,r_j|\ \sum_j w_j C_j$ διερευνήθηκε για πρώτη φορά στην εργασία των Phillips, Stein και Wein [74]. Σ' αυτή εφαρμόστηκαν απλοί κανόνες διάταξης καθώς και τεχνικές, όπως η *αποδυνάμωση ενός γραμμικού προγράμματος (LP relaxation)*, που οδήγησαν στην εύρεση καλών προσεγγίσεων. Ακολούθησε ένας σημαντικός αριθμός εργασιών από αποτελέσματα προσέγγισης για το ίδιο πρόβλημα (βλ. [10, 15, 14, 32, 36, 71, 84, 89]). Αναφορικά, στο [15] δόθηκε ένας 1.58-προσεγγιστικός αλγόριθμος για το πρόβλημα $1\,|\,r_j|\ \sum_j C_j$, ενώ στο [84] παρουσιάστηκαν 2-προσεγγιστικοί αλγόριθμοι για τα προβλήματα $P\,|\,r_j|\ \sum_j w_j C_j$ και $P\,|\,r_j,\ pmtn|\ \sum_j w_j C_j$. Στην πλειονότητα των αποτελεσμάτων αυτών η ιδέα είναι κοινή: εφαρμόζουμε την τεχνική αποδυνάμωσης ενός γραμμικού προγράμματος και βρίσκουμε μία βέλτιστη λύση πολυωνυμικού χρόνου για το αποδυναμωμένο (relaxed) πρόγραμμα. Από τη λύση αυτή κατασκευάζουμε μία διάταξη των διεργασιών και βάσει αυτής χρονοδρομολογούμε τις διεργασίες στις μηχανές.

Οι παραπάνω προσεγγιστικοί αλγόριθμοι, παρότι δίνουν αρκετά καλές (σταθερές) προσεγγίσεις, φαντάζει εξαιρετικά δύσκολο να μετατραπούν σε προσεγγιστικά σχήματα. Ο κύριος λόγος είναι ότι η διαφορά που υπάρχει ανάμεσα στην τιμή της λύσης του αποδυναμωμένου προγράμματος και στην τιμή της βέλτιστης λύσης, κληρονομείται από κάθε αλγόριθμο ο οποίος σχεδιάζεται με βάση τη λύση του πρώτου. Ένας τέτοιος αλγόριθμος δεν μπορεί να οδηγήσει στο σχεδιασμό ενός PTAS. Σύμφωνα με τους Torng και Uthaisombut [93] ένας αλγόριθμος για το πρόβλημα $1\,|\,r_j|\ \sum_j C_j$ ο οποίος βασίζεται στην λύση του αποδυναμωμένου προγράμματος που κατασκευάζεται από τον αλγόριθμο SRPT, δεν μπορεί να δώσει λύση με παράγοντα προσέγγισης μικρότερο από $e/(e-1) \simeq 1.58$, το οποίο είναι και το καλύτερο γνωστό κάτω φράγμα [15]. Επιπλέον, η μέθοδος των Skutella και Woeginger [91] για το σχεδιασμό ενός PTAS για το πρόβλημα $P\,||\ \sum_j w_j C_j$, δεν φαίνεται να έχει τα ίδια αποτελέσματα στην περίπτωση που οι διεργασίες διαθέτουν κά-

ποιον χρόνο αποδέσμευσης. Ο λόγος είναι ότι η αρχική σειρά εκτέλεσης των διεργασιών, βάσει του κανόνα του Smith, είναι πολύ πιθανό να μην ισχύει πλέον.

Τα πρώτα προσεγγιστικά σχήματα για προβλήματα ελαχιστοποίησης του μέσου (βεβαρημένου) χρόνου ολοκλήρωσης με δεδομένο κάποιο χρόνο αποδέσμευσης σχεδιάστηκαν εντέλει, ανεξάρτητα, από τουλάχιστον πέντε διαφορετικές ομάδες ερευνητών (Afrati, Bampis, Kenyon και Milis - Karger και Stein - Chekuri και Khanna - Queyranne και Sviridenko - Skutella). Τα απλούστερα από αυτά επιλέχθηκαν και παρουσιάζονται στην εργασία [2]. Στην Ενότητα 3.3.2 παρουσιάζουμε ένα PTAS για το πρόβλημα $1\,|\,r_j\,|\,\sum_j C_j$ το οποίο οφείλεται στους Karger και Stein [2] και στην Ενότητα 3.3.3 παρουσιάζουμε PTAS για τα προβλήματα $P\,|\,r_j\,|\,\sum_j w_j C_j$ και $P\,|\,r_j,\ pmtn\,|\,\sum_j w_j C_j$ τα οποία οφείλονται στους Chekuri και Khanna [2, 13]. Στην ίδια εργασία [2], παρουσιάζονται επίσης και δύο PTAS για τα προβλήματα $Rm\,|\,r_j\,|\,\sum_j w_j C_j$ και $Rm\,|\,r_j,\ pmtn\,|\,\sum_j w_j C_j$. Αυτά προκύπτουν άμεσα από τα αντίστοιχα αποτελέσματα των προβλημάτων $P\,|\,r_j\,|\,\sum_j w_j C_j$ και $P\,|\,r_j,\ pmtn\,|\,\sum_j w_j C_j$.

Για μία εκτενή επισκόπηση των αποτελεσμάτων που αναφέρθηκαν παραπάνω αλλά και άλλων σχετικών αποτελεσμάτων ο αναγνώστης παραπέμπεται στα [11, 13, 2].

3.2 Ένα FPTAS για το πρόβλημα $P2\,||\,\sum_j w_j C_j$

Το πρόβλημα $P2\,||\,\sum_j w_j C_j$ αποτελεί ειδική περίπτωση του προβλήματος $Pm\,||\,\sum_j w_j C_j$, όπου το πλήθος των μηχανών ισούται με $m = 2$. Το πρόβλημα $Pm\,||\,\sum_j w_j C_j$ για $m > 1$ είναι **NP**-hard υπό τη συνήθη έννοια [9] και η προσπάθεια εύρεσης ενός πλήρους πολυωνυμικού προσεγγιστικού σχήματος είναι θεμιτή (βλ. Ενότητα A.2.1). Στη συνέχεια παρουσιάζουμε ένα FPTAS για το πρόβλημα $P2\,||\,\sum_j w_j C_j$, το οποίο οφείλεται στους Schuurman και Woeginger [86] και αποτελεί μία απλοποιημένη εκδοχή του αποτελέσματος του Sahni [81].

Η τεχνική που ακολουθείται για το σχεδιασμό του αποτελέσματος αυτού καλείται *τεχνική αποκοπής* (trimming technique ή structuring the execution of an algorithm, βλ. Section 5 στο [86]) και πρωτοπαρουσιάστηκε από τους Ibarra και Kim [44] το 1975. Ο Sahni [81] τη χρησιμοποίησε, για την εύρεση FPTAS σε διάφορα προβλήματα χρονοδρομολόγησης μεταξύ των οποίων και το πρόβλημα $Pm\,||\,\sum_j w_j C_j$.

Η βασική ιδέα είναι να χρησιμοποιήσουμε έναν ακριβή, αλλά ψευδοπολυωνυμικού χρόνου, αλγόριθμο $\mathcal{EA}$ και να προσπαθήσουμε να επιταχύνουμε την εκτέλεση του αφαιρώντας με κατάλληλο τρόπο κάποια από τα δεδομένα που παράγονται κατά την εκτέλεσή του. Στην ιδανική περίπτωση η πολυπλοκότητα χρόνου του αλγορίθμου γίνεται πολυωνυμική και η έξοδος που παράγεται συνιστά μία καλή, σταθερή, προσέγγιση της βέλτιστης λύσης του προβλήματος. Είναι

σημαντικό να αναφέρουμε ότι η ιδέα αυτή, αν και απλή, αποδίδει μόνο όταν ο αλγόριθμος $\mathcal{EA}$ και κατ' επέκταση το αντίστοιχο πρόβλημα βελτιστοποίησης, έχει κάποια συγκεκριμένη και κάπως περιοριστική μορφή. Στην εργασία του Woeginger [95], μεταξύ άλλων, δίνονται όλες οι λεπτομέρειες σχετικά με τις χαρακτηριστικές ιδιότητες που πρέπει να έχει ένα πρόβλημα βελτιστοποίησης ώστε να μπορεί να προσεγγιστεί μέσω ενός FPTAS, βάσει της παραπάνω τεχνικής. Ενδεικτικά αναφέρουμε ότι για να εφαρμοστεί η τεχνική αυτή σε ένα πρόβλημα βελτιστοποίησης, θα πρέπει να μπορούμε να κατασκευάσουμε για αυτό έναν ακριβή αλγόριθμο ψευδοπολυωνυμικού χρόνου. Το γεγονός αυτό, την καθιστά μη αποδοτική για προβλήματα που είναι strongly **NP**-hard, αφού αυτά δεν επιδέχονται ψευδοπολυωνυμικό αλγόριθμο (βλ. Ενότητα Α.1). Στα περισσότερα γνωστά αποτελέσματα που βασίζονται στην τεχνική αποκοπής (βλ. [86]), ο αλγόριθμος $\mathcal{EA}$ δεν προσπαθεί καν να βελτιστοποιήσει κάποιο στόχο, παρά μόνο ενεργοποιεί όλες τις δυνατές εφικτές λύσεις του προβλήματος, απαλοίφοντας απλώς τις εφικτές λύσεις που έχουν κόστος ίδιο με αυτό μίας προγενέστερης λύσης.

Ο σχεδιασμός του ζητούμενου FPTAS για το πρόβλημα $P2\|\sum_j w_j C_j$ συνοψίζεται στα εξείς τρία στάδια: Α. Διαμόρφωση ενός ακριβούς αλγορίθμου, Β. Επιτάχυνση της εκτέλεσης του αλγορίθμου και Γ. Ανάλυση χείριστης περίπτωσης. Πριν περάσουμε στην επιμέρους ανάλυση, είναι χρήσιμο να παρατηρήσουμε ότι το μέγεθος $|I|$ ενός οποιουδήποτε στιγμιοτύπου του προβλήματος είναι τουλάχιστον ίσο με $\log(p_{sum} w_{sum})$, όπου $p_{sum} = \sum_{j=1}^{n} p_j$ και $w_{sum} = \sum_{j=1}^{n} w_j$.

Α. Διαμόρφωση ενός ακριβούς αλγορίθμου. Αρχικά διατάσσουμε και αριθμούμε τις διεργασίες βάσει τον κανόνα του Smith. Εύκολα, κάνοντας ανταλλαγή δύο διαφορετικών διεργασιών της διάταξης, αποδεικνύεται ότι υπάρχει μία βέλτιστη χρονοδρομολόγηση των διεργασιών η οποία δεν περιλαμβάνει αδρανή χρονικά διαστήματα και στην οποία οι διεργασίες εκτελούνται στις μηχανές σε αύξουσα σειρά, ως προς την αρίθμηση αυτών βάσει του κανόνα Smith (βλ. [62]). Ένας εύκολος τρόπος για να αναζητήσουμε μία βέλτιστη χρόνοδρομολόγηση είναι να θεωρήσουμε την παραπάνω διάταξη των διεργασιών και βάσει αυτής να εκτελέσουμε την εκάστοτε διεργασία και με τους δύο δυνατούς τρόπους, δηλαδή είτε στην πρώτη μηχανή είτε στη δεύτερη, υπολογίζοντας κάθε φορά το τρέχον φορτίο κάθε μηχανής σε κάθε περίπτωση, καθώς και τον αντίστοιχο μέσο βεβαρημένο χρόνο ολοκλήρωσης. Από τη διαδικασία αυτή δημιουργείται ένα ψευδοπολυωνυμικό πλήθος εφικτών χρονοδρομολογήσεων, ανάμεσα στις οποίες πρέπει να αναζητήσουμε αυτή με τον ελάχιστο μέσο βεβαρημένο χρόνο ολοκλήρωσης. Η ιδέα αυτή μας δίνει τον ακριβή αλγόριθμο που χρειαζόμαστε για την εφαρμογή της παραπάνω τεχνικής. Ακολούθως περιγράφουμε έναν κομψό τρόπο υλοποίησης αυτού του αλγορίθμου.

Οι χαρακτηριστικές ιδιότητες μίας εφικτής χρονοδρομολόγησης σ του προβλήματος μπορούν να κωδικοποιηθούν σε ένα διάνυσμα τριών διαστάσεων της μορφής $[L_1, L_2, Z]$, όπου L_1 είναι το

συνολικό φορτίο της σ στην πρώτη μηχανή, L_2 το συνολικό φορτίο της σ στη δεύτερη μηχανή και Z ο μέσος βεβαρημένος χρόνος ολοκλήρωσης των διεργασιών στη σ. Η τιμή της αντικειμενικής συνάρτησης για τη χρονοδρομολόγηση σ μπορεί να υπολογισθεί εύκολα, σε σταθερό χρόνο, προβάλλοντας την τρίτη συντεταγμένη του διανύσματος, μέσω μίας συνάρτησης προβολής $P_3^3(x_1, x_2, x_3) = x_3$. Ο ζητούμενος αλγόριθμος περιγράφεται στο ακόλουθο σχήμα.

Αλγόριθμος 4 Ένας ακριβής αλγόριθμος για το πρόβλημα $P2 \parallel \sum_j w_j C_j$.

1. Διάταξε τις διεργασίες εκτελώντας τον κανόνα του Smith.
2. Αρίθμησε τις διεργασίες, έτσι ώστε ο χρόνος επεξεργασίας p_i να αντιστοιχεί στην i-οστή διεργασία της διάταξης.
3. $VS_1 = \{[p_1, 0, p_1 w_1],\ [0, p_1, p_1 w_1]\}$
4. **Για** $\kappa = 2$ έως n
   - 4.1 **Για** κάθε διάνυσμα $[x, y, z] \in VS_{k-1}$
     - 4.1.1 $VS_k = VS_k \bigcup \{[x + p_k, y, z + w_k(x + p_k)],\ [x, y + p_k, z + w_k(y + p_k)]\}$
5. Επέλεξε το διάνυσμα $[x, y, z] \in VS_n$ με την ελάχιστη τιμή για το z.

Στο Βήμα 4, ο αλγόριθμος χρονοδρομολογεί τη διεργασία k με δύο διαφορετικούς τρόπους: α) στο τέλος της πρώτης μηχανής, αυξάνοντας έτσι το φορτίο αυτής κατά p_k και την τιμή της αντικειμενικής συνάρτησης κατά $w_k(x+p_k)$ και β) στο τέλος της δεύτερης μηχανής, αυξάνοντας το φορτίο της κατά p_k και την τιμή της αντικειμενικής συνάρτησης κατά $w_k(y+p_k)$. Λόγω του ότι οι δύο πρώτες συντεταγμένες κάθε διανύσματος παίρνουν τιμές από 0 έως p_{sum} και η τρίτη από 0 έως $p_{sum}w_{sum}$, ο πληθικός αριθμός ενός συνόλου διανυσμάτων VS_k θα είναι το πολύ $O(p_{sum}^3 w_{sum})$. Επιπλέον, η χρονική πολυπλοκότητα του αλγορίθμου είναι ανάλογη της ποσότητας $\sum_{k=1}^{n} |VS_k|$. Επομένως, έχουμε κατασκευάσει έναν ψευδοπολυωνυμικό αλγόριθμο με πολυπλοκότητα χρόνου της τάξης του $O(np_{sum}^3 w_{sum})$.

Β. Επιτάχυνση της εκτέλεσης του αλγορίθμου. Καθένα από τα παραπάνω διανύσματα μπορεί να θεωρηθεί ως ένα γεωμετρικό σημείο του πολυέδρου $[0, p_{sum}] \times [0, p_{sum}] \times [0, p_{sum}w_{sum}]$. Υποδιαιρούμε τον κύβο αυτό σε μικρά "κουτιά" δημιουργώντας τομές και στις τρείς διαστάσεις, στις συντεταγμένες που ορίζονται από τα Δ^i για $i = 1, 2, \ldots, L$, όπου

$$\Delta = 1 + \frac{\epsilon}{2n}. \tag{3.1}$$

και όπου

$$L \leq \lceil \log_\Delta(p_{sum} w_{sum}) \rceil \leq \lceil (1 + \frac{2n}{\epsilon}) \ln(p_{sum} w_{sum}) \rceil. \tag{3.2}$$

Η δεύτερη ανισότητα της (3.2) ισχύει λόγω του ότι για κάθε $x \geq 1$, ισχύει ότι $\ln x \geq (x-1)/x$, το οποίο προκύπτει αν αναπτύξουμε τη συνάρτηση $\ln x$ σε σειρά Taylor του $x-1$.

Από την επιλογή του Δ παρατηρούμε ότι τα διανύσματα που βρίσκονται στο ίδιο κουτί είναι σε πολύ μικρή απόσταση μεταξύ τους. Έτσι, μπορούμε να απλοποιήσουμε ένα σύνολο διανυσμάτων VS_k επιλέγοντας, από κάθε κουτί που περιέχει διανύσματα αυτού, ένα μοναδικό διάνυσμα. Τα διανύσματα που συλλέγονται συνιστούν ένα νέο σύνολο διανυσμάτων VS'_k το οποίο καλείται *σύνολο αποκοπής*. Όλα τα διανύσματα του συνόλου VS_k που δεν επιλέχθηκαν δεν λαμβάνουν μέρος στους επόμενους υπολογισμούς. Έτσι, ο νέος αλγόριθμος (αλγόριθμος αποκοπής) παράγει το επόμενο σύνολο διανυσμάτων VS_{k+1} με βάση το σύνολο VS'_k και όχι το σύνολο VS_k. Για να δούμε πως διαμορφώνεται η πολυπλοκότητα χρόνου του αλγορίθμου αποκοπής, αρκεί να παρατηρήσουμε ότι κάθε σύνολο αποκοπής διανυσμάτων περιέχει το πολύ ένα διάνυσμα από κάθε κουτί. Συνολικά υπάρχουν το πολύ $O(L^3)$ κουτιά και βάσει της (3.2) το πλήθος τους θα είναι πολυωνυμικό ως προς το μέγεθος εισόδου $|I|$ και το $1/\epsilon$. Ο αλγόριθμος αποκοπής τρέχει σε χρόνο ανάλογο της ποσότητας $\sum_{k=1}^{n} |VS'_k|$, άρα η χρονική του πολυπλοκότητα θα είναι πολυωνυμική ως προς το μέγεθος εισόδου και το $1/\epsilon$.

Γ. Ανάλυση χείριστης περίπτωσης Ο αρχικός Αλγόριθμος 4 κατασκευάζει τα σύνολα διανυσμάτων $VS_1, \ldots, VS_n$ υπολογίζοντας κάθε σύνολο VS_{k+1} μέσω του συνόλου VS_k, ενώ ο αλγόριθμος αποκοπής κατασκευάζει τα σύνολα διανυσμάτων $VS'_1, \ldots, VS'_n$ υπολογίζοντας κάθε σύνολο VS'_{k+1} μέσω του συνόλου VS'_k. Η ακόλουθη πρόταση εξασφαλίζει ότι το σύνολο αποκοπής διανυσμάτων VS'_k είναι μία ικανοποιητική προσέγγιση του συνόλου VS_k και μπορεί εύκολα να αποδειχθεί εφαρμόζοντας επαγωγή στο k (βλ. σελίδα 32 στο [86]).

Πρόταση 3.1 *Για κάθε διάνυσμα $[x, y, z] \in VS_k$, υπάρχει ένα διάνυσμα $[x', y', z'] \in VS'_k$ τέτοιο ώστε να ισχύουν οι ακόλουθες ανισότητες:*

$$x' \leq \Delta^k x, \qquad y' \leq \Delta^k y \qquad \textit{και} \qquad z' \leq \Delta^k z.$$

Θεωρούμε τώρα ότι ο Αλγόριθμος 4 στο Βήμα 5 επιλέγει ένα διάνυσμα $[x, y, z]$ από το σύνολο VS_n με την ελάχιστη τιμή $P_3^3(x, y, z)$. Από την Πρόταση 3.1 προκύπτει ότι υπάρχει ένα διάνυσμα $[x', y', z']$ στο σύνολο VS'_n, όπου κάθε συντεταγμένη του είναι το πολύ ίση με Δ^n επί την αντίστοιχη συντεταγμένη του διανύσματος $[x, y, z]$. Συνεπώς, θα ισχύει ότι

$$P_3^3(x', y', z') \leq P_3^3(\Delta^n x, \Delta^n y, \Delta^n z) = \Delta^n z = \Delta^n \, \mathrm{OPT}.$$

Η τελευταία σχέση μας δίνει ένα Δ^n-προσεγγιστικό αλγόριθμο για το πρόβλημα $P2 \parallel \sum_j w_j C_j$. Μένει να υπολογίσουμε πόσο μεγάλη είναι η ποσότητα Δ^n. Χρησιμοποιούμε τη γνωστή ανισότητα $(1+p/n)^n \leq 1+2n$, η οποία ισχύει για κάθε $0 \leq p \leq 1$ και από τη σχέση (3.1), για $p = \epsilon/2$, έχουμε ότι $\Delta^n \leq 1+\epsilon$. Καταλήγουμε έτσι στο ζητούμενο FPTAS.

3.3 Χρονοδρομολόγηση με δεδομένους χρόνους αποδέσμευσης

Όταν οι διεργασίες διαθέτουν επιπλέον και κάποιο θετικό ακέραιο χρόνο αποδέσμευσης, τα περισσότερα προβλήματα ελαχιστοποίησης του μέσου βεβαρημένου χρόνου ολοκλήρωσης, εκτός ελαχίστων εξαιρέσεων (βλ. Ενότητα 3.1), γίνονται strongly **NP**-hard. Επομένως, τεχνικές όπως η τεχνική αποκοπής δεν είναι σε θέση να αποδώσουν το επιθυμητό αποτέλεσμα, που είναι ο σχεδιασμός ενός PTAS. Στα πλαίσια της ενότητας αυτής παρουσιάζουμε προσεγγιστικά σχήματα πολυωνυμικού χρόνου για προβλήματα ελαχιστοποίησης του μέσου (βεβαρημένου) χρόνου ολοκλήρωσης στην περίπτωση όπου κάθε διεργασία j διαθέτει ένα προκαθορισμένο χρόνο αποδέσμευσης r_j, πριν το πέρας του οποίου δεν επιτρέπεται να αρχίσει να εκτελείται.

Στην Ενότητα 3.3.2 παρουσιάζουμε ένα PTAS για το πρόβλημα $1 \mid r_j \mid \sum_j C_j$, το οποίο οφείλεται στους Karger και Stein [2] και στην Ενότητα 3.3.3 παρουσιάζουμε δύο PTAS για τα προβλήματα $P \mid r_j \mid \sum_j w_j C_j$ και $P \mid r_j,\ pmtn \mid \sum_j w_j C_j$. Τα αποτελέσματα αυτά οφείλονται στους Chekuri και Khanna και περιγράφονται αναλυτικά στις εργασίες [2] και [13]

Η κεντρική ιδέα των προσεγγιστικών σχημάτων έγκειται στην εφαρμογή πολλαπλών μετασχηματισμών στο αρχικό πρόβλημα έτσι ώστε να είναι εφικτή η αποδοτική επίλυση του είτε με χρήση δυναμικού προγραμματισμού, είτε με εφαρμογή κάποιου κανόνα χρονοδρομολόγησης. Οι περισσότεροι από τους μετασχηματισμούς εφαρμόζονται στην βέλτιστη λύση του προβλήματος (την οποία προφανώς υποθέτουμε) έτσι ώστε να μπορούμε να ισχυριστούμε ότι μία λύση με τιμή κοντά σ' αυτή της βέλτιστης έχει σαφώς απλούστερη δομή. Οι υπόλοιποι μετασχηματισμοί εφαρμόζονται στην είσοδο του προβλήματος με σκοπό την απλοποίηση αυτής. Στην Ενότητα 3.3.1 που ακολουθεί περιγράφουμε αναλυτικά όλους αυτούς τους μετασχηματισμούς, οι οποίοι διακρίνονται σε δύο κατηγορίες και χαρακτηρίζονται συνολικά ως *μετασχηματισμοί εισόδου*.

3.3.1 Μετασχηματισμοί εισόδου

Ο στόχος των μετασχηματισμών εισόδου είναι να απλοποιήσουν το στιγμιότυπο εισόδου κατά τέτοιο τρόπο έτσι ώστε να διευκολύνουν την αποδοτική απαρίθμηση των εφικτών χρονοδρομολογήσεων, αλλά και την εφαρμογή δυναμικού προγραμματισμού.

Κάθε μετασχηματισμός ενδεχομένως αυξάνει την τιμή της αντικειμενικής συνάρτησης κατά ένα

παράγοντα $1 + O(\epsilon)$, όπου ϵ θετική σταθερά. Επομένως, εφαρμόζοντας ένα σταθερό αριθμό μετασχηματισμών αυξάνουμε την αρχική τιμή της βέλτιστης λύσης μόνο κατά ένα σταθερό παράγοντα. Στη συνέχεια όταν αναφερόμαστε σε κάποιο μετασχηματισμό θα λέμε ότι προκαλεί *απώλεια* $1 + O(\epsilon)$. Για λόγους απλοποίησης του προβλήματος, σε όλες τις ενότητες που ακολουθούν υποθέτουμε ότι το $1/\epsilon$ είναι θετικός ακέραιος και επιπλέον ότι $\epsilon \leq 1/4$. Επίσης χρησιμοποιούμε το συμβολισμό C_j και S_j για να δηλώσουμε το χρόνο ολοκλήρωσης και το χρόνο έναρξης της εκτέλεσης κάθε διεργασίας j.

Οι μετασχηματισμοί που εφαρμόζουμε διακρίνονται σε δύο κατηγορίες. Η πρώτη καλείται *γεωμετρική στρογγυλοποίηση (geometric rounding)* και περιλαμβάνει μετασχηματισμούς που αφορούν στη διακριτοποίηση των μεγεθών του στιγμιοτύπου εισόδου. Η δεύτερη κατηγορία περιλαμβάνει μετασχηματισμούς οι οποίοι εξασφαλίζουν ότι μόνο ένας μικρός αριθμός διεργασιών μπορεί να καταφθάνει σε κάθε χρονική στιγμή αποδέσμευσης. Οι μετασχηματισμοί της δεύτερης κατηγορίας καλούνται μετασχηματισμοί *μετατόπισης διεργασιών (job shifting)* (βλ. [13]).

Σημειώνουμε ότι οι ιδιότητες που αποδεικνύονται στις ακόλουθες δύο ενότητες ισχύουν τόσο στη γενική περίπτωση των παράλληλων μηχανών, $P\,|\,r_j|\,\sum_j w_jC_j$ και $P\,|\,r_j,\ pmtn|\,\sum_j w_jC_j$, όσο και στην περίπτωση της μοναδικής μηχανής, $1\,|\,r_j|\,\sum_j w_jC_j$ και $1\,|\,r_j|\,\sum_j C_j$. Τα αποτελέσματα που ακολουθούν παρουσιάζονται αναλυτικά στις εργασίες [13] και [2].

3.3.1.1 Γεωμετρική στρογγυλοποίηση

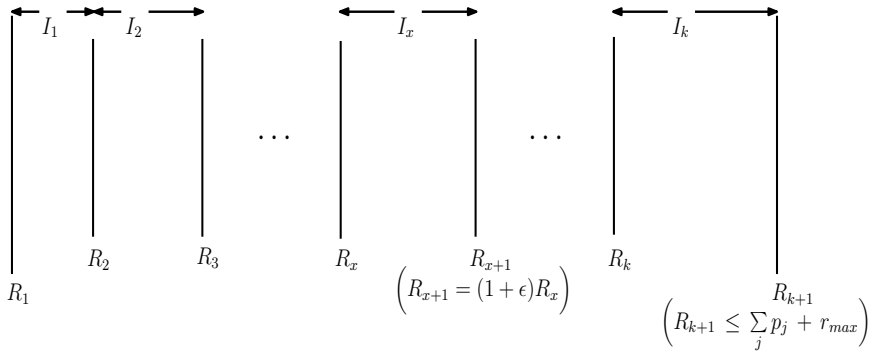
Ο πρώτος μετασχηματισμός έχει ως αποτέλεσμα τη μείωση του αριθμού των διακεκριμένων χρόνων επεξεργασίας και αποδέσμευσης, σε ένα δοθέν στιγμιότυπο εισόδου.

Λήμμα 3.1 *Με απώλεια* $1 + \epsilon$ *μπορούμε να υποθέσουμε ότι κάθε χρόνος επεξεργασίας και κάθε χρόνος αποδέσμευσης είναι ακέραια δύναμη του* $1 + \epsilon$.

Απόδειξη. Για κάθε διεργασία j, πολλαπλασιάζουμε τον χρόνο επεξεργασίας της p_j και το χρόνο αποδέσμευσης αυτής r_j με $1 + \epsilon$. Έτσι, η τιμή της αντικειμενικής συνάρτησης αυξάνεται επίσης κατά $1 + \epsilon$. Στρογγυλοποιούμε προς τα κάτω, στην αμέσως μικρότερη ακέραια δύναμη του $1 + \epsilon$, όλους τους νέους χρόνους αποδέσμευσης και επεξεργασίας για κάθε διεργασία. Οι τιμές που προκύπτουν από τη στρογγυλοποίηση θα είναι μεγαλύτερες από τις αντίστοιχες αρχικές, ενώ η τιμή της αντικειμενικής συνάρτησης θα είναι αυξημένη το πολύ κατά ένα παράγοντα $1 + \epsilon$. □

Για έναν αυθαίρετο θετικό ακέραιο x θέτουμε $R_x = (1+\epsilon)^x$. Βάσει του Λήμματος 3.1 μπορούμε να υποθέσουμε ότι κάθε χρόνος αποδέσμευσης θα είναι της μορφής R_x, για κάποιο x. Ορίζουμε έτσι μία διαμέριση του χρονικού ορίζοντα $[0,\ \infty)$ σε ξένα ανά δύο διαστήματα της μορφής $I_x = [R_x,\ R_{x+1})$, $x = 1, \ldots, k$ όπου $R_{k+1} \leq \sum_j p_j + r_{max}$ (βλ. Σχήμα 3.1). Στο εξής με I_x θα συμβολίζουμε

τόσο το αντίστοιχο διάστημα, όσο και το μήκος αυτού, το οποίο ισούται με $R_{x+1} - R_x$. Επομένως, το I_x θα ισούται με ϵ φορές το χρόνο έναρξης του αντίστοιχου διαστήματος, $I_x = \epsilon R_x$.



Σχήμα 3.1: Διαμέριση του χρονικού ορίζοντα σε διαστήματα με εφαρμογή γεωμετρικής στρογγυλοποίησης.

3.3.1.2 Μετατόπιση Διεργασιών

Η μετατόπιση διεργασιών έχει ως κύριο στόχο να εξασφαλίσει για κάθε διεργασία ότι η διαφορά μεταξύ του χρόνου αποδέσμευσης και του χρόνου ολοκλήρωσης αυτής θα εκτείνεται σε ένα σταθερό αριθμό διαστημάτων της μορφής I_x, για κάποιο ακέραιο x.

Αν θεωρήσουμε ένα στιγμιότυπο εισόδου του προβλήματος, τότε η παραπάνω ιδιότητα μπορεί να παραβιαστεί με τους ακόλουθους δύο τρόπους: α) μία διεργασία που αποδεσμεύεται σε κάποια χρονική στιγμή R_x, για κάποιο ακέραιο x, έχει χρόνο επεξεργασίας κατά πολύ μεγαλύτερο από $(1+\epsilon)^x$ και επομένως αρχίζει να εκτελείται μετά το πέρας ενός αρκετά μεγάλου αριθμού διαδοχικών διαστημάτων της μορφής I_x και β) πολλές διεργασίες αποδεσμεύονται ταυτόχρονα σε κάποια χρονική στιγμή R_x και δεν είναι εφικτό να ολοκληρώσουν την εκτέλεση τους μετά το πέρας ενός σταθερού αριθμού διαστημάτων της μορφής I_x. Για την αντιμετώπιση της πρώτης περίπτωσης μπορούμε απλώς να καθυστερήσουμε την αποδέσμευση κάθε διεργασίας αυξάνοντας τον αντίστοιχο χρόνο αποδέσμευσης. Η δεύτερη περίπτωση παρουσιάζει δυσκολίες και απαιτεί περισσότερη ανάλυση για να αντιμετωπιστεί.

Θα ξεκινήσουμε την επιμέρους ανάλυση όσων προαναφέραμε αφού πρώτα κατηγοριοποιήσουμε το σύνολο των διεργασιών σε μικρές και μεγάλες ως προς το χρόνο επεξεργασίας τους και το διάστημα στο οποίο αποδεσμεύονται. Στην περίπτωση του προβλήματος $1 \mid r_j \mid \sum_j C_j$ μία διεργασία j καλείται *μικρή* αν για το χρόνο επεξεργασίας της p_j ισχύει ότι $p_j \leq \epsilon I_x$, όπου I_x είναι το διάστημα στο οποίο αποδεσμεύεται η j, για κάποιο ακέραιο x. Σε αντίθετη περίπτωση η διεργασία καλείται *μεγάλη*. Αντίστοιχα, στην περίπτωση του προβλήματος $P \mid r_j \mid \sum_j w_j C_j$, μία διεργασία j καλείται *μικρή* αν ισχύει ότι $p_j \leq \epsilon^2 I_x$ και *μεγάλη* αν $p_j > \epsilon^2 I_x$. Είναι σημαντικό να

παρατηρήσουμε ότι, αν μία διεργασία j είναι μικρή για το διάστημα I_x στο οποίο αποδεσμεύεται, τότε θα είναι μικρή και για το διάστημα στο οποίο εκτελείται, καθώς ισχύει ότι $S_j \geq R_x$.

Τα δύο λήμματα που ακολουθούν εξασφαλίζουν ότι καμία διεργασία δεν μπορεί να είναι αυθαίρετα μεγάλη σε σύγκριση με το μήκος του διαστήματος στο οποίο αποδεσμεύεται.

Λήμμα 3.2 *Με απώλεια $1+\epsilon$ μπορούμε να εξασφαλίσουμε ότι $r_j \geq \epsilon p_j$ για κάθε διεργασία j.*

Απόδειξη. Για κάθε διεργασία j, αυξάνουμε το χρόνο επεξεργασίας της κατά ένα παράγοντα $1+\epsilon$, από p_j σε $(1+\epsilon)p_j$. Όπως και στο Λήμμα 3.1 η τιμή της αντικειμενικής συνάρτησης αυξάνεται επίσης κατά $1+\epsilon$. Θεωρούμε μία βέλτιστη χρονοδρομολόγηση για το στιγμιότυπο που προκύπτει και για κάθε διεργασία j αγνοούμε τις πρώτες ϵp_j μονάδες από το χρόνο εκτέλεσης αυτής. Προκύπτει ότι κάθε διεργασία j αρχίζει να εκτελείται είτε κατά τη χρονική στιγμή ϵp_j, είτε αργότερα, έχοντας στη διάθεση της χρόνο επεξεργασίας ίσο με p_j.

Η απόδειξη του λήμματος ολοκληρώνεται αν αυξήσουμε το χρόνο αποδέσμευσης κάθε διεργασίας j έτσι ώστε να ισχύει ότι $r_j \geq \epsilon p_j$. Η τιμή της αντικειμενικής συνάρτησης συνεχίζει να είναι ίση με το πολύ $1+\epsilon$ επί τη βέλτιστη τιμή του αρχικού στιγμιοτύπου. □

Το Λήμμα 3.2, εκτός των άλλων, εξασφαλίζει ότι καμία διεργασία δεν αποδεσμεύεται τη χρονική στιγμή 0. Από το ίδιο λήμμα προκύπτει επίσης ότι σε μία non-preemptive χρονοδρομολόγηση, κάθε διεργασία δεν μπορεί να διασχίζει[1] αυθαίρετο αριθμό διαστημάτων.

Πόρισμα 3.1 *Κάθε διεργασία διασχίζει το πολύ $s = \lceil \log_{1+\epsilon}(1+\frac{1}{\epsilon}) \rceil$ διαστήματα.*

Απόδειξη. Θεωρούμε μία διεργασία j η οποία αρχίζει να εκτελείται σε ένα διάστημα $I_x = [R_x, R_{x+1})$, για κάποιο ακέραιο x. Βάσει του Λήμματος 3.2 ισχύει ότι $r_j \geq \epsilon p_j$ και επιπλέον $R_x \geq r_j$. Επομένως, προκύπτει ότι $I_x = \epsilon R_x \geq \epsilon^2 p_j$ ή ισοδύναμα $I_x/\epsilon^2 \geq p_j$.

Το ζητούμενο προκύπτει αν εξισώσουμε το άθροισμα των διαστημάτων $I_{x'}$, όπου $x \leq x' \leq x+s$, με το I_x/ϵ^2 και λύσουμε ως προς s. □

Λήμμα 3.3 *Θεωρούμε ένα στιγμιότυπο του προβλήματος $P \mid r_j \mid \sum_j w_j C_j$. Υπάρχει μία $(1+O(\epsilon))$-προσεγγιστική χρονοδρομολόγηση των διεργασιών τέτοια ώστε να ισχύουν τα εξής:*

1. *Αν για οποιεσδήποτε δύο μικρές διεργασίες i και j ισχύει ότι $r_i \leq r_j$ και $\frac{p_i}{w_i} \leq \frac{p_j}{w_j}$ και $i < j$, τότε $x_i \leq x_j$, όπου x_i, x_j είναι τα διαστήματα στα οποία αρχίζουν να εκτελούνται οι διεργασίες i και j αντίστοιχα.*
2. *Η εκτέλεση κάθε μικρής διεργασίας ολοκληρώνεται εντός του διαστήματος στο οποίο αρχίζει.*

[1] Ο όρος διασχίζει αναφέρεται στο χρόνο που διαρκεί η εκτέλεση της εκάστοτε διεργασίας, ο οποίος μεταφράζεται σε αριθμό διαστημάτων I_x.

Απόδειξη. Θεωρούμε μία βέλτιστη χρονοδρομολόγηση και έστω s_x ο απαιτούμενος χρόνος για την ολοκλήρωση της εκτέλεσης των μικρών διεργασιών εντός του διαστήματος I_x, για κάποιο ακέραιο x. Υποθέτουμε ότι οι μικρές διεργασίες εκτελούνται διαδοχικά στο τέλος του αντίστοιχου διαστήματος, γεγονός που αυξάνει την τιμή της αντικειμενικής συνάρτησης το πολύ κατά ένα παράγοντα $1+\epsilon$. Συμβολίζουμε με σ_1 τη χρονοδρομολόγηση αυτή.

Για να ικανοποιήσουμε την πρώτη συνθήκη του λήμματος, κατασκευάζουμε μία νέα χρονοδρομολόγηση: αναθέτουμε τις μικρές διεργασίες σε διαστήματα χωρίς παράλληλα να καθορίσουμε επακριβώς τη χρονοδρομολόγηση τους σ' αυτά. Η ανάθεση γίνεται αναδρομικά ως εξής: έστω ότι έχουμε αναθέσει κάποιες μικρές διεργασίες στα διαστήματα I_1 έως I_{x-1} και θέλουμε να συνεχίσουμε στο διάστημα I_x. Ορίζουμε με A_x το σύνολο των μικρών διεργασιών που έχουν αποδεσμευτεί μέχρι και το διάστημα I_x και που δεν έχουν ακόμα ανατεθεί σε κάποιο προηγούμενο διάστημα. Σε κάθε διεργασία j του A_x αντιστοιχίζουμε μία διατεταγμένη τριάδα $(p_j/w_j, r_j, j)$ και ταξινομούμε τις διεργασίες σε αύξουσα, λεξικογραφική, διάταξη ως προς τις διατεταγμένες τριάδες τους. Βάσει της διάταξης αυτής, αναθέτουμε τις διεργασίες του A_x στο διάστημα I_x έως ότου ο συνολικός χρόνος επεξεργασίας τους υπερβεί για πρώτη φορά την τιμή s_x. Λόγω του ότι όλες οι διεργασίες του A_x είναι μικρές, ο συνολικός χρόνος επεξεργασίας τους θα είναι το πολύ ίσος με $s_x + \epsilon I_x$.

Σκοπός μας είναι να ελέγξουμε αν με την εφαρμογή της παραπάνω διαδικασίας ανάθεσης, δημιουργείται κάποια απώλεια σε σχέση με τη χρονοδρομολόγηση σ_1. Μία προφανής παρατήρηση, που προκύπτει άμεσα από την κατασκευή της διαδικασίας ανάθεσης, είναι ότι ο απαιτούμενος χρόνος για την ολοκλήρωση της εκτέλεσης μικρών διεργασιών εντός ενός διαστήματος I_x θα είναι το πολύ ίσος με $s_x+\epsilon I_x$, δηλαδή μεγαλύτερος ή ίσος από τον αντίστοιχο χρόνο στη σ_1. Μία δεύτερη παρατήρηση, όχι τόσο προφανής, είναι ότι το συνολικό βάρος των διεργασιών που ανατέθηκαν στα διαστήματα I_1 έως I_x, θα είναι τουλάχιστον ίσο με το συνολικό βάρος των μικρών διεργασιών που ολοκληρώνονται στα ίδια διαστήματα κατά τη σ_1: κατά τη διαδικασία ανάθεσης οι διεργασίες ανατίθενται σε κάθε διάστημα βάσει του κανόνα του Smith, οπότε αυτές με το μεγαλύτερο βάρος ανα μονάδα χρόνου επεξεργασίας προηγούνται στη σειρά εκτέλεσης. Λαμβάνοντας υπόψην ότι ο συνολικός χρόνος επεξεργασίας των μικρών διεργασιών, κατά τη διαδικασία ανάθεσης, $s_x + \epsilon I_x$, είναι τουλάχιστον όσος και ο αντίστοιχος χρόνος κατά τη σ_1, συμπεραίνουμε ότι το συνολικό βάρος των μικρών διεργασιών κάθε διαστήματος κατά τη διαδικασία ανάθεσης που περιγράψαμε, θα είναι μεγαλύτερο ή ίσο από το αντίστοιχο βάρος κατα τη σ_1. Για να εκτελέσουμε τώρα όλες τις μικρές διεργασίες που ανατίθενται σε κάποιο I_x στον αντίστοιχο χώρο που δεσμεύται για την εκτέλεση μικρών διεργασιών κατά τη σ_1, αρκεί να αυξήσουμε το μήκος κάθε διαστήματος I_x κατά παράγοντα $1+\epsilon$. Η επέκταση αυτή των διαστημάτων θα επιφέρει αύξηση της αντικειμενικής συνάρτησης κατά έναν παράγοντα $1+\epsilon$. Λόγω του ότι η τιμή της αντικειμενικής συνάρτησης

της σ_1 είναι ήδη αυξημένη κατά το πολύ $1+\epsilon$ σε σχέση με τη βέλτιστη χρονοδρομολόγηση, η συνολική αύξηση στην τιμή της αντικειμενικής συνάρτησης μετά την ολοκλήρωση της παραπάνω διαδικασίας ανάθεσης θα είναι της τάξης του $1+O(\epsilon)$.

Για να ικανοποιήσουμε την δεύτερη συνθήκη του λήμματος αρκεί απλώς να επεκτείνουμε κάθε διάστημα I_x, αυξάνοντας το μήκος του σε $(1+\epsilon)I_x$. Είναι εφικτό έτσι, για κάθε μικρή διεργασία που αρχίζει να εκτελείται σε κάποιο διάστημα, να μπορεί να ολοκληρώσει την εκτέλεση της εντός του διαστήματος αυτού. □

Σημειώνουμε ότι το Λήμμα 3.3 εφαρμόζεται και στην preemtive εκδοχή του προβλήματος, $P\,|\,r_j,\ pmtn|\ \sum_j w_jC_j$.

Βάσει του Λήμματος 3.3 μπορούμε να ταξινομήσουμε ως προς το λόγο p_j/w_j, όλες τις μικρές διεργασίες που αποδεσμεύονται την κάθε χρονική στιγμή R_x και να θεωρήσουμε ότι η χρονοδρομολόγηση τους θα γίνει με βάση τη διάταξη αυτή. Στο εξής θα συμβολίζουμε με T_x το σύνολο των μικρών διεργασιών που αποδεσμεύονται τη χρονική στιγμή R_x και με H_x το αντίστοιχο σύνολο για τις μεγάλες διεργασίες. Θυμίζουμε ότι στην περίπτωση του προβλήματος $P\,|\,r_j|\ \sum_j w_jC_j$ οι μικρές διεργασίες ενός διαστήματος I_x έχουν χρόνο επεξεργασίας μικρότερο από $\epsilon^2 I_x$. Επιπλέον, ορίζουμε με $p(A)$ το συνολικό χρόνο επεξεργασίας (ή συνολικό μέγεθος) των διεργασιών ενός συνόλου A. Το λήμμα που ακολουθεί εξασφαλίζει ότι με μικρή απώλεια μπορούμε να μετασχηματίσουμε κατάλληλα ένα στιγμιότυπο του $P\,|\,r_j|\ \sum_j w_jC_j$ έτσι ώστε το συνολικό μέγεθος των μικρών και των μεγάλων διεργασιών που αποδεσμεύονται τη χρονική στιγμή R_x να είναι της τάξης του $O(mI_x)$. Το λήμμα αυτό είναι πολύ σημαντικό για την εφαρμογή δυναμικού προγραμματισμού, όπως θα δούμε στην Ενότητα 3.3.3, καθώς μας επιτρέπει να αναπαριστούμε με συμπαγή τρόπο την πληροφορία για τις διεργασίες που δεν έχουν ακόμη ολοκληρώσει την εκτέλεση τους κατά τη χρονοδρομολόγηση.

Λήμμα 3.4 *Με απώλεια $1+O(\epsilon)$ μπορούμε να μετασχηματίσουμε ένα στιγμιότυπο I του προβλήματος $P\,|\,r_j|\ \sum_j w_jC_j$, σε ένα στιγμιότυπο I' τέτοιο ώστε να ισχύουν οι ακόλουθες συνθήκες:*

1. *$p(T'_x) \le (1+\epsilon)mI_x$, για όλα τα x.*
2. *Ο αριθμός των διακεκριμένων χρόνων επεξεργασίας των διεργασιών του συνόλου H'_x είναι το πολύ $\lfloor 1+4\log_{1+\epsilon}\frac{1}{\epsilon}\rfloor$.*
3. *Το πλήθος των διεργασιών του συνόλου H'_x που αντιστοιχούν σε κάθε διακεκριμένο χρόνο επεξεργασίας είναι το πολύ $\frac{m}{\epsilon^2}$.*

Απόδειξη. Αρχικά παρατηρούμε ότι στο στιγμιότυπο εισόδου I, ο συνολικός διαθέσιμος χρόνος επεξεργασίας σε κάθε διάστημα I_x ισούται με mI_x. Ταξινομούμε τις μικρές διεργασίες του T_x

σε αύξουσα διάταξη ως προς το λόγο p_j/w_j και αρχίζουμε να τις εκτελούμε βάσει αυτής. Την πρώτη φορά που ο συνολικός χρόνος επεξεργασίας υπερβεί την τιμή $(1+\epsilon)mI_x$ αναιρούμε την εκτέλεση της τελευταίας διεργασίας. Οι διεργασίες που δεν εκτελέσθηκαν εντός του διαστήματος I_x, ενώ αποδεσμεύτηκαν τη χρονική στιγμή R_x, μετατοπίζονται στην επόμενη χρονική στιγμή αποδέσμευσης R_{x+1}.

Για τη Συνθήκη 2, βάσει του Λήμματος 3.2, έχουμε ότι για κάθε διεργασία j του συνόλου H_x θα ισχύει ότι $R_x \geq \epsilon p_j$. Επιπλέον, αφού κάθε διεργασία του H_x είναι μεγάλη, θα ισχύει ότι $p_j > \epsilon^2 I_x = \epsilon^3 R_x$. Από τις δύο αυτές σχέσεις και από το γεγονός ότι κάθε χρόνος επεξεργασίας είναι ακέραια δύναμη του $1+\epsilon$ (βλ. Λήμμα 3.1) προκύπτει η ανίσωση $\frac{R_x}{\epsilon}/\epsilon^3 R_x \leq (1+\epsilon)^k$, για κάποιο θετικό ακέραιο k. Λύνοντας ως προς k έχουμε το ζητούμενο. Για την τρίτη συνθήκη του λήμματος έχουμε ότι το πλήθος των διεργασιών οι οποίες αντιστοιχούν σε μία από τις k διακεκριμένες τιμές και μπορούν να εκτελεσθούν εντός του διαστήματος I_x, δεν μπορεί να είναι περισσότερες από $\frac{mI_x}{\epsilon^3 R_x} = \frac{m}{\epsilon^2}$.

Σημειώνουμε τέλος ότι για να χρονοδρομολογήσουμε τις μεγάλες διεργασίες που αποδεσμεύονται σε ένα διάστημα I_x και έχουν ίδιο μέγεθος, τις διατάσσουμε σε φθίνουσα διάταξη ως προς το βάρος τους και τις εκτελούμε βάσει της διάταξης αυτής. □

3.3.2 Ελαχιστοποίηση του συνολικού χρόνου ολοκλήρωσης σε Μοναδική Μηχανή

Στην ενότητα αυτή παρουσιάζουμε ένα προσεγγιστικό σχήμα πολυωνυμικού χρόνου για το πρόβλημα $1\,|\,r_j\,|\,\sum_j C_j$, το οποίο οφείλεται στους Karger και Stein [2].

Θυμίζουμε ότι το πρόβλημα αυτό είναι ειδική περίπτωση του προβλήματος $1\,|\,r_j\,|\,\sum_j w_j C_j$, όπου τα βάρη των διεργασιών είναι μοναδιαία. Η ανάλυση του είναι ιδιαίτερα απλή, παρόλα αυτά δεν μπορεί να επεκταθεί ώστε να δώσει λύση και στη περίπτωση όπου τα βάρη των διεργασιών διαφέρουν μεταξύ τους. Η τελευταία εκδοχή αντιμετωπίζεται στα πλαίσια του σχεδιασμού ενός PTAS για το γενικότερο πρόβλημα $P\,|\,r_j\,|\,\sum_j w_j C_j$ και απαιτεί διαφορετική και βαθύτερη ανάλυση (βλ. Ενότητα 3.3.3).

Η κεντρική ιδέα του αποτελέσματος είναι να απλοποιήσουμε το αρχικό στιγμιότυπο εισόδου εφαρμόζοντας μετασχηματισμούς στρογγυλοποίησης και μετατόπισης έτσι ώστε να μπορέσουμε να χρονοδρομολογήσουμε το μεγαλύτερο μέρος των διεργασιών βάσει του κανόνα SPT. Οι εναπομείνασες διεργασίες θα είναι σταθερές στο πλήθος και επομένως, σε σταθερό χρόνο μπορούμε να απαριθμήσουμε όλες τις εφικτές χρονοδρομολογήσεις τους. Υπενθυμίζουμε ότι ο κανόνας SPT (βλ. Ενότητα 1.2) επιλέγει επαναληπτικά τη διεργασία με τον μικρότερο χρόνο επεξεργασίας, από τις διεργασίες που έχουν αποδεσμευτεί, αλλά δεν έχουν ακόμα εκτελεσθεί.

Το επικείμενο PTAS για το πρόβλημα $1\,|\,r_j|\,\sum_j C_j$ συνοψίζεται στον Αλγόριθμο 5 που ακολουθεί. Σημειώνουμε ότι κατά τη διάρκεια της εκτέλεσης του κανόνα SPT έχουμε υποθέσει ότι κάθε διεργασία j αποδεσμεύεται σε χρόνο $\max\{r_j, \frac{p_j}{\epsilon^2}\}$. Ο χρόνος αυτός είναι αυξημένος κατά ένα παράγοντα $1+3\epsilon$ σε σχέση με τον αρχικό χρόνο αποδέσμευσης της j.

Αλγόριθμος 5 Ένα PTAS για το πρόβλημα $1\,|\,r_j|\,\sum_j C_j$.

1. Εκτέλεσε τον κανόνα SPT έως ότου απομείνουν $\frac{3}{\epsilon^7}$ διεργασίες.
2. Απαρίθμησε όλες τις εφικτές χρονοδρομολογήσεις των εναπομεινασών διεργασιών και επέλεξε την καλύτερη δυνατή.

Για την ταξινόμηση των διεργασιών κατά την εφαρμογή του κανόνα SPT, απαιτείται χρόνος $O(n\log n)$. Για την απαρίθμηση στο Βήμα 2 του αλγορίθμου απαιτείται χρόνος $(3/\epsilon^7)!$. Συνολικά η πολυπλοκότητα χρόνου του αλγορίθμου είναι της τάξης του $O(n\log n + (3/\epsilon^7)!)$.

Στη συνέχεια αποδεικνύουμε ότι ο Αλγόριθμος 5 αποδίδει μία $(1+\epsilon)$-προσεγγιστική λύση για το πρόβλημα $1\,|\,r_j|\,\sum_j C_j$. Υπενθυμίζουμε ότι μία διεργασία j καλείται μικρή στο διάστημα I_x στο οποίο εκτελείται, αν για το χρόνο επεξεργασίας της ισχύει ότι $p_j \le \epsilon I_x$.

Λήμμα 3.5 *Αν σε μία βέλτιστη χρονοδρομολόγηση κάθε διεργασία είναι μικρή, τότε ο Αλγόριθμος 5 δίνει μία* $(1+\epsilon)$*-προσεγγιστική λύση.*

Απόδειξη. Θεωρούμε μία βέλτιστη χρονοδρομολόγηση των διεργασιών. Για κάθε μικρή διεργασία j σε ένα διάστημα I_x, παρατηρούμε ότι ο χρόνος έναρξης εκτέλεσης της θα είναι $S_j \ge R_x = I_x/\epsilon \ge p_j/\epsilon^2$. Επομένως, η αύξηση στους χρόνους αποδέσμευσης στον Αλγόριθμο 5 δεν αλλάζει ούτε την εφικτότητα, αλλά ούτε και τη βελτιστότητα της χρονοδρομολόγησης. Με βάση τους νέους χρόνους αποδέσμευσης θεωρούμε την preemptive εκδοχή του προβλήματος η οποία μπορεί να λυθεί βέλτιστα σε πολυωνυμικό χρόνο κάνοντας χρήση του κανόνα SRPT (βλ. [7]). Η τιμή αυτής της βέλτιστης λύσης θα είναι το πολύ ίση με την αντίστοιχη τιμή της non-preemptive χρονοδρομολόγησης (βλ. [69]). Έτσι μπορούμε, με μικρή απώλεια, να μετατρέψουμε τη βέλτιστη preemptive χρονοδρομολόγηση σε non-preemptive: αρκεί να παρατηρήσουμε ότι όλες οι διεργασίες αποδεσμεύονται στα άκρα ενός διαστήματος. Έτσι, αν μία διεργασία αποδεσμεύεται σε κάποια χρονική στιγμή R_x και εκτελείται στο τρέχον διάστημα I_x, μπορεί να διακόψει την εκτέλεση της μόνο κατά τη χρονική στιγμή R_{x+1}, κατά την οποία αποδεσμεύεται μία νέα διεργασία μικρότερου μεγέθους. Λαμβάνοντας υπόψη την υπόθεση ότι όλες οι διεργασίες είναι μικρές στο διάστημα στο οποίο εκτελούνται, αρκεί να αυξήσουμε το μήκος κάθε διαστήματος I_x σε $(1+\epsilon)I_x$ έτσι ώστε η διεργασία που διακόπτεται να μπορεί να ολοκληρώσει την εκτέλεση της στο ίδιο διάστημα και

συγκεκριμένα στον επιπλέον χρόνο ϵI_x που προσθέσαμε.

Η παραπάνω αύξηση στο μέγεθος κάθε διαστήματος, προκαλεί αύξηση στο χρόνο ολοκλήρωσης κάθε διεργασίας το πολύ κατά ένα παράγοντα $1+\epsilon$. Επομένως, η τιμή της αντικειμενικής συνάρτησης της non-preemptive χρονοδρομολόγησης θα είναι το πολύ ίση με $1+\epsilon$ επί την τιμή της βέλτιστης preemptive χρονοδρομολόγησης, άρα και το πολύ ίση με $1+\epsilon$ επί την τιμή της βέλτιστης non-preemptive χρονοδρομολόγησης. □

Είναι προφανές ότι το Λήμμα 3.5 δεν αρκεί από μόνο του για να μας δώσει το ζητούμενο PTAS και αυτό γιατί πολλές από τις διεργασίες θα είναι μεγάλες στα διαστήματα στα οποία εκτελούνται. Στη συνέχεια κάνουμε χρήση κάποιων εκ των μετασχηματισμών μετατόπισης διεργασιών που περιγράψαμε στην Ενότητα 3.3 έτσι ώστε να μετατρέψουμε, με μικρή απώλεια, το μεγαλύτερο μέρος των μεγάλων διεργασιών σε μικρές.

Για ένα δεδομένο στιγμιότυπο εισόδου, συμβολίζουμε με OPT την τιμή της αντικειμενικής συνάρτησης της βέλτιστης χρονοδρομολόγησης και ορίζουμε ένα χρονικό κατώφλι t να ισούται με $t = \epsilon^7 \,\mathrm{OPT}$. Εύκολα παρατηρούμε ότι το πολύ $\frac{1}{\epsilon^7}$ διεργασίες μπορούν να ολοκληρώσουν την εκτέλεση τους μετά το πέρας του χρόνου t. Το λήμμα που ακολουθεί αποδεικνύει ότι όλες οι μεγάλες διεργασίες που εκτελούνται πριν τη χρονική στιγμή t στη βέλτιστη χρονοδρομολόγηση, μπορούν να καθυστερήσουν την εκτέλεση τους, με μικρή απώλεια, έως ότου γίνουν μικρές. Υπενθυμίζουμε ότι αν μία διεργασία j είναι μικρή για το διάστημα στο οποίο εκτελείται, τότε ισχύει ότι $S_j \geq p_j/\epsilon^2$.

Λήμμα 3.6 *Υπάρχει μία* $(1+3\epsilon)$*-προσεγγιστική χρονοδρομολόγηση των διεργασιών τέτοια ώστε, για κάθε διεργασία* j*, να ισχύει ότι* $S_j \geq \min\left\{\frac{p_j}{\epsilon^2}, t\right\}$.

Απόδειξη. Θεωρούμε μία βέλτιστη χρονοδρομολόγηση και έστω x_j ο δείκτης του διαστήματος στο οποίο αρχίζει να εκτελείται μία διεργασία j. Κάθε μικρή διεργασία j θα ικανοποιεί τη συνθήκη του λήμματος, καθώς ισχύει ότι $S_j \geq p_j/\epsilon^2$.

Για τις μεγάλες διεργασίες, μετατοπίζουμε κατά $k = \lceil \log_{1+\epsilon} \frac{1}{\epsilon^4} \rceil$ διαστήματα προς τα εμπρός όσες απ᾽ αυτές αρχίζουν να εκτελούνται πριν το t. Έστω S'_j ο νέος χρόνος έναρξης εκτέλεσης κάθε τέτοιας διεργασίας j και $x'_j = x_j + k$ ο δείκτης του διαστήματος στο οποίο αυτή μετατοπίστηκε. Προκύπτει έτσι, ότι $I_{x'_j} \geq I_{x_j}/\epsilon^4$ ή (ισοδύναμα) $R_{x'_j} \geq R_{x_j}/\epsilon^4$. Λαμβάνοντας υπόψη το Λήμμα 3.2 και την προφανή σχέση $S'_j \geq R_{x'_j}$ έχουμε ότι

$$\epsilon p_j \leq r_j \leq R_{x_j} \leq \epsilon^4 R_{x'_j} \leq \epsilon^4 S'_j. \tag{3.3}$$

Διαιρώντας με ϵ^4 όλα τα μέλη της (3.3) προκύπτει η σχέση

$$S'_j \geq \frac{p_j}{\epsilon^3} \geq \frac{p_j}{\epsilon^2}. \tag{3.4}$$

Η (3.4) αποδεικνύει ότι κάθε μεγάλη διεργασία j που άρχισε να εκτελείται πριν το t, γίνεται μικρή στο νέο της διάστημα $I_{x'_j}$.

Για να εκτελέσουμε τις διεργασίες που μετατοπίζονται απαιτείται η δημιουργία κάποιου επιπλέον χώρου στα αντίστοιχα διαστήματα. Αυξάνουμε έτσι το μήκος κάθε διαστήματος I_x σε $(1+\epsilon)I_x$. Λαμβάνοντας υπόψη ότι κάθε μεγάλη διεργασία j του διαστήματος I_x έχει χρόνο επεξεργασίας $p_j > \epsilon I_x$, παρατηρούμε ότι εντός του I_x μπορούν να ξεκινήσουν την εκτέλεση τους το πολύ $I_x/\epsilon I_x = 1/\epsilon$ μεγάλες διεργασίες. Επομένως, το πολύ $1/\epsilon$ μεγάλες διεργασίες μπορεί να μετατοπιστούν από ένα διάστημα I_x σε ένα διάστημα $I_{x'}$. Από την (3.3), για κάθε μία από αυτές, ισχύει ότι $p_j \leq \epsilon^3 R_{x'} = \epsilon^2 I_{x'}$. Συνεπώς, ο συνολικός χρόνος επεξεργασίας τους θα είναι το πολύ $\epsilon I_{x'}$. Λόγω της αρχικής αύξησης στο μήκος του διαστήματος $I_{x'}$, μπορούμε να εκτελέσουμε όλες αυτές τις διεργασίες στον επιπλέον χρόνο $\epsilon I_{x'}$ που προστέθηκε. Βέβαια, υπάρχει περίπτωση το διάστημα $I_{x'}$ να δεσμεύεται πλήρως από μία μεγάλη διεργασία, έστω l, η οποία βάσει του Πορίσματος 3.1 διασχίζει συνολικά $s = \lceil \log_{1+\epsilon}(1+\frac{1}{\epsilon}) \rceil$ διαστήματα μέχρι να ολοκληρώσει την εκτέλεση της. Είναι εμφανές ότι στην περίπτωση αυτή ο επιπλέον χρόνος $\epsilon I_{x'}$ δεν ωφελεί. Μία λύση είναι να μεταφέρουμε όλες αυτές τις διεργασίες σε κάποιο διάστημα $I_{y'}$, ακριβώς πριν την έναρξη της εκτέλεσης της l και να τις εκτελέσουμε σ' αυτό. Η ενέργεια αυτή δεν αυξάνει τον χρόνο ολοκλήρωσης καμίας από τις διεργασίες. Επιπλέον, λόγω του Πορίσματος 3.1 το διάστημα $I_{y'}$ θα βρίσκεται το πολύ s διαστήματα πριν από το διάστημα $I_{x'}$, οπότε θα ισχύει ότι $I_{y'} \geq \epsilon I_{x'}$. Έτσι, αφού το μέγεθος κάθε διεργασίας j που μετατοπίζεται στο διάστημα $I_{x'}$ είναι το πολύ $\epsilon^2 I_{x'}$ θα είναι το πολύ ίσο με $\epsilon I_{y'}$ και επομένως η j θα είναι μικρή για το διάστημα $I_{y'}$ και θα "χωράει" να εκτελεσθεί σ' αυτό.

Για να προσδιορίσουμε το συνολικό κόστος της λύσης που κατασκευάσαμε, αρκεί να προσθέσουμε στο κόστος της μετατόπισης των μεγάλων διεργασιών το κόστος από την αύξηση στα μήκη των διαστημάτων. Για να βρούμε ένα άνω φράγμα στο κόστος της μετατόπισης των μεγάλων διεργασιών παρατηρούμε ότι κάθε μεγάλη διεργασία που μετατοπίζεται από ένα διάστημα I_x σε ένα διάστημα I_{x+k} ολοκληρώνει την εκτέλεση της το πολύ κατά τη χρονική στιγμή $R_{x+k+1} = (1+\epsilon)R_x/\epsilon^4$. Επιπλέον, υπάρχουν το πολύ $1/\epsilon$ τέτοιες διεργασίες σε κάθε διάστημα και το τελευταίο διάστημα στο οποίο έχουμε μετατοπίσει διεργασίες εκπνέει κατά τη χρονική στιγμή t. Προκύπτει έτσι, ότι

ο συνολικός χρόνος ολοκλήρωσης των μετατοπισμένων διεργασιών θα είναι ίσος με

$$\begin{aligned}\sum_{R_x<t} \frac{1}{\epsilon}\frac{(1+\epsilon)R_x}{\epsilon^4} &\leq \frac{t}{\epsilon^5}\sum_{i\geq 0}\frac{1}{(1+\epsilon)^i} \\ &= \frac{t}{\epsilon^5}\frac{(1+\epsilon)}{\epsilon} \\ &= \epsilon\,\mathrm{OPT}(1+\epsilon) \leq 2\epsilon\,\mathrm{OPT}\,. \end{aligned} \tag{3.5}$$

Αν στο αποτέλεσμα της (3.5) προσθέσουμε και το κόστος $(1+\epsilon)\,\mathrm{OPT}$, που οφείλεται στην αύξηση κατά παράγοντα $1+\epsilon$ στο μήκος κάθε διαστήματος, τότε έχουμε το ζητούμενο. □

Για να ολοκληρώσουμε το σχεδιασμό του PTAS συνδυάζουμε κατάλληλα τα δύο προηγούμενα λήμματα. Θεωρούμε ένα στιγμιότυπο εισόδου του προβλήματος, έστω I. Από το Λήμμα 3.6 μπορούμε να μετασχηματίσουμε το I έτσι ώστε σε μία υποτιθέμενη βέλτιστη χρονοδρομολόγηση οι μεγάλες διεργασίες να εμφανίζονται μετά τη χρονική στιγμή t. Με τη βοήθεια τώρα του Λήμματος 3.5 και με απώλεια $1+\epsilon$ επαναχρονοδρομολογούμε όλες τις μικρές διεργασίες του I, εφαρμόζοντας τον κανόνα SPT. Πιο αναλυτικά: προκαθορίζουμε αρχικά τη θέση των μεγάλων διεργασιών στους χρόνους στους οποίους αποδεσμεύονται και στη συνέχεια εκτελούμε μία preemptive χρονοδρομολόγηση των μικρών διεργασιών γύρω από από αυτές εφαρμόζοντας τον κανόνα SRPT. Λόγω του Λήμματος 3.6 οι μεγάλες διεργασίες εμφανίζονται μετά το t. Η χρονοδρομόλογηση αυτή θα είναι βέλτιστη (βλ. [7]) και επομένως με απώλεια $1+\epsilon$ μπορούμε να τη μετατρέψουμε σε non-preemptive (βλ. απόδειξη του Λήμματος 3.5). Έτσι, η συνολική απώλεια, μέχρι τη χρονική στιγμή t, μετά και την εφαρμογή του SPT θα είναι το πολύ $(1+\epsilon)(1+3\epsilon)$. Οπότε, μετά το t θα έχουν απομείνει να χρονοδρομολογηθούν το πολύ $(1+\epsilon)(1+3\epsilon)\frac{1}{\epsilon^7} \leq \frac{3}{\epsilon^7}$ διεργασίες, όπως ακριβώς ζητείται και στο Βήμα 1 του Αλγορίθμου 3.1. Για τις εναπομείνασες $\frac{3}{\epsilon^7}$ διεργασίες απαριθμούμε όλες τις εφικτές χρονοδρομολογήσεις τους και επιλέγουμε αυτή με τη μικρότερο κόστος. Αποδεικνύεται έτσι το ακόλουθο θεώρημα.

Θεώρημα 3.1 *Ο Αλγόριθμος 5 βρίσκει μία* $(1+\epsilon)$*-προσεγγιστική λύση στο πρόβλημα* $1\,|\,r_j\,|\,\sum_j C_j$ *σε χρόνο* $O(n\log n + (3/\epsilon^7)!)$.

Κλείνοντας, αναφέρουμε ότι με πιο πολύπλοκη ανάλυση μπορούμε να βελτιώσουμε το χρόνο του βήματος της απαρίθμησης σε $2^{o(1/\epsilon^3)}$ (βλ. Section 3 στην εργασία [2]).

3.3.3 Ελαχιστοποίηση του μέσου βεβαρημένου χρόνου ολοκλήρωσης σε παράλληλες πανομοιότυπες μηχανές

Η παραπάνω μέθοδος δεν φαίνεται να μπορεί να επεκταθεί ώστε να δώσει ένα *PTAS* στην περίπτωση που οι διεργασίες διαθέτουν και κάποιο θετικό ακέραιο βάρος. Ο κύριος λόγος είναι ότι η

παρουσία του βάρους κάνει πολύπλοκη την αλληλεπίδραση μεταξύ των μεγάλων και των μικρών διεργασιών κάθε διαστήματος, με αποτέλεσμα τόσο η εφαρμογή απλών κανόνων, όπως ο SPT και ο SRPT, όσο και η απλή απαρίθμηση, να μην διευκολύνει την αποδοτική χρονοδρομολόγηση των διεργασιών. Αν επιπλέον, το περιβάλλον μηχανής είναι αυτό των παράλληλων πανομοιότυπων μηχανών τότε η πολυπλοκότητα του προβλήματος γίνεται ακόμα μεγαλύτερη. Μία άμεση παρατήρηση είναι ότι αντί του κανόνα SPT (ή SRPT) μπορούμε να χρησιμοποιήσουμε τον κανόνα του Smith για τη χρονοδρομολόγηση διεργασιών που αποδεσμεύονται στο ίδιο διάστημα. Παρόλα αυτά θα χρειαστεί να εφαρμόσουμε συμπαγείς μεθόδους απαρίθμησης για να οδηγηθούμε στο ζητούμενο αποτέλεσμα.

Στην συνέχεια παρουσιάζουμε δύο προσεγγιστικά σχήματα πολυωνυμικού χρόνου για δύο διαφορετικές εκδοχές του προβλήματος ελαχιστοποίησης του μέσου βεβαρημένου χρόνου ολοκλήρωσης σε παράλληλες πανομοιότυπες μηχανές. Το πρώτο αφορά την non-preemptive εκδοχή του προβλήματος, $P\,|\,r_j|\,\sum_j w_jC_j$ και το δεύτερο την αντίστοιχη preemptive εκδοχή, $P\,|\,r_j,\ pmtn|\,\sum_j w_jC_j$. Τα αποτελέσματα αυτά οφείλονται στους Chekuri και Khanna και περιγράφονται αναλυτικά στις εργασίες [2] και [13].

Οι βασικές συνιστώσες για το σχεδιασμό των αποτελεσμάτων αυτών είναι η κατάλληλη εφαρμογή των μετασχηματισμών εισόδου που περιγράψαμε στην Ενότητα 3.3.1, σε συνδυασμό με την εφαρμογή δυναμικού προγραμματισμού. Συγκεκριμένα, εφαρμόζοντας μετασχηματισμούς στρογγυλοποίησης και μετατόπισης κατασκευάζουμε, με μικρή απώλεια, ένα υποσύνολο εφικτών χρονοδρομολογήσεων των διεργασιών και με χρήση δυναμικού προγραμματισμού αναζητούμε τη βέλτιστη χρόνοδρομολόγηση στο υποσύνολο αυτό.

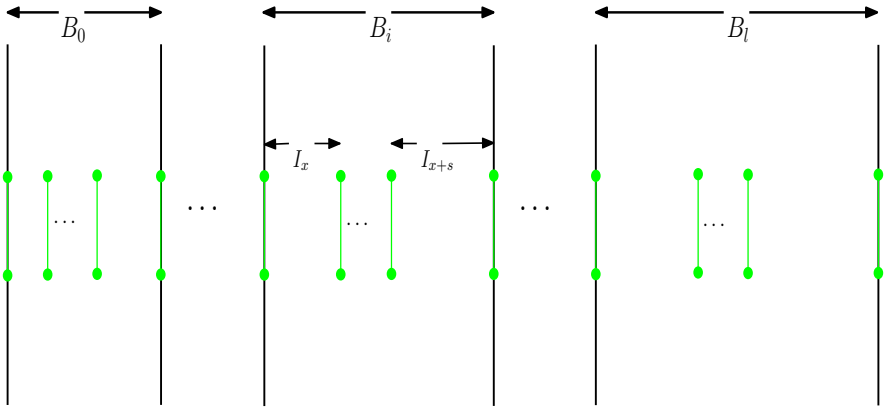
Θα επικεντρώσουμε το ενδιαφέρον μας κυρίως στο σχεδιασμό ενός PTAS για την non-preemptive εκδοχή του προβλήματος, $P\,|\,r_j|\,\sum_j w_jC_j$. Όπως θα δούμε, η εφαρμογή δυναμικού προγραμματισμού είναι αρκετά πιο περίπλοκη σ' αυτή την περίπτωση. Συγκεκριμένα, στις δύο υποενότητες που ακολουθούν περιγράφουμε αναλυτικά ένα PTAS για το πρόβλημα $P\,|\,r_j|\,\sum_j w_jC_j$ και στην τελευταία υποενότητα αναφέρουμε τις τροποποιήσεις που χρειάζονται για την preemptive εκδοχή του προβλήματος.

3.3.3.1 Δυναμικός προγραμματισμός

Η βασική ιδέα για την εφαρμογή δυναμικού προγραμματισμού είναι να διαμερίσουμε τον χρονικό ορίζοντα σε μία ακολουθία τμημάτων. Κάθε *τμήμα* θα αποτελείται από $s = \lceil \log_{1+\epsilon}(1+\frac{1}{\epsilon})\rceil \leq \frac{1}{\epsilon^2}$ [2] διαδοχικά διαστήματα της μορφής I_x, για κάποιο θετικό ακέραιο x (βλ. Σχήμα 3.2), ενώ ως χρονικό ορίζοντα θα θεωρήσουμε το διάστημα $[\min_j r_j, D)$, όπου η ποσότητα D είναι ένα άνω

[2] η δεύτερη ανισότητα ισχύει λόγω της αρχικής μας υπόθεσης ότι $\epsilon \leq 1/4$.

φράγμα στο makespan της χρονοδρομολόγησης των διεργασιών, π.χ $D = \sum_j p_j + \max_j r_j$.



Σχήμα 3.2: Διαμέριση του χρονικού ορίζοντα σε τμήματα, με σταθερό αριθμό διαστημάτων ανάλογο του ϵ.

Θεωρούμε, κατά συνέπεια, μία διαμέριση του χρονικού ορίζοντα στα τμήματα $B_0, B_1, \ldots, B_l$, όπου $l = O(\log D)$. Από το Πόρισμα 3.1 προκύπτει ότι καμία διεργασία δεν μπορεί να διασχίζει εξ ολοκλήρου ένα τμήμα. Με άλλα λόγια, αν μία διεργασία αρχίζει να εκτελείται στο τμήμα B_i, τότε αυτή θα ολοκληρώσει την εκτέλεση της είτε στο ίδιο τμήμα, είτε στο τμήμα B_{i+1}. Οι διεργασίες που αρχίζουν την εκτέλεση τους εντός ενός τμήματος και ολοκληρώνονται στο επόμενο τμήμα καλούνται *crossing διεργασίες*.

Ορίζουμε στη συνέχεια ένα βοηθητικό διάνυσμα, το οποίο θα περιγράφει τους διαφορετικούς τρόπους με τους οποίους οι crossing διεργασίες ενός τμήματος μπορούν να ολοκληρώσουν την εκτέλεση τους στο επόμενο τμήμα. Το διάνυσμα αυτό καλείται *σύνορο (frontier)* και μπορεί να είναι είτε εισερχόμενο είτε εξερχόμενο για ένα τμήμα B_i. Συγκεκριμένα ένα *εισερχόμενο σύνορο* ενός τμήματος B_i καθορίζει για κάθε μηχανή τη χρονική στιγμή στην οποία ολοκληρώνονται σ' αυτή οι crossing διεργασίες από το τμήμα B_{i-1}. Αντίστοιχα ορίζεται και το *εξερχόμενο σύνορο* (βλ. Σχήμα 3.3). Το ακόλουθο λήμμα διευκρινίζει το ρόλο των συνόρων στην εφαρμογή του δυναμικού προγραμματισμού.

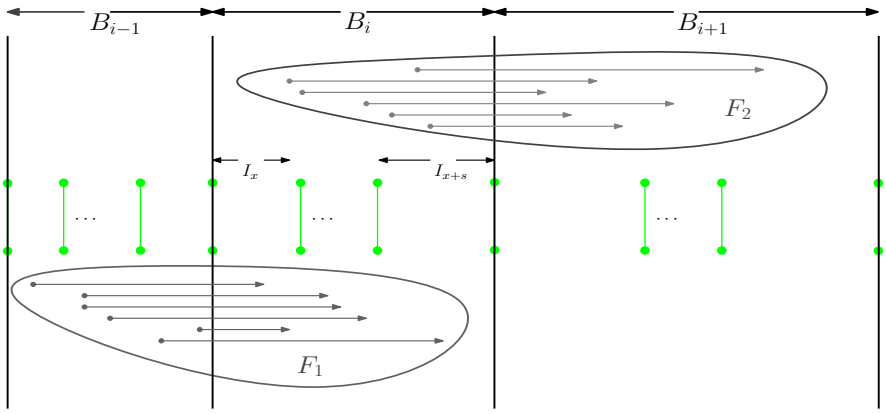
Λήμμα 3.7 *Υπάρχει μία* $(1+\epsilon)$*-προσεγγιστική χρονοδρομολόγηση η οποία θεωρεί μόνο* $(m+1)^{s/\epsilon}$ *σύνορα μεταξύ δύο οποιωνδήποτε διαδοχικών τμημάτων.*

Απόδειξη. Η ιδέα της απόδειξης είναι να διακριτοποιήσουμε τις χρονικές στιγμές όπου οι crossing διεργασίες ολοκληρώνουν την εκτέλεση τους.

Θεωρούμε αρχικά μία βέλτιστη χρονοδρομολόγηση των διεργασιών. Από το Λήμμα 3.3 θεωρούμε ότι οι μικρές διεργασίες ολοκληρώνονται εντός του διαστήματος στο οποίο αρχίζουν να εκτελούνται. Μία crossing διεργασία j που ξεκινά να εκτελείται σε ένα τμήμα B_{i-1} θα ολοκληρώ-

σει την εκτέλεση της σε κάποιο από τα s διαστήματα του τμήματος B_i, έστω στο διάστημα I_{x_j}. Στρογγυλοποιούμε προς τα πάνω το χρόνο ολοκλήρωσης της j σε C'_j, όπου $C'_j = R_{x_j} + i \cdot \epsilon I_{x_j}$, για ένα ακέραιο $0 < i \leq \frac{1}{\epsilon} - 1$. Το $C'_j \leq (1+\epsilon)R_{x_j} = R_{x_j+1}$, οπότε η j ολοκληρώνεται εντός του I_x, ενώ ο χρόνος ολοκλήρωσης κάθε crossing διεργασίας j αυξάνεται το πολύ κατά ένα παράγοντα $1+\epsilon$, το ίδιο και η τιμή της αντικειμενικής συνάρτησης. Παρατηρούμε ότι με τη στρογγυλοποίηση αυτή περιορίζουμε τις πιθανές χρονικές στιγμές ολοκλήρωσης (των crossing διεργασιών), σε το πολύ $\frac{1}{\epsilon}$ διακριτές τιμές για κάθε διάστημα. Επομένως, σε ένα ολόκληρο τμήμα, το οποίο αποτελείται από s διαστήματα, θα υπάρχουν το πολύ $\frac{s}{\epsilon}$ τέτοιες χρονικές στιγμές.

Κατά τη χρονοδρομολόγηση των crossing διεργασιών κάθε μηχανή μπορεί να περιέχει το πολύ $\frac{s}{\epsilon}$ δυνατούς χρόνους ολοκλήρωσης γι' αυτές. Έτσι, ως σύνορο μπορούμε να θεωρήσουμε ένα διάνυσμα της μορφής $(m_1, \ldots, m_{s/\epsilon})$ με διάσταση $\frac{s}{\epsilon}$. Με m_i συμβολίζουμε το πλήθος των μηχανών στις οποίες ολοκληρώνεται η εκτέλεση των crossing διεργασιών κατά τη χρονική στιγμή i. Συνεπώς μπορεί να υπάρχουν το πολύ $(m+1)^{s/\epsilon}$ διαφορετικά σύνορα μεταξύ δύο διαδοχικών τμημάτων. Το $m+1$ οφείλεται στο ότι τη χρονική στιγμή i μπορεί να μην υπάρχει καμία μηχανή στην οποία οι διεργασίες να ολοκληρώνουν την εκτέλεσή τους. □



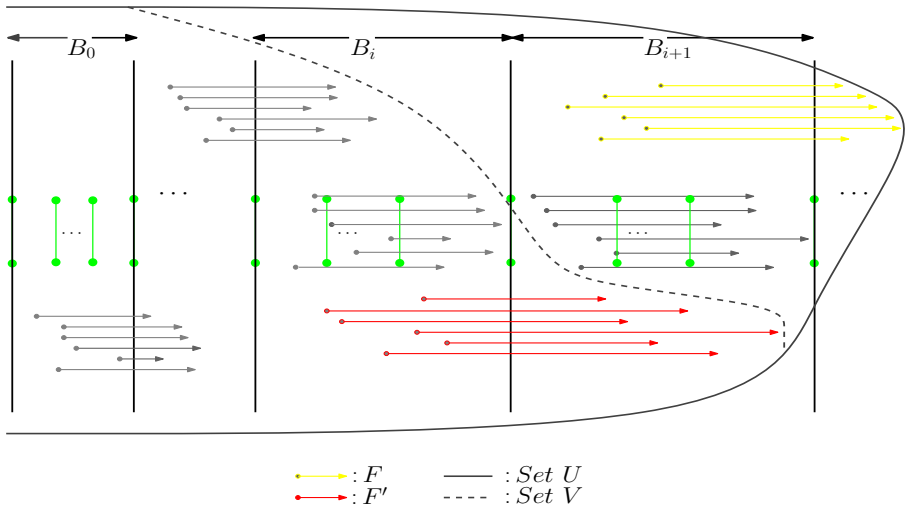
Σχήμα 3.3: Ένα εισερχόμενο σύνορο F_1 και ένα εξερχόμενο σύνορο F_2 για ένα τμήμα B_i.

Για την υλοποίηση του δυναμικού προγράμματος ορίζουμε έναν πίνακα εισόδου $A(\mathcal{L}, \mathcal{F}, \mathcal{U})$. Το $\mathcal{L} = O(\log D)$ είναι ένας θετικός ακέραιος που αντιστοιχεί στον συνολικό αριθμό των τμημάτων, το $\mathcal{F}$ είναι το σύνολο όλων των δυνατών συνόρων μεταξύ των τμημάτων και το $\mathcal{U}$ είναι το σύνολο όλων των ενδεχόμενων υποσυνόλων διεργασιών που έχουν αρχίσει να εκτελούνται πριν το τέλος κάθε τμήματος. Για μία θέση $A(i, F, U)$ του πίνακα, το $i \leq \mathcal{L}$ καθορίζει το τρέχον τμήμα B_i στο οποίο εργαζόμαστε, το $F \in \mathcal{F}$ είναι ένα εξερχόμενο σύνορο του B_i προς το τμήμα B_{i+1} και το $U \in \mathcal{U}$ είναι ένα ενδεχόμενο υποσύνολο διεργασιών που έχουν αποδεσμευτεί και έχουν αρχίσει να εκτελούνται πριν το τέλος του B_i. Στην $A(i, F, U)$ αποθηκεύουμε τον ελάχιστο μέσο

βεβαρημένο χρόνο ολοκλήρωσης που επιτυγχάνεται κατά τη χρονοδρομολόγηση των διεργασιών του U, έχοντας δεδομένο το εξερχόμενο σύνορο F. Για οποιαδήποτε i, F_1, F_2, K, η ποσότητα $W(i, F_1, F_2, K)$ αντιστοιχεί στον ελάχιστο μέσο βεβαρημένο χρόνο ολοκλήρωσης που επιτυγχάνεται από τη χρονοδρομολόγηση των διεργασιών του συνόλου K στο τμήμα B_i, έχοντας δεδομένο ένα εισερχόμενο σύνορο F_1 από το τμήμα B_{i-1} και ένα εξερχόμενο σύνορο F_2 προς το τμήμα B_{i+1}. Το K είναι ένα ενδεχόμενο υποσύνολο διεργασιών οι οποίες αρχίζουν την εκτέλεση τους εντός του τμήματος B_i. Αν θεωρήσουμε γνωστές τις τιμές του A για κάποιο i, τότε οι τιμές του για το $i+1$ υπολογίζονται από την ακόλουθη αναδρομή.

$$A(i+1, F, U) = \min_{F' \in \mathcal{F}, V \subset U} \{A(i, F', V) + W(i+1, F', F, U - V)\}. \tag{3.6}$$

Το U είναι ένα ενδεχόμενο υποσύνολο διεργασιών που αρχίζουν να εκτελούνται πρίν το τέλος του B_{i+1} και το $V \subset U$ είναι ένα ενδεχόμενο υποσύνολο διεργασιών που αρχίζουν την εκτέλεση τους πριν το τέλος του B_i. Επομένως το $U - V$ θα περιέχει εκείνες τις διεργασίες του U, οι οποίες αρχίζουν την εκτέλεση τους εντός του τμήματος B_{i+1} (βλ. Σχήμα 3.4). Για παράδειγμα, στο πρώτο βήμα της αναδρομής όπου $i = 0$, η ποσότητα $A(0, F, U)$, για ένα εξερχόμενο σύνορο F προς το τμήμα B_1 και για ένα ενδεχόμενο υποσύνολο διεργασιών U, που αρχίζουν να εκτελούνται πριν το τέλος του B_0 έχουμε ότι $A(0, F, U) = W(0, \emptyset, F, U)$. Το κενό σύνολο υποδηλώνει ότι δεν υπάρχει εισερχόμενο σύνορο για το τμήμα B_0.



Σχήμα 3.4: Απεικόνιση των ενδεχόμενων υποσυνόλων διεργασιών στο Βήμα $i+1$ του δυναμικού προγράμματος (3.6).

Υπάρχουν δύο σημαντικές δυσκολίες στην υλοποίηση του δυναμικού προγραμματισμού. Κατά πρώτον, δεν μπορούμε σε πολυωνυμικό χρόνο να διατηρούμε όλα τα ενδεχόμενα υποσύνολα διερ-

γασιών. Ο λόγος είναι απλός καθώς αυτά μπορεί να είναι αυθαίρετα πολλά για κάθε τμήμα που εξετάζουμε. Θα χρειαστεί έτσι να δείξουμε ότι υπάρχει μία προσεγγιστική χρονοδρομολόγηση των διεργασιών κατά την οποία, με μικρή απώλεια, η αναπαράσταση όλων αυτών των υποσυνόλων γίνεται με συμπαγή τρόπο. Κατά δεύτερον, θα πρέπει να βρούμε μία διαδικασία η οποία να υπολογίζει με αποδοτικό τρόπο την ποσότητα $W(i, F_1, F_2, V)$. Στην επόμενη υποενότητα περιγράφουμε αναλυτικά τις διαδικασίες αυτές.

Το ακόλουθο λήμμα είναι ιδιαίτερα σημαντικό καθώς βοηθάει στην αποδοτική διαχείρηση κατά την εκτέλεση του δυναμικού προγράμματος. Συγκεκριμένα, μας δίνει ένα άνω φράγμα για το χρόνο όπου μία διεργασία μπορεί να παραμένει ανεκτέλεστη μετά την αποδέσμευσή της.

Λήμμα 3.8 *Υπάρχει μία* $(1 + O(\epsilon))$*-προσεγγιστική χρονοδρομολόγηση των διεργασιών κατά την οποία η κάθε διεργασία ολοκληρώνει την εκτέλεση της μετά το πέρας το πολύ* $O(\frac{1}{\epsilon^2})$ *τμημάτων από τη στιγμή που αποδεσμεύθηκε.*

Απόδειξη. Η ιδέα είναι να συλλέξουμε όλες τις διεργασίες που "περισσεύουν" μετά το πέρας ενός τμήματος και να βρούμε ένα κατάλληλο επόμενο τμήμα στο οποίο να μπορούμε να τις εκτελέσουμε με μικρή απώλεια.

Θεωρούμε μία βέλτιστη χρονοδρομολόγηση και έστω A_i το σύνολο των διεργασιών που αποδεσμεύονται στο τμήμα B_i και που δεν έχουν εκτελεσθεί μέχρι και το τμήμα B_{l_i}, όπου $l_i = i + \frac{1}{\epsilon^2}$. Παραλλάσσουμε τη βέλτιστη χρονοδρομολόγηση και δρομολογούμε τις διεργασίες του A_i στο μικρότερο (πρώτο κατά σειρά) διάστημα του B_{l_i}, έστω I_x. Εφαρμόζοντας το Λήμμα 3.4 προκύπτει ότι ο συνολικός χρόνος επεξεργασίας των μικρών και μεγάλων αντίστοιχα διεργασιών που αποδεσμεύονται στο τμήμα B_i θα είναι

$$p\Big(\sum_{y:\, I_y \in B_i} T_y\Big) \leq s(1+\epsilon)m \max_{I_y \in B_i} I_y, \tag{3.7}$$

$$p\Big(\sum_{y:\, I_y \in B_i} H_y\Big) \leq \Big(s\frac{m}{\epsilon^2}\log_{1+\epsilon}\Big(\frac{1}{\epsilon}\Big)^4\Big)\Big(s \max_{I_y \in B_i} I_y\Big). \tag{3.8}$$

Αθροίζοντας τις (3.7) και (3.8) έχουμε

$$p\Big(\sum_{y:\, I_y \in B_i} T_y\Big) + p\Big(\sum_{y:\, I_y \in B_i} H_y\Big) \leq O(\epsilon)mI_x. \tag{3.9}$$

Για την (3.7) χρησιμοποιήσαμε τη Συνθήκη 1 του Λήμματος 3.4 στα s διαστήματα του τμήματος B_i. Για την (3.8) το πρώτο γινόμενο προέκυψε με εφαρμογή των Συνθηκών 2, 3 του Λήμματος 3.4 και αντιστοιχεί στο πλήθος των μεγάλων διεργασιών που αποδεσμεύονται στα s διαστήματα του B_i, ενώ το δεύτερο γινόμενο $(s \max_{I_y \in B_i} I_y)$ είναι ένα άνω φράγμα στο χρόνο επεξεργασίας κάθε

διεργασίας και προκύπτει μέσω της Πρότασης 3.1. Το $\max_{I_y \in B_i} I_y$ είναι το τελευταίο διάστημα του B_i και το μήκος του αποτελεί ένα άνω φράγμα για το μήκος των διαστημάτων του B_i. Η (3.9) προκύπτει αν λάβουμε υπόψην ότι το $s \leq 1/\epsilon^2$ και το $\epsilon \leq 1/4$ και κυρίως ότι το διάστημα $\max_{I_y \in B_i} I_y$ απέχει από το I_x ακριβώς $1/\epsilon^2$ τμήματα, βάση υπόθεσης, ή αλλιώς s/ϵ^2 διαστήματα, δηλαδή $x = x' + s/\epsilon^2$, όπου $I_{x'} = \max_{I_y \in B_i} I_y$. Έτσι έχουμε ότι $\max_{I_y \in B_i} I_y = I_{x-(s/\epsilon^2)} = \epsilon R_{x-(s/\epsilon^2)} = \epsilon \frac{(1+\epsilon)^x}{(1+\epsilon)^{s/\epsilon^2}} \leq \frac{(1+\epsilon)^x}{s/\epsilon^2} \leq \epsilon^5 I_x$. Επιπλέον το $\log_{1+\epsilon}(\frac{1}{\epsilon})^4 = 4\log_{1+\epsilon}\frac{1}{\epsilon} \leq 4s \leq (1/\epsilon)s \leq 1/\epsilon^3$. Οπότε για τη (3.7) έχουμε ότι $s(1+\epsilon)m \max_{I_y \in B_i} I_y \leq s(1+\epsilon)m\epsilon^5 I_x \leq s(1/\epsilon)m\epsilon^5 I_x \leq \epsilon^2 m I_x$ και για την (3.8) ότι, $(s\frac{m}{\epsilon^2}\log_{1+\epsilon}(\frac{1}{\epsilon})^4)(s\max_{I_y \in B_i} I_y) \leq (1/\epsilon^7)m(s\max_{I_y \in B_i} I_y) \leq (1/\epsilon^4)mI_x$. Αντικαθιστώντας τις αντίστοιχες τιμές στην (3.9) προκύπτει ότι $p(A_i) \leq (\epsilon^2 + 1/\epsilon^4)mI_x = O(\epsilon)mI_x$. Επιπλέον για κάθε διεργασία $j \in A_i$ θα ισχύει ότι $p_j \leq \epsilon^4 I_x$· όπως και πριν $p_j \leq s\max_{I_y \in B_i} I_y$ και $x = x' + s/\epsilon^2$, όπου $I_{x'} = \max_{I_y \in B_i} I_y$.

Χωρίζουμε το A_i σε ομάδες διεργασιών με κάθε ομάδα να έχει συνολικό χρόνο επεξεργασίας από ϵI_x έως $(\epsilon + \epsilon^4)I_x$. Λόγω της σχέσης $p(A_i) \leq O(\epsilon)mI_x$ θα υπάρχουν το πολύ m τέτοιες ομάδες. Αναθέτουμε κάθε ομάδα σε μία διαφορετική μηχανή M και την εκτελούμε εκεί αφού πρώτα περιμένουμε να ολοκληρώσει την εκτέλεση της n crossing διεργασία από το τμήμα $B_{l_{i-1}}$ (αν υπάρχει αυτή). Οι χρόνοι ολοκλήρωσης των διεργασιών του A_i, λόγω της πρόωρης χρονοδρομολόγησής τους θα είναι μικρότεροι ή ίσοι σε σχέση με τη βέλτιστη χρονοδρομολόγηση. Παρόλα αυτά, θα χρειαστεί να μετατοπίσουμε χρονικά κάποιες από τις διεργασίες που εκτελούνταν στη M κατά τη βέλτιστη χρονοδρομολόγηση. Αυξάνοντας το μήκος κάθε διαστήματος κατά ένα παράγοντα $1 + O(\epsilon)$, μπορούμε να διαχειριστούμε τη μετατόπιση αυτή με μικρή απώλεια. Η νέα χρονοδρομολόγηση θα ικανοποιεί την ιδιότητα του λήμματος. □

Στο σημείο αυτό, διαθέτουμε όλα όσα χρειαζόμαστε για να σκιαγραφήσουμε το σχεδιασμό ενός PTAS για το πρόβλημα $1\,|\,r_j\,|\,\sum_j w_j C_j$.

Από το Λήμμα 3.4 γνωρίζουμε ότι ο συνολικός χρόνος επεξεργασίας των διεργασιών που αποδεσμεύονται σε μία χρονική στιγμή R_x είναι το πολύ $O(I_x)$ (όπου $m = 1$). Βάσει τώρα του Λήμματος 3.8, ισχυριζόμαστε ότι όλες αυτές οι διεργασίες θα έχουν ολοκληρώσει την εκτέλεση τους μέσα στα επόμενα $O(s/\epsilon^2)$ διαστήματα. Αυτό μας δίνει τη δυνατότητα να διατηρούμε μία λίστα με τις μεγάλες διεργασίες που απομένουν να εκτελεστούν καθώς κινούμαστε από ένα τμήμα στο επόμενο· από τις διεργασίες που αποδεσμεύονται μέχρι το τέλος ενός τμήματος, αφαιρούμε εκείνες που ολοκληρώνουν την εκτέλεσή τους εντός αυτού.

Για τις μικρές διεργασίες που καταφθάνουν σε μία χρονική στιγμή R_x εκμεταλλευόμαστε το γεγονός ότι αυτές είναι διατεταγμένες βάσει του κανόνα του Smith. Διαμερίζουμε έτσι τη διάταξη αυτή σε $O(1/\epsilon^2)$ ομάδες διεργασιών, με περίπου ίδιο μέγεθος n κάθε ομάδα (τουλάχιστον $\epsilon^2 I_x$) και δείχνουμε ότι με μικρή αύξηση στο μήκος κάθε διαστήματος, μπορούμε να εκτελέσουμε ένα

ακέραιο μέρος από κάθε ομάδα σε κάθε τμήμα. Οι λεπτομέρειες της απόδειξης αυτής διαφαίνονται στην απόδειξη του Λήμματος 3.9, το οποίο αφορά στην γενική περίπτωση των παράλληλων πανομοιότυπων μηχανών. Μ' αυτό τον τρόπο διατηρούμε την απαραίτητη πληροφορία για τις μικρές διεργασίες. Για τον υπολογισμό της ποσότητας $W(i, F_1, F_2, V)$ δοκιμάζουμε όλους τους δυνατούς τρόπους με τους οποίους μπορούμε να χρονοδρομολογήσουμε τις μεγάλες διεργασίες του συνόλου V στο τμήμα B_i και σε κάθε περίπτωση χρονοδρομολογούμε τις μικρές διεργασίες του V γύρω από αυτές με βάση τον κανόνα του Smith. Για τις μεγάλες διεργασίες, αρκεί να παρατηρήσουμε ότι κάθε χρονική στιγμή R_x μπορούν να αρχίσουν να εκτελούνται το πολύ $1/\epsilon^2$ (βλ. Λήμμα 3.4). Επομένως, σε κάθε τμήμα θα αρχίζουν την εκτέλεσή τους το πολύ $O(s/\epsilon^2) = O(1/\epsilon^4)$ μεγάλες διεργασίες. Όπως θα δούμε στη συνέχεια, τα αντίστοιχα βήματα για την περίπτωση των παράλληλων πανομοιότυπων μηχανών απαιτούν επιπλέον ιδέες και ανάλυση για να υλοποιηθούν.

3.3.3.2 Non-preemptive χρονοδρομολόγηση

Στη συνέχεια περιγράφουμε λεπτομερώς πώς μπορούμε να υλοποιήσουμε αποδοτικά τις δύο βασικές συνιστώσες του δυναμικού προγραμματισμού: *i*) συμπαγής αναπαράσταση των υποσυνόλων διεργασιών και *ii*) αποδοτική χρονοδρομολόγηση των διεργασιών σε κάθε τμήμα. Όσα ακολουθούν αφορούν στην non-preemptive εκδοχή του προβλήματος.

Α. Συμπαγής αναπαράσταση των υποσυνόλων διεργασιών. Θυμίζουμε ότι με H_x και T_x συμβολίζουμε τις μεγάλες και μικρές αντίστοιχα διεργασίες που αποδεσμεύονται τη χρονική στιγμή R_x. Με $\mathcal{A}_{x_i}$ και $\mathcal{B}_{x_i}$ συμβολίζουμε τα σύνολα των μεγάλων και μικρών διεργασιών που αποδεσμεύονται τη στιγμή R_x και εκτελούνται στο τμήμα B_i. Για τις μεγάλες και τις μικρές διεργασίες που επιβιώνουν μετά το πέρας του τμήματος B_i, ορίζουμε τα σύνολα $\mathcal{U}_{x_i}$ και $\mathcal{V}_{x_i}$ αντίστοιχα. Στόχος μας είναι να δείξουμε ότι υπάρχουν $(1+\epsilon)$-προσεγγιστικές χρονοδρομολογήσεις των διεργασιών στις οποίες η αναπαράσταση των παραπάνω συνόλων γίνεται με συμπαγή τρόπο.

Θεωρούμε τις μεγάλες διεργασίες που αποδεσμεύονται σε ένα διάστημα I_x. Από το Λήμμα 3.4 έχουμε ότι υπάρχουν το πολύ $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ διαφορετικές τάξεις μεγεθών μεγάλων διεργασιών στο σύνολο H_x. Υποθέτουμε ότι για κάθε τάξη μεγέθους, οι διεργασίες ταξινομούνται σε αύξουσα διάταξη ως προς το βάρος τους. Τα σύνολα $\mathcal{U}_{x_i}$ και $\mathcal{A}_{x_i}$, για κάθε τμήμα B_i, καθορίζονται ως εξής: για κάθε τάξη μεγέθους του H_x θεωρούμε την πρώτη στη διάταξη διεργασία η οποία δεν εκτελείται εντός των ορίων του τμήματος B_i. Οι διεργασίες που προηγούνται αυτής στη διάταξη τοποθετούνται στο σύνολο $\mathcal{A}_{x_i}$. Η ίδια μαζί με τις διεργασίες που έπονται στη διάταξη, τοποθετούνται στο σύνολο $\mathcal{U}_{x_i}$. Βάσει του Λήμματος 3.4, προκύπτει ότι υπάρχουν το πολύ $(m/\epsilon^2)^{O(\log(1/\epsilon)/\epsilon)}$ δυνατές επιλογές για το σύνολο $\mathcal{U}_{x_i}$. Επιπλέον, πριν το τέλος του B_i θα υπάρχουν το πολύ $O(s/\epsilon^2)$ χρονικές στιγμές για τις οποίες οι μεγάλες διεργασίες που αποδεσμεύονται

παραμένουν ανεκτέλεστες (βλ. Λήμμα 3.8). Συνολικά θα υπάρχουν το πολύ $(m/\epsilon^2)^{O(\log(1/\epsilon)/\epsilon)(s/\epsilon^2)}$ ενδεχόμενα σύνολα μεγάλων διεργασιών, οι οποίες επιβιώνουν μετά το πέρας ενός τμήματος B_i.

Σε αντίθεση με τις μεγάλες, οι μικρές διεργασίες που επιβιώνουν μετά από το πέρας ενός τμήματος μπορεί να είναι αυθαίρετες στο πλήθος. Έτσι, επιλέγουμε να διατηρούμε πληροφορία για το συνολικό χρόνο επεξεργασίας (ή συνολικό μέγεθος) αυτών αντί για το πλήθος τους. Θυμίζουμε ότι έχουμε ήδη υποθέσει πως σε κάθε χρονική στιγμή αποδέσμευσης R_x οι διεργασίες του συνόλου T_x διατάσσονται βάσει του κανόνα του Smith (βλ. Λήμμα 3.3). Θυμίζουμε επίσης ότι κάθε μικρή διεργασία j σε ένα διάστημα I_x, θα έχει μέγεθος $p_j \leq \epsilon^2 I_x$.

Συμβολίζουμε με $a = \{1, 2, \ldots, z\}$ τη διάταξη των διεργασιών ενός συνόλου T_x και έστω σ_1 ο μικρότερος ακέραιος τέτοιος που ο συνολικός χρόνος επεξεργασίας των πρώτων σ_1 διεργασιών της a να υπερβαίνει το $\epsilon^2 I_x$. Συμβολίζουμε με $\Sigma_1(x)$ το σύνολο αυτών των διεργασιών. Έστω τώρα σ_2 ο μικρότερος ακέραιος τέτοιος που ο συνολικός χρόνος επεξεργασίας των επόμενων $\sigma_2 - \sigma_1$ διεργασιών της a υπερβαίνει το $\epsilon^2 I_x$ και $\Sigma_2(x)$ το αντίστοιχο σύνολο. Με τον τρόπο αυτό, κατασκευάζουμε μία ακολουθία συνόλων $\Sigma_1(x), \Sigma_2(x), \ldots, \Sigma_l(x)$, όπου το κάθε σύνολο $\Sigma_i(x)$, $1 \leq i \leq l$, περιέχει διεργασίες συνολικού μεγέθους τουλάχιστον $\epsilon^2 I_x$ και το πολύ $2\epsilon^2 I_x$. Από το Λήμμα 3.4 προκύπτει ότι $l \leq 2m$. Παρατηρούμε ότι,

Λήμμα 3.9 *υπάρχει μία* $(1+O(\epsilon))$*-προσεγγιστική χρονοδρομολόγηση των διεργασιών τέτοια ώστε, για κάθε χρόνο αποδέσμευσης* R_x*, όλες οι διεργασίες ενός συνόλου* $\Sigma_i(x)$ *εκτελούνται στο ίδιο διάστημα.*

Απόδειξη. Το Λήμμα 3.3 μας εξασφαλίζει την ύπαρξη μίας $(1+\epsilon)$-προσεγγιστικής χρονοδρομολόγησης κατά την οποία κάθε μικρή διεργασία που καταφθάνει σε μία χρονική στιγμή αποδέσμευσης, εκτελείται με βάση τον κανόνα του Smith. Έστω π μία τέτοια χρονοδρομολόγηση των διεργασιών. Θα δείξουμε ότι αν αυξήσουμε κάθε διάστημα κατά ένα παράγοντα $1+2\epsilon$, τότε η π ικανοποιεί τη συνθήκη του λήμματος. Η τιμή της αντικειμενικής συνάρτησης θα αυξηθεί επίσης κατά τον ίδιο παράγοντα.

Θεωρούμε ένα σύνολο $\Sigma_i(x)$ και έστω I_y το πρώτο διάστημα στο οποίο εκτελείται μία διεργασία του συνόλου. Έστω M η μηχανή στην οποία εκτελείται η διεργασία αυτή. Στην περίπτωση που κατά την π όλες οι διεργασίες του $\Sigma_i(x)$ εκτελούνται στο διάστημα I_y, τότε έχουμε το ζητούμενο. Διαφορετικά, γνωρίζουμε εκ των προτέρων ότι καμία διεργασία ενός συνόλου $\Sigma_{i'}(x)$, με $i' > i$, δεν μπορεί να εκτελείται στο διάστημα I_y και εκτελούμε όλες τις διεργασίες του $\Sigma_i(x)$ στη μηχανή M στο I_y. Ο επιπρόσθετος χρόνος επεξεργασίας στο φορτίο της M θα είναι το πολύ $2\epsilon^2 I_x$. Αθροίζοντας για όλα τα διαστήματα I_x, με $x < y$, μπορούμε να φράξουμε το συνολικό επιπρόσθετο χρόνο επεξεργασίας του φορτίου της M ώστε να είναι το πολύ $2\epsilon I_y$. Αυτό προκύπτει εύκολα με χρήση του Λήμματος 3.8: $\sum_{x<y} 2\epsilon^2 I_x \leq (s/\epsilon^2)(2\epsilon^2 I_x)$. Λόγω του ότι $s \leq 1/\epsilon^2$, η

ανισότητα γίνεται $\sum_{x<y} 2\epsilon^2 I_x \leq 2I_x/\epsilon^2 \leq 2\epsilon I_x(1+\epsilon)^{s/\epsilon^2}$. Όμως $2\epsilon I_x(1+\epsilon)^{s/\epsilon^2} = 2\epsilon^2 R_{x+s/\epsilon^2} = 2\epsilon I_y$. Αυξάνοντας τέλος τα μήκη των διαστημάτων κατά παράγοντα $1+2\epsilon$, μπορούμε να διαχειριστούμε τον επιπρόσθετο χρόνο επεξεργασίας και να ικανοποιήσουμε έτσι τη συνθήκη του λήμματος. □

Με τη βοήθεια του Λήμματος 3.9 μπορούμε τώρα να καθορίσουμε τα σύνολα διεργασιών $\mathcal{V}_{x_i}$ και $\mathcal{B}_{x_i}$, για κάθε τμήμα B_i. Για το σύνολο $\mathcal{B}_{x_i}$, αρκεί να προσδιορίσουμε τον μικρότερο ακέραιο i τέτοιο που οι διεργασίες του συνόλου $\Sigma_i(x)$ δεν έχουν ακόμη εκτελεσθεί. Οι δυνατές επιλογές για το σύνολο $\mathcal{V}_{x_i}$ θα είναι το πολύ m/ϵ^2 (βλ. Λήμμα 3.4). Επιπλέον, από το Λήμμα 3.8 θα υπάρχουν $O(s/\epsilon^2)$ πιθανοί χρόνοι αποδέσμευσης πριν το τέλος ενός τμήματος B_i, όπου οι διεργασίες που αποδεσμεύονται παραμένουν ανεκτέλεστες. Συνολικά θα υπάρχουν το πολύ $(m/\epsilon^2)^{O(s/\epsilon^2)}$ ενδεχόμενα σύνολα μικρών διεργασιών οι οποίες επιβιώνουν μετά το πέρας ενός τμήματος B_i.

Όλα τα παραπάνω συνοψίζονται στο ακόλουθο λήμμα.

Λήμμα 3.10 *Υπάρχει μία $(1+O(\epsilon))$-προσεγγιστική χρονοδρομολόγηση των διεργασιών τέτοια ώστε, για κάθε τμήμα B_i, να ισχύουν τα ακόλουθα:*

1. *Υπάρχουν $k = (\frac{m}{\epsilon^2})^{O(\frac{1}{\epsilon})}$ σύνολα διεργασιών, $G_i^1, G_i^2, \ldots, G_i^k$, τα οποία μπορούν να κατασκευαστούν σε πολυωνυμικό χρόνο.*

2. *Το G_i είναι το σύνολο των διεργασιών που αποδεσμεύονται πριν το πέρας του B_i ενώ δεν έχουν αρχίσει την εκτέλεσή τους και αντιστοιχεί σε ένα εκ των συνόλων $\{G_i^1, G_i^2, \ldots, G_i^k\}$.*

Β. Αποδοτική χρονοδρομολόγηση των διεργασιών στα τμήματα. Ο υπολογισμός της ποσότητας $W(i, F_1, F_2, V)$ για κάθε τμήμα B_i συνιστά από μόνος του ένα **NP**-hard πρόβλημα. Στη συνέχεια παρουσιάζουμε μία $(1+\epsilon)$-προσεγγιστική διαδικασία η οποία σε πολυωνυμικό χρόνο βρίσκει μία χρονοδρομολόγηση των διεργασιών στο εκάστοτε τμήμα B_i, τέτοια που ο μέσος βεβαρημένος χρόνος ολοκλήρωσης να είναι το πολύ $1+\epsilon$ επί του $W(i, F_1, F_2, V)$. Η διαδικασία αυτή, σε συνδυασμό με το δυναμικό πρόγραμμα (3.6), δίνει τελικά μία $(1+O(\epsilon))$-προσεγγιστική λύση στο πρόβλημα $P\,|\,r_j\,|\,\sum_j w_j C_j$.

Αρχικά, διαμερίζουμε τις διεργασίες του συνόλου V σε μεγαλύτερες και μικρότερες. Έστω I_x το πρώτο και μικρότερο επομένως διάστημα του τμήματος B_i. Μία διεργασία $j \in V$ καλείται *μεγαλύτερη* αν ισχύει ότι $p_j \geq \epsilon I_x$. Σε αντίθετη περίπτωση η j καλείται μικρότερη. Γνωρίζουμε ότι κάθε τμήμα B_i αποτελείται από s διαστήματα, οπότε προκύπτει ότι για μία μεγαλύτερη διεργασία $j \in V$ και για οποιοδήποτε διάστημα I_y του B_i θα ισχύει ότι $p_j \geq \epsilon^2 I_y$, δηλαδή ή j θα είναι μεγάλη

για κάθε διάστημα του B_i. Σημειώνουμε ότι δεν θα πρέπει να συγχέουμε τις μικρότερες και τις μεγαλύτερες διεργασίες με τις μικρές και τις μεγάλες διεργασίες των προηγούμενων ενοτήτων.

Κύριος στόχος μας είναι να απαριθμήσουμε με αποδοτικό τρόπο όλες τις ενδεχόμενες χρονοδρομολογήσεις των μεγαλύτερων διεργασιών. Για το σκοπό αυτό περιοριζόμαστε σε χρονοδρομολογήσεις κατά τις οποίες μία μεγαλύτερη διεργασία j αρχίζει να εκτελείται, για κάθε διάστημα I_x, σε μία χρονική στιγμή που δίνεται από τη σχέση $R_x + i \cdot \epsilon^3 I_x$, όπου $0 \leq i \leq \frac{1}{\epsilon^3} - 1$. Έτσι σε κάθε διάστημα I_x θα υπάρχουν το πολύ $1/\epsilon^3$ δυνατές διακριτές χρονικές στιγμές για κάθε μεγαλύτερη διεργασία. Σημαντικό ρόλο στην απαρίθμηση επιτελούν τα μεγέθη των μεγαλυτέρων διεργασιών. Ακολουθώντας την απόδειξη του Λήμματος 3.4 μπορούμε να δείξουμε ότι μέσα στα όρια ενός τμήματος θα υπάρχουν το πολύ $O(\log 1/\epsilon)$ διαφορετικά μεγέθη. Όπως και πριν, υποθέτουμε ότι οι μεγαλύτερες διεργασίες του ίδιου μεγέθους χρονοδρομολογούνται σε αύξουσα διάταξη ως προς το βάρος τους. Έτσι, για να καθορίσουμε απόλυτα τη χρονοδρομολόγηση των μεγαλύτερων διεργασιών σε μία μηχανή εντός ενός τμήματος, αρκεί να διαθέτουμε τα εξής τρία στοιχεία: *i*) ένα εισερχόμενο σύνορο F_1, *ii*) ένα εξερχόμενο σύνορο F_2 και *iii*) τα μεγέθη των διεργασιών που αρχίζουν να εκτελούνται στις διακριτές χρονικές στιγμές καθενός από τα s διαστήματα του τμήματος. Το συνολικό πλήθος των δυνατών διακριτών χρονικών στιγμών όπου μία μεγαλύτερη διεργασία μπορεί να αρχίσει να εκτελείται σε ένα τμήμα θα είναι το πολύ $s/\epsilon^3 \leq 1/\epsilon^5$. Επιπλέον, υπαρχουν το πολύ $O(\log 1/\epsilon)$ διαφορετικά μεγέθη μεγαλύτερων διεργασιών. Έτσι, για κάθε μηχανή θα υπάρχουν το πολύ $(O(\log 1/\epsilon))^{1/\epsilon^5} \leq 2^{O(1/\epsilon^6)}$ δυνατοί τρόποι για να χρονοδρομολογήσουμε σ' αυτή τις μεγαλύτερες διεργασίες ενός τμήματος. Για τις m μηχανές θα έχουμε συνολικά το πολύ $(m+1)^k$ πιθανές χρονοδρομολογήσεις, όπου $k = 2^{O(1/\epsilon^6)}$. Από αυτές, εφικτές είναι μόνο όσες συμφωνούν με τα εισερχόμενα και εξερχόμενα σύνορα F_1 και F_2. Κρατάμε επομένως αυτές και βρίσκουμε ανάμεσά τους τη βέλτιστη δυνατή χρονοδρομολόγηση για τις μεγαλύτερες διεργασίες του συνόλου V.

Οι μικρότερες διεργασίες του V θα χρονοδρομολογηθούν με άπλησστο τρόπο στα κενά που αφήνουν οι μεγαλύτερες διεργασίες: υποθέτουμε αρχικά ότι για κάθε διάστημα I_x οι μεγάλες διεργασίες που αρχίζουν και τελειώνουν σ' αυτό, εκτελούνται διαδοχικά στη μικρότερη κάθε φορά διαθέσιμη χρονική στιγμή του διαστήματος. Αυξάνουμε κατά παράγοντα $1 + \epsilon$ όλα τα κενά που δημιουργούνται μεταξύ δύο διαδοχικών εκτελέσεων έτσι ώστε να "χωρέσουμε" σ' αυτά τις μικρότερες διεργασίες. Σε κάθε διάστημα I_x του τμήματος B_i διατάσσουμε ως προς το λόγο p_j/w_j όλες τις μικρότερες διεργασίες του V που έχουν αποδεσμευθεί και δεν έχουν ακόμη εκτελεσθεί. Αυτές χρονοδρομολογούνται βάσει αυτής της διάταξης στα κενά που αφήνουν οι μεγαλύτερες διεργασίες, μέχρι να καλυφθεί ολόκληρο το διάστημα I_x. Επαναλαμβάνουμε για το επόμενο διάστημα I_{x+1} κ.ο.κ. Παραλείποντας τις λεπτομέρειες, η διαδικασία αυτή μπορεί να υλοποιηθεί σε

πολυωνυμικό χρόνο ως προς το πλήθος των μικρότερων διεργασιών καθώς και σε πολυωνυμικό χρόνο ως προς το m (αν διαμερίσουμε τις μικρότερες διεργασίες σε ομάδες μεγέθους το πολύ ϵI_x). Επισημαίνουμε ότι η χρονοδρομολόγηση των μικρότερων διεργασιών θα πρέπει πραγματοποιείται για κάθε ενδεχόμενη χρονοδρομολόγηση των μεγαλύτερων διεργασιών. Από τα παραπάνω προκύπτει το ακόλουθο λήμμα.

Λήμμα 3.11 *Υπάρχει μία* $(1+\epsilon)$*-προσεγγιστική διαδικασία για τον υπολογισμό της ποσότητας* $W(i, F_1, F_2, V)$ *η οποία τρέχει σε χρόνο* $(m+k)^k$*, όπου* $k = 2^{O(1/\epsilon^6)}$.

Σημειώνουμε ότι αν αντί της απαρίθμησης των χρονοδρομολογήσεων εφαρμόσουμε δυναμικό προγραμματισμό μεταξύ των διαστημάτων του κάθε τμήματος, ο χρόνος εκτέλεσης της παραπάνω διαδικασίας μπορεί να βελτιωθεί σε $(m+1)^{poly(1/\epsilon)}$ (βλ. [2]).

Αν τώρα, δεδομένων των Λημμάτων 3.10 και 3.11, εφαρμόσουμε το δυναμικό πρόγραμμα (3.6), καταλήγουμε στο ακόλουθο κεντρικό θεώρημα.

Θεώρημα 3.2 *Υπάρχει ένα* PTAS *για το πρόβλημα* $P \mid r_j \mid \sum_j w_j C_j$ *το οποίο εκτελείται σε χρόνο* $O(n \log n + n(m+1)^{poly(1/\epsilon)})$ *και βρίσκει μία* $(1+\epsilon)$*-προσεγγιστική χρονοδρομολόγηση των διεργασιών.*

Έχουμε θεωρήσει ότι το πλήθος των τμημάτων θα είναι το πολύ $O(\log D)$, όπου D είναι ένα άνω φράγμα στο makespan της χρονοδρομολόγησης των διεργασιών (π.χ. $D = \sum_j p_j + \max_j r_j$). Επί της ουσίας όμως, τα τμήματα που μας ενδιαφέρουν κατά την εκτέλεση του δυναμικού προγραμματισμού θα είναι το πολύ $O(n/\epsilon^3)$: κάθε διεργασία j διασχίζει το πολύ $1/\epsilon^2$ διαστήματα από τη στιγμή που αρχίζει να εκτελείται (βλ. Πρόταση 3.1) άρα θα υπάρχουν το πολύ n/ϵ^2 τμήματα στα οποία μία διεργασία μπορεί να αρχίσει την εκτέλεση της. Αν συνυπολογίσουμε το ότι κάθε διεργασία ολοκληρώνεται σε $1/\epsilon^2$ τμήματα από τη στιγμή που αποδεσμεύεται (βλ. Λήμμα 3.8), τότε ο αριθμός των τμημάτων που μας ενδιαφέρουν θα είναι το πολύ $n/\epsilon^2 + 1/\epsilon^2 \leq 1/\epsilon^2(n+1) \leq (1/\epsilon^2)(n/\epsilon) = n/\epsilon^3$. Το γεγονός αυτό μειώνει ακόμα περισσότερο την χρονική πολυπλοκότητα του αλγορίθμου.

3.3.3.3 Preemptive χρονοδρομολόγηση

Στην συνέχεια αναφέρουμε συνοπτικά τις απλοποιήσεις που χρείαζονται ώστε να μετατρέψουμε το παραπάνω PTAS σε ένα προσεγγιστικό σχήμα για την preemptive εκδοχή του προβλήματος, $P \mid r_j,\ pmtn \mid \sum_j w_j C_j$.

Αρχικά παρατηρούμε ότι σε μία preemptive χρονοδρομολόγηση οι crossing διεργασίες μπορούν να διακόψουν την εκτέλεση τους σε ένα διάστημα, έχοντας πριν εκτελέσει μόνο ένα μικρό μέρος από το συνολικό χρόνο επεξεργασίας τους. Συνεπώς, η εκτέλεση κάθε crossing διεργασίας

μπορεί να γίνει σε περισσότερα του ενός βήματα, χωρίς την εξ' αρχής δέσμευση μίας ακολουθίας διαστημάτων (βλ. Πόρισμα 3.1), εκτελώντας στο τρέχον κάθε φορά διάστημα μόνο ένα μέρος του συνολικού χρόνου επεξεργασίας της. Το γεγονός αυτό καθιστά περιττή την πληροφορία που φέρει το κάθε σύνορο κατά την εκτέλεση του δυναμικού προγραμματισμού (βλ. Υποενότητα 3.3.3.1). Επιπλέον, χρησιμοποιώντας επιχειρήματα παρόμοια μ' αυτά που παρουσιάσαμε προηγουμένως (βλ. Παράγραφο Α. Συμπαγής αναπαράσταση των υποσυνόλων διεργασιών) μπορούμε να διατηρούμε με αποδοτικό τρόπο την πληροφορία για τις ποσότητες των μεγεθών των διεργασιών που εκτελούνται σε κάθε διάστημα. Έτσι, ο δυναμικός προγραμματισμός απλουστεύεται καθώς μπορεί να εφαρμοστεί μεταξύ των διαδοχικών διαστημάτων και όχι μεταξύ ολόκληρων τμημάτων.

Οι παραπάνω απλοποιήσεις επιφέρουν σημαντική μείωση στο αρχικό κόστος της απαρίθμησης. Αν για παράδειγμα λάβουμε υπόψη μας ότι οι χρονικές στιγμές αποδέσμευσης εμφανίζονται στα άκρα κάθε διαστήματος και όχι στο εσωτερικό του, τότε μπορούμε να εφαρμόσουμε τον κανόνα του McNaughton (wrap around rule) [69] και να υπολογίσουμε μία preemptive χρονοδρομολόγηση με βέλτιστο makespan, σε χρόνο $O(n)$. Αν επιπλέον γνωρίζουμε τις ποσότητες των μεγεθών των διεργασιών που εκτελούνται σε κάθε διάστημα, τότε μπορούμε να βρούμε μία αποδοτική χρονοδρομολόγηση γι' αυτές. Παραλείποντας τις τεχνικές λεπτομέρειες παραθέτουμε το κεντρικό αποτέλεσμα.

Θεώρημα 3.3 *Υπάρχει ένα* PTAS *για το πρόβλημα* $P \mid r_j,\ pmtn \mid \sum_j w_j C_j$ *το οποίο εκτελείται σε χρόνο* $O(n \log n + 2^{poly(1/\epsilon)})$ *και βρίσκει μία* $(1+\epsilon)$*-προσεγγιστική χρονοδρομολόγηση των διεργασιών.*

3.4 Συμπεράσματα

Στο κεφάλαιο αυτό μελετήσαμε γνωστά προσεγγιστικά σχήματα πολυωνυμικού χρόνου για προβλήματα ελαχιστοποίησης του μέσου (βεβαρημένου) χρόνου ολοκλήρωσης.

Στην Ενότητα 3.2 παρουσιάσαμε ένα FPTAS για το πρόβλημα $P2 \parallel \sum_j w_j C_j$ χρησιμοποιώντας την τεχνική αποκοπής (βλ. [86]). Σε αντίθεση με την τεχνική μετασχηματισμού των δεδομένων της εισόδου (βλ. Ενότητα 2.2) και κατ' επέκταση των μεθόδων που χρησιμοποιήσαμε στα επόμενα αποτελέσματα, τα οποία εμπεριέχουν μετασχηματισμούς εισόδου (βλ. Ενότητα 3.3), η τεχνική αυτή δεν εφαρμόζει στρογγυλοποίηση των αριθμητικών δεδομένων της εισόδου. Χρησιμοποιεί εξ' αρχής ένα δυναμικό πρόγραμμα (έναν ακριβή αλγόριθμο) και μειώνει επαναληπτικά το πλήθος των εφικτών λύσεων που παράγονται κατά την εκτέλεσή του, αφαιρώντας σε κάθε επανάληψη λύσεις που βρίσκονται "κοντά" ως προς το κόστος τους. Ο στόχος της τεχνικής αποκοπής είναι να μετατρέψει το πλήθος των εφικτών λύσεων από ψευδοπολυωνυμικό που ήταν αρχικά, σε πολυωνυμικό ως προς το μέγεθος της εισόδου, με όσο το δυνατόν μικρότερη απόκλιση από τη

βέλτιστη λύση. Λόγω της απαίτησης εύρεσης ενός ακριβούς αλγορίθμου ψευδοπολυωνυμικού χρόνου, η τεχνική αυτή δεν βοηθαέι στην εύρεση ενός PTAS για ένα strongly **NP**-hard πρόβλημα (βλ. Ενότητα 3.2). Για το σχεδιασμό τέτοιων αποτελεσμάτων, δείχνει να είναι απαραίτητη η εφαρμογή μετασχηματισμών στα δεδομένα της εισόδου του προβλήματος. Αυτό διαφαίνεται και μέσα από τα αποτελέσματα των επόμενων ενοτήτων.

Στην Ενότητα 3.3.2 παρουσιάσαμε ένα PTAS για το πρόβλημα $1\,|\,r_j|\,\sum_j C_j$ και στην Ενότητα 3.3.3 παρουσιάσαμε ένα PTAS για το πρόβλημα $P\,|\,r_j|\,\sum_j C_j$. Τροποποιώντας κατάλληλα το τελευταίο, προκύπτει ένα ακόμα PTAS για την preemptive εκδοχή αυτού, $P\,|\,r_j,\ pmtn|\,\sum_j w_jC_j$ (βλ. Ενότητα 3.3.3). Το πρώτο αποτέλεσμα οφείλεται στους Karger και Stein [2], ενώ τα δύο επόμενα στους Chekuri και Khanna [2, 13].

Η βασική ιδέα για το σχεδιασμό των παραπάνω PTAS είναι κατά κάποιον τρόπο κοινή: εφαρμόζουμε ένα σταθερό πλήθος μετασχηματισμών που απλοποιούν την είσοδο του προβλήματος, χωρίς να αυξάνουν δραματικά την τιμή της αντικειμενικής συνάρτησης, έτσι ώστε το αποτέλεσμα που προκύπτει να μπορεί να επιλυθεί με αποδοτικό τρόπο. Βέβαια, τόσο οι μετασχηματισμοί όσο και ο τρόπος επίλυσης του μετασχηματισμένου στιγμιοτύπου διαφέρουν ανά περίπτωση. Συγκεκριμένα, για το πρόβλημα $1\,|\,r_j|\,\sum_j C_j$ εφαρμόσαμε μία σειρά από μετασχηματισμούς εισόδου στο αρχικό στιγμιότυπο και εκτελέσαμε τον κανόνα SPT για να χρονοδρομολογήσουμε το μεγαλύτερο μέρος των διεργασιών, ενώ για τις υπόλοιπες διεργασίες απαριθμήσαμε όλες τις εφικτές χρονοδρομολογήσεις τους. Για τα προβλήματα $P\,|\,r_j|\,\sum_j w_jC_j$ και $P\,|\,r_j,\ pmtn|\,\sum_j w_jC_j$, λόγω της ύπαρξης κάποιου θετικού βάρους w_j για κάθε διεργασία j, χρειάστηκε να εφαρμόσουμε κάποιους επιπλέον μετασχηματισμούς μετατόπισης (βλ. Λήμματα 3.3, 3.4 και 3.8). Βάσει αυτών κατασκευάσαμε ένα κατάλληλο (πολυωνυμικού μεγέθους ως προς τα $\{m, n\}$) υποσύνολο όλων των ενδεχόμενων εφικτών χρονοδρομολογήσεων και πάνω σ' αυτό εφαρμόσαμε δυναμικό προγραμματισμό για να επιλέξουμε την καλύτερη δυνατή χρονοδρομολόγηση. Αξίζει να σημειωθεί ότι οι περισσότερες από τις ιδέες αυτές, όπως π.χ. η απαρίθμηση ενός μικρού υποσυνόλου διεργασιών μετά τη χρονοδρομολόγηση των υπολοίπων, χρησιμοποιήθηκαν για πρώτη φορά στα πλαίσια των επικείμενων αποτελεσμάτων. Οι ίδιες ιδέες εφαρμόστηκαν και σε μεταγενέστερες εργασίες με την ίδια επιτυχία: στις εργασίες [12, 2] παρουσιάζονται προσεγγιστικά σχήματα για τα προβλήματα $Rm\,|\,r_j|\,\sum_j w_jC_j$ και $Q\,|\,r_j|\,\sum_j w_jC_j$ (το Q υποδηλώνει γενικό πλήθος από παράλληλες σχετιζόμενες μηχανές δεδομένης ταχύτητας εκτέλεσης (uniformly related parallel machines)).

Όπως αναφέραμε στην Ενότητα 3.1, πριν τη δημοσίευση των παραπάνω προσεγγιστικών σχημάτων είχαν προταθεί αρκετοί προσεγγιστικοί αλγόριθμοι για τα αντίστοιχα προβλήματα. Κάποιοι από αυτούς, όπως ο $(\frac{e}{e-1})$-προσεγγιστικός αλγόριθμος για το πρόβλημα $1\,|\,r_j|\,\sum_j C_j$ [15], συνεχίζουν να αποτελούν πρώτη επιλογή λόγω της απλότητας και της αποδοτικότητας τους.

Αυτό βέβαια σε καμία περίπτωση δεν μειώνει την αξία των προσεγγιστικών σχημάτων. Ιδιαίτερα αν λάβουμε υπόψη την υπολογιστική πολυπλοκότητα των προβλημάτων που μελετήθηκαν, αμέσως συμπεραίνουμε ότι έχουμε αποκτήσει το καλύτερο δυνατό αποτέλεσμα για καθένα από αυτά, εκτός βέβαια και αν **P** = **NP** (βλ. Ενότητα Α.2.1). Το πρόβλημα ελαχιστοίησης του μέσου (βεβαρημένου) χρόνου ολοκλήρωσης είναι δεδομένο ότι αποτελεί ένα από τα πιο ενδιαφέροντα προβλήματα της Θεωρίας Χρονοδρομολόγησης. Ιδιαίτερα τα τελευταία χρόνια, έχουν δοθεί πολλά σημαντικά αποτελέσματα για διάφορες εκδοχές αυτού, περιλαμβανομένων προσεγγιστικών σχημάτων, προσεγγιστικών αλγορίθμων αλλά και αποτελεσμάτων μη προσέγγισης. Παρόλα αυτά συνεχίζουν να υπάρχουν σημαντικά ανοιχτά προβλήματα. Ακολούθως παραθέτουμε κάποια από αυτά.

- $\mathbf{1}\,|\mathbf{prec}|\,\sum_{\mathbf{j}}\mathbf{w_jC_j}$ (η παράμετρος prec υποδηλώνει κάποια μερική διάταξη ως προς τη σειρά εκτέλεσης των διεργασιών). Η μείωση του χάσματος προσεγγισιμότητας του προβλήματος είναι ένα από τα σημαντικότερα ανοιχτά προβλήματα της Θεωρίας Χρονοδρομολόγησης. Μέχρι σήμερα έχουν δοθεί αρκετοί 2-προσεγγιστικοί αλγόριθμοι για το πρόβλημα (βλ. [36, 14, 17]), ενώ παραμένει άγνωστο αν το πρόβλημα ανήκει στην κλάση των **APX**-hard προβλημάτων. Σε μία πρόσφατη εργασία των Correa και Schulz [20] εξακριβώθηκε ότι το πρόβλημα είναι συναφές με το πρόβλημα Vertex Cover, ενώ οι Ambühl και Mastrolilli [3] απέδειξαν τελικά ότι πρόκειται για ειδική περίπτωση του προβλήματος Vertex Cover. Για το Vertex Cover γνωρίζουμε ότι δεν μπορεί να προσεγγιστεί με παράγοντα καλύτερο από $10\sqrt{5} - 21 \approx 1.36$ [45], ενώ εικάζεται ισχυρά ότι δεν επιδέχεται $(2-\delta)$-προσεγγιστικό αλγόριθμο πολυωνυμικού χρόνου, για μικρό $\delta > 0$. Θα είχε ίσως ενδιαφέρον να δούμε αν με τη βοήθεια αυτών των αποτελεσμάτων μπορούμε να οδηγηθούμε σε κάτι αντίστοιχο για το πρόβλημα $1\,|prec|\,\sum_j w_jC_j$.
- $\mathbf{P}\,|\mathbf{prec}|\,\sum_{\mathbf{j}}\mathbf{w_jC_j}$. Έχει δοθεί ένας 4-προσεγγιστικός αλγόριθμος [71] και παραμένει ανοιχτή η βελτίωση του παράγοντα προσέγγισης. Από πλευράς μη προσεγγισιμότητας θα είχε ενδιαφέρον να αποδειχθεί ότι το πρόβλημα είναι το ίδιο δύσκολο με το αντίστοιχο πρόβλημα ελαχιστοποίησης του makespan, $P\,|prec|\,C_{max}$, για το οποίο είναι γνωστός ένας 2-προσεγγιστικός αλγόριθμος [34] καθώς και ότι δεν μπορεί να προσεγγιστεί με παράγοντα μικρότερο του 4/3 [65].
- $\mathbf{R}\,||\,\sum_{\mathbf{j}}\mathbf{w_jC_j}$, $\mathbf{R}\,|\mathbf{r_j}|\,\sum_{\mathbf{j}}\mathbf{w_jC_j}$. Για το πρώτο έχουν δοθεί δύο 3/2-προσεγγιστικοί αλγόριθμοι, ανεξάρτητα, από τους Skutella[89] και Sethuraman & Squillante [88], ενώ για το δεύτερο ο Skutella [90] έχει παρουσιάσει ένα 2-προσεγγιστικό αλγόριθμο. Επιπλέον, οι Hoogeveen, Schuurman και Woeginger [41] απέδειξαν ότι και τα δύο προβλήματα είναι **MAX-SNP**-hard και επομένως δεν επιδέχονται το σχεδιασμό ενός PTAS (βλ. Ενότητα Α.2.1). Παραμένει

ανοιχτό ερώτημα ο σχεδιασμός αλγορίθμων με παράγοντες προσέγγισης $3/2 - \delta$, για το πρόβλημα $R \mid\mid \sum_j w_j C_j$ και $2 - \delta$ για το πρόβλημα $R \mid r_j \mid \sum_j w_j C_j$. Στην εργασία των Schulz και Skutella [83] δίνονται κάποιες κατευθύνσεις γι' αυτό.

Για μία περαιτέρω επισκόπηση ανοιχτών προβλημάτων που έχουν ως στόχο την ελαχιστοίηση του μέσου (βεβαρημένου) χρόνου ολοκλήρωσης ο αναγνώστης παραπέμπεται στις εργασίες [85, 13].

Κεφάλαιο 4

Χρονοδρομολόγηση με δεδομένες προθεσμίες

Μέχρι τώρα μελετήσαμε προβλήματα όπου η χρονοδρομολόγηση των διεργασιών δεν θέτει περιορισμούς όσον αφορά το χρόνο ολοκλήρωσης κάθε διεργασίας ατομικά. Η κύρια απαίτηση ήταν οι διεργασίες να ολοκληρώνονται στον καλύτερο δυνατό χρόνο με σκοπό την ελαχιστοποίηση της αντίστοιχης αντικειμενικής συνάρτησης. Αν συσχετίσουμε λοιπόν κάθε διεργασία με μία θετική ακέραια προθεσμία (due date) και με μία απώλεια για κάθε αποτυχία ολοκλήρωσης της εκτέλεσης της στον χρόνο αυτό, σε μία δεδομένη χρονοδρομολόγηση, τότε προκύπτει μία σειρά αντικειμενικών συναρτήσεων οι οποίες καλούνται *συναρτήσεις απώλειας*. Μερικές από τις περισσότερο διαδεδομένες, είναι η καθυστέρηση (tardiness), η αργοπορία (Lateness) και η πρόωρη ολοκλήρωση (earliness) (βλ. Ενότητα 1.1). Τα προβλήματα που μελετώνται στο τρέχον κεφάλαιο στοχεύουν στην βελτιστοποίηση κάποιας συνάρτησης απώλειας.

Τα περισσότερα προβλήματα που μελετάμε αποτελούν ειδικές περιπτώσεις του προβλήματος ελαχιστοποίησης της συνολικής βεβαρημένης καθυστέρησης (minimization of total weighted tardiness). Η είσοδος του προβλήματος αποτελείται από n διεργασίες και $m \geq 1$ μηχανές. Κάθε διεργασία j, $1 \leq j \leq n$, διαθέτει ένα χρόνο επεξεργασίας p_j, μία προθεσμία d_j και ένα θετικό βάρος w_j. Στόχος μας είναι να χρονοδρομολογήσουμε τις διεργασίες στις μηχανές έτσι ώστε να ελαχιστοποιήσουμε την συνολική βεβαρημένη καθυστέρηση $\sum_j w_j T_j$. Υπενθυμίζουμε ότι η καθυστέρηση μίας διεργασίας j, σε μία δεδομένη χρονοδρομολόγηση, ισούται με $T_j = \max\{0, C_j - d_j\}$. Περιορίζουμε τη μελέτη μας σε περιβάλλον Μοναδικής Μηχανής, όπου με βάση τη πρότυπη παράσταση τριών πεδίων (βλ. Ενότητα 1.1) το πρόβλημα συμβολίζεται με $1 \parallel \sum_j w_j T_j$. Στην ειδική περίπτωση του προβλήματος όπου τα βάρη των διεργασιών είναι *συμβατά (agreeable)*, ισχύει ότι $p_i < p_j$ τότε $w_i \geq w_j$ και το πρόβλημα συμβολίζεται με $1 \,|w_j \; agreeable| \sum_j w_j T_j$, ενώ στην

περίπτωση όπου όλες οι διεργασίες διαθέτουν μία κοινή προθεσμία d το πρόβλημα συμβολίζεται με $1\,|\,d_j = d\,|\,\sum_j w_j T_j$.

Συναφές με το πρόβλημα ελαχιστοποίησης της συνολικής βεβαρημένης καθυστέρησης είναι το πρόβλημα ελαχιστοποίησης του συνολικού βάρους των καθυστερημένων διεργασιών (minimization of the weighted number of tardy jobs). Η είσοδος είναι κοινή με το πρώτο και ο στόχος είναι να ελαχιστοποιήσουμε την ποσότητα $\sum_j w_j U_j$. Θυμίζουμε ότι σε μία δεδομένη χρονοδρομολόγηση, για κάθε διεργασία j, η συνάρτηση U_j παίρνει την τιμή 0 αν η j είναι πρόωρη (early), δηλαδή αν ολοκληρώνεται πριν την προθεσμία της d_j και την τιμή 1 αν η j είναι καθυστερημένη (tardy). Μελετάμε το πρόβλημα σε περιβάλλον Μοναδικής Μηχανής, $1\,\|\,\sum_j w_j U_j$.

Ένα ακόμα σχετικό πρόβλημα, είναι το πρόβλημα μεγιστοποίησης της συνολικής βεβαρημένης πρόωρης ολοκλήρωσης (maximization of total weighted earliness). Σε περιβάλλον Μοναδικής Μηχανής, η είσοδος είναι κοινή με το πρόβλημα $1\,\|\,\sum_j w_j T_j$ και ο στόχος είναι να μεγιστοποιήσουμε την ποσότητα $\sum_j w_j E_j$. Για μία δεδομένη χρονοδρομολόγηση, η πρόωρη ολοκλήρωση E_j κάθε διεργασίας j ισούται με $E_j = \max\{0, d_j - C_j\}$. Το πρόβλημα συμβολίζεται με $1\,\|\,\textit{maximize}\ \sum_j w_j E_j$.

Στην Ενότητα 4.2 παρουσιάζουμε αλγορίθμους ψευδοπολυωνυμικού χρόνου για τα προβλήματα $1\,\|\,\sum_j w_j U_j$, $1\,\|\,\textit{maximize}\ \sum_j w_j E_j$ και $1\,|\,d_j = d\,|\,\sum_j w_j T_j$. Οι αλγόριθμοι αυτοί προέρχονται από την εργασία των Lawler και Moore [63] και προκύπτουν μέσω κατάλληλης εφαρμογής ενός δυναμικού προγράμματος. Στην Ενότητα 4.3 παρουσιάζουμε έναν ψευδοπολυωνυμικό αλγόριθμο για το πρόβλημα $1\,|\,w_j\ \textit{agreeable}\,|\,\sum_j w_j T_j$. Ο αλγόριθμος αυτός οφείλεται στον Lawler [60]. Στην Ενότητα 4.4 δίνουμε ένα FPTAS για το πρόβλημα $1\,\|\,\sum_j T_j$ [61] το οποίο χρησιμοποιεί τον ψευδοπολυωνυμικό αλγόριθμο της Ενότητας 4.3. Τέλος, στην ίδια ενότητα, παρουσιάζουμε ένα FPTAS για το πρόβλημα $1\,\|\,\sum_j w_j U_j$ από την εργασία [86]. Για το αποτέλεσμα αυτό χρησιμοποιούμε τον ψευδοπολυωνυμικό αλγόριθμο των Lawler και Moore [63] που περιγράφεται στην Ενότητα 4.2.

4.1 Επισκόπηση γνωστών αποτελεσμάτων

Η χρονοδρομολόγηση με δεδομένες προθεσμίες αποτέλεσε αντικείμενο μελέτης από τα πρώτα χρόνια της Θεωρίας Χρονοδρομολόγησης. Σε μία από τις πρώτες εργασίες που παρουσιάστηκαν, ο Jackson [46] μελέτησε το πρόβλημα ελαχιστοποίησης της μέγιστης καθυστέρησης στην περίπτωση που υπάρχουν δεδομένοι χρόνοι αποδέσμευσης για κάθε διεργασία. Θυμίζουμε ότι σε μία δεδομένη χρονοδρομολόγηση η μέγιστη καθυστέρηση ισούται με $T_{max} = \max_j T_j$. Μεταξύ άλλων, απέδειξε ότι σε μία χρονοδρομολόγηση ενός συνόλου διεργασιών για τις οποίες δεν υπάρχουν δεδομένοι χρόνοι αποδέσμευσης, όλες οι διεργασίες ολοκληρώνουν την εκτέλεσή τους πρόωρα αν και μόνο αν αυτές ολοκληρώνονται πρόωρα όταν εκτελούνται σε αύξουσα διάταξη ως προς τις

προθεσμίες τους. Η εκτέλεση των διεργασιών βάσει αυτής της διάταξης είναι γνωστή ως κανόνας EDD (βλ. Ενότητα 1.2). Ο Smith, στη γνωστή εργασία του [92], στηριζόμενος στο αποτέλεσμα του Jackson απέδειξε πως η εφαρμογή του κανόνα EDD δίνει μία βέλτιστη πολυωνυμική λύση στα προβλήματα $1 \parallel T_{max}$ και $1 \parallel L_{max}$. Για μία δεδομένη χρονοδρομολόγηση το $L_{max} = \max_j L_j$ συμβολίζει τη μέγιστη αργοπορία (βλ. Ενότητα 1.1).

Ένα πολύ απλό και χρήσιμο, όπως θα δούμε, αποτέλεσμα δόθηκε από τον McNaughton [69], για το πρόβλημα $1 \parallel \sum_j w_j T_j$. Απέδειξε ότι σε μία βέλτιστη χρονοδρομολόγηση, οι καθυστερημένες διεργασίες είναι διατεταγμένες βάσει του κανόνα WSPT, δηλαδή σε φθίνουσα διάταξη ως προς το λόγο τους w_j/p_j. Αξίζει να σημειωθεί ότι το αποτέλεσμα αυτό προκύπτει έμμεσα μέσω του προγενέστερου αποτελέσματος του Smith - η εφαρμογή του κανόνα του Smith επιλύει βέλτιστα το πρόβλημα $1 \parallel \sum_j w_j C_j$ - αρκεί να παρατηρήσουμε ότι για οποιαδήποτε χρονοδρομολόγηση, το $\sum_j w_j T_j = \sum_{j' \in K} w_{j'} C_{j'} - \sum_{j' \in K} w_{j'} d_{j'}$, όπου K είναι το σύνολο των καθυστερημένων διεργασιών. Υπενθυμίζουμε ότι ο κανόνας WSPT είναι ισοδύναμος με τον κανόνα του Smith (βλ. Ενότητα 1.2). Στην ίδια εργασία του [69], ο McNaughton απέδειξε ότι οι preemptive και non-preemptive εκδοχές του προβλήματος, για οποιοδήποτε στιγμιότυπο, έχουν την ίδια ελάχιστη βεβάρημένη καθυστέρηση.

Ο Moore [70] μελέτησε την ειδική περίπτωση του προβλήματος $1 \parallel \sum_j w_j U_j$, όπου όλες οι διεργασίες έχουν μοναδιαίο βάρος, $1 \parallel \sum_j U_j$ και σχεδίασε έναν αλγόριθμο πολυωνυμικού χρόνου: εκτέλεσε επαναληπτικά τις διεργασίες βάσει του κανόνα EDD και κάθε φορά που μία διεργασία j ολοκληρώνει την εκτέλεσή της μετά την προθεσμία της d_j, αντάλλαξέ την με τη διεργασία που έχει το μεγαλύτερο χρόνο επεξεργασίας, από τις διεργασίες που έχουν ήδη εκτελεστεί. Ο αλγόριθμος του Moore, όπως είναι γνωστός, βρίσκει μία βέλτιστη λύση σε χρόνο $O(n \log n)$ στην οποία όλες οι πρόωρες διεργασίες είναι διατεταγμένες βάσει του κανόνα EDD και ακολουθούνται από τις καθυστερημένες διεργασίες σε αυθαίρετη διάταξη. Αρκετά αργότερα ο Lawler [59] απέδειξε, βάσει του αλγορίθμου του Moore, ότι και η περίπτωση του προβλήματος $1 \parallel \sum_j w_j U_j$ όπου τα βάρη είναι συμβατά, επιλύεται βέλτιστα στον ίδιο χρόνο.

Το πρόβλημα $1 \parallel \sum_j w_j U_j$ για γενικά δεδομένα εισόδου, μελετήθηκε στην εργασία των Lawler και Moore [63], όπου δόθηκε ένας βέλτιστος αλγόριθμος ψευδοπολυωνυμικού χρόνου. Ο αλγόριθμος αυτός προκύπτει έπειτα από κατάλληλη εφαρμογή ενός δυναμικού προγράμματος, λαμβάνοντας υπόψη το παραπάνω αποτέλεσμα του Moore [70], καθώς και ένα παραπλήσιο αποτέλεσμα του Rothkopf [79] (για γενικά δεδομένα εισόδου), υπό την έννοια πως μπορεί κανείς να υποθέσει ότι σε μία βέλτιστη χρονοδρομολόγηση οι πρόωρες διεργασίες προηγούνται των καθυστερημένων και επιπλέον βρίσκονται σε φθίνουσα διάταξη ως προς τις προθεσμίες τους. Στην ίδια εργασία [63], το ίδιο δυναμικό πρόγραμμα εφαρμόζεται κατάλληλα και σε περιπτώσεις άλλων

προβλημάτων, όπως το πρόβλημα $1 \parallel \textit{maximize} \sum_j w_j E_j$ και το πρόβλημα $1 \mid d_j = d \mid \sum_j w_j T_j$, δίνοντας βέλτιστους ψευδοπολυωνυμικούς αλγόριθμους γι' αυτά. Τα αποτελέσματα της εργασίας των Lawler και Moore [63] περιγράφονται αναλυτικά στην επόμενη ενότητα (βλ. Ενότητα 4.2).

Όπως μπορεί κανείς να παρατηρήσει, τα παραπάνω αποτελέσματα (εξαιρουμένου αυτού του Lawler [59]) προηγήθηκαν χρονικά της διατύπωσης της Θεωρίας της **NP**-πληρότητας. Έτσι, στα χρόνια που ακολούθησαν, δημιουργήθηκε έντονο ενδιαφέρον ως προς τη μελέτη της υπολογιστικής πολυπλοκότητας των ανωτέρω προβλημάτων. Τα αποτελέσματα που παρουσιάστηκαν προσέφεραν ουσιαστικές κατευθύνσεις για την περαιτέρω διερεύνησή τους.

Το πρώτο σχετικό αποτέλεσμα περιλαμβάνεται στη θεμελιώδη εργασία του Karp [49] και αποδεικνύει ότι το πρόβλημα $1 \parallel \sum_j w_j U_j$ είναι **NP**-hard υπό τη συνήθη έννοια. Αργότερα οι Gens και Levner [31] σχεδίασαν ένα FPTAS για το ίδιο πρόβλημα, το οποίο αποτελεί και το καλύτερο δυνατό αποτέλεσμα, εκτός βέβαια και αν **P** = **NP**. Το αποτέλεσμα αυτό προέκυψε έπειτα από εφαρμογή της τεχνικής αποκοπής των Ibarra και Kim [44] (βλ. Ενότητα 3.2), στον βέλτιστο ψευδοπολυωνυμικό αλγόριθμο των Lawler και Moore [63] (βλ. Ενότητα 4.4). Ο Lawler [60] απέδειξε ότι το πρόβλημα $1 \parallel \sum_j w_j T_j$ για γενικά δεδομένα εισόδου, είναι strongly **NP**-hard. Στην ίδια εργασία [60], ο Lawler σχεδίασε έναν βέλτιστο ψευδοπολυωνυμικό αλγόριθμο για την ειδική περίπτωση του προβλήματος όπου τα βάρη είναι συμβατά. Αργότερα, χρησιμοποιώντας τον αλγόριθμο αυτό, σχεδίασε ένα FPTAS για το πρόβλημα $1 \parallel \sum_j T_j$ [61], χωρίς όμως να καθορίσει την υπολογιστική δυσκολία του προβλήματος. Αρκετά χρόνια μετά το FPTAS του Lawler [61], οι Du και Leung [24] απέδειξαν ότι το πρόβλημα $1 \parallel \sum_j T_j$ είναι **NP**-hard υπό τη συνήθη έννοια. Τα δύο αποτελέσματα του Lawler [60, 61], παρουσιάζονται στις Ενότητες 4.3, 4.4.

Οι Lenstra, Rinnooy Kan και Brucker [67] έδειξαν ότι ακόμη και η ειδική περίπτωση του προβλήματος $1 \parallel \sum_j w_j T_j$ όπου το πλήθος των διακεκριμένων προθεσμιών των διεργασιών είναι σταθερό και ίσο με 2, είναι **NP**-hard υπό τη συνήθη έννοια. Παρέμενε όμως ανοιχτό το ερώτημα αν και το πρόβλημα $1 \mid d_j = d \mid \sum_j w_j T_j$ είναι το ίδιο δύσκολο. Το 1992 ο Yuan [96] έδειξε ότι είναι **NP**-hard υπό τη συνήθη έννοια. Αξίζει να σημειωθεί ότι οι ειδικές περιπτώσεις του προβλήματος όπου τα βάρη των διεργασιών είναι ίσα ή συμβατά επιλύονται βέλτιστα, σε χρόνο $O(n \log n)$ η πρώτη (εφαρμόζοντας τον κανόνα SPT) και σε χρόνο $O(n^2)$ η δεύτερη (βλ. [63]).

Πολύ πρόσφατα οι Kolliopoulos και Steiner [53] παρουσίασαν δύο διαφορετικούς βέλτιστους ψευδοπολυωνυμικούς αλγορίθμους για το γενικότερο πρόβλημα όπου το πλήθος των προθεσμιών είναι σταθερό (ίσο με $m > 1$) και βάσει αυτών σχεδίασαν ένα προσεγγιστικό σχήμα για το πρόβλημα. Η χρονική πολυπλοκότητα αυτού του προσεγγιστικού σχήματος είναι ψευδοπολυωνυμική λόγω του ότι φράσσεται από ένα πολυώνυμο του μέγιστου βάρους των διεργασιών. Έτσι, στην περίπτωση που τα βάρη των διεργασιών έχουν μέγεθος πολυωνυμικό ως προς το μέγεθος της

εισόδου προκύπτει ένα FPTAS για το πρόβλημα. Οι Kellerer και Strusevich [51] έδωσαν ένα FPTAS για το πρόβλημα $1 \mid d_j = d \mid \sum_j w_j T_j$ χωρίς τον παραπάνω περιορισμό στο μέγεθος των βαρών, αφήνωντας ανοιχτή την εύρεση ενός FPTAS για την περίπτωση που οι διεργασίες διαθέτουν σταθερό αριθμό διακεκριμένων προθεσμιών.

Οι Kovalyov και Werner [55] μελέτησαν την προσεγγισιμότητα του προβλήματος ελαχιστοποίησης της συνολικής βεβαρημένης καθυστέρησης όταν οι διεργασίες διαθέτουν μία κοινή προθεσμία, σε περιβάλλον Παράλληλων Μηχανών και στην ειδική περίπτωση που τα βάρη των διεργασιών είναι μοναδιαία, $Pm \mid d_j = d \mid \sum_j T_j$. Εκμεταλλευόμενοι το γεγονός ότι είναι **NP**-πλήρες να αποφασίσουμε αν υπάρχει χρονοδρομολόγηση με μηδενική συνολική καθυστέρηση, απέδειξαν ότι, εκτός αν **P** = **NP**, δεν υπάρχει ρ-προσεγγιστικός αλγόριθμος για το $Pm \mid d_j = d \mid \sum_j T_j$ με $\rho < \infty$. Το αποτέλεσμα αυτό οδηγεί άμεσα στο συμπέρασμα ότι για την επίτευξη ενός σταθερού παράγοντα προσέγγισης θα πρέπει να αποφευχθούν ενδεχόμενες λύσεις στις οποίες η αντικειμενική συνάρτηση έχει μηδενική τιμή. Αυτό επιτυγχάνεται με την πρόσθεση μίας κατάλληλης θετικής ποσότητας b στην τιμή της αντικειμενικής συνάρτησης. Ακολούθως στην ίδια εργασία [55], παρουσιάστηκε ένα FPTAS για το πρόβλημα $Pm \mid d_j = d \mid \sum_j T_j + d$, αφού πρροηγουμένως αποδείχθηκε ότι, εκτός αν **P** = **NP**, δεν υπάρχει $(\rho+1)$-προσεγγιστικός αλγόριθμος για το πρόβλημα $Pm \mid d_j = d \mid \sum_j T_j + b$ με $\rho < 1/b$. Το τελευταίο υποδηλώνει ότι δεν υπάρχει FPTAS για το πρόβλημα αν το b φράσσεται από ένα πολυώνυμο του μεγέθους της εισόδου ενός στιγμιοτύπου, εκτός βέβαια και αν **P** = **NP**.

Οι Kolliopoulos και Steiner [54] επέκτειναν την παραπάνω ιδέα στην βεβαρημένη περίπτωση όπου κάθε διεργασία διαθέτει μία διαφορετική προθεσμία και μελέτησαν τη σχέση μεταξύ των αντικειμενικών συναρτήσεων $\sum_j w_j(T_j + d_j)$ και $\sum_j w_j C_j$, σε διάφορα περιβάλλοντα χρονοδρομολόγησης. Απέδειξαν ότι κάθε ρ-προσεγγιστικός αλγόριθμος για την ελαχιστοποίηση της αντικειμενικής συνάρτησης $\sum_j w_j C_j$ είναι ένας $(\rho+1)$-προσεγγιστικός αλγόριθμος για την ελαχιστοποίηση της αντικειμενικής συνάρτησης $\sum_j w_j(T_j + d_j)$. Ειδικά για κάποια προβλήματα ελαχιστοποίησης της $\sum_j w_j C_j$, για τα οποία υπάρχουν αλγόριθμοι γραμμικού προγραμματισμού με συγκεκριμένα χαρακτηριστικά, η παραπάνω σχέση βελτιώνεται έτσι ώστε κάθε ρ-προσεγγιστικός αλγόριθμος για την ελαχιστοποίηση της $\sum_j w_j C_j$ να είναι επίσης ρ-προσεγγιστικός για την ελαχιστοποίηση της αντικειμενικής συνάρτησης $\sum_j w_j(T_j + d_j)$. Από τη συσχέτιση αυτή και βάσει των ήδη γνωστών αποτελεσμάτων προσέγγισης για αρκετά προβλήματα χρονοδρομολόγησης με αντικειμενική συνάρτηση $\sum_j w_j C_j$ (βλ. Ενότητα 3.1), προκύπτει ένας μεγάλος αριθμός αποτελεσμάτων προσέγγισης για προβλήματα με αντικειμενική συνάρτηση $\sum_j w_j(T_j + d_j)$. Στην ίδια εργασία [54] δίνεται ένα FPTAS για την ελαχιστοποίηση της αντικειμενικής συνάρτησης $\sum_j w_j(T_j + d_j)$ σε περιβάλλον μοναδικής μηχανής όταν οι διεργασίες έχουν μία κοινή προθεσμία. Το πρόβλημα αυτό συμβολίζεται με $1 \mid d_j = d \mid \sum_j w_j(T_j + d_j)$.

Όπως προαναφέραμε, το πρόβλημα $1 \parallel \sum_j w_j T_j$ για γενικά δεδομένα εισόδου έχει αποδειχθεί strongly **NP**-hard [60]. Το γεγονός αυτό αποκλείει την εύρεση ενός ψευδοπολυωνυμικού αλγορίθμου, εκτός και αν **P** = **NP**, σε αντίθεση με τις ειδικές περιπτώσεις του προβλήματος που αναφέραμε προηγουμένως. Το καλύτερο δυνατό αποτέλεσμα θα ήταν ο σχεδιασμός ενός PTAS, εκτός βέβαια και αν **P** = **NP** (βλ. Ενότητα Α.2.1), το οποίο όμως παραμένει ακόμα ένα πολύ σημαντικό ανοιχτό ερώτημα. Παρόλα αυτά, ξεχωριστό ενδιαφέρον παρουσιάζουν τα μέχρι τώρα γνωστά απότελέσματα για το πρόβλημα. Στη συνέχεια αναφέρουμε κάποια από αυτά.

Το μοναδικό, εξόσων γνωρίζουμε, μη τετριμμένο αποτέλεσμα προσέγγισης για το πρόβλημα παρουσιάστηκε πρόσφατα από τους Cheng, Ng, Yuan και Liu [16]. Απέδειξαν ότι η διάταξη χρονοδρομολόγησης που ελαχιστοποιεί την ποσότητα $\max_j w_j T_j$, αποδίδει μία $(n-1)$-προσέγγιση για το πρόβλημα. Σημειώνουμε ότι μία βέλτιστη τέτοια διάταξη μπορεί να κατασκευαστεί σε χρόνο $O(n^2)$, εφαρμόζοντας τον αλγόριθμο του Lawler [58]. Ένας από τους πιο αποδοτικούς (μη πολυωνυμικούς) αλγορίθμους δόθηκε το 1985 από τους Potts και Van Wassenhove [77]. Πρόκειται για έναν αλγόριθμο Διαχωρισμού και Αποτίμησης (Branch and Bound) που χρησιμοποιεί την τεχνική της Lagrangian αποδυνάμωσης (Lagrangian relaxation) για την εύρεση κάτω φραγμάτων. Ο αλγόριθμος αυτός, εκτός των άλλων, υποδεικνύει και πρακτικά το μέγεθος της δυσκολίας του προβλήματος, καθώς, για στιγμιότυπα με περισσότερες από 50 διεργασίες, παύει να είναι αποδοτικός ως προς την εύρεση μίας βέλτιστης λύσης.

Ακολούθως παρουσιάστηκαν αρκετοί αλγόριθμοι που βασίζονται σε τεχνικές προερχόμενες από την κλασσική μέθοδο της Τοπικής Αναζήτησης (Local Search): προσομοιωμένη ανόπτηση (simulated annealing), αναζήτηση ταμπού (tabu search), επαναλαμβανόμενη τοπική αναζήτηση (iterated local search) και γενετικοί αλγόριθμοι (genetic algorithms) (βλ. [68, 78, 21, 18, 22]). Ανάμεσα σ' αυτούς, ξεχωρίζουμε ως προς την αποδοτικότητά του έναν πρόσφατο αλγόριθμο των Congram, Potts και van de Velde [18], ο οποίος εφαρμόζει μία νέα τεχνική τοπικής αναζήτησης που καλείται dynasearch· χρησιμοποιεί δυναμικό προγραμματισμό για να εκτελέσει αναζήτηση πολυωνυμικού χρόνου σε γειτονιές εκθετικού μεγέθους. Για μία εκτενή επισκόπηση των παραπάνω, αλλά και επιπλέον αλγορίθμων, ο αναγνώστης παραπέμπεται στις εργασίες [1, 87].

4.2 Ψευδοπολυωνυμικοί αλγόριθμοι για τη βελτιστοποίηση συναρτήσεων απώλειας

Ας θεωρήσουμε το εξής πρόβλημα: διαθέτουμε n διεργασίες οι οποίες πρόκειται να χρονοδρομολογηθούν διαδοχικά, βάσει μίας προκαθορισμένης διάταξης, έστω $1, 2, \ldots, n$. Κάθε διεργασία μπορεί να χρονοδρομολογηθεί με δύο διαφορετικούς τρόπους. Με βάση τον πρώτο τρόπο, για

κάθε διεργασία j απαιτούνται a_j μονάδες χρόνου επεξεργασίας, ενώ η απώλεια που δημιουργείται κατά την ολοκλήρωση της j σε μία χρονική στιγμή t, ισούται με $\alpha_j(t)$ μονάδες. Βάσει του δευτέρου τρόπου, η επεξεργασία της j διαρκεί b_j μονάδες χρόνου και η αντίστοιχη απώλεια ισούται με $\beta_j(t)$ μονάδες. Ποιά ανάθεση των διαφορετικών τρόπων χρονοδρομολόγησης πρέπει να κάνουμε στις n διεργασίες έτσι ώστε να ελαχιστοποιήσουμε τη συνολική απώλεια της χρονοδρομολόγησης;

Στην Ενότητα 4.2.1 παρουσιάζουμε ένα δυναμικό πρόγραμμα [63] που επιλύει βέλτιστα, σε ψευδοπολυωνυμικό χρόνο, το παραπάνω πρόβλημα και στην Ενότητα 4.2.3 μελετάμε τις εφαρμογές του στα προβλήματα $1 \parallel \sum_j w_j U_j$, $1 \parallel \textit{maximize} \sum_j w_j E_j$ και $1 \mid d_j = d \mid \sum_j w_j T_j$.

4.2.1 Ένα "γνωστό" δυναμικό πρόγραμμα

Ακολούθως παρουσιάζουμε ένα δυναμικό πρόγραμμα για το πρόβλημα που ορίσαμε προηγουμένως. Έυκολα μπορεί κανείς να διαπιστώσει τις ομοιότητες με το γνωστό δυναμικό πρόγραμμα για το ακέραιο πρόβλημα του Σακιδίου (0-1 Knapsack) (βλ. Ενότητα 4.2.2).

Συμβολίζουμε με $f(j,t)$ την ελάχιστη συνολική απώλεια για τη χρονοδρομολόγηση των πρώτων j διεργασιών, υπό τον περιορισμό ότι η διεργασία j δεν ολοκληρώνεται μετά το πέρας της χρονικής στιγμής t. Το δυναμικό πρόγραμμα εκτελείται βάσει της ακόλουθης αναδρομής.

$$
\begin{aligned}
f(0,t) &= 0, && (t \geq 0),\\
f(j,t) &= +\infty, && (j = 0,1,\ldots,n; t < 0),\\
f(j,t) &= \min \left\{ \begin{array}{l} f(j,t-1), \\ \alpha_j(t) + f(j-1,t-a_j), \\ \beta_j(t) + f(j-1,t-b_j) \end{array} \right\}, && (j = 1,\ldots,n; t \geq 0).
\end{aligned}
\tag{4.1}
$$

Σε κάθε βήμα (j,t) της αναδρομής, η $f(j,t)$, στην μη τετριμμένη περίπτωση όπου $j = 1,\ldots,n; t \geq 0$, θα ισούται με την ελάχιστη των απωλειών που προκύπτουν κατά τη χρονοδρομολόγηση της τρέχουσας διεργασίας j με καθέναν από τους δύο διαφορετικούς τρόπους. Από τον πρώτο τρόπο, η συνολική απώλεια μετά την ολοκλήρωση της j θα είναι ίση με $\alpha_j(t) + f(j-1,t-a_j)$, ενώ από τον δεύτερο θα ισούται με $\beta_j(t) + f(j-1,t-b_j)$. Επιπλέον, πρέπει να λάβουμε υπόψη μας το ενδεχόμενο η j να ολοκληρώνει την εκτέλεση πριν τη δεδομένη στιγμή t, $C_j < t$. Έτσι, στον υπολογισμό της ελάχιστης συνολικής απώλειας $f(j,t)$ προσμετράμε και την τιμή $f(j,t-1)$. Είναι σημαντικό να παρατηρήσουμε ότι για μη τετριμμένες περιπτώσεις όπου η τρέχουσα διεργασία j αδυνατεί να ολοκληρώσει την εκτέλεση της μέχρι τη δεδομένη χρονική στιγμή t η τιμή της $f(j,t)$ καθορίζεται από τις δύο πρώτες εξισώσεις της (4.1).

Αν θεωρήσουμε ότι ο χρονικός ορίζοντας της παραπάνω διαδικασίας φράσεται άνω από μία

αρκετά μεγάλη ακέραια τιμή, έστω $T > 0$, οι διακεκριμένες τιμές της μεταβλητής t θα είναι το πολύ T. Μία ενδεχόμενη τιμή για το T θα μπορούσε να είναι το άθροισμα $\sum_j \max\{a_j, b_j\}$.

Η λύση στο πρόβλημα προκύπτει τελικά με τον υπολογισμό της τιμής $f(n, T)$, η οποία αντιστοιχεί στη ελάχιστη συνολική απώλεια από τη χρονοδρομολόγηση του συνόλου των διεργασιών βάσει της προκαθορισμένης διάταξης και υπό τον περιορισμό ότι η τελευταία διεργασία n ολκληρώνει την εκτέλεση της πριν τη χρονική στιγμή T. Ο υπολογισμός της $f(n, T)$ γίνεται σε ψευδοπολυωνυμικό χρόνο της τάξης του $O(nT)$.

Σε σχέση με προηγούμενες υπολογιστικές μεθόδους για αντίστοιχα προβλήματα χρονοδρομολόγησης (βλ. [37, 57, 82]) το δυναμικό πρόγραμμα (4.1) είναι κατά πολύ πιο αποδοτικό. Ο κύριος λόγος είναι ότι θεωρεί εξ' αρχής μία προκαθορισμένη διάταξη των διεργασιών περιορίζοντας έτσι το μέγεθος του χώρου αναζήτησης σε εκθετικό, ίσο με 2^n (για κάθε διεργασία υπάρχουν δύο διαφορετικοί τρόποι χρονοδρομολόγησης), σε αντίθεση με τις άλλες μεθόδους που εκτελούν αναζήτηση σε χώρο μεγέθους $n!$.

4.2.2 Εφαρμογές σε προβλήματα χρονοδρομολόγησης Μοναδικής Μηχανής

Στη συνέχεια μελετάμε τις εφαρμογές του δυναμικού προγράμματος (4.1) σε μία σειρά προβλημάτων χρονοδρομολόγησης Μοναδικής Μηχανής με δεδομένες προθεσμίες, που έχουν ως στόχο την ελαχιστοποίηση ή τη μεγιστοποίηση μίας συνάρτησης απώλειας. Επί της ουσίας το γενικό πρόβλημα που μελετάμε διατυπώνεται ως ακολούθως. Διαθέτουμε ένα σύνολο n διεργασιών οι οποίες πρόκειται να χρονοδρομολογηθούν διαδοχικά σε μία μηχανή. Κάθε διεργασία j έχει έναν θετικό ακέραιο χρόνο επεξεργασίας p_j και κατά την ολοκλήρωσή της σε μία χρονική στιγμή t, δημιουργείται απώλεια ίση με $l_j(t)$. Θεωρούμε επίσης ότι οι διεργασίες είναι διαθέσιμες τη χρονική στιγμή μηδέν και δεν διακόπτονται κατά την εκτέλεση τους (non-preemptive χρονοδρομολόγηση). Ψάχνουμε γία μία διάταξη χρονοδρομολόγησης των διεργασιών π ($\pi(j) = k$ αν η διεργασία j είναι η k-οστη που εκτελείται) η οποία να ελαχιστοποιεί την ποσότητα

$$\sum_{j=1}^{n} l_j(C_j)$$

όπου

$$C_j = \sum_{k=1}^{\pi(j)} p_{\pi^{-1}(k)}$$

είναι ο χρόνος ολοκλήρωσης της διεργασίας j.

Οι Held και Karp [37] σχεδίασαν ένα δυναμικό πρόγραμμα[1] για το πρόβλημα αυτό, το οποίο

[1]Το ίδιο δυναμικό πρόγραμμα, με μικρές τροποποιήσεις, επιλύει βέλτιστα στον ίδιο χρόνο το πρόβλημα του Περιοδεύοντος Πωλητή (TSP) (βλ. [37]). Μάλιστα, μέχρι σήμερα, αποτελεί τον ταχύτερο ακριβή αλγόριθμο που έχει σχεδιαστεί για το πρόβλημα.

βρίσκει μία βέλτιστη λύση σε χρόνο της τάξης του $O(n2^n)$.

Τα προβλήματα που εξετάζουμε στη συνέχεια αφορούν σε ειδικές περιπτώσεις του παραπάνω προβλήματος οι οποίες μπορούν να επιλυθούν σε ψευδοπολυωνυμικό χρόνο, μέσω της εφαρμογής του δυναμικού προγράμματος (4.1), αντί για εκθετικό χρόνο που χρειάζεται ο αλγόριθμος των Held και Karp. Η εφαρμογή του (4.1) έγκειται στην ακόλουθη βασική παρατήρηση: για κάθε πρόβλημα που μελετάται και για συγκεκριμένες συναρτήσεις απώλειας $l_j(t)$, το σύνολο των διεργασιών μπορεί να διαχωριστεί σε δύο διαφορετικές ομάδες. Οι διεργασίες της μίας ομάδας εκτελούνται βάσει μίας προκαθορισμένης διάταξης, ενώ οι διεργασίες της άλλης εκτελούνται με αυθαίρετο τρόπο είτε πριν, είτε μετά από όλες τις διεργασίες της πρώτης ομάδας. Αντιστοιχίζοντας την κάθε ομάδα διεργασιών σε έναν διαφορετικό τρόπο χρονοδρομολόγησης, μπορούμε να εφαρμόσουμε το δυναμικό πρόγραμμα (4.1).

4.2.2.1 Ελαχιστοίηση του συνολικού βάρους των καθυστερημένων διεργασιών

Στην υποενότητα αυτή μελετάμε την εφαρμογή του δυναμικού προγράμματος (4.1) στο πρόβλημα $1 \parallel \sum_j w_j U_j$. Το αποτέλεσμα αυτό προέρχεται από την εργασία [63]. Υπενθυμίζουμε ότι για μία δεδομένη χρονοδρομολόγηση η τιμή της U_j ισούται με 0, αν η διεργασία j είναι πρόωρη (δηλαδή αν $C_j \leq d_j$) και με 1 αν η j είναι καθυστερημένη.

Θεωρούμε την ακόλουθη συνάρτηση απώλειας.

$$l_j(t) = \begin{cases} 0, & \text{αν } t \leq d_j, \\ w_j, & \text{αλλιώς.} \end{cases}$$

Εύκολα παρατηρούμε ότι για δεδομένη χρονοδρομολόγηση κατά την οποία μία διεργασία j ολοκληρώνει την εκτέλεσή της σε χρόνο C_j, θα ισχύει ότι $l_j(C_j) = w_j U_j$. Επομένως, η συνολική απώλεια $\sum_j l_j(C_j)$, μετά την ολοκλήρωση της χρονοδρομολόγησης, ταυτίζεται με το συνολικό βάρος των καθυστέρημένων διεργασιών. Το ερώτημα που τίθεται είναι το εξής: με ποιά σειρά πρέπει να χρονοδρομολογήσουμε τις διεργασίες έτσι ώστε να ελαχιστοποιήσουμε τη συνολική απώλεια;

Στηριζόμενοι σε προηγούμενα αποτελέσματα των Moore [70] (στην ειδική περίπτωση όπου $w_j = 1$) και Rothkopf [79] (βλ. Ενότητα 4.1) παρατηρούμε ότι σε μία υποτιθέμενη βέλτιστη λύση οι διεργασίες μπορούν να διαχωριστούν σε δύο διαφορετικές ομάδες: η πρώτη περιέχει τις πρόωρες και η δεύτερη τις καθυστερημένες διεργασίες. Αξιοποιώντας τα αποτελέσματα των Jackson και Smith [46, 92] που αναφέραμε στην Ενότητα 4.1, συμπεραίνουμε ότι οι πρόωρες διεργασίες πρέπει να βρίσκονται σε EDD διάταξη. Οι καθυστερημένες διεργασίες χρονοδρομολογούνται, μετά την ολοκλήρωση των πρόωρων, σε αυθαίρετη διάταξη.

Το πρόβλημα λύνεται αν διατάξουμε αρχικά τις διεργασίες βάσει του κανόνα EDD και εφαρμόσουμε το δυναμικό πρόγραμμα (4.1) στην ακόλουθη μορφή.

$$\begin{aligned} f(0,t) &= 0, && (t \geq 0), \\ f(j,t) &= +\infty, && (j = 0,1,\ldots,n; t < 0), \\ f(j,t) &= \min\left\{\begin{array}{l} f(j,t-1), \\ f(j-1,t-p_j), \\ w_j + f(j-1,t) \end{array}\right\}, && (j = 1,\ldots,n; t \geq 0). \end{aligned} \tag{4.2}$$

όπου έχουμε θέσει

$$a_j = p_j, \qquad \alpha_j(t) = 0$$

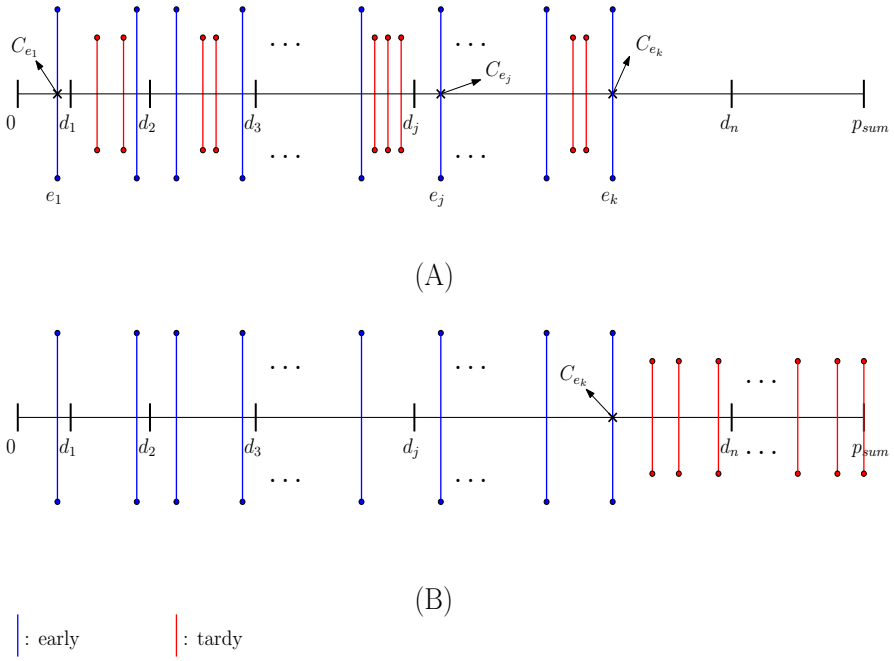
και

$$b_j = 0, \qquad \beta_j(t) = w_j.$$

Η πολυπλοκότητα χρόνου του δυναμικού προγράμματος (4.2), σε αναλογία με το (4.1), θα είναι της τάξης του $O(np_{sum})$, όπου $p_{sum} = \sum_j p_j$.

Είναι σημαντικό να παρατηρήσουμε ότι στη χρονοδρομολόγηση που προκύπτει οι καθυστερημένες διεργασίες έχουν μηδενικό χρόνο επεξεργασίας, σε αντίθεση με τις πρόωρες διεργασίες οι οποίες χρονοδρομολογούνται βάσει του κανόνα EDD διατηρώντας την αρχική τους διάταξη. Στο Σχήμα 4.1 απεικονίζουμε τη μορφή μίας λύσης που επιστρέφει ο αλγόριθμος και τη βέλτιστη λύση που εξάγεται από αυτή. Η επιλογή μας να τοποθετήσουμε τις καθυστερημένες διεργασίες σε χρονικές στιγμές μεταξύ διαδοχικών εκτελέσεων των πρόωρων διεργασιών (βλ. Σχήμα 4.1, (A)) αφορά στη σειρά εκλογής τους από τον αλγόριθμο και δεν υπονοεί ότι χρονοδρομολογούνται τη δεδομένη χρονική στιγμή. Με C_{e_j}, $1 \leq j \leq k$ και $k \leq n$, συμβολίζουμε το χρόνο ολοκλήρωσης της j-οστής πρόωρης διεργασίας. Επίσης, ισχύει ότι $C_{e_j} = C_{e_{j-1}} + p_{e_j}$ και $C_{e_1} = p_{e_1}$, όπου p_{e_j} ο χρόνος επεξεργασίας της πρόωρης διεργασίας e_j, το οποίο υποδεικνύει ότι οι πρόωρες διεργασίες εκτελούνται διαδοχικά. Για να εξάγουμε τη βέλτιστη λύση θα χρειαστεί να εκτελέσουμε τις καθυστερημένες διεργασίες, σε αυθαίρετη διάταξη, αμέσως μετά την ολοκλήρωση και της τελευταίας πρόωρης διεργασίας e_k, τη χρονική στιγμή C_{e_k} (βλ. Σχήμα 4.1, (B)).

Συνάφεια με το πρόβλημα του Σακιδίου. Το πρόβλημα του Σακιδίου αποτελεί ένα από τα περισσότερο μελετημένα προβλήματα στην ευρύτερη περιοχή της Συνδυαστικής Βελτιστοποίησης. Στην είσοδο του προβλήματος δίνεται ένα σύνολο n αντικειμένων και ένας θετικός ακέραιος W. Για κάθε αντικείμενο j δίνονται μία θετική ακέραια αξία p_j καθώς και ένα θετικό ακέραιο βάρος



Σχήμα 4.1: Μία λύση του δυναμικού προγράμματος (4.2) και η αντίστοιχη βέλτιστη.

w_j. Ο στόχος είναι να βρεθεί ένα υποσύνολο αντικειμένων A, $|A| \leq n$, με τη μέγιστη δυνατή αξία, τέτοιο ώστε να ισχύει ότι $\sum_{j \in A} w_j \leq W$. Ένα ακέραιο δυναμικό πρόγραμμα για το πρόβλημα του Σακιδίου διαμορφώνεται ως ακολούθως.

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{n} w_j x_j \\ \text{subject to} \quad & p_1 x_1 + p_2 x_2 + \ldots + p_n x_n \leq W \\ & x_j = \{0, 1\}, \qquad (j = 1, \ldots, n). \end{aligned}$$

Το πρόβλημα επιλύεται βέλτιστα σε ψευδοπολυωνυμικό χρόνο της τάξης του $O(nW)$, εφαρμόζοντας την ακόλουθη γνωστή αναδρομή.

$$\begin{aligned} f(j,t) &= \max \left\{ \begin{array}{l} f(j-1,t), \\ \\ w_j + f(j-1, t-p_j) \end{array} \right\}, \quad (t \leq W), \\ f(j,t) &= f(j,W), \qquad\qquad\qquad\qquad (t > W). \end{aligned} \tag{4.3}$$

Όπως έχουμε ήδη αναφέρει, είναι εύκολο να παρατηρήσει κανείς ομοιότητες ανάμεσα στα δυναμικά προγράμματα (4.3) και (4.1). Ακόμα πιο εύκολο δε, είναι να παρατηρήσουμε την ομοιότητα του (4.3) με ένα δυναμικό πρόγραμμα, ισοδύναμο του (4.1), το οποίο επιλύει μία ισοδύναμη εκδοχή του προβλήματος $1 \parallel \sum_{j=1}^{n} w_j U_j$.

Σε μία δεδομένη χρονοδρομολόγηση, παρατηρούμε ότι το συνολικό βάρος των καθυστερημένων διεργασιών θα ισούται με τη διαφορά του συνολικού βάρους όλων των διεργασιών από το

αντίστοιχο βάρος των πρόωρων διεργασιών. Έτσι, θα μπορούσαμε να υπολογίσουμε μία βέλτιστη λύση του προβλήματος $1 \parallel \textit{maximize} \sum_{j=1}^{n} w_j U_j$, αν εναλλακτικά επιλύαμε το πρόβλημα μεγιστοποίησης του συνολικού βάρους των πρόωρων διεργασιών. Θυμίζουμε ότι το πρόβλημα αυτό για περιβάλλον Μοναδικής Μηχανής συμβολίζεται με $1 \parallel \sum_{j=1}^{n} w_j U'_j$, όπου για δεδομένη χρονοδρομολόγηση το $U'_j = 1 - U_j$ (βλ. Ενότητα 1.1). Ένα ακέραιο γραμμικό πρόγραμμα για το συγκεκριμένο πρόβλημα διαμορφώνεται ως ακολούθως.

$$
\begin{aligned}
\text{maximize} \quad & \textstyle\sum_{j=1}^{n} w_j U'_j \\
\text{subject to} \quad & p_1 U'_1 && \le d_1 \\
& p_1 U'_1 + p_2 U'_2 && \le d_2 \\
& p_1 U'_1 + p_2 U'_2 + p_3 U'_3 && \le d_3 \\
& \qquad \vdots \\
& p_1 U'_1 + p_2 U'_2 + \ldots + p_n U'_n && \le d_n \\
& U'_j = \begin{cases} 0, & \text{αν η } j \text{ είναι καθυστερημένη,} \\ 1, & \text{αλλιώς.} \end{cases}
\end{aligned}
$$

Θεωρούμε ότι αρχικά οι διεργασίες διατάσσονται βάσει του κανόνα EDD. Κάθε περιορισμός υποδεικνύει ότι κατά τη χρονοδρομολόγηση της j-οστής στη σειρά διεργασίας, ο συνολικός χρόνος επεξεργασίας των πρόωρων διεργασιών δεν πρέπει να υπερβαίνει την προθεσμία d_j αυτής.

Η διατύπωση του προβλήματος ελαχιστοποίησης του συνολικού βάρος των καθυστερημένων διεργασιών, καθώς και η επίλυση του με την παραπάνω μέθοδο παρατηρήθηκαν από τον Rothkopf [79].

Το ισοδύναμο του δυναμικού προγράμματος (4.1) για το πρόβλημα $1 \parallel \textit{maximize} \sum_{j=1}^{n} w_j U'_j$ έχει ως εξής.

$$
\begin{aligned}
f(j,t) &= \max \left\{ \begin{array}{l} f(j-1,t), \\ \\ w_j + f(j-1, t-p_j) \end{array} \right\}, && (t \le d_j), \\
f(j,t) &= f(j,d_j), && (t > d_j).
\end{aligned} \tag{4.4}
$$

Παρατηρούμε ότι, σε αντίθεση με το (4.1), η ποσότητα $f(j,t-1)$ δεν συμμετέχει στους υπολογισμούς. Ο λόγος είναι ότι όταν το $t \le d_j$, αν η διεργασία j χρονοδρομολογηθεί ως πρόωρη η τιμή $f(j,t-1)$ θα ταυτίζεται με την $w_j + f(j-1, t-p_j)$, ενώ αν η j θεωρηθεί ως καθυστερημένη η $f(j,t-1)$ θα ισούται με $f(j-1,t)$. Επομένως, σε κάθε περίπτωση μπορεί να αντικατασταθεί με μία εκ των δύο εναλλακτικών τιμών.

Η μόνη βασική διαφορά ανάμεσα στα δυναμικά προγράμματα (4.4) και (4.3) είναι ότι η αρχική διάταξη των διεργασιών μπορεί να γίνει με αυθαίρετο τρόπο για το δεύτερο. Ο λόγος γι' αυτό είναι σε μία υποτιθέμενη βέλτιστη λύση του προβλήματος του Σακιδίου, με οποιονδήποτε τρόπο και αν διατάξουμε τα αντικείμενα που έχουν επιλεγεί, η τιμή της λύσης θα παραμείνει η ίδια.

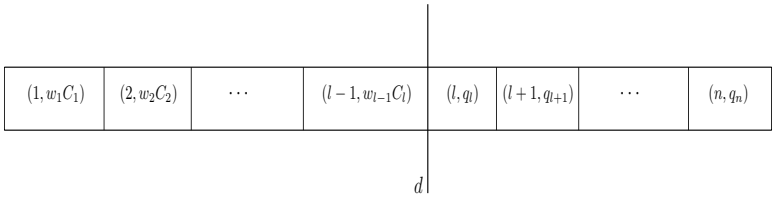
4.2.2.2 Μεγιστοποίηση της συνολικής βεβαρημένης πρόωρης ολοκλήρωσης

Το αποτέλεσμα που παρουσιάζεται στη συνέχεια προέρχεται από την εργασία [63] των Lawler και Moore. Υποθέτουμε αρχικά, ότι οι n διεργασίες που διαθέτουμε πρόκειται να χρονοδρομολογηθούν σε μία μηχανή, με απώλειες που δίνονται από την ακόλουθη συνάρτηση.

$$l_j(t) = \begin{cases} w_j t, & \text{αν } t \leq d, \\ q_j, & \text{αλλιώς,} \end{cases}$$

όπου d είναι μία κοινή προθεσμία για όλες τις διεργασίες και q_j είναι μία προκαθορισμένη θετική σταθερά για κάθε διεργασία j. Το ερώτημα που τίθεται είναι με ποιά σειρά πρέπει να χρονοδρομολογήσουμε τις διεργασίες έτσι ώστε να ελαχιστοποιήσουμε τη συνολική απώλεια;.

Για τη λύση του προβλήματος απαιτείται ο διαχωρισμός των διεργασιών σε δύο διαφορετικές ομάδες. Η πρώτη ομάδα περιέχει τις διεργασίες που ολοκληρώνουν την εκτέλεσή τους πριν την προθεσμία d και η δεύτερη τις καθυστερημένες διεργασίες. Σύμφωνα με τα αποτελέσματα των Smith [92] και McNaughton [69] (βλ. Ενότητα 4.1), σε μία υποτιθέμενη βέλτιστη χρονοδρομολόγηση, οι πρόωρες διεργασίες θα είναι διατεταγμένες βάσει του κανόνα του Smith (ισοδύναμα, του κανόνα WSPT) και θα προηγούνται όλων των καθυστερημένων διεργασιών, οι οποίες θα βρίσκονται σε αυθαίρετη διάταξη. Μία βέλτιστη χρονοδρομολόγηση για το πρόβλημα απεικονίζεται στο Σχήμα 4.2. Κάθε ζεύγος της μορφής $(j, w_j C_j)$ υποδεικνύει ότι η j-οστή διεργασία χρονοδρομολογείται ως πρόωρη και προσδίδει συνολική απώλεια ίση με $w_j C_j$, όπου C_j ο χρόνος ολοκλήρωσης της j. Αντίθετα, κάθε ζεύγος της μορφής (j, q_j) υποδεικνύει ότι η j-οστή διεργασία χρονοδρομολογείται ως καθυστερημένη προσδίδοντας συνολική απώλεια ίση με q_j.



Σχήμα 4.2: Μία διάταξη χρονοδρομολόγησης που ελαχιστοποιεί τη συνολική απώλεια $\sum_{j=1}^{n} l_j(C_j)$.

Το παραπάνω πρόβλημα επιλύεται βέλτιστα σε ψευδοπολυωνυμικό χρόνο της τάξης του $O(nd)$, αν εφαρμόσουμε το δυναμικό πρόγραμμα (4.1) θέτοντας

$$a_j = p_j, \qquad \alpha_j(t) = w_j t$$

και

$$b_j = 0, \qquad \beta_j(t) = q_j.$$

Στη συνέχεια, για δεδομένες χρονοδρομολογήσεις, θα συσχετίσουμε την παραπάνω συνάρτηση απώλειας με τη συνάρτηση της βεβαρημένης πρόωρης ολοκλήρωσης. Ως αποτέλεσμα, η εφαρμογή του (4.1) στην παραπάνω μορφή, θα μας δώσει μία βέλτιστη λύση ψευδοπολυωνυμικού χρόνου για το πρόβλημα $1 \parallel \mathit{maximize} \sum_j w_j E_j$. Θυμίζουμε ότι η βεβαρημένη πρόωρη ολοκλήρωση κάθε διεργασίας j, σε μία δεδομένη χρονοδρομολόγηση, ισούται με $w_j E_j = w_j \max\{d_j - C_j, 0\}$.

Υποθέτουμε ότι κατά τη χρονοδρομολόγηση των διεργασιών ισχύει ότι $w_j d \geq q_j$, για κάθε διεργασία j. Τα δύο λήμματα που αποδεικνύονται ακολούθως έξασφαλίζουν τη ζητούμενη συσχέτιση. Ο όρος ελαχιστική (minimal) διάταξη (αντίστοιχα μεγιστική (maximal) διάταξη) ερμηνεύεται ως μία διάταξη στην οποία οποιαδήποτε αντιμετάθεση μεταξύ των διεργασιών της δεν μπορεί να μειώσει (αυξήσει) περαιτέρω την υπάρχουσα συνολική απώλεια.

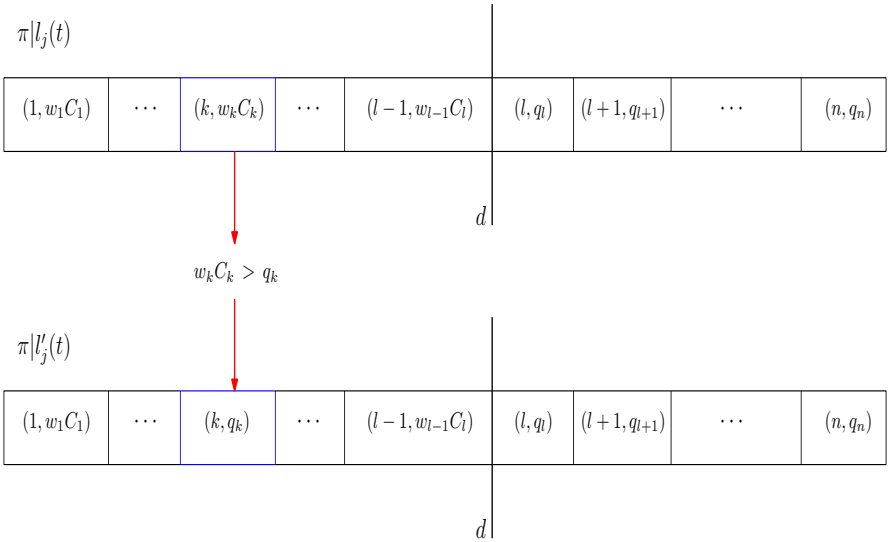
Λήμμα 4.1 *Αν μία διάταξη χρονοδρομολόγησης είναι ελαχιστική για τις συναρτήσεις απώλειας $l_j(t)$, τότε είναι ελαχιστική και για συναρτήσεις απώλειας της μορφής*

$$l'_j(t) = \min\{w_j t, q_j\}.$$

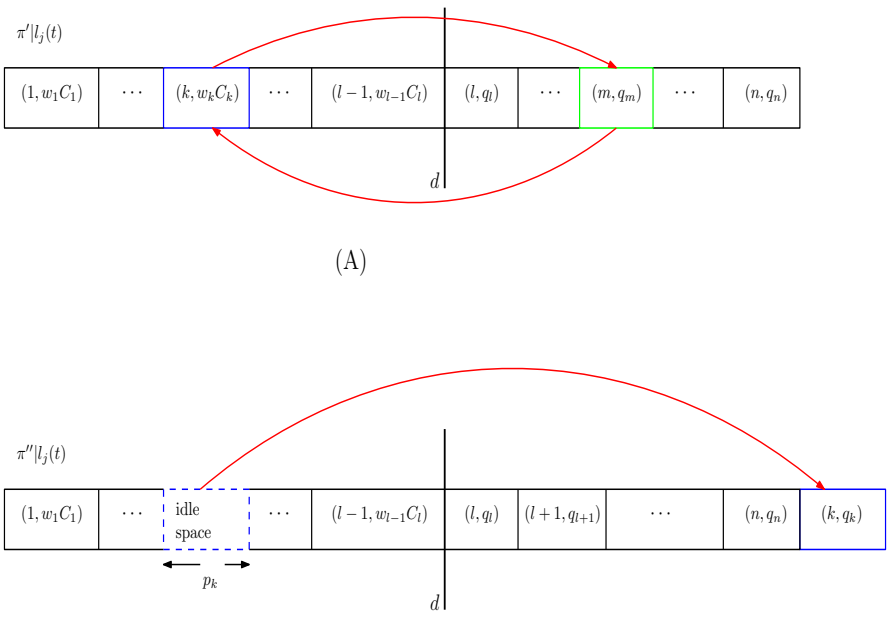
Απόδειξη. Θεωρούμε μία ελαχιστική διάταξη χρονοδρομολόγησης π για τις συναρτήσεις απώλειας $l_j(t)$ (βλ. Σχήμα 4.3, διάταξη $\pi|l_j(t)$). Θεωρούμε επιπλέον, την ίδια διάταξη χρονοδρομολόγησης για τις συναρτήσεις απώλειας $l'_j(t)$. Λόγω της σχέσης $w_j d \geq q_j$, κάθε διεργασία j που ολοκληρώνεται μετά το d, θα προσδίδει την ίδια ακριβώς απώλεια q_j. Θα δείξουμε ότι και για τις συναρτήσεις $l'_j(t)$, η π είναι ελαχιστική.

Έστω, προς άτοπο, ότι η διάταξη $\pi|l_j(t)$ δεν είναι ελαχιστική για τις συναρτήσεις $l'_j(t)$. Τότε θα υπάρχει τουλάχιστον μία διεργασία k στην $\pi|l_j(t)$, η οποία ολοκληρώνεται πριν τη χρονική στιγμή d, $C_k \leq d$ και για την οποία ισχύει ότι $w_k C_k > q_k$ (βλ. Σχήμα 4.3). Οπότε η συνολική απώλεια της $\pi|l'_j(t)$, θα είναι $\sum_{j=1}^{n} l'_j(C_j) \leq \sum_{j=1}^{n} l_j(C_j) - (w_k C_k - q_k) < \sum_{j=1}^{n} l_j(C_j)$.

Παρατηρούμε όμως, ότι η ύπαρξη μίας τέτοιας διεργασίας k στην $\pi|l_j(t)$, οδηγεί στη δημιουργία μίας νέας "καλύτερης" διάταξης. Συγκεκριμένα, παρατηρούμε ότι μπορούμε να μειώσουμε την τιμή της συνολικής απώλειας της $\pi|l_j(t)$, $\sum_{j=1}^{n} l_j(C_j)$, αν μεταθέσουμε την διεργασία k, ώστε αυτή να



Σχήμα 4.3: Η διάταξη χρονοδρομολόγησης $\pi|l_j(t)$ είναι ελαχιστική για τις συναρτήσεις απώλειας $l_j(t)$. Η ύπαρξη μίας τουλάχιστον διεργασίας k - εντός του μπλέ περιγράμματος - μειώνει την συνολική απώλεια, στην αντίστοιχη διάταξη ως προς τις συναρτήσεις $l'_j(t)$. Η τελευταία διάταξη συμβολίζεται με $\pi|l'_j(t)$.



Σχήμα 4.4: Δύο ενδεχόμενες διατάξεις χρονοδρομολόγησης που προκύπτουν από την $\pi|l_j(t)$, όταν υπάρχει μία διεργασία k η οποία ολοκληρώνει την εκτέλεσή της πριν τη χρονική στιγμή d, σημειώνοντας απώλεια $w_kC_k > q_k$.

ολοκληρώνει την εκτέλεσή της μετά τη χρονική στιγμή d, όπου θα εμφανίζει μικρότερη απώλεια $q_k < w_k C_k$. Αυτό μπορεί να γίνει με έναν από τους ακόλουθους δύο τρόπους.

1. Έστω ότι υπάρχει μία διεργασία m που ολοκληρώνεται μετά το d με απώλεια q_m - από υπόθεση έχουμε ότι $w_m C_m \geq q_m$ - για την οποία ισχύει ότι $w_m t \leq q_m$ για $t < d$. Τότε, αν αντιμεταθέσουμε τις διεργασίες k και m (βλ. Σχήμα 4.4, (Α)), η νέα διάταξη χρονοδρομολόγησης $\pi' | l_j(t)$ που θα προκύψει, θα έχει συνολική απώλεια ίση με $\sum_{j=1}^{n} l_j(C_j) - (w_k C_k - q_k) - (q_m - w_m C'_m)$, όπου ο νέος χρόνος ολοκλήρωσης της διεργασίας m, C'_m, θα είναι $C'_m \leq d$. Η απώλεια αυτή είναι γνήσια μικρότερη από τη συνολική απώλεια της $\pi | l_j(t)$, $\sum_{j=1}^{n} l_j(C_j)$.

2. Αν δεν υπάρχει αντίστοιχη της l διεργασία, που να μπορεί να αντιμετατεθεί με την k, τότε δημιουργούμε μία νέα διάταξη $\pi'' | l_j(t)$, όπως φαίνεται στο Σχήμα 4.4, (Β): η διεργασία k μετατίθεται στο τέλος της $\pi | l'_j(t)$, οπότε προκαλεί μικρότερη απώλεια q_k. Στο νεκρό διάστημα p_k δεν εκτελείται καμία διεργασία με αποτέλεσμα οι χρόνοι ολοκλήρωσης των υπολοίπων διεργασιών να διατηρούνται ως είχαν και κατά την $\pi | l_j(t)$. Έτσι, η μετάθεση της διεργασίας k αυξάνει (μειώνει) κατά μία τις καθυστερημένες (πρόωρες) διεργασίες και κατ΄ επέκταση μειώνει τη συνολική απώλεια κατά $w_k C_k - q_k$.

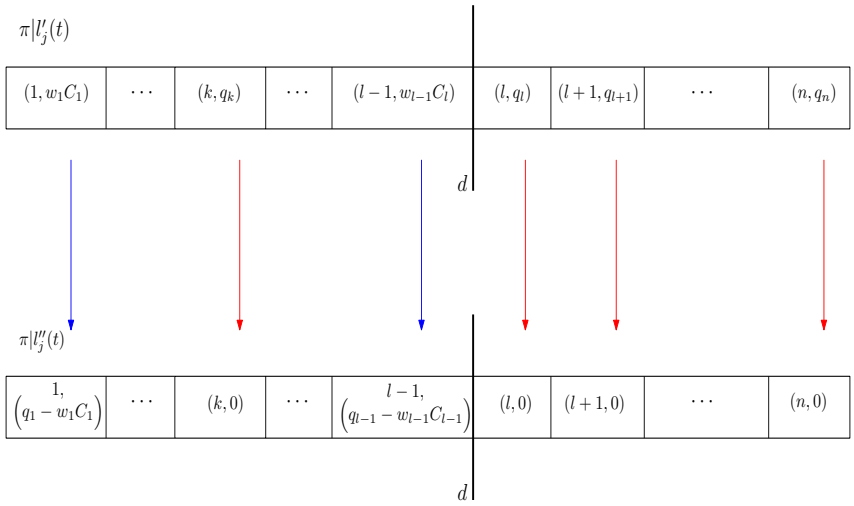
Και στις δύο περιπτώσεις, προκύπτει μία νέα διάταξη χρονοδρομολόγησης με γνήσια μικρότερη συνολική απώλεια από αυτή της $\pi | l_j(t)$. Βάσει της υπόθεσής μας, ότι δηλαδή η $\pi | l_j(t)$ είναι ελαχιστική, έχουμε οδηγηθεί σε άτοπο. Επομένως, η διάταξη $\pi | l_j(t)$ θα είναι ελαχιστική και για τις συναρτήσεις $l'_j(t)$. □

Λήμμα 4.2 *Για κάθε διεργασία j, ορίζουμε μία προθεσμία d_j που ισούται με $d_j = q_j / w_j$. Αν μία διάταξη χρονοδρομολόγησης είναι ελαχιστική για τις συναρτήσεις απώλειας $l'_j(t)$, τότε είναι μεγιστική (maximal) για συναρτήσεις απώλειας της μορφής*

$$l''_j(t) = w_j \max\{d_j - t, 0\}.$$

Απόδειξη. Αρχικά παρατηρούμε ότι κάθε συνάρτηση $l''_j(t)$ γράφεται ισοδύναμα ως $\max\{w_j(d_j - t), 0\}$. Αντικαθιστώντας το d_j με την τιμή που ορίσαμε, $d_j = q_j / w_j$, η $l''_j(t)$ γίνεται ίση με $\max\{q_j - w_j t, 0\}$.

Θεωρούμε τώρα μία διάταξη χρονοδρομολόγησης π, η οποία είναι ελαχιστική για τις συναρτήσεις $l'_j(t)$ (βλ. Σχήμα 4.5, διάταξη $\pi | l'_j(t)$). Οι αντίστοιχες τιμές απώλειας της π, ως προς τις συναρτήσεις $l''_j(t)$, διαμορφώνονται όπως φαίνεται στο ίδιο σχήμα (βλ. διάταξη $\pi | l''_j(t)$). Είναι σημαντικό να παρατηρήσουμε ότι για κάθε διεργασία j που σημειώνει μηδενική απώλεια στην $\pi | l''_j(t)$,



Σχήμα 4.5: Μία ελαχιστική διάταξη χρονοδρομολόγησης $\pi|l'_j(t)$ για τις συναρτήσεις $l'_j(t)$ και η αντίστοιχη διάταξη $\pi|l''_j(t)$, για τις συναρτήσεις $l''_j(t)$.

$l''_j(C_j) = 0$, ισχύει ότι $w_jC_j \geq q_j$· για τις διεργασίες που ολοκληρώνονται μετά το d, λόγω της αρχικής υπόθεσης ότι $w_jd \geq q_j$, αυτό είναι προφανές. Θεωρούμε, προς άτοπο, ότι η $\pi|l'_j(t)$ δεν είναι μεγιστική για τις συναρτήσεις $l''_j(t)$. Αυτό προϋποθέτει ότι τουλάχιστον μία από τις διεργασίες, έστω η $l-1$, που σημειώνουν θετική απώλεια, $q_{l-1} - w_{l-1}C_{l-1} > 0$, θα μπορούσε να είχε χρονοδρομολογηθεί έτσι ώστε να ολοκληρώσει την εκτέλεσή της σε συντομότερο χρόνο, $C'_{l-1} < C_{l-1}$. Επιπλέον, αν οποιαδήποτε διεργασία j εμφανίζει θετική απώλεια $l''_j(C_j) = q_j - w_jC_j > 0$ στην $\pi|l''_j(t)$, τότε για τη διάταξη $\pi|l'_j(t)$ θα ισχύει ότι $l'_j(C_j) = w_jC_j$. Έτσι, ο μόνος τρόπος για να μειωθεί ο χρόνος ολοκλήρωσης αυτής, είναι να αντιμετατεθεί με κάποια διεργασία που προηγείται στη διάταξη $\pi|l''_j(t)$. Ομοίως με το Λήμμα 4.1, διαπιστώνουμε ότι η ύπαρξη τέτοιων διεργασιών, δημιουργεί νέες διατάξεις $\pi|l'_j(t)$ μικρότερης συνολικής απώλειας και κατά συνέπεια οδηγεί σε άτοπο. □

Η απώλεια $l''_j(t)$ σε μία δεδομένη χρονοδρομολόγηση όπου η διεργασία j ολοκληρώνει την εκτέλεσή της κατά τη χρονική στιγμή $t = C_j$, ταυτίζεται με την βεβαρημένη πρόωρη ολοκλήρωση της j, $w_jE_j = w_j \max\{d_j - C_j, 0\}$. Επομένως, η αντικειμενική συνάρτηση $\sum_j w_jE_j$, σε μία δεδομένη χρονοδρομολόγηση, ταυτίζεται με την συνάρτηση $\sum_j l''_j(C_j)$. Το ακόλουθο θεώρημα συμπυκνώνει τα παραπάνω αποτελέσματα.

Θεώρημα 4.1 *Θεωρούμε για κάθε διεργασία j ότι $q_j = d_jw_j$. Έστω π η βέλτιστη διάταξη χρονοδρομολόγησης που υπολογίζεται από την εφαρμογή του δυναμικού προγράμματος* (4.1) *θέτοντας όπου $a_j = p_j$, $\alpha_j(t) = w_jt$, $b_j = 0$ και $\beta_j(t) = q_j$. Από τα Λήμματα* 4.1 *και* 4.2 *προκύπτει ότι η π μεγιστοποιεί την συνάρτηση $\sum_j l''_j(C_j)$ και επομένως αποτελεί μία βέλτιστη διάταξη χρονοδρομολόγησης για το πρόβλημα μεγιστοποίησης της συνολικής βεβαρημένης πρόωρης ολοκλήρωσης.*

Ο συνολικός χρόνος για την εύρεση μίας βέλτιστης λύσης στο πρόβλημα $1 \parallel \textit{maximize} \ \sum_j w_j E_j$, ταυτίζεται με το χρόνο εκτέλεσης του δυναμικού προγράμματος (4.1) και είναι της τάξης του $O(nd)$.

Με όμοιο τρόπο, μπορούμε να αποδείξουμε την ισχύ του Λήμματος 4.2 και για συναρτήσεις $l''(j)(t)$ της μορφής $w_j \max\{t - d_j, 0\}$. Κατά συνέπεια, η παραπάνω διαδικασία οδηγεί σε έναν ψευδοπολυωνυμικό αλγόριθμο για το πρόβλημα μεγιστοποίησης της συνολικής βεβαρημένης καθυστέρησης. Όπως θα δούμε στην επόμενη ενότητα, το πρόβλημα ελαχιστοποίησης της συνολικής βεβαρημένης καθυστέρησης είναι κατά πολύ πιο δύσκολο, ακόμα και στην περίπτωση όπου οι διεργασίες διαθέτουν μία κοινή προθεσμία.

4.2.2.3 Ελαχιστοποίηση της συνολικής βεβαρημένης καθυστέρησης δεδομένης μίας κοινής προθεσμίας

Στη συνέχεια μελετάμε την εφαρμογή του δυναμικού προγράμματος (4.1) στο πρόβλημα $1 \mid d_j = d \mid \sum_j w_j T_j$ (βλ. [63]). Θυμίζουμε ότι η καθυστέρηση μίας διεργασίας j, σε μία δεδομένη χρονοδρομολόγηση, ισούται με $T_j = \max\{C_j - d, 0\}$, όπου d είναι μία κοινή προθεσμία των διεργασιών.

Θεωρούμε την ακόλουθη συνάρτηση απώλειας

$$l_j(t) = \max\{w_j(t - d), 0\}.$$

Η $l_j(t)$, για δεδομένη χρονοδρομολόγηση, αναπαριστά τη βεβαρημένη καθυστέρηση της διεργασίας j κατά τη χρονική στιγμή t της χρονοδρομολόγησης. Αν το t ταυτίζεται με το χρόνο ολοκλήρωσης της j, τότε η $l_j(t) = w_j T_j$. Επομένως, για να βρούμε μία βέλτιστη διάταξη χρονοδρομολόγησης για το πρόβλημα $1 \mid d_j = d \mid \sum_j w_j T_j$, αρκεί να βρούμε μία διάταξη χρονοδρομολόγησης που να ελαχιστοποιεί τη συνολική απώλεια $\sum_{j=1}^{n} l_j(C_j)$.

Λόγω της μορφής που έχει μία υποτιθέμενη βέλτιστη διάταξης χρονοδρομολόγησης (βλ. Σχήμα 4.6), παρατηρούμε ότι για τη λύση του προβλήματος απαιτείται ο διαχωρισμός του συνόλου των διεργασιών σε τρείς διαφορετικές ομάδες. Η πρώτη ομάδα περιέχει τις πρόωρες διεργασίες οι οποίες βρίσκονται σε αυθαίρετη διάταξη. Η δεύτερη ομάδα περιέχει τις καθυστερημένες διεργασίες οι οποίες ακολουθούν των πρόωρων διεργασιών και επιπλέον βρίσκονται διατεταγμένες βάσει του κανόνα WSPT (βλ. Ενότητα 4.1). Η τρίτη ομάδα αποτελείται από μία και μόναδική διεργασία η οποία αρχίζει να εκτελείται πριν τη χρονική στιγμή d, μετά την ολοκλήρωση και της τελευταίας πρόωρης διεργασίας της πρώτης ομάδας και ολοκληρώνει την εκτέλεσή της, είτε ακριβώς, είτε μετά τη χρονική στιγμή d (δηλαδή, είτε ως πρόωρη, είτε ως καθυστερημένη αντίστοιχα). Προσφάτως έχει επικρατήσει ο όρος straddling για τη διεργασία αυτή (βλ. [53]).

Είναι εμφανές ότι δεν υπάρχει κάποιος τρόπος ώστε να γνωρίζουμε εκ των προτέρων, αν

η straddling διεργασία θα χρονοδρομολογηθεί στην πρώτη ομάδα ως πρόωρη ή στην δεύτερη αντίστοιχα ως καθυστερημένη. Κατά συνέπεια, φαίνεται σκόπιμο να επιλύσουμε το πρόβλημα χωριστά για κάθε μία από τις n ενδεχόμενες straddling διεργασίες και να διαλέξουμε την καλύτερη λύση.

Αλγόριθμος 6 Ένας βέλτιστος αλγόριθμος ψευδοπολυωνυμικού χρόνου για το πρόβλημα $1 \mid d_j = d \mid \sum_j w_j T_j$.

---

1. Διάταξε τις διεργασίες σε φθίνουσα διάταξη ως προς το λόγο τους w_j/p_j: $w_1/p_1 \leq w_2/p_2 \leq \ldots \leq w_n/p_n$. Έστω π η διάταξη που προκύπτει.

2. **Για** $k = 1$ έως n

   2.1. Εκτέλεσε το δυναμικό πρόγραμμα (4.1) για το υποσύνολο διεργασιών $J - \{k\}$, θέτοντας

   $$a_j = p_j, \qquad \alpha_j(t) = 0,$$
   $$b_j = 0, \qquad \beta_j(t) = w_j(t + \mathcal{A}_j).$$

   Έστω $f^{(k)}(n,t)$ η λύση που προκύπτει.

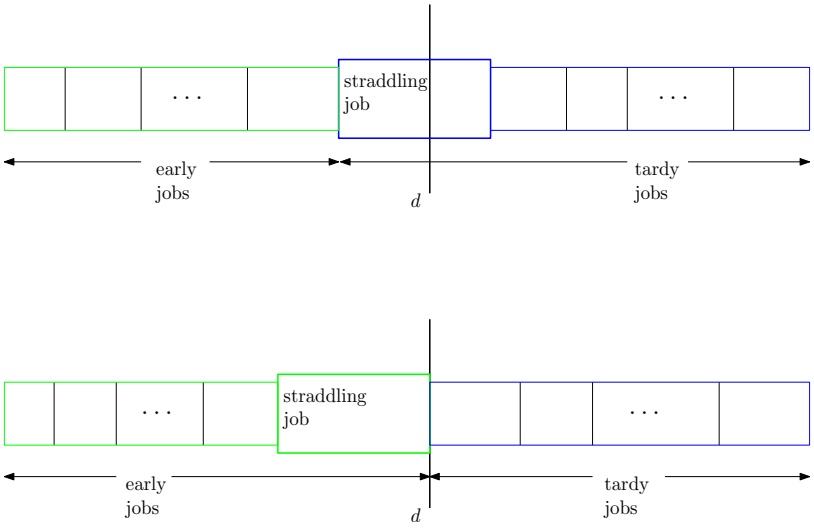
3. Υπολόγισε τις τιμές των k και t για τις οποίες η

   $$f^{(k)}(n,t) + w_k(t + p_k), \qquad (d - p_k \leq t < d),$$

   γίνεται ελάχιστη.

Πριν προχωρήσουμε στην παρουσίαση ενός βέλτιστου ψευδοπολυωνυμικού αλγορίθμου για το παραπάνω πρόβλημα, υπενθυμίζουμε (βλ. Ενότητα 4.1) ότι σε μία δεδομένη χρονοδρομολόγηση η συνολική βεβαρημένη καθυστέρηση ισούται με $\sum_j w_j T_j = \sum_{j' \in K} w_{j'} C_{j'} - d \sum_{j' \in K} w_{j'}$, όπου K το σύνολο των καθυστερημένων διεργασιών. Επομένως, για την εύρεση μίας βέλτιστης λύσης, αρκεί να ελαχιστοποιήσουμε το μέσο βαβαρημένο χρόνο ολοκλήρωσης των καθυστερημένων διεργασιών, $\sum_{j'} w_{j'} C_{j'}$. Ο Αλγόριθμος 6 βρίσκει μία διάταξη χρονοδρομολόγησης που ελαχιστοποιεί την ποσότητα αυτή.

Με $J = \{1, 2, \ldots, n\}$, συμβολίζουμε το σύνολο των n διατεταγμένων διεργασιών. Σε κάθε επανάληψη k του Βήματος 2, η διεργασία k λαμβάνεται ως straddling και το δυναμικό πρόγραμμα

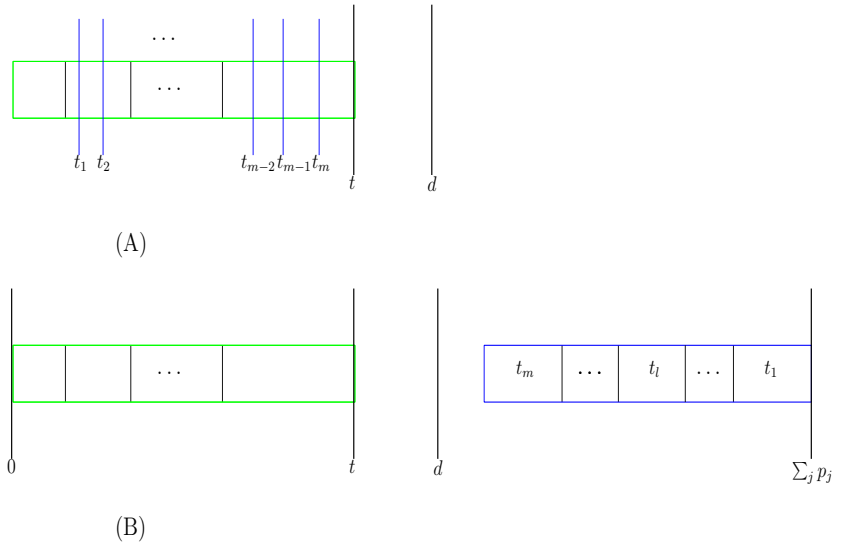


Σχήμα 4.6: Δύο βέλτιστες διατάξεις χρονοδρομολόγησης για το πρόβλημα $1\,|d_j = d|\sum_j w_j T_j$. Στην πρώτη η straddling διεργασία χρονοδρομολογείται ως καθυστερημένη και στη δεύτερη ως πρόωρη. Όλες οι καθυστερημένες διεργασίες βρίσκονται σε WSPT διάταξη.

βρίσκει μία βέλτιστη χρονοδρομολόγηση για τις υπόλοιπες $n-1$ διεργασίες του συνόλου $J-\{k\}$. Η συνολική απώλεια αυτής συμβολίζεται με $f^{(k)}(n,t)$ και αντιστοιχεί στο μέσο βεβαρημένο χρόνο ολοκλήρωσης των καθυστερημένων διεργασιών του συνόλου $J-\{k\}$. Με $t+\mathcal{A}_j$ συμβολίζουμε το χρόνο ολοκλήρωσης της τρέχουσας διεργασίας j, αν αυτή χρονοδρομολογηθεί ως καθυστερημένη, όπου $\mathcal{A}_j$ είναι μία σταθερή ποσότητα διαφορετική για κάθε καθυστερημένη διεργασία j. Στις επόμενες παραγράφους δίνουμε τις απαραίτητες διευκρινίσεις σχετικά με την εκτέλεση του δυναμικού προγράμματος στο Βήμα 2 του αλγορίθμου.

Το Σχήμα 4.7, (Α) αναπαριστά μία διάταξη χρονοδρομολόγησης που προκύπτει μετά την εκτέλεση του δυναμικού προγράμματος σε μία επανάληψη του Βήματος 2. Όπως και στην περίπτωση του δυναμικού προγράμματος (4.2) για το προβλήμα $1\,\|\sum_{j=1}^{n} w_j U_j$, οι καθυστερημένες διεργασίες συμμετέχουν στη χρονοδρομολόγηση με μηδενικό χρόνο επεξεργασίας ($b_j = 0$). Για το σκοπό αυτό έχουν τοποθετηθεί ανάμεσα σε διαδοχικές εκτελέσεις των πρόωρων διεργασιών με τη σειρά που επιλέγονται κατά την εκτέλεση του δυναμικού προγράμματος, χωρίς να χρονοδρομολογούνται. Με t_l, όπου $l \leq n-1$, συμβολίζουμε την l-οστή καθυστερημένη διεργασία που επιλέγεται από το δυναμικό πρόγραμμα. Οι πρόωρες διεργασίες ολοκληρώνουν την εκτέλεση σε χρόνο t, $d - p_k \leq t < d$, όπου p_k ο χρόνος επεξεργασίας της straddling διεργασίας k, σημειώνοντας μηδενική απώλεια ($\alpha_j(t) = 0$). Ο συνολικός χρόνος εκτέλεσης του δυναμικού προγράμματος θα είναι της τάξης του $O((n-1)t) = O(nd)$. Σε αντίθεση με το δυναμικό πρόγραμμα (4.2), για να ελαχιστοποίησουμε τη συνολική απώλεια, θα πρέπει οι καθυστερημένες διεργασίες να εκτελεστούν με βάση τον κανόνα WSPT και όχι σε αυθαίρετη διάταξη. Παρατηρούμε όμως, ότι η αρχική διάταξη χρονοδρομολό-

γησης π είναι αντίστροφη της διάταξης WSPT. Οπότε και η σειρά με την οποία επιλέγονται οι καθυστερημένες διεργασίες είναι αντίστροφη της WSPT διάταξης. Για να χρονοδρομολογήσουμε τις καθυστερημένες διεργασίες βάσει της WSPT διάταξης, κατά την εκτέλεση του δυναμικού προγράμματος χρησιμοποιούμε την ποσότητα $\mathcal{A}_j$, η οποία καθορίζει το χρόνο ολοκλήρωσης της τρέχουσας διεργασίας j όταν αυτή χρονοδρομολογηθεί ως καθυστερημένη και κατ' επέκταση την απώλεια της j.



Σχήμα 4.7: Η διάταξη χρονοδρομολόγησης ενός υποσυνόλου $n-1$ διεργασιών σε μία επανάληψη του Βήματος 2 του Αλγορίθμου (6). Ο χρόνος t είναι ο χρόνος ολοκλήρωσης των πρόωρων διεργασιών, ο οποίος συμπίπτει με το συνολικό χρόνο εκτέλεσης του δυναμικού προγράμματος, ενώ οι χρόνοι $0, \sum_j p_j$ υποδεικνύουν το χρονικό ορίζοντας της χρονοδρομολόγησης. Οι καθυστερημένες διεργασίες, $(t_1, \ldots, t_m)$, χρονοδρομολογούνται τελικά σε WSPT διάταξη, αντίστροφη από αυτή που επιλέγονται από το δυναμικό πρόγραμμα.

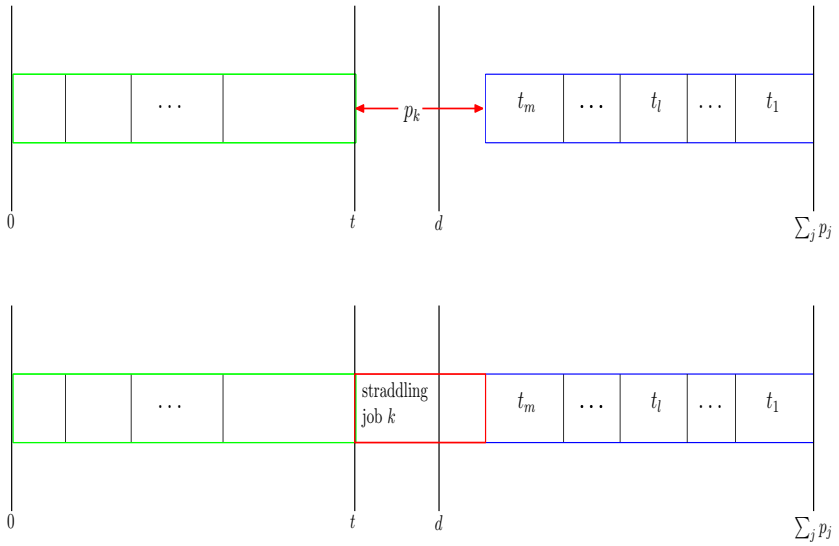
Θεωρούμε χ.β.τ.γ[2], ότι η j-οστή διεργασία της αρχικής διάταξης π είναι η πρώτη που επιλέγεται να χρονοδρομολογηθεί ως καθυστερημένη στην j-οστή επανάληψη του δυναμικού προγράμματος (βλ. διεργασία t_1 στο Σχήμα 4.7, (A)). Αν θέσουμε την ποσότητα $\mathcal{A}_j$ ίση με $\mathcal{A}_j = \sum_{i=1}^{n} p_i - \sum_{i=1}^{j-1} p_i$, τότε η διεργασία j ολοκληρώνει την εκτέλεσή της τη χρονική στιγμή $t + (\sum_{i=1}^{n} p_i - \sum_{i=1}^{j-1} p_i)$. Λόγω του ότι ο χρόνος επεξεργασίας της είναι ίσος με 0, ο χρόνος t θα ισούται με το συνολικό χρόνο επεξεργασίας των πρόωρων διεργασιών, οι οποίες έχουν επιλεγεί πριν την διεργασία j, δηλαδή όλων των προηγούμενων $j-1$ διεργασιών της διάταξης π. Επομένως, θα ισχύει ότι $t = \sum_{i=1}^{j-1} p_i$ και $t + \mathcal{A}_j = \sum_{i=1}^{n} p_i$. Κατά συνέπεια, η πρώτη καθυστερημένη διεργασία που επιλέγεται, χρονοδρομολογείται τελευταία στη προκύπτουσα διάταξη και ο χρόνος ολοκλήρωσής της συμπίπτει με το τέλος της χρονοδρομολόγησης. Ομοίως, η δεύτερη καθυστερημένη διεργασία t_2

[2]Χωρίς βλάβη της γενικότητας.

χρονοδρομολογείται αμέσως πριν την t_1 κ.ο.κ.

Γενικά η l-οστή καθυστερημένη διεργασία t_l που επιλέγεται κατά την m-οστή επανάληψη του δυναμικού προγράμματος, θα ολοκληρώσει την εκτέλεσή τη χρονική στιγμή $t + \mathcal{A}_m = t + (\sum_{i=1}^{n} p_i - \sum_{i=1}^{m-1} p_i) = \sum_{i=1}^{n} p_i - (\sum_{i=1}^{m-1} p_i - t)$, αμέσως πριν τη έναρξη της t_{l-1}. Η ποσότητα $\sum_{i=1}^{m-1} p_i - t$ αντιστοιχεί στο συνολικό χρόνο επεξεργασίας των καθυστερημένων διεργασιών που έχουν επιλεγεί από το δυναμικό πρόγραμμα πριν τη διεργασία l. Στη προκύπτουσα διάταξη χρονοδρομολόγησης, οι καθυστερημένες διεργασίες θα βρίσκονται σε αντίστροφη διάταξη από αυτή που είχαν επιλεγεί από το δυναμικό πρόγραμμα, δηλαδή σε WSPT διάταξη.

Στο Σχήμα 4.7, (Β), απεικονίζουμε τη διάταξη χρονοδρομολόγησης που εξάγεται βάσει της παραπάνω διαδικασίας σε μία επανάληψη του Βήματος 2 του Αλγορίθμου 6. Μία τέτοια διάταξη θα είναι βέλτιστη για το τρέχον υποσύνολο διεργασιών $J - \{k\}$, με συνολική απώλεια ίση με $f^{(k)}(n,t)$. Στο τέλος του Βήματος 2 θα έχουμε υπολογίσει την αντίστοιχη βέλτιστη διάταξη για κάθε δυνατό υποσύνολο $n-1$ διεργασιών. Συνολικά υπάρχουν n τέτοια υποσύνολα που πρέπει να εξεταστούν, οπότε ο συνολικός χρόνος εκτέλεσης του Βήματος 2 θα είναι της τάξης του $O(n^2 d)$. Για να είμαστε ακριβείς, το δυναμικό πρόγραμμα (4.1) δεν είναι σε θέση να παρέχει πληροφορία για το ποιές από τις διεργασίες της αρχικής διάταξης π είναι καθυστερημένες, παρά μόνο για τη συνολική απώλεια αυτών. Χωρίς να μπούμε σε λεπτομέρειες, δεχόμαστε ότι με κατάλληλες προσθήκες στο δυναμικό πρόγραμμα, μπορούμε να αποθηκεύσουμε την πληροφορία αυτή στην έξοδο του προγράμματος. Ο συνολικός χρόνος εκτέλεσης παραμένει ίδιος.



Σχήμα 4.8: Η προσθήκη μίας straddling διεργασίας k σε μία βέλτιστη διάταξη χρονοδρομολόγησης των υπολοίπων $n-1$ διεργασιών.

Η προσθήκη της εκάστοτε straddling διεργασίας k στην αντίστοιχη διάταξη χρονοδρομολόγη-

σης, συνολικής απώλειας $f^{(k)}(n,t)$, μπορεί να πραγματοποιηθεί σε σταθερό χρόνο. Όπως φαίνεται και στο Σχήμα 4.8, η k αρχίζει να εκτελείται αμέσως μετά την ολοκλήρωση και της τελευταίας πρόωρης διεργασίας, δηλαδή τη χρονική στιγμή $d - p_k \leq t < d$, όπου ολοκληρώνεται και η εκτέλεση του δυναμικού προγράμματος. Η διεργασία k ολοκληρώνεται αμέσως πριν την έναρξη της πρώτης καθυστερημένης διεργασίας. Η επιπλέον απώλεια που προσδίδει στην $f^{(k)}(n,t)$, ισούται με $w_k(t+p_k)$. Παρατηρούμε ότι το κενό διάστημα μεταξύ του t και του χρόνου έναρξης της πρώτης καθυστερημένης διεργασίας "χωράει" ακριβώς τις p_k μονάδες χρόνου επεξεργασίας της διεργασίας k: βάσει της εκτέλεσης του δυναμικού προγράμματος, η πρώτη καθυστερημένη διεργασία t_m ολοκληρώνεται τη χρονική στιγμή $C_{t_m} = \sum_{i=1}^{n} p_i - (\mathcal{A}_{n-2} - t)$, όπου $\mathcal{A}_{n-2} = \sum_{i=1}^{n} p_i - p_k - p_{t_m}$ και επομένως αρχίζει να εκτελείται τη χρονική στιγμή $C_{t_m} - p_{t_m}$. Όμως, $(C_{t_m} - p_{t_m}) - t = p_k$.

Στο Βήμα 3 του Αλγορίθμου (6), η παραπάνω διαδικασία προσθήκης της straddling διεργασίας εκτελείται συνολικά n φορές, όσα και τα ζεύγη $(k, f^{(k)}(n,t))$, υπολογίζοντας κάθε φορά μία εφικτή διάταξη χρονοδρομολόγησης για το αρχικό πρόβλημα $1\,|\,d_j = d\,|\,\sum_j w_j T_j$, καθώς και την συνολική απώλεια αυτής. Στο τέλος του Βήματος 3 θα έχουμε επιλέξει τη χρονοδρομολόγηση με την ελάχιστη συνολική απώλεια, έστω $f^{(l)}(n,t) + w_l(t+p_l)$. Ο χρόνος που απαιτείται για το Βήμα 3 είναι της τάξης του $O(n)$.

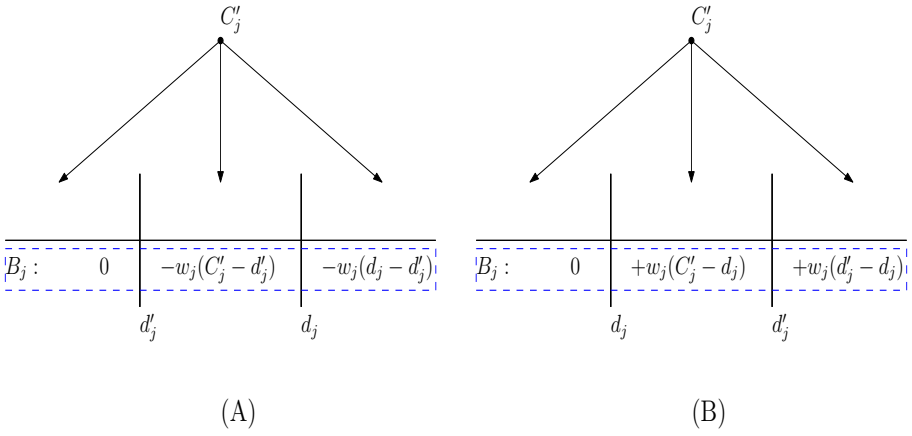
Όπως αναφέραμε αρχικά, η απώλεια $f^{(l)}(n,t) + w_l(t+p_l)$ αντιστοιχεί στην μέση βεβαρημένη ολοκλήρωση των καθυστερημένων διεργασιών της βέλτιστης χρονοδρομολόγησης. Επομένως, για να υπολογίσουμε την ελάχιστη συνολική βεβαρημένη καθυστέρηση, αρκεί να αφαιρέσουμε από την συνολική απώλεια $f^{(l)}(n,t) + w_l(t+p_l)$ την ποσότητα $d\sum_{j'\in K} w_{j'}$, όπου K είναι το σύνολο των καθυστερημένων διεργασιών.

Ο συνολικός χρόνος εκτέλεσης του αλγορίθμου θα είναι της τάξης του $O(n^2 d)$.

4.3 Ένας ψευδοπολυωνυμικός αλγόριθμος για την ελαχιστοποίηση της συνολικής καθυστέρησης

Προηγουμένως, περιγράψαμε έναν ψευδοπολυωνυμικό αλγόριθμο δυναμικού προγραμματισμού για το πρόβλημα ελαχιστοποίησης της συνολικής βεβαρημένης καθυστέρησης σε Μοναδική Μηχανή, στην περίπτωση που οι διεργασίες διαθέτουν μία κοινή προθεσμία, $1\,|\,d_j = d\,|\,\sum_j w_j T_j$ [63]. Στην ενότητα αυτή μελετάμε την εκδοχή του προβλήματος, όπου η κάθε διεργασία j διαθέτει μία διαφορετική προθεσμία d_j, ενώ τα βάρη των διεργασιών είναι συμβατά, υπό την έννοια ότι αν $p_i < p_j$ τότε $w_i \geq w_j$. Συγκεκριμένα, παρουσιάζουμε έναν ψευδοπολυωνυμικό αλγόριθμο δυναμικού προγραμματισμού ο οποίος οφείλεται στον Lawler [60]. Θυμίζουμε ότι το πρόβλημα συμβολίζεται με $1\,|\,w_j \ agreeable\,|\,\sum_j w_j T_j$.

Η βασική ιδέα του αλγορίθμου είναι να περιορίσει το μέγεθος του χώρου των εφικτών χρονοδρομολογήσεων, έτσι ώστε εκτελώντας αποδοτική απαρίθμηση μέσω δυναμικού προγραμματισμού να επιλέξει σε ψευδοπολυωνυμικό χρόνο τη χρονοδρομολόγηση με την ελάχιστη συνολική καθυστέρηση. Τα κριτήρια βελτιστοποίησης που αποδεικνύονται στη συνέχεια, αποκλείουν έναν σημαντικό αριθμό ενδεχόμενων χρονοδρομολογήσεων από τη συμπερίληψή τους στις εφικτές χρονοδρομολογήσεις που εξετάζονται από τον αλγόριθμο.



Σχήμα 4.9: Η j-οστή διεργασία της διάταξης π' όσον αφορά τις προθεσμίες $d'_1, d'_2, \ldots, d'_n$, μπορεί να ολοκληρώσει την εκτέλεσή της σε ένα από τα τρία διαστήματα του σχήματος. Σε κάθε διάστημα, η διαφορά B_j από τη βεβαρημένη καθυστέρηση της διεργασίας στην διάταξη π', όσον αφορά τις προθεσμίες $d_1, d_2, \ldots, d_n$, θα ισούται με την αντίστοιχη ποσότητα που εμφανίζεται στο μπλέ περίγραμμα.

Θεώρημα 4.2 *Υποθέτουμε ότι οι διεργασίες διαθέτουν γενικά βάρη. Έστω π μία βέλτιστη διάταξη χρονοδρομολόγησης για δεδομένες προθεσμίες $d_1, d_2, \ldots, d_n$ και C_j ο χρόνος ολοκλήρωσης της j-οστής διεργασίας στην π. Έστω ότι η προθεσμία d'_j επιλέγεται έτσι ώστε*

$$\min\{d_j, C_j\} \leq d'_j \leq \max\{d_j, C_j\}.$$

Τότε, οποιαδήποτε διάταξη χρονοδρομολόγησης είναι βέλτιστη όσον αφορά τις προθεσμίες $d'_1, d'_2, \ldots, d'_n$, θα είναι βέλτιστη και όσον αφορά τις προθεσμίες $d_1, d_2, \ldots, d_n$. Το αντίστροφο δεν ισχύει, ενώ οι τιμές των δύο βέλτιστων λύσεων μπορεί να διαφέρουν.

Απόδειξη. Έστω T, T' η συνολική βεβαρημένη καθυστέρηση όσον αφορά τις προθεσμίες $d_1, d_2, \ldots, d_n$ και $d'_1, d'_2, \ldots, d'_n$ αντίστοιχα. Έστω π' μία βέλτιστη διάταξη χρονοδρομολόγησης όσον αφορά τις προθεσμίες $d'_1, d'_2, \ldots, d'_n$ και C'_j ο χρόνος ολοκλήρωσης της j-οστής διεργασίας στην π'. Θα δείξουμε ότι $T(\pi') \leq T(\pi)$.

Θέτουμε

$$T(\pi) = T'(\pi) + \sum_j A_j,$$
$$T(\pi') = T'(\pi') + \sum_j B_j.$$

Για οποιαδήποτε διεργασία j της π όσον αφορά τις προθεσμίες $d_1, d_2, \ldots, d_n$, διακρίνουμε δύο περιπτώσεις.

1. $C_j \leq d_j$, οπότε από την αρχική υπόθεση έχουμε ότι $C_j \leq d'_j \leq d_j$. Λόγω του ότι η j παραμένει πρόωρη ($C_j \leq d'_j$) στην π, όσον αφορά τις νέες προθεσμίες $d'_1, d'_2, \ldots, d'_n$, η ποσότητα A_j θα ισούται με $A_j = 0$. Για να προσδιορίσουμε την ποσότητα B_j θα πρέπει να λάβουμε υπόψην ότι ο χρόνος ολοκλήρωσης C'_j της διεργασίας j στην π', όσον αφορά τις προθεσμίες $d'_1, d'_2, \ldots, d'_n$, μπορεί να βρίσκεται σε ένα από τα τρία διαστήματα του σχήματος 4.9, (Α). Συγκεκριμένα θα βρίσκεται, είτε πριν το d'_j, είτε μεταξύ του d'_j και του d_j, είτε μετά το d_j. Σε κάθε περίπτωση η βεβαρημένη καθυστέρηση της j στην π', όσον αφορά τις προθεσμίες $d'_1, d'_2, \ldots, d'_n$, θα είναι μεγαλύτερη ή ίση σε σχέση με την βεβαρημένη καθυστέρησή της στην π' όσον αφορά τις αρχικές προθεσμίες $d_1, d_2, \ldots, d_n$. Η ακριβής διαφορά μεταξύ των δύο για κάθε ενδεχόμενο διάστημα ολοκλήρωσης, προσδιορίζεται αν θέσουμε την ποσότητα B_j ίση με $B_j = -w_j \max\{0, \min\{C'_j, d_j\} - d'_j\}$.

2. $C_j \geq d_j$, οπότε από την αρχική υπόθεση έχουμε ότι $d_j \leq d'_j \leq C_j$. Τότε, η βεβαρημένη καθυστέρηση της j στην π, όσον αφορά τις προθεσμίες $d'_1, d'_2, \ldots, d'_n$, θα είναι μικρότερη κατά $A_j = w_j(d'_j - d_j)$ σε σχέση με την βεβαρημένη καθυστέρησή της στην π, όσον αφορά τις αρχικές προθεσμίες $d_1, d_2, \ldots, d_n$. Σε αντίθεση με την πρώτη περίτωση, λόγω του ότι $d_j \leq d'_j$, η βεβαρημένη καθυστέρηση της j στην π', όσον αφορά τις προθεσμίες $d_1, d_2, \ldots, d_n$, θα είναι μεγαλύτερη ή ίση σε σχέση με την βεβαρημένη καθυστέρησή της στην π', όσον αφορά τις αρχικές προθεσμίες $d'_1, d'_2, \ldots, d'_n$ (βλ. Σχήμα 4.9, (Β)). Η μεταξύ τους διαφορά θα ισούται με $B_j = w_j \max\{0, \min\{C'_j, d'_j\} - d_j\}$.

Σε κάθε περίπτωση θα ισχύει ότι $A_j \geq B_j$ και επομένως $\sum_j A_j \geq \sum_j B_j$. Επιπλέον, από την υπόθεση έχουμε ότι η π' είναι μία βέλτιστη διάταξη χρονοδρομολόγησης όσον αφορά τις προθεσμίες $d'_1, d'_2, \ldots, d'_n$. Επομένως, θα ισχύει ότι $T'(\pi) \geq T'(\pi')$. Συνεπώς, έχουμε ότι $T(\pi) \geq T(\pi')$ το οποίο είναι και το ζητούμενο. □

Θεώρημα 4.3 *Υποθέτουμε ότι οι διεργασίες διαθέτουν συμβατά βάρη. Τότε, υπάρχει μία βέλτιστη διάταξη χρονοδρομολόγησης π τέτοια ώστε:*

1. *για οποιεσδήποτε διεργασίες i, j, αν $d_i \leq d_j$ και $p_i < p_j$, τότε η i προηγείται της j στην π και*

2. *όλες οι πρόωρες διεργασίες βρίσκονται σε EDD διάταξη.*

Απόδειξη. Για την πρώτη συνθήκη, έστω $i, i+1$ δύο διαδοχικές διεργασίες της διάταξης π τέτοιες που να ισχύει ότι $d_i \leq d_{i+1}$ και $p_i < p_{i+1}$. Αντιμεταθέτουμε τις διεργασίες $i, i+1$ στην π και προκύπτει μία νέα διάταξη χρονοδρομολόγησης, έστω π'. Έστω T, T' η συνολική βεβαρημένη καθυστέρηση της π και π' αντίστοιχα. Θεωρούμε προς άτοπο ότι $T' < T$.

Παρατηρούμε ότι οι $i-1$ διεργασίες που προηγούνται της i στην π, καθώς και οι $n-(i+1)$ διεργασίες που έπονται της $i+1$ στην π, διατηρούν την ίδια διάταξη και στην π'. Επιπλέον, λόγω του ότι και στις δύο διατάξεις, μεταξύ των διεργασιών $i-1$ και $i+2$ μεσολαβούν $p_i + p_{i+1}$ μονάδες χρόνου επεξεργασίας, η συνολική βεβαρημένη καθυστέρηση των πρώτων $i-1$ και των τελευταίων $n-(i+1)$ διεργασιών θα είναι ίδια για τις π, π'. Έστω A η καθυστέρηση αυτή. Τότε θα ισχύει ότι

$$T = A + w_i \max\{0, C_{i-1} + p_i - d_i\} + w_{i+1} \max\{0, C_{i-1} + p_i + p_{i+1} - d_{i+1}\}$$

και

$$T' = A + w_{i+1} \max\{0, C_{i-1} + p_{i+1} - d_{i+1}\} + w_i \max\{0, C_{i-1} + p_i + p_{i+1} - d_i\}.$$

Αφαιρώντας κατά μέλη τις δύο εξισώσεις έχουμε ότι

$$\begin{aligned} T' - T = {} & w_i(\max\{0, C_{i-1} + p_i + p_{i+1} - d_i\} - \max\{0, C_{i-1} + p_i - d_i\}) \\ & - w_{i+1}(\max\{0, C_{i-1} + p_i + p_{i+1} - d_{i+1}\} - \max\{0, C_{i-1} + p_{i+1} - d_{i+1}\}). \end{aligned}$$

Λόγω της σχέσης $d_i \leq d_{i+1}$, αν η $i+1$ είναι πρόωρη στην π, τότε θα είναι πρόωρη και στην π', δηλαδή θα εμφανίζει μηδενική καθυστέρηση. Αντίστοιχα, αν η i είναι πρόωρη στην π', τότε θα είναι πρόωρη και στην π, οπότε επίσης θα εμφανίζει μηδενική καθυστέρηση. Κατά συνέπεια, μπορούμε να απλοποιήσουμε την τελευταία σχέση στην ακόλουθη μορφή

$$T' - T = w_i \max\{0, p_{i+1}\} - w_{i+1} \max\{0, p_i\} = \max\{0, w_i p_{i+1} - w_{i+1} p_i\}.$$

Από την υπόθεση έχουμε ότι $p_i < p_{i+1}$. Επειδή τα βάρη των διεργασιών είναι συμβατά θα ισχύει ότι $w_i \geq w_{i+1}$ και επομένως $p_i/w_i \leq p_{i+1}/w_{i+1}$ ή ισοδύναμα $w_i p_{i+1} - w_{i+1} p_i \geq 0$. Έτσι, έχουμε ότι

$$T' - T = \max\{0, w_i p_{i+1} - w_{i+1} p_i\} \geq w_i p_{i+1} - w_{i+1} p_i,$$

ή ισοδύναμα

$$T' \geq (w_i p_{i+1} - w_{i+1} p_i) + T.$$

Από την τελευταία σχέση προκύπτει ότι $T' \geq T$, το οποίο οδηγεί σε άτοπο βάσει της αρχικής υπόθεσης.

Για τη δεύτερη συνθήκη του λήμματος, αρκεί να παρατηρήσουμε ότι αν σε μία διάταξη χρονοδρομολόγησης αντιμεταθέσουμε δύο πρόωρες διεργασίες i, j για τις οποίες ισχύει ότι $d_i \leq d_j$, τότε στην νέα διάταξη που προκύπτει η i μπορεί να σημειώσει καθυστέρηση, αυξάνοντας έτσι την τιμή της αντικειμενικής συνάρτησης σε σχέση με την αρχική διάταξη. □

Για την απόδειξη του Θεωρήματος 4.4 που ακολουθεί, θεωρούμε χ.β.τ.γ ότι οι χρόνοι επεξεργασίας των διεργασιών είναι διακεκριμένοι. Σε αντίθετη περίπτωση, μπορούμε να εφαρμόσουμε μία διαταραχή στους χρόνους αυτούς χωρίς παράλληλα να επηρεάσουμε τη σχέση συμβατότητας που έχουμε υποθέσει για τα βάρη των διεργασιών.

Θεώρημα 4.4 *Υποθέτουμε ότι οι διεργασίες διαθέτουν συμβατά βάρη και είναι διατεταγμένες βάσει του κανόνα EDD, $d_1 \leq d_2 \leq \ldots \leq d_n$. Έστω η διεργασία k τέτοια ώστε $p_k = \max_j p_j$. Τότε, υπάρχει ένας ακέραιος δ, $0 \leq \delta \leq n - k$, τέτοιος ώστε να υπάρχει μία βέλτιστη διάταξη χρονοδρομολόγησης π στην οποία η k έπεται όλων των διεργασιών j, όπου $j \leq k+\delta$ και προηγείται όλων των διεργασιών j, όπου $j > k + \delta$.*

Απόδειξη. Έστω C'_k ο μέγιστος χρόνος ολοκλήρωσης της διεργασίας k για κάθε βέλτιστη διατάξη χρονοδρομολόγησης, όσον αφορά τις προθεσμίες $d_1, d_2, \ldots, d_n$. Έστω π μία βέλτιστη διάταξη χρονοδρομολόγησης όσον αφορά τις προθεσμίες $d_1, \ldots, d_{k-1}, d'_k = \max(C'_k, d_k), d_{k+1}, \ldots, d_n$. Λόγω της υπόθεσης ότι τα βάρη των διεργασιών είναι συμβατά, θεωρούμε ότι η π ικανοποιεί τις δύο συνθήκες του Θεωρήματος 4.3 όσον αφορά τις παραπάνω προθεσμίες. Έστω C_k ο χρόνος ολοκλήρωσης της k στην π.

Από το Θεώρημα 4.2 προκύπτει ότι η π θα είναι βέλτιστη όσον αφορά τις αρχικές προθεσμίες $d_1, d_2, \ldots, d_n$. Επομένως, από την υπόθεση θα ισχύει ότι $C_k \leq C'_k$ και $C_k \leq \max\{C'_k, d_k\} = d'_k$, δηλαδή η διεργασία k θα είναι πρόωρη. Από τη δεύτερη συνθήκη του Θεωρήματος 4.3 η k δεν μπορεί να έπεται μίας διεργασίας j τέτοιας που $d_j > d'_k$, αφού τότε και η j θα είναι πρόωρη, οπότε οδηγούμαστε σε άτοπο. Επίσης από την πρώτη συνθήκη του Θεωρήματος 4.3 η k θα έπεται όλων των διεργασιών για τις οποίες ισχύει ότι $d_j \leq d'_k$. Επιλέγοντας το δ ως τον μέγιστο ακέραιο έτσι ώστε $d_{k+\delta} \leq d'_k$, έχουμε το ζητούμενο. □

Δυναμικός Προγραμματισμός. Το δυναμικό πρόγραμμα που περιγράφεται ακολούθως προέρχεται από την εργασία [60] του Lawler. Υποθέτουμε ότι οι διεργασίες διαθέτουν συμβατά βάρη

και είναι αριθμημένες βάσει του κανόνα EDD. Στόχος μας είναι να βρούμε μία βέλτιστη διάταξη χρονοδρομολόγησης των διεργασιών $1, 2, \ldots, n$, όπου η πρώτη διεργασία αρχίζει να εκτελείται τη χρονική στιγμή $t \geq 0$. Έστω k η διεργασία με το μέγιστο χρόνο επεξεργασίας. Από το Θεώρημα 4.4 προκύπτει ότι για κάποιο δ, $0 \leq \delta \leq n-k$, υπάρχει μία βέλτιστη διάταξη χρονοδρομολόγησης με την ακόλουθη μορφή.

(α) Οι διεργασίες $1, 2, \ldots, k-1, k+1, \ldots, k+\delta$, σε κάποια διάταξη, αρχίζουν να εκτελούνται τη χρονική στιγμή t και ακολουθούνται από τη

(β) διεργασία k, η οποία ολοκληρώνει την εκτέλεσή της σε χρόνο $C_k(\delta) = t + \sum_{j \leq k+\delta} p_j$ και ακολουθείται από

(γ) τις διεργασίες $k+\delta+1, k+\delta+2, \ldots, n$, σε κάποια διάταξη, οι οποίες αρχίζουν να εκτελούνται τη χρονική στιγμή $C_k(\delta)$.

Παρατηρούμε ότι η συνολική διάταξη χρονοδρομολόγησης των διεργασιών θα είναι βέλτιστη μόνο αν οι επιμέρους διατάξεις (α) και (γ) των υποσυνόλων των διεργασιών είναι βέλτιστες, με χρόνους έναρξης εκτέλεσης t και $C_k(\delta)$ αντίστοιχα. Για την επίλυση του προβλήματος κρίνεται σκόπιμο να εφαρμόσουμε δυναμικό προγραμματισμό: για ένα δοθέν υποσύνολο διεργασιών S, με χρόνο έναρξης εκτέλεσης t, υπολογίζουμε αναδρομικά μία βέλτιστη λύση μέσω των βέλτιστων λύσεων για γνήσια υποσύνολα του S, $S' \subset S$, με χρόνο έναρξης $t' \geq t$. Κάθε υποσύνολο S αποτελείται από όλες τις διεργασίες ενός διαστήματος $(i, j) = \{i, i+1, \ldots, j\}$ εξαιρουμένης της διεργασίας k, η οποία θα διαθέτει αυστηρά μεγαλύτερο χρόνο επεξεργασίας p_k από τις υπόλοιπες διεργασίες. Ορίζουμε ένα τέτοιο υποσύνολο διεργασιών ως

$$S(i, j, k) = \{j' \mid i \leq j' \leq j,\ p_{j'} < p_k\}.$$

και έστω $T(S(i, j, k), t)$ η συνολική βεβαρημένη καθυστέρηση μίας βέλτιστης χρονοδρομολόγησης των διεργασιών του συνόλου $S(i, j, k)$, για χρόνο έναρξης εκτέλεσης t. Από την εφαρμογή του Θεωρήματος 4.4 προκύπτει το ακόλουθο δυναμικό πρόγραμμα.

$$\begin{aligned}
T(\emptyset, t) &= 0, \\
T(\{j\}, t) &= w_j \max\{0, t + p_j - d_j\}, \\
T(S(i, j, k), t) &= \min_{\delta} \Big\{ T(S(i, k'+\delta, k'), t) + w_{k'} \max\{0, C_{k'}(\delta) - d_{k'}\} \\
&\qquad + T(S(k'+\delta+1, j, k'), C_{k'}(\delta)) \Big\},
\end{aligned} \tag{4.5}$$

όπου k' η διεργασία με το μέγιστο χρόνο επεξεργασίας στο σύνολο $S(i, j, k)$, $p_{k'} = \max_{j' \in S(i,j,k)} p_{j'}$ και $C_{k'}(\delta) = t + \sum_{j' \in S(i,k'+\delta,k')} p_{j'} + p_{k'}$ ο χρόνος ολοκλήρωσης αυτής. Η διαδικασία εκτέλεσης του

δυναμικού προγράμματος διαιρείται στις εξής δύο φάσεις: την *παραγωγή υποπροβλημάτων* και τη *φάση της αναδρομής*. Στην πρώτη αρχίζουμε με ένα πρόβλημα ($S = \{1, 2, \ldots, n\}, t = 0$) και το διαιρούμε επαναληπτικά σε υποπροβλήματα, για τα οποία πρέπει να επιλύσουμε τις αντίστοιχες εξισώσεις (4.5). Στην φάση της αναδρομής επιλύουμε τα υποπροβλήματα αυτά, σε αντίστροφη σειρά από αυτή που παρήχθησαν στην πρώτη φάση.

Για τη χρονική πολυπλοκότητα του δυναμικού προγράμματος, παρατηρούμε ότι οι μεταβλητές i, j, k μπορούν να πάρουν συνολικά το πολύ n διαφορετικές τιμές, όσος και ο αριθμός των διεργασιών. Επομένως, θα χρειαστεί να εξετάσουμε το πολύ $O(n^3)$ υποσύνολα διεργασιών $S(i, j, k)$, για τον υπολογισμό μίας βέλτιστης λύσης του αρχικού συνόλου των n διεργασιών· πρακτικά, σε μία υλοποίηση θα χρειαστεί να εξεταστούν αρκετά λιγότερα υποσύνολα, καθώς πολλοί διαφορετικοί συνδυασμοί των i, j, k αφορούν στο ίδιο υποσύνολο. Επιπλέον, οι διαφορετικές τιμές που μπορεί να πάρει ο χρόνος t θα είναι το πολύ $P = \sum_j p_j \leq np_{max}$, όπου $p_{max} = \max_j p_j$. Έτσι, το πλήθος των αναδρομικών εξισώσεων (4.5) που επιλύονται θα είναι το πολύ $O(n^4 p_{max})$. Για την επίλυση κάθε αναδρομικής εξίσωσης απαιτείται ελαχιστοποίηση γύρω από $\delta \leq n$ εναλλακτικές τιμές και συνολικός χρόνος $O(n)$. Κατά συνέπεια, το παραπάνω δυναμικό πρόγραμμα βρίσκει μία βέλτιστη λύση σε χρόνο το πολύ $O(n^5 p_{max})$.

Ακολούθως, παρουσιάζουμε μία εφαρμογή του παραπάνω δυναμικού προγράμματος σε ένα στιγμιότυπο I δεκαπέντε διεργασιών,[3] $n = 15$. Θεωρούμε ότι το βάρος κάθε διεργασίας j ισούται με $w_j = 1$. Οι χρόνοι επεξεργασίας και οι προθεσμίες των διεργασιών απεικονίζονται στον παρακάτω πίνακα. Οι διεργασίες είναι αριθμημένες και διατεταγμένες βάσει του κανόνα EDD.

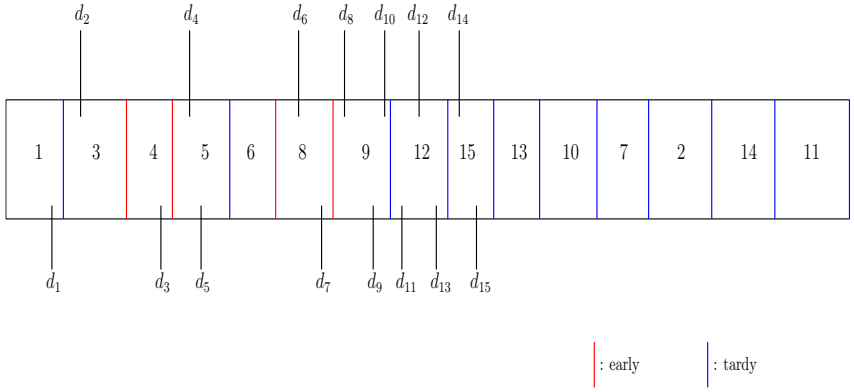
j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
p_j	131	168	157	66	153	69	162	123	114	151	192	126	127	188	70
d_j	102	137	350	370	375	641	674	718	802	811	859	874	886	963	991

Η διαδικασία εκτέλεσης αρχίζει με το πρόβλημα ($S = \{1, 2, \ldots, 15\}, t = 0$). Από το σύνολο S επιλέγεται η διεργασία $j = 11$, η οποία διαθέτει το μέγιστο χρόνο επεξεργασίας $p_j = 192$ και το πρόβλημα διαιρείται σε πέντε υποπροβλήματα (το δ θα ανήκει στο διάστημα $0 \leq \delta \leq n - k = 4$). Η αναδρομική εξίσωση (4.5) αρχικά διαμορφώνεται ως ακολούθως.

[3] Το αποτελέσμα που παρουσιάζεται προέκυψε μέσω μίας υλοποίησης του δυναμικού προγράμματος (4.5).

$$T(\{1,2,\ldots,15\},0) = \min_{0\le\delta\le 4}\left\{\begin{array}{l} T(S(1,11,11),0)+627+T(S(12,15,11),1486),\\ T(S(1,12,11),0)+753+T(S(13,15,11),1612),\\ T(S(1,13,11),0)+880+T(S(14,15,11),1739),\\ T(S(1,14,11),0)+1068+T(S(15,15,11),1927),\\ T(S(1,15,11),0)+1138+T(\emptyset,1997)\end{array}\right\} \quad (4.6)$$

Στη συνέχεια καθένα από τα πέντε υποπροβλήματα, υποδιαιρείται με όμοιο τρόπο σε μικρότερα υποπροβλήματα κ.ο.κ. Μόλις ολοκληρωθεί η παραγωγή υποπροβλημάτων, αρχίζει η φάση της αναδρομής: η τιμή της συνολικής καθυστέρησης $T(S',t)$ κάθε υποπροβλήματος (S',t) υπολογίζεται μέσω ελαχιστοποίησης από τις τιμές της συνολικής καθυστέρησης των υποπροβλημάτων στα οποία έχει υποδιαιρεθεί. Η έξοδος του δυναμικού προγράμματος δίνει μία βέλτιστη διάταξη



Σχήμα 4.10: Η βέλτιστη διάταξη χρονοδρομολόγησης μετά την ολοκλήρωση της εκτέλεσης του δυναμικού προγράμματος σε ένα στιγμιότυπο δεκαπέντε διεργασιών.

χρονοδρομολόγησης για τις διεργασίες του παραπάνω στιγμιοτύπου (βλ. Σχήμα 4.10). Όπως μπορούμε να διακρίνουμε, η βέλτιστη διάταξη χρονοδρομολόγησης για το παραπάνω στιγμιότυπο είναι η $\{1,3,4,5,6,8,9,12,15,13,10,7,2,14,11\}$. Οι διεργασίες $\{3,4,6,8\}$ ολοκληρώνονται πρόωρα, ενώ οι υπόλοιπες είναι καθυστερημένες. Στον παρακάτω πίνακα σημειώνουμε τις τιμές του χρόνου ολοκλήρωσης και της αντίστοιχης καθυστέρησης, για κάθε διεργασία στη βέλτιστη διάταξη. Η ελάχιστη συνολική καθυστέρηση $T(S = \{1,2,3,\ldots,15\},0)$ θα ισούται με $\sum_{j=1}^{15} T_j = 5216$.

j	1	3	4	5	6	8	9	12	15	13	10	7	2	14	11
C_j	131	288	354	507	576	699	813	939	1009	1136	1287	1449	1617	1805	1997
T_j	29	0	0	132	0	0	11	65	18	250	476	775	1480	842	1138

Διαδικασίες βελτίωσης της απόδοσης του δυναμικού προγράμματος. Υπάρχουν αρκετοί τρόποι με τους οποίους μπορούμε να βελτιώσουμε τον χρόνο εκτέλεσης του δυναμικού προγράμματος

(βλ. [60]). Στη συνέχεια περιγράφουμε τρείς από αυτούς.

Όπως ήδη αναφέραμε, τα υποσύνολα διεργασιών που χρειάζεται να εξετάσουμε είναι πολύ λιγότερα από $O(n^3)$. Μπορούμε εύκολα να παρατηρήσουμε ότι υπάρχουν αρκετά υποσύνολα $S(i,j,k)$, $S(i',j',k')$, με $i \neq i', j \neq j', k \neq k'$, τα οποία αντιστοιχούν στο ίδιο υποσύνολο διεργασιών, π.χ. $S(11,14,11) = \{12,13,14\} = S(12,15,15)$.

Σε αρκετές περιπτώσεις είναι εφικτό να περιορίσουμε το πλήθος των τιμών της μεταβλητής δ, η οποία καθορίζει το πλήθος των υποπροβλημάτων που πρέπει να επιλυθούν στη φάση παραγωγής υποπροβλημάτων. Γενικά, το δ μπορεί να πάρει το πολύ $n-k$ διαφορετικές τιμές, $\delta = \delta_i, i = 0,1,\ldots,n-k$, όπου k είναι η διεργασία του τρέχοντος υποσυνόλου S με το μέγιστο χρόνο επεξεργασίας $p_k = \max_{j \in S} p_j$. Η διαδικασία ($\mathcal{A}$) καθορίζει ποιές τιμές του δ_i επαρκούν για την ελαχιστοποίηση μέσω της εξίσωσης (4.5). Υποθέτουμε ότι οι διεργασίες είναι διατεταγμένες βάσει του κανόνα EDD, ενώ τα βάρη τους είναι μοναδιαία.

($\mathcal{A}$)

1. $i = 1$.
2. $d'_k = t + \sum_{j \in S'} p_j$, $S' = \{j \mid d_j \leq d_k,\ j \in S\}$.
3. **Αν** $d'_k > d_k$ **τότε** $d_k = d'_k$ και επέστρεψε στο Βήμα 2.

   Θέσε $\delta_i = j - k$, όπου j ο μεγαλύτερος ακέραιος στο S τέτοιος που $d_j \leq d_k$.

   Έστω $S'' = \{j \mid d_j > d_k,\ j \in S\}$.

   Αν $S'' = \emptyset$ **τότε** τερμάτισε.

   Αλλιώς έστω j' τέτοιο που $d_{j'} = \min_{j' \in S''} d_j$.

   Θέσε $d_k = d_j$ και $i = i + 1$ και επέστρεψε στο Βήμα 2.

Σε κάθε επανάληψη των Βημάτων 1 και 2 δοκιμάζουμε να εκτελέσουμε τη διεργασία k ως πρόωρη, μετατοπίζοντας την σε μία θέση $j \geq k$ στην EDD διάταξη, όπου $d_j \leq d_k$, αυξάνοντας αντίστοιχα την προθεσμία της σε $d'_k = d_k$. Μόλις βρεθεί μία θέση j στην οποία η k ολοκληρώνεται πρόωρα, $d'_k \leq d_k$, μετατοπίζουμε τη διεργασία στη θέση αυτή και ενημερώνουμε την μεταβλητή δ, $\delta_i = j-k$. Επαναλαμβάνουμε για τις διεργασίες-θέσεις $j' \in S''$ της EDD διάταξης που απομένουν μετά την j, αρχίζοντας με $d_k = min_{j' \in S} d_{j'}$. Μετά την ολοκλήρωση της ($\mathcal{A}$), θεωρούμε προς επίλυση μόνο εκείνα τα υποπροβλήματα τα οποία υποδεικνύονται από τις τιμές δ_i που προέκυψαν.

Η ορθότητα της διαδικασίας ($\mathcal{A}$) αποδεικνύεται βάσει των Θεωρημάτων 4.2 και 4.3. Από το Θεώρημα 4.3 θα υπάρχει μία βέλτιστη διάταξη χρονοδρομολόγησης, στην οποία ο χρόνος ολοκλήρωσης της k θα είναι τουλάχιστον ίσος με d'_k και βάσει του Θεωρήματος 4.2 θα υπάρχει μία βέλτιστη διάταξη, όσον αφορά την προθεσμία d'_k, η οποία θα είναι βέλτιστη και όσον αφορά

την αρχική προθεσμία d_k.

Πέραν του περιορισμού, μπορούμε να επιτύχουμε ακόμα και την διακοπή της παραγωγής υποπροβλημάτων για την επίλυση ενός υποπρόβληματος. Οι δύο διαδικασίες που περιγράφονται ακολούθως, σε αρκετές περιπτώσεις, δίνουν τη δυνατότητα να επιλύσουμε απευθείας κάποιο υποπρόβλημα αποφεύγοντας την διαίρεσή του σε μικρότερα υποπροβλήματα. Η ορθότητά τους στηρίζεται στα επόμενα δύο θεωρήματα.

Θεώρημα 4.5 *Υποθέτουμε ότι οι διεργασίες διαθέτουν αυθαίρετα βάρη. Έστω π η διάταξη χρονοδρομολόγησης των διεργασιών βάσει του κανόνα WSPT. Αν όλες οι διεργασίες είναι καθυστερημένες, τότε η π είναι βέλτιστη.*

Απόδειξη. Μία απόδειξη δίνεται στην εργασία του McNaughton [69] (βλ. Ενότητα 4.1). Εναλλακτικά, παρατηρούμε ότι η συνολική βεβαρημένη καθυστέρηση θα ισούται με

$$\sum_j w_j T_j \;=\; \sum_j w_j C_j + \sum_j w_j \max\{0, d_j - C_j\} - \sum_j w_j d_j.$$

Είναι γνωστό από τον Smith [92] ότι η π ελαχιστοποιεί την ποσότητα $\sum_j w_j C_j$. Επομένως, αν όλες οι διεργασίες είναι καθυστερημένες το άθροισμα $\sum_j w_j \max\{0, d_j - C_j\}$ ισούται με μηδέν. Συνεπώς, το υπόλοιπο άθροισμα ελαχιστοποιείται. □

Αν επιπλέον, είχαμε θεωρήσει ότι τα βάρη των διεργασιών είναι συμβατά και οι χρόνοι επεξεργασίας τους είναι διακεκριμένοι, τότε θα αρκούσε η χρήση της SPT διάταξης, αφού σ' αυτή την περίπτωση ισοδυναμεί με την WSPT.

Η απόδειξη του ακόλουθου θεωρήματος είναι προφανής.

Θεώρημα 4.6 *Υποθέτουμε ότι οι διεργασίες διαθέτουν αυθαίρετα βάρη. Έστω π η διάταξη χρονοδρομολόγησης των διεργασιών που ελαχιστοποιεί την ποσότητα*

$$\max_j w_j T_j.$$

Αν το πολύ μία διεργασία είναι καθυστερημένη, τότε η π είναι βέλτιστη.

Θυμίζουμε ότι μία διάταξη χρονοδρομολόγησης που ελαχιστοποιεί την ποσότητα $\max_j w_j T_j$ μπορεί να κατασκευαστεί σε χρόνο $O(n^2)$ (βλ. [58])· στην μη βεβαρημένη περίπτωση ο κανόνας EDD ελαχιστοποιεί την μέγιστη καθυστέρηση.

Θεωρούμε τώρα ένα υποπρόβλημα (S, t) και έστω ότι οι διεργασίες του S αριθμούνται έτσι ώστε $p_1 > p_2 > \ldots > p_n$. Για την διαδικασία (Α), βάσει του Θεωρήματος 4.5, μειώνουμε τις τιμές των d_j ώστε να αυξήσουμε το πλήθος των καθυστερημένων διεργασιών. Από το Θεώρημα

4.1, μία βέλτιστη διάταξη χρονοδρομολόγησης όσον αφορά τις νέες προθεσμίες, θα είναι βέλτιστη και όσον αφορά τους αρχικούς. Για τη διαδικασία (Β), βάσει του Θεωρήματος 4.6, αυξάνουμε τις τιμές των d_j ώστε να μειώσουμε το πλήθος των καθυστερημένων διεργασιών. Αντίστοιχα, από το Θεώρημα 4.1 μία βέλτιστη διάταξη χρονοδρομολόγησης, όσον αφορά τις νέες προθεσμίες, θα είναι βέλτιστη και όσον αφορά τους αρχικούς.

(Α) 1. $k = n + 1$.

2. **Αν** $k = 1$ **τότε** τερμάτισε. **Αλλιώς** $k = k - 1$.

3. Θέσε $d'_k = d_k$.

4. Έστω $S^{(k)} = \{j \mid j \in S,\ d_j \geq d'_k,\ p_j > p_k\}$. Θέσε $C_k = t + \sum_{j \in S - S^{(k)}} p_j$.

5. **Αν** $C_k < d'_k$ **τότε** $d'_k = C_k$ και επέστρεψε στο Βήμα 4.

Αλλιώς επέστρεψε στο Βήμα 2.

Στο Βήμα 4, παρατηρούμε ότι το σύνολο $S^{(k)}$ θα περιέχει όλες εκείνες τις διεργασίες για τις οποίες, βάσει του Θεωρήματος 4.2, μπορούμε να υποθέσουμε ότι ακολουθούν την διεργασία k.

(Β) 1. $k = 1$.

2. **Αν** $k = n$ **τότε** τερμάτισε. **Αλλιώς** $k = k + 1$.

3. Θέσε $d'_k = d_k$.

4. Έστω $S^{(k)} = \{j \mid j \in S,\ d_j \leq d'_k,\ p_j < p_k\}$. Θέσε $C_k = t + p_k + \sum_{j \in S^{(k)}} p_j$.

5. **Αν** $C_k > d'_k$ **τότε** $d'_k = C_k$ και επέστρεψε στο Βήμα 4.

Αλλιώς επέστρεψε στο Βήμα 2.

Για μία εφαρμογή των διαδικασιών ($\mathcal{A}$), (Α), (Β), σε ένα στιγμιότυπο δέκα διεργασιών με μοναδιαία βάρη, ο αναγνώστης παραπέμπεται στην εργασία [60].

Ωστόσο, καμία από τις παραπάνω διαδικασίες δεν βελτιώνει την θεωρητική πολυπλοκότητα του δυναμικού προγράμματος, η οποία στη χείριστη περίπτωση παραμένει ψευδοπολυωνυμική της τάξης του $O(n^5 p_{max})$. Αξίζει να σημειωθεί, ότι τη χρονική περίοδο που σχεδιάστηκε το παραπάνω δυναμικό πρόγραμμα [60], δεν ήταν γνωστό αν το πρόβλημα $1 \parallel \sum_{j=1}^{n} w_j T_j$ παραμένει **NP**-hard στην περίπτωση που τα βάρη των διεργασιών είναι συμβατά ή μοναδιαία· αρκετά χρόνια αργότερα αποδείχθηκε ότι το πρόβλημα είναι **NP**-hard υπό τη συνήθη έννοια [24] (βλ. Ενότητα 4.1). Ο Lawler, στην ίδια εργασία [60], επισημαίνει ότι υπάρχουν ενδεχόμενα τα οποία δεν φαίνεται να δικαιώνουν την αναζήτηση ενός πολυωνυμικού αλγορίθμου. Συγκεκριμένα, για ένα δοθέν σύνολο διεργασιών S, η $T(S, t)$ είναι κατά τμήματα γραμμική συνάρτηση (piecewise linear)

του t. Για παράδειγμα αν υποθέσουμε ότι έχουμε υπολογίσει την τιμή $T(S,0)$ (για μοναδιαία βάρη) και αυξήσουμε το t από 0 σε $t' > 0$, τότε για την τιμή $T(S,t')$ θα ισχύει ότι $T(S,t') \leq T(S,0) + kt'$, όπου k είναι το πλήθος των διεργασιών του S οι οποίες ήταν ήδη καθυστερημένες, οπότε η καθυστέρησή τους αυξήθηκε κατά t', συν τις διεργασίες που ήταν πρόωρες και εμφανίζουν καθυστέρηση, η οποία θα είναι το πολύ ίση με t'. Αν η $T(S,t)$ στρέφει τα κοίλα άνω (convex), τότε θα πρέπει ο ρυθμός μεταβολής της συνολικής καθυστέρησης να αυξάνεται μονοτονικά με το χρόνο, δηλαδή η γραφική παράσταση της $T(S,t)$ θα αποτελείται από το πολύ $n+1$ διαδοχικά γραμμικά τμήματα με αυξανόμενη κλίση στα σημεία $0,1,\ldots,n$. Στην περίπτωση αυτή, θα μπορούσαμε να υπολογίσουμε την τιμή της $T(S,t)$ σε πολυωνυμικό χρόνο, χρησιμοποιώντας την αναδρομική εξίσωση (4.5). Παρόλα αυτά, μπορούν να βρεθούν αντιπαραδείγματα όπου η αντίστοιχη γραφική παράσταση δεν στρέφει τα κοίλα άνω.

4.4 Δύο πλήρη πολυωνυμικά προσεγγιστικά σχήματα

Κρίνοντας από τα μέχρι τώρα γνωστά αποτελέσματα, στις περισσότερες περιπτώσεις ο σχεδιασμός ενός πλήρους πολυωνυμικού προσεγγιστικού σχήματος προϋποθέτει την εύρεση ενός ψευδοπολυωνυμικού αλγορίθμου για το αντίστοιχο πρόβλημα. Ένα από τα πρώτα FPTAS που παρουσιάστηκαν, ήταν αυτό των Gens και Levner [31] για το πρόβλημα $1 \parallel \sum_j w_j U_j$. Για το σχεδιασμό του, εφαρμόστηκε η τεχνική αποκοπής των Ibarra και Kim [44] (βλ. Ενότητα 3.2) στον ψευδοπολυωνυμικό αλγόριθμο των Lawler και Moore [63] που παρουσιάσαμε στην Ενότητα 4.2.2. Αργότερα, Ο Lawler [61] σχεδίασε ένα FPTAS για το πρόβλημα $1 \parallel \sum_j T_j$, χρησιμοποιώντας τον δικό του ψευδοπολυωνυμικό αλγόριθμο για το πρόβλημα $1 \mid w_j \; agreeable \mid \sum_j w_j T_j$ [60] (βλ. Ενότητα 4.3). Στην Ενότητα 4.4.1 περιγράφουμε μία απλοποιημένη εκδοχή του αποτελέσματος Gens και Levner [31] και στην Ενότητα 4.4.2 παρουσιάζουμε το αποτέλεσμα του Lawler [61].

4.4.1 Ένα FPTAS για το πρόβλημα $1 \parallel \sum_j w_j U_j$

Όπως αναφέραμε στην Ενότητα 4.1, το πρόβλημα $1 \parallel \sum_j w_j U_j$ είναι **NP**-hard υπό τη συνήθη έννοια Karp [49]. Στη συνέχεια, παρουσιάζουμε ένα FPTAS από την εργασία [86] των Schuurman και Woeginger.

Εφαρμόζουμε την τεχνική αποκοπής που περιγράψαμε στην Ενότητα 3.2. Ο σχεδιασμός του FPTAS συνοψίζεται στα ακόλουθα τρία στάδια. Σημειώνουμε ότι το μέγεθος $|I|$ ενός στιγμιοτύπου I του προβλήματος, είναι τουλάχιστον ίσο με $\log(p_{sum} w_{sum})$, όπου $p_{sum} = \sum_{j=1}^{n} p_j$ και $w_{sum} = \sum_{j=1}^{n} w_j$.

Α. Διαμόρφωση ενός ακριβούς αλγορίθμου. Το δυναμικό πρόγραμμα (4.2) της Ενότητας 4.2.2

αποτελεί έναν ακριβή αλγόριθμο για το πρόβλημα καθώς βρίσκει μία βέλτιστη λύση σε ψευδοπολυωνυμικό χρόνο της τάξης του $O(np_{sum})$. Θυμίζουμε ότι αρχικά οι διεργασίες διατάσσονται και αριθμούνται βάσει του κανόνα EDD. Στη βέλτιστη διάταξη χρονοδρομολόγησης που προκύπτει οι πρόωρες διεργασίες εκτελούνται σύμφωνα με την αρχική τους διάταξη, ενώ οι καθυστερημένες διεργασίες εκτελούνται σε αυθαίρετη διάταξη, μετά την ολοκλήρωση των πρόωρων.

Για την εφαρμογή της τεχνικής αποκοπής θα χρειαστεί να διαμορφώσουμε κατάλληλα τον παραπάνω αλγόριθμο με κάποιο επιπλέον κόστος, όπως θα δούμε, στη χρονική του πολυπλοκότητα. Παρατηρούμε ότι οι χαρακτηριστικές ιδιότητες μίας εφικτής χρονοδρομολόγησης των διεργασιών μπορούν να κωδικοποιηθούν σε ένα διάνυσμα δύο διαστάσεων της μορφής $[P, W]$. Με P συμβολίζουμε τον συνολικό χρόνο επεξεργασίας των πρόωρων διεργασιών που έχουν χρονοδρομολογηθεί και με W το συνολικό βάρος των καθυστερημένων διεργασιών, οι οποίες εκτελούνται αμέσως μετά την ολοκλήρωση των πρόωρων. Η τιμή της αντικειμενικής συνάρτησης $\sum_j w_j U_j$ μπορεί να υπολογισθεί σε σταθερό χρόνο, προβάλλοντας την δεύτερη συντεταγμένη του διανύσματος μέσω μίας συνάρτησης προβολής $P_2^2(x_1, x_2) = x_2$. Ο αλγόριθμος περιγράφεται στο ακόλουθο σχήμα.

Αλγόριθμος 7 Ένας ακριβής αλγόριθμος για το πρόβλημα $1 \parallel \sum_j w_j U_j$.

1. Διάταξε και αρίθμησε τις διεργασίες εκτελώντας τον κανόνα EDD. : $d_1 \leq d_2 \leq \ldots \leq d_n$.

2. **Αν** $p_1 \leq d_1$ **τότε** $VS_1 = \{[p_1, 0],\ [0, w_1]\}$. Αλλιώς $VS_1 = \{[0, w_1]\}$.

3. **Για** $\kappa = 2$ έως n

   3.1 **Για** κάθε διάνυσμα $[x, y] \in VS_{k-1}$

      3.1.1 $VS_k = VS_k \bigcup \{[x, y + w_k]\}$.

      3.1.2 **Αν** $x + p_k \leq d_k$ **τότε** $VS_k = VS_k \bigcup \{[x, y + w_k],\ [x + p_k, y]\}$.

4. Επέλεξε το διάνυσμα $[x, y] \in VS_n$ με την ελάχιστη τιμή για το y.

Στο Βήμα 3, ο Αλγόριθμος 7 χρονοδρομολογεί αρχικά την τρέχουσα διεργασία k ως καθυστερημένη, αυξάνοντας την τιμή της αντικειμενικής συνάρτησης κατά w_k και στη συνέχεια (Βήμα 3.1.2) ελέγχει αν μπορεί να τη χρονοδρομολογήσει ως πρόωρη. Αν ναι, τότε αυξάνει το συνολικό χρόνο επεξεργασίας των πρόωρων διεργασιών κατά p_k και προσθέτει στο σύνολο VS_k το αντίστοιχο διάνυσμα. Λόγω του ότι η πρώτη συντεταγμένη κάθε διανύσματος παίρνει τιμές από 0 έως p_{sum} και η δεύτερη από 0 έως w_{sum}, ο πληθικός αριθμός ενός συνόλου διανυσμάτων VS_k θα είναι το πολύ

$O(p_{sum}w_{sum})$. Επιπλέον, η χρονική πολυπλοκότητα του αλγορίθμου είναι ανάλογη της ποσότητας $\sum_{k=1}^{n}|VS_k|$. Επομένως, η πολυπλοκότητα χρόνου του Αλγορίθμου 7 θα είναι ψευδοπολυωνυμική της τάξης του $O(np_{sum}w_{sum})$, σαφώς μεγαλύτερη από αυτή του δυναμικού προγράμματος (4.2).

Β. Επιτάχυνση της εκτέλεσης του αλγορίθμου. Σε πλήρη αντιστοιχία με το Στάδιο Β του FPTAS της Ενότητας 3.2, θεωρούμε καθένα από τα παραπάνω διανύσματα ως ένα γεωμετρικό σημείο του ορθογωνίου παραλληλογράμμου $[0, p_{sum}] \times [0, w_{sum}]$. Υποδιαιρούμε το παραλληλόγραμμο σε μικρά "κουτιά" δημιουργώντας τομές και στις δύο διαστάσεις, στις συντεταγμένες που ορίζονται από τα Δ^i για $i = 1, 2, \ldots, L$, όπου

$$\Delta \;=\; 1 + \frac{\epsilon}{2n}. \tag{4.7}$$

και όπου

$$L \;\leq\; \lceil \log_{\Delta}(\max\{p_{sum}, w_{sum}\}) \rceil \;\leq\; \lceil (1 + \frac{2n}{\epsilon}) \ln(\max\{p_{sum}, w_{sum}\}) \rceil. \tag{4.8}$$

Παρατηρούμε ότι τα διανύσματα που βρίσκονται στο ίδιο κουτί είναι σε πολύ μικρή απόσταση μεταξύ τους. Έτσι, μπορούμε να απλοποιήσουμε ένα σύνολο διανυσμάτων VS_k επιλέγοντας, από κάθε κουτί που περιέχει διανύσματα αυτού, ένα μοναδικό διάνυσμα. Τα διανύσματα που συλλέγονται συνιστούν το σύνολο αποκοπής VS'_k. Σε αντίθεση με την Ενότητα 3.2, η επιλογή αυτού του διανύσματος δεν γίνεται αυθαίρετα. Είναι σημαντικό να παρατηρήσουμε ότι σε αντίθεση με τον Αλγόριθμο 4 της Ενότητας 3.2, στον Αλγόριθμο 7 υπάρχουν συνθήκες της μορφής "αν ... τότε". Έτσι, αν υποθέσουμε ότι για την κατασκευή ενός συνόλου αποκοπής VS'_k επιλέγουμε αυθαίρετα ένα μοναδικό διάνυσμα από κάθε κουτί, τότε είναι πολύ πιθανό ο αλγόριθμος αποκοπής να οδηγηθεί σε μία προσεγγιστική λύση με πολύ μεγάλη απόκλιση από τη βέλτιστη (βλ. σελίδα 34 στο [86]). Το πρόβλημα αυτό αντιμετωπίζεται αν για κάθε κουτί επιλέξουμε να προσθέσουμε στο σύνολο VS'_k το διάνυσμα με τη μικρότερη x-συντεταγμένη. Όλα τα διανύσματα του συνόλου VS_k που δεν επιλέχθηκαν δεν λαμβάνουν μέρος στους επόμενους υπολογισμούς. Έτσι, ο αλγόριθμος αποκοπής παράγει το επόμενο σύνολο διανυσμάτων VS_{k+1} με βάση το σύνολο VS'_k και όχι το σύνολο VS_k. Κάθε σύνολο αποκοπής περιέχει το πολύ ένα διάνυσμα από κάθε κουτί και συνολικά υπάρχουν το πολύ $O(L^2)$ κουτιά. Βάσει της (4.8) το πλήθος τους θα είναι πολυωνυμικό ως προς το μέγεθος εισόδου $|I|$ και το $1/\epsilon$. Επομένως, ο αλγόριθμος αποκοπής, που τρέχει σε χρόνο ανάλογο της ποσότητας $\sum_{k=1}^{n}|VS'_k|$, θα έχει πολυπλοκότητα χρόνου πολυωνυμική ως προς το μέγεθος της εισόδου και το $1/\epsilon$.

Γ. Ανάλυση χείριστης περίπτωσης Ο Αλγόριθμος 7 κατασκευάζει τα σύνολα διανυσμάτων $VS_1, \ldots, VS_n$ υπολογίζοντας κάθε σύνολο VS_{k+1} μέσω του συνόλου VS_k, ενώ ο αλγόριθμος α-

ποκοπής κατασκευάζει τα σύνολα διανυσμάτων $VS'_1, \ldots, VS'_n$ υπολογίζοντας κάθε σύνολο VS'_{k+1} μέσω του συνόλου VS'_k. Η Πρόταση 4.1 εξασφαλίζει ότι το σύνολο αποκοπής VS'_k είναι μία ικανοποιητική προσέγγιση του συνόλου VS_k και μπορεί εύκολα να αποδειχθεί εφαρμόζοντας επαγωγή στο k (βλ. σελίδα 32 στο [86]).

Πρόταση 4.1 *Για κάθε διάνυσμα $[x, y] \in VS_k$, υπάρχει ένα διάνυσμα $[x', y'] \in VS'_k$ τέτοιο ώστε να ισχύουν οι ακόλουθες ανισότητες:*

$$x' \leq x, \qquad y' \leq \Delta^k y.$$

Παρατηρούμε ότι η ανισότητα $x' \leq x$ είναι πιο ισχυρή από την αντίστοιχη ανισότητα $x' \leq \Delta^k x$ του Πορίσματος 3.1. Έστω, $[x, y] \in VS_n$ το διάνυσμα που επιλέγεται στην έξοδο του Αλγορίθμου 7. Από το Πόρισμα 4.1 θα υπάρχει ένα διάνυσμα $[x', y'] \in VS'_n$, τέτοιο ώστε $x' \leq x$ και $y' \leq \Delta^n y$. Θα ισχύει ότι $P_2^2(x', y') \leq P_2^2(x, \Delta^n y) = \Delta^n y = \Delta^n \, \mathrm{OPT}$. Επομένως, ο αλγόριθμος αποκοπής είναι ένας Δ^n-προσεγγιστικός αλγόριθμος για το πρόβλημα $P2 \parallel \sum_j w_j C_j$. Η ποσότητα Δ^n, βάσει της γνωστής ανισότητας $(1 + p/n)^n \leq 1 + 2n$, η οποία ισχύει για κάθε $0 \leq p \leq 1$ και της σχέσης (4.7), για $p = \epsilon/2$, θα είναι $\Delta^n \leq 1 + \epsilon$. Καταλήγουμε έτσι, ότι ο αλγόριθμος αποκοπής αποτελεί ένα FPTAS για το πρόβλημα $1 \parallel \sum_j w_j U_j$.

Στην Ενότητα 4.2.2 παρατηρήσαμε ότι μπορούμε να υπολογίσουμε μία βέλτιστη λύση του προβλήματος $1 \parallel \sum_{j=1}^n w_j U_j$, αν εναλλακτικά επιλύσουμε το πρόβλημα μεγιστοποίησης του συνολικού βάρους των πρόωρων διεργασιών. Θεωρούμε λοιπόν ότι αντί του διανύσματος $[P, W]$, που κωδικοποιεί μία εφικτή λύση του προβλήματος $1 \parallel \sum_{j=1}^n w_j U_j$, χρησιμοποιούμε ένα διάνυσμα τριών διαστάσεων $[P, W, W']$, όπου W' είναι το συνολικό βάρος των πρόωρων διεργασιών της χρονοδρομολόγησης. Επιπλέον, αντί της συνάρτησης προβολής $P_2^2(x_1, x_2) = x_2$ θεωρούμε τη συνάρτηση $P_3^3(x_1, x_2, x_3) = x_3$ και στο Βήμα 4 του Αλγορίθμου 7 επιλέγουμε το διάνυσμα με τη μέγιστη x_3-συντεταγμένη. Τότε, η παραπάνω διαδικασία, με τις προφανείς τροποποποιήσεις στα Στάδια Α και Γ, δίνει ένα FPTAS για το πρόβλημα μεγιστοποίησης του συνολικού βάρους των πρόωρων διεργασιών.

4.4.2 Ένα FPTAS για το πρόβλημα $1 \parallel \sum_j T_j$

Το πρόβλημα $1 \parallel \sum_j T_j$ αποδείχθηκε **NP**-hard υπό τη συνήθη έννοια από τους Du και Leung [24]. Στη συνέχεια παρουσιάζουμε ένα FPTAS το οποίο σχεδιάστηκε από τον Lawler [61] και αποτελεί το καλύτερο δυνατό αποτέλεσμα προσέγγισης για το πρόβλημα, εκτός βέβαια και αν **P** = **NP**.

Έστω λοιπόν μία βέλτιστη διάταξη χρονοδρομολόγησης $\pi^\star$, για ένα στιγμιότυπο I του προβλήματος $1 \parallel \sum_j T_j$ και έστω $T(\pi^\star)$ η αντίστοιχη ελάχιστη συνολική καθυστέρηση. Είναι γνωστό

από προηγούμενα αποτελέσματα των Jackson [46] και Smith [92] (βλ. Ενότητα 4.1), ότι σε μία χρονοδρομολόγηση ενός συνόλου διεργασιών, όλες οι διεργασίες ολοκληρώνουν με μηδενική καθυστέρηση την εκτέλεσή τους αν και μόνο αν αυτό συμβαίνει όταν εκτελούνται με βάση τον κανόνα EDD. Επιπλέον, η διάταξη με βάση τον κανόνα EDD ελαχιστοποιεί την μέγιστη καθυστέρηση (βλ. [92]). Συμβολίζουμε με T_{EDD} και $T_{max} = \max_j T_j$, τη συνολική και τη μέγιστη καθυστέρηση της EDD διάταξης των διεργασιών του στιγμιοτύπου I. Εύκολα παρατηρούμε ότι θα ισχύει η ακόλουθη σχέση:

$$T_{max} \leq T(\pi^\star) \leq T_{EDD} \leq nT_{max}. \tag{4.9}$$

Χρησιμοποιώντας το δυναμικό πρόγραμμα (4.5) που περιγράψαμε στην Ενότητα 4.3 [60], μπορούμε να βρούμε μία βέλτιστη διάταξη χρονοδρομολόγησης ελάχιστης συνολικής καθυστέρησης, για οποιοδήποτε στιγμιότυπο του προβλήματος, σε ψευδοπολυωνυμικό χρόνο της τάξης του $O(n^4P)$, $P = \sum_j p_j$. Ακολουθώντας το συμβολισμό του δυναμικού προγράμματος, έστω $T(S,t)$ η ελάχιστη συνολική καθυστέρηση για ένα υποσύνολο S των διεργασιών του στιγμιοτύπου I, με χρόνο έναρξης εκτέλεσης t. Για κάθε δοθέν υποσύνολο S, υπάρχει μία εύκολα υπολογίσιμη χρονική στιγμή $t^\star$, τέτοια ώστε

$$T(S,t) = 0 \qquad \text{για } t \leq t^\star,$$
$$T(S,t) > 0 \qquad \text{για } t > t^\star.$$

Παρατηρούμε ότι για οποιονδήποτε ακέραιο $\Delta \geq 0$ θα ισχύει ότι $T(S, t^\star + \Delta) \geq \Delta$. Έτσι, για $\Delta = nT_{max}$ έχουμε ότι $T(S, t^\star + nT_{max}) \geq nT_{max}$ και επομένως, κατά την εκτέλεση του δυναμικού προγράμματος αρκεί να υπολογίσουμε την ποσότητα $T(S,t)$ μόνο για $t \in [t^\star, t^\star + nT_{max}]$, δηλαδή για το πολύ nT_{max} χρονικές στιγμές. Κατά συνέπεια, μπορούμε να αντικαταστήσουμε την ποσότητα P με nT_{max}, στην χρονική πολυπλοκότητα του δυναμικού προγράμματος και να ισχυριστούμε ότι η βέλτιστη διάταξη χρονοδρομολόγησης $\pi^\star$, ελάχιστης συνολικής καθυστέρησης $T(\pi^\star)$, υπολογίζεται σε χρόνο το πολύ $O(n^5 T_{max})$.

Μετασχηματίζουμε τώρα τα δεδομένα του στιγμιοτύπου χρησιμοποιώντας μία θετική σταθερά K, την οποία θα καθορίσουμε αργότερα. Για κάθε διεργασία j, περιορίζουμε την προθεσμία της σε $\overline{d}_j = d_j/K$. Επιπλέον, περιορίζουμε και στρογγυλοποιούμε προς τα κάτω το χρόνο επεξεργασίας της σε $\overline{p}_j = \lfloor p_j/K \rfloor$. Θεωρούμε, ότι εφαρμόζουμε το δυναμικό πρόγραμμα (4.5) στο μετασχηματισμένο στιγμιότυπο και έστω π_A η βέλτιστη διάταξη χρονοδρομολόγησης που προκύπτει. Έστω $\overline{T}_{\pi_A(j)}$ η καθυστέρηση της j-οστής διεργασίας στην π_A για τα μετασχηματισμένα δεδομένα και έστω $T_{\pi_A(j)}$ η αντίστοιχη καθυστέρηση ως προς τα αρχικά δεδομένα. Τότε, για κάθε διεργασία j, θα ισχύει ότι $\overline{T}_{\pi_A(j)} \leq T_{\pi_A(j)}/K$. Επιπλέον, λόγω του ότι η π_A είναι βέλτιστη για τα

μετασχηματισμένα δεδομένα, θα ισχύει ότι $\overline{T}_{\pi_A} = \sum_{j=1}^{n} \overline{T}_{\pi_A(j)} \leq T(\pi^\star)/K$. Έστω T'_{π_A} η συνολική καθυστέρηση της διάταξης π_A για χρόνους επεξεργασίας $p'_j = K\overline{p_j}$ και προθεσμία d_j, για κάθε διεργασία j. Παρατηρούμε ότι $p'_j = K\overline{p}_j \leq p_j \leq K(\overline{p}_j + 1)$. Για $T_{\pi_A} = \sum_{j=1}^{n} T_{\pi_A(j)}$, θα ισχύει ότι

$$\begin{aligned} K\overline{T}_{\pi_A} \leq T(\pi^\star) \leq T_{\pi_A} &\leq \sum_{j=1}^{n} \max\{0, K\sum_{i=1}^{j}(\overline{p}_{\pi_A(i)} + 1) - d_j\} \\ &\leq T'_{\pi_A} + \frac{Kn(n+1)}{2}. \end{aligned} \tag{4.10}$$

και

$$\begin{aligned} K\overline{T}_{\pi_A} &= K\sum_{j=1}^{n} \max\{0, \overline{p}_{\pi_A(j)} - \overline{d}_{\pi_A(j)}\} \\ &= \sum_{j=1}^{n} \max\{0, K\overline{p}_{\pi_A(j)} - d_j\} = T'_{\pi_A}. \end{aligned} \tag{4.11}$$

Από τις (4.10) και (4.11) έχουμε ότι

$$T'_{\pi_A} \leq T(\pi^\star) \leq T_{\pi_A} \leq T'_{\pi_A} + \frac{Kn(n+1)}{2}$$

απ' όπου προκύπτει η ακόλουθη σχέση.

$$T_{\pi_A} - T(\pi^\star) \leq \frac{Kn(n+1)}{2} \tag{4.12}$$

Η χρονική πολυπλοκότητα του δυναμικού προγράμματος (4.5) για το μετασχηματισμένο στιγμιότυπο του προβλήματος είναι το πολύ $O(n^5 T_{max}/K)$. Επομένως, αν επιλέξουμε το K να ισούται με $K = \frac{2\epsilon}{n(n+1)} T_{max}$ η πολυπλοκότητα χρόνου θα είναι το πολύ $O(n^7/\epsilon)$, ενώ από την (4.12) το $T_{\pi_A} - T(\pi^\star) \leq \epsilon T_{max}$. Από την (4.9) προκύπτει ότι $T_{\pi_A} - T(\pi^\star) \leq \epsilon T(\pi^\star)$. Με τον τρόπο αυτό, καταλήγουμε στο ζητούμενο FPTAS για το πρόβλημα $1 \parallel \sum_j T_j$.

Εναλλακτικά, μπορούμε να διαμορφώσουμε κατάλληλα την διαδικασία σχεδιασμού του παραπάνω FPTAS, ώστε αυτή να συμφωνεί απόλυτα με την τεχνική μετασχηματισμού των δεδομένων της εισόδου που περιγράψαμε στην Ενότητα 2.2. Συγκεκριμένα, στο Βήμα της απλοποίησης του στιγμιοτύπου I, εφαρμόζουμε τους ίδιους ακριβώς μετασχηματισμούς και καθορίζουμε το K. Στο Βήμα της επίλυσης του απλοποιημένου στιγμιοτύπου εφαρμόζουμε το δυναμικό πρόγραμμα (4.5) στο μετασχηματισμένο στιγμιότυπο και στο Βήμα της προσαρμογής χρησιμοποιούμε τις ανισότητες (4.10), (4.11) και (4.12) για να δείξουμε ότι η ελάχιστη συνολική καθυστέρηση του μετασχηματισμένου στιγμιοτύπου προσεγγίζει την ελάχιστη συνολική καθυστέρηση του αρχικού στιγμιοτύπου I με παράγοντα $1+\epsilon$. Το αποτέλεσμα αυτό παρουσιάζεται αναλυτικά στην εργασία [86] (βλ. Section

3).

4.5 Συμπεράσματα

Στο κεφάλαιο αυτό μελετήσαμε αλγορίθμους για προβλήματα χρονοδρομολόγησης τα οποία έχουν ως στόχο την ελαχιστοποίηση συγκεκριμένων συναρτήσεων απώλειας. Στην Ενότητα 4.2.1 παρουσιάσαμε ένα δυναμικό πρόγραμμα για ένα γενικό πρόβλημα χρονοδρομολόγησης, όπου κάθε διεργασία μπορεί να χρονοδρομολογηθεί με δύο διαφορετικούς τρόπους, έχοντας διαφορετικό χρόνο επεξεργασίας και σημειώνοντας διαφορετική απώλεια για καθέναν απ' αυτούς. Σε αντίθεση με προηγούμενα δυναμικά προγράμματα, όπως αυτό των Held και Karp [37], θεωρούμε ότι στην είσοδο του δυναμικού προγράμματος οι διεργασίες είναι διατεταγμένες βάσει συγκεκριμένης διάταξης. Αυτό έχει ως αποτέλεσμα το δυναμικό πρόγραμμα να βρίσκει μία βέλτιστη λύση σε ψευδοπολυωνυμικό χρόνο, αντί εκθετικού.

Στην Ενότητα 4.2.2 εφαρμόσαμε το παραπάνω δυναμικό πρόγραμμα σε τρία προβλήματα χρονοδρομολόγησης, $1 \parallel \sum_j w_j U_j$, $1 \parallel$ *maximize* $\sum_j w_j E_j$ και $1 \mid d_j = d \mid \sum_j w_j T_j$. Τα προβλήματα αυτά έχουν το εξής κοινό χαρακτηριστικό: σε μία υποτιθέμενη βέλτιστη λύση το σύνολο των διεργασιών μπορεί να διαχωριστεί σε δύο διαφορετικές ομάδες (πρόωρες και καθυστερημένες). Οι διεργασίες της μίας ομάδας εκτελούνται με βάση μία προκαθορισμένη διάταξη και οι διεργασίες της άλλης με αυθαίρετο τρόπο είτε πριν, είτε μετά από όλες τις διεργασίες της πρώτης ομάδας. Έτσι, αν αντιστοιχίσουμε την κάθε ομάδα διεργασιών σε έναν διαφορετικό τρόπο χρονοδρομολόγησης, μπορούμε να εφαρμόσουμε το δυναμικό πρόγραμμα (4.1). Η προκαθορισμένη διάταξη που αφορά τη μία από τις δύο ομάδες σε κάθε περίπτωση, αποτελεί την αρχική διάταξη των διεργασιών που απαιτείται από το δυναμικό πρόγραμμα. Με τον τρόπο αυτό προκύπτουν ψευδοπολυωνυμικοί αλγόριθμοι για καθένα από τα παραπάνω προβλήματα. Τα παραπάνω αποτελέσματα προέρχονται από την εργασία [63] των Lawler και Moore.

Στην ίδια εργασία [63] παρουσιάζονται εφαρμογές του δυναμικού προγράμματος (4.1) σε δύο ακόμη προβλήματα χρονοδρομολόγησης. Το πρώτο είναι το πρόβλημα ελαχιστοποίησης της συνολικής καθυστέρησης σε περιβάλλον Μοναδικής Μηχανής, στην περίπτωση που όλες οι διεργασίες διαθέτουν δύο κοινές προθεσμίες D και D', όπου η D είναι μία σχετική και η D' μία απόλυτη προθεσμία. Σε μία δεδομένη χρονοδρομολόγηση, η καθυστέρηση T_j μίας διεργασίας j είναι μηδέν όταν αυτή ολοκληρώνεται πριν τη σχετική προθεσμία D, ενώ είναι σταθερή και ίση με μία προκαθορισμένη τιμή q_j, όταν η j ολοκληρώνεται μετά την απόλυτη προθεσμία D'. Η T_j είναι γραμμική συνάρτηση του χρόνου , $T_j = \max\{0, t - d\}$, μόνο στην περίπτωση που η j ολοκληρώνεται μεταξύ των προθεσμιών D, D'. Η ισοδύναμη μορφή του δυναμικού προγράμματος

(4.1) που επιλύει βέλτιστα το πρόβλημα αυτό είναι όμοια με αυτή του προβλήματος μεγιστοποίησης της βεβαρημένης πρόωρης ολοκλήρωσης (βλ. Ενότητα 4.2.2), με τη μόνη διαφορά ότι η τιμή $\alpha_j(t) = \max\{0, t-d\}$ αντί για $\alpha_j(t) = w_j t$. Πρακτικές εφαρμογές του παραπάνω προβλήματος υπάρχουν στην εργασία [69] του McNaughton.

Το δεύτερο πρόβλημα στο οποίο εφαρμόζεται το δυναμικό πρόγραμμα (4.1) είναι ένα flow shop πρόβλημα (βλ. Ενότητα 1.1) δύο μηχανών όπου οι διεργασίες διαθέτουν μία κοινή προθεσμία και ο στόχος είναι η ελαχιστοποίηση του συνολικού βάρους των καθυστερημένων διεργασιών. Κάθε διεργασία j διαθέτει έναν χρόνο επεξεργασίας $p_{j,1}$ για την πρώτη μηχανή m_1 και έναν χρόνο επεξεργασίας $p_{j,2}$ για τη δεύτερη μηχανή m_2. Σε μία δεδομένη χρονοδρομολόγηση, κάθε διεργασία j ολοκληρώνει την εκτέλεσή της στη m_2, αφού προηγουμένως έχει εκτελέσει $p_{j,1}$ μονάδες χρόνου επεξεργασίας στην μηχανή m_1. Το πρόβλημα αυτό συμβολίζεται με $F2 \mid d_j = d \mid \sum_j w_j U_j$. Ομοίως με τα προηγούμενα προβλήματα, σε μία βέλτιστη διάταξη χρονοδρομολόγησης οι διεργασίες χωρίζονται σε πρόωρες και καθυστερημένες. Οι πρόωρες διεργασίες θα βρίσκονται διατεταγμένες με βάσει τον κανόνα του Johnson [47], ο οποίος επιλύει βέλτιστα σε πολυωνυμικό χρόνο το πρόβλημα $F2 \parallel C_{max}$. Η μορφή του δυναμικού προγράμματος δεν διαφέρει πολύ από την (4.2) (βλ. σελίδα 83 στο [63]).

Στην Ενότητα 4.3 παρουσιάσαμε έναν βέλτιστο ψευδοπολυωνυμικό αλγόριθμο δυναμικού προγραμματισμού για το πρόβλημα $1 \mid w_j \ agreeable \mid \sum_j w_j T_j$ [60]. Ο αλγόριθμος στηρίζεται σε συγκεκριμένα κριτήρια βελτιστοποίησης (βλ. Θεωρήματα 4.2, 4.3, 4.4) βάσει των οποίων περιορίζεται το μέγεθος του χώρου των εφικτών χρονοδρομολογήσεων που απαριθμούνται κατά την εκτέλεσή του. Οι διεργασίες διατάσσονται αρχικά με βάση τον κανόνα EDD και η εκτέλεση του δυναμικού προγράμματος γίνεται σε δύο φάσεις, την παραγωγή υποπροβλημάτων και την αναδρομική τους επίλυση. Στην έξοδο του αλγορίθμου υπολογίζεται η ελάχιστη συνολική βεβαρημένη καθυστέρηση $T(S,t)$ για τη χρονοδρομολόγηση ενός συνόλου διεργασιών S, οι οποίες αρχίζουν να εκτελούνται τη χρονική στιγμή t. Η χρονική πολυπλοκότητα του αλγορίθμου είναι της τάξης του $O(n^4P)$, $P = \sum_j p_j$. Στην Ενότητα 4.4.2 ο παραπάνω αλγόριθμος χρησιμοποιήθηκε για τον σχεδιασμό ενός FPTAS για το πρόβλημα $1 \parallel \sum_j T_j$ [61]: παρατηρούμε αρχικά ότι η πολυπλοκότητα του παραπάνω αλγορίθμου, για οποιοδήποτε στιγμιότυπο του προβλήματος, μπορεί να γίνει της τάξης του $O(n^5 T_{max})$, όπου T_{max} η μέγιστη καθυστέρηση της EDD διάταξης των διεργασιών του στιγμιοτύπου. Μετασχηματίζουμε κατάλληλα τους χρόνους επεξεργασίας και τις προθεσμίες των διεργασιών του στιγμιοτύπου και εφαρμόζουμε τον αλγόριθμο στο μετασχηματισμένο στιγμιότυπο. Στην Ενότητα 4.4.1 παρουσιάσαμε ένα ακόμα FPTAS, αυτή τη φορά για το πρόβλημα $1 \parallel \sum_j w_j U_j$ [86]. Για το σχεδιασμό του διαμορφώσαμε κατάλληλα τον βέλτιστο ψευδοπολυωνυμικό αλγόριθμο της Ενότητας 4.2.2 (βλ. δυναμικό πρόγραμμα (4.2)) και εφαρμόσαμε σ' αυτόν την

τεχνική αποκοπής (βλ. Ενότητα 3.2).

Όπως αναφέραμε στην Ενότητα 4.1, καθένα από τα παραπάνω προβλήματα αποδείχθηκε **NP**-hard υπό τη συνήθη έννοια. Ιδιαίτερα για τα προβλήματα $1 \parallel \sum_j T_j$ και $1 \mid d_j = d \mid \sum_j w_j T_j$ τα αντίστοιχα αποτελέσματα παρουσιάστηκαν σχετικά πρόσφατα (βλ. [24, 96]). Αν στα αποτελέσματα που παρουσιάσαμε, συμπεριλάβουμε και ένα πολύ πρόσφατο FPTAS για το πρόβλημα $1 \mid d_j = d \mid \sum_j w_j T_j$ [51], τότε μπορούμε να συμπεράνουμε ότι, εκτός και αν **P** = **NP**, τα προβλήματα $1 \parallel \sum_j w_j U_j$, $1 \parallel \sum_j T_j$ και $1 \mid d_j = d \mid \sum_j w_j T_j$ έχουν μελετηθεί πλήρως ως προς την προσεγγισιμότητά τους. Αντιθέτως, πολύ λίγα γνωρίζουμε για την προσεγγισιμότητα του προβλήματος $1 \parallel \sum_j w_j T_j$, για γενικά δεδομένα εισόδου. Το πρόβλημα αυτό είναι strongly **NP**-hard [60] και επομένως, εκτός και αν **P** = **NP**, η ύπαρξη ενός ψευδοπολυωνυμικού αλγορίθμου άρα και η ύπαρξη ενός FPTAS είναι αδύνατη. Το μοναδικό μη τετριμμένο αποτέλεσμα προσέγγισης, απ' όσο γνωρίζουμε, είναι ένας $(n-1)$-προσεγγιστικός αλγόριθμος των Cheng, Ng, Yuan και Liu [16] (βλ. Ενότητα 4.1). Αυτό που κάνει ιδιαίτερα δύσκολη την εύρεση μίας σταθερής προσέγγισης για το πρόβλημα είναι η επίδραση που έχουν τα βάρη των διεργασιών στην αντικειμενική συνάρτηση. Έτσι, παραμένει πολύ σημαντικό ανοιχτό ερώτημα η εύρεση ενός προσεγγιστικού αλγορίθμου με καλύτερο παράγοντα προσέγγισης.

Ένα επίσης ανοιχτό πρόβλημα είναι η εύρεση ενός FPTAS για την περίπτωση του προβλήματος $1 \parallel \sum_j w_j T_j$, όπου το πλήθος των διακεκριμένων προθεσμιών είναι σταθερό και ίσο με $m > 1$. Οι Kolliopoulos και Steiner [53] παρουσίασαν δύο βέλτιστους ψευδοπολυωνυμικούς αλγορίθμους, οι οποίοι γενικεύουν κατά μία έννοια την ιδέα του αλγορίθμου των Lawler και Moore [63] για το πρόβλημα $1 \mid d_j = d \mid \sum_j w_j T_j$. Συγκεκριμένα, παρατηρείται ότι σε μία υποτιθέμενη βέλτιστη διάταξη χρονοδρομολόγησης θα υπάρχει μία ομάδα το πολύ m διεργασιών (straddlers) καθεμία από τις οποίες αρχίζει να εκτελείται πριν από μία εκ των m προθεσμιών και ολοκληρώνει την εκτέλεσή της, είτε ακριβώς, είτε μετά από αυτήν. Οι υπόλοιπες διεργασίες, είτε πρόωρες, είτε καθυστερημένες, θα αρχίζουν και θα ολοκληρώνουν την εκτέλεσή τους εντός κάποιου από τα $m+1$ διαστήματα που ορίζουν οι m προθεσμίες. Στην ίδια εργασία, σχεδίαζεται ένα προσεγγιστικό σχήμα το οποίο προκύπτει με εφαρμογή της ιδέας του Lawler [61] που παρουσιάσαμε στην Ενότητα 4.2.2, σε έναν από τους δύο αλγόριθμους. Η χρονική πολυπλοκότητα του προσεγγιστικού σχήματος είναι ανάλογη του μέγιστου βάρους των διεργασιών (βλ. Ενότητα 4.1) και αποτελεί ένα FPTAS για το πρόβλημα μόνο στην περίπτωση που τα βάρη των διεργασιών έχουν πολυωνυμικό μέγεθος ως προς το μέγεθος της εισόδου.

Πέραν των δύο ανωτέρω ανοιχτών προβλημάτων, υπάρχουν αρκετά ακόμα προβλημάτα χρονοδρομολόγησης με στόχο την ελαχιστοποίηση αντικειμενικών συναρτήσεων όπως η $\sum_j w_j T_j$ και η $\sum_j w_j U_j$, για τα οποία παραμένει ανοιχτό ερώτημα ο καθορισμός της υπολογιστικής τους πο-

λυπλοκότητας. Ακολούθως αναφέρουμε κάποια από αυτά.

- $\mathbf{P2|p_j = p; r_j|\sum w_j T_j}$: ελαχιστοποίηση της συνολικής βεβαρημένης καθυστέρησης σε δύο παράλληλες πανομοιότυπες μηχανές με δεδομένους χρόνους αποδέσμευσης για κάθε διεργασία. Όλες οι διεργασίες διαθέτουν τον ίδιο χρόνο επεξεργασίας p.
- $\mathbf{P|p_j = p; r_j|\sum U_j}$: ελαχιστοποίηση του συνολικού πλήθους των καθυστερημένων διεργασιών σε γενικό αριθμό παράλληλων πανομοιότυπων μηχανών με δεδομένους χρόνους αποδέσμευσης και κοινό χρόνο επεξεργασίας για κάθε διεργασία.
- $\mathbf{P2|pmtn|\sum T_j}$: ελαχιστοποίηση της συνολικής καθυστέρησης σε δύο παράλληλες πανομοιότυπες μηχανές για preemptive χρονοδρομολόγηση των διεργασιών.
- $\mathbf{R2|pmtn|\sum U_j}$: ελαχιστοποίηση της συνολικού πλήθους των καθυστερημένων διεργασιών σε δύο παράλληλες μη σχετιζόμενες μηχανές για preemptive χρονοδρομολόγηση των διεργασιών.

Για επιπρόσθετα ανοιχτά προβλήματα, ο αναγνώστης παραπέμπεται στην ιστοσελίδα των Brucker και Knust [8].

Κεφάλαιο 5

Επίλογος

Μελετήσαμε προσεγγιστικά σχήματα και ψευδοπολυωνυμικούς αλγορίθμους για αρκετά προβλήματα χρονοδρομολόγησης. Η συνεισφορά των περισσοτέρων αποτελεσμάτων που παρουσιάσαμε στην εξέλιξη της Θεωρίας Χρονοδρομολόγησης είναι ιδιαίτερα σημαντική: ο σχεδιασμός προσεγγιστικών σχημάτων για τα προβλήματα $P\,|\,r_j\,|\,\sum_j w_j C_j$ [2] και $P\,||\,C_{max}$ [39] παρέμενε για αρκετά χρόνια σημαντικό ανοιχτό πρόβλημα, ενώ τα αποτελέσματα των εργασιών [63, 60] αποτέλεσαν τη βάση για τη μελέτη της προσεγγισιμότητας αρκετών προβλημάτων χρονοδρομολόγησης με δεδομένες προθεσμίες (βλ. [61, 31, 53, 51]). Οι τεχνικές και οι ιδέες στις οποίες στηρίζονται τα αποτελέσματα αυτά αποτελούν βασικά εργαλεία για το σχεδιασμό προσεγγιστικών σχημάτων στην ευρύτερη περιοχή της Συνδυαστικής Βελτιστοποίησης.

Πέραν των αποτελεσμάτων που παρουσιάσαμε, υπάρχει ένας μεγάλος αριθμός εξίσου ενδιαφέροντων αποτελεσμάτων στα οποία αναφερθήκαμε αλλά δεν περιγράψαμε λεπτομερώς. Ακολούθως επισημαίνουμε μερικά από αυτά. Για μία εκτενή επισκόπηση παραπέμπουμε τον αναγνώστη στα συγγράμματα [28, 62, 75, 26].

◊ Μελετήσαμε το πρόβλημα ελαχιστοποίησης του makespan σε παράλληλες πανομοιότυπες μηχανές. Ξεχωριστό ενδιαφέρον παρουσιάζει η μελέτη του προβλήματος σε περιβάλλον παράλληλων μη σχετιζόμενων μηχανών. Στην εργασία [86] δίνεται ένα PTAS για το πρόβλημα $R2\,||\,C_{max}$, το οποίο βασίζεται σε ένα προγενέστερο αποτέλεσμα του Potts [76]. Η βασική ιδέα για το σχεδιασμό του είναι να διαμερίσουμε κατάλληλα το χώρο των εφικτών λύσεων σε πολυωνυμικό πλήθος μικρότερων χώρων για καθέναν από τους οποίους μπορούμε εύκολα να προσεγγίσουμε το πρόβλημα. Για γενικό αριθμό μη σχετιζόμενων μηχανών, οι Lenstra, Shmoys και Tardos [66] σχεδίασαν έναν 2-προσεγγιστικό αλγόριθμο για το πρόβλημα $R\,||\,C_{max}$ και απέδειξαν ότι δεν μπορεί να προσεγγιστεί με παράγοντα μικρότερο του 3/2, εκτός και αν **P** = **NP**.

◊ Πριν το σχεδιασμό των προσεγγιστικών σχημάτων [2] για τα προβλήματα $1\,|\,r_j|\,\sum_j C_j$, $P\,|\,r_j|\,\sum_j w_jC_j$ και $P\,|\,r_j,\ pmtn|\,\sum_j w_jC_j$, είχαν προταθεί αρκετοί προσεγγιστικοί αλγόριθμοι. Κάποιοι από αυτούς, όπως ο $(\frac{e}{e-1})$-προσεγγιστικός αλγόριθμος για το πρόβλημα $1\,|\,r_j|\,\sum_j C_j$ [15] και ο 2-προσεγγιστικός αλγόριθμος για το πρόβλημα $P\,|\,r_j|\,\sum_j w_jC_j$ [83], συνεχίζουν να αποτελούν πρώτη επιλογή λόγω της απλότητας και της αποδοτικότητάς τους. Στην εργασία [12] παρουσιάζεται ένα PTAS για το πρόβλημα $Q\,|\,r_j|\,\sum_j w_jC_j$. Το αποτέλεσμα αυτό προκύπτει με παρόμοιο τρόπο με το αντίστοιχο αποτέλεσμα για το πρόβλημα $P\,|\,r_j|\,\sum_j w_jC_j$.

◊ Περιγράψαμε έναν ψευδοπολυωνυμικό αλγόριθμο δυναμικού προγραμματισμού για το πρόβλημα $1\,|\,d_j = d|\,\sum_j w_jT_j$ [63]. Προσφάτως οι Kellerer και Strusevich [51], τροποποίησαν κατάλληλα τον παραπάνω αλγόριθμο και τον χρησιμοποίησαν για το σχεδιασμό ενός FPTAS για το πρόβλημα. Οι Kolliopoulos και Steiner σχεδίασαν ψευδοπολυωνυμικούς αλγορίθμους για την πιο γενική εκδοχή του προβλήματος, όπου το πλήθος των διακεκριμένων προθεσμιών είναι σταθερό και ίσο με $m > 1$ [53]. Οι αλγόριθμοι αυτοί γενικεύουν κατά μία έννοια την ιδέα του αλγορίθμου των Lawler και Moore [63] για το πρόβλημα $1\,|\,d_j = d|\,\sum_j w_jT_j$, για περισσότερους του ενός straddlers.

◊ Υπάρχουν πολλά ενδιαφέροντα αποτελέσματα μη προσέγγισης τα οποία προκύπτουν μέσω κατάλληλων αναγωγών διατήρησης προσέγγισης (βλ. Α.2.1). Στην εργασία [41] των Hoogeveen, Schuurman και Woeginger παρουσιάζονται τα περισσότερα μη τετριμμένα γνωστά αποτελέσμα μη προσέγγισης, τα οποία κατανέμουν αρκετά προβλήματα χρονοδρομολόγησης στην κλάση των **APX**-hard προβλημάτων.

Στις Ενότητες 2.5, 3.4 και 4.5 αναφέραμε αρκετά ανοιχτά προβλήματα που αφορούν στη βελτιστοποίηση των επικείμενων αντικειμενικών συναρτήσεων. Στη συνέχεια επισημαίνουμε τα σημαντικότερα από αυτά.

♦ $\mathbf{R}\,\|\,\mathbf{C_{max}}$. Παραμένει σημαντικό ανοιχτό ερώτημα αν υπάρχει προσέγγιση με παράγοντα $\rho \in (3/2, 2)$. Το καλύτερο γνωστό αποτέλεσμα προσέγγισης είναι ο 2-προσεγγιστικός αλγόριθμος των Lenstra, Shmoys και Tardos [66], ενώ από την ίδια εργασία [66] γνωρίζουμε ότι το πρόβλημα δεν μπορεί να προσεγγιστεί με παράγοντα μικρότερο από 3/2, εκτός και αν **P** = **NP**.

♦ $\mathbf{1\,|prec|\,\sum_j w_jC_j}$. Η μείωση του χάσματος προσεγγισιμότητας του προβλήματος είναι ένα από τα σημαντικότερα ανοιχτά προβλήματα της Θεωρίας Χρονοδρομολόγησης. Μέχρι σήμερα έχουν δοθεί αρκετοί 2-προσεγγιστικοί αλγόριθμοι για το πρόβλημα (βλ. [36, 14, 17]), ενώ παραμένει άγνωστο αν ανήκει στην κλάση των **APX**-hard προβλημάτων. Οι Ambühl

και Mastrolilli [3] απέδειξαν πρόσφατα ότι πρόκειται για ειδική περίπτωση του προβλήματος Vertex Cover για το οποίο είναι γνωστό ότι δεν μπορεί να προσεγγιστεί με παράγοντα μικρότερο από $10\sqrt{5} - 21 \approx 1.36$ [45], ενώ εικάζεται ισχυρά ότι δεν επιδέχεται $(2 - \delta)$-προσεγγιστικό αλγόριθμο πολυωνυμικού χρόνου, για μικρό $\delta > 0$. Θα είχε ίσως ενδιαφέρον να εξετάσουμε αν με τη βοήθεια αυτών των αποτελεσμάτων μπορούμε να οδηγηθούμε σε κάτι αντίστοιχο για το πρόβλημα $1 \mid prec \mid \sum_j w_j C_j$.

- $\mathbf{1 \parallel \sum_j w_j T_j}$. Το μοναδικό μη τετριμμένο αποτέλεσμα προσέγγισης, απ' όσο γνωρίζουμε, είναι ένας $(n-1)$-προσεγγιστικός αλγόριθμος των Cheng, Ng, Yuan και Liu [16]. Παραμένει πολύ σημαντικό ανοιχτό ερώτημα η βελτίωση του παράγοντα προσέγγισης. Ακόμα και στην περίπτωση όπου το πλήθος των διακεκριμένων προθεσμιών είναι σταθερό η εύρεση ενός FPTAS αποτελεί επίσης ανοιχτό πρόβλημα. Οι Kolliopoulos και Steiner [53] σχεδίασαν ένα πλήρες προσεγγιστικό σχήμα το οποίο όμως έχει χρονική πολυπλοκότητα ανάλογη του μέγιστου βάρους των διεργασιών. Επομένως, αποτελεί ένα FPTAS για το πρόβλημα μόνο στην περίπτωση που τα βάρη των διεργασιών φράσσονται άνω από ένα πολυώνυμο του μεγέθους της εισόδου.

Παράρτημα Α

Θεωρία Πολυπλοκότητας

Ο κύριος στόχος της παρούσας εργασίας είναι να διερευνήσει κάποια από τα κεντρικά προβλήματα της Θεωρίας Χρονοδρομολόγησης για τα οποία μέχρι σήμερα δεν είναι γνωστός κάποιος αποδοτικός αλγόριθμος που να τα επιλύει. Ο όρος "αποδοτικός αλγόριθμος" ερμηνεύεται ως ένας αλγόριθμος που απαιτεί πολυωνυμικό αριθμό βημάτων ως προς το μέγεθος της εισόδου. Για την κατανόηση των αποτελεσμάτων που παρουσιάζονται, παραθέτουμε στις δύο επόμενες ενότητες τις απαραίτητες, βασικές, έννοιες από τη Θεωρία Πολυπλοκότητας.

Α.1 Αναγωγές και NP−πληρότητα

Στη Θεωρία Πολυπλοκότητας ενδιαφερόμαστε κυρίως για προβλήματα στα οποία η απάντηση είναι είτε "ΝΑΙ" είτε "ΟΧΙ". Ένα πρόβλημα αυτής της μορφής ονομάζεται *πρόβλημα απόφασης*. Ένα απλό παράδειγμα προβλήματος απόφασης είναι το πρόβλημα SAT: Δίνεται λογική πρόταση φ σε κανονική συζευκτική μορφή και ρωτάμε αν υπάρχει μία ανάθεση αληθοτιμών που να ικανοποιεί την φ.

Μας διευκολύνει να ταυτίζουμε κάθε πρόβλημα απόφασης A με ένα σύνολο πεπερασμένων δυαδικών συμβολοσειρών, καθεμία από τις οποίες αποτελεί την δυαδική αναπαράσταση των δεδομένων κάποιας εισόδου (στιγμιοτύπου) του προβλήματος και για τις οποίες η απάντηση είναι "ΝΑΙ". Συμβολίζουμε το μήκος (ή μέγεθος) μιας συμβολοσειράς εισόδου x, με $|x|$.

Ένας αλγόριθμος B για ένα πρόβλημα απόφασης δέχεται ως είσοδο μία συμβολοσειρά x και επιστρέφει την τιμή "ΝΑΙ" ή "ΟΧΙ", την οποία συμβολίζουμε με $B(x)$. Λέμε ότι ο B επιλύει το πρόβλημα A, αν για κάθε συμβολοσειρά εισόδου x ισχύει ότι, $B(x) = \text{ΝΑΙ}$ αν και μόνο αν $x \in A$. Ο αλγόριθμος B είναι *πολυωνυμικού χρόνου* αν υπάρχει πολυωνυμική συνάρτηση p τέτοια που για κάθε συμβολοσειρά εισόδου, ο B τερματίζει σε το πολύ $O(p(|x|))$ βήματα. Το σύνολο όλων των

προβλημάτων για τα οποία υπάρχει αλγόριθμος πολυωνυμικού χρόνου που τα επιλύει ορίζεται ως το σύνολο **P**.

Ο αλγόριθμος B, όπως ορίστηκε προηγουμένως, επιδιώκει να λύσει αποδοτικά ένα πρόβλημα A. Ανεξάρτητα όμως από το αν ένα πρόβλημα μπορεί να λυθεί αποδοτικά είναι σημαντικό να μπορούμε να πιστοποιούμε με αποδοτικό τρόπο μία φερόμενη ως λύση του προβλήματος. Ένας "αλγόριθμος πιστοποίησης" για ένα πρόβλημα A έχει διαφορετική δομή από έναν αλγόριθμο που επιδιώκει να λύσει αποδοτικά το πρόβλημα: για την πιστοποίηση μιας λύσης, εκτός από την συμβολοσειρά εισόδου x, χρειάζεται και μία συμβολοσειρά y, η οποία καλείται *πιστοποιητικό (certificate)* και εμπεριέχει την ένδειξη ότι η x είναι ένα "ΝΑΙ" στιγμιότυπο του προβλήματος A. Ο αλγόριθμος πιστοποίησης είναι ένας πολυωνυμικού χρόνου αλγόριθμος.

Ορισμός Α.1 *Ένα πρόβλημα $A \in$ **NP** αν υπάρχει μία πολυωνυμική συνάρτηση p και ένας αλγόριθμος πιστοποίησης B, έτσι ώστε για κάθε συμβολοσειρά εισόδου x:*

- *$x \in A$ αν και μόνο αν υπάρχει πιστοποιητικό y, με $|y| \leq p(|x|)$, έτσι ώστε $B(x, y) =$ "ΝΑΙ".*

Από τον Ορισμό Α.1 είναι σημαντικό να παρατηρήσουμε ότι για κάθε "ΝΑΙ" στιγμιότυπο x, υπάρχει κάποιο πιστοποιητικό y που να το αποδεικνύει, ενώ για στιγμιότυπα με αρνητική απάντηση αγνοούμε την ύπαρξη ενός τέτοιου πιστοποιητικού. Επιπλέον, μπορεί να μην γνωρίζουμε πώς να βρίσκουμε ένα τέτοιο πιστοποιητικό σε πολυωνυμικό χρόνο, όμως είμαστε σίγουροι ότι υπάρχει αν η απάντηση για ένα στιγμιότυπο x είναι "ΝΑΙ". Για παράδειγμα στο πρόβλημα SAT, το πιστοποιητικό της λογικής πρότασης φ είναι μία ανάθεση αληθοτιμών T που ικανοποιεί την φ. Η T υπάρχει αν και μόνο αν η πρόταση φ είναι ικανοποιήσιμη. Είναι επίσης εύκολο να δούμε ότι, ισοδύναμα, ένα πρόβλημα απόφασης ανήκει στο **NP** αν υπάρχει μη ντετερμινιστικός αλγόριθμος πολυωνυμικού χρόνου που το αποφασίζει. Χωρίς να μπούμε σε λεπτομέρειες, ένας τέτοιος αλγόριθμος μπορεί να πάρει τυχαίες αποφάσεις, και επιπλέον, δικαιούται κάποιες φορές να απαντήσει "ΟΧΙ" όταν η απάντηση είναι "ΝΑΙ", αλλά δεν δικαιούται να απαντήσει "ΝΑΙ" όταν η απάντηση είναι "ΟΧΙ".

Εύκολα προκύπτει από τους ορισμούς ότι αν ένα πρόβλημα A ανήκει στο **P**, θα ανήκει και στο **NP**, δηλαδή $\mathbf{P} \subseteq \mathbf{NP}$, αφού αν μας δοθεί μία είσοδος x και ένα πιστοποιητικό y μπορούμε να υπολογίσουμε πολυωνυμικά την λύση του x και να απαντήσουμε αν $x \in A$ αγνοώντας το y.

Για να συγκρίνουμε δύο προβλήματα απόφασης ως προς τη δυσκολία τους χρησιμοποιούμε την μέθοδο της αναγωγής.

Ορισμός Α.2 *Ένα πρόβλημα A_1 ανάγεται σε ένα πρόβλημα A_2 ($A_1 \leq A_2$) αν υπάρχει συνάρτηση f τέτοια ώστε για κάθε συμβολοσειρά εισόδου x:*

- *$x \in A_1$ αν και μόνο αν $f(x) \in A_2$.*

Η f ονομάζεται *αναγωγή* του A_1 στο A_2 και διαισθητικά επιβεβαιώνει πως αν το πρόβλημα A_2 είναι "εύκολο" τότε και το A_1 είναι "εύκολο", ενώ αν το A_1 είναι "δύσκολο" τότε και το A_2 είναι "δύσκολο". Συνήθως, όταν ένα πρόβλημα A_1 ανάγεται σε ένα πρόβλημα A_2, λέμε ότι "Το A_2 είναι τουλάχιστον εξίσου δύσκολο με το A_1 ".

Αν επιπλέον η f υπολογίζεται από έναν αλγόριθμο σε χώρο $O(\log|x|)$ τότε θα υπολογίζεται και σε πολυωνυμικό χρόνο[1] και ονομάζεται *πολυωνυμική αναγωγή*. Είναι εμφανές πως αν $A_1 \leq A_2$ και το A_2 αποφασίζεται σε πολυωνυμικό χρόνο, τότε και το A_1 αποφασίζεται σε πολυωνυμικό χρόνο.

Ένα πρόβλημα A είναι **NP**-*hard* αν, για κάθε πρόβλημα $A' \in$ **NP**, $A' \leq A$. Το A είναι **NP**-*πλήρες (***NP**-*complete)* αν είναι **NP**-hard και επιπλέον $A \in$ **NP**. Ένα **NP**-πλήρες πρόβλημα είναι το δυσκολότερο πρόβλημα της κλάσης **NP**, υπό την έννοια ότι εάν υπήρχε πολυωνυμικού χρόνου αλγόριθμος γι' αυτό, θα είχαμε, μέσω μιας πολυωνυμικής αναγωγής, έναν πολυωνυμικού χρόνου αλγόριθμο για οποιοδήποτε πρόβλημα στο **NP**, οπότε θα ίσχυε ότι **P** = **NP**. Κάτι τέτοιο όμως δεν μοιάζει πιθανό με βάση τις σημερινές μας γνώσεις.

Όπως αναφέραμε αρχικά, μας διευκολύνει να θεωρούμε τη δυαδική αναπαράσταση των δεδομένων της εισόδου για κάθε πρόβλημα. Η δυαδική αναπαράσταση αποτελεί πρότυπη αναπαράσταση δεδομένων και χρησιμοποιείται από όλους τους ηλεκτρονικούς υπολογιστές. Παρόλα αυτά, υπάρχουν πολλές αποδεκτές αναπαραστάσεις οι οποίες σχετίζονται πολυωνυμικά μεταξύ τους. Αν θεωρήσουμε για παράδειγμα ένα οποιοδήποτε γραφοθεωρητικό πρόβλημα όπου η είσοδος του αποτελείται από ένα γράφημα G με n το πλήθος κορυφές, καμία από τις οποίες δεν είναι απομονωμένη (δηλαδή ο βαθμός της είναι τουλάχιστον 1), τότε η αναπαράσταση του G με πίνακα γειτνίασης (adjacency matrix) απαιτεί χώρο μνήμης $O(n^2)$, ενώ αν χρησιμοποιήσουμε λίστα γειτνίασης ο χώρος μνήμης που απαιτείται μειώνεται σε $O(n)$. Ένας ακόμη τρόπος να αναπαραστήσουμε τα δεδομένα της εισόδου ενός προβλήματος, είναι η unary αναπαράσταση, για παράδειγμα ο αριθμός 9 αναπαρίσταται ως "IIIIIIIII", η οποία απαιτεί εκθετικά περισσότερα σύμβολα σε σχέση με τη δυαδική αναπαράσταση.

Είναι σημαντικό να παρατηρήσουμε, ότι οι έννοιες της επιλυσιμότητας σε πολυωνυμικό χρόνο και της **NP**-πληρότητας για ένα πρόβλημα, είναι άμεσα συνδεδεμένες με τον τρόπο αναπαράστασης των δεδομένων της εισόδου του. Αν αλλάξουμε την αναπαράσταση από δυαδική σε unary, τότε το πρόβλημα μπορεί να γίνει ευκολότερο λόγω του ότι η είσοδος μεγαλώνει και επομένως, οι απαιτήσεις σε χρονική πολυπλοκότητα από έναν αλγόριθμο πολυωνυμικού χρόνου γίνονται λιγότερο αυστηρές. Ένα πρόβλημα που παραμένει **NP**-hard όταν τα δεδομένα του αναπαρίστανται σε unary μορφή καλείται *strongly* **NP**-*hard*. Ένα πρόβλημα που επιλύεται σε πολυωνυμικό χρό-

[1] δεν γνωρίζουμε αναγωγή που να υλοποιείται σε πολυωνυμικό χρόνο και όχι σε λογαριθμικό χώρο.

νο, στα πλαίσια μιας unary αναπαράστασης των δεδομένων της εισόδου του, λέμε ότι επιλύεται ψευδοπολυωνυμικά και ο αντίστοιχος αλγόριθμος που το επιλύει καλείται *ψευδοπολυωνυμικός αλγόριθμος*. Προβλήματα τα οποία επιλύονται ψευδοπολυωνυμικά και είναι **NP**-hard, στα πλαίσια μίας δυαδικής αναπαράστασης των δεδομένων, συνηθίζεται να καλούνται απλώς **NP**-hard ή **NP**-hard *υπό τη συνήθη έννοια*. Αν υποθέσουμε ότι ένα strongly **NP**-hard πρόβλημα επιδέχεται ψευδοπολυωνυμικό αλγόριθμο, τότε όλα τα προβλήματα στο **NP** θα επιδέχονται αλγορίθμους πολυωνυμικού χρόνου και επομένως θα ισχύει ότι **P** = **NP**.

Για μία εκτενή διερεύνηση της κλάσης **NP** και γενικότερα της Θεωρίας Πολυπλοκότητας ο αναγνώστης παραπέμπεται στα βιβλία των Garey & Johnson [30] και Papadimitriou [73].

Α.2 Προσεγγιστικοί αλγόριθμοι

Τα προβλήματα χρονοδρομολόγησης, όπως και τα περισσότερα πρακτικά προβλήματα, εντάσσονται σε μία ευρύτερη κατηγορία προβλημάτων που ονομάζονται *προβλήματα βελτιστοποίησης (optimization problems)*. Ένα πρόβλημα βελτιστοποίησης προσδιορίζεται από ένα σύνολο στιγμιοτύπων (ή εισόδων) $\mathcal{I}$, από ένα μη κενό σύνολο εφικτών λύσεων SOL(I) για κάθε $I \in \mathcal{I}$ και από μία αντικειμενική συνάρτηση c, που αντιστοιχεί σε κάθε εφικτή λύση f του SOL(I) μία τιμή (ή κόστος) $c(f)$. Στο εξής θα θεωρούμε προβλήματα βελτιστοποίησης όπου όλες οι εφικτές λύσεις έχουν μη-αρνητικό κόστος.

Τα προβλήματα βελτιστοποίησης διακρίνονται σε *προβλήματα ελαχιστοποίησης (minimization problems)*, όπου η βέλτιστη λύση είναι η εφικτή λύση με τη μικρότερη δυνατή τιμή, και σε *προβλήματα μεγιστοποίησης (maximization problems)*, όπου η βέλτιστη λύση είναι η εφικτή λύση με τη μεγαλύτερη δυνατή τιμή. Σε κάθε περίπτωση θα συμβολίζουμε την τιμή της βέλτιστης λύσης ενός στιγμιοτύπου (εισόδου) I με OPT(I) και το μέγεθος αυτού (δηλαδή τον αριθμό των bits που χρειάζονται για τη δυαδική αναπαράσταση των δεδομένων του στιγμιοτύπου) με $|I|$.

Ένα πρόβλημα βελτιστοποίησης το οποίο ανήκει στο **P** ορίζεται ως ένα πρόβλημα για το οποίο υπάρχει πολυωνυμικού χρόνου αλγόριθμος A ο οποίος, για κάθε στιγμιότυπο $I \in \mathcal{I}$ επιστρέφει μία βέλτιστη λύση $f^* \in$ SOL(I) η οποία έχει τιμή $c(f^*) =$ OPT(I). Δύο πολύ γνωστά, θεμελιώδη, προβλήματα βελτιστοποίησης τα οποία έχει αποδειχθεί ότι ανήκουν στο **P** είναι το πρόβλημα Shortest Path: δίνεται γράφημα G με ένα κόστος σε κάθε ακμή του και δύο κορυφές u, v και ζητάμε να βρούμε το μονοπάτι ελαχίστου κόστους από την u στην v και το Minimum Spanning Tree πρόβλημα: δίνεται γράφημα G με κάποιο κόστος σε κάθε ακμή του και ζητάμε να βρούμε το επικαλύπτον δένδρο (spanning tree) με το ελάχιστο άθροισμα κόστους ακμών. Παρόλα αυτά τα περισσότερα ενδιαφέροντα προβλήματα βελτιστοποίησης είναι υπολογιστικά δύσκολα.

Ορισμός Α.3 *Ένα πρόβλημα βελτιστοποίησης* Π *ανήκει στην κλάση* **NP** *αν ισχύουν τα ακόλουθα:*

1. *Το σύνολο των στιγμιοτύπων I αναγνωρίζεται σε πολυωνυμικό χρόνο.*
2. *Για κάθε εφικτή λύση $f \in \mathrm{SOL}(I)$, ισχύει ότι $|f| \leq p(|I|)$, όπου $|f|$ είναι το μέγεθος της λύσης f και p μία πολυωνυμική συνάρτηση. Επιπλέον, υπάρχει πολυωνυμικού χρόνου αλγόριθμος που αποφασίζει αν $f \in \mathrm{SOL}(I)$.*
3. *Η αντικειμενική συνάρτηση c υπολογίζεται σε πολυωνυμικό χρόνο.*

Το αντίστοιχο πρόβλημα απόφασης ενός προβλήματος βελτιστοποίησης Π της κλάσης **NP**, αποτελείται από το εκάστοτε στιγμιότυπο I του προβλήματος και από έναν ρητό αριθμό B, ο οποίος συνηθίζεται να καλείται προϋπολογισμός (budget). Αν το Π είναι πρόβλημα ελαχιστοποίησης (μεγιστοποίησης) τότε η απάντηση στο αντίστοιχο πρόβλημα απόφασης είναι "ΝΑΙ" αν και μόνο αν υπάρχει μία εφικτή λύση f για το I, κόστους $c(f) \leq B$ ($\geq B$). Επομένως, ένας πολυωνυμικού χρόνου αλγόριθμος για το Π μπορεί να επιλύσει και το αντίστοιχο πρόβλημα απόφασης, υπολογίζοντας αρχικά την τιμή της βέλτιστης λύσης και συγκρίνοντάς την με το B. Αντιθέτως, για να αποδείξουμε την υπολογιστική δυσκολία ενός προβλήματος βελτιστοποίησης που ανήκει στο **NP** αρκεί να δείξουμε ότι το αντίστοιχο πρόβλημα απόφασης είναι **NP**-hard.

Όπως αναφέραμε και στην προηγούμενη ενότητα, η εύρεση ενός αλγορίθμου πολυωνυμικού χρόνου για οποιοδήποτε **NP**-hard πρόβλημα θα αποδείκνυε ότι **P** = **NP**. Μέχρι σήμερα, οι γρηγορότεροι αλγόριθμοι που έχουν παρουσιαστεί για **NP**-hard προβλήματα βελτιστοποίησης, βρίσκουν τη βέλτιστη λύση σε χρόνο εκθετικό ως προς το μέγεθος της εισόδου. Το γεγονός αυτό έστρεψε το ενδιαφέρον των ερευνητών στην επινόηση προσεγγιστικών, έναντι των βέλτιστων, αλγορίθμων. Οι προσεγγιστικοί αλγόριθμοι βρίσκουν μία εφικτή λύση η οποία πλησιάζει σε τιμή τη βέλτιστη και επιπλέον είναι αποδοτικοί όσον αφορά το χρόνο εκτέλεσής τους. Το κύριο μέτρο της ποιότητας ενός προσεγγιστικού αλγορίθμου είναι η ίδια η προσέγγιση την οποία επιτυγχάνει και όχι η χρονική του πολυπλοκότητα, αφού είναι σύνηθες πλέον όταν αναφερόμαστε σε έναν προσεγγιστικό αλγόριθμο να υπονοούμε έναν αλγόριθμο πολυωνυμικού χρόνου. Για να γίνουμε πιο συγκεκριμένοι δίνουμε τον τυπικό ορισμό του προσεγγιστικού αλγορίθμου.

Ορισμός Α.4 *Θεωρούμε ένα πρόβλημα ελαχιστοποίησης (αντίστοιχα μεγιστοποίησης)* Π *και έστω συνάρτηση $\rho : \mathbb{Z}^+ \to \mathbb{Q}^+$, με $\rho \geq 1$ (αντίστοιχα $\rho \leq 1$). Ένας αλγόριθμος A καλείται ρ-προσεγγιστικός αλγόριθμος για το πρόβλημα* Π*, αν για κάθε στιγμιότυπο I του* Π *βρίσκει μία εφικτή λύση f κόστους $c(f)$ τέτοια ώστε,*

- $c(f) \leq \rho(|I|) \cdot \mathrm{OPT}(I)$, *αν το* Π *είναι πρόβλημα ελαχιστοποίησης.*
- $c(f) \geq \rho(|I|) \cdot \mathrm{OPT}(I)$, *αν το* Π *είναι πρόβλημα μεγιστοποίησης.*

Η τιμή $\rho(|I|)$ καλείται παράγοντας (ή λόγος) προσέγγισης του προσεγγιστικού αλγορίθμου A. Επιπλέον, θεωρούμε ότι ο A είναι ένας αλγόριθμος πολυωνυμικού χρόνου.

Η τιμή της ρ αποτελεί το κύριο μέτρο της ποιότητας ενός προσεγγιστικού αλγορίθμου και όσο πιο κοντά στο 1 είναι αυτή τόσο καλύτερος είναι ο αλγόριθμος που σχεδιάστηκε. Για παράδειγμα, αν η τιμή της ρ είναι 0 για ένα πρόβλημα μεγιστοποίησης ή 10^5 για ένα πρόβλημα ελαχιστοποίησης, τότε δεν μπορούμε να μιλάμε για ποιοτικούς προσεγγιστικούς αλγορίθμους. Το καλύτερο που μπορεί κάποιος να ελπίζει για ένα **NP**-hard πρόβλημα βελτιστοποίησης, είναι ένας παράγοντας προσέγγισης $1+\epsilon$ αν πρόκειται για πρόβλημα ελαχιστοποίησης ή $1-\epsilon$ αν πρόκειται για πρόβλημα μεγιστοποίησης, για κάθε σταθερά $0 < \epsilon < 1$.

Ορισμός Α.5 *Θεωρούμε ένα πρόβλημα ελαχιστοποίησης (αντίστοιχα μεγιστοποίησης)* Π.

- *Μία οικογένεια $\mathcal{A}_\epsilon$ από $(1+\epsilon)$-προσεγγιστικούς αλγορίθμους ($(1-\epsilon)$-προσεγγιστικούς αλγορίθμους) για το πρόβλημα* Π*, για κάθε σταθερά $0 < \epsilon < 1$, αποτελεί ένα προσεγγιστικό σχήμα (approximation scheme) για το* Π.
- *Ένα προσεγγιστικό σχήμα πολυωνυμικού χρόνου (polynomial time approximation scheme ή* PTAS*) για ένα πρόβλημα* Π *είναι ένα προσεγγιστικό σχήμα με πολυπλοκότητα χρόνου πολυωνυμική ως προς το μέγεθος της εισόδου.*
- *Ένα πλήρες πολυωνυμικό προσεγγιστικό σχήμα (fully polynomial time approximation scheme ή* FPTAS*) για ένα πρόβλημα* Π *είναι ένα προσεγγιστικό σχήμα με πολυπλοκότητα χρόνου πολυωνυμική, τόσο ως προς το μέγεθος της εισόδου, όσο και ως προς το $\frac{1}{\epsilon}$.*

Ένα PTAS με πολυπλοκότητα χρόνου $O(|I|^{3/\epsilon})$ είναι αποδεκτό, παρότι εκθετικό ως προς το $1/\epsilon$, αφού είναι πολυωνυμικό ως προς το μέγεθος της εισόδου $|I|$. Ένα FPTAS αντίθετα, απαγορεύεται να έχει πολυπλοκότητα χρόνου εκθετική ως προς το $\frac{1}{\epsilon}$, ενώ επιτρέπεται η πολυπλοκότητα χρόνου να είναι της τάξης $O(|I|^3(1/\epsilon^4))$. Η απόκτηση ενός FPTAS για ένα **NP**-hard πρόβλημα βελτιστοποίησης αποτελεί και το καλύτερο δυνατό αποτέλεσμα (εκτός και αν **P** = **NP**) για το πρόβλημα αυτό. Όπως θα δούμε στη συνέχεια, υπάρχουν προβλήματα για τα οποία είναι αδύνατο να αποκτηθεί ένα PTAS, εκτός και αν **P** = **NP**, όμως είναι εφικτό ένα ασυμπτωτικό προσεγγιστικό σχήμα.

Ορισμός Α.6 *Μία οικογένεια αλγορίθμων $\mathcal{A}_\epsilon$ αποτελεί ένα ασυμπτωτικό πολυωνυμικό προσεγγιστικό σχήμα για ένα πρόβλημα ελαχιστοποίησης* Π*, αν για κάθε $\epsilon > 0$ υπάρχει σταθερά $N > 0$ και ένας πολυωνυμικού χρόνου αλγόριθμος $\mathcal{B}$ της οικογένειας $\mathcal{A}_\epsilon$, ο οποίος πετυχαίνει παράγοντα προσέγγισης $1+\epsilon$ για κάθε στιγμιότυπο του* Π *για το οποίο* OPT $\geq N$.

Ένα ασυμπτωτικό πολυωνυμικό προσεγγιστικό σχήμα συμβολίζεται συνήθως ως PTAS$^{\infty}$. Αν επιπλέον, η πολυπλοκότητα χρόνου είναι πολυωνυμική ως προς το $1/\epsilon$, τότε ο συμβολισμός είναι FPTAS$^{\infty}$. Προφανώς υπάρχει ανάλογος του Ορισμού Α.6 για προβλήματα μεγιστοποίησης.

Όλοι οι ορισμοί που διατυπώθηκαν παραπάνω παρουσιάζονται στα συγγράμματα [94], [27]. Στα ίδια συγγράμματα, παρουσιάζονται αποτελέσματα προσέγγισης για πολλά σημαντικά προβλήματα βελτιστοποίησης.

Α.2.1 Αποτελέσματα μη προσέγγισης

Παρόλη την ποικιλία τεχνικών που σχεδιάστηκαν και σχεδιάζονται για την απόκτηση ποιοτικών προσεγγιστικών αλγορίθμων, υπάρχουν προβλήματα για τα οποία η εύρεση ενός καλού προσεγγιστικού αλγορίθμου δείχνει εξαιρετικά δύσκολη. Η εξέλιξη της θεωρίας Πολυπλοκότητας τα τελευταία χρόνια απέδειξε, για αρκετά **NP**-hard προβλήματα βελτιστοποίησης, πως το να επιτύχουμε έναν συγκεκριμένο παράγοντα προσέγγισης είναι εξίσου δύσκολο με το να υπολογίσουμε την βέλτιστη λύση γι᾽ αυτά. Στην ενότητα αυτή παρουσιάζουμε κάποιες βασικές τεχνικές που εφαρμόζονται για την απόδειξη τέτοιων αποτελεσμάτων. Στην Υποενότητα Α.2.1.1 ασχολούμαστε με την απόδειξη της μη ύπαρξης ενός FPTAS, ενώ στην Υποενότητα Α.2.1.2 παρουσιάζουμε δύο τεχνικές που αποδεικνύουν την μη ύπαρξη ενός PTAS για κάποιο **NP**-hard πρόβλημα βελτιστοποίησης. Τέλος, δίνουμε μία γενική εικόνα για το πώς ιεραρχούνται τα **NP**-hard προβλήματα βελτιστοποίησης βάσει της προσεγγισιμότητας που αυτά επιδέχονται.

Α.2.1.1 Πώς αποδεικνύουμε ότι δεν υπάρχει FPTAS

Η μη ύπαρξη ενός FPTAS για ένα πρόβλημα βελτιστοποίησης είναι άμεσα συνδεδεμένη με τη διαπίστωση ότι πρόκειται για ένα strongly **NP**-hard πρόβλημα. Τα περισσότερα γνωστά **NP**-hard προβλήματα, συμπεριλαμβανομένων και των προβλημάτων χρονοδρομολόγησης, είναι strongly **NP**-hard. Στην Ενότητα Α.2 είχαμε παρατηρήσει ότι αν $\mathbf{P} \neq \mathbf{NP}$, τότε ένα strongly **NP**-hard πρόβλημα δεν επιδέχεται ψευδοπολυωνυμικό αλγόριθμο. Βάσει αυτής της παρατήρησης και υπό κάποιους ασθενείς περιορισμούς, μπορούμε να αποδείξουμε ότι κάθε **NP**-hard πρόβλημα το οποίο επιδέχεται ένα FPTAS θα πρέπει να επιδέχεται έναν ψευδοπολυωνυμικό αλγόριθμο, εκτός και αν $\mathbf{P} = \mathbf{NP}$. Η παρατήρηση αυτή αποσαφηνίζεται στο Θεώρημα Α.1 (βλ. Chapter 8 από [94]).

Θυμίζουμε ότι με $|I|$ συμβολίζουμε το μέγεθος της εισόδου I ενός προβλήματος βελτιστοποίησης, δεδομένης μίας δυαδικής αναπαράστασης αυτής. Όταν η είσοδος αναπαρίσταται σε unary μορφή θα συμβολίζουμε το αντίστοιχο μέγεθος αυτής με $|I|_{unary}$.

Θεώρημα Α.1 *Θεωρούμε μία πολυωνυμική συνάρτηση p και ένα* **NP***-hard πρόβλημα ελαχιστοποίησης ή μεγιστοποίησης* Π *τέτοιο ώστε:*

1. *Κάθε εφικτή λύση οποιουδήποτε στιγμιοτύπου I έχει ακέραιο κόστος.*

2. $\mathrm{OPT}(I) \leq p(|I|_{unary})$ *για κάθε στιγμιότυπο* I.

Αν το Π *επιδέχεται* FPTAS, *τότε επιδέχεται και ψευδοπολυωνυμικό αλγόριθμο.*

Απόδειξη. Έστω Π πρόβλημα ελαχιστοποίησης. Υποθέτουμε ότι υπάρχει FPTAS για το Π. Η χρονική πολυπλοκότητα αυτού θα είναι πολυωνυμική ως προς τα $|I|$ και $1/\epsilon$, για παράμετρο $\epsilon > 0$.

Στο στιγμιότυπο I θέτουμε $\epsilon = 1/p(|I|_{unary})$ και εκτελούμε το FPTAS. Βάσει του Ορισμού Α.4 η λύση που παράγεται θα έχει κόστος μικρότερο ή ίσο από

$$(1+\epsilon)\,\mathrm{OPT}(I) < \mathrm{OPT}(I) + \epsilon \cdot p(|I|_{unary}) = \mathrm{OPT}(I) + 1.$$

Συμπεραίνουμε επομένως, ότι το FPTAS για το συγκεκριμένο ϵ θα παράγει μία βέλτιστη λύση. Η χρονική πολυπλοκότητα θα είναι πολυωνυμική ως προς το $|I|$ και το $p(|I|_{unary})$, δηλαδή πολυωνυμική ως προς το $|I|_{unary}$. Επομένως, έχουμε αποκτήσει ένα ψευδοπολυωνυμικό αλγόριθμο για το πρόβλημα Π. Με όμοιο τρόπο αποδεικνύουμε ότι το θεώρημα Α.1 ισχύει και για προβλήματα μεγιστοποίησης. □

Το ακόλουθο πόρισμα είναι άμεση συνέπεια του Θεωρήματος Α.1

Πόρισμα Α.1 *Θεωρούμε ένα* **NP**-*hard πρόβλημα ελαχιστοποίησης* Π *που ικανοποιεί τις συνθήκες* 1 *και* 2 *του Θεωρήματος 1.1. Αν το* Π *είναι strongly* **NP**-*hard, τότε δεν επιδέχεται* FPTAS, *εκτός και αν* **P** = **NP**.

Απόδειξη. Αν το Π επιδέχεται FPTAS, τότε θα επιδέχεται ψευδοπολυωνυμικό αλγόριθμο βάση του Θεωρήματος Α.1. Τότε όμως το Π δεν είναι strongly **NP**-hard, έχοντας υποθέσει ότι **P** ≠ **NP**, το οποίο οδηγεί σε αντίφαση. □

Μπορούμε να παρατηρήσουμε ότι οι συνθήκες 1 και 2 του Θεωρήματος Α.1 δεν είναι απόλυτα αυστηρές και ικανοποιούνται από τα περισσότερα κοινά προβλήματα βελτιστοποίησης. Σαφώς θα μπορούσε κανείς να κατασκευάσει προβλήματα για τα οποία αυτές δεν ισχύουν. Επομένως, κάθε πρόβλημα που ικανοποιεί τις συνθήκες αυτές και επιδέχεται FPTAS λύνεται βέλτιστα σε ψευδοπολυωνυμικό χρόνο. Το ερώτημα που τίθεται είναι αν κάθε πρόβλημα που ικανοποιεί τις συνθήκες αυτές και επιπλέον επιδέχεται ψευδοπολυωνυμικό αλγόριθμο, επιδέχεται FPTAS. Η απάντηση στο ερώτημα αυτό είναι αρνητική και μπορεί κανείς να το ελέγξει κατασκευάζοντας ένα κατάλληλο αντιπαράδειγμα το οποίο επιλύεται βέλτιστα σε ψευδοπολυωνυμικό χρόνο, αλλά δεν επιδέχεται FPTAS (βλ. Section 6 στο [86]). Στις περιπτώσεις αυτές η δεύτερη συνθήκη του Θεωρήματος Α.1 αντικαθίσταται από την ισχυρότερη συνθήκη $\mathrm{OPT}(I) \leq p(|I|)$.

Α.2.1.2 Πώς αποδεικνύουμε ότι δεν υπάρχει PTAS

Η πιο απλή τεχνική για την απόδειξη της μη ύπαρξης PTAS, δεδομένης της υπόθεσης ότι **P** ≠ **NP**, είναι η *τεχνική εισαγωγής χάσματος (gap technique)*. Η τεχνική αυτή χρησιμοποιήθηκε αρχικά στα μέσα της δεκαετίας του 1970 και η ιδέα της έγκειται στην κατασκευή μίας πολυωνυμικής αναγωγής που δημιουργεί ένα χάσμα ανάμεσα στην τιμή της αντικειμενικής συνάρτησης ενός "ΟΧΙ" στιγμιοτύπου και στην αντίστοιχη τιμή ενός "ΝΑΙ" στιγμιοτύπου του επικείμενου προβλήματος βελτιστοποίησης. Η ιδέα αυτή διασαφηνίζεται στο ακόλουθο θεώρημα (βλ. [86]).

Θεώρημα Α.2 *Θεωρούμε ένα* **NP***-hard πρόβλημα απόφασης* Π_1 *και ένα πρόβλημα ελαχιστοποίησης* Π_2 *της κλάσης* **NP** *και υποθέτουμε ότι υπάρχει πολυωνυμική αναγωγή* f *από το σύνολο των στιγμιοτύπων του* Π_1 *στο σύνολο των στιγμιοτύπων του* Π_2*, η οποία ικανοποιεί τις ακόλουθες δύο συνθήκες για προκαθορισμένους ακεραίους* a*,* b*, με* $a < b$*:*

1. *Κάθε "ΝΑΙ" στιγμιότυπο του* Π_1 *αντιστοιχίζεται μέσω της* f *σε ένα στιγμιότυπο του* Π_2*, του οποίου η τιμή της βέλτιστης λύσης είναι το πολύ* a*.*
2. *Κάθε "ΟΧΙ" στιγμιότυπο του* Π_1 *αντιστοιχίζεται μέσω της* f *σε ένα στιγμιότυπο του* Π_2*, του οποίου η τιμή της βέλτιστης λύσης είναι τουλάχιστον* b*.*

Τότε δεν υπάρχει ρ*-προσεγγιστικός αλγόριθμος πολυωνυμικού χρόνου, με* $\rho < \frac{b}{a}$*, για το πρόβλημα* Π_2*, εκτός και αν* **P** = **NP***. Ειδικότερα, το πρόβλημα* Π_2 *δεν επιδέχεται* PTAS*, εκτός και αν* **P** = **NP***.*

Απόδειξη. Υποθέτουμε ότι υπάρχει ρ-προσεγγιστικός αλγόριθμος πολυωνυμικού χρόνου με $\rho < b/a$ για το πρόβλημα Π_2 και τον εφαρμόζουμε σε ένα στιγμιότυπο $f(I)$ του Π_2, το οποίο προέκυψε με εφαρμογή της πολυωνυμικής αναγωγής f σε ένα στιγμιότυπο I του προβλήματος Π_1. Διακρίνουμε δύο περιπτώσεις:

1. Αν το I είναι ένα "ΝΑΙ" στιγμιότυπο, τότε η τιμή της βέλτιστης λύσης του $f(I)$ είναι το πολύ a και ο προσεγγιστικός αλγόριθμος δίνει μία λύση με τιμή αυστηρά μικρότερη του $(b/a) \cdot a = b$.
2. Αν το I είναι ένα "ΟΧΙ" στιγμιότυπο, τότε η η τιμή της βέλτιστης λύσης του $f(I)$ είναι τουλάχιστον b και ο προσεγγιστικός αλγόριθμος δεν μπορεί να δώσει λύση με τιμή καλύτερη της βέλτιστης τιμής b.

Επομένως, μπορούμε να διακρίνουμε σε πολυωνυμικό χρόνο αν ένα στιγμιότυπο του **NP**-hard προβλήματος Π_1 είναι "ΝΑΙ" στιγμιότυπο (η προσεγγιστική τιμή του $f(I)$ θα είναι $< b$) ή "ΟΧΙ" στιγμιότυπο (η προσεγγιστική τιμή του $f(I)$ θα είναι $\geq b$). Καταλήγουμε έτσι στο συμπέρασμα ότι **P** = **NP**, το οποίο αντιβαίνει στην αρχική μας υπόθεση. □

Το Θεώρημα Α.2 αποδεικνύεται με παρόμοιο τρόπο και στην περίπτωση που το Π_2 είναι πρόβλημα μεγιστοποίησης. Οι εφαρμογές του Θεωρήματος Α.2 καλύπτουν ένα μεγάλο αριθμό προβλημάτων. Ένα παράδειγμα περιγράφεται στη συνέχεια και αφορά το πρόβλημα **Bin Packing**: Δίνονται n αντικείμενα με μεγέθη $a_1, \ldots, a_n \in (0, 1]$ και ζητάμε να βρούμε μία ανάθεση των αντικειμένων σε κάδους χωρητικότητας 1 η οποία να ελαχιστοποιεί τον αριθμό των κάδων που χρησιμοποιούνται.

Θεώρημα Α.3 *Δεν υπάρχει ρ-προσεγγιστικός αλγόριθμος πολυωνυμικού χρόνου με $\rho < \frac{3}{2}$ για το πρόβλημα Bin Packing, εκτός και αν* **P** = **NP**. *Ειδικότερα, το πρόβλημα Bin Packing δεν επιδέχεται* PTAS, *εκτός και αν* **P** = **NP**.

Απόδειξη. Αν υπήρχε ένας τέτοιος αλγόριθμος, τότε θα επιλύαμε σε πολυωνυμικό χρόνο το πρόβλημα που αποφασίζει αν μπορούμε να διαμερίσουμε ένα σύνολο από n μη αρνητικούς ακεραίους $a_1, \ldots, a_n$ σε δύο σύνολα, καθένα από τα οποία έχει συνολικό άθροισμα $\frac{1}{2}\sum_i a_i$. Είναι εμφανές πως η απάντηση στο ερώτημα αυτό είναι "ΝΑΙ" αν και μόνο αν τα n αντικείμενα μπορούν να ανατεθούν σε δύο κάδους μεγέθους $\frac{1}{2}\sum_i a_i$ ο καθένας. Αν η απάντηση είναι "ΝΑΙ", τότε ο ρ-προσεγγιστικός αλγόριθμος θα πρέπει να δώσει μία βέλτιστη ανάθεση και επομένως να επιλύσει το πρόβλημα της διαμέρισης (το οποίο είναι γνωστό **NP**-hard πρόβλημα [30]). □

Ένας άλλος τρόπος για να αποδεικνύουμε ότι ένα πρόβλημα δεν επιδέχεται PTAS είναι μέσω της εφαρμογής μίας *αναγωγής διατήρησης προσέγγισης (approximation preserving reduction)*. Πρόκειται για μία αναγωγή η οποία συνδέει δύο προβλήματα βελτιστοποίησης διατηρώντας την προσεγγισιμότητα σε μία σταθερά πολλαπλάσια ποσότητα. Κατά καιρούς έχουν διατυπωθεί αρκετές αναγωγές διατήρησης προσέγγισης, όπως η L-αναγωγή [72], η A-αναγωγή και η AP-αναγωγή [5]. Στη συνέχεια δίνουμε έναν ορισμό μίας αναγωγής διατήρησης προσέγγισης, ο οποίος αποτελεί μια πιο αυστηρή εκδοχή των αναγωγών που αναφέρθηκαν προηγουμένως και που στην πραγματικότητα διατηρεί αυτούσιο τον παράγοντα προσέγγισης (βλ. [94]). Επιπλέον, τα σχετιζόμενα προβλήματα είναι και τα δυο είτε προβλήματα ελαχιστοποίησης, είτε προβλήματα μεγιστοποίησης.

Ορισμός Α.7 *Θεωρούμε δύο προβλήματα ελαχιστοποίησης Π_1 και Π_2 (ο ορισμός είναι παρόμοιος για δύο προβλήματα μεγιστοποίησης). Μία πολυωνυμική αναγωγή f από το πρόβλημα Π_1 στο πρόβλημα Π_2, ονομάζεται αναγωγή διατήρησης προσέγγισης (approximation preserving reduction) αν υπάρχει συνάρτηση g, που υπολογίζεται σε πολυωνυμικό χρόνο, έτσι ώστε:*

1. *Για κάθε στιγμιότυπο I του Π_1, το $I' = f(I)$ είναι ένα στιγμιότυπο του Π_2, τέτοιο ώστε να ισχύει:*

$$\mathrm{OPT}(I') \leq \mathrm{OPT}(I).$$

2. *Για κάθε εφικτή λύση s' του I' κόστους $c_1(s')$, η $s = g(s')$ είναι μία εφικτή του I, τέτοια ώστε:*

$$c_1(s) \leq c_2(s').$$

Με c_1, c_2 συμβολίζουμε τις αντικειμενικές συναρτήσεις των προβλημάτων Π_1 και Π_2 αντίστοιχα. Εύκολα παρατηρούμε, ότι η παραπάνω αναγωγή διατήρησης προσέγγισης μαζί με έναν ρ-προσεγγιστικό αλγόριθμο για το πρόβλημα Π_2 δίνουν έναν ρ-προσεγγιστικό αλγόριθμο για το πρόβλημα Π_1. Επομένως, αν αντί ενός ρ-προσεγγιστικού αλγορίθμου θεωρήσουμε ένα πολυωνυμικό προσεγγιστικό σχήμα για το πρόβλημα Π_2, προκύπτει το ακόλουθο θεώρημα.

Θεώρημα Α.4 *Δεδομένης μίας αναγωγής διατήρησης προσέγγισης από ένα πρόβλημα βελτιστοποίησης Π_1 σε ένα πρόβλημα βελτιστοποίησης Π_2, αν υπάρχει* PTAS *για το Π_2, τότε υπάρχει* PTAS *και για το Π_1. Ισοδύναμα, αν δεν υπάρχει* PTAS *για το Π_1, τότε δεν υπάρχει* PTAS *για το Π_2.*

Χωρίς να είμαστε ιδιαίτερα αυστηροί, μπορούμε να πούμε ότι οι αναγωγές διατήρησης προσέγγισης υποστηρίζουν την μη ύπαρξη ενός PTAS με τον ίδιο τρόπο που οι κλασσικές πολυωνυμικές αναγωγές υποστηρίζουν την μη ύπαρξη αλγορίθμων πολυωνυμικού χρόνου.

Οι Papadimitriou και Yannakakis [72], ενδιαφερόμενοι να καθορίσουν ποια προβλήματα επιδέχονται PTAS και ποια όχι, όρισαν μία κλάση προβλημάτων βελτιστοποίησης, την **MAX-SNP**, καθώς και μία έννοια πληρότητας για την κλάση αυτή. Διαισθητικά, η **MAX-SNP** είναι μία συντακτικά ορισμένη κλάση, η οποία περιέχει προβλήματα βελτιστοποίησης του **NP** που μπορούν να εκφραστούν ως προβλήματα ικανοποίησης ενός φραγμένου πλήθους περιορισμών. Ένα πρόβλημα Π της κλάσης **MAX-SNP** είναι πλήρες για αυτή, αν κάθε πρόβλημα αυτής ανάγεται στο Π με L-αναγωγή.

Προκύπτει έτσι ότι ένα **MAX-SNP**-πλήρες πρόβλημα επιδέχεται PTAS αν και μόνο αν όλα τα προβλήματα της **MAX-SNP** κλάσης επιδέχονται PTAS (βλ. [72]). Αρκετά προβλήματα, όπως το MAX-3SAT που είναι το αντίστοιχο πρόβλημα βελτιστοποίησης του 3SAT και το MAX-CUT, αποδείχθηκαν **MAX-SNP**-πλήρη, ενώ για κανένα από αυτά δεν βρέθηκε κάποιο PTAS, οπότε πιστεύουμε πως αν ένα πρόβλημα είναι **MAX-SNP**-hard, τότε αυτό δεν επιδέχεται PTAS, χωρίς όμως να αποδεικνύεται κάτι τέτοιο. Επιπλέον, αποδείχθηκε (βλ. [72]) ότι για κάθε **MAX-SNP** πρόβλημα Π υπάρχει σταθερά $C_\Pi \geq 1$, έτσι ώστε ένας πολυωνυμικού χρόνου αλγόριθμος να πετυχαίνει παράγοντα προσέγγισης C_Π για το πρόβλημα, χωρίς όμως να έχει καθοριστεί ποια είναι η μικρότερη τιμή του C_Π για την οποία αυτό αληθεύει.

Έχοντας ως βάση ένα **MAX-SNP**-πλήρες πρόβλημα, όπως για παράδειγμα το MAX-3SAT και με τη βοήθεια μιας κατάλληλης L-αναγωγής, φαίνεται να μπορούμε να προσδιορίσουμε τη μη προσεγγισιμότητα ή την προσεγγισιμότητα για το σύνολο των **NP**-hard προβλημάτων βελτιστοποίησης. Παρόλα αυτά, κάτι τέτοιο φαντάζει εξαιρετικά δύσκολο και η δυσκολία έγκειται στην εύρεση κατάλληλης L-αναγωγής, αλλά και οποιασδήποτε αναγωγής διατήρησης προσέγγισης. Για παράδειγμα, υπάρχουν εμφανώς σχετιζόμενα προβλήματα, όπως το MAXIMUM-CLIQUE και το CHROMATIC-NUMBER (βλ. [6] για αποτελέσματα μη προσέγγισης αυτών), για τα οποία δεν είναι γνωστή κάποια αναγωγή διατήρησης προσέγγισης (βλ. [5]). Περιπτώσεις σαν και αυτές αντιμετωπίσθηκαν εντέλει με την εισαγωγή νέων τεχνικών, οι οποίες βασίζονται σε πιθανοτικούς χαρακτηρισμούς της κλάσης **NP**.

Ο πιο γνωστός τέτοιος χαρακτηρισμός είναι το λεγόμενο *PCP (Probabilistically Checkable Proof system) Θεώρημα*. Θυμίζουμε ότι η κλάση **NP** είναι η κλάση των προβλημάτων απόφασης, όπου για κάθε "ΝΑΙ" στιγμιότυπο ενός προβλήματος υπάρχει κάποιο πιστοποιητικό, πολυωνυμικού μήκους, ως προς την είσοδο, το οποίο μπορεί να επαληθευθεί από ένα αλγόριθμο πιστοποίησης σε πολυωνυμικό χρόνο (βλ. Ορισμό Α.1). Άτυπα, ένα Πιθανοτικά Ελέγξιμο Σύστημα Απόδειξης Probabilistically Checkable Proof system για ένα πρόβλημα στο **NP** κωδικοποιεί το πιστοποιητικό με ειδικό τρόπο έτσι ώστε να το επαληθεύει, με τον αλγόριθμο πιστοποίησης, επιλέγοντας με πιθανοτικό τρόπο ένα σταθερό αριθμό από bits αυτού.

Ο *αλγόριθμος πιστοποίησης* είναι ένας πιθανοτικός αλγόριθμος πολυωνυμικού χρόνου ο οποίος χρησιμοποιεί ένα μαντείο (oracle) έτσι ώστε να έχει τυχαία πρόσβαση στην απόδειξη: όταν δίνεται στο μαντείο μία θέση (διεύθυνση) της απόδειξης, αυτό επιστρέφει την τιμή του αντίστοιχου bit. Δηλαδή, για δεδομένη απόδειξη π το μαντείο θα επιστρέψει ένα σύνολο διευθύνσεων της π που αντιστοιχούν σε 1-bits. Ο αλγόριθμος εξετάζει (ντετερμινιστικά) τα bits που του επιστράφηκαν και είτε αποδέχεται (απαντάει "ΝΑΙ") είτε απορρίπτει (απαντάει "ΟΧΙ"), ανάλογα με τις τιμές αυτών.

Ορισμός Α.8 *Θεωρούμε δύο συναρτήσεις $a(n), b(n)$, όπου n το μήκος της εισόδου. Ένα πρόβλημα $\mathcal{A} \in$ **PCP**$(a(n), b(n))$, αν υπάρχει πιθανοτικός αλγόριθμος V, έτσι ώστε σε κάθε είσοδο x, ο V λαμβάνει $O(a(n))$ τυχαία bits και εξετάζει $O(b(n))$ bits της απόδειξης. Επιπλέον,*

1. *Αν $x \in \mathcal{A}$, τότε υπάρχει απόδειξη y, που οδηγεί τον V σε αποδοχή με πιθανότητα 1.*

2. *Αν $x \notin \mathcal{A}$, τότε υπάρχει απόδειξη y, που οδηγεί τον V σε αποδοχή με πιθανότητα $< \frac{1}{2}$.*

Η πιθανότητα οφείλεται στην τυχαία συμβολοσειρά που λαμβάνει ο V από κάθε είσοδο x. Η πιθανότητα αποδοχής στην περίπτωση που $x \notin \mathcal{A}$, ονομάζεται πιθανότητα λάθους.

Βάσει του Ορισμού Α.8 προκύπτει ότι $\mathbf{NP} = \bigcup_{c\geq 0} \mathbf{PCP}(0, n^c)$, αφού ο αλγόριθμος πιστοποίησης δεν απαιτεί κανένα τυχαίο bit και επομένως, ντετερμινιστικά αποδέχεται εισόδους $x \in \mathcal{A}$ και απορρίπτει εισόδους $x \notin \mathcal{A}$, για ένα πρόβλημα $\mathcal{A}$, ακριβώς όπως και στον ορισμό του **NP**. Το PCP Θεώρημα διατυπώνεται ως ακολούθως:

$$\mathbf{NP} = \mathbf{PCP}(\log n, 1)$$

Για τις λεπτομέρειες της απόδειξης του PCP Θεωρήματος ο αναγνώστης παραπέμπεται στο [4], ενώ μία επισκόπηση των συνεπειών του προσφέρεται στα [6] και [5].

Η χρήση του PCP Θεωρήματος, σε συνάρτηση με κατάλληλη εφαρμογή της gap technique (βλ. Θεώρημα Α.2) οδήγησαν σε πολλά νέα αποτελέσματα μη προσέγγισης.

Σε μία γενική επισκόπηση του θέματος από τους Arora και Lund [6] εξηγείται με συνοπτικό τρόπο πώς, βάση αυτής της τεχνικής, αποδεικνύονται τα πιο γνωστά αποτελέσματα μη προσέγγισης. Επιπλέον, συλλέγοντας όλα τα προβλήματα των οποίων η μη προσεγγισιμότητα αποδεικνύεται με βάση την τεχνική αυτή, παρατηρούμε ότι αυτά εν γένει κατανέμονται σε τέσσερις ευρύτερες κλάσεις, με βάση τον παράγοντα προσέγγισης τον οποίο έχει αποδειχθεί ότι δεν μπορούν να επιτύχουν. Η κλάση I περιέχει όλα τα προβλήματα για τα οποία η επίτευξη ενός παράγοντα προσέγγισης $1+\epsilon$, για συγκεκριμένο $\epsilon > 0$, αποδεικνύεται **NP**-hard. Οι κλάσεις II, III και IV περιέχουν προβλήματα για τα οποία ο αντίστοιχος παράγοντας προσέγγισης είναι $\Theta(\log n)$, $2^{\log^{1-\epsilon} n}$ για κάθε προκαθορισμένο $\epsilon > 0$ και n^{ϵ} για κάποιο συγκεκριμένο $\epsilon > 0$ (φυσικά κάθε κλάση εμπεριέχει τις κλάσεις με μικρότερη από αυτήν αρίθμηση). Κάθε μία από τις παραπάνω τέσσερις κλάσεις περιέχει ένα αντιπροσωπευτικό πρόβλημα το οποίο μπορεί να χρησιμοποιηθεί, με εφαρμογή της gap technique, για την απόδειξη αποτελεσμάτων μη προσεγγισιμότητας για όλα τα προβλήματα της κλάσης. Η μη προσεγγισιμότητα των αντιπροσωπευτικών αυτών προβλημάτων μπορεί να αποδειχθεί χρησιμοποιώντας το πρόβλημα MAX-3SAT, δηλαδή, εφαρμόζοντας την gap technique από το MAX-3SAT σε κάθε αντιπροσωπευτικό πρόβλημα.

Σε αντίθεση με τις συντακτικά ορισμένες κλάσεις, όπως η **MAX-SNP** η οποία επιτρέπει αποτελέσματα για τεχνητά προβλήματα, οι παραπάνω τέσσερις κλάσεις περιέχουν προβλήματα των οποίων η προσεγγισιμότητα είναι πλήρως καθορισμένη, εκτός και αν $\mathbf{P} = \mathbf{NP}$. Οι κλάσεις αυτές χαρακτηρίζονται ως υπολογιστικά ορισμένες και μία σημαντική υπολογιστικά ορισμένη κλάση είναι η κλάση **APX**, η οποία περιέχει όλα εκείνα τα προβλήματα βελτιστοποίησης της κλάσης **NP** για τα οποία υπάρχει πολυωνυμικού χρόνου αλγόριθμος με παράγοντα προσέγγισης φραγμένο από μία σταθερά $c > 0$. Η σχέση ανάμεσα στη συντακτική κλάση **MAX-SNP** και στην υπολογιστική κλάση **APX**, αλλά και γενικότερα η σχέση μεταξύ των δύο διαφορετικού τύπου κλάσεων προβλημάτων, συντακτικές και υπολογιστικές, εξετάζεται στο [80] (μεταξύ άλλων αποδεικνύεται ότι για κάθε πρόβλημα της **APX** υπάρχει αναγωγή διατήρησης προσέγγισης σε ένα

πρόβλημα της **MAX-SNP**).

Η κλάση **APX** αντιστοιχεί στην κλάση I που ορίσαμε προηγουμένως. Αντίστοιχα μπορούμε να συμβολίζουμε και τις άλλες τρείς κλάσεις, π.χ. η κλάση II συμβολίζεται με log-**APX**. Ένα πρόβλημα Π είναι **APX**-hard αν υπάρχει $\epsilon > 0$, τέτοιο ώστε δεν υπάρχει πολυωνυμικού χρόνου προσεγγιστικός αλγόριθμος για το Π, με παράγοντα προσέγγισης $1+\epsilon$. Ένα **APX**-hard πρόβλημα μπορεί να ανήκει, είτε στην κλάση APX, οπότε είναι **APX**-πλήρες (το MAX-3SAT είναι ένα **APX**-hard πρόβλημα) είτε σε κάποια από τις άλλες τρείς κλάσεις, II, III, IV. Προκύπτει έτσι από τους ορισμούς, ότι αν υπάρχει PTAS για κάποιο **APX**-hard πρόβλημα τότε **P** = **NP**. Το αποτέλεσμα αυτό έρχεται να διασαφηνίσει την προηγούμενή μας υπόθεση ότι αν ένα πρόβλημα αποδειχθεί **MAX-SNP**-hard τότε δεν επιδέχεται PTAS, εκτός βέβαια και αν **P** = **NP**.

Τέλος, αν από την κλάση **APX** επιλέξουμε εκείνα τα προβλήματα που επιδέχονται PTAS προκύπτει η αντίστοιχη κλάση **PTAS**. Ομοίως προκύπτουν και οι κλάσεις προβλημάτων **FPTAS** αλλά και $\mathbf{PTAS}^{\infty}$. Αν επιπλέον υποθέσουμε ότι **P** ≠ **NP** μπορούμε να αποδείξουμε τις ακόλουθες συμπεριλήψεις κλάσεων:

$$\mathbf{FPTAS} \subset \mathbf{PTAS} \subset \mathbf{PTAS}^{\infty} \subset \mathbf{APX}.$$

Για μία εκτενή διερεύνηση των κλάσεων προσέγγισης ο αναγνώστης παραπέμπεται στο [27].

Γλωσσάρι

Smith's rule	κανόνας του Smith, 7
average weighted completion time	μέσος βεβαρημένος χρόνος ολοκλήρωσης, 29
completion time	χρόνος ολοκλήρωσης, 4
due date	προθεσμία, 1
earliest due date (EDD)	μικρότερη προθεσμία, 7
earliness	πρόωρη ολοκλήρωση, 5
early jobs	πρόωρες διεργασίες, 5
geometric rounding	γεωμετρική στρογγυλοποίηση, 37
lateness	αργοπορία, 5
makespan	ο απαιτούμενος χρόνος για την ολοκλήρωση της εκτέλεσης όλων των διεργασιών, 4
parallel identical machines	παράλληλες πανομοιότυπες μηχανές, 3
parallel unrelated machines	παράλληλες μη σχετιζόμενες μηχανές, 3
precedence constraints	περιορισμοί προτεραιότητας, 1
preemptive schedule	χρονοδρομολόγηση κατά την οποία οι διεργασίες μπορούν να διακόπτουν την εκτέλεσή τους και να τη συνεχίζουν σε επόμενη χρονική στιγμή, 29
release date	χρόνος αποδέσμευσης, 1
scheduling problems	προβλήματα χρονοδρομολόγησης, 1
shortest processing time (SPT)	μικρότερος χρόνος επεξεργασίας, 6
single machine	μοναδική μηχανή, 3
tardiness	καθυστέρηση, 5
tardy jobs	καθυστερημένες διεργασίες, 5
total completion time	μέσος ή συνολικός χρόνος ολοκλήρωσης, 4
total weighted earliness	συνολική βεβαρημένη πρόωρη ολοκλήρωση, 64
total weighted tardiness	συνολική βεβαρημένη καθυστέρηση, 63

trimming technique	τεχνική αποκοπής, 32
weighted number of tardy jobs	συνολικό βάρος των καθυστερημένων διεργασιών, 64
fully polynomial time approximation scheme (FPTAS)	πλήρες πολυωνυμικό προσεγγιστικό σχήμα, 116
polynomial time approximation scheme (PTAS)	προσεγγιστικό σχήμα πολυωνυμικού χρόνου, 116

Βιβλιογραφία

[1] T. S. Abdul-Razaq, C.N. Potts, and L. N. Van Wassenhove. A survey of algorithms for the single machine total weighted tardiness scheduling problem. *Discrete Applied Mathematics*, 26:235–253, 1990.

[2] F. Afrati, E. Bampis, C. Chekuri, D. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Skutella, C. Stein, and M. Sviridenko. Approximation schemes for minimizing average weighted completion time with release dates. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 32–43, 1999.

[3] C. Ambühl and M. Mastrolilli. Single machine precedence constrained scheduling is a vertex cover problem. In *Proceedings of the 14th Annual European Symposium on Algorithms*, pages 28–39, 2006.

[4] S. Arora. *Probabilistic Checking of Proofs and Hardness of Approximation Problems*. PhD thesis, Princeton University, Department of Computer Science, 1994.

[5] S. Arora. The approximability of NP-hard problems. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, 1998. Survey based upon a plenary lecture at ACM STOC.

[6] S. Arora and C. Lund. Hardness of approximations. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-hard problems*, pages 399–446. PWS, Boston, 1997.

[7] K. R. Baker. *Introduction to Sequencing and Scheduling*. Wiley, 1974.

[8] P. Brucker and S. Knust. Complexity results for scheduling problems. http://www.mathematik.uni-osnabrueck.de/research/or/class/.

[9] J.L. Bruno, E.G. Coffman Jr, and R. Sethi. Scheduling independent tasks to reduce mean finishing time. *Communications of the ACM*, 17:382–387, 1974.

[10] S. Chakrabarti, C. A. Phillips, A. S. Schulz, D. B. Shmoys, C. Stein, and J. Wein. Improved scheduling algorithms for minsum criteria. In F. Meyer auf der Heide and B. Monien, editors,

Automata, Languages and Programming, number 1099 in Lecture Notes in Computer Science. Springer, Berlin, 1996. Proceedings of the 23rd International Colloquium (ICALP'96).

[11] C. Chekuri. *Approximation Algorithms for Scheduling Problems*. PhD thesis, Stanford University, Department of Computer Science, 1998.

[12] C. Chekuri and S. Khanna. A PTAS for minimizing weighted completion time on uniformly related machines. In *Proceedings of the 28th ICALP*, pages 848–861, 2001.

[13] C. Chekuri and S. Khanna. Approximation algorithms for minimizing the weighted sum of completion times. In J. Y-T. Leung, editor, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Chapman & Hall/CRC, 2004.

[14] C. Chekuri and R. Motwani. Precedence constrained scheduling to minimize sum of weighted completion times on a single machine. *Discrete Applied Mathematics*, 98:29–38, 1999.

[15] C. Chekuri, R. Motwani, B. Natarajan, and C. Stein. Approximation techniques for average completion time scheduling. In *Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms*, pages 609–618, 1997.

[16] T. C. E. Cheng, C. T. Ng, J. J. Yuan, and Z. H. Liu. Single machine scheduling to minimize total weighted tardiness. *European Journal of Operational Research*, 165:423–443, 2005.

[17] F. A. Chudak and D. S. Hochbaum. A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine. *Operations Research Letters*, 25:199–204, 1999.

[18] R. K. Congram, C. N. Potts, and S. L. van de Velde. An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing*, 14(1):52–67, 2002.

[19] R. W. Conway, W. L. Maxwell, and L. W. Miller. *Theory of Scheduling*. Addison-Wesley, 1967.

[20] J. R. Correa and A. S. Schulz. Single-machine scheduling with precedence constraints. *Mathematics of Operations Research*, 30:1005–1021, 2005.

[21] H. A. Crauwels, C. N. Potts, and L. N. Van Wassenhove. Local search heuristics for the single-machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing*, 10:341–350, 1998.

[22] F. Dela Croce, A. Grosso, and V. T. Paschos. Lower bounds on the approximation ratios of leading heuristics for the single-machine total tardiness problem. *Journal of Scheduling*, 7:85–91, 2004.

[23] W. Fernandez de la Vega and G.S. Lueker. Bin packing can be solved within $1 + \epsilon$ in linear time. *Combinatorica*, 1:349–355, 1981.

[24] J. Du and J. Y-T. Leung. Minimizing total tardiness on one machine is NP-hard. *Mathematics of Operations Research*, 15:483–495, 1990.

[25] Jr. E. G. Coffman and E.N. Gilbert. On the expected relative performance of list scheduling. *Operations Research*, 33:548–561, 1985.

[26] J. Y-T. Leung (editor). *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Chapman & Hall/CRC, 2004.

[27] G. Ausiello et al. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer-Verlag, Berlin, 2003.

[28] Philippe Chretienne et al. (editors). *Scheduling theory and its applications*. John Wiley and Sons, 1995.

[29] D. K. Friezen. Tighter bounds for the multifit processor scheduling algorithm. *SIAM Journal on Computing*, 13:170–181, 1984.

[30] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[31] G. V. Gens and E. V. Levner. Fast approximation algorithms for job sequencing with deadlines. *Discrete Applied Mathematics*, 3:313–318, 1981.

[32] Michel X. Goemans. Improved approximation algorithms for scheduling with release dates. In *Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms*, pages 591–598, 1997.

[33] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[34] R.L. Graham. Bounds for certain multiprocessor anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.

[35] R.L. Graham. Bounds on multiprocessing anomalies. *SIAM Journal of Applied Mathematics*, 17:263–269, 1969.

[36] L. A. Hall, A.S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Mathematics of Operations Research*, 22:513–544, 1997.

[37] M. Held and R. M. Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):196–210, 1962.

[38] D. S. Hochbaum. Various notions of approximations: good, better, best, and more. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-hard problems*, pages 346–398. PWS, 1997.

[39] D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of the ACM*, 34:144–162, 1987.

[40] D. S. Hochbaum and D. B. Shmoys. A polynomial approximation scheme for machine scheduling on uniform processors: using the dual approximation approach. *SIAM Journal on Computing*, 17:539–551, 1988.

[41] J.A. Hoogeveen, P. Schuurman, and G.J. Woeginger. Non–aproximability results for scheduling problems with minsum criteria. In *Proceedings of the 6th Conference on Integer Programming and Combinatorial Optimization*, volume 1412 of *LNCS*, pages 353–366. Springer-Verlag, Berlin, 1998.

[42] W. Horn. Minimizing average flow time with parallel machines. *Operations Research*, 21:846–847, 1973.

[43] Horowitz and S. Sahni. Exact and approximate algorithms for scheduling non identical processors. *Journal of the ACM*, 23:317–327, 1976.

[44] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the Knapsack and sum of subset problems. *Journal of the ACM*, 22:463–468, 1975.

[45] I.Dinur and S. Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162(1):439–486, 2005.

[46] J. R. Jackson. Scheduling a production line to minimize maximum tardiness. Res. Rep. 43, Management Science Research Project, UCLA, 1955.

[47] S. M. Johnson. Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, pages 61–68, 1954.

[48] N. Karmarkar and R. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 312–320, 1982.

[49] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[50] T. Kawaguchi and S. Kyan. Worst case bound of an LRF schedule for the mean weighted flow-time problem. *SIAM Journal on Computing*, 15:1119–1129, 1986.

[51] H. Kellerer and V. A. Strusevitch. A fully polynomial approximation scheme for the single machine weighted total tardiness problem with a common due date. *Theoretical Computer Science*, 369:230–238, 2006.

[52] L. G. Khachian. A polznomial algorithm for linear programming. *Doklady Akad. Nauk USSR*, no. 5:1093–1096, 1979. Translated in Soviet Math. Doklady.

[53] S. G. Kolliopoulos and G. Steiner. Approximation algorithms for minimizing the total weighted tardiness on a single machine. *Theoretical Computer Science*, 355:261–273, 2006.

[54] S. G. Kolliopoulos and G. Steiner. Approximation algorithms for scheduling problems with a modified total weighted tardiness objective. *Operations Research Letters*, 35:685–692, 2006.

[55] M. Y. Kovalyov and F. Werner. Approximation schemes for scheduling jobs with common due date on parallel machines to minimize total tardiness. *Journal of Heuristics*, 8:415–428, 2002.

[56] M. A. Langston. *Processor scheduling with improved heuristic algorithms*. PhD thesis, Texas A&M University, College Station, Texas, 1981.

[57] E. L. Lawler. On scheduling problems with deferral costs. *Management Science*, 11(2):280–288, 1964.

[58] E. L. Lawler. Optimal sequencing of a single processor subject to precedence constraints. *Management Science*, 19:544–546, 1973.

[59] E. L. Lawler. Sequencing to minimize the weighted number of tardy jobs. *Recherche Operationnel*, 10:27–33, 1976.

[60] E. L. Lawler. A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics*, 1:331–342, 1977.

[61] E. L. Lawler. A fully polynomial approximation scheme for the total tardiness problem. *Operations Research Letters*, 1:207–208, 1982.

[62] E. L. Lawler, J. K. Lenstra, A. H. G. Rinooy Kan, and D. B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In S. C. Graves, A. H. G. Rinnooy Kan, and P. H. Zipkin, editors, *Handbooks in Operations Research and Management Science, Vol 4., Logistics of Production and Inventory*, pages 445–522. North-Holland, 1993.

[63] E. L. Lawler and J. M. Moore. A functional equation and its application to resource allocation and sequencing problems. *Management Science*, 16:77–84, 1969.

[64] H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

[65] J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26:22–35, 1978.

[66] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming A*, 46:259–271, 1990.

[67] J.K. Lenstra, A.H.G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362, 1977.

[68] H. C. Matsuo, C. J. Suh, and R.S. Sullivan. A controlled search simulated annealing method for the single machine weighted tardiness problem. Working Paper 87-12-2, Department of Management, University of Texas, Austin, TX, 1987.

[69] R. McNaughton. Scheduling with deadlines and loss functions. *Management Science*, 6:1–12, 1959.

[70] J. M. Moore. An n job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Science*, 15(1):102–109, 1968.

[71] A. Munier, M. Queyranne, and A. Schulz. Approximation bounds for a general class of precedence constrained parallel machine scheduling problems. In *Proceedings of the 6th Conference on Integer Programming and Combinatorial Optimization*, pages 367–382, 1998.

[72] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. In *Journal of Computer and System Sciences*, volume 43, pages 425–440, 1991.

[73] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[74] C. Phillips, C. Stein, and J. Wein. Minimizing average completion time in the presence of release dates. To appear in Math Programming, 1995.

[75] Michael Pinedo. *Scheduling: theory, algorithms, and systems*. Prentice-Hall, 1995.

[76] C. N. Potts. Analysis of a linear programming heuristic for scheduling unrelated parallel machines. *Discrete Applied Mathematics*, 10:155–164, 1985.

[77] C. N. Potts and L. N. Van Wassenhove. A branch and bound algorithm for the total weighted tardiness problem. *Operations Research*, 33:363–377, 1985.

[78] C. N. Potts and L. N. Van Wassenhove. Single machine tardiness sequencing heuristics. *IIE Transactions*, 13:346–354, 1991.

[79] M. H. Rothkopf. *Scheduling independent tasks on one or more processors*. PhD thesis, M.I.T. School of industrial management, 1964. Also available as Interim Technical Report No. 2, Operations Research Center, Massachusetts Institute of Technology.

[80] M. Sudan S. Khanna, R. Motwani and U. Vazirani. On syntactic versus computational views of approximability. *SIAM Journal on Computing*, 28:164–191, 1999.

[81] S. Sahni. Algorithms for scheduling independent tasks. *Journal of the ACM*, 23:116–127, 1976.

[82] A. Schild and I. J. Fredman. Scheduling tasks with deadlines and nonlinear loss functions. *Management Science*, 9(1):73–81, 1962.

[83] A. Schulz and M. Skutella. Scheduling unrelated machines by randomized rounding. *SIAM Journal on Discrete Mathematics*, 15:450–469, 2002.

[84] A. S. Schulz and M. Skutella. Scheduling–LPs bear probabilities: Randomized approximations for min–sum criteria. In R. E. Burkard and G. J. Woeginger, editors, *Proceedings of the 5th Annual European Symposium on Algorithms*, volume 1284 of *LNCS*, pages 416–429. Springer-Verlag, 1997.

[85] P. Schuurman and G. J. Woeginger. Polynomial time approximation algorithms for machine scheduling: ten open problems. *Journal of Scheduling*, 2:203–213, 1999.

[86] P. Schuurman and G. J. Woeginger. Approximation schemes–a tutorial. START Project Woe-65, TU Graz, January 2001.

[87] T. Sen, J. M. Sulek, and P. Dileepan. Static scheduling research to minimize weighted and unweighted tardiness: a state-of-the-art survey. *International Journal of Production Economics*, 83:1–12, 2003.

[88] J. Sethuraman and M. S. Squillante. Optimal scheduling of multiclass parallel machines. In *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms*, pages 963–964, 1999.

[89] M. Skutella. Semidefinite relaxations for parallel machine scheduling. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 472–481, 1998.

[90] M. Skutella. Convex quadratic and semidefinite programming relaxations in scheduling. *Journal of the ACM*, 48:206–242, 2001.

[91] M. Skutella and G. J. Woeginger. A ptas for minimizing the weighted sum of job completion times on parallel machines. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 400–407, 1999.

[92] W.E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3:59–66, 1956.

[93] E. Torng and P. Uthaisombut. Lower bounds for srpt–subsequence algorithms for nonpreemptive scheduling. In *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms*, pages 973–974, 1999.

[94] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, Berlin, 2003.

[95] G. J. Woeginger. When Does a Dynamic Programming Formulation Guarantee the Existence of a Fully Polynomial Time Approximation Scheme (FPTAS). *INFORMS Journal on Computing*, 12(1):57–74, 2000.

[96] J. Yuan. The NP-hardness of the single machine common due date weighted tardiness problem. *Systems Science and Mathematical Sciences*, 5:328–333, 1992.