Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
τομεας τεχνολογιας πληροφορικης και υπολογιστων

Μέθοδοι Επαναχρησιμοποίησης Αποτελεσμάτων Εξόρυξης Γνώσης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Σοφίας Μ. Διαμαντίδου

Επιβλέπων: Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2006

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και
Μηχανικων Υπολογιστων
τομεας τεχνολογιας πληροφορικης και υπολογιστων

Μέθοδοι Επαναχρησιμοποίησης Αποτελεσμάτων Εξόρυξης Γνώσης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Σοφίας Μ. Διαμαντίδου

Επιβλέπων: Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 27η Ιουλίου 2006.

.....	.....	.....
Τιμολέων Σελλής	Ανδρέας Σταφυλοπάτης	Νικόλαος Παπασπύρου
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Λέκτορας Ε.Μ.Π.

Αθήνα, Ιούλιος 2006

.....

Σοφία Μ. Διαμαντίδου
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

στους γονείς μου

Περίληψη

Στην διπλωματική αυτή εργασία, προτείνεται και υλοποιείται μία μέθοδος *ένωσης* δύο FP-trees που αντιστοιχούν σε δύο βάσεις δεδομένων με δοσοληψίες πάνω στο ίδιο σύνολο αντικειμένων και τα οποία έχουν κατασκευαστεί σε προηγούμενη εφαρμογή της μεθόδου εξόρυξης συχνών προτύπων FP-growth. Στόχος είναι το mining της ένωσης των δύο βάσεων να γίνεται γρηγορότερα από την κλασική εφαρμογή της FP-growth η οποία προϋποθέτει την κατασκευή του FP-tree για την ενωμένη βάση από την αρχή. Η μέθοδος FP-growth είναι μία divide-and-conquer προσέγγιση του προβλήματος της εξόρυξης συχνών προτύπων η οποία εκτελεί το mining αποκλειστικά σε μία πρότυπη και συμπαγή δομή, το FP-tree. Ως επέκταση της κατασκευής του FP-tree, η εργασία αυτή μελετά και περιγράφει αναλυτικά τις μεθόδους ενημέρωσης και μετασχηματισμού των δέντρων που απαιτούνται για την ένωσή τους έτσι ώστε το FP-tree που προκύπτει να είναι το ίδιο με αυτό που θα κατασκευαζόταν από την εφαρμογή της FP-growth στην ένωση των βάσεων. Τέλος γίνεται υλοποίηση του προτεινόμενου αλγορίθμου και μελέτη της επίδοσής του με πειράματα πάνω σε συνθετικά δεδομένα, απ' όπου προκύπτει ότι υπερτερεί της κλασικής διαδικασίας, όσον αφορά τον χρόνο εκτέλεσης.

Λέξεις κλειδιά: δοσοληψία, εξόρυξη συχνών προτύπων, ένωση δέντρων συχνών προτύπων, εξόρυξη γνώσης.

Abstract

In this thesis, we present a method that constructs an FP-tree over the union of two transaction databases (TDBs) by applying the *merge* operation on the FP-trees constructed over each database respectively. The main goal is to achieve better performance in mining the frequent itemsets of the union database using the *merged* FP-tree, compared to the performance achieved by integrating the datasets and applying the FP-growth algorithm. FP-growth is a *divide-and-conquer* approach for mining the complete set of frequent items that takes advantage of a novel and compact data structure, called FP-tree. Our method takes advantage of the FP-trees previously constructed over two TDBs and merges them, so that the resulting FP-tree is exactly the same as the one constructed over the union of the respective data sets. In order to achieve that, we carefully analyze the required update and adjusting processes of the FP-trees and we propose efficient algorithms. Finally, we implement and test our method using sythetic data for a variety of database and domain sizes. The performance evaluation shows that our approach outperforms the classical process, in terms of the execution time.

Keywords: transaction, mining frequent itemsets, merge frequent pattern trees, data mining.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Αλγορίθμων

Πρόλογος

Η εξόρυξη γνώσης προσέλκυε πάντοτε το ενδιαφέρον των ανθρώπων καθώς εξυπηρετεί την πρωταρχική ανάγκη μάθησης μέσα από καταστάσεις, εμπειρίες και φυσικά πληροφορίες που αποκτώνται καθημερινά. Σίγουρα, οι πληροφορίες από μόνες τους δεν αποτελούν γνώση· μπορούν όμως να μετατραπούν. Η σημασία αυτής της προοπτικής ήταν που ανήγαγε την εξόρυξη γνώσης σε επιστημονικό πεδίο, το οποίο βρίσκεται σε πλήρη εξέλιξη τα τελευταία χρόνια και έχει θέσει τις βάσεις για περισσότερες προκλήσεις στο μέλλον.

Κατανοώ σημαίνει ανακαλύπτω πρότυπα. Αυτό ακριβώς προσπαθεί να κάνει η εξόρυξη γνώσης, χρησιμοποιώντας τεχνικές και μεθόδους από διάφορα πεδία, όπως η τεχνολογία βάσεων δεδομένων, η μηχανική μάθηση, τα νευρωνικά δίκτυα, η τεχνητή νοημοσύνη, η στατιστική. Τα αποτελέσματα που προκύπτουν, καλύπτουν ένα ευρύ φάσμα εφαρμογών τόσο επιχειρησιακού όσο και επιστημονικού ενδιαφέροντος, καθιστώντας τα ένα κατα βάση χρήσιμο και πολλές φορές απαραίτητο εργαλείο.

Η διπλωματική αυτή εργασία, που γίνεται σε προπτυχιακό επίπεδο, δεν εξυπηρετεί τόσο έναν σκοπό, όσο έναν αυτοσκοπό· δίνει τη δυνατότητα στον συγγραφέα να μελετήσει και να παρουσιάσει ενδιαφέροντα θέματα εξόρυξης γνώσης, ως έναυσμα για περαιτέρω μελέτη του αντικειμένου. Ταυτόχρονα όμως προτείνει μία μέθοδο για την εξυπηρέτηση των στόχων του data mining, η οποία ελπίζει να αποδειχθεί ενδιαφέρουσα και χρήσιμη για τον αναγνώστη.

Οργάνωση του τόμου

Ο τόμος αυτός χωρίζεται σε 6 κεφάλαια.

Το κεφάλαιο 1 ορίζει το data mining και αναδεικνύει τη χρησιμότητά του τη σημερινή εποχή. Παράλληλα γίνεται μία περιγραφή του αντικειμένου της διπλωματικής αυτής εργασίας με παράθεση ενός παραδείγματος από την καθημερινή ζωή για καλύτερη κατανόηση.

Το κεφάλαιο 2 παρουσιάζει μία επισκόπηση του προβλήματος της εύρεσης συχνών προτύπων από μία βάση με δοσοληψίες εισάγοντας ορισμένες βασικές έννοιες. Στη συνέχεια αναλύεται

λεπτομερώς η δομή FP-tree και η μέθοδος FP-growth στις οποίες βασίζεται μεγάλο μέρος της παρούσας εργασίας.

Το κεφάλαιο 3 περιγράφει αναλυτικά τον αλγόριθμο `merge` δύο FP-trees που προτείνεται για την εύρεση συχνών προτύπων από την ένωση δύο βάσεων δοσοληψιών.

Το κεφάλαιο 4 περιλαμβάνει την περιγραφή της υλοποίησης του αλγορίθμου με παράθεση των σημαντικότερων τμημάτων του κώδικα που αναπτύχθηκε.

Το κεφάλαιο 5 παρουσιάζει την πειραματική μελέτη του προτεινόμενου αλγορίθμου και γίνεται αποτίμηση των αποτελεσμάτων.

Το κεφάλαιο 6 συνοψίζει τα συμπεράσματα που προέκυψαν από την πειραματική μελέτη και προτείνει κατευθύνσεις για περαιτέρω έρευνα.

Ευχαριστίες

Η εκπόνηση αυτής της εργασίας υπήρξε κοπιώδης αλλά ενδιαφέρουσα και ιδιαιτέρως δημιουργική. Δε θα μπορούσε όμως να ολοκληρωθεί χωρίς την πολύτιμη βοήθεια ορισμένων ανθρώπων. Γι' αυτό θα ήθελα να ευχαριστήσω θερμά κατ' αρχήν τον καθηγητή Ε.Μ.Π., και επιβλέποντα της εργασίας μου, κ. Τίμο Σελλή, για τις συμβουλές και την αμέριστη υποστήριξη του. Επίσης, τον κ. Μανώλη Τερροβίτη, υποψήφιο διδάκτορα Ε.Μ.Π., για τη βοήθεια και τις σημαντικές κατευθύνσεις του και τους φίλους μου Γεώργιο Τσουκαλά και Κορνήλιο Κούρτη για την προθυμία τους να με βοηθήσουν όποτε το χρειάστηκα. Τέλος, θα ήθελα να εκφράσω ένα ειλικρινές ευχαριστώ στη συμφοιτήτρια και κυρίως φίλη μου Δήμητρα Ζαρμπούτη για την συμπαράσταση της τα τελευταία χρόνια.

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή

1.1 Εξόρυξη Γνώσης

Με την ψηφιακή επανάσταση, πλήθος πληροφοριών διατίθενται εύκολα και γρήγορα για επεξεργασία, αποθήκευση, διανομή και μετάδοση. Η ταυτόχρονη αλματώδης πρόοδος στην επιστήμη των υπολογιστών και σε άλλες παρεμφερείς τεχνολογίες σε συνδυασμό με την όλο και πιο διαδεδομένη χρήση τους σε όλες τις πτυχές της ζωής, για παράδειγμα το World Wide Web, οδήγησε στη συλλογή και αποθήκευση τεράστιου όγκου δεδομένων σε πολλές και μεγάλες βάσεις δεδομένων.

Η ταχύτατη αυτή αύξηση του όγκου των δεδομένων έχει υπερβεί την ανθρώπινη δυνατότητα για κατανόηση χωρίς τη χρήση ισχυρών υπολογιστικών εργαλείων. Έτσι, οι βάσεις στις οποίες αποθηκεύονται γίνονται ''τάφοι'' δεδομένων που σπάνια δέχονται επισκέψεις. Βασική αιτία για την οποία η εκμετάλλευση αυτών των δεδομένων έχει ελκύσει το ενδιαφέρον της βιομηχανίας της πληροφορίας τα τελευταία χρόνια είναι ακριβώς αυτή η ευρεία διάθεσή τους και η επακόλουθη ανάγκη μετατροπής τους σε χρήσιμες πληροφορίες. Η εξόρυξη γνώσης μέσα από αυτά είναι μία ακόμα πρόκληση.

Ως *εξόρυξη γνώσης* (data mining) ορίζεται η διαδικασία εξαγωγής υποκρύπτουσας, προηγουμένως άγνωστης και πιθανώς χρήσιμης πληροφορίας από δεδομένα [13] ή αλλιώς η διερεύνηση και ανάλυση, με τη βοήθεια αυτόματων ή ημι-αυτόματων μέσων, μεγάλου όγκου δεδομένων με στόχο την ανακάλυψη χρήσιμων προτύπων, όπως για παράδειγμα κανόνων συσχέτισης [13]. Με τη βοήθεια κατάλληλα ανεπτυγμένων υπολογιστικών εργαλείων τα αποθηκευμένα δεδομένα αναλύονται και αποκαλύπτουν σημαντικά πρότυπα συνεισφέροντας σημαντικά σε στρατηγικές επιχειρήσεων, βάσεις γνώσεων όσο και στην επιστημονική και ιατρική έρευνα.

1.2 Αντικείμενο της διπλωματικής

Σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη μίας μεθόδου επαναχρησιμοποίησης αποτελεσμάτων που έχουν προκύψει από διαδικασίες εξόρυξης συχνών προτύπων (frequent pattern mining). Συγκεκριμένα, προτείνεται και υλοποιείται μία μέθοδος *ένωσης* δύο FP-trees που αντιστοιχούν σε δύο βάσεις δεδομένων με δοσοληψίες πάνω στο ίδιο σύνολο αντικειμένων και τα οποία έχουν κατασκευαστεί σε προηγούμενη εφαρμογή της data mining μεθόδου FP-growth. Στόχος είναι το mining της ένωσης των δύο βάσεων να γίνεται γρηγορότερα από την κλασική εφαρμογή της FP-growth η οποία προϋποθέτει τη κατασκευή του FP-tree για την ενωμένη βάση από την αρχή.

Ας θεωρήσουμε για παράδειγμα μία αλυσίδα supermarket η οποία έχει υποκαταστήματα σε διάφορες περιοχές της Αθήνας. Κάθε υποκατάστημα συλλέγει και αποθηκεύει τις αποδείξεις πωλήσεων, δημιουργώντας έτσι μία βάση με δοσοληψίες, όπου ως δοσοληψία χαρακτηρίζεται το σύνολο των προϊόντων που αγοράστηκαν από κάθε πελάτη, όπως αυτά καταγράφονται στις αποδείξεις. Ανά τακτά χρονικά διαστήματα, το υποκατάστημα εφαρμόζει κάποια μέθοδο εξόρυξης συχνών προτύπων στη βάση του με στόχο τη μελέτη των αγοραστικών συνηθειών των πελατών του για την χάραξη αποτελεσματικότερης στρατηγικής πωλήσεων (π.χ. τρόπος διάταξης των διατιθέμενων προϊόντων στα ράφια, πολιτική ειδικών προσφορών, βελτίωση εξυπηρέτησης σε τμήματα του καταστήματος κ.τ.λ).

Το κεντρικό κατάστημα όμως θέλει να έχει μία γενική επισκόπηση της αγοραστικής κίνησης των προϊόντων που διαθέτει, καθιστώντας απαραίτητη την εφαρμογή των μεθόδων εξόρυξης συχνών προτύπων στο σύνολο των δοσοληψιών όλων των υποκατάστημάτων του. Η κλασική προσέγγιση του προβλήματος αυτού είναι η ενοποίηση των βάσεων των υποκαταστημάτων και η εφαρμογή των μεθόδων στη νέα ενιαία βάση δοσοληψιών. Το ερώτημα που τίθεται είναι, πώς θα ήταν δυνατόν να εκμεταλλευτεί κανείς τα αποτελέσματα της εφαρμογής των μεθόδων εξόρυξης στο κάθε υποκατάστημα χωριστά ώστε η διαδικασία εξόρυξης συχνών προτύπων για την ένωση των βάσεων να γίνεται γρηγορότερα.

Μία από τις αποδοτικότερες μεθόδους που έχουν προταθεί μέχρι σήμερα για την εξόρυξη συχνών προτύπων από μία βάση με δοσοληψίες, και που υιοθετείται στην παρούσα εργασία, είναι η FP-growth. Η εφαρμογή της γίνεται πάνω σε μία πρότυπη δομή που ονομάζεται FP-tree και η οποία πρόκειται ουσιαστικά για ένα εκτεταμένο δέντρο προθεμάτων (prefix tree) κατάλληλα κατασκευασμένο για την αποθήκευση συμπιεσμένης, καίριας πληροφορίας αναφορικά με τα συχνά πρότυπα της βάσης [8].

Βάσει αυτής της δομής, θα αναπτυχθεί μία νέα μέθοδος που θα υλοποιεί το `merge` δύο

αποθηκευμένων FP-trees, που αντιστοιχούν σε δύο βάσεις με δοσοληψίες πάνω στο ίδιο σύνολο αντικειμένων, κατασκευάζοντας έτσι το FP-tree της ένωσής τους. Η αποδοτικότητα της μεθόδου αποδεικνύεται από πειραματικές εφαρμογές, οι οποίες δείχνουν ότι, με την προτεινόμενη μέθοδο, η εξόρυξη συχνών προτύπων για την ένωση των βάσεων γίνεται, στις πλείστες των περιπτώσεων, πολύ πιο γρήγορα από την κλασική εφαρμογή της FP-growth, κατά την οποία η κατασκευή του FP-tree ακολουθεί την κλασική διαδικασία.

ΚΕΦΑΛΑΙΟ 2

Εξόρυξη Συχνών Προτύπων

Η εξόρυξη συχνών προτύπων παίζει πρωτεύοντα ρόλο στην εύρεση κανόνων συσχέτισης και αιτιωδών σχέσεων, μέγιστων και κλειστών προτύπων, σειριακών και πολυδιάστατων προτύπων, καθώς και άλλων σημαντικών εργασιών εξόρυξης γνώσης. Στο κεφάλαιο αυτό, θα οριστούν αρχικά κάποιες απαραίτητες βασικές έννοιες, στη συνέχεια θα παρουσιαστούν οι αλγόριθμοι που έχουν αναπτυχθεί έως σήμερα για την εξόρυξη συχνών προτύπων ενώ θα δοθεί ιδιαίτερη έμφαση στον αλγόριθμο FP-growth με αναλυτική περιγραφή της λειτουργίας του.

2.1 Βασικές έννοιες

Έστω ένα σύνολο αντικειμένων Σ και μία βάση δεδομένων DB η οποία περιλαμβάνει δοσοληψίες πάνω στο Σ. Κάθε δοσοληψία αποτελείται από ένα σύνολο T, τέτοιο ώστε να ισχύει $T \subseteq \Sigma$, και έχει τις εξής ιδιότητες:

- Χαρακτηρίζεται από ένα μοναδικό αναγνωριστικό για να διακρίνεται από τις υπόλοιπες δοσοληψίες που περιέχονται στη βάση
- Τα αντικείμενα που την απαρτίζουν είναι αταξινόμητα
- Κάθε αντικείμενο μπορεί να εμφανιστεί το πολύ μία φορά σε κάθε δοσοληψία

Μήκος δοσοληψίας ορίζεται το πλήθος των διαφορετικών αντικειμένων που περιέχει.

Τέτοιες δοσοληψίες ανήκουν στην ευρύτερη κατηγορία των *set-valued attributes*, δηλαδή γνωρισμάτων που έχουν ως τιμή σύνολα αντικειμένων. Τα γνωρίσματα αυτά είναι ένας φυσικός και περιεκτικός τρόπος μοντελοποίησης πολύπλοκων πληροφοριών που χρησιμοποιείται ευρέως στις αντικειμενοστραφείς και αντικειμενο-σχεσιακές βάσεις δεδομένων, την ανάκτηση πληροφοριών (information retrieval), την εξόρυξη δεδομένων κ.τ.λ. Ο τύπος των αντικειμένων εξαρτάται

από την οντότητα που μοντελοποιείται. Ένα παράδειγμα τέτοιου γνωρίσματος είναι το σύνολο των αλληλίων που βρίσκονται σε μία συγκεκριμένη θέση του χρωμοσώματος για κάθε άνθρωπο στη βάση των ανθρώπων [12].

Στον Πίνακα 2.1 φαίνεται ένα στιγμιότυπο μιας βάσης δοσοληψιών, όπου οι δοσοληψίες μπορούν να παριστάνουν τις συναλλαγές ενός καταστήματος λιανικής πώλησης (π.χ. supermarket), με το κάθε αντικείμενο να αντιπροσωπεύει ένα ξεχωριστό είδος προϊόντος, ή ένα σύνολο από χημικές ενώσεις, με το κάθε αντικείμενο να αντιστοιχεί σε ένα χημικό στοιχείο της ένωσης.

TID	**Αντικείμενα**
100	f, a, c, d, g, i, m, p
200	a, b, c, f, l, m, o
300	b, f, h, j, o
400	b, c, k, s, p
500	a, f, c, e, l, p, m, n

Πίνακας 2.1: Παράδειγμα βάσης με δοσοληψίες.

Με τον όρο *πρότυπα* (patterns) εννοούμε κάθε δυνατό συνδυασμό των αντικειμένων στο Σ. Επομένως, για το σύνολο των δυνατών προτύπων, έστω P, ισχύει $P = \Sigma^*$. Το μέγεθος ενός προτύπου αντιστοιχεί στο πλήθος των αντικειμένων που περιλαμβάνει. Έτσι, ένα πρότυπο μεγέθους-1 είναι ένα μονοσύνολο που μπορεί να περιέχει οποιοδήποτε από τα αντικείμενα του Σ, ένα πρότυπο μεγέθους-2 είναι ένα σύνολο που περιέχει οποιαδήποτε δύο από τα αντικείμενα του Σ κ.ο.κ.

Ως *support* ενός προτύπου A, όπου $A \subseteq \Sigma$, ορίζεται το πλήθος των δοσοληψιών της DB που περιέχουν το A. Επομένως, οι τιμές που μπορεί να πάρει το support ενός προτύπου είναι $\text{support}_\text{A} \in (0, n] \subset \mathbb{N}$, όπου n το συνολικό πλήθος των δοσοληψιών στη βάση. Συνήθως, το support ορίζεται και σαν ποσοστό επί τις εκατό των δοσοληψιών που περιέχουν ένα συγκεκριμένο πρότυπο. Δηλαδή,

$$\text{support}_\text{A} = \frac{\text{πλήθος δοσοληψιών που περιέχουν το } A}{\text{συνολικό πλήθος δοσοληψιών στη βάση}}$$

Για να θεωρηθεί ένα πρότυπο *συχνό*, θα πρέπει το πλήθος των εμφανίσεών του στη βάση DB να μην είναι μικρότερο από ένα ελάχιστο προκαθορισμένο κατώφλι (support threshold) ξ.

Ας υποθέσουμε τώρα ότι ζητείται η εύρεση των συχνών προτύπων μίας βάσης, η οποία αποτελείται από δοσοληψίες πάνω σε k διαφορετικά αντικείμενα. Η brute force προσέγγιση του προβλήματος θα ήταν η παραγωγή όλων των υποψήφιων συχνών προτύπων, τα οποία για k αντικείμενα είναι $2^k - 1$, και η σύγκρισή τους με τα πρότυπα που εμφανίζονται στις δοσοληψίες της βάσης ώστε να καθοριστεί το support τους. Επειδή το k, σε πραγματικές εφαρμογές, μπορεί

να πάρει μεγάλες τιμές, ο χώρος αναζήτησης των συχνών προτύπων γίνεται εκθετικά μεγάλος. Οι βασικοί άξονες στους οποίους κινούνται οι αλγορίθμοι που έχουν αναπτυχθεί έως σήμερα για την αντιμετώπιση αυτού του προβλήματος, είναι δύο:

- **Μείωση του πλήθους των υποψήφιων συχνών προτύπων**. Προς την κατεύθυνση αυτή, η *Apriori* αρχή, που παρουσιάζεται στη συνέχεια, αποδεικνύεται ιδιαίτερα αποτελεσματική.

- **Μείωση του πλήθους των συγκρίσεων που γίνονται στη βάση**. Αντί να συγκρίνεται κάθε υποψήφιο πρότυπο με κάθε δοσοληψία της βάσης, μπορούμε να μειώσουμε τις συγκρίσεις χρησιμοποιώντας ειδικές δομές που είτε αποθηκεύουν τα υποψήφια σύνολα είτε συμπιέζουν τη βάση.

2.2 *Apriori* αλγόριθμοι

Η *Apriori* προσέγγιση του προβλήματος κινείται προς την κατεύθυνση της μείωσης του πλήθους των υποψήφιων συχνών προτύπων χρησιμοποιώντας την αντι-μονότονη Apriori ευριστική συνάρτηση.

Ορισμός 2.1 (αντί-μονότονη *apriori* ευριστική [2]). *Έστω πρότυπο μεγέθους−k το οποίο δεν είναι συχνό στη βάση. Τότε κανένα από τα πρότυπα μεγέθους−$(k+1)$ που το περιέχουν δε θα είναι συχνό*

Η βασική ιδέα είναι η παραγωγή όλων των υποψήφιων συχνών προτύπων μεγέθους−$(k+1)$ από τα συχνά πρότυπα μεγέθους−k (για $k \geq 1$) και ο έλεγχος της συχνότητάς τους στη βάση. Η διαδικασία αυτή συνεχίζεται επαναληπτικά μέχρι το μέγιστο δυνατό μέγεθος προτύπου, που ταυτίζεται με το μέγιστο μήκος των δοσοληψίων. Αναλυτικά, ο apriori αλγόριθμος δε θα παρουσιαστεί, αν και έχει ενδιαφέρον, καθώς δεν εξυπηρετεί τους σκοπούς της διπλωματικής.

Παρότι ο apriori αλγόριθμος αποτελεί σημαντική βελτίωση της brute force προσέγγισης, συνεχίζει να παράγει μεγάλο αριθμό υποψήφιων προτύπων, πολλά από τα οποία τελικά δεν είναι συχνά. Υλοποιήσεις του αλγορίθμου [4],[3] χρησιμοποιώντας διαφορετικές αναπαραστάσεις των δεδομένων της βάσης [13] έχουν βελτιστοποιήσει κατά πολύ την επίδοσή του, ωστόσο στις περιπτώσεις που η βάση είναι πολύ μεγάλη και το μέγεθος των δοσοληψιών επίσης μεγάλο, το κόστος από τις συνεχείς πράξεις I/O αυξάνεται δραματικά.

2.3 Ο αλγόριθμος FP-growth

Μία διαφορετική αντιμετώπιση είναι η προσπάθεια μείωσης των συγκρίσεων που απαιτούνται για τον καθορισμό των συχνών προτύπων. Στη βάση αυτή, προτάθηκε μία μέθοδος εξόρυξης του συνόλου των συχνών προτύπων *χωρίς την παραγωγή υποψηφίων συνόλων*. Η μέθοδος αυτή λέγεται **frequent pattern growth** [8], ή απλά **FP-growth**, και ακολουθεί μία στρατηγική *divide-and-conquer* συμπιέζοντας την βάση και οργανώνοντας τα συχνά σύνολα αντικειμένων της σε μία πρότυπη δομή που ονομάζεται **frequent pattern tree**, ή εν συντομία **FP-tree**, βασιζόμενη στη ιδέα προβολής των προτύπων σε δομή λεξικογραφικού δέντρου [1]. Στη συνέχεια, διαιρεί την συμπιεσμένη βάση σε εξαρτημένες βάσεις (conditional databases), κάθε μία από τις οποίες συσχετίζεται με ένα συχνό αντικείμενο, και εκτελεί την εξόρυξη σε κάθε βάση ξεχωριστά.

2.3.1 Σχεδίαση και κατασκευή FP-tree

Το FP-tree αποτελεί μία αναπαράσταση της βάσης των δοσοληψιών που διακρίνεται για τη συμπαγή δομή της. Κατασκευάζεται αντιστοιχίζοντας τις δοσοληψίες σε μονοπάτια του δέντρου με στόχο, όσες δοσοληψίες μοιράζονται κοινά αντικείμενα να μοιράζονται και το ίδιο μονοπάτι στο δέντρο. Όσο πιο πολλές δοσοληψίες αντιστοιχούν στο ίδιο μονοπάτι, τόσο πιο συμπαγής είναι η δομή. Σε περίπτωση που το μέγεθος του FP-tree είναι τόσο μικρό ώστε να χωράει στην κύρια μνήμη, η διαδικασία της εξόρυξης μπορεί να ολοκληρωθεί χρησιμοποιώντας μόνο το δέντρο που βρίσκεται στη μνήμη και αποφεύγοντας τις επαναληπτικές επισκέψεις στη βάση που βρίσκεται αποθηκευμένη στον σκληρό δίσκο. Πριν τον επίσημο ορισμό του FP-tree, θα ήταν χρήσιμο να μελετήσουμε ένα παράδειγμα.

Παράδειγμα 2.1.

Ας θεωρήσουμε την βάση δοσοληψιών του Πίνακα 2.1 και ως mininum support threshold $\xi = 3$. Για τη σχεδίαση μιας συμπαγούς δομής η οποία θα περιέχει όλη την απαραίτητη πληροφορία για την εξόρυξη του συνόλου των συχνών προτύπων που εμφανίζονται στη βάση, μπορούν να γίνουν οι παρακάτω παρατηρήσεις.

- Καθώς μόνο τα συχνά αντικείμενα παίζουν ρόλο στην εξόρυξη των συχνών προτύπων, πρέπει κατ' αρχήν να γίνει μία πρώτη σάρωση της βάσης ώστε να καθοριστεί το σύνολο των συχνών αντικειμένων, βρίσκοντας ταυτόχρονα και την συχνότητα εμφάνισης τους (support).

- Αν υπάρχουν περισσότερες από μία δοσοληψίες που περιέχουν τα ίδια συχνά αντικείμενα, τότε αυτές μπορούν να συγχωνευθούν ενημερώνοντας αντίστοιχα το support count κάθε

αντικειμένου. Σε περίπτωση, μάλιστα, που τα συχνά αντικείμενα κάθε δοσοληψίας είναι ταξινομημένα, τότε ο έλεγχος για πιθανή συγχώνευση μπορεί να γίνει ιδιαίτερα εύκολα.

□ Αν υπάρχουν δοσοληψίες στις οποίες τα ταξινομημένα συχνά αντικείμενα τους έχουν κοινό πρόθεμα, τότε τα κοινά αντικείμενα μπορούν να γίνουν merge στην δομή, αρκεί το support count τους να ενημερωθεί αναλόγως. Ειδικά στην περίπτωση που τα συχνά αντικείμενα κάθε δοσοληψίας έχουν ταξινομηθεί με φθίνουσα σειρά συχνότητας, η πιθανότητα ύπαρξης κοινού προθέματος αυξάνεται σημαντικά.

Με βάση τις παραπάνω παρατηρήσεις, το FP-tree κατασκευάζεται ως εξής:

Πρώτα σαρώνεται ολόκληρη η βάση και συλλέγονται τα support των αντικειμένων. Έτσι, καθορίζεται ποια από τα αντικείμενα είναι συχνά, καθώς εξαιρούνται τα αντικείμενα για τα οποία το support είναι μικρότερο από ξ. Το σύνολο των αντικειμένων αυτών ταξινομείται με φθίνουσα σειρά συχνότητας. Η λίστα συχνών αντικειμένων που προκύπτει είναι:

$$\textit{f-list}= \{(f:4),(c:4),(a:3),(b:3),(m:3),(p:3)\}$$

Η σειρά αυτή είναι πολύ σημαντική καθώς κάθε κλάδος του δέντρου θα κατασκευάζεται σύμφωνα με αυτή.

Στη συνέχεια σαρώνεται για δεύτερη φορά η βάση. Για κάθε μία δοσοληψία, τα συχνά αντικείμενά της, ταξινομούνται με βάση τη σειρά συχνότητας και εισάγονται στο δέντρο. Η ταξινομημένη σειρά συχνών αντικειμένων κάθε δοσοληψίας για το συγκεκριμένο παράδειγμα φαίνεται στον Πίνακα 2.2.

TID	**Αντικείμενα**	**Συχνά αντικείμενα**
100	f,a,c,d,g,i,m,p	f,c,a,m,p
200	a,b,c,f,l,m,o	f,c,a,b,m
300	b,f,h,j,o	f,b
400	b,c,k,s,p	c,b,p
500	a,f,c,e,l,p,m,n	f,c,a,m,p

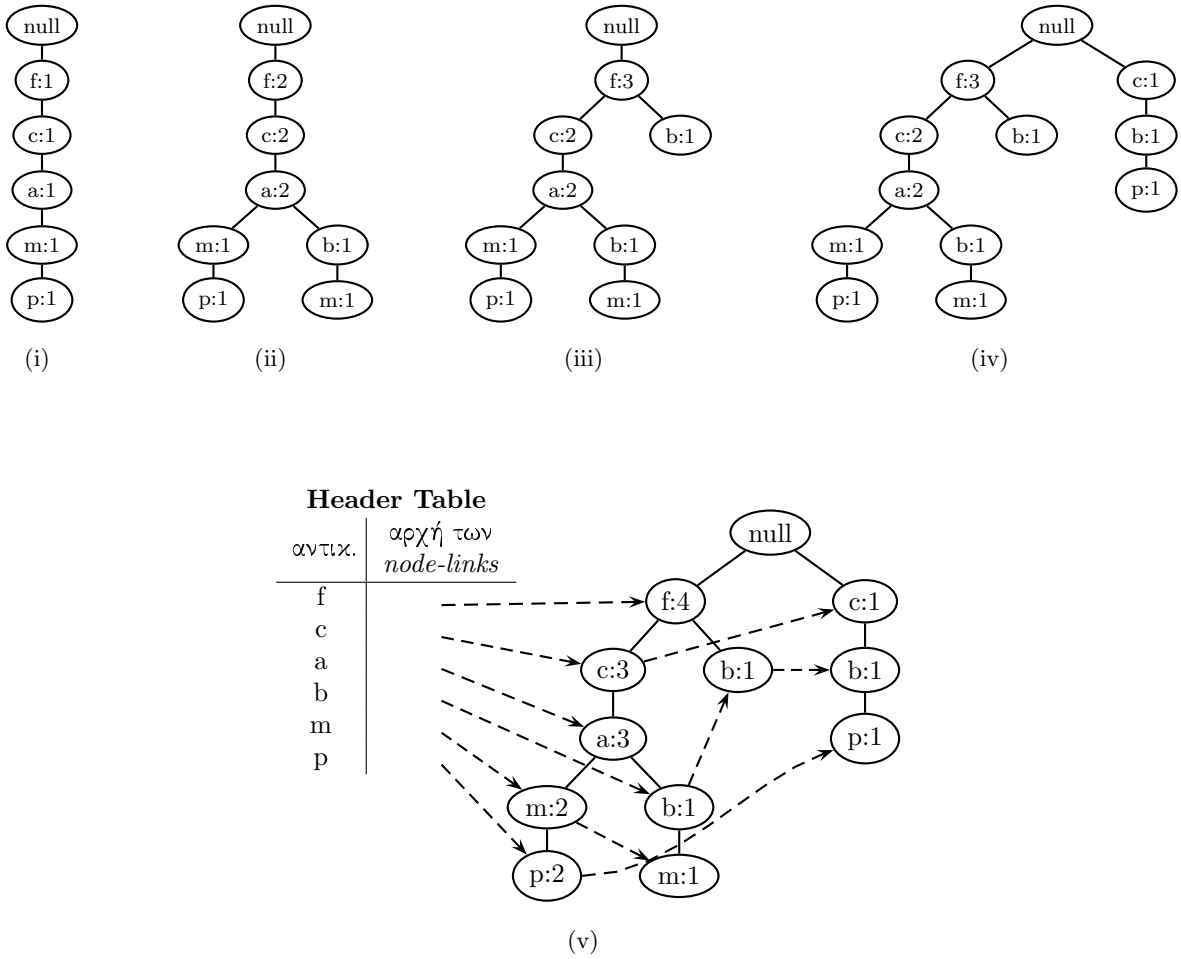
Πίνακας 2.2: Ταξινόμηση συχνών αντικειμένων κάθε δοσοληψίας.

Για την εισαγωγή των συχνών αντικειμένων κάθε δοσοληψίας στο δέντρο ακολουθείται η παρακάτω διαδικασία:

1. Το δέντρο αρχικοποιείται έχοντας έναν κόμβο, την ρίζα, η οποία παίρνει την τιμή `null`.

2. Η σάρωση της πρώτης δοσοληψίας οδηγεί στην κατασκευή του πρώτου κλάδου του δέντρου που αποτελείται από τα αντικείμενα $\langle(f:1),(c:1),(a:1),(m:1),(p:1)\rangle$.

3. Η δεύτερη δοσοληψία έχει κοινό πρόθεμα με την προηγούμενη. Επομένως, το support count των αντικειμένων f, c και a αυξάνεται κατά 1, δημιουργείται ένας κόμβος $(b:1)$ ως παιδί του $(a:2)$ και ένας ακόμα, ο $(m:1)$ ως παιδί του $(b:1)$.



Σχήμα 2.1: Παράδειγμα κατασκευής FP-tree.

4. Η τρίτη δοσοληψία θα μοιραστεί τον κόμβο f με το f-προθεματικό δέντρο (f-prefix tree) και θα προστεθεί ένας κόμβος $(b:1)$ ως παιδί του $(f:3)$.

5. Η τέταρτη δοσοληψία δεν έχει κοινό πρόθεμα με καμία από τις προηγούμενες. Έτσι, κατασκευάζεται ένας νέος κλάδος που ξεκινά από τη ρίζα $\langle (c:1), (b:1), (p:1) \rangle$.

6. Η τελευταία δοσοληψία ταυτίζεται με την πρώτη, επομένως αρκεί να αυξηθεί το count των κόμβων του αντίστοιχου μονοπατιού στο δέντρο κατά 1 μονάδα.

Η διαδικασία κατασκευής του FP-tree βήμα προς βήμα φαίνεται στο Σχήμα 2.1. □

Από τον τρόπο κατασκευής του FP-tree προκύπτει η παρακάτω ιδιότητα.

Ιδιότητα 2.1. *Κάθε συχνό αντικείμενο, συμμετέχει στο πρότυπο που ορίζεται από κάθε μονοπάτι του FP-tree που το περιέχει, με support count ίσο με το μικρότερο count των αντικειμένων που απαρτίζουν το μονοπάτι.*

Ορισμός 2.2 (FP-tree [8])**.**

- Το FP-tree αποτελείται από τη ρίζα που έχει την τιμή `null`, ένα σύνολο από υποδέντρα προθεμάτων (prefix subtrees) ως παιδιά της ρίζας και έναν πίνακα επικεφαλίδων (header table) συχνών αντικειμένων.

- Κάθε κόμβος των υποδέντρων αποτελείται από τρία πεδία: το όνομα του αντικειμένου του κόμβου, *item-name*, το *count*, που είναι το πλήθος των δοσοληψιών που περιέχουν το μονοπάτι του υποδέντρου μέχρι το συγκεκριμένο κόμβο, και το *node-link*, που είτε δείχνει στον επόμενο κόμβο του δέντρου με το ίδιο αντικείμενο είτε παίρνει την τιμή `null` αν δεν υπάρχει άλλος τέτοιος κόμβος.

- Κάθε εγγραφή του header-table συχνών αντικειμένων έχει δύο πεδία: το όνομα του αντικειμένου *item-name* και έναν δείκτη *head of node-links* που δείχνει στην πρώτη εμφάνιση του συγκεκριμένου αντικειμένου στο δέντρο.

Αλγόριθμος 2.1 Κατασκευή FP-tree.

Είσοδος : Μία βάση δοσοληψιών *DB* και ένα ελάχιστο support ξ.

Έξοδος : Το FP-tree που αντιστοιχεί στη βάση.

Μέθοδος :

1. Σάρωσε τη βάση *DB* και βρες το σύνολο συχνών αντικειμένων της βάσης όπως και τα αντίστοιχα supports τους. Ταξινόμησε το με φθίνουσα σειρά $\Rightarrow$ *f-list*.

2. Αρχικοποίησε το FP-tree, έστω T, κατασκευάζοντας τη ρίζα με τιμή `null`. Για κάθε δοσοληψία *Trans* της βάσης, ακολούθησε την παρακάτω διαδικασία: Βρες τα αντικείμενα της δοσοληψίας που είναι συχνά και ταξινόμησέ τα με βάση την *f-list*. Έστω ότι η ταξινομημένη λίστα των συχνών αντικειμένων της *Trans* είναι της μορφή $[p|P]$, όπου p είναι το πρώτο αντικείμενο και P η υπόλοιπη λίστα. Κάλεσε τη μέθοδο *insert_tree*$([p|P], T)$.

Αλγόριθμος 2.2 Εισαγωγής δοσοληψίας στο FP-tree.

procedure `insert_tree`$([p|P], T)$

(1) **Αν** υπάρχει N παιδί του T με $N.item\text{-}name = p.item\text{-}name$ **τότε**

(2) αύξησε το *count* του N κατά 1;

(3) **αλλιώς** {

(4) κατασκεύασε έναν νέο κόμβο N στο δέντρο;

(5) κρέμασέ τον απ' το T;

(6) θέσε το *count* του ίσο με 1;

(7) συνέδεσέ τον μέσω του *node-link* με τον επόμενο κόμβο στο δέντρο που έχει το ίδιο *item-name*; }

(8) **Αν** $P \neq \emptyset$ **τότε**

(9) κάλεσε αναδρομικά την $insert\_tree(P, N)$;

Ιδιότητες FP-tree

Ο αλγόριθμος κατασκευής του FP-tree που περιγράφηκε διασφαλίζει τόσο τη συμπαγή μορφή της δομής όσο και την πληρότητά της για την εξόρυξη του συνόλου των συχνών προτύπων της.

Λήμμα 2.1. *Δεδομένης μίας βάσης DB και ενός ελάχιστου support ξ:*

- *το μέγεθος του αντίστοιχου FP-tree, χωρίς να συμπεριλαμβάνεται η ρίζα του δέντρου, φράσσεται άνω από $\sum_{T \in DB} |freq(T)|$*
- *το ύψος του δέντρου φράσσεται άνω από $max_{T \in DB}\{|freq(T)|\}$*

όπου T δοσοληψία και $freq(T)$ το σύνολο των συχνών αντικειμένων της δοσοληψίας.

Απόδειξη. Σύμφωνα με τον αλγόριθμο κατασκευής του FP-tree, κάθε δοσοληψία T της βάσης μπορεί να συνεισφέρει το πολύ ένα μονοπάτι στο δέντρο, ενώ παραπάνω από μία δοσοληψίες μπορεί να αντιστοιχούν στο ίδιο μονοπάτι, αν έχουν ακριβώς τα ίδια συχνά αντικείμενα. Η ρίζα είναι ο μόνος κόμβος που κατασκευάζεται ανέξαρτητα από τα αντικείμενα των δοσοληψιών, ενώ κάθε άλλος κόμβος περιλαμβάνει μόνο το count του αντικειμένου και το nodelink. Εξού και το άνω φράγμα του μεγέθους του δέντρου.

Και πάλι σύμφωνα με τον αλγόριθμο κατασκευής, το ύψος του δέντρου, χωρίς να περιλαμβάνεται η ρίζα, ισούται με το μέγιστο ύψος των προθεματικών υπο-δέντρων, δηλαδή με το μέγιστο πλήθος συχνών αντικειμένων που έχουν οι δοσοληψίες της βάσης. Εξού και το άνω φράγμα του ύψους του δέντρου. □

Λήμμα 2.2. *Δεδομένης μίας βάσης DB και ενός ελάχιστου support ξ, το σύνολο των συχνών προτύπων της βάσης μπορεί να εξαχθεί από το αντίστοιχο FP-tree.*

Απόδειξη. Βάσει του αλγορίθμου κατασκευής του FP-tree κάθε δοσοληψία συνεισφέρει το πολύ ένα μονοπάτι στο δέντρο. Έστω ένα μονοπάτι του δέντρου $a_1, a_2, \ldots, a_k$ και c_{a_k} το count του τελευταίου κόμβου του μονοπατιού που έχει το αντικειμένου a_k και έστω c'_{a_k} το άθροισμα των counts όλων των παιδιών του a_k. Τότε, το μονοπάτι του δέντρου αντιστοιχεί σε $c_{a_k} - c'_{a_k}$ δοσοληψίες της βάσης. Επομένως, το δέντρο περιλαμβάνει κάθε μία από τις δοσοληψίες της βάσης χωρίς διπλές καταχωρήσεις. □

2.3.2 Περιγραφή μεθόδου FP-growth

Η FP-growth είναι μία μέθοδος παραγωγής όλων των συχνών προτύπων ενός FP-tree διασχίζοντάς το από κάτω προς τα πάνω (Bottom Up). Για την καλύτερη κατανόηση της λογικής της μεθόδου και του τρόπου με τον οποίο εκμεταλλεύεται την δομή FP-tree, θα μελετήσουμε πρώτα ένα παράδειγμα.

Παράδειγμα 2.2.

Ας θεωρήσουμε το FP-tree του Σχήματος 2.2 (v). Για να βρούμε τα πρότυπα που περιέχουν κάθε ένα από τα συχνά αντικείμενα της *f-list* αρκεί, σύμφωνα με την Ιδιότητα 2.1 να ξεκινήσουμε από το header table του FP-tree και για κάθε αντικείμενο να βρούμε τα μονοπάτια στο δέντρο στα οποία συμμετέχουν ακολουθώντας τα node-links. Η διαδικασία ξεκινά από το τέλος του header table.

αντικείμενο *p* Το συχνό πρότυπο που προκύπτει άμεσα είναι το $\langle p : 3\rangle$, ενώ στο δέντρο εμφανίζεται σε δύο μονοπάτια. Το πρώτο είναι το $\langle (f : 4), (c : 3), (a : 3), (m : 2), (p : 2)\rangle$ και δείχνει ότι η τα αντικείμενα $\{f, c, a, m, p\}$ εμφανίζονται μαζί στη βάση δύο φορές. Το ίδιο μονοπάτι δείχνει ότι τα αντικείμενα $\{f, c, a, m\}$ εμφανίζονται μαζί επίσης δύο φορές, τα $\{f, c, a\}$ τρεις, τα $\{f, c\}$ τρεις και το $\{f\}$ τέσσερις. Επειδή όμως εδώ αναζητούμε τα πρότυπα που περιέχουν το p, μόνο το προθεματικό μονοπάτι του p (p's prefix path) $\langle (f : 2), (c : 2), (a : 2), (m : 2)\rangle$ είναι αυτό που μας ενδιαφέρει. Το δεύτερο μονοπάτι $\langle (c : 1), (b : 1), (p : 1)\rangle$, σχηματίζει το prefix path του p $\langle (c : 1), (b : 1)\rangle$. Από αυτά τα prefix-paths κατασκευάζεται η conditional pattern base του p, $\{(fcam : 2), (cb : 1)\}$, δηλαδή η βάση με τα αντικείμενα που υπάρχουν δεδομένης της ύπαρξης του p. Για την βάση αυτή, κατασκευάζεται με τον γνωστό τρόπο το αντίστοιχο FP-tree που ονομάζεται

conditional FP-tree του p, με έναν μόνο κλάδο, τον $\langle(c:3)\rangle$. Επομένως, το δεύτερο συχνό πρότυπο που προκύπτει είναι το $cp:3$. Εδώ λήγει και η διαδικασία εύρεσης συχνών προτύπων που περιέχουν το αντικείμενο p.

αντικείμενο m Το συχνό πρότυπο που προκύπτει άμεσα είναι το $\langle m:3\rangle$. Το m εμφανίζεται σε δύο μονοπάτια του δέντρου, στον κλάδο $\langle(f:4),(c:3),(a:3),(m:2)\rangle$ και στον κλάδο $\langle(f:4),(c:3),(a:3),(b:1),(m:1)\rangle$, απ' όπου προκύπτει η conditional pattern base του αντικειμένου m, $\{(fca:2),(fcab:1)\}$. Το conditional FP-tree που κατασκευάζεται έχει έναν και μόνο κλάδο, τον $\langle(f:3),(c:3),(a:3)\rangle$. Στο δέντρο αυτό θα βρεθούν τα συχνά πρότυπα για τα αντικείμενα a, c, f με αυτή τη σειρά, δεδομένης βέβαια της ύπαρξης του m.

Έτσι, για το αντικείμενο a το συχνό πρότυπο που προκύπτει άμεσα είναι το $\langle am:3\rangle$ ενώ η conditional pattern base του είναι η $\{(fc{:}3)\}$. Αναδρομικά καλείται η διαδιακασία του mining για το conditional FP-tree $\langle(f{:}3,(c{:}3)\rangle|\ am$. Το πρώτο συχνό πρότυπο που προκύπτει απ' αυτή είναι το $cam{:}3$ ενώ η conditional pattern base θα είναι η $\{(f{:}3)\}$ με αντίστοιχο conditional FP-tree το $\langle(f{:}3)\rangle|\ cam$. Και πάλι αναδρομικά προκύπτει άμεσα το συχνό πρότυπο $\langle fcam:3\rangle$.

Τώρα, επιστρέφουμε στο conditional FP-tree του m και καλούμε τη διαδικασία για το αντικείμενο c. Το συχνό πρότυπο που προκύπτει άμεσα είναι το $\langle cm:3\rangle$ ενώ η conditional pattern base είναι η $\{(f{:}3)\}$ και το conditional FP-tree της $\langle(f{:}3)\rangle|\ cm$. Σε αυτό, το άμεσο συχνό πρότυπο είναι το $\langle fcm{:}3\rangle$ αλλά η conditional pattern base $= \emptyset$, οπότε η αναδρομική διαδικασία τερματίζεται.

Τέλος, για το αντικείμενο f από το αρχικό conditional FP-tree προκύπτει μόνο το συχνό πρότυπο $fm{:}3$ και η διαδικασία τερματίζεται ολοκληρωτικά αφού δεν υπάρχει conditional pattern base και όλα τα αντικείμενα του αρχικού conditional FP-tree έχουν ελεγχθεί.

Κατά συνέπεια, τα συχνά πρότυπα που περιέχουν το m είναι τα $fm{:}3$, $cm{:}3$, $am{:}3$, $fcm{:}3$, $fam{:}3$, $cam{:}3$ και $fcam{:}3$. Όπως παρατηρούμε, τα συχνά πρότυπα που προκύπτουν από ένα FP-tree το οποίο περιέχει μόνο ένα απλό μονοπάτι είναι ουσιαστικά όλοι οι δυνατοί συνδυασμοί των αντικειμένων του μονοπατιού. Αυτή την ιδιότητα εκμεταλλεύεται ο αλγόριθμος FP-growth καθώς σε περίπτωση που συναντήσει δέντρο με απλό μονοπάτι θεωρεί το μονοπάτι σαν ένα σύνολο και βρίσκει απλά το δυναμοσύνολό του, χωρίς να μπει στην διαδικασία κατασκευής conditional pattern bases και conditional FP-trees.

αντικείμενο b Το άμεσο πρότυπο είναι το $\langle b:3\rangle$, ενώ το b εμφανίζεται σε τρία μονοπάτια του δέντρου. Τα prefix paths που προκύπτουν είναι τα $\langle(f:1),(c:1),(a:1),(b:1)\rangle$, $\langle(f:1),(b:1)\rangle$ και $\langle(c:1),(b:1)\rangle$. Από αυτά φαίνεται ότι δεν υπάρχουν αντικείμενα που

να εμφανίζονται μαζί με το b παραπάνω από δύο φορές, ώστε να ικανοποιείται το support threshold $\xi = 3$. Επομένως η conditional pattern base του b δεν περιέχει κανένα αντικείμενο και η διαδικασία για το b τερματίζεται.

αντικείμενο *a* Το μοναδικό μονοπάτι του δέντρου που το περιέχει είναι το $\langle (f:4), (c:3), (a:3)\rangle$. Η conditional pattern base είναι η $\{(fc:3)\}$ και το conditional FP-tree $\langle (f:3), (c:3)\rangle$. Και πάλι συναντούμε την περίπτωση απλού μονοπατιού, οπότε τα συχνά πρότυπα που θα προκύψουν θα είναι τα a:3, fa:3, ca:3 και fca:3.

αντικείμενο *c* Το c υπάρχει σε δύο μονοπάτια, το $\langle (f:4), (c:3)\rangle$ και το $\langle (c:1)\rangle$. Από αυτά η προκύπτουσα conditional pattern base θα είναι $\{(f:3)\}$ και το conditional FP-tree το $\langle (f:3)\rangle$. Επομένως, τα συχνά πρότυπα θα είναι c:3 και fc:3.

αντικείμενο *f* Εκτός από το άμεσα προκύπτον πρότυπο f:4 δεν υπάρχουν άλλα, καθώς από το μοναδικό μονοπάτι του δέντρου που περιέχει το f $\langle (f:4)\rangle$ προκύπτει ότι η conditional pattern base= $\emptyset$.

Η πορεία της διαδικασίας και τα συχνά πρότυπα που προκύπτουν φαίνονται στον Πίνακα 2.3.

□

item	**Conditional pattern base**	**Conditional FP-tree**	**Frequent patterns**
p	{(fcam:2), (cb:1)}	⟨(c:3)⟩|p	{p:3}, {cp:3}
m	{(fca:2), (fcab:1)}	⟨(f:3), (c:3), (a:3)⟩|m	{m:3}, {fm:3}, {cm:3}, {am:3} {fcm:3}, {fam:3}, {cam:3}, {fcam:3}
b	{(fca:1),(f:1),(c:1)}	∅	{b:3}
a	{(fc:3)}	⟨(f:3), (c:3)⟩|a	{a:3}, {fa:3}, {ca:3}, {fca:3}
c	{(f:3)}	⟨(f:3)⟩|c	{c:4}, {fc:3}
f	∅	∅	{f:4}

Πίνακας 2.3: Εξόρυξη συχνών προτύπων με την FP-growth.

Ιδιότητα 2.2. *Όταν ένα FP-tree περιέχει απλό μονοπάτι, δηλαδή δεν υπάρχουν διακλαδώσεις κατά μήκος του, τα συχνά πρότυπα που προκύπτουν είναι όλοι οι δυνατοί συνδυασμοί των αντικειμένων που βρίσκονται στους κόμβους του μονοπατιού με support το ελάχιστο count των αντικειμένων του κάθε συνδυασμού.*

Την ιδιότητα αυτή εκμεταλλεύεται η FP-growth με τρόπο που περιγράφεται στο παρακάτω παράδειγμα.

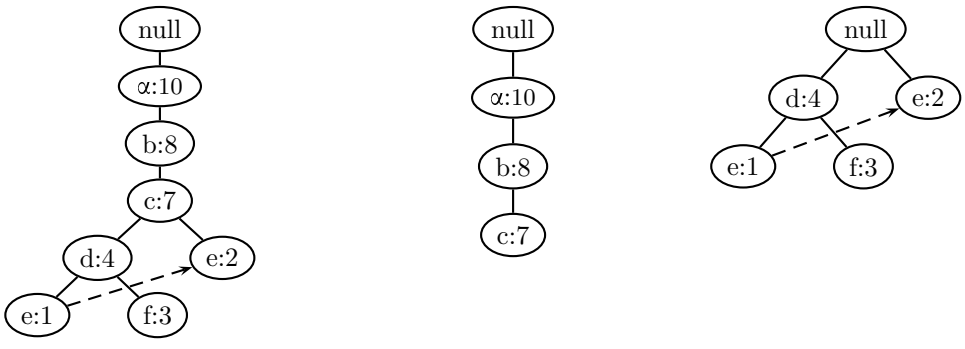
Παράδειγμα 2.3.

Έστω το FP-tree του Σχήματος 2.2(i) και $\xi = 3$. Παρατηρούμε ότι το δέντρο αυτό περιλαμβάνει ένα απλό μονοπάτι από τη ρίζα ως τον κόμβο (c:7). Σε αυτή την περίπτωση η FP-growth μπορεί να χωριστεί σε δύο φάσεις. Πρώτα θα γίνει mining στο απλό μονοπάτι, έπειτα στο υπόλοιπο δέντρο ενώ στο τέλος θα γίνει ένας συνδυασμός των αποτελεσμάτων που θα προκύψουν. Έτσι, για το συγκεκριμένο παράδειγμα το δέντρο χωρίζεται σε δύο τμήματα.

Το πρώτο τμήμα αποτελείται από το απλό μονοπάτι του δέντρου, έστω P, όπως φαίνεται στο Σχήμα 2.2(ii). Σύμφωνα με την Ιδιότητα 2.2, το σύνολο των συχνών προτύπων για το P θα είναι το $freq(P)$ ={(a:10), (b:8), (c:7), (ab:8), (ac:7), (bc:7), (abc:7)}.

Το δεύτερο τμήμα, έστω Q, είναι το δέντρο του Σχήματος 2.2(iii), όπου ο κόμβος (c:7) του αρχικού δέντρου έχει αντικατασταθεί από έναν κόμβο-ρίζα. Εδώ εφαρμόζεται η FP-growth με τον τρόπο που περιγράφηκε στο Παράδειγμα 2.2 και προκύπτουν τα αποτελέσματα $freq(Q)$ ={(d:4), (e:3), (f:3), (df:3)}.

Τώρα μένει ο συνδυασμός αυτών των αποτελεσμάτων. Κάθε ένα από σύνολα των αντικειμένων που αντικατροπτρίζονται στο δέντρο Q έχει το $freq(P)$ ως conditional pattern base. Αυτό συμβαίνει επειδή σύμφωνα με την Ιδιότητα 2.1 κάθε αντικείμενο συμμετέχει στο pattern που ορίζεται από το μονοπάτι του δέντρου που το περιέχει. Για παράδειγμα, το αντικείμενο d, εμφανίζεται μαζί με τα a, b, c του απλού μονοπατιού 4 φορές και άρα τα συχνά πρότυπα στα οποία συμμετέχει θα είναι τα (ad:4), (ab:4), (ac:4), (abd:4), (acd:4), (bcd:4), ($abcd$:4), δηλαδή ουσιαστικά το (d:4)×$freq(P)$. Αυτό ισχύει για κάθε μέλος του συνόλου $freq(Q)$ και άρα τα συχνά πρότυπα που παράγονται από τον συνδυασμό των προηγούμενων αποτελεσμάτων θα είναι το $freq(Q) \times freq(P)$. Τελικά, το σύνολο των συχνών προτύπων που παράγει η FP-growth θα είναι το $freq(P) \cup freq(Q) \cup \{freq(Q) \times freq(P)\}$. □



(i) FP-tree με απλό μονοπάτι

(ii) Απλό μονοπάτι P

(iii) Τμήμα με διακλαδώσεις Q

Σχήμα 2.2: FP-growth σε δέντρο με απλό μονοπάτι.

Αρχές της FP-growth

Τα παραδείγματα 2.2 και 2.3 φανερώνουν τις ιδιότητες και τις βασικές αρχές στις οποίες βασίζεται η FP-growth συμπεριλαμβανομένης και της Ιδιότητας 2.2 που αναφέρθηκε παραπάνω.

Ιδιότητα 2.3. *Για ένα συχνό αντικείμενο a_i, όλα τα συχνά πρότυπα που το περιέχουν μπορούν να βρεθούν ξεκινώντας από το header table του FP-tree και ακολουθώντας τα node-links.*

Ιδιότητα 2.4. *Η εύρεση των συχνών προτύπων που έχουν επίθεμα a_i γίνεται απομονώνοντας από το FP-tree μόνο τα prefix paths των κόμβων που έχουν τιμή a_i ενώ το count των κόμβων του μονοπατιού παίρνει την τιμή του count του κόμβου a_i σε κάθε μονοπάτι.*

Λήμμα 2.3. *(Fragment growth) Έστω α ένα σύνολο αντικειμένων στη βάση DB, B η conditional pattern base του α και β ένα σύνολο αντικειμένων στην B. Τότε, το support του συνόλου $\alpha \cup \beta$ στη βάση DB ισούται με το support το β στη βάση B.*

Απόδειξη. Από τον ορισμό της conditional pattern base, κάθε υπο-δοσοληψία στην B υπάρχει με την προϋπόθεση ύπαρξης του α στην αντίστοιχη δοσοληψία της DB. Έτσι, αν ένα σύνολο αντικειμένων β εμφανίζεται x φορές στη B, τότε εμφανίζεται μαζί με το α ακριβώς x φορές στη DB. Επίσης, επειδή και κάθε αντικείμενο του β, ανήκει στην conditional pattern base του α, το $\alpha \cup \beta$ εμφανίζεται στη βάση DB ακριβώς x φορές. □

Από το Λήμμα 2.3 προκύπτει άμεσα το παρακάτω πόρισμα.

Πόρισμα 2.1. *(Pattern growth) Έστω α ένα σύνολο συχνών αντικειμένων στην βάση DB, B η conditional pattern base του α και β ένα σύνολο αντικειμένων στην B. Τότε, το σύνολο $\alpha \cup \beta$ είναι συχνό στη βάση DB αν και μόνον αν το β είναι συχνό στη βάση B.*

Λήμμα 2.4. *Έστω ένα FP-tree T το οποίο αποτελείται από ένα απλό μονοπάτι P και ένα τμήμα με διακλαδώσεις Q. Τότε, το σύνολο των συχνών προτύπων που παράγονται από την FP-growth αποτελείται από:*

1. *Το σύνολο των συχνών προτύπων $freq(P)$ που παράγονται από το P.*
2. *Το σύνολο των συχνών προτύπων $freq(Q)$ που παράγονται από το Q.*
3. *Το εξωτερικό γινόμενο $freq(Q) \times freq(P)$.*

Αλγόριθμος 2.3 FP-growth.

Είσοδος : Το FP-tree μίας βάσης δοσοληψιών DB που κατασκευάστηκε σύμφωνα με τον Αλγόριθμο 2.1 και ένα ελάχιστο support ξ.

Έξοδος : Το σύνολο των συχνών προτύπων της βάσης.

Μέθοδος : κλήση της διαδικασίας *FP–growth*(FP-tree, `null`)

procedure `FP_growth`(*Tree,a*)

(1) **αν** το *Tree* περιέχει απλό μονοπάτι P **τότε** {
(2) έστω P το απλό μονοπάτι του *Tree*;
(3) έστω Q το υπόλοιπο δέντρο με τις διακλαδώσεις όπου ο κόμβος στον οποίο γίνεται η διακλάδωση αντικαθίσταται με `null`;
(4) **για κάθε** συνδυασμό (που συμβολίζεται με β) των κόμβων στο μονοπάτι P
(5) παρήγαγε όλα τα πρότυπα $\beta \cup \alpha$ με *support* = *το ελάχιστο support* στο β
(6) έστω $freq(P)$ το σύνολο των προτύπων που παράγονται;}
(7) **αλλιώς**, για κάθε α_i του header table του *Tree* {
(8) παρήγαγε το πρότυπο $\beta = \alpha_i \cup \alpha$ με $support = \alpha_i.support$;
(9) κατασκεύασε την conditional pattern base του β και μετά το conditional pattern tree του β, $Tree_\beta$;
(10) **αν** $Tree_\beta \neq \emptyset$ **τότε**
(11) κάλεσε την `FP_growth`$(Tree_\beta, \beta)$;}
(12) έστω $freq(Q)$ το σύνολο των προτύπων που παράγονται;
(13) **επέστρεψε** το $freq(P) \cup freq(Q) \cup (freq(P) \times freq(Q))$

2.3.3 Σύνοψη

Η μέθοδος εύρεσης συχνών προτύπων FP-growth είναι μία εναλλακτική *divide-and-conquer* μέθοδος σε απάντηση των Apriori αλγορίθμων, που προτάθηκε πρώτα από Han et al. [8] και παρουσιάζει τα εξής πλεονεκτήματα.

- Συμπιέζει τη βάση των δοσοληψιών σε μία συμπαγή δομή δέντρου FP-tree η οποία:
  - Αποθηκεύει όλες τις απαραίτητες πληροφορίες για την εξόρυξη των συχνών προτύπων, αφού οργανώνει κατάλληλα μόνο τα συχνά αντικείμενα της βάσης. Αυτά, σύμφωνα με την Apriori ευριστική, είναι τα μόνα που παίζουν ρόλο στην εξόρυξη.
  - Χρησιμοποιώντας την ταξινόμηση των συχνών αντικειμένων με φθίνουσα σειρά συ-

χνότητας έχει τη δυνατότητα να συγχωνεύσει τις δοσοληψίες που μοιράζονται κοινό πρόθεμα, μειώνοντας έτσι τον αριθμό των κόμβων του δέντρου και άρα το μέγεθός του.

- Διατηρεί αναλοίωτους όλους τους κανόνες συσχέτισης που υπάρχουν και στην αρχική βάση, επομένως διασφαλίζεται η ορθότητα των αποτελεσμάτων του mining.
- Το μέγεθος του δέντρου φράσσεται άνω από το μέγεθος της βάσης, καθώς κάθε δοσοληψία μπορεί να συνεισφέρει το πολύ έναν κλάδο στο δέντρο. Επίσης είναι μικρότερο και από το σύνολο των υποψηφίων συχνών προτύπων που παράγονται από τους υπόλοιπους αλγορίθμους.

□ Δεν παράγει υποψήφια σύνολα (candidate sets) όπως οι Apriori αλγόριθμοι.

□ Αποφεύγει τις επαναληπτικές σαρώσεις της αρχικής βάσης καθώς την επισκέπτεται μόνο δύο φορές κατά την κατασκευή του FP-tree και η διαδικασία του mining γίνεται μόνο στο δέντρο, το μέγεθος του οποίου είναι σημαντικά μικρότερο από αυτό της αρχικής βάσης.

□ Έχει ως βασική λειτουργία την απαρίθμηση και όχι χρονοβόρες μεθόδους τύπου pattern matching.

Έχουν προταθεί διάφορες επεκτάσεις για την βελτίωση της επίδοσης της μεθόδου όπως για παράδειγμα ο τεμαχισμός της αρχικής βάσης σε ένα σύνολο από projected databases και η εκτέλεση του mining χωριστά σε κάθε μία απ' αυτές [8]. Η μέθοδος αυτή είναι χρήσιμη σε περίπτωση που η βάση είναι πάρα πολύ μεγάλη και το support threshold πολύ μικρό, οπότε και το FP-tree δεν θα χωράει στην κύρια μνήμη. Επίσης, έχουν προταθεί παραλλαγές της FP-growth για την εξόρυξη μέγιστων [15] και κλειστών προτύπων που φαίνεται να περιέχουν όλη την απαραίτητη πληροφορία για την εξόρυξη κανόνων συσχέτισης καθώς και άλλες παραλλαγές του αλγορίθμου(Top-Down FP-growth [14]). Αυτές όμως οι προεκτάσεις ξεφεύγουν από το αντικείμενο της εργασίας και γι' αυτό δε θα δοθεί περαιτέρω έκταση.

ΚΕΦΑΛΑΙΟ 3

Αλγόριθμος merge δύο FP-trees

Ας υποθέσουμε ότι ζητείται η εφαρμογή της FP-growth για την εξόρυξη του συνόλου των συχνών προτύπων από μία βάση που περιλαμβάνει τις δοσοληψίες διαφορετικών υποκαταστημάτων ενός supermarket. Ο ευθύς τρόπος θα ήταν η ενοποίηση των βάσεων των υποκαταστημάτων και η εκτέλεση του mining στην ενοποιημένη βάση. Πολλές φορές, όμως, δε μας ενδιαφέρουν μόνο τα συχνά πρότυπα των συνολικών πωλήσεων αλλά και αυτά που προκύπτουν από τις πωλήσεις κάθε υποκαταστήματος ξεχωριστά. Είναι μάλιστα πιθανό, το mining να έχει ήδη εκτελεστεί στα υποκαταστήματα και να έχουν κατασκευαστεί τα αντίστοιχα FP-trees. Πώς θα ήταν δυνατό, λοιπόν, να ενώσουμε ανά δύο, τέτοιες δομές δέντρου που αντιστοιχούν σε ξεχωριστές βάσεις με δοσοληψίες πάνω στο ίδιο σύνολο αντικειμένων ώστε το τελικό δέντρο να αντιστοιχεί στην ένωση των βάσεων; Το παρόν κεφάλαιο εξετάζει αυτό το ερώτημα και προτείνει έναν αλγόριθμο που επιλύει το πρόβλημα.

3.1 Εισαγωγή

Έστω δύο βάσεις DB_1 και DB_2 με N_1 και N_2 δοσοληψίες πάνω στο ίδιο πεπερασμένο σύνολο αντικειμένων $\Sigma = \{i_1, i_2, i_3, \ldots, i_n\}$. Στα πλαίσια της εξόρυξης των συχνών προτύπων τους, κατασκευάζονται τα αντίστοιχα FP-trees, fpt_1 για ελάχιστο support ξ_1 και fpt_2 για ελάχιστο support ξ_2. Εξ' ορισμού τα δέντρα περιλαμβάνουν μόνο τα αντικείμενα που η συχνότητά τους είναι τουλάχιστον ίση με τα αντίστοιχα support, δηλαδή όσα αντικείμενα $i_j \in F_i \subseteq \Sigma$, όπου F_i η ταξινομημένη λίστα, με φθίνουσα σειρά συχνότητας, των συχνών προτύπων μήκους -1 κάθε βάσης $\mathrm{DB_i}$. Έχοντας ως κριτήριο τη συχνότητά τους σε κάθε μία από τις βάσεις DB_1 και DB_2, τα αντικείμενα της βάσης $DB = DB_1 \cup DB_2$ μπορούν να χωριστούν σε τρεις κατηγορίες:

1. Τα αντικείμενα που είναι συχνά τόσο στη βάση DB_1 όσο και στη βάση DB_2.

2. Τα αντικείμενα που δεν είναι συχνά ούτε στη βάση DB_1 ούτε στη βάση DB_2.

3. Τα αντικείμενα που είναι συχνά στη βάση DB_1 αλλά δεν είναι στη βάση DB_2 (ή το αντίστροφο).

Το ερώτημα που τίθεται είναι *πώς μπορούμε να τα κατατάξουμε ως προς τη συχνότητά τους στη βάση* $DB = DB_1 \cup DB_2$ *με δεδομένες μόνο τις λίστες* F_1 *και* F_2. Θεωρούμε, λοιπόν, τα ακόλουθα.

Λήμμα 3.1. *Έστω δύο βάσεις* DB_1 *και* DB_2 *με δοσοληψίες πάνω στο ίδιο σύνολο* Σ, ξ_1, ξ_2 *τα αντίστοιχα support και ένα αντικείμενο* $i_j \in \Sigma$. *Αν* $i_j \in F_1$ *και* $i_j \in F_2$ *τότε* $i_j \in F$, *δηλαδή* $F \supseteq F_1 \cap F_2$, *όπου* F_1, F_2 *οι λίστες των συχνών αντικειμένων δύο βάσεων και* F *η λίστα συχνών αντικειμένων της βάσης* $DB = DB_1 \cup DB_2$.

Λήμμα 3.2. *Έστω δύο βάσεις* DB_1 *και* DB_2 *με δοσοληψίες πάνω στο ίδιο σύνολο* Σ, ξ_1, ξ_2 *τα αντίστοιχα support και ένα αντικείμενο* $i_j \in \Sigma$. *Αν* $i_j \notin F_1$ *και* $i_j \notin F_2$ *τότε* $i_j \notin F$, *δηλαδή* $\mathrm{co}{-}F \supseteq \mathrm{co}{-}F_1 \cap \mathrm{co}{-}F_2$, *όπου* F_1, F_2 *οι λίστες των συχνών αντικειμένων δύο βάσεων και* F *η λίστα συχνών αντικειμένων της βάσης* $DB = DB_1 \cup DB_2$.

Με βάση τα Λήμματα 3.1 και 3.2, προκύπτει το παρακάτω θεώρημα για το support της ένωσης δύο βάσεων.

Θεώρημα 3.1. *Έστω δύο βάσεις* DB_1 *και* DB_2 *με δοσοληψίες πάνω στο ίδιο σύνολο* Σ, ξ_1, ξ_2 *τα αντίστοιχα support και ένα αντικείμενο* $i_j \in \Sigma$. *Για να είναι το αντικείμενο αυτό συχνό στη βάση* $\mathrm{DB} = \mathrm{DB}_1 \cup \mathrm{DB}_2$ *θα πρέπει το πλήθος των εμφανίσεών του να είναι τουλάχιστον ίσο με* $\xi = \xi_1 + \xi_2$.

Απόδειξη. Διακρίνουμε τις εξής περιπτώσεις:

- Έστω ότι το support της βάσης DB είναι $\xi > \xi_1 + \xi_2$. Τότε υπάρχει περίπτωση ένα αντικείμενο i_j που είναι συχνό στη DB_1 με support $s_{i_j1} = \xi_1$ και συχνό στη DB_2 με $s_{i_j2} = \xi_2$ το οποίο να μην είναι συχνό στη DB καθώς οι συνολικές εμφανίσεις του θα είναι $s_{i_j} = s_{i_j1} + s_{i_j2} = \xi_1 + \xi_2 < \xi$. Άτοπο.

- Έστω ότι το support της βάσης DB είναι $\xi < \xi_1 + \xi_2$. Τότε υπάρχει περίπτωση ένα αντικείμενο i_j να μην είναι συχνό ούτε στη DB_1 ούτε στη DB_2 αλλά το άθροισμα των εμφανίσεών του στις δύο βάσεις να είναι $s_{i_j} \geq \xi$ και επομένως να είναι συχνό στη DB. Άτοπο.

Επομένως, προκύπτει ότι το support της βάσης $\mathrm{DB} = \mathrm{DB}_1 \cup \mathrm{DB}_2$ είναι $\xi = \xi_1 + \xi_2$. □

Επειδή πολλές φορές το support εκφράζεται και σαν ποσοστό επί τις εκατό, ακολουθεί το αντίστοιχο συμπέρασμα για το s[%] της ένωσης δύο βάσεων.

Πόρισμα 3.1. *Έστω δύο βάσεις* DB_1 *και* DB_2 *με δοσοληψίες πάνω στο ίδιο σύνολο* Σ *και ένα αντικείμενο* $i_j \in \Sigma$*. Για να είναι το αντικείμενο αυτό συχνό στη βάση* $\mathrm{DB} = \mathrm{DB}_1 \cup \mathrm{DB}_2$ *θα πρέπει το επί τις εκατό support του* $s_{i_j}[\%]$ *να είναι*

$$s_{i_j}[\%] \geq s = s_1 \frac{N_1}{N_1 + N_2} + s_2 \frac{N_2}{N_1 + N_2}$$

όπου s_1*,* s_2*,* s *το ελάχιστο επί τις εκατό support που αντιστοιχεί στις βάσεις* DB_1*,* DB_2 *και* DB*.*

Απόδειξη. Τα ελάχιστα επί τις εκατό support των βάσεων DB_1, DB_2 και DB είναι $s_1 = \frac{\xi_1}{N_1}$, $s_2 = \frac{\xi_2}{N_2}$ και $s = \frac{\xi}{N_1+N_2}$ αντίστοιχα. Από το Θεώρημα 3.1 όμως, ισχύει ότι: $\xi = \xi_1 + \xi_2 \Rightarrow s(N_1 + N_2) = s_1 \cdot N_1 + s_2 \cdot N_2 \Rightarrow s = s_1 \frac{N_1}{N_1+N_2} + s_2 \frac{N_2}{N_1+N_2}$. □

Από το Θεώρημα 3.1 προκύπτει το παρακάτω πόρισμα για τα αντικείμενα που δεν είναι συχνά σε μία από τις δύο βάσεις (κατηγορία 3).

Πόρισμα 3.2. *Έστω* F_1*,* F_2 *οι λίστες των συχνών αντικειμένων δύο βάσεων* DB_1 *και* DB_2 *με δοσοληψίες πάνω στο ίδιο σύνολο* Σ *και για ελάχιστο απόλυτο support* ξ_1 *και* ξ_2 *αντίστοιχα. Έστω τώρα ένα αντικείμενο* $i_j \in \Sigma$ *για το οποίο ισχύει* $i_j \in F_1$ *αλλά* $i_j \notin F_2$ *(ή αντιστρόφως). Το αντικείμενο αυτό είναι σίγουρα συχνό στη βάση* $\mathrm{DB} = \mathrm{DB}_1 \cup \mathrm{DB}_2$ *μόνο όταν* $s_{i_j1} \geq \xi$ *(ή* $s_{i_j2} \geq \xi$*), όπου* $\xi = \xi_1 + \xi_2$ *το απόλυτο support της βάσης DB.*

Απόδειξη. Για να είναι ένα αντικείμενο i_j συχνό στη DB θα πρέπει $s_{i_j} \geq \xi$. Αλλά, $s_{i_j} = s_{i_j1} + s_{i_j2}$. Επειδή $i_j \in F_1$ (ή $i_j \in F_2$) γνωρίζουμε το s_{i_j1} (ή s_{i_j2}) αλλά δε γνωρίζουμε το s_{i_j2} (ή s_{i_j1}) παρά μόνο αν επισκεφτούμε ξανά την αντίστοιχη βάση και συλλέξουμε το πλήθος των εμφανίσεών του. Επομένως, η μόνη περίπτωση για την οποία είμαστε σίγουρη ότι το i_j θα είναι συχνό στη DB είναι αν το $s_{i_j1} \geq \xi$ (ή $s_{i_j2} \geq \xi$) οπότε και $s_{i_j} = s_{i_j1} + s_{i_j2} \geq \xi$ ανεξάρτητα από το πλήθος των εμφανίσεων s_{i_j2} (ή s_{i_j1}). □

Με βάση τα παραπάνω, ακολουθεί ο ορισμός της πράξης merge μεταξύ δύο FP-trees.

Ορισμός 3.1 (merged FP-tree)**.** *Έστω δύο βάσεις* DB_1 *και* DB_2 *με* N_1 *και* N_2 *δοσοληψίες πάνω στο ίδιο σύνολο αντικειμένων* Σ *και* fpt_1*,* fpt_2 *τα FP-trees που κατασκευάζονται για minimum support* ξ_1 *και* ξ_2 *αντίστοιχα. Ορίζουμε ως* $\mathrm{fpt_m} = \mathrm{fpt}_1$ `merge` fpt_2 *το FP-tree που θα κατασκευαζόταν για τη βάση* $\mathrm{DB} = \mathrm{DB}_1 \cup \mathrm{DB}_2$ *με minimum support* $\xi = \xi_1 + \xi_2$*.*

3.2 Κατασκευή merged FP-tree

Ουσιαστικά, η πράξη merge μεταξύ δύο FP-trees αντιστοιχεί στην ενοποίηση των δέντρων με συγχώνευση των μονοπατιών που περιέχουν τα ίδια αντικείμενα. Για να είναι, όμως, το προκύπτον δέντρο συνεπές με τη βάση $\mathrm{DB} = \mathrm{DB}_1 \cup \mathrm{DB}_2$ και με το αντίστοιχο support, θα πρέπει τα αντικείμενα στα FP-trees fpt_1 και fpt_2 των βάσεων DB_1 και DB_2, να βρίσκονται στην ίδια σειρά και να είναι ακριβώς αυτά που θα υπήρχαν στο δέντρο αν κατασκευαζόταν εξ' αρχής για την ένωση των βάσεων. Η λογική του αλγορίθμου που προτείνεται είναι, χρησιμοποιώντας όλες τις πληροφορίες των fpt_1 και fpt_2, να φέρουμε τα δέντρα σε μορφή συνεπή με τη βάση DB και η πράξη του merge να εκφυλιστεί σε απλή ένωση των δέντρων.
Με βάση τα παραπάνω, ο αλγόριθμος που προτείνεται είναι ο ακόλουθος.

Αλγόριθμος 3.1 `Merge` FP-trees.

Είσοδος: Δύο βάσεις DB_1, DB_2 με δοσοληψίες πάνω στο ίδιο σύνολο αντικειμένων Σ και τα αντίστοιχα FP-trees fpt_1 και fpt_2 για support $\xi_1,\ \xi_2$.

Έξοδος: Το FP-tree της βάσης $\mathrm{DB} = \mathrm{DB}_1 \cup \mathrm{DB}_2$ για support $\xi = \xi_1 + \xi_2$.

Μέθοδος:

1. Με βάση τις σειρές συχνότητας F_1 και F_2 που αντιστοιχούν στα δέντρα fpt_1 και fpt_2, βρες τη σειρά συχνότητας F.
2. Μετασχημάτισε, όπου χρειάζεται, τα δέντρα fpt_1 και fpt_2 ώστε να αποκτήσουν το ίδιο σχήμα ακολουθώντας τη νέα σειρά συχνότητας F.
3. Ένωσε τα δέντρα.

Ορισμός 3.2. *Θα λέμε ότι δύο FP-trees έχουν το ίδιο σχήμα αν οι λίστες συχνών αντικειμένων σύμφωνα με τις οποίες έγινε η κατασκευή τους περιέχουν ακριβώς τα ίδια αντικείμενα και έχουν ακριβώς την ίδια σειρά ταξινόμησης.*

3.2.1 Καθορισμός της *f-list* του merged FP-tree

Σύμφωνα με τον ορισμό 3.1, το merged FP-tree πρέπει να περιλαμβάνει όλες τις δοσοληψίες της βάσης $\mathrm{DB} = \mathrm{DB}_1 \cup \mathrm{DB}_2$ ταξινομημένες σύμφωνα με τη σειρά συχνότητας F. Τα FP-trees fpt_1 και fpt_2 που αντιστοιχούν στις βάσεις DB_1 και DB_2 για ελάχιστο support ξ_1 και ξ_2, περιέχουν αθροιστικά το σύνολο των δοσοληψιών, όμως αυτές είναι ταξινομημένες σύμφωνα με τις λίστες

F_1 και F_2. Επομένως, αρχικά θα πρέπει να καθοριστεί η λίστα συχνότητας F χρησιμοποιώντας τις πληροφορίες που διαθέτουν οι λίστες F_1 και F_2. Διακρίνουμε τις εξής περιπτώσεις:

1. **$\mathbf{F_1}$, $\mathbf{F_2}$ περιέχουν τα ίδια αντικείμενα και η σειρά ταξινόμησης τους είναι η ίδια**. Τότε η F θα περιλαμβάνει ακριβώς αυτά τα αντικείμενα ταξινομημένα με την ίδια σειρά.

2. **$\mathbf{F_1}$, $\mathbf{F_2}$ περιέχουν τα ίδια αντικείμενα αλλά η σειρά ταξινόμησης δεν είναι η ίδια**. Τότε η F θα περιλαμβάνει ακριβώς αυτά τα αντικείμενα αλλά η νέα σειρά θα προκύψει από την πρόσθεση των supports των ίδιων αντικειμένων και την εκ νέου ταξινόμηση τους. Σε αυτή την περίπτωση απαιτείται έλεγχος για πιθανό μετασχηματισμό των δέντρων ώστε να αντικατοπτρίζεται η νέα σειρά συχνότητας.

3. **Υπάρχει τουλάχιστον ένα $\mathbf{i_j \in F_1}$ με $\mathbf{i_j \notin F_2}$ (ή και το αντίστροφο)**. Τότε η F, όπως περιγράφηκε στην παράγραφο 3.1, δε μπορεί να καθοριστεί απ' ευθείας. Σε αυτή την περίπτωση απαιτείται:

   - *Ενημέρωση* των fpt_1 και fpt_2 από τις βάσεις DB_1 και DB_2 με προσθήκη των αντικειμένων που λείπουν απ' το κάθε δέντρο ώστε τελικά τα δέντρα να περιέχουν όλα τα αντικείμενα $F_1 \cup F_2$. Η ενημέρωση αυτή είναι απαραίτητη για να έχουμε όλη την πληροφορία που απαιτεί ο καθορισμός της νέας σειράς συχνότητας.
   - Πρόσθεση των supports των αντικειμένων ώστε να προκύψει η νέα λίστα συχνών αντικειμένων F της βάσης DB.

   Και εδώ είναι απαραίτητος ο έλεγχος για πιθανό μετασχηματισμό των δέντρων, όπου αυτός κρίνεται απαραίτητος.

Παράδειγμα 3.1.

Ας θεωρήσουμε δύο βάσεις δοσοληψιών DB_1, DB_2 με support $\xi_1 = 3$ και $\xi_2 = 2$ αντίστοιχα. Τα αντικείμενα της κάθε βάσης και οι ταξινομημένες λίστες συχνών αντικειμένων τους φαίνονται στον Πίνακα 3.1.

Το support της $\text{DB} = \text{DB}_1 \cup \text{DB}_2$ θα είναι, σύμφωνα με το Θεώρημα 3.1, $\xi = 5$. Από το περιεχόμενο των F_1 και F_2, προκύπτει ότι η περίπτωση αυτή ανήκει στην τρίτη κατηγορία που περιγράφηκε παραπάνω. Γι' αυτό, επιστρέφουμε στις δύο βάσεις και προσθέτουμε τα αντικείμενα που δεν υπάρχουν στη μία λίστα αλλά υπάρχουν στην άλλη, δηλαδή το d στη λίστα F_1 και το g στην F_2. Μετά την ενημέρωση η λίστες γίνονται:

TID	Αντικείμενα
100	m, n, o, a, b, c
200	c, f, g, o, m
300	k, l, b, h, f, g
400	m, n, f, a, c, d
500	c, m, k, g, b

$\mathrm{F}_1 = \{(\mathrm{c} : 4), (\mathrm{m} : 4), (\mathrm{b} : 3), (\mathrm{f} : 3), (\mathrm{g} : 3)\}$

(α) Βάση δοσοληψιών DB_1, $\xi_1 = 3$

TID	Αντικείμενα
600	c, m, d, b
700	b, c, n, k, d
800	l, b, a, h, f
900	o, d, f, b, g

$\mathrm{F}_2 = \{(\mathrm{b} : 4), (\mathrm{d} : 3), (\mathrm{c} : 2), (\mathrm{f} : 2)\}$

(β) Βάση δοσοληψιών DB_2, $\xi_2 = 2$

Πίνακας 3.1: Βάσεις δοσοληψιών και αντίστοιχες λίστες συχνών αντικειμένων

$$\mathrm{F}_1 = \{(\mathrm{c} : 4), (\mathrm{m} : 4), (\mathrm{b} : 3), (\mathrm{f} : 3), (\mathrm{g} : 3), (\mathrm{d} : 1)\}$$
$$\mathrm{F}_2 = \{(\mathrm{b} : 4), (\mathrm{d} : 3), (\mathrm{c} : 2), (\mathrm{f} : 2), (\mathrm{g} : 1)\}$$

Στη συνέχεια προσθέτουμε τα support των αντίστοιχων αντικειμένων, απομακρύνουμε όσα έχουν πλήθος εμφανίσεων μικρότερο από ξ, ταξινομούμε τα αντικείμενα και προκύπτει η λίστα συχνών αντικειμένων της DB: $F = \{(b : 7), (c : 6), (f : 5), (m : 5)\}$

Για λόγους ομοιογένειας θεωρούμε ότι όταν δύο αντικείμενα έχουν το ίδιο support, πρώτο στη λίστα θα μπαίνει αυτό που έχει υψηλότερη θέση στην αύξουσα αλφαριθμητική σειρά. Γι' αυτό και το f στη λίστα έχει τοποθετηθεί πριν το m. Αν και η διαφορά δεν είναι θεωρητικά ουσιαστική, παίζει σημαντικό ρόλο στην επιλογή των μετασχηματισμών που θα γίνουν στα δέντρα.

□

3.2.2 Μετασχηματισμός FP-tree

Η σειρά συχνότητας του merged FP-tree που προκύπτει για κάθε μία από τις περιπτώσεις της προηγούμενης παραγράφου, καθορίζει και την ανάγκη ή όχι μετασχηματισμού των fpt_1, fpt_2. Το κριτήριο είναι η σχέση μεταξύ της F και των F_1, F_2. Αν οι τελευταίες έχουν ένα τουλάχιστον αντικείμενο σε διαφορετική θέση σε σχέση με τη θέση του στη σειρά F τότε απαιτείται μετασχηματισμός τέτοιος ώστε να αντικατοπτρίζεται η νέα σειρά συχνότητας.

Ο αλγόριθμος μετασχηματισμού του FP-tree που χρησιμοποιούμε, σε μία παραλλαγή του, λέγεται AFPIM (Adjusting FP-tree for Incremental Mining) [11] και αναφέρεται στην περίπτωση *ενημέρωσης* ενός FP-tree με την προσθήκη/αφαίρεση μίας ή περισσοτέρων δοσοληψιών. Ουσιαστικά, και το πρόβλημα του merge δύο FP-trees μπορεί να θεωρηθεί ως ενημέρωση του ενός δέντρου από το άλλο, καθώς κάθε δέντρο αντιπροσωπεύει δοσοληψίες.

Αλγόριθμος 3.2 Παραλλαγή του AFPIM.

Είσοδος: FP-tree T με σειρά συχνότητας F, νέα σειρά συχνότητας F'

Έξοδος: FP-tree T' με σειρά συχνότητας F'

Μέθοδος:

(1) **για κάθε** αντικείμενο που υπάρχει στην F αλλά δεν υπάρχει στην F' {
(2) αφαίρεσε τους αντίστοιχους κόμβους από το T;
(3) αποκατέστησε τη δομή του δέντρου;
(4) αφαίρεσε τις αντίστοιχες εγγραφές από τον header table;}
(5) εφαρμόζοντας τη bubblesort στην F με βάση την F'
για κάθε ζεύγος αντικειμένων που ανταλλάσουν θέση στην F
(6) **για κάθε** ζεύγος κόμβων (X,Y) του T που περιέχουν τα αντικείμενα αυτά
(7) κάλεσε τη μέθοδο *path_adjust*(T,X,Y,P) όπου P ο πατέρας του κόμβου X;

Ανάλυση. Ο αλγόριθμος ξεκινά αφαιρώντας από το αρχικό δέντρο όσους κόμβους περιέχουν αντικείμενα που δεν ανήκουν στη νέα σειρά συχνότητας F', αν φυσικά υπάρχουν τέτοια. Τα αντικείμενα αυτά δε θεωρούνται πλέον συχνά και επομένως δεν πρέπει να βρίσκονται στο FP-tree ενώ ταυτόχρονα αφαιρούνται οι αντίστοιχες καταχώρησεις στον headertable και τα nodelinks. Ένας κόμβος που τυχόν αφαιρείται, μπορεί να βρίσκεται οπουδήποτε στο δέντρο. Γι' αυτό θα πρέπει να αποκατασταθεί η δομή του δέντρου, αναθέτοντας τα παιδιά του αφαιρούμενου κόμβου στον πατέρα του.

Στη συνέχεια εφαρμόζουμε τη bubblesort ταξινομώντας την αρχική λίστα συχνοτήτων με βάση τη νέα λίστα, ώστε να επιλεγούν τα αντικείμενα που πρέπει να ανταλλάξουν θέσεις στο δέντρο. Ο λόγος για τον οποίο επιλέγεται η $O(n^2)$ bubblesort [5] ως αλγόριθμος ταξινόμησης και όχι κάποιος πιο γρήγορος αλγόριθμος, όπως για παραάδειγμα η $O(n \log n)$ mergesort, είναι γιατί πρέπει τα αντικείμενα που επιλέγονται για ανταλλαγή να έχουν κόμβους στο δέντρο με σχέση πατέρα-παιδιού, και άρα πρέπει να ελεγχθούν όλα τα *γειτονικά* αντικείμενα της σειράς συχνότητας. Κι αυτό, γιατί η διαδικασία ανταλλαγής σε περίπτωση που οι κόμβοι βρίσκονταν σε απόσταση μεγαλύτερη του ενός επιπέδου στο δέντρο, θα γινόταν ιδιαίτερα πολύπλοκη.

Για κάθε ζεύγος αντικειμένων που ανταλλάσσεται στη σειρά συχνότητας F, ελέγχονται οι κόμβοι στο δέντρο που τα περιέχουν. Αυτό γίνεται πρακτικά, με διάσχιση του δέντρου FP-tree ξεκινώντας από τον headertable και ακολουθώντας τα nodelinks των αντικειμένων. Οι κόμβοι που επιλέγονται προς ανταλλαγή έχουν τη σχέση πατέρα-παιδιού στο δέντρο και η ανταλλαγή τους γίνεται σύμφωνα μέθοδο προσαρμογής μονοπατιού [11]. Έστω, λοιπόν, X, Y οι κόμβοι που

Αλγόριθμος 3.3 procedure path_adjust.

procedure path_adjust(T,X,Y,P))

(1) **αν** $X.count > Y.count$ {

(2) πρόσθεσε έναν νέο κόμβο στο T, έστω X', με πατέρα τον P;

(3) θέσε το count του νέου κόμβου $X'.count = X.count - Y.count$;

(4) κρέμασε από τον X' όλα τα παιδιά του κόμβου X εκτός από τον Y;

(5) θέσε το count του κόμβου $X.count = Y.count$;}

(6) αντάλλαξε τις θέσεις των κόμβων X και Y στο T;

(7) **αν** υπάρχει παιδί του P, έστω Z, που να έχει το ίδιο αντικείμενο με τον Y {

(8) θέσε $Y.count = Y.count + Z.count$;

(9) κρέμασε τα παιδιά του Z, αν υπάρχουν, απ' τον Y;

(10) αφαίρεσε τον κόμβο Z απ' το T;}

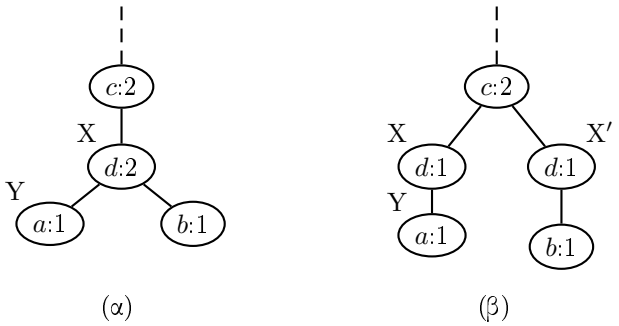
πρέπει να ανταλλάξουν θέσεις στο δέντρο και X πατέρας του Y.

Διακρίνουμε δύο περιπτώσεις:

1. Ο X έχει μεγαλύτερο count από τον Y, που σημαίνει ότι ο X έχει και άλλα παιδιά.

   Σε αυτή την περίπτωση, πριν την ανταλλαγή των θέσεων, θα πρέπει να γίνει αναδιάταξη των υπόλοιπων παιδιών του X ώστε να διατηρηθεί η πληροφορία των αντίστοιχων δοσοληψιών στο FP-tree. Για το λόγο αυτό δημιουργείται ένας νέος κόμβος X′ που έχει το ίδιο αντικείμενο με τον X, έχει πατέρα τον πατέρα του X και το count του παίρνει την τιμή X′.count = X.count − Y.count, ενώ όλα τα παιδιά του X εκτός από τον κόμβο Y ανατίθενται σ' αυτόν.

   Για παράδειγμα, έστω το τμήμα ενός FP-tree που φαίνεται στο Σχήμα 3.1(α).



Σχήμα 3.1: Παράδειγμα περίπτωσης ανταλλαγής κόμβων

Οι δοσοληψίες που παριστάνονται σ' αυτό είναι η $\langle(\ldots c, d, a) : 1\rangle$ και η $\langle(\ldots c, d, b) : 1\rangle$. Η τελευταία πρέπει να παραμείνει ως έχει και μετά το τέλος της ανταλλαγής των κόμβων

X, Y. Έτσι, προστίθεται ο νέος κόμβος X′ όπως φαίνεται στο Σχήμα 3.1(β).

2. Οι X, Y έχουν το ίδιο count, οπότε ο X δεν έχει παιδιά. Πρέπει να παρατηρήσουμε ότι αυτή τη μορφή έχει και το δέντρο της περίπτωσης 1 μετά το πέρας των μετασχηματισμών που περιγράφηκε παραπάνω.

   Σε αυτή την περίπτωση η διαδικασία ανταλλαγής είναι σχετικά ξεκάθαρη. Ο κόμβος X ανταλλάσει τη θέση του με τον κόμβο Y, δηλαδή, ο πατέρας του X γίνεται πατέρας του Y και ο X γίνεται παιδί του Y.

Για να ολοκληρωθεί η διαδικασία του μετασχηματισμού πρέπει να γίνει ένας τελευταίος έλεγχος. Υπάρχει ενδεχόμενο, ο κόμβος Y, που πλέον έχει πάρει τη σωστή θέση του στο δέντρο, να έχει αδερφό έναν κόμβο, έστω Z, ο οποίος έχει το ίδιο αντικείμενο. Αυτός θα πρέπει να συγχωνευθεί με τον Y, καθώς σε άλλη περίπτωση θα έχουμε δύο προθεματικά δέντρα για το ίδιο αντικείμενο που έχουν τον ίδιο πατέρα, πράγμα που απαγορεύεται από τον ορισμό του FP-tree. Έτσι, το count του Y γίνεται Y.count = Y.count + Z.count και τα παιδιά του Z, αν υπάρχουν, αναθέτονται στο Y.

Η λειτουργία του αλγορίθμου μετασχηματισμού γίνεται πιο ξεκάθαρη μελετώντας το παράδειγμα που ακολουθεί.

Παράδειγμα 3.2 (Μετασχηματισμός FP-tree)**.**

Έστω η βάση δοσοληψιών του Πίνακα 3.1(α). Η εφαρμογή του αλγορίθμου 2.1 κατασκευής FP-tree, που παρουσιάστηκε στο Κεφάλαιο 2, δίνει την ακόλουθη σειρά συχνότητας:

$$\textit{f-list}= \{(c:4),(m:4),(b:3),(f:3),(g:3)\}$$

Η ταξινόμηση των αντικειμένων κάθε δοσοληψίας φαίνεται στον Πίνακα 3.2.

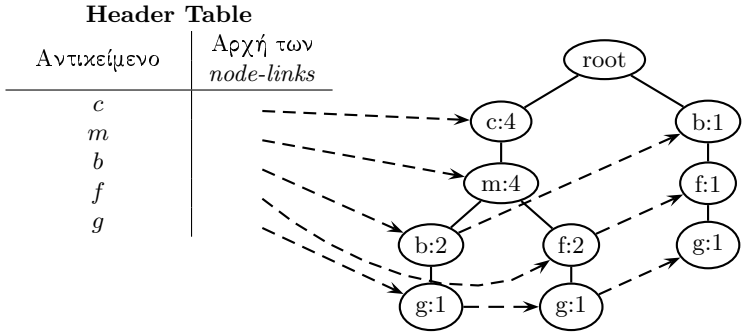
TID	Αντικείμενα	Συχνά αντικείμενα
100	m,n,o,a,b,c	c,m,b
200	c,f,g,o,m	c,m,f,g
300	k,l,b,h,f,g	b,f,g
400	m,n,f,a,c,d	c,m,f
500	c,m,k,g,b	c,m,b,g

Πίνακας 3.2: Αρχική ταξινόμηση συχνών αντικειμένων κάθε δοσοληψίας.

Το αντίστοιχο FP-tree είναι αυτό του Σχήματος 3.2.

Έστω τώρα ότι στα πλαίσια της εκτέλεσης του merge καθορίζεται νέα σειρά συχνότητας

$$\text{νέα } \textit{f-list}= \{b,c,f,m\}$$
$$\text{παλιά } \textit{f-list}= \{c,m,b,f,g\}$$



Σχήμα 3.2: FP-tree παραδείγματος 3.2.

Σύμφωνα με τον Αλγόριθμο 3.2:

1. Συγκρίνοντας τις δύο λίστες, παρατηρούμε ότι το αντικείμενο g δεν υπάρχει στη νέα λίστα και αρα πρέπει να αφαιρεθεί από το δέντρο. Όπως φαίνεται στο Σχήμα 3.2 υπάρχουν τρεις κόμβοι με το αντικείμενο προς αφαίρεση, οι οποίοι τυχαίνει να είναι και φύλλα. Επομένως, η αφαίρεσή τους γίνεται απλά, χωρίς να απαιτείται αποκατάσταση της δομής του δέντρου.

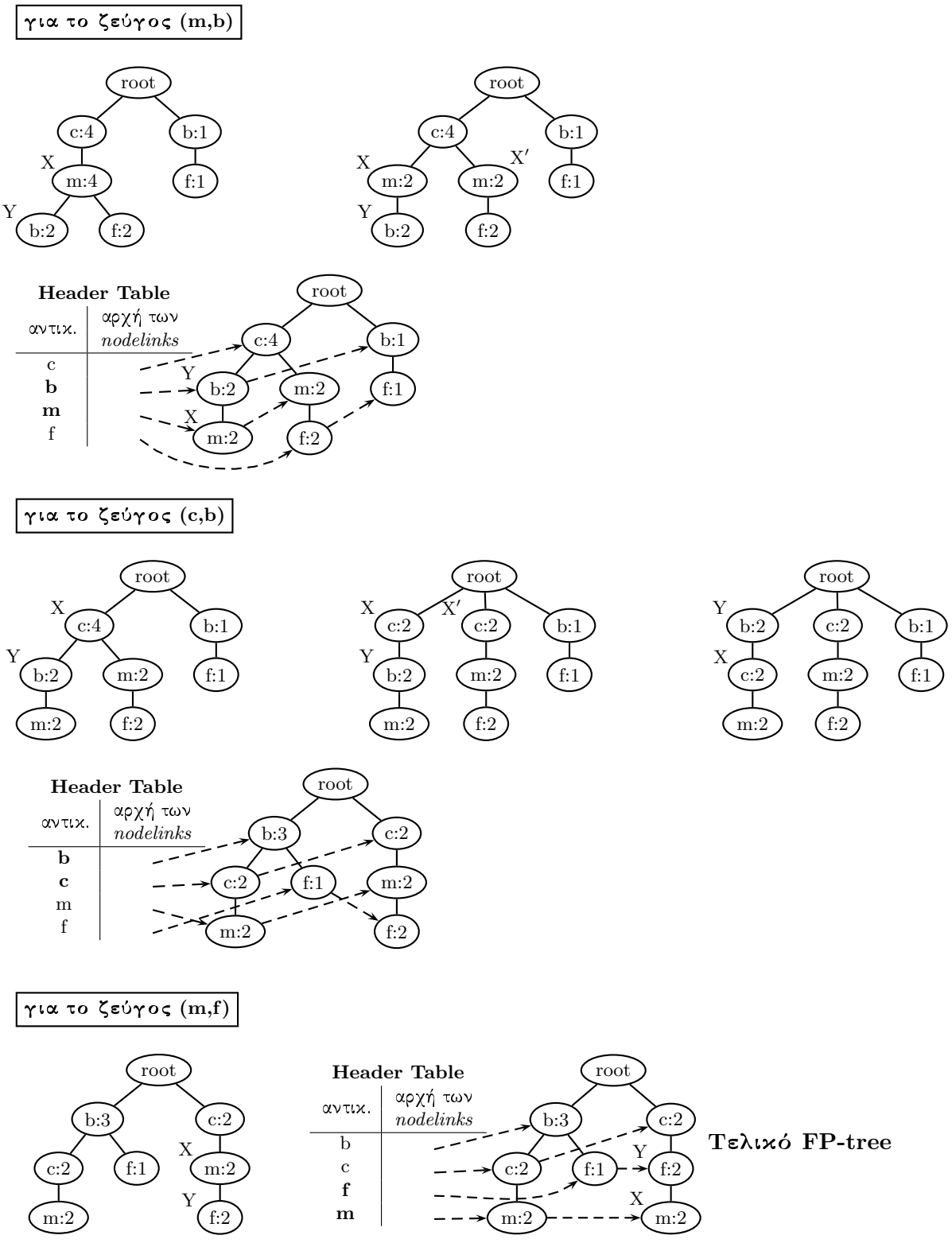
2. Επιλέγουμε τα αντικείμενα που πρέπει να ανταλλάξουν τους κόμβους τους στο δέντρο εφαρμόζοντας την bubblesort στην παλιά λίστα με βάση την νέα (3.3).

νέα *f-list*: **b**	**c**	**f**	**m**	Επιλεγόμενο ζεύγος
c	m	b	f	∅
c	m	b	f	(m,b)
c	b	m	f	(c,b)
b	c	m	f	(m,f)
b	c	f	m	∅
b	c	f	m	∅

Σχήμα 3.3: Αλγόριθμος bubblesort στη λίστα συχνών αντικειμένων

Για κάθε ένα από τα ζεύγη αντικειμένων που επιλέχθηκαν και φαίνονται στο σχήμα 3.3, εφαρμόζεται η μέθοδος `path_adjust`. Η πορεία της διαδικασίας φαίνεται στο σχήμα 3.4.

□



Σχήμα 3.4: Μετασχηματισμοί FP-tree

3.2.3 Ένωση FP-trees

Ορισμός 3.3 (union FP-trees)**.** *Έστω δύο βάσεις* DB_1 *και* DB_2 *με* N_1 *και* N_2 *δοσοληψίες πάνω στο ίδιο σύνολο αντικειμένων* Σ *και* fpt_1, fpt_2 *τα FP-trees που κατασκευάζονται για minimum support* ξ_1 *και* ξ_2 *αντίστοιχα. Ορίζουμε ως* $\mathrm{fpt_u} = \mathrm{fpt}_1$ `union` fpt_2 *το FP-tree που προκύπτει από τη συχνώνευση των μονοπατιών των δύο δέντρων οι οποίοι περιλαμβάνουν τα ίδια αντικείμενα, όπου για κάθε κόμβο που συγχωνεύεται* $count_{i_u} = count_{i_1} + count_{i_2}$.

Λήμμα 3.3. *Έστω* fpt_1 *και* fpt_2 *δύο FP-trees με δοσοληψίες πάνω στο ίδιο σύνολο αντικειμένων. Η πράξη* fpt_1 `union` fpt_2 *ορίζεται αν και μόνον αν τα δέντρα έχουν το ίδιο σχήμα.*

Απόδειξη. ($\Leftarrow$) Αν τα FP-trees έχουν το ίδιο σχήμα, σημαίνει ότι οι σειρές συχνών αντικειμένων τους έχουν την ίδια ταξινόμηση. Επομένως, στην ένωσή τους δε θα υπάρχουν μονοπάτια που να απεικονίζουν δοσοληψίες με τα ίδια συχνά αντικείμενα αλλά διαφορετική διάταξη των κόμβων, πράγμα που απαγορεύεται από τον ορισμό και τον τρόπο κατασκευής του FP-tree.
($\Rightarrow$) Το αποτέλεσμα της πράξης `union` μεταξύ δύο FP-trees σύμφωνα με τον ορισμό 3.3 είναι ένα FP-tree όπου οι κλάδοι που περιλαμβάνουν τα ίδια αντικείμενα έχουν συγχωνευθεί. Επομένως, δεν υπάρχουν μονοπάτια στα δύο δέντρα που να έχουν τα ίδια αντικείμενα στους κόμβους αλλά διαφορετική σειρά και άρα τα δέντρα έχουν το ίδιο σχήμα. □

Αλγόριθμος 3.4 Ένωση FP-trees.

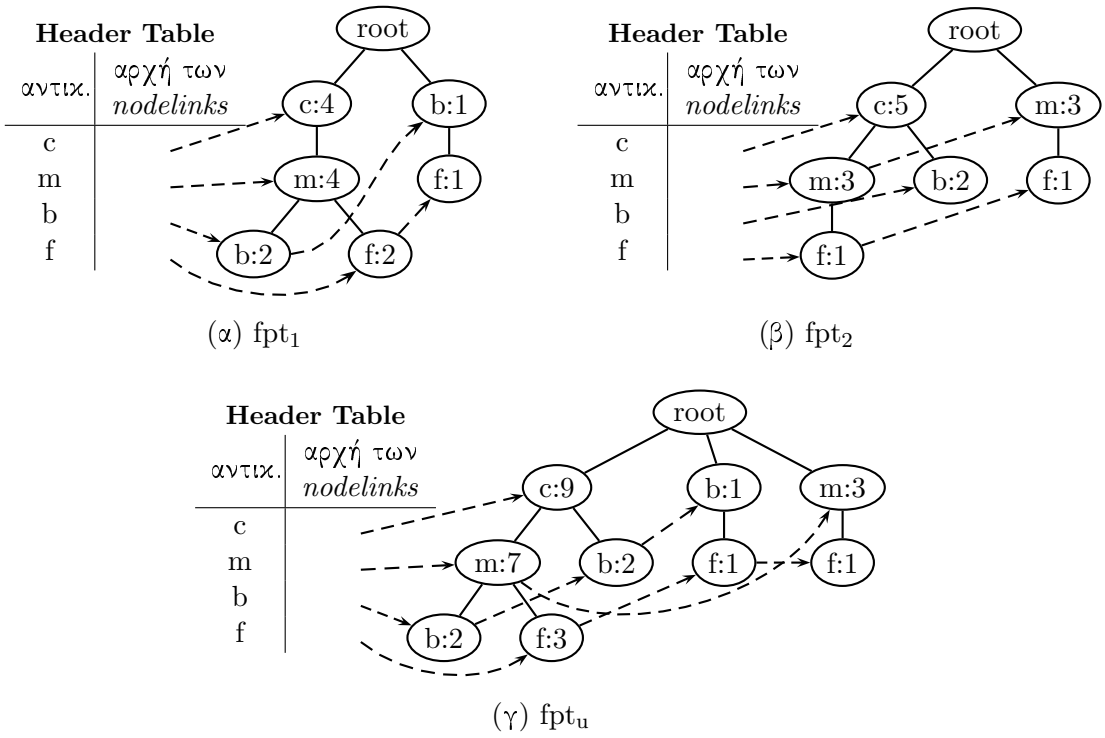
Είσοδος: FP-tree T_1, FP-tree T_2

Έξοδος: FP-tree $T_u = T_1$ `union` T_2

Μέθοδος: κλήση της **procedure** `add_trees`($T_1.root$,$T_2.root$)

procedure `add_trees`(n_1,n_2)

(1) **για κάθε** παιδί c_2 του κόμβου n_2 {
(2) **αν** υπαρχει παιδί c_1 του κόμβου n_1 που έχει το ίδιο αντικείμενο {
(3) θέσε $c_1.count = c_1.count + c_2.count$;
(4) κάλεσε την `add_trees`(T_1, T_2, c_1, c_2); }
(5) **αλλιώς**
(6) κρέμασε το υποδέντρο με ρίζα c_2 από τον κόμβο n_1;



(α) fpt_1 (β) fpt_2

(γ) fpt_u

Σχήμα 3.5: Παράδειγμα ένωσης δύο FP-trees

3.3 Σύνοψη

Ο Αλγόριθμος 3.1 του merge FP-trees εκμεταλλεύεται προηγούμενες εφαρμογές της FP-growth σε βάσεις με δοσοληψίες πάνω στο ίδιο σύνολο αντικειμένων, ενώνοντας τα FP-trees που έχουν κατασκευαστεί με στόχο αποδοτικότερο mining της ένωσης των βάσεων. Τα πλεονεκτήματα του αλγορίθμου είναι τα εξής:

- αποφεύγεται η εξ' αρχής κατασκευή του δέντρου για την ενοποιημένη βάση, η οποία, σύμφωνα με τον αλγόριθμο κατασκευής του FP-tree, προϋποθέτει διπλή επίσκεψη στη βάση. Το πολύ πολύ να χρειαστεί σάρωση των επιμέρους βάσεων για ενημέρωση των FP-trees τους, η οποία αφενός δεν είναι πάντα απαραίτητη, αφετέρου αντιστοιχεί ουσιαστικά σε μία σάρωση της ενοποιημένης βάσης και όχι σε δύο.

- από το Λήμμα 2.1 του Κεφαλαίου 2, το FP-tree έχει, στη γενική περίπτωση, μικρότερο πλήθος ξεχωριστών μονοπατιών από το συνολικό πλήθος των δοσοληψιών στη βάση. Ως εκ τούτου, οι πράξεις ενημέρωσης στα ήδη υπάρχοντα FP-tree είναι κατά βάση λιγότερες από τις πράξεις εισαγωγής δοσοληψιών στο FP-tree που κατασκευάζεται εξ' αρχής για την ενοποιημένη βάση.

ΚΕΦΑΛΑΙΟ 4

Υλοποίηση

Μετά την θεωρητική ανάλυση του `merge` δύο FP-trees, όπως παρουσιάστηκε στο προηγούμενο κεφάλαιο, ακολουθεί η περιγραφή της υλοποίησης που έγινε στα πλαίσια της μελέτης της επίδοσής του. Συγκεκριμένα, υλοποιήθηκε η κατασκευή της δομής FP-tree, η κατασκευή του merged FP-tree και η μέθοδος FP-growth για τον έλεγχο της ορθότητας του προτεινόμενου αλγορίθμου. Η οργάνωση και τα κυριότερα τμήματα του κώδικα που αναπτύχθηκε, παραθέτονται στη συνέχεια, μαζί με επαρκή σχολιασμό.

4.1 Περιγραφή

Η υλοποίηση έγινε σε γλώσσα προγραμματισμού C++, σε πλατφόρμα Windows XP SP2 και περιβάλλον cygwin 1.5.21-1 με τον μεταγλωττιστή gcc-g++ 3.4.4–1 του gnu project.

Οι κύριες λειτουργίες του προγράμματος είναι τρεις:

1. Κατασκευή του δέντρου συχνών προτύπων FP-tree μίας βάσης με δοσοληψίες για προκαθορισμένο support και αποθήκευσή του στο δίσκο.

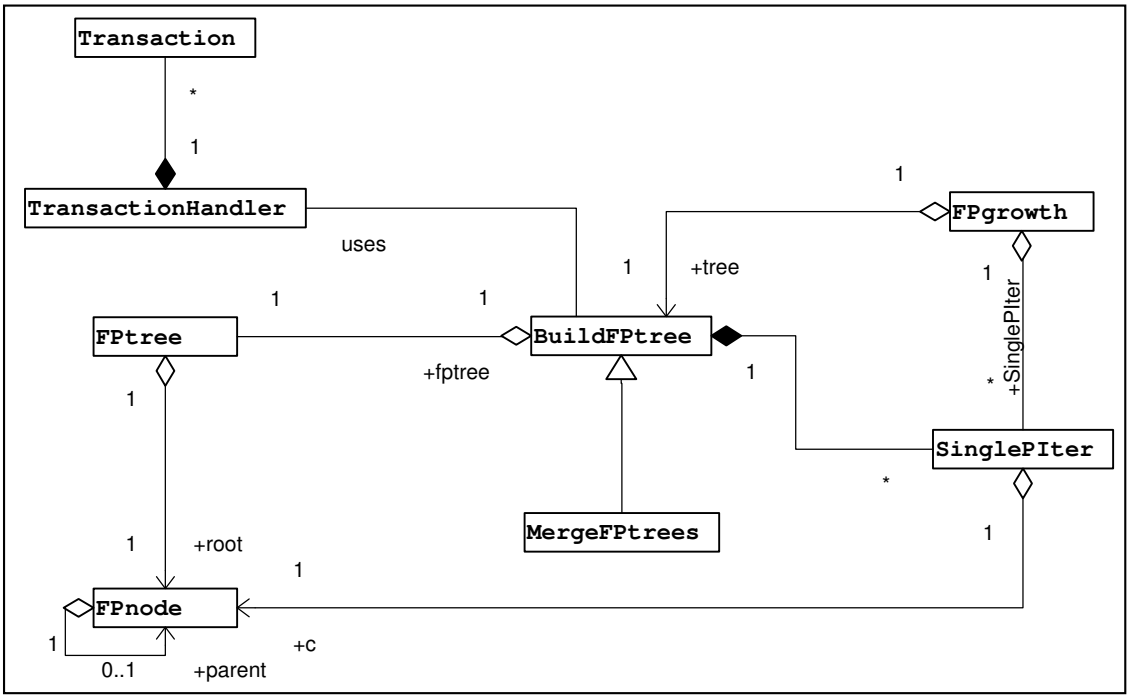
2. Ανάκτηση από το δίσκο δύο αποθηκευμένων FP-trees και κατασκευή του merged FP-tree με τον προτεινόμενο αλγόριθμο.

3. Εφαρμογή της μεθόδου FP-growth σε ένα FP-tree για την εξόρυξη του συνόλου των συχνών προτύπων της αντίστοιχης βάσης.

Η αποθήκευση των FP-trees μετά την κατασκευή τους κρίθηκε σκόπιμο να υλοποιηθεί για την αξιόπιστη διενέργεια των πειραμάτων, καθώς σε διαφορετική περίπτωση η διαδικασία του `merge` θα εφαρμοζόταν σε δέντρα που υπήρχαν ήδη στη μνήμη και άρα θα υπήρχε μεγαλύτερη πιθανότητα να είναι πιο γρήγορη από την εξ' αρχής κατασκευή του FP-tree για την ένωση των

βάσεων. Σε πραγματικές συνθήκες, η περίπτωση αυτή αποτελεί την εξαίρεση, χωρίς βέβαια το ενδεχόμενο να αποκλείεται τελείως.

Μία επισκόπηση των κλάσεων που αναπτύχθηκαν και των σχέσεων μεταξύ τους φαίνεται στο UML διάγραμμα του Σχήματος 4.1.



Σχήμα 4.1: UML διάγραμμα των κλάσεων.

Την κατασκευή και αποθήκευση του FP-tree χειρίζεται η κλάση `BuildFPtree` ενώ η επεξεργασία της βάσης με τις δοσοληψίες γίνεται από την κλάση `TransactionHandler`. Η τελευταία, βρίσκει τη σειρά συχνότητας των αντικειμένων της βάσης και ταξινομεί κάθε δοσοληψία με βάση αυτή. Κάθε ταξινομημένη δοσοληψία εισάγεται με την αντίστοιχη μέθοδο της `BuildFPtree` στο δέντρο. Μετά το τέλος της διαδικασίας κατασκευής, το δέντρο αποθηκεύεται στον σκληρό δίσκο σε δυαδική μορφή. Η `BuildFPtree` υποστηρίζει και άλλες μεθόδους επεξεργασίας του FP-tree, οι οποίες χρησιμοποιούνται από τις υπόλοιπες κλάσεις του προγράμματος και περιγράφονται στη συνέχεια του κεφαλάιου.

Η υλοποίηση του προτεινόμενου αλγορίθμου `merge` δύο δέντρων συχνών προτύπων γίνεται μέσω της κλάσης `MergeFPtrees`. Η κλάση αυτή, ανασύρει από το δίσκο δύο FP-trees που έχουν αποθηκευτεί σε προηγούμενες κλήσεις του προγράμματος και περιλαμβάνει μεθόδους για τον καθορισμό της σειράς συχνότητας του merged FP-tree, τον έλεγχο για πιθανή ανάγκη ενημέρωσης των δέντρων από τις αντίστοιχες βάσεις δοσοληψιών, τον μετασχηματισμό και την ένωση των δέντρων. Το τελικό δέντρο που προκύπτει μπορεί εκ νέου να αποθηκευτεί στο δίσκο, για επαυξητική (incremental) εφαρμογή της μεθόδου.

Η κλάση `FPgrowth` υλοποιεί την αντίστοιχη μέθοδο εξόρυξης συχνών προτύπων. Χρησιμοποιεί το FP-tree που έχει κατασκευαστεί από την `BuildFPtree` ή που έχει προκύψει από την `MergeFPtrees` καθώς και τις μεθόδους της πρώτης για την κατασκευή των conditional FP-tree και εκτελεί το mining σύμφωνα με τον Αλγόριθμο 2.3 που περιγράφηκε στο Κεφάλαιο 2. Τα συχνά πρότυπα που προκύπτουν μαζί με το support τους στη βάση, αποθηκεύονται σε αρχείο στο δίσκο για περαιτέρω επεξεργασία.

4.2 Λεπτομέρειες υλοποίησης

4.2.1 Κατασκευή και αποθήκευση FP-tree

Για την κατασκευή του FP-tree μίας βάσης δοσοληψιών για ένα προκαθορισμένο ελάχιστο support συνεργάζονται δύο κλάσεις: η `TransactionHandler` και η `BuildFPtree`.

Κώδικας 4.1 Ορισμός κλάσης `TransactionHandler`

```
class TransactionHandler {
public:

    class Transaction:public std::set<item_t> {
    public:
        Transaction() : TID(-1) {}

        ~Transaction() {}

        item_t& getTID(){ return TID; }

    private:
        item_t TID;
    };

    typedef Transaction::iterator TIter;

    template<class T>
    count_t findFreqItems(T& freqItems, double& support);

    template<class T>
    count_t nextTransaction(T& transaction);

    template<class T, class V>
    count_t SortTransaction(T& trans, V& freq);

    TransactionHandler (std::istream& in): in(in) {}
    ~TransactionHandler() {}

private:
    bool getNextItem(item_t& item, char& c);

    std::istream& in;
};
```

Η `TransactionHandler` χειρίζεται τις δοσοληψίες της βάσης, η οποία βρίσκεται αποθηκευ-

μένη σε ένα αρχείο. Αρχικά, συλλέγει τις συχνότητες κάθε αντικειμένου διατρέχοντας μία φορά ολόκληρο το αρχείο, βρίσκει ποια από αυτά είναι συχνά με τη μέθοδο `findFreqItems` και με βάση το support που δίνεται από τον χρήστη και τα ταξινομεί με φθίνουσα σειρά συχνότητας. Από τη διαδικασία αυτή προκύπτει η ταξινομημένη λίστα συχνών αντικειμένων της βάσης, με κριτήριο την οποία θα γίνει η εισαγωγή των αντικειμένων κάθε δοσοληψίας στο FP-tree.

Στη συνέχεια επισκέπτεται ξανά τη βάση και διατρέχει το αρχείο, αυτή τη φορά ανά δοσοληψία. Κάθε γραμμή του αρχείου περιλαμβάνει μία δοσοληψία της μορφής $T = [\text{TID}, s]$, όπου TID το μοναδικό αναγνωριστικό της και s το σύνολο των αντικειμένων που περιέχει. Γι' αυτό και κάθε δοσοληψία αντιστοιχίζεται σε ένα αντικείμενο της κλάσης `Transaction` (Κώδικας 4.1) ως επέκταση του C++ STL container set με την προσθήκη του χαρακτηρισικού αναγνωριστικού της TID στα χαρακτηριστικά της κλάσης. Ο λόγος για τον οποίο επιλέχθηκε το set είναι επειδή ταξινομεί αυτόματα τα στοιχεία του με αύξουσα αλφαριθμητική σειρά και έτσι ικανοποιείται το δεύτερο κριτήριο ταξινόμησης των αντικειμένων της δοσοληψίας. Κάθε τέτοια δοσοληψία ταξινομείται εκ νέου με τη μέθοδο `SortTransaction` απ' όπου προκύπτει ένας vector με τα συχνά αντικείμενα της που θα εισαχθούν στο δέντρο, ταξινομημένα με την επιθυμητή σειρά.

Το δέντρο συχνών αντικειμένων υλοποιήθηκε με τη χρήση maps της C++ Standard Template Library [10]. Οι maps είναι ταξινομημένοι associative containers που αποτελούνται από ζεύγη κλειδιού/τιμής και υλοποιούνται εσωτερικά με δυαδικά δέντρα. Η αναζήτηση γίνεται με βάση το κλειδί ενώ δεν επιτρέπεται να υπάρχουν δύο καταχωρήσεις με το ίδιο κλειδί. Η επιλογή αυτή έγινε με στόχο το FP-tree να υποστηρίζει *δυαδική* αναζήτηση και όχι γραμμική, μειώνοντας έτσι τον χρόνο που απαιτείται για ενημέρωση του δέντρου και διαγραφή κόμβων από αυτό.

Κάθε κόμβος του FP-tree αναπαρίσταται από ένα αντικείμενο της κλάσης `FPnode` (Κώδικας 4.2) και αποτελείται από:

- Το όνομα `item` του αντικειμένου του κόμβου.
- Το πλήθος `count` των δοσοληψιών που περιέχουν το μονοπάτι του δέντρου μέχρι τον κόμβο αυτό.
- Έναν δείκτη `parent` στον κόμβο–πατέρα του στο δέντρο.
- Έναν map `children`, τύπου `Children`, που περιλαμβάνει δείκτες σε όλους τους κόμβους – παιδιά του στο δέντρο.

Κώδικας 4.2 Ορισμός κλάσης `FPnode`.

```
typedef std::map<item_t,FPnode*> Children;

class FPnode {
public:
  item_t item;
  count_t count;
  FPnode* parent;
  Children children;
};
```

Το FP-tree υλοποιείται μέσω της κλάσης `FPtree` (Κώδικας 4.3) και περιλαμβάνει:

- Έναν δείκτη `root` στον κόμβο-ρίζα του δέντρου, τύπου `FPnode`. Η ρίζα είναι ο μόνος κόμβος που δεν αντιστοιχεί σε αντικείμενο της βάσης και δεν έχει πατέρα. Γι' αυτό και το γνώρισμα `item` παίρνει την τιμή -1 και ο `parent` την τιμή `null`. Το `count` του αυξάνεται κάθε φορά που ένα κόμβος προστίθεται στο δέντρο και άρα στο τέλος της κατασκευής του, θα περιέχει το συνολικό πλήθος των κόμβων του δέντρου.

- Έναν map `freqs`, τύπου `FreqsTable`, που αντιστοιχίζει κάθε συχνό αντικείμενο στο support του. Με βάση αυτόν γίνεται ο έλεγχος για πιθανή διαγραφή αντικειμένου μετά από κάποια ενημέρωση του δέντρου ή κατά την κατασκευή των conditional FP-trees για την FP-growth.

- Έναν map `headerTable`, τύπου `HeaderTable`, ο οποίος αντιστοιχίζει κάθε αντικείμενο σε έναν vector με όλους τους κόμβους στο δέντρο που το περιέχουν. Ο vector αυτός υλοποιεί τη διασύνδεση των κόμβων που έχουν το ίδιο αντικείμενο που είναι απαραίτητη για την κατασκευή των conditional FP-trees και την διαγραφή αντικειμένων. Επομένως η κατά πλάτος διάσχιση του δέντρου, η οποία από τον ορισμό 2.2 γίνεται μέσω των nodeliks, τώρα γίνεται διατρέχοντας τον vector του αντίστοιχου αντικειμένου.

Κώδικας 4.3 Ορισμός κλάσης `FPtree`.

```
typedef std::map<item_t,count_t> FreqsTable;
typedef std::vector<FPnode*> NodeVec;
typedef std::map<item_t,NodeVec> HeaderTable;

class FPtree {
public:
  FPnode* root;
  FreqsTable freqs;
  HeaderTable headerTable;
};
```

Η κλάση `BuildFPtree` είναι η βασική κλάση χειρισμού και επεξεργασίας του FP-tree. Ως

γνωρίσματα έχει το ελάχιστο support threshold της βάσης `minsupp`, το συνολικό πλήθος δοσοληψιών που περιέχει `nr_of_trans`, έναν πίνακα `freqOrder` τύπου vector με την ταξινομημένη λίστα συχνών αντικειμένων και φυσικά το FP-tree `fptree`. Η εισαγωγή των ταξινομημένων συχνών αντικειμένων κάθε δοσοληψίας της βάσης στο FP-tree γίνεται με τη μέθοδο `addTransToTree` ακολουθώντας τον Αλγόριθμο 2.1. Μετά την κατασκευή του, το δέντρο αποθηκεύεται μαζί με τα υπόλοιπα γνωρίσματα της κλάσης στον σκληρό δίσκο σε δυαδική μορφή με τη μέθοδο `dump`, ώστε να καταστεί δυνατή η ανάκτησή της. Επειδή, όμως, τόσο η εγγραφή όσο και το διάβασμα του αρχείου γίνεται σειριακά ενώ το FP-tree περιλαμβάνει δείκτες μεταξύ των κόμβων, πρέπει να υπάρχει μία μέθοδος που θα μετατρέπει το δέντρο σε σειριακή μορφή. Την εργασία αυτή αναλαμβάνει η `serializeTree` (Κώδικας 4.4) η οποία βασίζεται στην αναζήτηση κατά βάθος (DFS) και ουσιαστικά επισκέπτεται όλους τους κόμβους του δέντρου αντιστοιχίζοντας στον καθένα μία μοναδική ετικέτα. Έτσι, κατά την αποθήκευση, οι δείκτες αντικαθιστώνται από την ετικέτα του κόμβου στον οποίο δείχνουν.

Κώδικας 4.4 Μέθοδος `serializeTree`.

```
size_t BuildFPtree::serializeTree(FPnode* visit, map<FPnode*,item_t>& nodeMap)
{
    size_t size = nodeMap.size();
    nodeMap[visit] = size;
    for ( CIter it = visit->children.begin(); it!= visit->children.end(); it++ )
        serializeTree(it->second,nodeMap);
    return nodeMap.size();
}
```

Η `BuildFPtree` περιλαμβάνει και άλλες μεθόδους επεξεργασίας του FP-tree που χρησιμοποιούνται στην εκτέλεση διαφόρων εργασιών πάνω στο δέντρο ενώ ταυτόχρονα εξυπηρετούν και τους σκοπούς των υπόλοιπων κλάσεων του προγράμματος. Ο ορισμός της φαίνεται παρακάτω.

Κώδικας 4.5 Ορισμός κλάσης `BuildFPtree`.

```
class BuildFPtree {
public:
    count_t minsupp;
    count_t nr_of_trans;
    ItemVector freqOrder;

    BuildFPtree() {}

    virtual ~BuildFPtree() {}
```

```
    class SinglePIter {
    public:
        FPnode* c;

        SinglePIter(FPnode* c): c(c) {}

        void incr() { c = c->parent; }

        bool atEnd(){
            if(c->item==-1) return true;
            else return false;
        }

        item_t getCurrItem(){ return c->item; }
    };

    SinglePIter getSinglePIter(FPtree* t,item_t curritem) {
        return SinglePIter(t->headerTable[curritem][0]); }

    FPtree* getFPtree() { return &fptree; }

    void initFPtree(FPtree* tree);

    template<class T>
    void addTransToTree(T& trans, FPtree* tree,count_t c);

    bool buildTree(std::istream& in, double& relminsup);

    template<class T>
    void updateTransToTree(T& trans, ItemVector& newitems);

    count_t updateTree(std::istream& in, ItemVector& newitems);

    item_t singlePrefixPath(FPtree* tree);

    void deleteSinglePath(FPtree* t, item_t curritem);

    FPtree* projectTree(FPtree* t, item_t curritem);

    bool pruneTree(FPtree* tree, bool FPGROWTH);

    void appendChildren (Children& target, Children& source, FPnode* parent,
                         FPtree* tree, bool SWAP );

    void deallocNode(FPnode* node);

    void deallocTree(FPtree* t);

    size_t serializeTree(FPnode* visitor, std::map<FPnode*,item_t>& nodeMap);

    void dump(std::ostream& out);

    void retrieveFPtree(std::istream& in);

protected:
    FPtree fptree;

    void deleteNode(FPtree* tree, FPnode* node);

    void removeItem(FPtree* tree, item_t item);
};
```

4.2.2 Κατασκευή merged FP-tree

Ο προτεινόμενος αλγόριθμος `merge` δύο FP-trees, που περιγράφηκε θεωρητικά στο Κεφάλαιο 3, υλοποιείται μέσω της κλάσης `MergeFPtrees` (Κώδικα 4.6).Η κλάση αυτή, όπως φαίνεται και από το UML διάγραμμα του Σχήματος 4.1, είναι μία *υποκλάση* της `BuildFPtree`, επομένως κληρονομεί όλα τα χαρακτηριστικά και τις μεθόδους της ενώ προσθέτει επιπλέον τις μεθόδους που χρειάζονται για το `merge`.

Κώδικας 4.6 Ορισμός κλάσης `MergeFPtrees`.

```
class MergeFPtrees : public BuildFPtree {
public:
    MergeFPtrees() { }

    ~MergeFPtrees() { }

    FPtree* addTrees(FPtree* tree, FPtree* addtree);

    void swapChildren (FPnode* a, FPnode* b);

    void assignChildren(FPnode* jnode, FPnode* jjnode, FPnode* nnode);

    void swapNodes(item_t& item_j, item_t& item_jj, FPtree* tree);

    void adjustFPtree(BuildFPtree* str);

    void initTree(BuildFPtree* first, BuildFPtree* second);

    void mergeTrees(BuildFPtree& str1, BuildFPtree& str2);

    unsigned checkForUpdate(BuildFPtree* str1, BuildFPtree* str2,
                            ItemVector& newitems);
};
```

Ο λόγος για τον οποίο επιλέχθηκε αυτή η υλοποίηση είναι οι εξής:

- Το `merge` μπορεί να θεωρηθεί σαν ενημέρωση του ενός δέντρου από το άλλο και εννοιολογικά εντάσσεται στις μεθόδους επεξεργασίας ενός FP-tree, οι οποίες περιλαμβάνονται στην βασική κλάση `BuildFPtree` (διαγραφή κόμβων, προσθήκη-αφαίρεση δοσοληψιών στο δέντρο, έλεγχος για ύπαρξη απλού μονοπατιού κ.τ.λ).

- Πολλές από τις μεθόδους της `BuildFPtree` είναι απαραίτητες για την υλοποίηση των Αλγορίθμων 3.1 και 3.2, οπότε δε χρειάζεται να οριστούν ξανά αλλά χρησιμοποιούνται απευθείας από την `MergeFPtrees`. Επιπλέον, μπορεί να χρησιμοποιηθεί απευθείας και η μέθοδος αποθήκευσης της κλάσης στον σκληρό δίσκο και άρα να δοθεί η δυνατότητα για incremental εφαρμογή του `merge`.

- Το αποτέλεσμα της ένωσης των δύο δέντρων ουσιαστικά αποτελεί ένα αντικείμενο της κλάσης `BuildFPtree` αφού είναι ένα FP-tree που έχει το δικό του support, τη δική του σειρά

συχνότητας και αντιστοιχεί σε μία βάση με n δοσοληψίες –έχει δηλαδή όλα τα γνωρίσματα της BuildFPtree.

Ακολουθώντας τα βήματα του Αλγορίθμου 3.1 γίνονται οι παρακάτω διαδικασίες.

Βήμα 1ο: Καθορισμός σειρά συχνότητας του merged FP-tree

Αρχικά, τα δέντρα ανακτώνται από τον δίσκο με τη μέθοδο retrieveFPtree της BuildFPtree, η οποία είναι η ακριβώς αντίστροφη της dump. Δημιουργούνται έτσι δύο κλάσεις τύπου BuildFPtree που έχουν ως γνωρίσματα όλα τα απαραίτητα στοιχεία για το merge. Στα FP-trees των κλάσεων γίνεται έλεγχος μέσω της checkForUpdate για πιθανή ανάγκη ενημέρωσης από τις αντίστοιχες βάσεις δοσοληψιών, ανιχνεύοντας αν υπάρχουν αντικείμενα στο ένα δέντρο που δεν υπάρχουν στο άλλο και αντιστρόφως. Αν προκύψουν τέτοια αντικείμενα, συλλέγονται σε έναν πίνακα vector newitems για κάθε δέντρο και καλείται η μέθοδος updateTree της BuildFPtree για την εισαγωγή των νέων αντικειμένων στο δέντρο, η οποία χρησιμοποιεί τη μέθοδο updateTransToTree.

Κώδικας 4.7 Μέθοδος updateTransToTree.

```
template<class T>
void BuildFPtree::updateTransToTree(T& trans, ItemVector& newitems) {
    FPnode* cnode = fptree.root;
    for ( size_t i = 0; i < trans.size(); i++ ) {
        CIter it = cnode->children.find(trans[i]);
        if ( it == cnode->children.end() ) {
            FPnode* nnode = new FPnode;
            nnode->item = trans[i];
            nnode->count = 1;
            nnode->parent = cnode;
            nnode->children.clear();
            cnode->children[trans[i]]=nnode;
            fptree.root->count+=1;
            fptree.freqs[trans[i]]+=1;
            fptree.headerTable[trans[i]].push_back(nnode);
            cnode = nnode;
        }
        else {
            VIter it = find(newitems.begin(), newitems.end(),trans[i]);
            if(it!=newitems.end()) {
                fptree.root->count+=1;
                fptree.freqs[trans[i]]+=1;
                cnode->children[trans[i]]->count+=1;
            }
            cnode = cnode->children[trans[i]];
        }
    }
}
```

Η διαδικασία εισαγωγής, updateTransToTree, είναι μια παραλλαγή της addTransToTree, καθώς ουσιαστικά εισάγει όσες δοσοληψίες της βάσης περιέχουν αντικείμενα που πρέπει να προ-

στεθούν αλλά χωρίς να αυξάνει το `count` των κόμβων που υπάρχουν ήδη στο δέντρο.

Μετά την ενημέρωση τα δέντρα έχουν ακριβώς τα ίδια αντικείμενα αλλά όχι κατ' ανάγκη με την ίδια σειρά. Για να καθοριστεί η τελική σειρά, που θα είναι και η σειρά συχνότητας των αντικειμένων για το merged FP-tree, καλείται η διαδικασία `initTree` κατά την οποία οι maps `freqs` των δύο δέντρων αθροίζουν τις αντίστοιχες καταχωρήσεις τους. Έτσι, προκύπτουν οι τελικές συχνότητες των αντικειμένων στο merged FP-tree. Τέλος, τα attributes `minsupp` και `nr_of_trans` της κλάσης `MergeFPtrees` παίρνουν τις σωστές τους τιμές.

Βήμα 2ο : Μετασχηματισμός των FP-trees

Με βάση τις νέες συχνότητες των αντικειμένων, όπως καθορίστηκαν στο προηγούμενο βήμα, και το νέο ελάχιστο support `minsupp` γίνεται ο μετασχηματισμός των δύο δέντρων. Τη διαδικασία αυτή αναλαμβάνει η `adjustFPtree` (Κώδικας 4.8) η οποία καλείται μία φορά για κάθε δέντρο και είναι η υλοποίηση της του Αλγορίθμου 3.2. Η procedure `path_adjust` του εν λόγω αλγορίθμου αντιστοιχεί στη μέθοδο `swapNodes`.

Κώδικας 4.8 Μέθοδος μετασχηματισμού FP-tree `adjustFPtree`.

```
void MergeFPtrees::adjustFPtree(BuildFPtree* str) {
    //prune the tree to remove infrequent items
    pruneTree(str->getFPtree(),false);

    //apply bubbleSort to the frequency order according to the total
    //frequencies of each item in the freqs table
    for ( size_t i = 0; i < str->freqOrder.size()-1; i++ ) {
        for ( size_t j = 0; j < str->freqOrder.size()-1-i; j++ ) {
            //in case two adjacent items have to be swaped
            if ( (str->getFPtree()->freqs[str->freqOrder[j]] <
                    str->getFPtree()->freqs[str->freqOrder[j+1]]) ||
                ( (str->getFPtree()->freqs[str->freqOrder[j]] ==
                        str->getFPtree()->freqs[str->freqOrder[j+1]]) &&
                  (str->freqOrder[j] > str->freqOrder[j+1])) ) {

            //call swap nodes for the respective items
            swapNodes(str->freqOrder[j],str->freqOrder[j+1],str->getFPtree());

            //swap the items in the frequency order
            swap(str->freqOrder[j], str->freqOrder[j+1]);
            }
        }
    }
}
```

Βήμα 3ο : Ένωση των FP-trees

Μετά το τέλος των μετασχηματισμών τα δέντρα εκτός από τα ίδια αντικείμενα έχουν και το ίδιο σχήμα. Το μόνο που απομένει για την ολοκλήρωση της διαδικασίας του `merge` των FP-trees

είναι η ένωση των δέντρων. Η μέθοδος `addTrees`(Κώδικας 4.9), ουσιαστικά προσθέτει το ένα

Κώδικας 4.9 Ένωση FP-trees με τη μέθοδο `addTrees`.

```
FPtree* MergeFPtrees::addTrees(FPtree* tree, FPtree* addtree) {
    //update root's count
    tree->root->count+=addtree->root->count;
    //add all nodes of addtree to the respective elements of tree's headerTable
    for ( HIter it1 = tree->headerTable.begin();
          it1 != tree->headerTable.end(); it1++ )
        for ( NIter it2 = addtree->headerTable[it1->first].begin();
              it2 != addtree->headerTable[it1->first].end(); it2++ ) {
            tree->headerTable[it1->first].push_back(*it2);
        }
    //call appendChildren to hang addtree from tree's root
    appendChildren ( tree->root->children, addtree->root->children,
                     tree->root, tree, false );
    return tree;
}
```

δέντρο στο άλλο καλώντας την `appendChildren` –η οποία υλοποιεί τον Αλγόριθμο 3.4 και έχει οριστεί στην `BuildFPtree`– και επιστρέφει το FP-tree `fptree` της κλάσης `MergeFPtrees`.

Κώδικας 4.10 Μέθοδος `appendChildren`.

```
void BuildFPtree::appendChildren ( Children& target, Children& source,
                                   FPnode* parent, FPtree* t, bool SWAP )
{
    for ( CIter it = source.begin(); it!=source.end(); ) {
        CIter iter = target.find(it->first);
        if ( iter!=target.end() ) {
            iter->second->count += it->second->count;
            NIter itr = find (t->headerTable[it->first].begin(),
                              t->headerTable[it->first].end(),it->second);
            t->headerTable[it->first].erase(itr);
            if(!SWAP)
                appendChildren(iter->second->children, it->second->children,
                               iter->second, t, false);
            deallocNode(it->second);
            source.erase(it++);
        } else {
            source[it->first]->parent = parent;
            target[it->first]=it->second;
            ++it;
        }
    }
}
```

Πρέπει να σημειωθεί ότι η `appendChildren` είναι μία μέθοδος συγχώνευσης κόμβων που έχουν ίδιο αντικείμενο και ίδιο πατέρα και καλείται ύστερα από διαγραφές ή μετασχηματισμούς.

4.2.3 Μέθοδος FP-growth

Στο FP-tree που προκύπτει είτε από την `BuildFPtree` είτε από την `MergeFPtrees` εκτελείται ο αλγόριθμος FP-growth για την εύρεση των συχνών προτύπων της βάσης στην οποία αντιστοιχούν. Η υλοποίηση του αλγορίθμου έγινε μέσω της ανίστοιχης κλάσης `FPgrowth` ο ορισμός της οποίας φαίνεται στον Κώδικα 4.11. Η κλάση έχει παραμετροποιηθεί για να μπορεί να χρησιμοποιηθεί για οποιοδήποτε υλοποίηση του FP-tree αρκεί αυτή να περιλαμβάνει μεθόδους κατασκευής conditional FP-tree και ελέγχου για ύπαρξη απλού μονοπατιού.

Κώδικας 4.11 Ορισμός κλάσης `FPgrowth`.

```
template<class STRUCTURE> class FPgrowth { public:

    typedef typename STRUCTURE::SinglePIter SinglePIter;

    count_t minsupp;
    std::ofstream& out;
    STRUCTURE* tree;
    count_t cfreq;
    ItemVector curritemset;

    void outputPush(item_t item) {
        curritemset.push_back(item); }

    void outputPop() {
        curritemset.pop_back();
    }

    void outputWriteLow(count_t supp) {
        writeItemsetAndCounter(out,curritemset.begin(),curritemset.end(),supp);
    }

    void outputWrite(count_t supp) {
        outputWriteLow(supp);
    }

    void outputPushAndWrite(item_t item, count_t supp) {
        outputPush(item); outputWrite(supp);
    }

    void singlePRecurse(typename STRUCTURE::SinglePIter it){
        while(it.incr(),!it.atEnd()) {
            outputPushAndWrite(it.getCurrItem(),cfreq);
            singlePRecurse(it);
        }
        outputPop();
    }

    void mineFI(FPtree* t);

    void findFI(STRUCTURE& str);

    FPgrowth(std::ofstream& out): out(out) {}

    ~FPgrowth() {} ;
```

Η βασική μέθοδος που εκτελεί το mining στο FP-tree είναι η `mineFI` (Κώδικας 4.12). Σύμφωνα με τον Αλγόριθμο 2.3, πρώτα ελέγχεται αν το δέντρο έχει απλό μονοπάτι. Αυτό γίνεται μέσω της μεθόδου `singlePrefixPath` της `BuildFPtree` η οποία επιστρέφει το αντικείμενο του τελευταίου κόμβου του μονοπατιού ή -1 αν δεν υπάρχει απλό μονοπάτι. Αν υπάρχει, ακολουθείται η αναδρομική διαδικασία `singlePRecurse` για την εύρεση όλων των δυνατών συνδυασμών του μονοπατιού με το αντίστοιχο support χρησιμοποιώντας ένα αντικείμενο της κλάσης `SinglePIter` που έχει οριστεί στην `BuildFPtree` και αποτελεί ουσιαστικά έναν iterator για το απλό μονοπάτι. Κάθε συνδυασμός που προκύπτει γράφεται στο αρχείο των αποτελεσμάτων με τη μέθοδο

Κώδικας 4.12 Εύρεση συχνών προτύπων με τη μέθοδο `mineFI`.

```
template<class STRUCTURE>
void FPgrowth<STRUCTURE>::mineFI(FPtree* t) {
    if(t) {
        item_t spdepth;
        spdepth = tree->singlePrefixPath(t);
        if(spdepth) {
            SinglePIter it = tree->getSinglePIter(t,spdepth);
            while(!it.atEnd()) {
                this->cfreq = t->freqs[it.c->item];
                outputPushAndWrite(it.c->item,cfreq);
                singlePRecurse(it);
                it.incr();
            }
            tree->deleteSinglePath(t,spdepth);
        }
        for(RVIter it = tree->freqOrder.rbegin();
            it != tree->freqOrder.rend(); it++)
        {
            if(t->freqs[*it] < minsupp) continue;
            this->cfreq = t->freqs[*it];
            outputPushAndWrite(*it,cfreq);
            outputPop();
            outputPush(*it);
            FPtree* cfpt = new FPtree();
            cfpt = tree->projectTree(t,*it);
            mineFI(cfpt);
            outputPop();
            tree->deallocTree(cfpt);
        }
    }
}
```

`outputPushAndWrite`. Στη συνέχεια, για να απομονώσουμε το υπόλοιπο δέντρο ώστε να εκτελέσουμε το mining, 'διαγράφουμε' το απλό μονοπάτι απ' το FP-tree με την `deleteSinglePath` της `BuildFPtree`. Η διαγραφή όμως δε συνεπάγεται διαγραφή των κόμβων του απλού μονοπατιού από το δέντρο αλλά μηδενισμό των support τους από τον map `freqs` του FP-tree. Αυτό γίνεται για να παραμείνουν μεν στο δέντρο και να συμμετάσχουν στην κατασκευή των conditional FP-trees των υπόλοιπων αντικειμένων του δέντρου αλλά να μην εγείρουν την αναδρομική διαδικασία γιατί τότε θα προκύψουν διπλά αποτελέσματα. Έτσι, για κάθε αντικείμενο της σει-

ράς συχνότητας, ξεκινώντας από το λιγότερο συχνό, που έχει support μεγαλύτερο ή ίσο του `minsupp` της βάσης, κατασκευάζεται το conditional FP-tree με τη μέθοδο `projectTree` (Κώδικας 4.13) της `BuildFPtree` η οποία ουσιαστικά προβάλλει το FP-tree ως προς ένα αντικείμενο και η μέθοδος `mineFI` καλείται αναδρομικά.

Κώδικας 4.13 Μέθοδος κατασκευής conditional FP-tree.

```
FPtree* BuildFPtree::projectTree(FPtree* full, item_t curritem) {
    FPtree* projtree = new FPtree();
    initFPtree(projtree);
    deque<item_t> trans;
    //for all nodes that contain curritem
    for ( NIter it = full->headerTable[curritem].begin();
            it!=full->headerTable[curritem].end(); it++ ) {
        FPnode* cnode = *it;
        trans.clear();
        count_t c = cnode->count;
        //catch paths that correspond the conditional pattern base of curritem
        FPnode* nnode = cnode->parent;
        while ( nnode!=full->root ) {
            trans.push_front(nnode->item);
            nnode = nnode->parent;
        }
        //add these to the conditional fp-tree
        addTransToTree(trans, projtree, c);
    }
    //prune infrequent items
    pruneTree(projtree,true);
    if(!projtree->root->children.empty())
        return projtree;
    else {
      deallocTree(projtree);
      return NULL;
    }
}
```

ΚΕΦΑΛΑΙΟ 5

Μελέτη Επίδοσης του αλγορίθμου

Η θεωρητική μελέτη του merge δύο δέντρων συχνών αντικειμένων ανέδειξε τα πλεονεκτήματά της σε σχέση με την κλασική κατασκευή του FP-tree για την ένωση δύο βάσεων με δοσοληψίες. Η υλοποίησή του, δίνει τη δυνατότητα για πειραματική επιβεβαίωση της θεωρίας. Στο κεφάλαιο που ακολουθεί παρουσιάζεται η μελέτη επίδοσης της προτεινόμενης μεθόδου ένωσης δύο FP-trees. Περιγράφεται αναλυτικά η πειραματική διαδικασία και τα αποτελέσματα που προέκυψαν ενώ στο τέλος παραθέτονται τα συμπεράσματα της εργασίας καθώς και προτάσεις για πιθανές επεκτάσεις.

5.1 Τα δεδομένα

Για την διενέργεια των πειραμάτων κατασκευάστηκαν συνθετικά δεδομένα που ακολουθούν την *Zipfian* κατανομή. Η κατανομή αυτή βασίζεται στον εμπειρικό *Νόμο του Zipf* [16] που προτάθηκε πρώτα από τον Αμερικανό γλωσσολόγο και φιλόλογο George Kingsley Zipf και δηλώνει [9] ότι:

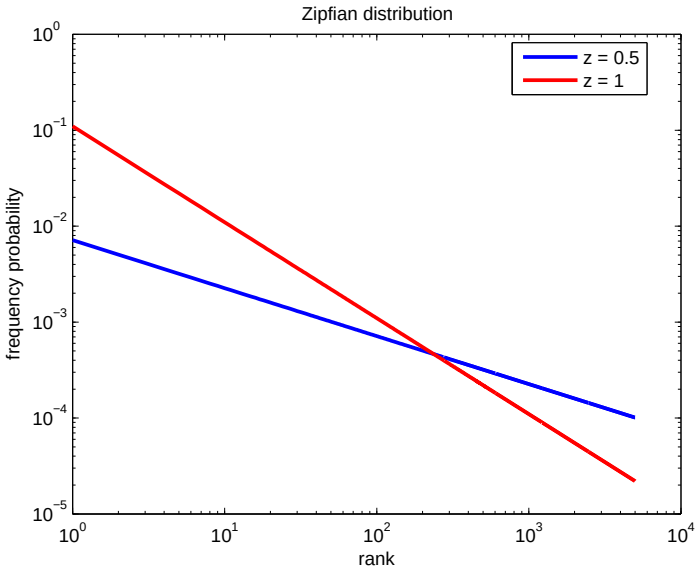
> *η συχνότητα P_i με την οποία απαντάται η i-οστή συχνότερη λέξη σε μία γλώσσα είναι σχεδόν αντιστρόφως ανάλογη της τάξης-i της λέξης στο πίνακα συχνοτήτων, $P_i \propto 1/i^z$, όπου η παράμετρος z είναι κοντά στη μονάδα.*

Ο νόμος του Zipf σημαίνει πρακτικά ότι λίγες λέξεις χρησιμοποιούνται πολύ συχνά ενώ πολλές είναι οι λέξεις που η συχνότητά τους είναι μικρή.

Ορισμός 5.1 (Zipfian distribution [6]). *Έστω p διαφορετικές τιμές (1,...,p) οι οποίες ακολουθούν Zipfian κατανομή με παράμετρο z. Τότε η πιθανότητα της τιμής i είναι $1/\, ci^z$, όπου $c = \sum_{i=1}^{p} 1/\, i^z$ για την κανονικοποίηση της κατανομής. Η Zipfian κατανομή λέγεται συχνά και κατανομή Z ενώ η παράμετρος z παίρνει συνήθως τιμές κοντά στη μονάδα.*

Ο λόγος για τον οποίο επιλέχθηκε η συγκεκριμένη κατανομή, είναι γιατί απαντάται πολύ συχνά σε πραγματικά δεδομένα και επομένως διασφαλίζει την αξιοπιστία των πειραματικών αποτελεσμάτων.

Τα δεδομένα που κατασκευάστηκαν ακολουθούν *Zipfian* κατανομή με παράμετρο $z = 0.5$ και $z = 1$ και με μέγεθος λεξιλογίου $p = 5000$. Στο Σχήμα 5.1 φαίνεται η κατανομή των εμφανίσεων των αντικειμένων του λεξιλογίου για κάθε μία από τις δύο παραμέτρους. Οι βάσεις αποτελούνται από 10^5, $2.5 \cdot 10^5$ και 10^6 δοσοληψίες. Επομένως, κατασκευάζονται συνολικά έξι διαφορετικές βάσεις με δοσοληψίες.



Σχήμα 5.1: *Zipfian* κατανομή των πειραματικών δεδομένων.

Παρατήρηση 5.1. Από το διάγραμμα του σχήματος 5.1 προκύπτει ότι για $z = 0.5$ τα αντικείμενα με μεγάλο rank έχουν μεγαλύτερη πιθανότητα εμφάνισης απ' ότι στην αντίστοιχη περίπτωση για $z = 1$. Επομένως, για χαμηλό support, το πλήθος των συχνών αντικειμένων της βάσης με $z = 0.5$ θα είναι μεγαλύτερο απ' ο,τι για $z = 1$.

5.2 Μεθοδολογία

Για κάθε μία από τις έξι βάσεις N δοσοληψιών ακολουθήκε η εξής διαδικασία:

Βήμα 1ο: Εκτελείται ο αλγόριθμος FP-growth με κατωφλικό support $s[\%]$ απ' όπου προκύπτει το σύνολο των συχνών προτύπων της βάσης ενώ ταυτόχρονα καταγράφεται ο χρόνος διάρκειας της διαδικασίας $t_{rebuild}$.

Βήμα 2ο: Η βάση χωρίζεται σε δύο ίσα μέρη, δημιουργώντας δύο νέες βάσεις με πλήθος δοσοληψιών $N_1 = N_2 = N/2$.

Βήμα 3ο: Εφαρμόζεται η FP-growth σε κάθε μία από τις δύο βάσεις με ελάχιστα support $s_1[\%]$ και $s_2[\%]$ αντίστοιχα, για τα οποία σύμφωνα με το Πόρισμα 3.1 του Κεφαλαίου 3 θα πρέπει να ισχύει:

$$\frac{s_1 + s_2}{2} = s \tag{5.1}$$

Τα FP-trees που κατασκευάζονται αποθηκεύονται στον δίσκο.

Βήμα 4ο: Εκτελείται ο αλγόριθμος του merge ανακτώντας τα αποθηκευμένα δέντρα, εφαρμόζεται η FP-growth στο τελικό δέντρο, απ' όπου προκύπτουν τα συχνά πρότυπα για την ένωση των δύο βάσεων, και καταγράφεται ο χρόνος διάρκειας της διαδικασίας t_{merge}.

Βήμα 5ο: Συγκρίνονται τα αποτελέσματα της FP-growth για την αρχική βάση με αυτά που προκύπτουν από την FP-growth στο merged FP-tree, για να ελεγχθεί η ορθότητα του αλγορίθμου, και οι χρόνοι $t_{rebuild}$ και t_{merge} για να αποτιμηθεί η επίδοσή του.

Βήμα 6ο: Τα βήματα 3–5 επαναλαμβάνονται για διάφορα support s_1 και s_2 τέτοια ώστε να ισχύει η (5.1) .

Βήμα 7ο: Τα βήματα 1–5 επαναλαμβάνονται για διαφορετικό support s.

Επειδή τα αρχεία των βάσεων ήταν από 5 έως 51 MB για τη διαίρεσή τους αναπτύξαμε ένα απλό πρόγραμμα σε Perl ενώ η σύγκριση των αρχείων αποτελεσμάτων της FP-growth έγινε με τη μέθοδο `cmp` του UNIX.

Τα πειράματα διενεργήθηκαν σε πλατφόρμα Windows XP SP2 και περιβάλλον cygwin 1.5.21-1 με επεξεργαστή Intel(R) Pentium(R) D CPU 3.00GHz και 1,00 GB RAM.

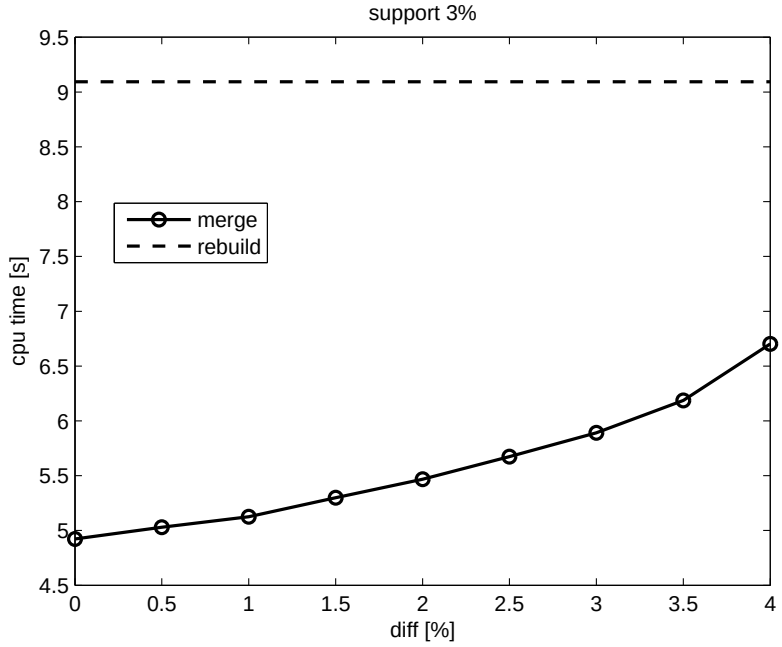
5.3 Αποτελέσματα

Ο έλεγχος των αρχείων αποτελεσμάτων της FP-growth έδειξε ότι, σε κάθε περίπτωση, τα συχνά πρότυπα που προκύπτουν από το `merge` των FP-trees είναι ακριβώς τα ίδια με αυτά που προκύπτουν για το εξ' αρχής κατασκευασμένο FP-tree, αποδεικνύοντας έτσι την ορθότητα του αλγορίθμου. Στα διαγράμματα που ακολουθούν παρουσιάζεται η επίδοση του σε σχέση με την απόλυτη διαφορά diff μεταξύ των supports των δύο υποβάσεων και σε αντιπαραβολή με την κλασική κατασκευή του FP-tree.

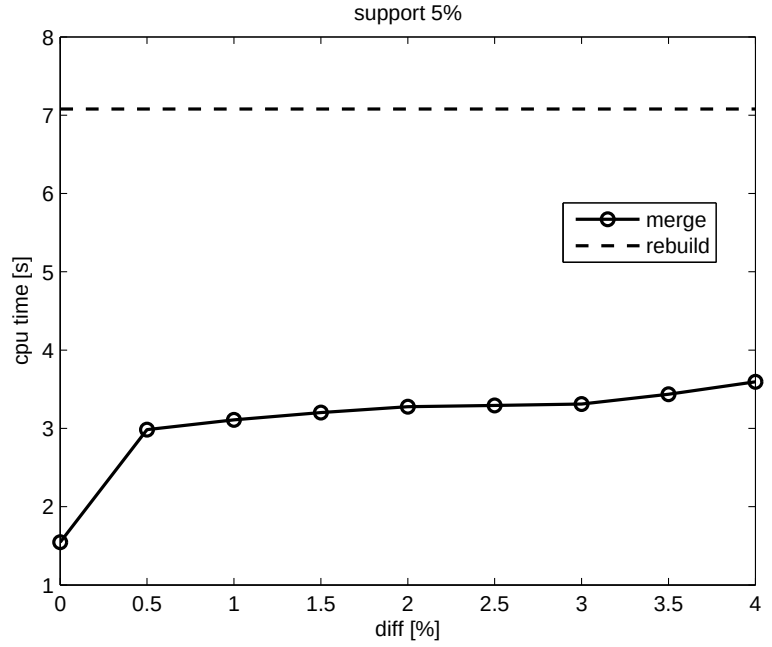
5.3.1 Γενική Περιγραφή

Μέγεθος βάσης 100.000 δοσοληψίες

Στο Σχήμα 5.2 φαίνεται η επίδοση της μεθόδου για 10^5 δοσοληψίες και παράμετρο κατανομής $z = 1$. Παρατηρούμε ότι και στις δύο περιπτώσεις το `merge` είναι πιο γρήγορο από την κλασική



(i)



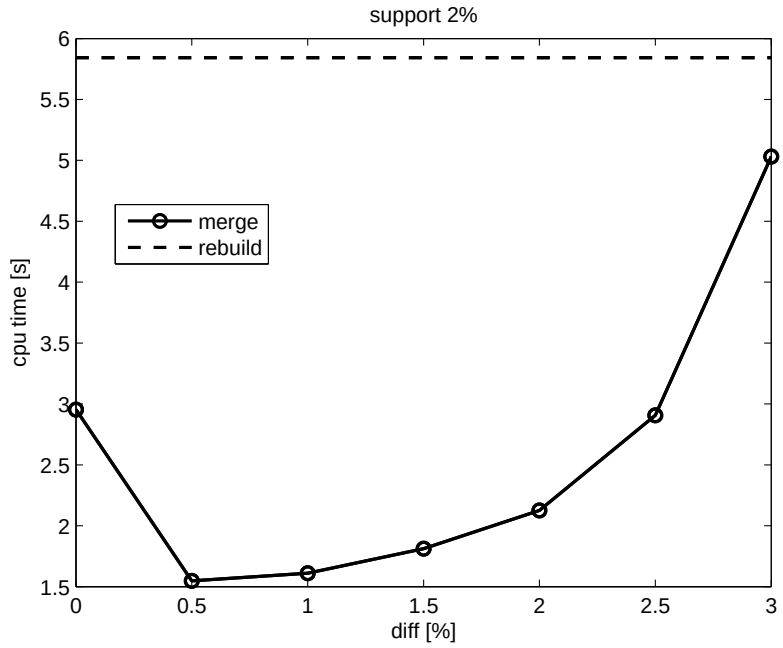
(ii)

Σχήμα 5.2: Επίδοση της μεθόδου για 10^5 δοσοληψίες, z = 1.

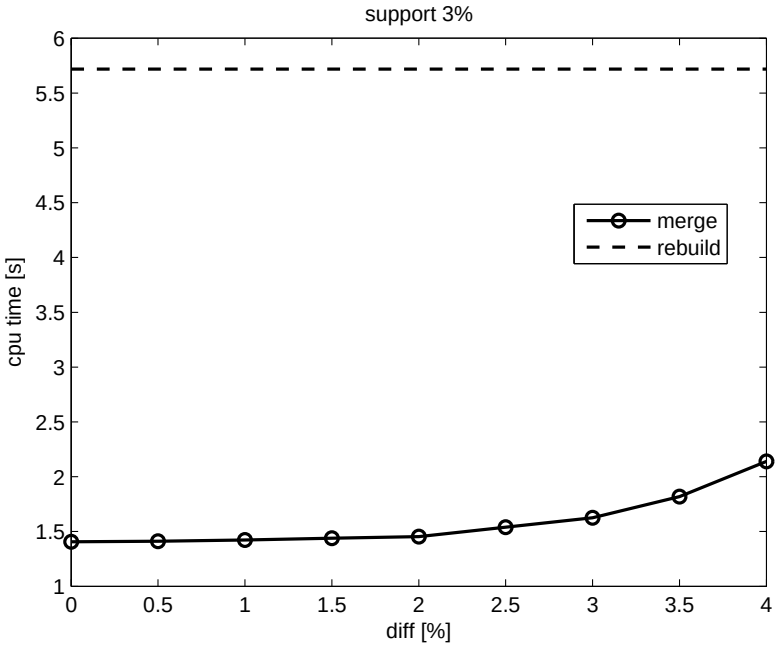
διαδικασία κατασκευής του FP-tree. Ιδιαίτερα όταν η απόλυτη διαφορά μεταξύ των supports των δύο υποβάσεων είναι μικρή, η μέθοδος είναι μέχρι και 5 φορές πιο γρήγορη. Γενικά, παρατηρούμε ότι όσο μεγαλώνει η διαφορά των supports ο χρόνος που απαιτείται για το `merge` αυξάνεται –χωρίς βέβαια να ξεπερνάει ποτέ το `rebuild`. Αυτό οφείλεται στο γεγονός ότι όσο μεγαλύτερη είναι η διαφορά μεταξύ των supports, τόσο διαφέρει και το πλήθος των αντικειμένων που υπάρχουν στα δύο δέντρα. Συγκεκριμένα, στην επιμέρους βάση με το μικρότερο support, τα αντικείμενα που υπάρχουν στο FP-tree της είναι περισσότερα, με αποτέλεσμα να απαιτείται εκτεταμένη ενημέρωση του δεύτερου FP-tree πριν την πράξη της ένωσης των δέντρων. Η ενημέρωση αντιστοιχεί σε επίσκεψη της αντίστοιχης βάσης, δηλαδή σε πράξη I/O, που κατά τεκμήριο είναι χρονοβόρα. Επομένως, το μεγαλύτερο κόστος για το `merge` είναι, όπως αναμενόταν, η ενημέρωση των δύο δέντρων καθώς οι πράξεις μετασχηματισμού και ένωσης μπορούν να γίνουν πάρα πολύ γρήγορα.

Στο πρώτο από τα δύο διαγράμματα του Σχήματος 5.3, τα οποία αντιστοιχούν σε βάσεις με παράμετρο κατανομής $z = 0.5$, παρατηρείται το εξής. Για diff$= 0$ ο χρόνος του `merge` είναι μεγαλύτερος απ' ότι για μερικά diff$\neq 0$. Αυτό σημαίνει απλά, ότι κατά τον χωρισμό της βάσης, τα συχνά αντικείμενα δεν κατανεμήθηκαν ομοιόμορφα στις δύο επιμέρους βάσεις. Δηλαδή, ένα συχνό αντικείμενο της βάσης εμφανίζεται τις περισσότερες φορές στη μία επιμέρους βάση και λίγο στην άλλη, ενώ για ένα άλλο ισχύει το αντίθετο. Επομένως, τα δύο δέντρα έχουν πολλά διαφορετικά αντικείμενα και κατ' επέκταση απαιτούν πολλές ενημερώσεις. Ακριβώς λόγω αυτής της ανομοιομορφίας, όταν τα support των υποβάσεων είναι διαφορετικά, τυχαίνει τα δέντρα να έχουν περισσότερα κοινά αντικείμενα. Ωστόσο, σε όλες τις περιπτώσεις, τυχαίες ή μη, το `merge` συνεχίζει να είναι πιο γρήγορο.

Στο διάγραμμα του Σχήματος 5.3(ii), όπου το support της ολικής βάσης είναι 3%, το `merge` γίνεται πολύ πιο γρήγορα. Κι αυτό γιατί το ελάχιστο support που μπορεί να έχει η μία από τις δύο επιμέρους βάσεις ώστε να ισχύει η (1.1) δεν είναι τόσο μικρό όσο στην προηγούμενη περίπτωση, κι έτσι αν και η μέγιστη απόσταση diff είναι η ίδια ή και μεγαλύτερη, οι ενημερώσεις που απαιτούνται δεν είναι τόσες πολλές. Η επίδοση του αλγορίθμου εδώ είναι εντυπωσιακή, όχι τόσο για την τάξη της διαφοράς της από το rebuild όσο για την σταθερότητά της.



(i)

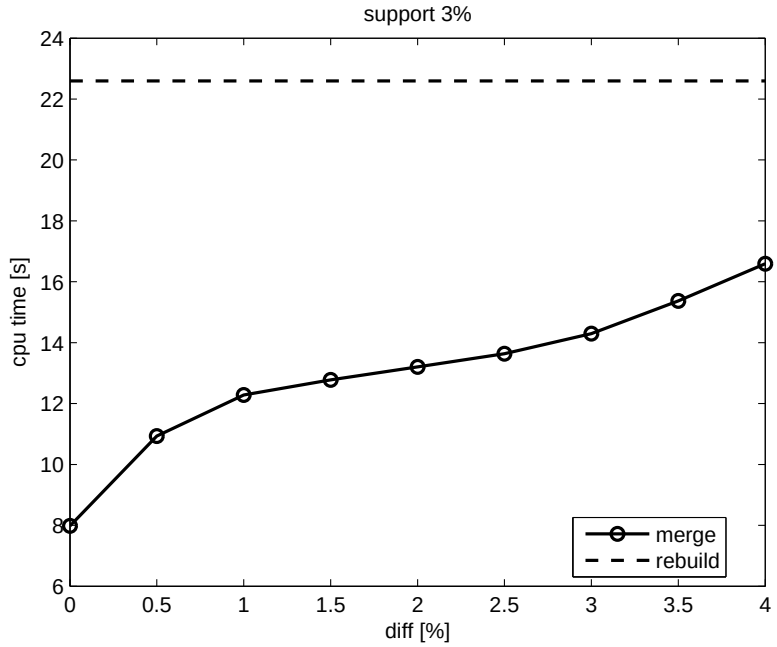


(ii)

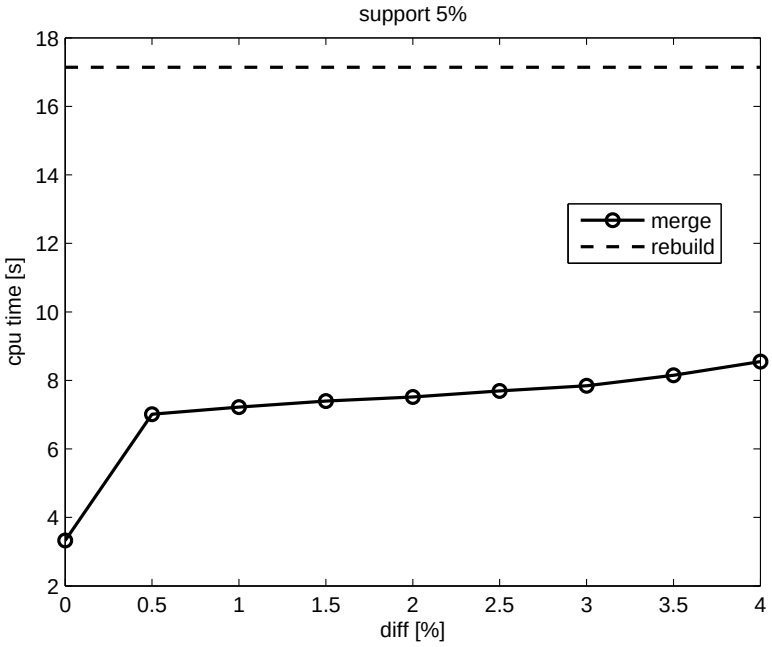
Σχήμα 5.3: Επίδοση της μεθόδου για 10^5 δοσοληψίες, z = 0.5.

Μέγεθος βάσης 250.000 δοσοληψίες

Τα αντίστοιχα διαγράμματα για 250.000 δοσοληψίες παρουσιάζονται στα Σχήματα 5.4, 5.5. Και εδώ τα συμπεράσματα δε διαφέρουν. Η μόνη επίδραση της αύξησης του μεγέθους της βάσης



(i)

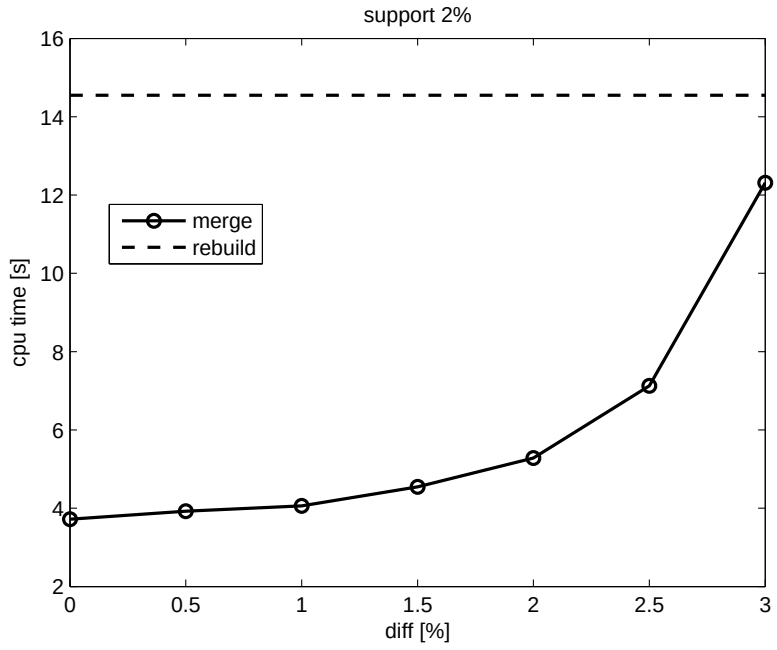


(ii)

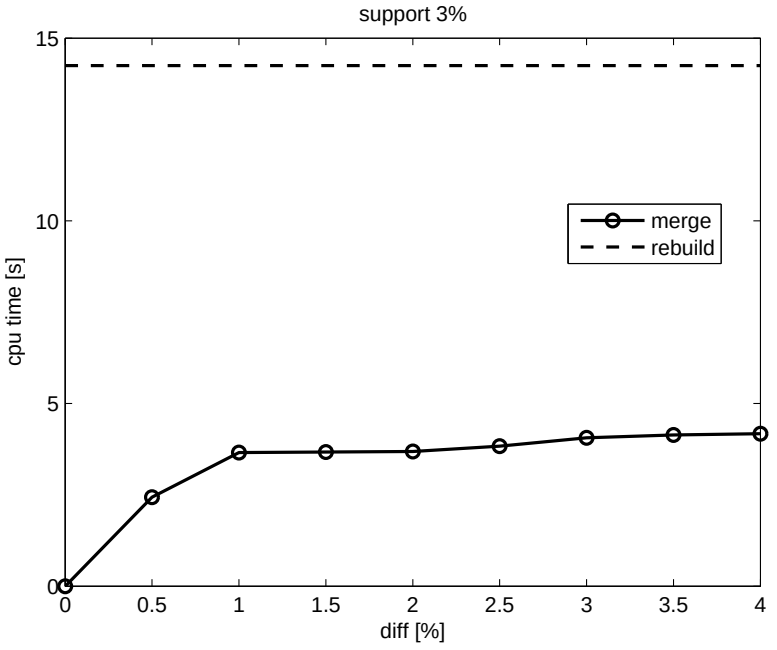
Σχήμα 5.4: Επίδοση της μεθόδου για $2.5 \cdot 10^5$ δοσοληψίες, z = 1.

είναι στην αύξηση του απαιτούμενου χρόνου και για τους δύο τρόπους κατασκευής του FP-

tree. Και σε αυτές τις περιπτώσεις το `merge` δεν ξεπερνάει το `rebuild`, αν και μπορεί να φτάσει αρκετά κοντά του (Σχήμα 5.5(i)). Πρέπει να σημειώσουμε ιδιαίτερα την περίπτωση του



(i)

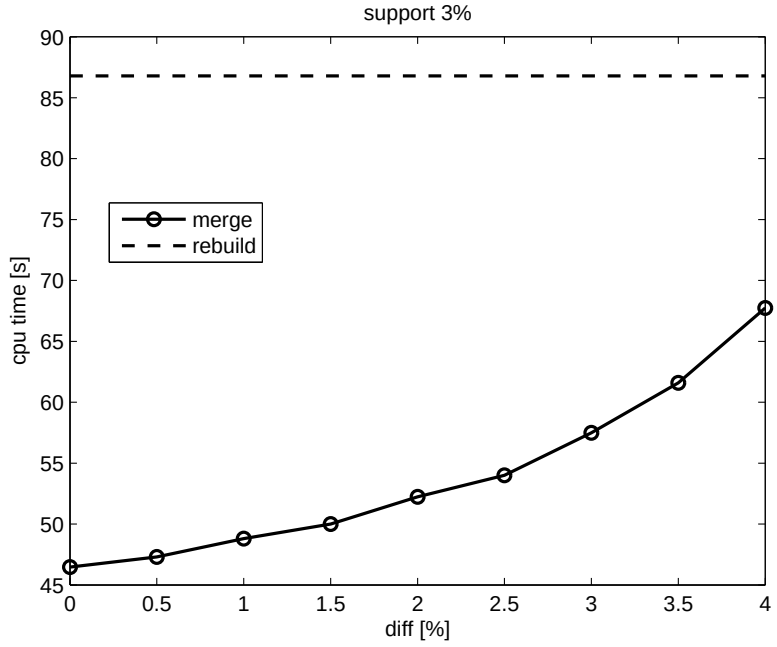


(ii)

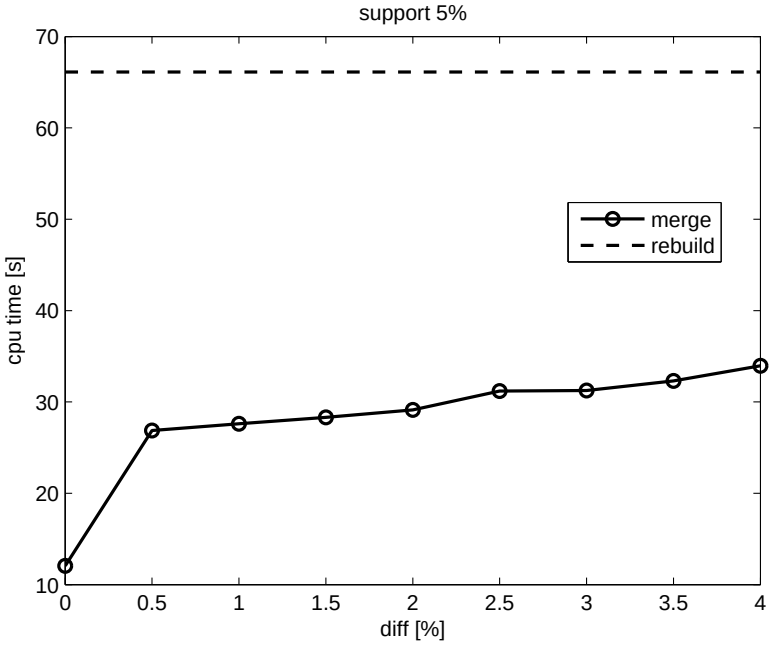
Σχήμα 5.5: Επίδοση της μεθόδου για $2.5 \cdot 10^5$ δοσοληψίες, z = 0.5.

Σχήματος 5.5(ii), όπου όταν τα support των υποβάσεων είναι ίδια, ο χρόνος που απαιτείται για το `merge` τείνει στο μηδέν, υποδηλώνοντας ότι δεν απαιτήθηκαν καθόλου ενημερώσεις και πιθανώς

ούτε μετασχηματισμοί. Αν και αυτή η περίπτωση δεν είναι ο κανόνας, η εμφάνισή της αποδεικνύει ότι, σε ιδανικές συνθήκες, ο αλγόριθμος είναι πάρα πολύ αποτελεσματικός.



(i)

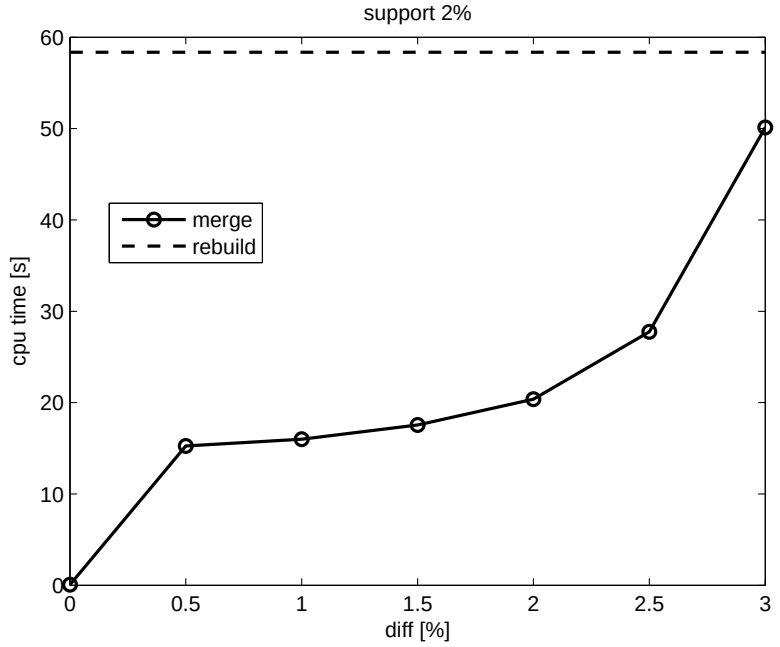


(ii)

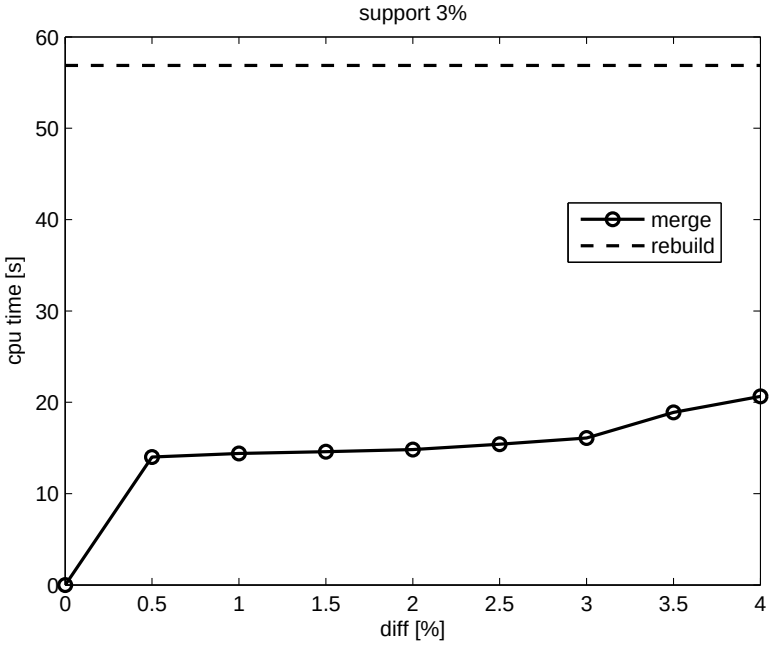
Σχήμα 5.6: Επίδοση της μεθόδου για 10^6 δοσοληψίες, z = 1.

Μέγεθος βάσης 1.000.000 δοσοληψίες

Παρότι το μέγεθος της βάσης έχει γίνει μία τάξη μεγαλύτερο από αυτό των βάσεων που παρουσιασιάστηκαν παραπάνω, η επίδοση του αλγορίθμου συγκριτικά με το rebuild δε μοιάζει να επηρεάζεται. Για κάθε περίπτωση, ισχύουν αντίστοιχες παρατηρήσεις με αυτές που έχουν γίνει



(i)



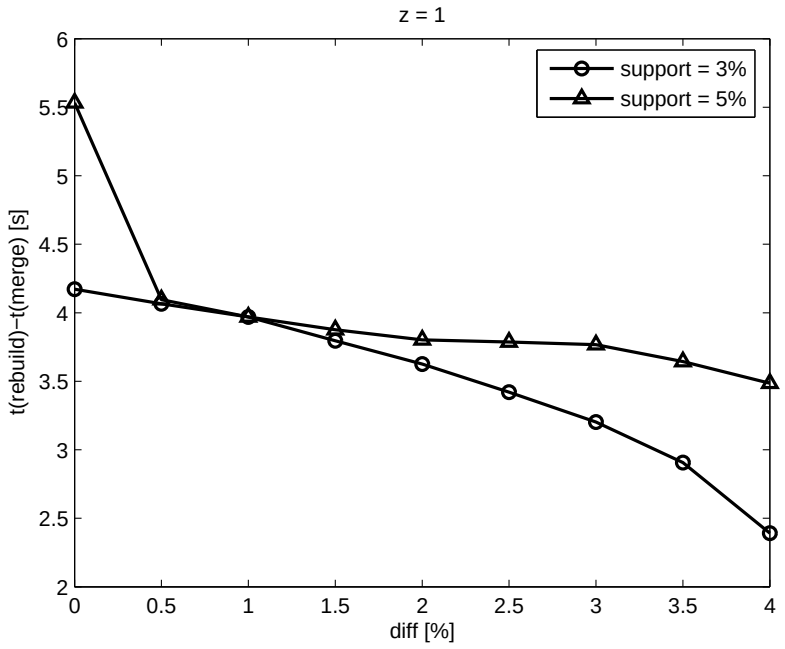
(ii)

Σχήμα 5.7: Επίδοση της μεθόδου για 10^6 δοσοληψίες, z = 0.5.

για τις μικρότερες βάσεις. Τα αποτελέσματα των πειραμάτων φαίνονται στα Σχήματα 5.6– 5.7. Και πάλι ο προτεινόμενος αλγόριθμος `merge` αποδεικνύεται ιδιαίτερα αποτελεσματικός, ενώ στα Σχήματα 5.7(i)–(ii), και για την περίπτωση της ισότητας των supports των επιμέρους βάσεων, ο χρόνος για το `merge` των FP-trees τείνει στο μηδέν.

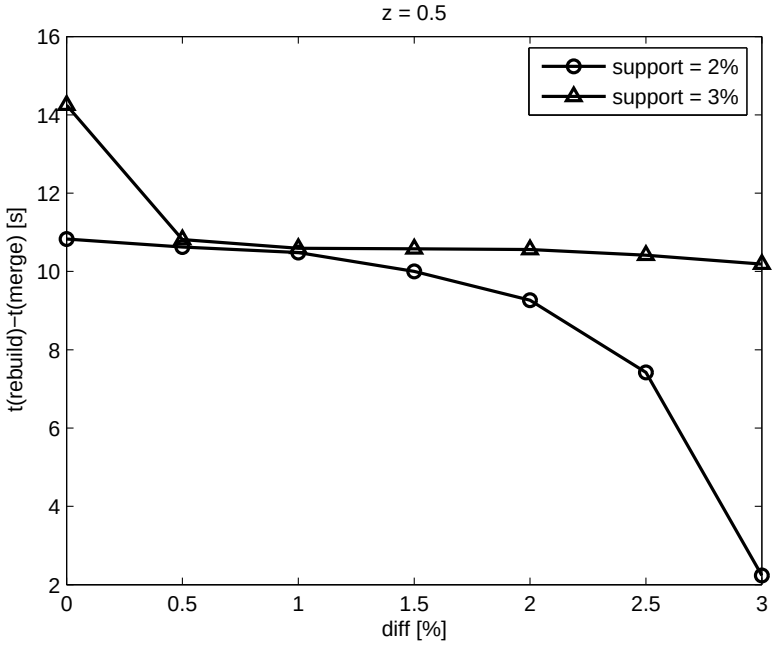
5.3.2 Επίδραση της μεταβολής του support

Στα διαγράμματα των Σχημάτων 5.8, 5.9, 5.10 παρουσιάζεται η επίδοση του αλγορίθμου `merge` σε σχέση με το `rebuild` για δύο διαφορετικά support. Η γενική παρατήρηση είναι ότι, όταν το

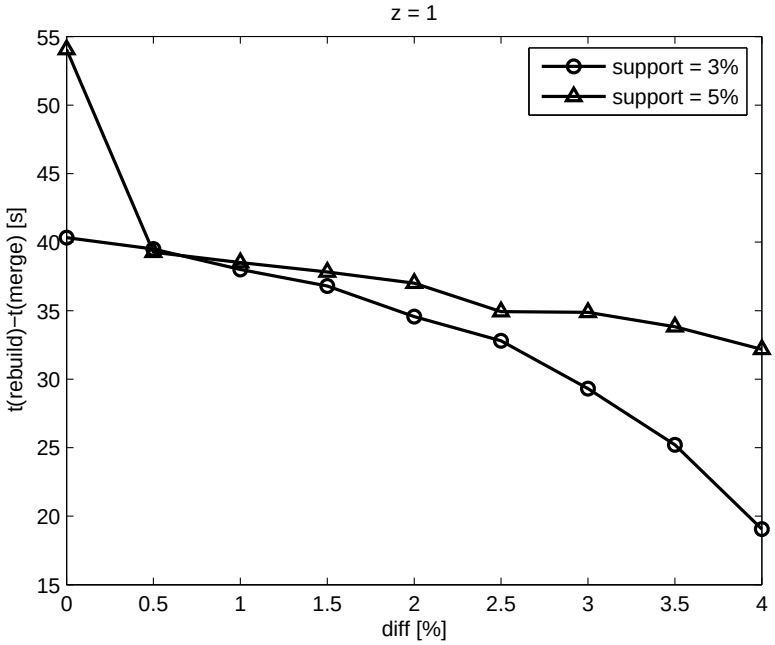


Σχήμα 5.8: Επίδραση του support για 10^5 δοσοληψίες, z = 1.

support της βάσης αυξάνεται, ο αλγόριθμος αποδίδει καλύτερα ακόμα και για μεγάλη απόλυτη διαφορά μεταξύ των supports των επιμέρους βάσεων. Αντιθέτως, για μικρότερο support η επίδοση του αλγορίθμου πέφτει απότομα με την αύξηση του diff. Αυτό συμβαίνει γιατί, μειώνοντας το support, η χαμηλότερη τιμή που μπορεί να πάρει το s_1 (ή το s_2) ώστε να ισχύει η (1.1), κινείται σε αρκετά χαμηλά επίπεδα (για τα πειράματά μας, κοντά στο 0.25%). Έτσι, είναι πολλά τα αντικείμενα που εισάγονται στο ένα FP-tree και δεν εισάγονται στο άλλο, τα οποία στο τελικό δέντρο δε θα είναι συχνά. Ωστόσο, η ενημέρωση του δεύτερου FP-tree πρέπει να γίνει για να ελεγχθούν οι συχνότητές τους. Η ενημέρωση αυτή κοστίζει υπονομεύοντας την επίδοση του αλγορίθμου.



Σχήμα 5.9: Επίδραση του support για $2.5 \cdot 10^5$ δοσοληψίες, z = 0.5.

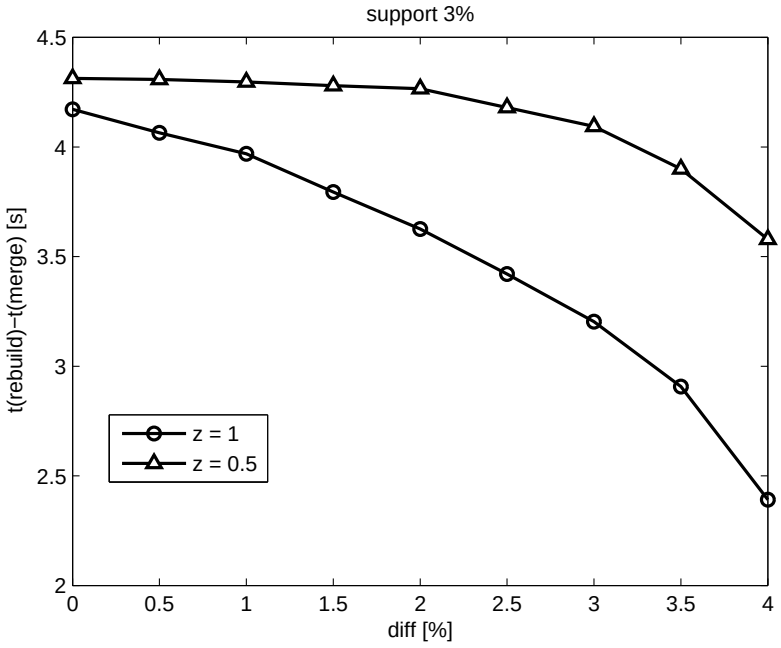


Σχήμα 5.10: Επίδραση του support για 10^6 δοσοληψίες, z = 1.

5.3.3 Επίδραση της μεταβολής του z

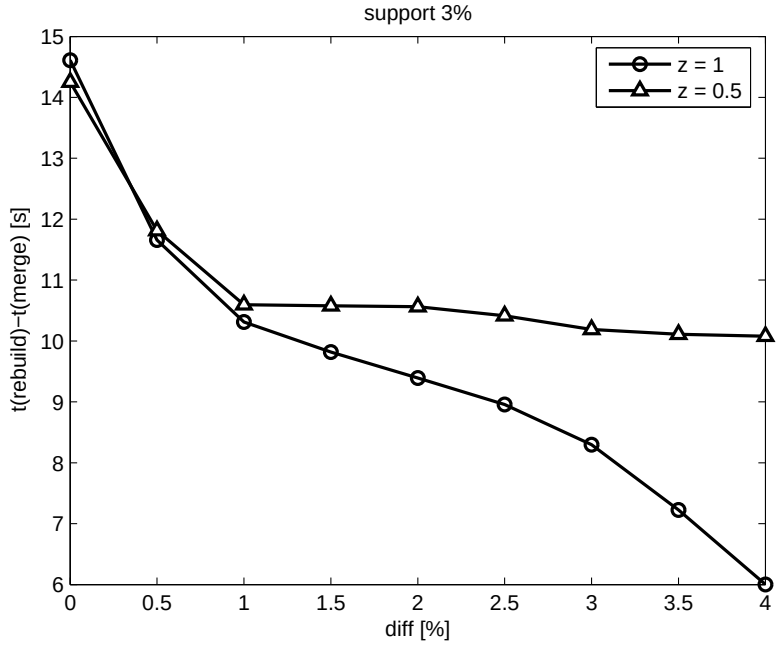
Στα διαγράμματα των Σχημάτων 5.11, 5.12, 5.13 παρουσιάζεται η επίδοση του αλγορίθμου `merge` σε σχέση με το `rebuild` για τις δύο παραμέτρους κατανομής $z = 0.5$ και $z = 1$. Παρατηρούμε ότι για όλα τα μεγέθη βάσης και για το ίδιο support, ο αλγόριθμος είναι πιο αποδοτικός

όταν $z = 0.5$. Επιπλέον, όσο η διαφορά μεταξύ των support των επιμέρους βάσεων αυξάνεται, η



Σχήμα 5.11: Επίδραση της μεταβολής του z στη βάση με 10^5 δοσοληψίες.

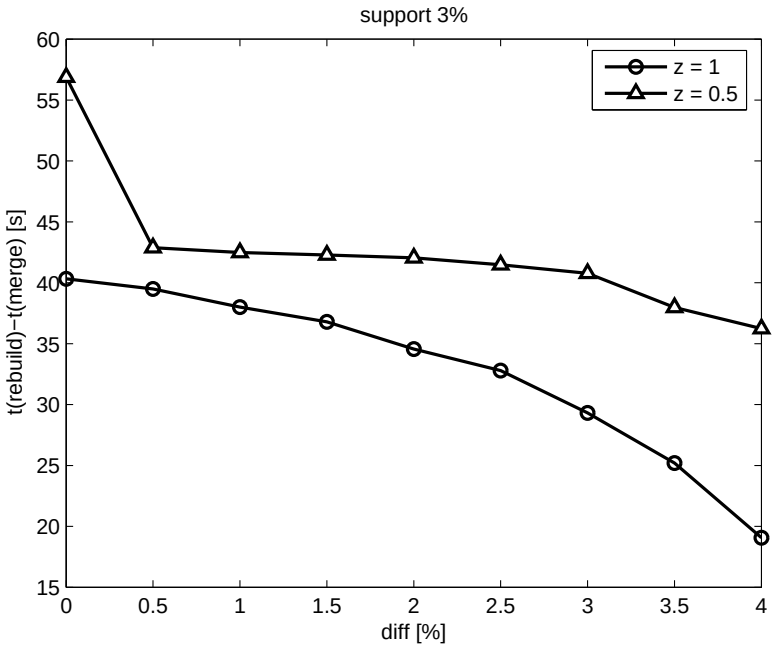
επίδοσή του για $z = 1$ μειώνεται πάρα πολύ, πλησιάζοντας αρκετά το `rebuild` ενώ αντιθέτως, για $z = 0.5$, παρουσιάζεται πολύ πιο σταθερή. Η συμπεριφορά αυτή οφείλεται στο γεγονός ότι για το μικρότερο z η διασπορά των συχνών εμφανίσεων είναι μεγαλύτερη (Παρατήρηση 5.1). Έτσι,



Σχήμα 5.12: Επίδραση της μεταβολής του z στη βάση με $2.5 \cdot 10^5$ δοσοληψίες.

ακόμα κι όταν η διαφορά των support των επιμέρους βάσεων φτάνει στο όριο, η πιθανότητα τα

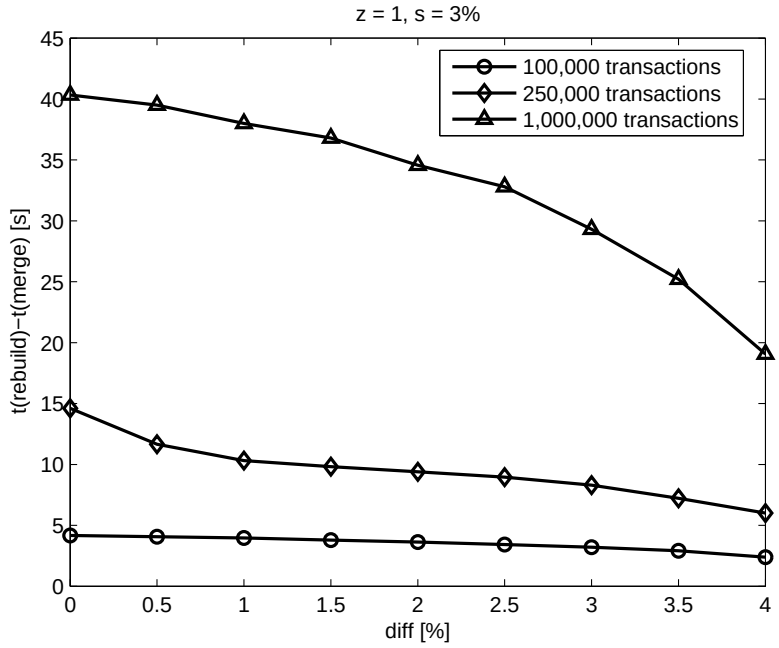
δύο FP-trees να έχουν αρκετά κοινά αντικείμενα είναι μεγαλύτερη με αποτέλεσμα να μη χρειάζονται χρονοβόρες ενημερώσεις από τη βάση. Πρέπει, βέβαια, να σημειώσουμε και πάλι την υπεροχή του `merge` έναντι του `rebuild`.



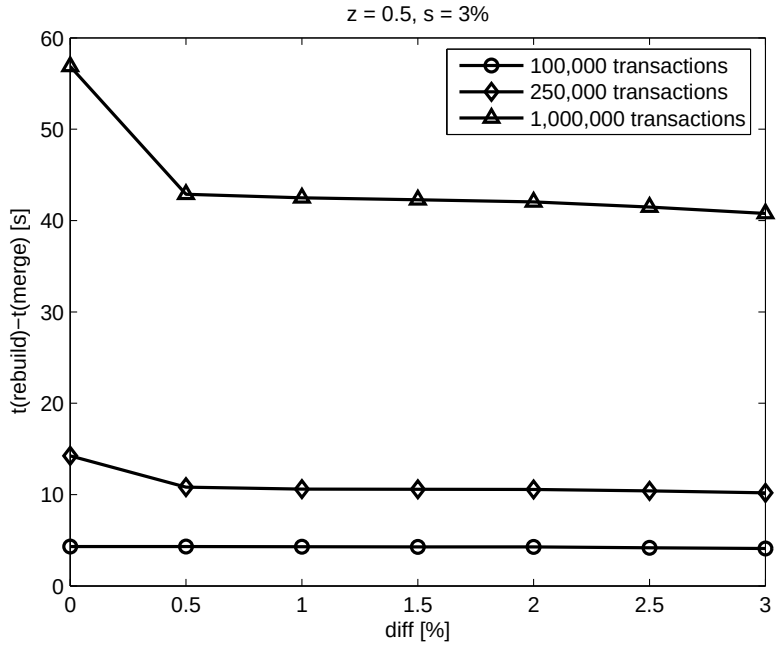
Σχήμα 5.13: Επίδραση της μεταβολής του z στη βάση με 10^6 δοσοληψίες.

5.3.4 Επίδραση του μεγέθους της βάσης

Τα διαγράμματα που ακολουθούν αποκαλύπτουν την επίδραση του μεγέθους της βάσης στην επίδοση του αλγορίθμου. Είναι εμφανές, τόσο από το διάγραμμα του Σχήματος 5.14 όσο και από αυτό του Σχήματος 5.15, ότι για περισσότερες δοσοληψίες η μέθοδος του `merge` δίνει καλύτερα αποτελέσματα. Η εξήγηση είναι η εξής: το χρονικό κόστος κατασκευής του FP-tree μέσω της κλασικής μεθόδου είναι ευθέως ανάλογο του μεγέθους της βάσης. Έτσι, για τη μεγαλύτερη από τις βάσεις των πειραμάτων, ο χρόνος που απαιτείται είναι ήδη μεγάλος. Αντιθέτως, το `merge` δεν εξαρτάται μόνο από το μέγεθος των επιμέρους βάσεων, καθώς η επίσκεψη ή όχι σε αυτές καθορίζεται από τις ενημερώσεις που τυχόν είναι απαραίτητες. Συνεπώς, για τις περιπτώσεις σχετικά μικρής διαφοράς των support, το κόστος των ενημερώσεων είναι μικρό, οι μετασχηματισμοί ούτως ή άλλως δεν διαρκούν πολύ, κι έτσι η σύγκριση με το rebuild δείχνει μεγάλη διαφορά στους χρόνους.



Σχήμα 5.14: Επίδραση του μεγέθους της βάσης για $z = 1$ και support$= 3\%$.



Σχήμα 5.15: Επίδραση του μεγέθους της βάσης για $z = 0.5$ και support$= 3\%$.

5.4 Συμπεράσματα

Από την μελέτη των αποτελεσμάτων της πειραματικής διαδικασίας προκύπτουν τα παρακάτω συμπεράσματα σχετικά με την επίδοση του αλγορίθμου.

- Η προτεινόμενη μέθοδος `merge` δύο FP-trees που αντιστοιχούν σε δύο βάσεις με δοσοληψίες πάνω στα ίδια αντικείμενα είναι πάντοτε πιο γρήγορη από την κλασική κατασκευή του FP-tree για την ένωση των βάσεων. Σε ιδανική περίπτωση μάλιστα, όπου δεν απαιτούνται ενημερώσεις ή μετασχηματισμοί, η επίδοση του είναι εντυπωσιακή.

- Το μεγαλύτερο κόστος προκύπτει όταν γίνονται ενημερώσεις των FP-trees από τις αντίστοιχες βάσεις δοσοληψιών. Γι' αυτό, όσο μεγαλύτερη είναι η διαφορά μεταξύ των support των δύο βάσεων, τόσο υπονομεύεται η επίδοση του αλγορίθμου.

- Ο αλγόριθμος συμπεριφέρεται καλύτερα για μεγάλη διασπορά των συχνών εμφανίσεων αντικειμένων στη βάση.

- Τέλος, η επίδοσή του σε σχέση με την κλασική κατασκευή του FP-tree είναι κατά βάση καλύτερη για μεγαλύτερο πλήθος δοσοληψιών.

Κατά συνέπεια, η προτεινόμενη μέθοδος μπορεί να αποτελέσει ένα χρήσιμο εργαλείο για την εξόρυξη συχνών προτύπων και ανταποκρίνεται στις απαιτήσεις και τους στόχους που θέσαμε στην αρχή της εργασίας.

5.5 Προτάσεις για περαιτέρω έρευνα

- Στην εργασία αυτή ασχοληθήκαμε με την ανάπτυξη μίας μεθόδου επαναχρησιμοποίησης των FP-trees που έχουν προκύψει από προηγούμενες εφαρμογές της FP-growth. Μία πιθανή επέκταση θα ήταν να μελετηθεί ένας τρόπος επαναχρησιμοποίησης και των αποτελεσμάτων της FP-growth που σε συνδυασμό με το `merge` θα δίνει ακόμα καλύτερα αποτελέσματα.

- Η υλοποίηση του FP-tree έγινε με τη χρήση maps της C++ STL για τη βελτίωση του κόστους αναζήτησης στο δέντρο. Θα μπορούσε να γίνει βελτιστοποίησή της χρησιμοποιώντας έναν κατάλληλα κατασκευασμένο hash table ώστε η αναζήτηση για τις ενημερώσεις και τις διαγραφές να γίνεται ακόμα γρηγορότερα. Εξάλλου, το μεγαλύτερο κόστος του αλγορίθμου εντοπίζεται στις πιθανές ενημερώσεις του δέντρου. Επομένως, μία τέτοια κατεύθυνση θα μπορούσε να βελτιώσει περαιτέρω την επίδοσή του.

Βιβλιογραφία

[1] Agarwal, R., Aggarwal, C., Prasad, V.V.V. A Tree Projection Algorithm For Generation of Frequent Itemsets. *Journal of Parallel and Distributed Computing*, 61:350–371, 2001.

[2] Agrawal, R. and Srikant, R. Fast algorithms for mining association rules. In *Proceedings of 1994 International Conference Very Large Databases (VLDB'94)*, pages 487–499, Santiago, Chile, 1994.

[3] Bodon, F. A trie-based Apriori implementation for mining frequent item sequences. In *Proc. 1st International Workshop on open source data mining: frequent pattern mining implementaions*, pages 56–65, Chicago, Illinois, 2005.

[4] Borgelt, C. and Kruse R. Induction of Association Rules: Apriori Implementation. In *Proc. 14th Conference on Computational Statistics (COMPSTAT)*, pages 395–400, 2002.

[5] Cormen, T., Leiserson, C., Rivest, R. and Stein C. *Intorduction to Algorithms.* The MIT Press, 2nd Edition, 2001.

[6] Godfrey, P. *Skyline Cardinality for Relational Processing: How many vectors are maximal?.* In *Proceedings 3rd International Symposium on Foundations of Information and Knowledge Systems, FoIKS 2004*, Wilheminenburg Castle, Austria, February, 2004.

[7] Han, J. and Kamber, M. *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2001.

[8] Han, J., Pei, J., Yin, Y. and Mao, R. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8:53–87, 2004.

[9] http://www.nslij-genetics.org/wli/zipf

[10] Josuttis, N. M. *The C++ Standard Library: A Tutorial and Reference.* Addison Wesley Longman, Inc., 1999.

[11] Koh, J. L. and Shieh, S. F. An Efficient Approach for Maintaining Association Rules Based on Adjusting FP-Tree Structures. In *Proc. 9th International Conference, DASFAA 2004*, Jesu Island, Korea, March, 2003.

[12] Shoemaker, C and Ruiz, C. *Association Rule Mining Algorithms for Set-Valued Data.* In *Proc. Intelligent Data Engineering and Automated Learning, 4th International Conference, IDEAL 2003*, Hong Kong, China, March 21-23, 2003.

[13] Tan, P. N., Steinbach, M. and Kumar, V. *Introduction to Data Mining.* Addison Wesley, 2005.

[14] Wang, K., Tang, L., Han, J. and Liu, J. *Top Down FP-Growth for Association Rule Mining.* In *Proc. Advances in Knowledge Discovery and Data Mining : 6th Pacific-Asia Conference, PAKDD 2002*, Taipel, Taiwan, May 6-8, 2002.

[15] Yan, Y., Li, Z., Wang, T., Chen, Y. and Chen, H. *Mining Maximal Frequent ItemSets Using Combined FP-Tree. Australian Conference on Artificial Intelligence 2004*, pages 475-487. Springer Berlin/Heidelberg, 2004.

[16] Zipf, G. K. *Human Behavior and the Principle of Least–Effort.* Addison Wesley, Cambridge MA, 1949.