Paper 1: Huffman Coding

Το paper αυτό κατατοπίζει τον αναγνώστη για μια πληθώρα από τρόπους και τεχνικές με τους οποίους μπορεί να κωδικοποιηθεί ένα αλφάβητο εισόδου με χρήση Κώδικα Huffman και κάνει μια έρευνα πάνω στην ανάπτυξη που έχει προκύψει ως αποτέλεσμα της ανακάλυψης και εκτεταμένης χρήσης του Κώδικα Huffman.

1 - Εισαγωγή

Σε αυτό το paper περιλαμβάνονται λεπτομέρειες υπολογισμού κώδικα καθώς επίσης και λειτουργιών κωδικοποίησης και αποκωδικοποίησης. Επίσης, εξετάζονται θέματα που περιλαμβάνουν σχετικούς μηχανισμούς με συστήματα αριθμητικής κωδικοποίησης τα σχετικά πρόσφατα ανεπτυγμένα ασύμμετρη προσέγγιση αριθμητικών συστημάτων και αναφέρονται μερικές άλλες παραλλαγές Κωδικοποίησης Huffman συμπεριλαμβανομένων των Κωδίκων περιορισμένου μήκους.

Αρχικά, γίνεται μια εκτενής εισαγωγή στην οποία παρουσιάζεται η σημασία του κώδικα Huffman. Η Κωδικοποίηση με Huffman αντιμετωπίζει το πολύ σημαντικό πρόβλημα της Αναπαράστασης Δεδομένων σε Υπολογιστικά Συστήματα. Η δουλειά του Huffman αποτελεί ορόσημο για την αποδοτική αποθήκευση πληροφορίας σε ένα σύστημα. Αν και η συνηθισμένη μορφή, με την οποία παρουσιάζεται μια Κωδικοποίηση σε Κώδικα Huffman, είναι με χρήση Δέντρου, όπου τα κλαδιά έχουν αριθμούς, η χρήση δέντρου είναι κατά τον συγγραφέα, ένας περισπασμός και παρακάτω στο paper τονίζεται ότι θα εφαρμοστούν υλοποιήσεις που αποφεύγουν την αναπαράσταση αποκλειστικά με δέντρα. Επίσης, ο συγγραφέας τονίζει πως πρέπει να η ικανότητα να γίνεται διαχωρισμός μεταξύ της εννοιών «σύνολο από κωδικολέξεις» και της έννοιας «σύνολο από μήκη κωδικολέξεων». Ορίζοντας τον κώδικα Huffman ως ένα σύνολο από μήκη κωδικολέξεων επιτρέπει να γίνει μια επιλογή μεταξύ διαφορετικών συνόλων κωδικολέξεων (που αναπαριστούν την πληροφορία σωστά) η οποία όπως θα φανεί είναι πολύ σημαντική ως προς την απόδοση της υλοποίησης.

Τα πρώτα τρία τμήματα του άρθρου αυτού δίνουν έναν οδηγό που περιγράφεται ο Κώδικας Huffman. Στην συνέχεια στο τέταρτο τμήμα δίνονται βελτιώσεις και παραλλαγές οι οποίες έχουν προέλθει μέσα από τα 60 χρόνια ύπαρξης του Κώδικα ενώ στην συνέχεια στο Τμήμα 5 δίνονται δυο συμπληρωματικές τεχνικές οι οποίες πετυχαίνουν Κωδικοποίηση-Εντροπίας, δηλαδή την μέγιστη δυνατή Κωδικοποίηση με τα ελάχιστα σύμβολα που μπορεί να υπάρξει, χωρίς να υπάρξει απώλεια πληροφορίας. Τέλος, στο τμήμα 6 αξιολογούνται τα σχετικά πλεονεκτήματα και μειονεκτήματα που

έχουν οι νέες τεχνικές που παρουσιάστηκαν απέναντι στα χαρακτηριστικά γνωρίσματα που χρησιμοποίησε πρωταρχικά ο Huffman.

1.1 - Κώδικες Ελάχιστου-Πλεονασμού

Το πρόβλημα που υπάρχει είναι το πρόβλημα αναπαράστασης ενός προθεματικού δέντρου με ελάχιστο πλεονασμό δηλαδή όσο πιο κοντά στην εντροπία γίνεται. Πρέπει να οριστούν τα παρακάτω:

- ***Αλφάβητο Πηγής (Source Alphabet)*** είναι ένα αλφάβητο n διακριτών συμβόλων θεωρείται πως είναι δεδομένο και αναπαρίσταται με ακεραίους στο εύρος [0,n-1].
- ***Βάρη (weights)*** δίνονται μαζί με το αλφάβητο και είναι ακέραιοι. Επίσης, μπορεί να είναι πραγματικοί που αν προστεθούν όλοι μαζί δίνουν μονάδα Συμβολίζονται με W και γίνεται η παραδοχή πως τα βάρη είναι μη αυξανόμενα (δηλαδή $w_i >= w_{i+1}$ για 0 <= i < n-1). Τέλος αν υπάρξει η περίπτωση όπου $w_i = 0$ τότε αυτό σημαίνει ότι το συγκεκριμένο σύμβολο μπορεί να αγνοηθεί και να χρησιμοποιηθεί ένα μειωμένο αλφάβητο με n’ < n σύμβολα.
- ***Αλφάβητο εξόδου (output alphabet)*** επίσης δίνεται συνήθως. Για μελέτη σε υπολογιστικά συστήματα είναι τα σύμβολα {0,1} συνήθως. Αυτό σημαίνει ότι θα υπάρχει δυαδική κωδικοποίηση Huffman.
- ***Κώδικας (code)*** είναι μια λίστα από n θετικούς ακεραίους $T = [l_i > 0 \mid 0<= I < n]$. Η ερμηνεία αυτού του ορισμού είναι ότι το σύμβολο I στο αλφάβητο αντιστοιχίζεται σε μια μοναδική και καθορισμένη κωδικολέξη με μήκος l_i και αποτελείται από σύμβολα του αλφάβητου εξόδου. Στη συνέχεια παραθέτοντας την ανισότητα Kraft-McMillan διατυπώνεται το συμπέρασμα ότι αν κάποιος κώδικας T ακολουθεί την ανισότητα Kraft-McMillan καλείται *εφικτός*. Ένας εφικτός κώδικας σημαίνει πως μπορεί να δοθεί σε κάθε σύμβολο μια κωδικολέξη με τέτοιον τρόπο ώστε καμία κωδικολέξη να μην είναι πρόθεμα μιας άλλης κωδικολέξης.
- ***Κόστος Εφικτού Κώδικα*** αντικατοπτρίζει το κόστος σε bits που απαιτείται για να κωδικοποιηθεί όλα το αλφάβητο εισόδου. Συμβολίζεται με: $C(W,T) = \sum w_i * l_i$ για i στο εύρος [0,n-1].

Για να είναι ο κώδικας T ελάχιστου πλεονασμού για βάρη W, θα πρέπει για κάθε άλλο κώδικα n-συμβόλων που επίσης είναι εφικτός, έστω T’, το κόστη αυτών των δύο κωδίκων να επιβεβαιώνουν ότι: C(W,T) <= C(W,T’).

2 - Αλγόριθμος Huffman

Αυτό το τμήμα παρουσιάζει την αρχή που διέπει την κατασκευή με κώδικα Huffman. Επίσης παρουσιάζει με αυξανόμενα επίπεδα λεπτομέρειας πως ο κώδικας Huffman υλοποιείται, επιδεικνύει ότι όντως τα τμήματα κώδικα είναι ελάχιστου πλεονασμού και στο τέλος παρουσιάζει και χρήση του κώδικα Huffman χωρίς να απαιτείται να χρησιμοποιηθούν δυαδικά αλφάβητα εξόδου → όχι δυαδική κωδικοποίηση.

2.1 - Η ιδέα του Huffman

Η απλή ιδέα του Huffman έχει ως εξής: Χρησιμοποιούνται n σύμβολα ως αλφάβητο εισόδου τα οποία θα χρησιμοποιηθούν για κωδικοποίηση και τους εκχωρούνται βάρη. Στην συνέχεια, χρησιμοποιώντας μια άπληστη διαδικασία τα δύο σύμβολα με το μικρότερο βάρος συνενώνονται σε ένα νέο σύμβολο το οποίο ως βάρος, του ανατίθεται το άθροισμα από τα βάρη των δύο επιμέρους συμβόλων. Μετά από n-1 επαναλήψεις το αλφάβητο εισόδου έχει συρρικνωθεί σε ένα σύμβολο, το οποίο περιέχει εσωτερικά του όλα τα σύμβολα από το αλφάβητο εισόδου. Μόλις συμβεί αυτό η διαδικασία κωδικοποίησης έχει ολοκληρωθεί.

Το μήκος κωδικολέξης li, που συσχετίζεται με το σύμβολο i, μπορεί να δοθεί μέσα από το πλήθος των φορών που το σύμβολο i, συμμετέχει στα βήματα συνένωσης με άλλα σύμβολα. Αν κατά την διαδικασία συνένωσης συμβόλων, στα βάρη που υπάρχουν, προκύψει ισοψηφία τότε ο τρόπος με τον οποίο θα γίνει η επιλογή για το ποιο σύμβολο θα επιλεγεί, ποικίλλουν. Η επιλογή μπορεί να είναι τυχαία ή με κάποιο συγκεκριμένο κριτήριο. Χωρίς απώλεια της γενικότητας ο συγγραφέας επιλέγει να αριθμήσει κάθε σύμβολο με αύξουσα σειρά και με αυτό τον τρόπο όταν υφίσταται ισοψηφία, να επιλέγεται ο κόμβος με τον μεγαλύτερο αριθμό. Αντίστοιχα θα μπορούσαν να είναι γράμματα, σύμβολα ή οτιδήποτε άλλο με τον μόνο περιορισμό να αντιστοιχίζονται μοναδικά Στο παράδειγμα που δίνεται όταν καταλήγει στο τελικό σύμβολο, δίνει την πληροφορία, πως το πόσο βαθιά βρίσκεται κάθε αριθμός αντιστοιχεί στο μήκος της κάθε κωδικολέξης. Το Μήκος της Κωδικολέξης δείχνει πόσες φορές έχει χρησιμοποιηθεί κάθε σύμβολο στην συνένωση και πόσα bits απαιτούνται για την κωδικοποίηση του. Το Κόστος Κωδικοποίησης δεν μπορεί να αυξηθεί ή μικρύνει ανάλογα με τον κώδικα ωστόσο, μπορεί να υπάρχουν διαφορετικοί κώδικες που επιτυγχάνουν το ίδιο αποτέλεσμα για το ίδιο αλφάβητο με διαφορετικά μέγιστα μήκη κωδικολέξεων. Παρακάτω στο κείμενο, παρουσιάζονται διάφορες υλοποιήσεις κωδικοποιήσεων με κώδικα Huffman.

2.2 - Υλοποίηση από Εγχειρίδια

Σε αυτή την κατηγορία υλοποίησης εντάσσεται η υλοποίηση που διδάσκεται στα περισσότερα εγχειρίδια κωδικοποίησης δεδομένων και είναι ένας τρόπος ο οποίος βασίζεται στην κατασκευή δέντρων. Το πρόβλημα κωδικοποίησης με κώδικα Huffman με χρήση δέντρων περιλαμβάνει την δημιουργία δέντρων και πράξεις επί αυτών. Αρχικά ο κώδικας Huffman με χρήση δέντρων περιλαμβάνει την εισαγωγή αλφαβήτου εισόδου μήκους n και με χρήση βαρών W. Τα βήματα υλοποίησης είναι τα παρακάτω:

1. Εισαγωγή n και W
2. Δημιουργία ουράς προτεραιότητας Q και προσθήκη σε αυτήν όλων των επιμέρους φύλλων (τα οποία φύλλα είναι κόμβοι με αριθμό ή γράμμα στο παραδείγματα του paper μαζί με βάρη δηλαδή (n,W)).
3. Επαναληπτική αναπροσαρμογή του Q με συνένωση κόμβων έτσι ώστε να προστεθούν οι κόμβοι με συνθήκη τερματισμού το μέγεθος των στοιχείων του Q να είναι 1.
4. Εξαγωγή του Q με τέτοιο τρόπο ώστε να φαίνεται η ιεραρχία των πραγματοποιημένων συνενώσεων.

Στα παραδείγματα του paper γίνεται η συνηθισμένη παραδοχή ανάθεσης bits κατά την οποία στην αριστερή ακμή αντιστοιχίζεται το 0 και στην δεξιά ακμή αντιστοιχίζεται το 1. Επίσης στο ένα δέντρο γίνεται και αναπροσαρμογή των κόμβων κατά λεξικογραφική σειρά για να υπάρχουν ταξινομημένα τα φύλλα ενώ στο άλλο δέντρο εφαρμόζεται αυστηρά ο αλγόριθμος που παρουσιάστηκε παραπάνω. Ο κώδικας Huffman για τον οποίο ο κανόνας για επίλυση των ισοπαλιών που θα προκύψουν στα βάρη, προτιμά σύμβολα με υψηλούς δείκτες (Γράμματα, αριθμοί ή Σύμβολα με Κωδικοποίηση ASCII) αντί για σύμβολα με χαμηλούς δείκτες έχει το πλεονέκτημα ότι περιέχει το ελάχιστο μέγιστο μήκος κωδικολέξης L. Τέλος, για κάθε συνδυασμό βαρών με αναδιατάξεις μπορούν να προκύψουν πολλαπλά σύνολα από κωδικολέξεις. Πιο συγκεκριμένα, αναφέρεται ότι 2^{n-1} αναδιατάξεις μπορούν να προκύψουν μόνο από αναδιατάξεις ακμών και ακόμα περισσότερες μπορούν να προκύψουν αν μη γειτονικά φύλλα ίδιου βάθους αλλάξουν το ένα με το άλλο.

Ο χρόνος που απαιτείται είναι O(nlogn) χρόνος για την εκτέλεση της υλοποίησης. Πιο συγκεκριμένα, απαιτούνται 2n-1 Εισαγωγές στοιχείων στην ουρά Q και 2n-1 εξαγωγές των κατάλληλων στοιχείων (με τα μικρότερα βάρη) από την Q ουρά για την τελική κατασκευή της Q ουράς και τον υπολογισμό του κώδικα Huffman.Σε αυτό το σημείο ο συγγραφέας κάνει την παρατήρηση πως τόσο στα περισσότερα εγχειρίδια

όσο και στην Wikipedia η διαδικασία της κωδικοποίησης και αποκωδικοποίησης θεωρείται ότι γίνεται, ακολουθώντας ένα μονοπάτι από το δέντρο στην ρίζα καθώς τα bits αποστέλλονται 1 προς 1 από το μέσο μετάδοσης.

2.2 - Προσέγγιση Υλοποίησης από van Leeuwen

Η παρατήρηση που έκανε ο van Leeuwen είναι πως υπό προϋποθέσεις, η υλοποίηση κώδικα Huffman μπορεί να γίνει σε O(n) χρόνο. Η πρόταση η οποία έκανε σχετίζεται με την παρατήρηση πως αν τα βάρη εισόδου W παρουσιάζονται ταξινομημένα, τότε οι εσωτερικοί κόμβοι που δημιουργούνται σε κάθε βήμα από συνένωση μικρότερων κόμβων ή φύλλων παράγονται σε μη φθίνουσα σειρά βαρών και επίσης απορροφώνται με τον ίδιο ακριβώς τρόπο. Επομένως το μόνο που χρειάζεται για να διαχειριστεί κανείς όσον αφορά τους εσωτερικούς κόμβους είναι μια ουρά. Ως εκ τούτου, αν τα βάρη εισόδου δίνονται ταξινομημένα 2 ουρές με σειρά προτεραιότητας FIFO, μπορούν να χρησιμοποιηθούν. Ο ένας αποθηκεύει τα φύλλα και κατασκευάζεται από τα αρχικά βάρη, τα οποία εισάγονται σε αύξουσα σειρά βαρών. Η δεύτερη ουρά, περιλαμβάνει όλους τους κόμβους που κατασκευάζονται πάλι με αύξουσα σειρά βαρών. Κάθε φορά που γίνεται εξαγωγή του ελάχιστου στοιχείου απαιτείται μόνο η σύγκριση των δύο μπροστινών στοιχείων και να εξαχθούν τα 2 μικρότερα. Κάτι που απαιτεί σταθερό χρόνο για συνολικά 4n-2 τέτοιες πράξεις. Ουσιαστικά δηλαδή αν εξαλείφεται η ανάγκη για αναζητήσεις εντός της ουράς αυτό σημαίνει ότι ο χρόνος υλοποίησης μπορεί να γίνει O(n).

Η προσέγγιση αυτή αν και ελπιδοφόρα ωστόσο η απαίτηση του να είναι ταξινομημένα τα στοιχεία εισόδου, είναι ένας πολύ σημαντικός περιορισμός. Επίσης, αν η δοθεί μη ταξινομημένη είσοδος και θελήσουμε να χρησιμοποιήσουμε την προσέγγιση του van Leeuwen πάλι δεν αλλάζει κάτι σημαντικά καθώς η ταξινόμηση των βαρών απαιτεί (n*logn) χρόνο κάτι που δεν δίνει πλεονέκτημα στην εκτέλεση του αλγορίθμου έναντι της προσέγγισης που έκανε ο Huffman.

2.4 – Υλοποίηση Σε-θέση (In-place)

Η Υλοποίηση Σε-Θέση βασίζεται στην προσέγγιση υλοποίησης του van Leeuwen. Το πιο σημαντικό της χαρακτηριστικό είναι ότι απαιτεί γραμμικό χρόνο για να εκτελεστεί ο υπολογισμός O(n). Ως όρισμα δίνεται ένας πίνακας W που περιέχει τα βάρη του συμβόλου και υπάρχουν 2 ουρές οι οποίες αναλύονται στην συνέχεια. Ο αλγόριθμος αποτελείται από 3 διαδοχικές φάσεις στις οποίες ανάλογα με την φάση ο πίνακας μετασχηματίζεται ανάλογα. Στην πρώτη φάση όλα τα αντικείμενα που υπάρχουν στην

ουρά των βαρών W και είναι ταξινομημένα κατά φθίνουσα σειρά καθώς επίσης και τα βάρη των εσωτερικών κόμβων που ανήκουν σε άλλη ουρά (επίσης ταξινομημένα κατά φθίνουσα σειρά) συνδυάζονται σε ζεύγη ανάλογα με το ποιος έχει τα μικρότερα βάρη. Όταν ξεκινά ο αλγόριθμος η ουρά που περιέχει εσωτερικούς κόμβους είναι άδεια και επομένως συνδυάζονται μόνο τα φύλλα. Επομένως στην πρώτη φάση, εντοπίζονται οι 2 κόμβοι που θα συγχωνευθούν.

Η δεύτερη φάση του αλγορίθμου αναστρέφει το δέντρο από τη ρίζα προς τα φύλλα μετατρέποντας τον πίνακα τον δεικτών σε πίνακα βαθών εσωτερικών κόμβων. Για να εκτελεστεί η δεύτερη φάση είναι προφανές ότι η πρώτη φάση πρέπει να έχει ολοκληρωθεί. Η Τρίτη φάση ως κύριο στόχο έχει την επεξεργασία του πίνακα που προέκυψε από την δεύτερη φάση , μετατρέποντας τα βάθη στους εσωτερικούς κόμβους σε βάθη φύλλων. Όσο τα επίπεδα του δέντρου επεξεργάζονται κάποιες κενές θέσεις που πιθανόν να έχει το δέντρο χρησιμοποιούνται για να κρατήσουν τον απαιτούμενο αριθμό εσωτερικών κόμβων ενώ τα υπόλοιπα στοιχεία, πρέπει υποχρεωτικά να είναι φύλλα. Ο αριθμός των διαθέσιμων στοιχείων στο επόμενο βάθος είναι ο διπλάσιος από τον αριθμό στοιχείων στο τωρινό βάθος.

Μόλις ολοκληρωθεί η Τρίτη φάση κάθε αυθεντικό βάρος συμβόλου στον πίνακα στοιχείων W έχει αντικατασταθεί από το μήκος κωδικολέξης του κώδικα Huffman. Αυτό που είναι αξιοσημείωτο, είναι πως δεν υπάρχει ανάγκη για δέντρα, δείκτες, σωρούς ή δυναμική μνήμη για την υλοποίηση που παρουσιάζει το paper.

2.5 – Ανάθεση Κωδικολέξεων

Ορίζεται το L ως το μέγιστο μήκος κωδικολέξης που απαιτείται. Σε αυτή την περίπτωση όταν τα βάρη είναι μη αύξοντα, έχουμε $L = l_{n-2} = l_{n-1}$. Μια κωδικολέξη μήκους li μπορεί και να γίνει αντιληπτή ως είτε ένας ακέραιος με μήκος li bits. Οι πρώτοι L bit ακέραιοι που αντιστοιχούν στο πιο συχνό σύμβολο είναι πάντα μηδέν επομένως ο i+1 bit ακέραιος υπολογίζεται προσθέτοντας 2^{L-li} στον L-bit ακέραιο που συσχετίζεται με το i-οστό σύμβολο. Μόλις το σύνολο των L-bit ακεραίων έχει υπολογιστεί, οι αντίστοιχες li-bit τιμές βρίσκονται, λαμβάνοντας τα πρώτα li bits του L bit ακεραίου. Αυτοί οι li-bit ακέραιοι είναι ακριβώς τα bitstrings που ανατίθενται στην στήλη που καλείται κωδικολέξη. Για κωδικοποίηση ενός στιγμιότυπου του i-οστου συμβόλου τα li χαμηλής τάξης bits, της i-οστής τιμής από την στήλη “ακέραιος με li bits” εξάγονται ως εξωτερικός buffer. Οι κωδικολέξεις ανατίθενται με αυτόν τον καθορισμένο τρόπο που σημαίνει ότι τα βάθη φύλλων, οι προσδιοριστές συμβόλων και οι ίδιες οι κωδικολέξεις (ως L-bit ακέραιοι) είναι με την ίδια σειρά. Το αποτέλεσμα ονομάζεται κανονικός κώδικας (canonical code). Ο κανονικός κώδικας μπορεί να εξαχθεί από τον κώδικα Huffman με εναλλαγή εσωτερικών κόμβων-παιδιών. Ωστόσο,

αυτή η αντιμετάθεση δεν είναι πάντα επιτεύξιμη. Οι εσωτερικοί κόμβοι στα 2 δέντρα έχουν διαφορετικά βάρη και δεν υπάρχει ακολουθία από εναλλαγές μεταξύ δεξιών και αριστερών εσωτερικών κόμβων, ακόμα και αν οι δύο κώδικες έχουν το ίδιο κόστος. Στη συνέχεια, ο συγγραφέας τονίζει την διαφορά που έχουν οι κώδικες ελάχιστου πλεονασμού και οι κώδικες Huffman. Πιο συγκεκριμένα, η εφαρμογή ενός κώδικα Huffman θα οδηγήσει πάντα σε κώδικα ελάχιστου πλεονασμού. Ωστόσο ένας κώδικας ελάχιστου πλεονασμού δεν προκύπτει μόνο από Κώδικα Huffman. Η αυστηρά ιεραρχημένη οργάνωση είναι πάντα επιθυμητή και αν ο ορισμός ενός κώδικα είναι αυτή του συνόλου από μήκη κωδικολέξεων τότε η διαφοροποίηση μεταξύ κωδίκων ελάχιστου πλεονασμού και Huffman είναι ασήμαντη.

2.6 – Βελτιστότητα

Ο κώδικας Huffman οδηγεί σε κώδικας ελάχιστου πλεονασμού. Για να αποδειχθεί αυτή η πρόταση απαιτούνται 2 βήματα σύμφωνα με το paper. Το πρώτο είναι η απόδειξη της ιδιότητας sibling (sibling property). Για την ιδιότητα αυτή αυτό που αναφέρει είναι το εξής: Κάθε κόμβος εκτός ρίζας έχει τουλάχιστον έναν αδερφό και οι κόμβοι μπορούν να τοποθετηθούν με μη αύξοντα βάρη με κάθε κόμβο να είναι δίπλα στα αδέρφια του και έναν κοινό πατέρα. Έχοντας αποδείξει την sibling property το επόμενο βήμα για να ολοκληρωθεί η απόδειξη είναι να επιβεβαιωθεί ότι ενώνοντας τα δύο αυτά σύμβολα, σε έναν ενοποιημένο κόμβο με βάρος να είναι το άθροισμα των δύο επιμέρους βαρών, τότε κατασκευάζοντας έναν κώδικα ελάχιστου πλεονασμού για το μειωμένο κατά ένα σύνολο συμβόλων, έπειτα επεκτείνοντας ξανά το σύμβολο στα δύο επιμέρους στοιχεία του δίνει πάλι έναν κώδικα ελάχιστου μήκους για το αυθεντικό σύνολο συμβόλων. Η επαγωγική αρχή στην συνέχεια μπορεί να αποδείξει το ίδιο για οποιοδήποτε πλήθος και σύνολο συμβόλων με το πλήθος του n>=2. Στην συνέχεια εντός του paper παρατίθεται η απόδειξη…

2.7– Αποτελεσματικότητα Συμπίεσης

Αρχικά, ο συγγραφέας παρουσιάζει τους τύπους υπολογισμού της εντροπίας H(W) και της σχετικής απώλειας αποτελεσματικότητας E(W,T). Για την εντροπία ο τύπος είναι $H(W) = -\Sigma wi*\log_2(wi/m)$ για $0 < i < n-1$ ενώ για την σχετική απώλεια αποτελεσματικότητας $E(W,T) = (C(W,T) – H(W)) / H(W)$. Ο παράγοντας $-\log_2(wi/m)$ για $0 < i < n-1$ καλείται κόστος εντροπίας και μπορεί να αντιπροσωπεύει διάφορους παράγοντες. Για παράδειγμα μπορεί να είναι πιθανότητα ή συχνότητα εμφάνισης συμβόλων και για m=1, τότε η εντροπία μετριέται σε bits/σύμβολο και αντιπροσωπεύει τον αναμενόμενο αριθμό συμβόλων εξόδου που παράγονται για κάθε σύμβολο πηγής.

Παράλληλα, αν και δεν αναφέρεται μέσα στο paper η εντροπία εκφράζει και ένα ελάχιστο όριο κωδικοποίησης της πηγής για την οποία δεν θα προκύψουν σφάλματα κατά την αποσυμπίεση (Θεώρημα Κωδικοποίησης Πηγής ή αλλιώς 1ο Θεώρημα Shannon). Η σχετική απώλεια αποτελεσματικότητας στο μηδέν, δείχνει ότι τα σύμβολα που κωδικοποιούνται στα κόστη εντροπίας τους, ενώ τιμές μεγαλύτερες του 0, δείχνουν ότι ο κωδικοποιητής εναπόκειται σε απώλειες κατά την συμπίεση. Στη συνέχεια παρουσιάζεται ένα παράδειγμα κατά το οποίο ενώ ο κώδικας είναι ελάχιστου πλεονασμού λόγω μεγάλης σχετικής απώλειας αποτελεσματικότητας ο κώδικας δεν είναι καλός. Τέλος δίνονται όρια στην αποτελεσματικότητα των Κωδίκων Huffman τα οποία για ένα σύνολο από βάρη W για n σύμβολα τα οποία αθροίζονται στο m = Σwi για 0< i < n-1 και ελάχιστου πλεονασμού κώδικας T οι οποίοι χρησιμοποιούνται στον τύπο:

*C(W,T) – H(W) <= w0 + 0,086*m όταν w0 < m/2*

C(W,T) – H(W) <= w0 όταν w0 >= m/2

Οι δύο σχέσεις χρησιμοποιούνται για εύρεση ενός άνω φράγματος για την σχετική απώλεια αποτελεσματικότητας.

2.8 – Μη δυαδικά Αλφάβητα Εξόδου

Σε αυτή την παράγραφο αναφέρεται η περίπτωση που το αλφάβητο εξόδου δεν είναι δυαδικό αλλά r-αδικό με r>2. Ο Huffman είχε σημειώσει ότι αν κάθε εσωτερικός κόμβος έχει r παιδιά τότε το τελικό δέντρο θα έχει k(r-1) + 1 φύλλα για κάποιο ακέραιο k. Ακόμα αν μια είσοδος από n βάρη W δίνεται τότε μια επαυξημένη είσοδος W’ μήκους n’ = (r-1) ⌈ (n-1) / (r-1) ⌉ + 1δημιουργείται και επεκτείνεται με την εισαγωγή n’-n συμβόλων βάρους w_{n+1}’ =…=$w_{n'-1}$’ = 0 με τα μηδενικά βάρη να είναι επιτρεπτά σε αυτή την περίπτωση. Ο Αλγόριθμος Huffman ή η προσεγγίσεις που παρουσιάστηκαν στην συνέχεια μπορούν να εφαρμοστούν με μοναδική αλλαγή την πρόσθεση r κόμβων σε έναν εσωτερικό κόμβο και όχι 2 όπως είχαμε για δυαδικό αλφάβητο εισόδου. Η ανισότητα Kraft-McMillan για αυτή την περίπτωση δίνεται από τον τύπο: K(T) = Σr^{-li} <= 1 για 0 < i < n-1 με την ισότητα να είναι πιθανή μόνο αν n=n’ και το n είναι ήδη μεγαλύτερο από πολλαπλάσιο του r-1. Ομοίως, για το κόστος εντροπίας η βάση του λογαρίθμου από 2 αλλάζει σε r.

Μια ενδιαφέρουσα σκέψη κατά τον συγγραφέα θα ήταν η επιλογή r = 2^8 = 256 συμβόλων που αντιστοιχεί σε ένα byte ως σύμβολο εξόδου. Για μεγάλα αλφάβητα όπου οι συχνότητες των συμβόλων είναι πολύ μικρές η σχετική απώλεια αποτελεσματικότητας θα είναι μικρή και η δυνατότητα για λήψη bytes στην αποκωδικοποίηση αντί για bits δίνει ένα τεράστιο πλεονέκτημα.

3 - Κωδικοποίηση και Αποκωδικοποίηση Κωδικών Ελάχιστου Πλεονασμού

Σε αυτή την ενότητα υποτίθεται ότι ο κώδικας που εφαρμόζεται είναι κανονικός στον οποίο το σύνολο των κωδικολέξεων έχει ταξινομηθεί λεξικολογικά. Το paper εξετάζει την κωδικοποίηση την αποκωδικοποίηση και διερευνά το ερώτημα του πως ο κωδικοποιητής να επικοινωνήσει τον κώδικα T στον αποκωδικοποιητή αποδοτικά.

3.1 – Κωδικοποίηση Κανονικών Κωδικών

Η έως τώρα περιγραφή του κώδικα Huffman κοστίζει με πλήθος τρόπων σύμφωνα με τον συγγραφέα: Ο ρητός χειρισμός ενός δέντρου απαιτεί σημαντικό χώρο μνήμης. Επίσης, διασχίζοντας ένα δείκτη σε μια μεγάλη δομή δεδομένων για κάθε bit που δημιουργείται, μπορεί να περιλαμβάνει cache αποτυχίες και το γράψιμο ένα bit κάθε φορά σε ένα αρχείο εξόδου είναι αργό. Για επίλυση των παραπάνω προβλημάτων θα οριστούν συγκεκριμένες έννοιες. Αρχικά το μήκος του κώδικολέξης θα μπει σε έναν πίνακα με όνομα array_len[] και θα μπορεί να προσδιοριστεί από έναν δείκτη συμβόλου. Ένας ακόμα πίνακας με όνομα li-bit ακέραιος αποθηκεύεται σε πίνακα με όνομα code_word[]. Στον συγκεκριμένο πίνακα θα τοποθετηθεί ως ακέραιος η τιμή που έχει η κωδικολέξη. Για κωδικοποίηση ενός συμβόλου s όπου 0 <= s < n αυτό που χρειάζεται είναι η εξαγωγή του code_len[s] χαμηλής τάξης bits του ακεραίου στο code_word[s]. Η συνάρτηση η οποία γράφει τα bits αυτά στο stream εξόδου καλείται putbits(code_word[s],l) όπου η putbits(val,count) γράφει τα χαμηλής τάξης bits από τον ακέραιο val στο stream εξόδου και συνήθως υλοποιείται με χρήση μάσκας χαμηλού επιπέδου και τελεστές ολίσθησης. Αυτή η διαδικασία εξαλείφει την ανάγκη για δέντρο και παράλληλα εξασφαλίζει ότι σε κάθε κύκλο δημιουργείται μια ολόκληρη κωδικολέξη αντί για ένα bit.

Ο αποθηκευτικός χώρος του code_len[] είναι σχετικά φθηνός – 1 byte ανά τιμή και επιτρέπει κωδικολέξεις έως L=255 bits κάτι που σχεδόν πάντα αρκεί. Ο πίνακας code_word[] όμως, ακόμα έχει την προοπτική να είναι ακριβός καθώς ακόμα και 32-bits ανά τιμή ίσως δεν αρκούν για κωδικολέξεις που συσχετίζονται με ένα μεγάλο αλφάβητο. Ωστόσο, ο πίνακας code_word[] μπορεί να αντικατασταθεί από 2 άλλους πίνακες με L+2 διευθύνσεις ο καθένας όπου όπως πριν L είναι το μέγεθος της μεγαλύτερης κωδικολέξης. Αν ο χώρος της μνήμης την στιγμή της κωδικοποίησης (encode-time memory) είναι αρκετός μπορούν να εξαλειφθούν και τα n bytes που καταναλώνονται από τον code_len[] πίνακα αλλά με αντίτιμο ταχύτητα κωδικοποίησης. Μια από τις επιπτώσεις που είναι χρήσιμες της εστίασης σε κανονικούς κώδικες είναι

το γεγονός πως η εύρεση ενός συμβόλου και ο προσδιορισμός της τιμής του μήκους συμβόλου για κωδικοποίηση, μπορεί να φραχτεί για έγκυρα μήκη κωδικολέξεων.

Η τεχνικές κωδικοποίησης που αναφέρθηκαν υπόκεινται σε κάποιους περιορισμούς. Πιο συγκεκριμένα, το αλφάβητο πηγής πρέπει να έχει ταξινομηθεί, τα βάρη των συμβόλων να είναι μη αύξοντα και οι κανονικές κωδικολέξεις ελάχιστου πλεονασμού να είναι λεξικογραφικά ταξινομημένες. Σε κάποιες εφαρμογές αυτοί οι περιορισμοί πιθανόν να μην ικανοποιούνται και σε αυτή την περίπτωση πρέπει να δεσμευθεί ακόμα χώρος στη μνήμη ίσος με n λέξεις ο οποίος θα ανήκει σε ένα διάνυσμα αναδιατάξεων (permutation vector) το οποίο χαρτογραφεί προσδιοριστές πηγής σε ταξινομημένους αριθμούς συμβόλων και χρησιμοποιείται αυτό για να γίνει ο κώδικας κανονικός.

3.2 – Αποκωδικοποίηση Κανονικών Κωδικών

Οι περιορισμένες δομές κανονικών κωδικών σημαίνουν ότι είναι επίσης δυνατό, να αποφευχθούν οι ανεπάρκειες που σχετίζονται με την επεξεργασία bit-by-bit, με βάση το δέντρο κατά την αποκωδικοποίηση. Τώρα ακολουθείται η αντίστροφη διαδικασία για αποκωδικοποίηση. Επίσης, χρησιμοποιείται ένας buffer ο οποίος μπορεί ν αποθηκεύσει μέχρι L μήκος bits και υπάρχει για να αποθηκεύει τα κομμάτια της κωδικολέξης που λαμβάνει από το κανάλι για να παραλάβει την κωδικολέξη που μεταδίδεται και να αποφασίσει ποιο σύμβολο αντιστοιχεί σε αυτή την κωδικολέξη. Ο buffer είναι μεταβλητή. Στην συνέχεια παρατίθεται η διαδικασία αποκωδικοποίησης υλοποιημένη με ψευδοκώδικα. Αρχικά προσδιορίζεται ποιο είναι το μήκος της κωδικολέξης που λαμβάνεται με αναζήτησης στον πίνακα συμβόλων. Η διαδικασία αυτή κοστίζει O(l) χρόνο και είναι το πιο ακριβό βήμα σε χρόνο. Αν μπορεί να δοθεί παραπάνω μνήμη ο προσδιορισμός του l μπορεί να γίνει και με άλλους τρόπους όπως για παράδειγμα με άμεσο έλεγχο πίνακα. Αν 2^L bytes μνήμης επιτραπούν ένας πίνακας ο οποίος δεικτοδοτείται από τον buffer μπορεί να χρησιμοποιηθεί για αποθήκευση του μήκους της πρώτης κωδικολέξης που περιλαμβάνεται στον buffer. Το πρόβλημα που προκύπτει όμως είναι ότι το 2^L μπορεί να γίνει πολύ μεγαλύτερο του n κάτι που θα κάνει τον πίνακα ακριβό. Επιπρόσθετα, σε μεγάλους πίνακες το κόστος των αποτυχιών στην cache μπορεί να σημαίνει ότι μια γραμμική ή δυαδική αναζήτηση ίσως συμφέρει περισσότερο στην πράξη.

Οι τεχνικές αναζήτησης πίνακα μπορούν επίσης να συνδυαστούν και ο μερικός πίνακας που προκύπτει επιτάχυνε τη διαδικασία γραμμικής αναζήτησης, όντας ένα αποτελεσματικό υβρίδιο. Σε αυτή την πρόταση ένας πίνακας search_start[] με 2^t εισόδους δημιουργείται για $l_{min} <= t <= L$. Τα πρώτα t bits του buffer χρησιμοποιούνται για δεικτοδότηση σε αυτόν τον πίνακα με τις αποθηκευμένες τιμές να δείχνουν είτε το

σωστό μήκος l αν τα t bits αρκούν για σαφή προσδιορισμό κωδικολέξης ή την μικρότερη έγκυρη τιμή του l αν δεν αρκούν. Όπως και να έχει, η γραμμική αναζήτηση που προκύπτει χρησιμοποιείται για επιβεβαίωση της σωστής τιμής του l, σε σχέση με τα πλήρη περιεχόμενα του buffer. Επειδή οι μικρές κωδικολέξεις είναι εξ' ορισμού οι πιο συχνά χρησιμοποιούμενες ένα μεγάλο πλήθος από τα κωδικοποιημένα σύμβολα μπορεί να εντοπιστεί επακριβώς στον υβριδικό πίνακα αναζήτησης.

Ένας πίνακας αποκωδικοποίησης μπορεί να υπολογίσει κανονικούς κώδικες Huffman σε σχεδόν σταθερό χρόνο. Ανά σύμβολο εξόδου. Επίσης, αν μπορεί να διατεθεί σχετικά μεγάλος χώρος ή αν τα L και n bits μπορούν να περιοριστούν σε μικρές σχετικά τιμές η αποκωδικοποίηση μπορεί να είναι αποκλειστικά βασισμένη σε πίνακα. Ο λόγος για αυτό προκύπτει επειδή για n-1 εσωτερικούς κόμβους του δέντρου από τον κώδικα μπορεί να θεωρηθεί και ως κατάσταση σε ένα αυτόματο το οποίο στόχος τους είναι να αποκωδικοποιεί κωδικολέξεις, με κάθε κατάσταση να αντιστοιχεί σε μια πρόσφατη ιστορία από μη προσδιορισμένα bits. Με είσοδο k-bits για είσοδο θα προκύψει διαφορετικό σύνολο εξόδων.

Μεταξύ όλων των n-1 εσωτερικών κόμβων, το πλήρες σύνολο από $(n-1)2^k$ μεταβάσεων μπορεί να υπολογιστεί εκ των προτέρων και να αποθηκευτεί σε 2 δισδιάστατους πίνακες με το πολύ k σύμβολα να εκπέμπονται καθώς k-bit μπλοκ επεξεργάζεται. Η αποκωδικοποίηση τότε είναι απλώς μια αρχικοποίηση στην κατάσταση της ρίζας και διαδοχικών μεταβάσεων στις κατάλληλες καταστάσεις. Κάθε στοιχείο του πίνακα αποκωδικοποίησης αποτελείται από μια κατάσταση προορισμό, έναν μετρητή εξωτερικών συμβόλων και μια λίστα μέχρι k συμβόλων εξόδου. Ο πίνακας τότε απαιτεί 2^k (n-1) τέτοια στοιχεία κάτι που σημαίνει πως απαιτείται σημαντικός χώρος μνήμης. Επίσης, αν και μπορεί ο χώρος αυτός καθαυτός να μην είναι τεράστιος αλλά οι αποτυχίες που θα υπάρχουν στην cache θα έχουν σημαντική επίπτωση στην εκτέλεση. Για αυτό τον λόγο για να επιτευχθεί μείωση στην απαιτούμενη χωρητικότητα αλλά παράλληλα να αξιοποιηθούν κάποια πλεονεκτήματα δουλεύοντας με ομάδες k-bit, χρησιμοποιούνται υβριδικές μέθοδοι που συνδυάζουν την τεχνική για «αναζήτηση για l» με την προσέγγιση με ντετερμινιστικό αυτόματο που επίσης είναι πιθανές.

3.3 – Διαχείριση

Εκτός και αν ο κώδικας ελάχιστου πλεονασμού δεν έχει αναπτυχθεί εκ των προτέρων και στη συνέχεια να μεταγλωττιστεί σε προγράμματα κωδικοποίησης και αποκωδικοποίησης, ένα προανακρουσμένο στοιχείο (prelude component), επίσης πρέπει να συσχετίζεται με κάθε μήνυμα που μεταδίδεται, περιγράφοντας τις λεπτομέρειες του κώδικα που θα χρησιμοποιηθεί για το κύριο μέρος του μηνύματος.

Τα στοιχεία που απαιτούνται στο προανάκρουσμα περιλαμβάνουν το μήκος m του μηνύματος έτσι ώστε ο αποκωδικοποιητής να γνωρίζει πότε να σταματήσει να αποκωδικοποιεί, μια περιγραφή του μεγέθους n και την σύνθεση του αλφαβήτου πηγής αν δεν μπορεί να υποτεθεί εξ' ορισμού και το μήκος li της κωδικολέξης που συσχετίζεται με κάθε ένα από τα n σύμβολα πηγής που εμφανίζονται στο μήνυμα. Οι βαθμωτοί n, m έχουν μηδαμινό κόστος. Ωστόσο, αν το αλφάβητο πηγής είναι μέρος ενός μεγαλύτερου συνόλου τότε απαιτείται ένα διάνυσμα επιλογής υπό-αλφαβήτου (subalphabet selection vector) απαιτείται και ακόμα και αυτό ίσως έχει μεγαλύτερο κόστος. Ο πιο απλός τρόπος επιλογής είναι η διάθεση λίστας από n 4-byte ακεραίων, οι οποίοι περιέχουν τα σύμβολα που εμφανίζονται στο τωρινό μήνυμα. Επίσης πιθανό είναι να θεωρήσουμε του υπό-αλφάβητο ως καθορισμένο από ένα bit-διάνυσμα (bitvector) μήκους 2^{32} όπου 1 στην θέση u δείχνει ότι το σύμβολο είναι μέρος του υπό-αλφάβητου και έχει κώδικα που συνδέεται με αυτό.

Το τελευταίο στοιχείο που απαιτείται είναι ένα σύνολο από μήκη κωδικολέξεων T, τα οποία είναι πιο οικονομικό να μεταδοθούν σε σχέση με τα βάρη από τα οποία προέρχονται και επίσης πιο οικονομικά από τις πραγματικές κωδικολέξεις, αφού οι πραγματικές κωδικολέξεις μπορεί να εκτείνονται σε ένα μεγάλο εύρος τιμών. Τα μήκη κωδικολέξεων δίνονται με την ίδια σειρά που δίνονται και τα σύμβολα. Αν το μήνυμα είναι πολύ μεγάλο ή έχει άγνωστο μήκος μπορεί να διασπαστεί σε μεγάλα προκαθορισμένου-μήκους μπλοκ και το προανάκρουσμα (prelude) κατασκευάζεται για κάθε μπλοκ. Το κόστος των πολλαπλών προανακρουσμάτων πρέπει να μπορεί να γίνει αποδεκτό. Στην πραγματικότητα όμως, εξαιτίας της χρήση τοπικών κωδίκων και πιθανώς διαφορετικών υπό-αλφάβητων εντός κάθε μπλοκ οδηγούν στο συμπέρασμα ότι πετυχαίνουν καλύτερη συμπίεση από το να υπάρχει ένα μόνο προανάκρουσμα και ένα καθολικού-μηνύματος κατασκευασμένος κώδικας.

4 – Ειδικές Περιπτώσεις Κωδικών Ελάχιστου Πλεονασμού

Παρακάτω το paper αναλύει παραλλαγές του προβλήματος δημιουργίας κώδικα ελάχιστου πλεονασμού που συνήθως έχουν παραπάνω περιορισμούς ή απαιτήσεις.

4.1 – Περιορισμένου Μήκους

Ο επιθυμητός κώδικας πρέπει να έχει κωδικολέξεις με μήκη $l_i <= L<L_{HUF}$ όπου L_{HUF} είναι το μέγιστο μήκος κωδικολέξης στον κώδικα Huffman και είναι άγνωστο και L είναι το μέγιστο μήκος κωδικολέξης που έχει κατασκευαστεί από τον κώδικα. Επίσης ο κώδικας πρέπει να έχει το μικρότερο δυνατό κόστος. Η υλοποίηση που παρουσιάζει το paper,

βασίζεται στην ιδέα ότι η κατασκευή πακέτων γίνεται όπως στον Huffman αλλά κάθε στοιχείο δεν μπορεί να συμμετέχει σε παραπάνω από L συνενώσεις. Μια λίστα από υπό-δένδρα βάθους το πολύ δύο παράγεται και δείχνει τρόπους με τους οποίους το άθροισμα Kraft ίσως μειωθεί κατά 2^{-L+1} kai και αντιστοιχεί στη μετακίνηση υπό-δένδρου από βάθος L-2 σε βάθος L-1. Για να γίνει αυτό τα μικρότερα πακέτα συνενώνονται από τα μικρά προς τα μεγάλα. Η διαδικασία παράγει [n/2] πακέτα με τα βάρη τους να είναι τα αθροίσματα των μικρότερων επιμέρους πακέτων τους, κάθε ένα εκ των οποίων αντιπροσωπεύει έναν εσωτερικό κόμβο. Η λίστα των πακέτων επεκτείνεται καθώς ενώνεται με άλλο αντίγραφο του W[], παράγοντας την πλήρη λίστα πακέτων με n + [n/2] υπό-δέντρα με βάθος έως 2 πάλι ταξινομημένα κατά κόστος. Για να φτάσουμε σε εφικτό κώδικα, το άθροισμα Kraft πρέπει να μειωθεί κατά n-1 σε σχέση με την αρχική κατάσταση. Επομένως, επιλέγοντας το μικρότερο βάρος, 2n-2 στοιχεία στα πακέτα κάνουν ένα σύνολο από στοιχεία που ονομάζονται λύση- όταν κάθε στοιχείο υποβιβάζεται κατά ένα επίπεδο, οδηγεί στον απαιτούμενο κωδικό. Αλλά για να επιτευχθούν αυτοί οι υποβιβασμοί, καθένα από τα σύνθετα στοιχεία της λύσης [L] πρέπει να προωθήσει τις υποτιμήσεις του στο επόμενο επίπεδο προς τα κάτω για να σχηματίσει την καθορισμένη λύση [L - 1].

Η υλοποίηση του συγκεκριμένου αλγορίθμου αν και είναι υλοποίηση που αντιστοιχεί σε Huffman αφήνει αρκετές λεπτομέρειες υλοποίησης κενές. Ο χρόνος εκτέλεσης και ο χώρος που απαιτείται είναι O(nL). Ωστόσο, υπάρχουν και πιο πολύπλοκες υλοποιήσεις που αναφέρονται μέσα στο κείμενο για τις οποίες ο χώρος γίνεται O(n) και προστίθεται ένας σταθερός παράγοντας στον χρόνο εκτέλεσης. Τέλος αναφέρονται υλοποιήσεις που θέλουν $O(n(L - \log_2 n + 1))$ χρόνο και $O(L^2)$ χώρο καθώς επίσης και υλοποίηση με $O(n(L_{HUF} - L + 1))$ χρόνο αντίστοιχα. Η σχετική αποτελεσματικότητα έχουν ελάχιστα χειρότερη απώλεια συμπίεσης από τον Κώδικα Huffman.

4.2 – Δυναμική Κωδικοποίηση Huffman

Το αλφάβητο που θα χρησιμοποιηθεί είναι γνωστό αλλά οι συχνότητες εμφάνισης, όχι. Μια επιλογή είναι η σάρωση του μηνύματος και ο προσδιορισμός τον συχνοτήτων συμβόλου και εν συνεχεία η κατασκευή του κώδικα. Μια άλλη επιλογή είναι να χρησιμοποιηθούν προσαρμοστικής πιθανοτικές προσεγγίσεις. Ξεκινάμε με μια εκτίμηση όπως για παράδειγμα πως κάθε σύμβολο είναι ισοπίθανο. Μετά από την επεξεργασία κάθε συμβόλου τόσο ο κωδικοποιητής όσο και ο αποκωδικοποιητής ρυθμίζουν την εκτίμησή τους για το σύμβολο κατάλληλα. Το ίδιο συμβαίνει και για τα υπόλοιπα σύμβολα. Μόλις μεταδοθεί και κωδικοποιηθεί και το τελευταίο σύμβολο ο κωδικοποιητής και ο αποκωδικοποιητής γνωρίζουν επακριβώς το σύνολο και τις συχνότητες εμφάνισης των συμβόλων επομένως μπορούν να κατασκευάσουν τον

κώδικα. Η κοινή ιδέα για την ανάπτυξη κάθε μηχανισμού εντοπισμού των συχνοτήτων είναι πως η λίστα με τα φύλλα και τους εσωτερικούς κόμβους καθώς και τα βάρη αυτών διατηρούνται με ταξινομημένη μη αύξουσα σειρά καθώς επίσης και πως η λίστα ενημερώνεται με ολισθήσεις φύλλων προς νέες θέσεις στο δέντρο και στη συνέχεια εκτιμώντας τι επίδραση θα έχει αυτή η αλλαγή στην ακολουθία συνένωσης που οδήγησε στο δέντρο Huffman. Αν ένα φύλλο αλλάξει θέση με έναν άλλο κόμβο τότε πρέπει να αλλάξουν θέσεις στο δέντρο Huffman. Όλες οι αλλαγές που απαιτούνται απαιτούν γραμμικό χρόνο. Οι προσεγγίσεις για δυναμική κωδικοποίηση Huffman είναι αργές και απαιτούν αποθήκευση αρκετών τιμών για κάθε σύμβολο από το αλφάβητο. Τέλος αναφέρεται πως δεν υπάρχουν συστήματα συμπίεσης γενικού σκοπού που χρησιμοποιούν δυναμική κωδικοποίηση.

4.3 – Προσαρμοστικοί Αλγόριθμοι

Προσαρμοστικός αλγόριθμος είναι αυτός που εμπίπτει σε κάποιον περιορισμό λόγω της φύσης του προβλήματος. Σε αυτή την περίπτωση χρησιμοποιείται μια δεύτερη ποσότητα για προσδιορισμό της πολυπλοκότητας κάθε προβλήματος. Το paper αναφέρει πως εισάγεται μια νέα έννοια που καλείται run. Σε πολλά προβλήματα με μεγάλα αλφάβητα υπάρχουν αρκετά σύμβολα με παρόμοιες μικρές συχνότητες. Η σύμβαση $r_i(w_i)$ που ορίζεται στην συνέχεια έχει ως στόχο να δώσει την δεύτερη ποσότητα r_i που δείχνε πόσες φορές εμφανίζεται το βάρος w_i. Αν το στιγμιότυπο κωδικοποίησης παρουσιάζεται ως ταξινομημένη λίστα συχνοτήτων τότε απαιτείται O(n) χρόνος για να μετατραπεί σε μορφή βασισμένη σε runs άρα δεν υπάρχει όφελος. Αν η είσοδος αποτελείται από n σύμβολα με r διακριτά βάρη συμβόλων ο υπολογισμός του κώδικα απαιτεί O(r(1+log(n/r))) χρόνο που δεν είναι ποτέ χειρότερος του O(n). Επίσης, το πρόβλημα μπορεί να αντιμετωπιστεί θεωρώντας ως δεύτερη ποσότητα το μέγιστο μήκος κωδικολέξης L και τότε για ταξινομημένες εισόδους προκύπτουν γραμμικοί χρόνοι O(n) και O(L) απαίτηση σε επιπλέον χώρο.

Άλλη προσέγγιση παίρνει ως δεύτερη ποσότητα τον αριθμό των εναλλαγών που προκαλούνται από την ακολουθία των βαρών εισόδου, όπου κάθε εναλλαγή είναι η μετάβαση από την κατανάλωση των φύλλων στην κατανάλωση εσωτερικών κόμβων κατά την δημιουργία του δέντρου. Αν ο αριθμός των εναλλαγών στο W που καλούνται a(W) είναι υψηλός, το δέντρο που προκύπτει είναι σχετικά αβαθές και εύκολο να δημιουργηθεί. Βάσει αυτής της προσέγγισης μπορεί να δημιουργηθεί μηχανισμός για υπολογισμό του κώδικα Huffman, από μη διατεταγμένα W,σε O(n(1 + loga(W))) χρόνο. Ο χρόνος αυτός είναι βέλτιστα προσαρμοστικός σύμφωνα με απόδειξη του συγγραφέα που τον πρότεινε. Ωστόσο, άποψη του συγγραφέα είναι ότι μπορεί να υπάρξει και μικρότερος χρόνος $O(n(1 + \log L_{diff}))$ χρόνος.

5 - Άλλες Τεχνικές Κωδικοποίησης Κοντά στην Εντροπία

Σε αυτό το κεφάλαιο παρουσιάζονται δύο τεχνικές που συμπιέζουν ακόμα πιο κοντά στην εντροπία από ότι ο κώδικας Huffman. Ο λόγος για αυτό έχει να κάνει με το γεγονός πως χρησιμοποιούν μεταβλητές κατάστασης, οι οποίες μεταφέρουν πληροφορία προς τα εμπρός από το ένα κωδικοποιημένο σύμβολο προς το επόμενο. Οι μέθοδοι διαφέρουν μεταξύ τους ως προς τι αντιπροσωπεύουν οι μεταβλητές κατάστασής τους και πως τα σύμβολα καναλιού εξάγονται από τις μεταβλητές κατάστασης και μεταδίδονται στο stream εξόδου κατά αύξοντα τρόπο. Πιο συγκεκριμένα, στον αριθμητικό κωδικοποιητή μια κλασματική τιμή κατάστασης (fractional state value) γίνεται όλο και πιο ακριβής καθώς τα σύμβολα κωδικοποιούνται με όλο και περισσότερα bits να δεσμεύονται στις τελικές τιμές τους ενώ σε έναν ANS κωδικοποιητή μια ακέραια τιμή κατάστασης μεγαλώνει όλο και περισσότερο καθώς τα σύμβολα κωδικοποιούνται.

5.1 – Αριθμητική Κωδικοποίηση

Η βασική ιδέα που διέπει όλες τις υλοποιήσεις αριθμητικής κωδικοποίησης είναι αυτή της κατάστασης κωδικοποίησης η οποία μπορεί να περιγραφεί από ένα ζεύγος τιμών <lower,range> όπου και οι δύο είναι τιμές μεταξύ μηδέν και ένα. Μαζί κατασκευάζουν μια τιμή που αντιπροσωπεύει το μήνυμα εισόδου και είναι επίσης στο εύρος [0,1). Επίσης, ισχύει: lower <= value < lower + range. Πριν την κωδικοποίηση lower=0 και range=1 με την value να μπορεί να πάρει είτε 0, είτε 1, είτε κάποια ενδιάμεση τιμή. Καθώς κάθε σύμβολο s επεξεργάζεται, το εύρος που αντιστοιχεί στην τρέχουσα κατάσταση αντικαθίσταται από ένα πιο στενό εύρος τιμών που φράζει περισσότερο την value. Η σμίκρυνση του εύρους διεξάγεται σε αυστηρά αριθμητικό ποσοστό εντός του εύρους [0,1) που συσχετίζεται με το s, βασίζεται στο συσσωρευμένο βάρος W όλων των συμβόλων. Μόλις όλα τα σύμβολα έχουν επεξεργαστεί, η πιο μικρή αναμφίβολη δυαδική τιμή που υπάρχει εντός του τελευταίου εύρους εντοπίζεται και αποστέλλεται στον αποκωδικοποιητή. Το τελευταίο εύρος [lower, lower + range) συλλαμβάνει το μήνυμα πλήρως και μπορεί να αντιπροσωπευθεί από οποιαδήποτε τιμή value εντός αυτού. Μια τέτοια τιμή value απαιτεί το πολύ $O(-\log_2 \text{range}) + 2$ bits για να περιγραφεί.

Για αποκωδικοποίηση του μηνύματος που υπάρχει στην value,ο αποκωδικοποιητής χρησιμοποιεί base[] για να καθορίσει έναν προσδιοριστή συμβόλου s, του οποίου η εκχώρηση πιθανότητας στο εύρος [0,1) διατρέχει τη σχέση που έχει η value με το τρέχων εύρος [lower, lower + range). Μόλις τελειώσει η διαδικασία υπολογισμού του εύρους για τον κωδικοποιητή η ίδια ακριβώς διαδικασία εκτυλίσσεται και στον

αποκωδικοποιητή. Στη συνέχεια, τονίζεται ότι πρακτικές υλοποιήσεις του αριθμητικού κώδικα αποφεύγουν ασαφή ακρίβεια και περιλαμβάνουν μια διαδικασία επανακανονικοποίησης που εκπέμπει bit ή byte και διπλασιάζει το range ή το πολλαπλασιάζει με 256, όταν ένα bit μπορεί να καθοριστεί σωστά χωρίς παρερμηνείες.

Συγκρίνοντας κωδικοποίηση Huffman και αριθμητική κωδικοποίηση, η αριθμητική κωδικοποίηση διαχειρίζεται με ακρίβεια τις διαστρεβλωμένες κατανομές πιθανότητας και υποστηρίζει εύκολα προσαρμοστικά μοντέλα, στα οποία οι εκτιμήσεις πιθανότητας προσαρμόζονται κατά την επεξεργασία του μηνύματος και στα οποία μπορεί να υπάρχουν πολλαπλά περιβάλλοντα, με την επιλογή ως προς το ποια θα χρησιμοποιηθεί για οποιοδήποτε σύμβολο που αποφασίζεται από την ακολουθία προηγούμενων συμβόλων. Τα πλεονεκτήματα αυτά έχουν χρησιμοποιηθεί κυρίως για PPM (Partial Prediction Matching) συστήματα συμπίεσης. Ένα μειονέκτημα είναι ότι οι πολλαπλασιασμοί και οι διαιρέσεις οδηγούν σε πιο αργή αποκωδικοποίηση.

5.2 – Ασυμμετρικά Αριθμητικά Συστήματα

Τα Ασυμμετρικά Αριθμητικά Συστήματα (Asymmetric Numeral Systems - ANS) αντιμετωπίζουν το μήνυμα ως ολόκληρο και ένας απλός αριθμός αναδύεται στο τέλος της εισόδου και θεωρείται αντιπροσωπευτικός όλου του μηνύματος εισόδου και επίσης όπως στην αριθμητική κωδικοποίηση μια κατάσταση επίσης διατηρείται. Στην περίπτωση της ANS κωδικοποίησης σαν απλός αριθμός. Η αρχική τιμή της κατάστασης είναι 0 και αντιπροσωπεύει την άδεια αλληλουχία χαρακτήρων. Από αυτό το αρχικό σημείο κάθε σύμβολο εισόδου s χρησιμοποιείται για να μεταβεί από την τωρινή κατάσταση σε μια νέα κατάσταση βάσει ενός πίνακα καταστάσεων που έχουν ήδη προ-υπολογιστεί.

Η αποκωδικοποίηση λειτουργεί με τον αντίθετο τρόπο: Η τελευταία τιμή για την κατάσταση εντοπίζεται με τον αριθμό κατάστασης να δείχνει το σύμβολο που θα στοιβαχθεί (τα σύμβολα αποφασίζονται σε αντίθετη σειρά κατά την αποκωδικοποίηση) και ο αριθμός στήλης καταδεικνύει την προηγούμενη κατάσταση που θα επανέλθει. Παρά τις φαινομενικά απρόβλεπτες τιμές των τιμών που εμπεριέχονται η τεχνική που περιεγράφηκε δίνει μια ντετερμινιστική 1-προς-1 χαρτογράφηση ακολουθιών από χαρακτήρες (strings) σε ακέραιες τιμές. Οι απλοί υπολογισμοί αρκούν για τον υπολογισμό του μετασχηματισμού κωδικοποίησης και αποκωδικοποίησης, δεδομένου μόνο μιας συστοιχίας n στοιχείων, με ενσωματωμένα σύμβολα βάρη W[] και μια προ-υπολογισμένη συσσωρευτικό άθροισμα με όνομα base[] του ίδιου μεγέθους. Στον αποκωδικοποιητή, θεωρείται μια επιπλέον συστοιχία, το αντίστροφο στοιχείο m της base[]. Σε αυτόν τον πίνακα, το σύμβολο [r] είναι το σύμβολο προέλευσης που σχετίζεται με την πρώτη αντιστάθμιση σε κάθε έναν από τους κύκλους ANS. Ένας πολύ

σημαντικός παράγοντας που επιτρέπει γρήγορη αποκωδικοποίηση είναι η ικανότητα ρύθμισης του m έτσι ώστε το m’ στην αποκωδικοποίηση να είναι δύναμη του2. Αν αυτό επιτευχθεί τότε η διαδικασία της ανάποδης χαρτογράφησης μπορεί να εφαρμοστεί με ολισθήσεις και πράξεις με χρήση μάσκας. Το paper στην συνέχεια αναπτύσσοντας περαιτέρω τον ισχυρισμό καταλήγει στο ότι ο κώδικας Huffman είναι μια ειδική περίπτωση ANS κώδικα.

Ο περιορισμός του εύρους της κατάστασης στον κωδικοποιητή, επιτυγχάνεται προβλέποντας το τι θα συμβεί σε κάθε επόμενο βήμα κωδικοποίησης και αν η μεταβλητή next_state είναι μεγαλύτερη του άνω ορίου του εύρους, η κατάσταση μειώνεται πριν την διαδικασία κωδικοποίησης. Τέλος, ένα ακόμα σημαντικό στοιχείο έχει να κάνει με την Csταθερά που δίνει την δυνατότητα επιλογής μεταξύ πιστότητας του αριθμητικού στο εντροπικό κόστος και του πλήθους των bits που χρησιμοποιούνται στον υπολογισμό και επομένως το μήκος των πινάκων που χρησιμοποιούνται σε υλοποίηση που χρησιμοποιεί πίνακες. Όταν C=1 η αποτελεσματικότητα συμπίεσης διακινδυνεύει από σφάλματα λόγω στρογγυλοποιήσεων καθώς η κατάσταση εντοπίζεται αλλά είναι πιθανό και πάλι να είναι καλύτερη από έναν κώδικα Huffman καθώς συμπίεση κοντά στο κόστος εντροπίας συμβαίνει όταν το C είναι μεγαλύτερο.

6 - Συμπέρασμα: Ο κώδικας Huffman έχει αφανιστεί;

Η μεγάλη αδυναμία του κώδικα Huffman είναι η ανικανότητα του να προσεγγίσει το κόστος εντροπίας όταν η πιθανοτική κατανομή κυριαρχείται από το πιο συνηθισμένο σύμβολο. Όσον αφορά την ταχύτητα η αριθμητική κωδικοποίηση απαιτεί περισσότερους υπολογισμούς από Huffman και ANS και αν υλοποιηθεί προσαρμοστικά μια από τις βασικές περιοχές εφαρμογής η ταχύτητα μειώνεται. Επίσης, η αποκωδικοποίηση Huffman είναι πιο γρήγορη σε σχέση με την αποκωδικοποίηση ANS ενώ η αριθμητική αποκωδικοποίηση είναι η πιο αργή και από τις τρεις μεθόδους. Η ANS προσέγγιση δεν μπορεί να χρησιμοποιηθεί όπου υπάρχουν προσαρμοστικές πιθανοτικές εκτιμήσεις καθώς τα σύμβολα που αναπαράγονται στον αποκωδικοποιητή είναι με την ανάποδη φορά που κωδικοποιούνται από τον κωδικοποιητή. Επίσης, το υψηλό κόστος του δυναμικού κώδικα Huffman επομένως σημαίνει ότι πολύπλοκα μοντέλα συνδυάζονται με αριθμητική κωδικοποίηση. Το paper αναφέρει και ορισμένες λεπτομέρειες ακόμα συγκρίνοντας τις 3 μεθόδους κωδικοποίησης και καταλήγει στο τελικό συμπέρασμα πως η απήχηση του Huffman πραγματικά συστήματα έχει μειωθεί αρκετά ενώ η πιο δημοφιλείς μέθοδοι κωδικοποίησης μοιράζονται ως προς τις προτιμήσεις για κωδικοποίηση ανάλογα με την ταχύτητα κωδικοποίησης και αποκωδικοποίησης αλλά και τον τύπο των μηνυμάτων καθώς επίσης και την περιγραφή του συστήματος κωδικοποίησης που πρέπει να κατασκευαστεί.

Paper 2: Algorithms for Infinite Huffman-Codes

Παρουσιάζεται η πρώτη αλγοριθμική προσέγγιση της δημιουργίας προθεματικού κώδικα ελαχίστου κόστους για (γενικευμένες) γεωμετρικές κατανομές. Μιλώντας για άπειρες πηγές, ο λόγος που ο βασικός αλγόριθμος Huffman δεν μπορεί να τροποποιηθεί για να λειτουργήσει στην συγκεκριμένη περίπτωση είναι διότι ξεκινάει εντοπίζοντας τις δύο μικρότερες πιθανότητες στην κατανομή και τις συγχωνεύει. Σε μία άπειρη πηγή δεν υπάρχει μικρότερη πιθανότητα.

Εύκολα αποδεικνύεται με την βοήθεια του Golomb ότι κάθε άπειρο αλφαριθμητικό μπορεί να γραφεί μοναδικά σαν την συνένωση διαφορετικών X_i με την πιθανότητα κάποιου X_i να συμβαίνει να είναι $(1-p)p^i$. Έτσι καταλήγουμε σε μια κατάσταση όπου τα αλφαριθμητικά αποτελούνται από λέξεις από μια άπειρη πηγή με δεδομένη μια κατανομή P_P. Σαν αποτέλεσμα οι Gallager και Van Voorhis ανέπτυξαν το παρακάτω θεώρημα:

Δεδομένου p, έστω m ο μοναδικός ακέραιος που ικανοποιεί την σχέση:

$$P^m + p^{m+1} <= 1 < p^m + p^{m-1}$$

Έστω ένα δέντρο T να περιγράφεται από μία ακολουθία l_i, i= 0,1,2,3,4… όπου το l_i, είναι ο αριθμός των εσωτερικών κόμβων στο επίπεδο i. Τότε το δέντρο που περιγράφεται ως l_0, l_1, l_2, l_3…. = $1,2,4,\ldots,2^{[\log_2 m]}$, m,m,m,… είναι το βέλτιστο για P_P.

Χρησιμοποιώντας απόρροιες του θεωρήματος και την βασική ιδέα ότι κατασκευάζουμε έναν βέλτιστο κώδικα/δέντρο για την άπειρη πηγή ως εξής:

1. Κατασκευάζουμε ακολουθίες από άπειρες πηγές που είναι ολοένα και καλύτερες προσεγγίσεις της άπειρης πηγής P
2. Μαντεύουμε την δομή του βέλτιστου κώδικα Huffman για αυτές τις άπειρες πηγές
3. Επαληθεύουμε την ορθότητα της πρόβλεψης χρησιμοποιώντας ιδιότητες του Huffman
4. Δείχνουμε ότι οι αριθμήσιμοι αυτοί κώδικες/δέντρα συγκλίνουν σε ένα άπειρο δέντρο που θα πρέπει να είναι βέλτιστο για την άπειρη πηγή.

Το πλεονέκτημα αυτής της προσέγγισης αυτής που χρησιμοποιήθηκε είναι ότι μαντεύοντας την δομή του κώδικα μας φέρνει σε όλους τους βέλτιστους ελεύθερους προθεματικούς κώδικες για όλες τις κατανομές στην κλάση. Ωστόσο ένα μεγάλο μειονέκτημα είναι ότι η προσέγγιση είναι λειτουργική άμα είναι δυνατό να επικυρώσουμε την πρόβλεψη μας για την δομή του κώδικα. Ένα ακόμα μειονέκτημα είναι ότι ακόμα και για κανονικούς προθεματικούς κώδικες μπορεί να είναι αρκετά δύσκολο να προβλέψουμε και να επικυρώσουμε τους βέλτιστους κώδικες για περίπλοκες πηγές.

Παρακάτω θα αναπτύξουμε την πρώτη αλγοριθμική τεχνική για κατασκευή βέλτιστων προθεματικών κωδίκων. Η είσοδος του αλγόριθμου θα είναι το κόστος των γραμμάτων κωδικοποίησης, ο περιορισμός L στης λέξεις του κώδικα καθώς και η κατανομή. Η έξοδος θα είναι μια περιγραφή του άπειρου βέλτιστου δέντρου κωδικοποίησης. Η ιδέες πίσω από τον αλγόριθμο είναι οι παρακάτω.

Αρχικά, αφού έχουν αναπτυχθεί για αριθμήσιμες πηγές με άνισα κόστη κωδικοποίησης και έπειτα τροποποιηθεί για περιορισμένη κωδικοποίηση, τα προθεματικά αυτά δέντρα μπορούν να παρασταθούν από μονοπάτια σε ένα γράφημα με βάρη, όπου ένα δέντρο ελάχιστου κόστους ανταποκρίνεται σε ένα μονοπάτι ελάχιστου κόστους. Στην περίπτωσή μας αυτό σημαίνει ότι αναζητάμε για μονοπάτια με ελάχιστο κόστος αλλά με άπειρους συνδέσμους σε κάποιο άπειρο γράφημα με βάρη. Επιπλέον η δεύτερη ιδέα γενικεύει την παρατήρηση ότι ο βέλτιστος κώδικας Huffman θα πρέπει να είναι κυκλικός. Έτσι μπορούμε να πούμε ότι στην περιγραφή μας με το γράφημα σε ένα μονοπάτι ελάχιστου κόστους αλλά απείρων συνδέσμων θα υπάρχει μία ιδιαίτερη αριθμήσιμη κυκλική δομή. Πριν προχωρήσουμε στην περιγραφή του αλγόριθμου υπογραμμίζεται πως τα βέλτιστα (1,1) δέντρα για τις γεωμετρικές κατανομές έχουν φραγμένο πλάτος.

Μάλιστα χρησιμοποιώντας συνέπειες των παραπάνω αποδεικνύεται πως υπάρχουν το πολύ $2B^3(p)$ πιθανά $\{(l_l, C_l, E_l)\}$ που εμφανίζονται σε ένα βέλτιστο δέντρο για το P_p . Αυτό σημαίνει πως κάπου στα πρώτα $2B^3(p)$ επίπεδα ενός βέλτιστου δέντρου κάποιο επίπεδο πρέπει να επαναλαμβάνεται.

Ο Αλγόριθμος

Παρουσιάζεται ένας αλγόριθμος που κατασκευάζει ένα βέλτιστο (1,2) μονόπλευρο δέντρο για P_p δεδομένου P.Η ιδέα πίσω από τον αλγόριθμο είναι να τροποποιήσουμε τον αλγόριθμο που βρίσκει ένα μονόπλευρο δέντρο για μία αριθμήσιμη πηγή ώστε να βρίσκει ένα μονοπάτι ελαχίστου κόστους σε ένα κατευθυνόμενο γράφημα. Ο αλγόριθμος δούλεψε ξεκινώντας από έναν κόμβο που ανταποκρίνεται στην ρίζα του δέντρου και φτιάχνοντας δέντρα από πάνω προς τα κάτω. Μια ακμή στο γράφημα προσθέτει ένα επίπεδο στην βάση του δέντρου. Το κόστος της της ακμής εκφράζει την συμβολή του συγκεκριμένου επιπέδου στο κόστος του δέντρου. Έτσι κάθε μονοπάτι στο γράφημα που ξεκινάει από τον κόμβο ρίζα ανταποκρίνεται σε κάποιο δέντρο με κόστος ένα μονοπάτι που είναι ίσο με το κόστος του δέντρου που ανταποκρίνεται σε αυτό.

Αφού συγκεκριμενοποιούμε έναν κατάλληλο κόμβο προορισμού ο αλγόριθμος για να βρούμε ένα μονόπλευρο δέντρο ελάχιστου κόστους είναι απλά να βρούμε ένα μονοπάτι από την ρίζα προς τον προορισμό με ελάχιστο κόστος στο γράφημα. Γνωρίζουμε ωστόσο ότι τα δέντρα και τα συσχετιζόμενα μονοπάτια είναι άπειρα αλλά ξέουμε επίσης ότι τα βέλτιστα δέντρα έχουν ένα μοτίβο από ένα «κεφάλι» που ακολουθείται από κύκλους. Σαν αποτέλεσμα ο χώρος μας δεν είναι πια άπειρος αλλά

αριθμήσιμος, και μπορούμε έτσι να τροποποιήσουμε τον αλγόριθμο ώστε να βρίσκουμε ένα μονοπάτι ελάχιστου κόστους. Προκειμένου να γίνει κατανοητός ο αλγόριθμος θα εισάγουμε κάποιες έννοιες. Αρχικά κατασκευάζουμε ένα άπειρο κατευθυνόμενο γράφημα G=(V,E) όπου V= {(m;e,c) : m,e,c ακέραιοι >= 0)}. Κάθε κορυφή (m;e,c) έχει e+1 κατευθυνόμενες εξερχόμενες ακμές. Για 0 <= q <= e οι εξερχόμενες ακμές είναι:

((m;e,c), (m+e-q; c+q, q))

Το βάρος όλων των εξερχόμενων ακμών της (m;e,c) θα είναι $\frac{p^{m+1}}{1-p}$.Ο λόγος που εισάγεται αυτή η εξήγηση είναι ότι θεωρούμε ότι οι κόμβοι ανταποκρίνονται σε επίπεδα ενός δέντρου, ενώ οι ακμές στην πιθανότητα ένα επίπεδο να ακολουθεί ένα άλλο. Έτσι δημιουργούμε μια 1-1 αντιστοίχιση μεταξύ δέντρων και μονοπατιών στο γράφημα G αρχίζοντας από την (0;1,1). Ένα μονοπάτι μήκους k θα αντιστοιχίζεται σε ένα δέντρο με βάθος k, ενώ ένα άπειρο μονοπάτι σε ένα άπειρο δέντρο. Χρησιμοποιώντας κάποιες συνέπειες των παραπάνω οι οποίες παραλείπονται για λόγους συντομίας μπορούμε να καταλήξουμε στο ότι το πρόβλημα να βρούμε ένα δέντρο ελάχιστου κόστους είναι ισοδύναμο με το να βρούμε με το να βρούμε ένα ελάχιστου κόστους άπειρο μονοπάτι στο γράφημα G που ξεκινά στον κόμβο (0;1,1). Εφόσον όμως το γράφημα μας είναι άπειρο δεν μπορούμε να χρησιμοποιήσουμε έναν απλό αλγόριθμο εύρεσης ελάχιστου μονοπατιού.

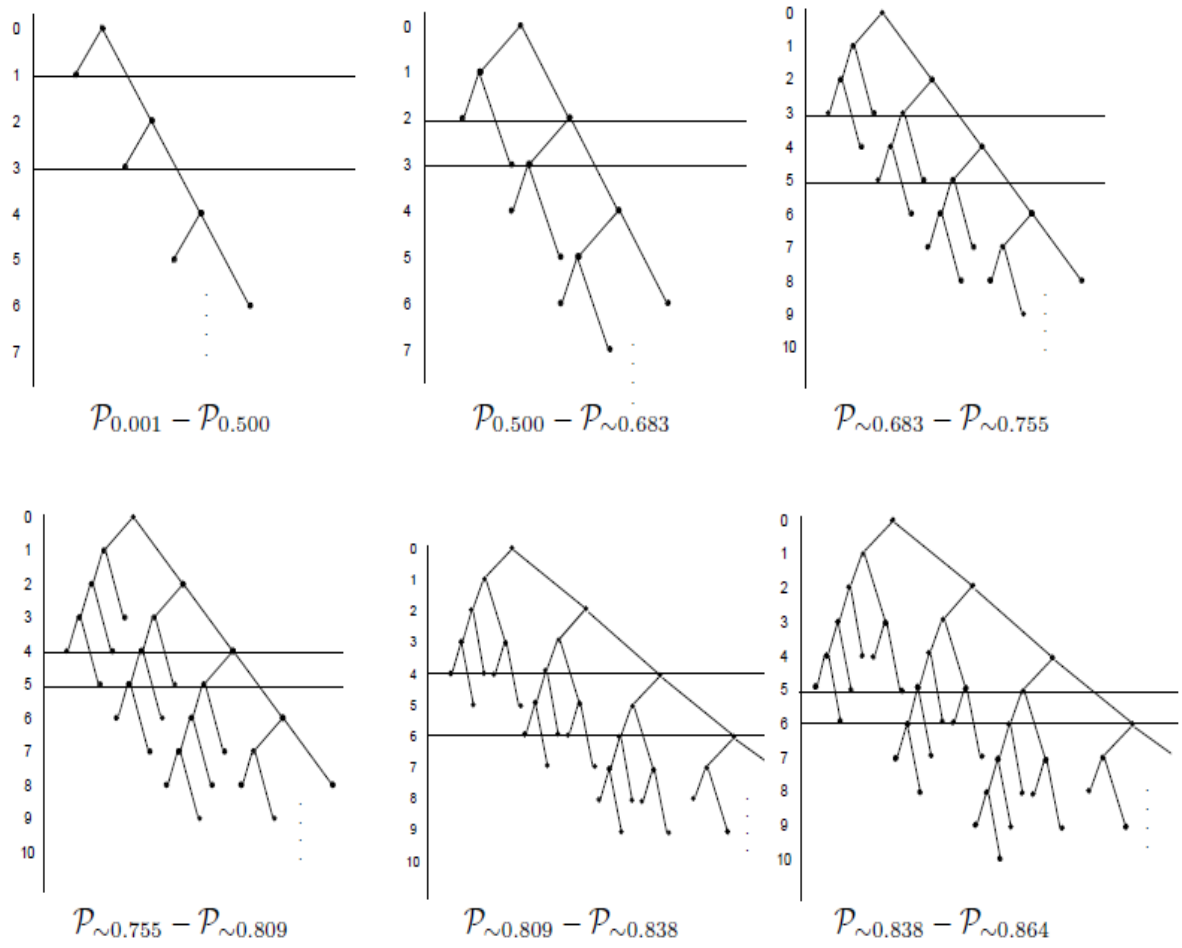
Αντί να ακολουθήσουμε την διαδικασία της παραπάνω προσέγγισης, θα περιορίσουμε τον χώρο σε e,c <= B(p), κάτι που έχει αποδειχθεί νωρίτερα. Έπειτα χρησιμοποιούμε τον αλγόριθμο του Dijkstra που ξεκινάει από τον κόμβο (0;1,1) για να σχεδιάσουμε το συντομότερο μονοπάτι στο γράφημα. Η κύρια παρατήρηση που κάνουμε εδώ είναι ότι τα μονοπάτια που κατασκευάζουμε στο δέντρο συντομότερου μονοπατιού μπορούν να θεωρηθούν σαν προθέματα ενός άπειρου μονοπατιού.

Χρησιμοποιώντας κάποια γνωστά λήμματα μπορούμε εύκολα να βρούμε πως θα είναι το κυκλικό δέντρο που αντιστοιχεί στο αναφερόμενο μονοπάτι καθώς και το κόστος του. Σαν αποτέλεσμα τροποποιούμε τον αλγόριθμο του Dijkstra ώστε κάθε φορά που προσθέτουμε έναν κόμβο (m;e,c) στο δέντρο πηγαίνουμε προς τα πίσω από τον κόμβο (m;e,c) έως να βρούμε είτε έναν κόμβο (m';e,c) αν υπάρχει είτε την ρίζα. Αν βρούμε την ρίζα, συνεχίζουμε με τον αλγόριθμο του Dijkstra. Αν βρούμε έναν κόμβο (m';e,c) τότε βρήκαμε έναν κύκλο αντίστοιχο με το υποψήφιο κυκλικό δέντρο. Υπολογίζουμε το κόστος αυτού του δέντρου και αν έχει το ελάχιστο κόστος κυκλικού δέντρου που έχουμε βρει ως τώρα το κρατάμε σαν νέο ελάχιστο αλλιώς δεν κάνουμε τίποτα. Σημειώνουμε πως ο αλγόριθμος θα τερματίσει εφόσον κάθε μονοπάτι που έχει περισσότερες από $2B^2(p)$ ακμές θα περιέχει κάποια επαναλαμβανόμενη (e,c) τιμή. Έτσι έχουμε ένα αριθμήσιμο γράφημα.

```
   Input: p
1. Initialization
   Set COST[m; e, c] := ∞ for all entries
       with m < 4B^3(p) and e, c < B(p).
   Set COST[0; 1, 1] := 0.  PUSH(Q, (0; 1, 1)).
   /* The current minimum cost of infinite path. */
   Set min_cost := ∞.
2. while Q ≠ ∅ do
2a.      (m; e, c) ← EXTRACT_MIN(Q).
         /* Check for repetition of (e, c) */
         for each vertex (m'; e', c') on the
             path from the root to (m; e, c) do
             if (e', c') = (e, c) then
                /* Calculate cyclic tree cost */
2b.             cost := COST[m'; e', c']
                        + (COST[m; e, c] − COST[m'; e', c']) / (1 − p^(m−m')).
                if min_cost > cost then
                   min_cost := cost.
                   minpath ← a list of vertices on the path.
                Goto 2.
         end for
         /* Relaxation */
2c.      new_cost := COST[m; e, c] + p^(m+1)/(1 − p).
2d.      for q := 0 to e do
2e.          Let (m'; e', c') := (m + e − q; c + q, q).
             if COST[m'; e', c'] > new_cost then
                COST[m'; e', c'] := new_cost.
             if (m'; e', c') is not in Q
                AND COST[m'; e', c'] < min_cost then
                PUSH(Q, (m'; e', c')).
         end for
   end while
3. Extract tree from minpath.
```

Εικόνα 1 – Προσέγγιση του Αλγορίθμου με χρήση ψευδοκώδικα

Τα αποτελέσματα που βρέθηκαν από την τεχνική που ενσωματώσαμε στον αλγόριθμο για να βρούμε βέλτιστα (1,2)μονόπλευρα δέντρα χρησιμοποιήθηκαν για εύρος το p από 0.001 έως 0.960 με αυξήσεις 0.001. Τα ευρήματα φαίνονται παρακάτω στις δύο εικόνες.



Εικόνα 2.*Τα βέλτιστα δέντρα για τα πρώτα 6 διαστήματα.*

Interval	Expansion
$\mathcal{P}_{0.001} - \mathcal{P}_{0.500}$	$\underline{(0;1,1)} \rightarrow (1;1,0) \rightarrow \underline{(1;1,1)}$
$\mathcal{P}_{0.500} - \mathcal{P}_{\sim 0.683}$	$(0;1,1) \rightarrow \underline{(0;2,1)} \rightarrow \underline{(1;2,1)}$
$\mathcal{P}_{\sim 0.683} - \mathcal{P}_{\sim 0.755}$	$(0;1,1) \rightarrow (0;2,1) \rightarrow \underline{(0;3,2)} \rightarrow (2;3,1) \rightarrow \underline{(3;3,2)}$
$\mathcal{P}_{\sim 0.755} - \mathcal{P}_{\sim 0.809}$	$(0;1,1) \rightarrow (0;2,1) \rightarrow \underline{(0;3,2)} \rightarrow \underline{(1;4,2)} \rightarrow \underline{(3;4,2)}$
$\mathcal{P}_{\sim 0.809} - \mathcal{P}_{\sim 0.838}$	$(0;1,1) \rightarrow (0;2,1) \rightarrow (0;3,2) \rightarrow \underline{(0;5,3)} \rightarrow (3;5,2) \rightarrow \underline{(5;5,3)}$
$\mathcal{P}_{\sim 0.838} - \mathcal{P}_{\sim 0.864}$	$(0;1,1) \rightarrow (0;2,1) \rightarrow (0;3,2) \rightarrow (0;5,3) \rightarrow \underline{(2;6,3)} \rightarrow \underline{(5;6,3)}$

Εικόνα 3. *Επεκτάσεις για τα πρώτα 6 διαστήματα. Η υπογράμμιση δηλώνει την ύπαρξη επανάληψης επιπέδου.*

Paper 3: Design and Analysis of Dynamic Huffman-Codes

Σε αυτό το paper το κύριο θέμα μελέτης είναι η παρουσίαση πρόσφατων αλγόριθμων μονού-περάσματος για δημιουργία δυναμικών Κωδικών Huffman.

Εισαγωγή

Οι κώδικες μεταβλητού μήκους χρησιμοποιούν λιγότερα bits για να κωδικοποιήσουν κάθε γράμμα της πηγής από ότι οι κώδικες σταθερού μήκους όπως για παράδειγμα ο ASCII που απαιτεί log n bits για κάθε γράμμα, όπου το n είναι το μέγεθος του αλφαβήτου. Αυτό μπορεί να μειώσει το κόστος της επικοινωνίας σε πολύ μεγάλο βαθμό. Εξάλλου το buffering που απαιτείται για την υποστήριξη κωδίκων μεταβλητού μήκους αποτελεί πια μέρος αρκετών συστημάτων. Είναι γνωστό πως το δυαδικό δέντρο που παράγει ο αλγόριθμος του Huffman ελαχιστοποιεί το μήκος του εξωτερικού μονοπατιού σε σχέση με όλα τα υπόλοιπα δυαδικά δέντρα.

Το μειονέκτημα αυτού του αλγόριθμου είναι ότι πρέπει να γίνουν δύο περάσματα στα δεδομένα: ένα για την συλλογή των συχνοτήτων κάθε γράμματος, την δημιουργία του δέντρου και την αποστολή του στον αποστολέα, και άλλο ένα για την κωδικοποίησης και την μετάδοση των γραμμάτων βασισμένος στη στατική δομή του δέντρου. Σαν αποτέλεσμα αυτό προκαλεί καθυστέρηση στην επικοινωνία αλλά και σε συμπίεση δεδομένων οι πολλές προσπελάσεις του δίσκου καθυστερούν τον αλγόριθμο. Προτείνεται λοιπόν ένας αλγόριθμος ενός περάσματος για την δημιουργία δυναμικού κώδικα Huffman. Το δυαδικό δέντρο που στέλνει αποστολές για την κωδικοποίηση του (t + 1) γράμματος μέσα στο μήνυμα είναι ένα δυαδικό δέντρο Huffman για τα πρώτα t γράμματα του μηνύματος. Έτσι και ο δέκτης και ο αποστολέας θα ξεκινήσουν με το ίδιο αρχικό δέντρο και έτσι μετά θα μείνουν συγχρονισμένοι, ουσιαστικά θα χρησιμοποιήσουν τον ίδιο αλγόριθμο για να τροποποιήσουν το δέντρο μετά την επεξεργασία κάθε γράμματος.

Αυτό μας οδηγεί στο ότι δεν απαιτείται η μετάδοση του δέντρου από τον αποστολέα, όπως γίνεται στον αλγόριθμο με τα δύο περάσματα. Βέβαια αυτή η μέθοδος δεν είναι ιδιαίτερα ενδιαφέρουσα αν ο αριθμός των bits που μεταδίδονται είναι αρκετά μεγαλύτερος από την μέθοδο των δύο περασμάτων. Για τον λόγο αυτό πρέπει να γίνει μια ακριβής μέτρηση της διαφοράς στο μήκος μεταξύ του κωδικοποιημένου μηνύματος μέσω του δυναμικού Huffman και της κωδικοποίησης μέσω του στατικού Huffman. Αποδεικνύεται πως το μήκος (bits) του δυναμικού Huffman (FGK) είναι το πολύ 2S +t όπου το S είναι το μήκος της κωδικοποίησης ενός στατικού κώδικα Huffman και το t είναι ο αριθμός των γραμμάτων στο αρχικό μήνυμα.

Μέσα από τα αποτελέσματα της έρευνας αυτής καταλήγουμε στην δημιουργία ενός νέο αλγόριθμου που ονομάζεται Λ και παράγει κωδικοποιήσεις μικρότερου μήκους από 2S +t bits που αν συγκριθεί με την μέθοδο των δύο περασμάτων χρησιμοποιεί λιγότερο από ένα bit για κάθε γράμμα. Αυτός μάλιστα είναι και ο βέλτιστος αλγόριθμος για την χειρότερη περίπτωση ανάμεσα σε όλες τις μεθόδους Huffman ενός περάσματος. Μια διαίσθηση γιατί ο αλγόριθμος Λ είναι ο βέλτιστος είναι ότι το δέντρο που χρησιμοποιεί για να επεξεργαστεί το t + 1 γράμμα είναι δέντρο Huffman σε σχέση με τα πρώτα t γράμματα αλλά και ελαχιστοποιεί το εξωτερικό μήκος του μονοπατιού καθώς και το ύψος σε σχέση με όλα τα Huffman δέντρα. Στο επόμενο μέρος ακολουθεί μια ανάπτυξη του κάθε αλγορίθμου δηλαδή Huffman δύο περασμάτων και Huffman ενός περάσματος η αλλιώς FGK.

Αλγόριθμος Huffman δύο περασμάτων

Αυτός ο αλγόριθμος αρχικά υπολογίζει τις συχνότητες του κάθε γράμματος μέσα σε ολόκληρο το μήνυμα. Έπειτα δημιουργείται ένα φύλλο στο δέντρο για κάθε γράμμα του μηνύματος, και το βάρος του φύλλου του γράμματος είναι η αντίστοιχη συχνότητα που έχει ήδη υπολογιστεί. Ακολουθεί ο ψευδοκώδικας για την διαδικασία δημιουργίας του δυαδικού δέντρου με το μονοπάτι με το ελάχιστο βάρος.

```
Store the k leaves in a list L;
while L contains at least two nodes do
  begin
  Remove from L two nodes x and y of smallest weight;
  Create a new node p, and make p the parent of x and y;
  p's weight := x's weight + y's weight;
  Insert p into L
  end;
```

Εικόνα 1 – Ψευδοκώδικας για διαδικασία δημιουργίας δυαδικού δέντρου

Σε κάθε προσπέλαση ενδεχομένως να υπάρχει επιλογή μεταξύ κόμβων με το ίδιο ελάχιστο βάρος ώστε να βγουν από την λίστα οπότε να προκύψουν διαφορετικά δέντρα Huffman. Σε κάθε περίπτωση ωστόσο όλα τα πιθανά αυτά δέντρα θα έχουν ένα μονοπάτι με ίδιο βεβαρυμένο μήκος. Υπενθυμίζεται ότι κώδικες σαν τον από πάνω όπου κάθε κωδικοποίηση για οποιοδήποτε γράμμα δεν μπορεί να είναι πρόθεμα κάποιας κωδικοποίησης για κάποιο άλλο γράμμα λέγονται προθεματικοί. Τα δύο κύρια μειονεκτήματα του παραπάνω κώδικα είναι ότι απαιτούνται δύο περάσματα και επιπλέον το κόστος που απαιτείται για να μεταδώσουμε την δομή του δέντρου. Για τον λόγο αυτό ψάχνουμε εναλλακτικές μεθόδους με ένα πέρασμα όπου

τα γράμματα θα κωδικοποιούνται απευθείας. Δεν χρησιμοποιούμε έναν στατικό κώδικα που είναι βασισμένος σε ένα μοναδικό δυαδικό δέντρο αλλά αντιθέτως η κωδικοποίηση βασίζεται σε ένα δέντρο Huffman που αλλάζει δυναμικά.

Ουσιαστικά, το δέντρο που χρησιμοποιείται για την κωδικοποίηση του t +1 γράμματος είναι ένα δέντρο Huffman σε σχέση με το M_t. Ο αποστολέας κωδικοποιεί το t +1 γράμμα στο μήνυμα με την ακολουθία από μηδέν και ένα που συγκεκριμενοποιεί το μονοπάτι από την ρίζα στο φύλλο του γράμματος αυτού. Ο δέκτης έπειτα λαμβάνει το αντίστοιχο γράμμα από την επεξεργασία του αντίγραφου που έχει από το δέντρο αυτό. Ύστερα αποστολέας και δέκτης τροποποιούν τα αντίγραφα των δέντρων τους πριν επεξεργαστούν το επόμενο γράμμα ώστε να έχουν και οι δύο ένα δέντρο Huffman για το M_{t+1}.

Το κλειδί σε αυτήν την μέθοδο είναι ότι ούτε το δέντρο αλλά ούτε και η κωδικοποίηση του χρειάζεται να μεταδοθεί εφόσον αποστολέας και δέκτης χρησιμοποιούν τον ίδιο αλγόριθμο τροποποίησης και έτσι θα έχουν ισοδύναμα αντίγραφα των δέντρων σε κάθε στιγμή.

Παρακάτω παρουσιάζεται ένα χαρακτηριστικό των δέντρων Huffman που θα βοηθήσει στην συνέχεια:

Ένα δυαδικό δέντρο με p φύλλα με μη-αρνητικά βάρη είναι ένα δέντρο Huffman αν και μόνον αν:

1) **Τα p φύλλα έχουν μη-αρνητικά βάρη w1, w2, ……, wp και το βάρος του κάθε εσωτερικού κόμβου είναι το άθροισμα των βαρών των παιδιών του.**

2) **Οι κόμβοι μπορούν να αριθμηθούν σε αύξουσα σειρά βάρους, ώστε ο κόμβος 2j-1 και ο 2j είναι αδέρφια, για 1<= j <= p-1 και ο κοινός τους κόμβος πατέρας είναι μεγαλύτερος στην αρίθμηση.**

Έστω λοιπόν ότι έχουμε ένα μήνυμα $M_t = a_{i,1}$, $a_{i,2}$,…….$a_{i,t}$ το οποίο έχει ήδη επεξεργαστεί. Το επόμενο γράμμα $a_{i,t+1}$ κωδικοποιείται και αποκωδικοποιείται χρησιμοποιώντας το δέντρο για το M_t. Η κύρια δυσκολία είναι το πώς να τροποποιηθεί το δέντρο γρήγορα ώστε να έχουμε το δέντρο Huffman για το μήνυμα M_{t+1}. Η λύση για να μην παραβιάζεται η παραπάνω ιδιότητα είναι μια διαδικασία δύο φάσεων. Στην πρώτη φάση μετατρέπουμε το δέντρο σε ένα άλλο δέντρο Huffman για το M_t στο οποίο απλά αυξάνοντας τα βάρη κατά ένα στην φάση δύο θα έχουμε το δέντρο Huffman δέντρο για το M_{t+1}. Η πρώτη φάση ξεκινάει με το φύλλο $a_{i,t+1}$ ως το τωρινό κόμβο. Εναλλάσσουμε επαναλαμβανόμενα τα περιεχόμενα του τωρινού κόμβου, συμπεριλαμβανομένου και του υπο-δέντρου, με αυτά του κόμβου με το ίδιο βάρος αλλά μεγαλύτερη αρίθμηση και κάνουμε πατέρα του τελευταίου τον νέο τωρινό κόμβο. Στην δεύτερη φάση μετατρέπουμε το δέντρο αυτό στο επιθυμητό Huffman δέντρο για το M_{t+1} απλά αυξάνοντας τα βάρη του φύλλου $a_{i,t+1}$ και των

απόγονών του κατά 1. Στο νέο δέντρο παρατηρούμε ότι ικανοποιείται η παραπάνω ιδιότητα άρα θα έχουμε ένα δέντρο Huffman. Οι φάσεις 1 και 2 παρουσιάζονται συνδυασμένες στον ψευδοκώδικα που παρουσιάζεται παρακάτω:

```
procedure Update;
begin
q := leaf node corresponding to a_{i_{t+1}};
if (q is the 0-node) and (k < n − 1) then
   begin
   Replace q by a parent 0-node with two leaf 0-node children, numbered in the order left
      child, right child, parent;
   q := right child just created
   end;
if q is the sibling of a 0-node then
   begin
   Interchange q with the highest numbered leaf of the same weight;
   Increment q's weight by 1;
   q := parent of q
   end;
while q is not the root of the Huffman tree do
   begin   {Main loop}
   Interchange q with the highest numbered node of the same weight;
   {q is now the highest numbered node of its weight}
   Increment q's weight by 1;
   q := parent of q
   end
end;
```

Εικόνα 2 – Ψευδοκώδικας ο οποίος προκύπτει από τις φάσεις 1 και 2 για την μέθοδο δημιουργίας δυναμικού Κώδικα Huffman

Ας αναλύσουμε τώρα την αποτελεσματικότητα της κωδικοποίησης του αλγόριθμου Huffman ενός περάσματος με τους αλγόριθμους δύο περασμάτων. Θα μετρήσουμε μόνο τα bits στα μονοπάτια που διατρέχονται κατά την διάρκεια της κωδικοποίησης. Για τον αλγόριθμο ενός περάσματος δεν θα μετρήσουμε τα bits που απαιτούνται για να ξεχωρίσουμε ποιο νέο γράμμα όταν ένα νέο γράμμα συναντάται για κωδικοποίηση για πρώτη φορά. Για τον αλγόριθμο των δύο περασμάτων δεν θα συμπεριλάβουμε τα bits που απαιτούνται για κωδικοποιήσουμε το σχήμα του δέντρου και για τις ετικέτες των φύλλων.

Για τον αλγόριθμο του ενός περάσματος η μη- μετρήσιμη ποσότητα αυτών των bits είναι μεταξύ $k(\log_2 n-1)$ και $k\log_2 n$ bits χρησιμοποιώντας έναν απλό προθεματικό κώδικα, ενώ η μη μετρήσιμη ποσότητα για την μέθοδο των δύο περασμάτων είναι σχεδόν 2k bits περισσότερη από τον αλγόριθμο ενός περάσματος. Προχωράμε σε κάποιες διευκρινίσεις απαραίτητες για τα επόμενα.

Έστω μήνυμα $M_t = a_{i,1}, a_{i,2},\ldots\ldots a_{i,t}$ όπου t>=0 το οποίο έχει επεξεργαστεί μέχρι τώρα. Καθορίζουμε S_t να είναι το κόστος της επικοινωνίας για μία στατική

κωδικοποίηση Huffman του M_t χρησιμοποιώντας ένα δέντρο Huffman βασισμένο μόνο στο M_t όπου ισούται με:

$$S_t = \sum_j w_j l_j,$$

Όπου το άθροισμα υπολογίζεται από κάθε Huffman δέντρο για το M_t. Επιπλέον ορίζουμε το s_t να είναι το κόστος αύξησης:

$$s_t = S_t - S_{t-1}.$$

Σημειώνουμε ως d_t να είναι το κόστος επικοινωνίας για την κωδικοποίηση $a_{i,t}$ χρησιμοποιώντας έναν κώδικα Huffman όπου ισούται με :

$$d_t = l_{i_t}$$

Για το δυναμικό δέντρο Huffman όσον αφορά το M_{t-1}. Ορίζουμε D_t να είναι το ολικό κόστος επικοινωνίας για όλα τα t γράμματα όπου ισούται με:

$$D_t = D_{t-1} + d_t, \qquad D_0 = 0.$$

Εφόσον καθορίσαμε την σημειολογία διαπιστώνεται πως τα σημαντικά σημεία για τον βέλτιστο αλγόριθμο είναι οι παρακάτω:

1) Ο αριθμός των εναλλαγών μεταξύ του τωρινού κόμβου (q στον ψευδοκώδικα) θα πρέπει να φράσσεται από έναν μικρό αριθμό σε κάθε διαδικασία Update.

2) Το δυναμικό Huffman δέντρο θα πρέπει να κατασκευαστεί ώστε να ελαχιστοποιεί όχι μόνο το $\Sigma w_j l_j$ αλλά και τα $\sum j\ lj$, $\max(l_j)$ εφόσον έτσι δεν μπορεί να υπάρξει μεγάλου μήκους κωδικοποίηση για το επόμενο γράμμα μέσα στο μήνυμα.

Ο αλγόριθμος που ικανοποιεί τα παραπάνω και είναι βέλτιστος είναι ο αλγόριθμος Λ. συγκεκριμένα ο αλγόριθμος Λ ελαχιστοποιεί την διαφορά χειρότερης περίπτωσης $D_t - S_t$ για όλα τα μηνύματα μήκους t σε σχέση με όλους τους αλγόριθμους Huffman ενός περάσματος.

```
procedure Update;
begin
leafToIncrement := 0;
q := leaf node corresponding to a_{i_{t+1}};
if (q is the 0-node) and (k < n − 1) then
   begin  {Special Case #1}
   Replace q by an internal 0-node with two leaf 0-node children, such that the right child
        corresponds to a_{i_{t+1}};
   q := internal 0-node just created;
   leafToIncrement := the right child of q
   end
else begin
   Interchange q in the tree with the leader of its block;
   if q is the sibling of the 0-node then
      begin  {Special Case #2}
      leafToIncrement := q;
      q := parent of q
      end
   end;
while q is not the root of the Huffman tree do
   {Main loop; q must be the leader of its block}
   SlideAndIncrement(q);
if leafToIncrement ≠ 0 then  {Handle the two special cases}
   SlideAndIncrement(leafToIncrement)
end;

procedure SlideAndIncrement(p);
begin
wt := weight of p;
b := block following p's block in the linked list;
if ((p is a leaf) and (b is the block of internal nodes of weight wt))
   or ((p is an internal node) and
       (b is the block of leaves of weight wt + 1)) then
      begin
      Slide p in the tree ahead of the nodes in b;
      p's weight := wt + 1;
      if p is a leaf then p := new parent of p
      else p := former parent of p
      end
end;
```

Εικόνα 3 – Ψευδοκώδικας Αλγορίθμου Λ για την μέθοδο Update

Παρακάτω παρουσιάζονται τα βασικά χαρακτηριστικά της δομής δεδομένων για τον αλγόριθμο Λ.

- Θα πρέπει να αναπαριστά ένα δυαδικό δέντρο με μη-μηδενικά βάρη
- Θα πρέπει να αποθηκεύει μια συνεχόμενη λίστα από εσωτερικούς κόμβους σε μη-φθίνουσα σειρά βάρους όπου κόμβοι με το ίδιο βάρος στοιχίζονται με βάση την αρίθμηση. Μια παρόμοια λίστα αποθηκεύεται για τα φύλλα.
- Πρέπει να βρίσκεται ένας αρχηγός του μπλοκ του κόμβου για κάθε κόμβο με βάση την αρίθμηση
- Πρέπει να εναλλάσσει τα περιεχόμενα δύο φύλλων του ίδιου βάρους
- Πρέπει να αυξάνει το βάρος του αρχηγού κάθε μπλοκ κατά 1 το οποίο μπορεί να οδηγήσει στην αρίθμηση του κόμβου να μεταφερθεί πίσω από τα

νούμερα των κόμβων στο επόμενο μπλοκ, με αποτέλεσμα τα νούμερα αυτά να μειωθούν κατά 1

- Πρέπει να αναπαριστά τα n-k γράμματα του αλφάβητου που δεν έχουν ακόμα εμφανιστεί στο μήνυμα με ένα μοναδικό φύλλο 0-κόμβος στο δέντρο Huffman

Αποτελέσματα

Προκειμένου να αναφερθούμε στα κόστη επικοινωνίας του αλγόριθμου Huffman, του αλγόριθμου Λ και του αλγόριθμου FGK θα χρησιμοποιήσουμε τα S_t , D_t^{Λ}, και D_t^{FGK} αντίστοιχα. Όσον αφορά τα bits/ γράμμα (b/l) ο αλγόριθμος Λ είναι πολύ αποτελεσματικότερος από τον αλγόριθμο των δύο περασμάτων σε όλα τα πειράματα όπου $t <= 10^4$. Ο αλγόριθμος FGK χρησιμοποίησε ελαφρώς περισσότερα bits ανά γράμμα, ωστόσο ήταν σχετικά αποτελεσματικός. Ο αλγόριθμος Λ έχει το πλεονέκτημα ότι χρησιμοποιεί λιγότερα bits ανά γράμμα για μικρά μηνύματα ενώ οι διαφορές στην αποτελεσματικότητα της κωδικοποίησης είναι σχετικά πιο σημαντικές. Παρακάτω παρουσιάζονται τα αποτελέσματα των πειραμάτων σε πίνακες:

t	k	S_t	b/l	D_t^{Λ}	b/l	D_t^{FGK}	b/l
100	96	664	13.1	569	10.2	659	11.2
500	96	3320	7.9	3225	7.4	3335	7.6
960	96	6400	7.1	6305	6.8	6415	6.9

t	k	S_t	b/l	D_t^{Λ}	b/l	D_t^{FGK}	b/l
100	10	340	4.2	340	4.0	345	4.1
500	50	2860	6.5	2820	6.2	2863	6.3
960	96	6400	7.1	6305	6.8	6393	6.9

t	k	S_t	b/l	D_t^{Λ}	b/l	D_t^{FGK}	b/l
100	34	434	7.1	420	6.3	444	6.5
500	52	2429	5.7	2445	5.5	2489	5.6
1000	58	4864	5.3	4900	5.2	4953	5.3
10000	74	47710	4.8	47852	4.8	47938	4.8
12280	76	58457	4.8	58614	4.8	58708	4.8

t	k	S_t	b/l	D_t^{Λ}	b/l	D_t^{FGK}	b/l
100	9	124	2.1	117	1.9	122	2.0
500	9	524	1.2	517	1.2	522	1.2
1000	9	1024	1.1	1017	1.1	1022	1.1
10000	249	52407	5.5	52608	5.4	52868	5.5
34817	256	205688	6.0	206230	6.0	206585	6.0

t	k	S_t	b/l	D_t^{Λ}	b/l	D_t^{FGK}	b/l
100	40	372	7.7	345	6.7	378	7.0
500	123	2566	7.5	2514	6.9	2625	7.1
1000	177	5904	7.6	5875	7.2	6029	7.3
10000	248	67505	7.0	67769	6.9	67997	7.0
100000	256	691897	6.9	692591	6.9	692858	6.9
588868	256	4170298	7.1	4171314	7.1	4171616	7.1

Συνοψίζοντας ο αλγόριθμος Λ κάνει κωδικοποίηση και αποκωδικοποίηση πραγματικού χρόνου των μηνυμάτων σε ένα πέρασμα χρησιμοποιώντας λιγότερο από 1 επιπλέον bit για κάθε γράμμα για την κωδικοποίηση του μηνύματος συγκρινόμενος με τον αλγόριθμο Huffman δύο περασμάτων. Αυτό είναι το βέλτιστο της χειρότερης περίπτωσης σε σχέση με όλες της μεθόδους Huffman ενός περάσματος. Τα αποτελέσματα των παραπάνω πειραμάτων δείχνουν ότι ο αριθμός των bits για κάθε γράμμα είναι ίσος και συχνά καλύτερος από την μέθοδο των δύο περασμάτων. Για το λόγω αυτό είναι δυνητικά καλύτερος για χρήση σε συμπίεση αρχείων και σε επικοινωνία διαδικτύου καθώς και για ενσωμάτωση στο υλικό(hardware).