ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ – ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

Distributed and Online Maintenance of Graphical Models in Apache Flink

Νικόλαος Τζήμος

Εξεταστική επιτροπή:

Καθηγητής Αντώνιος Δεληγιαννάκης (Επιβλέπων)

Καθηγητής Μίνως Γαροφαλάκης

Αναπληρωτής Καθηγητής Βασίλειος Σαμολαδάς

Φεβρουάριος 2023

Abstract

With the growing need for large scale data analysis, distributed machine learning has grown importance in recent years. The raw data is described by large number of interrelated variables and an important task is to describe the joint probability distribution over these variables, allowing simultaneously interferences and predications to be made. Directly modeling of joint probability distribution of all these variables may be infeasible, since the complexity of such model grown exponential with the number of variables. We focus on Bayesian Networks, the father of graphical models and present a different communication-efficient approach using the well-known method of Functional Geometric Monitoring, for continuously learning and maintenance of Bayesian Networks in a distributed streaming environment. Finally, the experimental results confirmed the functionality of proposed method.

Περίληψη

Με την αυξανόμενη ανάγκη για ανάλυση των δεδομένων σε μεγάλη κλίμακα, η κατανεμημένη μηχανική μάθηση έχει αποκτήσει σημασία τα τελευταία χρόνια. Τα δεδομένα συνήθως περιγράφονται από ένα μεγάλο αριθμό από αλληλοσχετιζόμενες μεταβλητές και μια σημαντική διεργασία είναι να μπορούμε να περιγράψουμε την από κοινού κατανομή όλων των μεταβλητών, επιτρέποντας την λήψη συμπερασμών και προβλέψεων. Ωστόσο η "απευθείας" μοντελοποίηση της από κοινού κατανομής όλων των μεταβλητών είναι μη εφικτή, αφού η πολυπλοκότητα ενός τέτοιου μοντέλου αυξάνεται εκθετικά με τον αριθμό των μεταβλητών. Εστιάζουμε στα Bayesian Networks , τα οποία αποτελούν τον "πατέρα" των γραφικών μοντέλων και παρουσιάζουμε μια διαφορετική προσέγγιση που είναι "επικοινωνιακά" αποδοτική χρησιμοποιώντας την ευρέως γνωστή μέθοδο του Functional Geometric Monitoring, για την συνεχή εκμάθηση και διατήρηση των Bayesian Networks πάνω σε κατανεμημένο περιβάλλον. Τέλος τα πειραματικά αποτελέσματα επιβεβαιώνουν την λειτουργικότητα την προτεινομένης προσέγγισης.

Ευχαριστίες

Θα ήθελα αρχικά να ευχαριστήσω τον καθηγητή μου Αντώνιο Δεληγιαννάκη για την επίβλεψη της διπλωματικής εργασίας μου αλλά και για την διαρκεί υποστήριξη σε όλες τις δυσκολίες που αντιμετώπισα καθόλη της διάρκεια εκπόνησης της παρούσας διπλωματικής εργασίας. Επίσης θα ήθελα να εκφράσω το θαυμασμό μου και για τα άλλα δύο μέλη της επιτροπής Καθ. Μίνω Γαρουφαλάκη και Βασίλειο Σαμολαδά για τα εφόδια και την παρακίνηση που μου πρόσφεραν καθόλη την διάρκεια των σπουδών μου.

Επιπλέον θα ήθελα να ευχαριστήσω την οικογένεια μου για την συνεχή υποστήριξη καθόλη την διάρκεια και τέλος θα ήθελα να ευχαριστήσω τους φίλους μου στα Χανιά για όλες τις εμπειρίες που περάσαμε μαζί.

Νικόλαος Τζήμος

Πίνακας περιεχομένων

Εισαγωγή ... 10
1.1 Αντικείμενο της διπλωματικής ... **10**
1.2 Οργάνωση διπλωματικής ... **11**
Θεωρητικό Υπόβαθρο ... 12
2.1 Σημειογραφία ... **12**
2.2 Bayesian Networks ... **15**
2.2.1 Εισαγωγή στα Bayesian Networks ... 15
2.2.2 Bayesian Network Semantics ... 19
2.2.3 *Student Bayesian Network* ... 22
2.3 Naïve Bayes Classifiers ... **24**
2.3.1 Εισαγωγή στα Naïve Bayes Classifiers ... 24
2.3.2 Naïve Bayes Classifiers Semantics ... 24
2.3.3 Spam classification πρόβλημα ... 27
2.4 Forward Sampling ... **29**
2.5 Εκμάθηση παραμέτρων ... **34**
2.5.1 Maximum Likelihood Estimation (MLE) ... 34
2.6 Laplace Smoothing ... **38**
2.7 Distributed Continuous Model ... **40**
Δήλωση του προβλήματος ... 43
3.1 Ορισμός του προβλήματος ... **43**
3.2 Η γενική προσέγγιση ... **45**
Ανάλυση του προβλήματος ... 48
4.1 Μια πρώτη προσέγγιση ... **48**
4.1.1 Approximated Distributed counters ... 49
4.1.1.1 Randomized Counters ... 50
4.1.1.2 Deterministic Counters ... 53
4.1.1.3 Σύγκριση μεταξύ RANDOMIZED και DETERMINISTIC ... 55
4.1.2 Ανάλυση των αλγορίθμων BASELINE, UNIFORM και NON_UNIFORM ... 56
4.1.2.1 Dummy Father ... 60
4.1.2.2 Σύγκριση μεταξύ των αλγορίθμων BASELINE, UNIFORM, NON_UNIFORM 61
4.1.3 Communication cost από Naïve Bayes Classifier ... 62
4.2 Μια δεύτερη προσέγγιση ... **63**
4.2.1 Functional Geometric Monitoring (FGM) ... 64
Σχεδίαση και Υλοποίηση του Συστήματος ... 67

5.1 Apache Flink **67**
5.2 Apache Kafka **68**
5.3 Αρχιτεκτονική του Συστήματος **69**
Πειραματική Αξιολόγηση 72
6.1 Datasets **72**
6.2 Μετρικές απόδοσης **74**
6.3 Πειραματικά αποτελέσματα **74**
6.3.1 Communication cost σε σχέση με τον αριθμό των training instances 75
6.3.2 Communication cost σε σχέση με το approximation factor ε 79
6.3.3 Communication cost σε σχέση με τον αριθμό των workers 81
6.3.4 Scalability 82
6.3.5 Dummy Father 87
Επίλογος 88
7.1 Συμπεράσματα **88**
7.2 Μελλοντικές Επεκτάσεις **88**
Παράρτημα 89
Βιβλιογραφία 102

Κατάλογος Εικόνων

Εικόνα 1: Directed Acyclic Graph(DAG) από το Student Bayesian δίκτυο 13
Εικόνα 2: Συνάρτηση κατανομή πιθανότητας(pmf) από μια binary-valued τυχαία μεταβλητή Xi 17
Εικόνα 3: Κομμάτι από τον γράφο του Student Bayesian Network..... 18
Εικόνα 4: Tabular CPD του κόμβου Grade 20
Εικόνα 5: Equal width binning 21
Εικόνα 6: Student Bayesian Network μαζί με τα CPDs..... 22
Εικόνα 7: Naïve Bayes Classifier 25
Εικόνα 8: Naïve Bayes Classifier για spam filtering 27
Εικόνα 9: Topological ordering από το Student Bayesian Δίκτυο 30
Εικόνα 10: Δειγματοληψία multinomial κατανομής..... 32
Εικόνα 11: Distributed Continuous Model 41
Εικόνα 12: Τιμές του $\bar{n}_{RC}$ και του $\bar{n}_{DC}$για διάφορες τιμές του k 56
Εικόνα 13: Dummy Father..... 60
Εικόνα 14: Apache Flink 67
Εικόνα 15: Feedback loop 68
Εικόνα 16: Apache Kafka 69
Εικόνα 17: Αρχιτεκτονική του συστήματος..... 69
Εικόνα 18: Hashing 71
Εικόνα 19: Error to GT σε σχέση με τον αριθμό των training instances και για τις δυο προσεγγίσεις, για το dataset HEPAR II 76
Εικόνα 20: Error to EXACTMLE σε σχέση με τον αριθμό των training instances και για τις δυο προσεγγίσεις, για το dataset HEPAR II 76
Εικόνα 21: Communication cost σε σχέση με τον αριθμό των training instances και για τις δυο προσεγγίσεις, για το dataset HEPAR II 77
Εικόνα 22: Communication cost σε σχέση με τον αριθμό των training instances και για τις δυο προσεγγίσεις για το dataset HEPAR II-Best results 78
Εικόνα 23: Communication cost σε σχέση με το approximation factor ε, για το dataset HEPAR II και τον αλγόριθμο UNIFORM 79
Εικόνα 24: Communication cost σε σχέση με το approximation factor ε και για τις δύο προσεγγίσεις, για το dataset HEPAR II και τον αλγόριθμο UNIFORM 80
Εικόνα 25: Error to GT σε σχέση με το approximation factor ε για τις δύο προσεγγίσεις, για το dataset HEPAR II και τον αλγόριθμο UNIFORM 80
Εικόνα 26: Communication cost σε σχέση με το αριθμό των sites για τις δύο προσεγγίσεις, για το dataset HEPAR II και τον αλγόριθμο UNIFORM..... 81
Εικόνα 27: Throughput(events/sec) σε σχέση με το αριθμό των sites για τις δύο προσεγγίσεις, για το dataset HEPAR II 83
Εικόνα 28: Throughput(events/sec) σε σχέση με το αριθμό των sites για την πρώτη προσέγγιση, για το dataset HEPAR II μετά το αρχικό στάδιο 83
Εικόνα 29: Throughput(events/sec) σε σχέση με το αριθμό των sites για τις δύο προσεγγίσεις, για το dataset HEPAR II-Best results 84
Εικόνα 30: Throughput(events/sec) σε σχέση με το αριθμό του παραλληλισμού για τις δύο προσεγγίσεις, για το dataset HEPAR II 85
Εικόνα 31: Throughput(events/sec) σε σχέση με το αριθμό του παραλληλισμού για την πρώτη προσέγγιση, για το dataset HEPAR II μετά το αρχικό στάδιο 85

Εικόνα 32: Throughput(events/sec) σε σχέση με το αριθμό του παραλληλισμού για τις δύο προσεγγίσεις, για το dataset HEPAR II - Best results 86
Εικόνα 33: Communication cost σε συνδυασμό με την μέθοδο του Dummy Father(DF), για τα datasets HEPAR II,ALARM..... 87
Εικόνα 34: Mean error to GT σε σχέση με τον αριθμό των training instances και για τις δυο προσεγγίσεις, για το dataset HEPAR II..... 89
Εικόνα 35: Communication cost σε σχέση με τον αριθμό των training instances για τα datasets LINK,ALARM..... 90
Εικόνα 36: Communication cost σε σχέση με το approximation factor ε για τις δύο προσεγγίσεις, για το dataset HEPAR II..... 91
Εικόνα 37: Communication cost σε σχέση με το αριθμό των sites για τις δύο προσεγγίσεις, για το dataset HEPAR II 92
Εικόνα 38: Runtime(sec) σε σχέση με το αριθμό των sites για τις δύο προσεγγίσεις, για το dataset HEPAR II..... 92
Εικόνα 39: Runtime(sec) σε σχέση με το αριθμό των sites για την πρώτη προσέγγιση, για το dataset HEPAR II μετά το αρχικό στάδιο..... 93
Εικόνα 40: Runtime(sec) σε σχέση με το αριθμό των sites για τις δύο προσεγγίσεις, για το dataset HEPAR II-Best results..... 93
Εικόνα 41: Runtime(sec) σε σχέση με το αριθμό του παραλληλισμού για τις δύο προσεγγίσεις, για το dataset HEPAR II 94
Εικόνα 42: Runtime(sec) σε σχέση με το αριθμό του παραλληλισμού για την πρώτη προσέγγιση, για το dataset HEPAR II μετά το αρχικό στάδιο 94
Εικόνα 43: Runtime(sec) σε σχέση με το αριθμό των sites για τις δύο προσεγγίσεις, για το dataset HEPAR II - Best results 95
Εικόνα 44: Δομή και οργάνωση του project 96

Κατάλογος Πινάκων

Πίνακας 1: Περίληψη σημειογραφίας 50
Πίνακας 2: Space και communication cost από τον κάθε counter 55
Πίνακας 3: Approximation factor και Communication Cost για όλους τους αλγορίθμους 60
Πίνακας 4: Bayesian Networks..... 73

Κεφάλαιο 1:

Εισαγωγή

1.1 Αντικείμενο της διπλωματικής

Η παρούσα διπλωματική παρουσιάζει ένα σύστημα το οποίο είναι ικανό να υποστηρίξει την κατασκευή και το *maintenance* ενός ευρέως γνωστού *graphical structure* δηλαδή τα *Bayesian Networks(BNs)*, πάνω σε δεδομένα τα οποία αναφέρονται σε μεγάλο όγκο δεδομένων, είναι *"εξελισσόμενα", κατανεμημένα και πολυδιάστατα*, χρησιμοποιώντας ταυτόχρονα όσον το δυνατό λιγότερο *communication cost* γίνεται. Σκοπός του συστήματος είναι το *maintenance* και η συνεχή εκμάθηση των παραμέτρων(*continuously parameter learning*) χρησιμοποιώντας *communication-efficient* αλγορίθμους πάνω στο *distributed continuously* μοντέλο με βάση τον αλγόριθμο του *Maximum Likelihood Estimation (MLE)*.

Η πρώτη προσέγγιση βασίζεται στην χρησιμοποίηση *approximate distributed counters* σε συνδυασμό με τους αλγορίθμους *BASELINE,UNIFORM* και *NON_UNIFORM*. Αυτοί οι αλγόριθμοι καθορίζουν το τρόπο με τον οποίο θα ορίσουμε το *approximation factor ε* σε κάθε *counter* με σκοπό να παρέχουμε τα κατάλληλα *error guarantees* από το *joint probability distribution* του *MLE*. Το σύστημα μπορεί να υποστηρίξει δυο από τα βασικά *approximate distributed counter*, το πρώτο αφορά *RANDOMIZED counters* ενώ το δεύτερο αφορά *DETERMINISTIC counters*. Αυτή η προσέγγιση οδηγεί σε εκθετική μείωση του *communication cost* σε σχέση με την συντηρητική προσέγγιση του *EXACTMLE* το οποίο επιτυγχάνεται με την διατήρηση των *EXACT counters*.

Επιπλέον το σύστημα που υλοποιήσαμε υποστηρίζει το *maintenance* από μια ειδική περίπτωση των *Bayesian Networks* αυτήν των *Naïve Bayes Classifier(NBC)*. Ολόκληρο το σύστημα υλοποιήθηκε στο *Apache Flink* σε συνδυασμό με το *Apache Kafka*.

Η παρούσα διπλωματική πέραν από την πρώτη προσέγγιση προτείνει μια εναλλακτική προσέγγιση για το *maintenance* και το *continuously learning* των παραμέτρων ενός *Bayesian Networks*. Συγκεκριμένα η προσέγγιση αυτή βασίζεται στην ευρέων γνωστή μέθοδο του *Functional Geometric Protocol (FGM)* σε συνδυασμό με τους δυο αλγορίθμους *BASELINE* και *UNIFORM* που καθορίζουν με τα κατάλληλο τρόπο το διαθέσιμο *error budget*. Η ιδέα για να μπορέσουμε να χρησιμοποιήσουμε την μέθοδο *FGM* είναι ότι πλέον τα *approximate distributed counters* χρειάζεται να τα βλέπουμε ως *frequency vectors* δηλαδή το *state* που χρησιμοποιούμε να αντιστοιχεί σε κάποιο *vector* και όχι να αντιμετωπίσει τον κάθε *counter* ξεχωριστά(*individually*). Η προσέγγιση που προτείναμε οδηγεί σε μείωση του *communication cost* έως και μια τάξη μεγέθους από την προηγούμενη προσέγγιση και έως δυο τάξεις μεγέθους σε σχέση με το *EXACTMLE*. Τέλος όπως φαίνεται και από την πειραματική ανάλυση η προσέγγιση που προτείναμε μπορεί να διαχειριστεί με ικανοποιητικό τρόπο μεγάλο όγκο δεδομένων παρέχοντας ταυτόχρονα και το κατάλληλο *scaling*.

Τέλος, η παρούσα διπλωματική ενσωμάτωσε την ευρέως γνωστή μέθοδο του *Laplace Smoothing* για την διαχείριση *zero true* παραμέτρων και επιπλέον πρότεινε την μέθοδο του *Dummy Father(DF)* σε συνδυασμό με τον αλγόριθμο *NON_UNIFORM*, το οποίο όπως επιβεβαιώνεται και από την πειραματική ανάλυση οδηγεί σε ένα ποσοστό μείωσης της τάξης του 50%.

1.2 Οργάνωση διπλωματικής

Η παρούσα διπλωματική οργανώνεται με τον ακόλουθο τρόπο. Συγκεκριμένα το ***Κεφάλαιο 2*** περιέχει όλο το θεωρητικό υπόβαθρο που απαιτείται, περιέχει όλα τα απαραίτητα στοιχεία γύρω από τα ***Bayesian Networks*** και τα ***Naïve Bayes Classifiers*** μαζί με ένα παράδειγμα για τον καθένα. Επιπλέον περιέχει μια ανάλυση του αλγορίθμου ***Forward Sampling*** και της τεχνικής του ***Laplace Smoothing***. Τέλος περιγράφει το αλγόριθμο ***Maximum Likelihood Estimate(MLE)*** που χρησιμοποιούμε για την εκμάθηση των παραμέτρων και το μοντέλο το οποίο εστιάζουμε το οποίο είναι το ***Distributed Continuous Model***.

Το ***Κεφάλαιο 3*** περιέχει τον ορισμό του προβλήματος το οποίο εστιάζουμε μαζί με την γενική προσέγγιση που ακολουθείται και από τις δυο προσεγγίσεις. Το ***Κεφάλαιο 4*** περιέχει την ανάλυση του προβλήματος, συγκεκριμένα περιέχει την ανάλυση της πρώτης προσέγγισης δηλαδή το να χρησιμοποιούμε ***approximate distributed counters*** τα οποία τα αντιμετωπίζουμε ως ***ξεχωριστές οντότητες*** και τέλος περιέχει την δεύτερη προσέγγιση δηλαδή την προσέγγιση που προτείναμε και βασίζεται στο ***Functional Geometric Monitoring(FGM)***. Αντίστοιχα το ***Κεφάλαιο 5*** περιέχει την σχεδίαση και την υλοποίηση του συστήματος που προτείναμε ενώ το ***Κεφάλαιο 6*** περιέχει την πειραματική αξιολόγηση και των δυο προσεγγίσεων από διαφορετικές οπτικές. Τέλος το ***Κεφάλαιο 7*** περιέχει τα συμπεράσματα και τις μελλοντικές επεκτάσεις του συστήματος μας.

Κεφάλαιο 2:

Θεωρητικό Υπόβαθρο

2.1 Σημειογραφία

Αρχικά το μεγαλύτερος μέρος της σημειογραφίας(*notation*) που χρησιμοποιούμε, βασίζεται στο [1].

Καθόλη την έκταση της παρούσας εργασίας όλες οι *τυχαίες μεταβλητές(random variables)* που θα χρησιμοποιηθούν θα συμβολίζονται με κεφαλαία λατινικά γράμματα X, Y, Z ενώ όταν πρόκειται για τις τιμές τους θα χρησιμοποιούνται τα αντίστοιχα λατινικά γράμματα x, y, z. Επιπλέον θα χρησιμοποιούμε το $Val(X)$ για να δηλώσουμε το σύνολο των τιμών της τυχαίας μεταβλητής X. Συνήθως επί τον πλείστων αναφερόμαστε σε *categorical(discrete)* τυχαίες μεταβλητές επομένως όταν θέλουμε να αναφερθούμε(ή να απαριθμήσουμε) στις τιμές της τυχαίας μεταβλητής X θα χρησιμοποιούμε τον ακόλουθο συμβολισμό $x^1, \dots, x^k$ αν υποθέσουμε ότι η τυχαία μεταβλητή X περιέχει k διαφορετικές τιμές. Τέλος $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ υποδηλώνει ότι έχουμε ένα σύνολο από τυχαίες μεταβλητές ενώ $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ υποδηλώνει τις τιμές για το σύνολο των μεταβλητών, αντίστοιχα μπορούμε να ορίσουμε $Val(\boldsymbol{X})$ ώστε πλέον να αναφέρεται στο σύνολο των τιμών των μεταβλητών του συνόλου $\boldsymbol{X}$.

Επιπροσθέτως η συνάρτηση $|\cdot|$ θα συμβολίζει την πληθικότητα της εκάστοτε συνάρτησης-σχέσης εκτός αν και ορίζεται διαφορετικά. Για παράδειγμα $|Val(X)|$ ισοδυναμεί με τον αριθμό των τιμών που μπορεί να πάρει η τυχαία μεταβλητή X ή ισοδύναμα με τον αριθμό των στοιχείων που περιέχονται στο σύνολο τιμών της τυχαίας μεταβλητής X. Επιπλέον σε πολλές περιπτώσεις "γράφουμε" $P(x)$ αντί $P(X = x)$ δεδομένου ότι το x αποτελεί τιμή που αναφέρεται στην τυχαία μεταβλητή X. Επίσης θεωρούμε ότι $P((X = x) \cap (Y = y))$ ισοδυναμεί με $P(X = x, Y = y)$ ή $P(x, y)$ ή $\mathrm{P}(X, Y)$.

Τέλος αν υποθέσουμε ότι έχουμε ένα σύνολο από τυχαίες μεταβλητές $\boldsymbol{X} = \{X_1, \dots, X_n\}$ τότε το *joint probability distribution* πάνω στο σύνολο $\boldsymbol{X}$ ισοδυναμεί με $P(X_1, \dots, X_n)$ ή $P(\boldsymbol{X})$ ενώ αντίστοιχα η κατανομή $P(X_i)$ αναφέρεται στο *marginal distribution* της τυχαίας μεταβλητής X_i .Επιπροσθέτως όταν θέλουμε να αναφερθούμε σε *conditional probability distributions(CPDs)* χρησιμοποιούμε τον ακόλουθο συμβολισμό $P(X \mid Y)$ με το X, Y να αποτελούν δυο οποιεσδήποτε τυχαίες μεταβλητές. Επιπλέον χρησιμοποιούμε το ξ για να αναφερθούμε σε ένα πλήρης στιγμιότυπο των τιμών των μεταβλητών ενός οποιουδήποτε σύνολο τυχαίων μεταβλητών $\boldsymbol{X}$.

Δυο σημαντικοί "κανόνες" που χρησιμοποιούνται σε αρκετές περιπτώσεις και αξίζει να σημειωθούν είναι ο *Chain Rule* [1] και ο *Bayes Rule* [2]. Αρχικά και οι δύο κανόνες βασίζονται στην ευρέως γνωστή "*υπό-συνθήκη*" πιθανότητα (*conditional probability*). Παρακάτω βλέπουμε τον ορισμό από το *conditional probability*.

Αν υποθέσουμε ότι α, β δυο οποιαδήποτε γεγονότα τότε η "*υπό-συνθήκη*" πιθανότητα του β δεδομένου του α δίνεται από τον παρακάτω τύπο :

$$Conditional\ Probability:\ P(\beta \mid \alpha) = \frac{P(\alpha \cap \beta)}{P(\alpha)}$$

Παρατηρούμε ότι "*υπό-συνθήκη*" πιθανότητα ορίζεται μόνον όταν το $P(\alpha) > 0$.

Όσον αφορά τον πρώτο κανόνα δηλαδή τον κανόνα *Chain Rule,* αν υποθέσουμε ότι έχουμε στην διάθεση μας μια ακολουθία από γεγονότα έστω $\alpha_1, \dots, \alpha_\kappa$,τότε ισχύει ότι :

$$Chain\ Rule: P(a_1 \cap \dots \cap a_k) = P(a_1) \cdot P(a_2|a_1) \cdots P(a_k \mid a1 \cap \dots \cap a_{k-1})$$

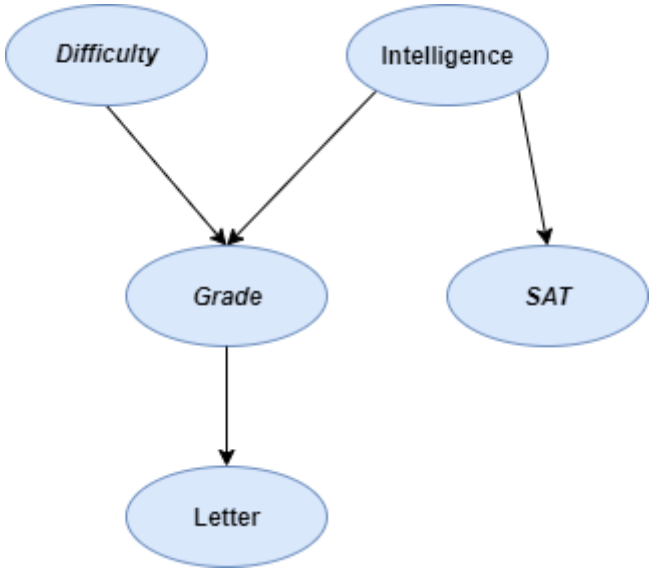
Επιπλέον αν θεωρήσουμε ότι η ακολουθία από γεγονότα αποτελείται από γεγονότα τα οποία είναι ανεξάρτητα μεταξύ τους δηλαδή ισχύει ότι $P(a_i \cap a_j) = P(a_i) \cdot P(a_j)$ για κάθε $i, j \in \{1, \dots, \kappa\}$ τότε προκύπτει ότι:

$$P(a_1 \cap \dots \cap a_k) = P(a_1) \cdot P(a_2) \cdots P(\alpha_\kappa)$$

Τώρα όσον αφορά τον δεύτερο κανόνα δηλαδή τον κανόνα *Bayes' Rule* ο οποίος αποτελεί άμεση συνέπεια από τον ορισμό του *conditional probability* , αν υποθέσουμε δυο οποιαδήποτε γεγονότα α, β (με το β συχνά να αναφέρεται ως *evidence)* τότε ισχύει ότι:

$$Bayes'\ Rule: P(\alpha \mid \beta) = \frac{P(\beta \mid \alpha) \cdot P(a)}{P(\beta)}$$

Γράφοι(Graph)



Εικόνα 1: Directed Acyclic Graph(DAG) από το Student Bayesian δίκτυο

Η ενότητα αυτή περιέχει τους βασικούς συμβολισμούς που απαιτούνται και χρησιμοποιούνται καθόλη την διάρκεια της εργασίας και απευθύνεται στο κομμάτι των γράφων. Περισσότερες λεπτομέρειες παρουσιάζονται στο [1, Ch. 2].

Το κυρίαρχο *data structure* που χρησιμοποιείται καθόλη την έκταση της παρούσας εργασίας αντιστοιχεί στην ευρέως γνωστή δομή του *γράφου(graph).* Κάθε γράφος αποτελείται από ένα σύνολο *κόμβων(nodes)* $\mathcal{V}$ και από ένα σύνολο *ακμών(edges)* $\mathcal{E}$ και από εδώ και στο εξής θα χρησιμοποιούμε τον ακόλουθο συμβολισμό $\mathcal{G}(\mathcal{V}, \mathcal{E})$ ή $\mathcal{G}(\mathcal{X}, \mathcal{E})$, δηλαδή για έναν γράφο

$\mathcal{G}$ για τον οποίο το σύνολο των κόμβων ισοδυναμεί με το $\mathcal{V} = \{V_1 , \dots , V_n\}$ ενώ αντίστοιχα το σύνολο των ακμών να αντιστοιχεί με το $\mathcal{E} = \{E_1, \dots , E_n\}$.

Συγκεκριμένα εστιάζουμε σε ένα συγκεκριμένος είδος της προηγούμενης δομής και αυτό είναι ο *Directed Acyclic Graph(DAG)*. Επομένως σε αυτήν την περίπτωση ένα ζευγάρι κόμβων $V_i, V_j \in \mathcal{V}$ μπορεί να συνδέεται αποκλειστικά και μόνο μέσω ενός *directed edge* $V_i \rightarrow V_j$, με το σύνολο των ακμών $\mathcal{E}$ ουσιαστικά να περιέχει όλα τα δυνατά ζευγάρια κόμβων που συνδέονται μεταξύ τους. Επιπλέον δεδομένου ότι πρόκειται για ένα *acyclic* γράφο ισχύει ότι ο γράφος δεν περιέχει κανένα *directed path* $V_1, \dots , V_k$ όπου το $V_1 = V_k$, δηλαδή δεν παρουσιάζει κάποιο *loop*.

Αν υποθέσουμε ότι έχουμε στην διάθεση μας ένα *directed acyclic graph* $\mathcal{G}(\mathcal{X}, \mathcal{E})$ για οποιαδήποτε ακμή $X_i \rightarrow X_j \in \mathcal{E}$ με $X_i, X_j \in \mathcal{X}$, ισχύει ότι ο κόμβος X_j είναι *child* του κόμβου X_i και θα χρησιμοποιούμε το συμβολισμό $Child(X_j)$ για εκφράσουμε των σύνολο των κόμβων οι οποίοι αποτελούν *child* κόμβους για τον κόμβο X_j. Αντίστοιχα το X_i αποτελεί *parent* κόμβο του X_j και θα χρησιμοποιούμε το συμβολισμό $Par(X_i)$ για εκφράσουμε των σύνολο των κόμβων οι οποίοι αποτελούν *parents* κόμβους για τον κόμβο X_i. Τέλος θα χρησιμοποιούμε τον συμβολισμό $NonDescedants(X_i)$ για να εκφράσουμε το σύνολο των κόμβων που δεν είναι *descendants* κόμβοι από τον κόμβο X_i.

Η *Εικόνα 1* αποτελεί ένα παράδειγμα από ένα *DAG* με το σύνολο των κόμβων $\mathcal{V}$ να ισοδυναμεί με :

$$\mathcal{V} = \{Difficulty, Intelligence, Grade, SAT, Letter\}$$

Ενώ το σύνολο των ακμών $\mathcal{E}$ να αντιστοιχεί με:

$$\mathcal{E} = \{\, Difficulty \rightarrow Grade \,, Intelligence \rightarrow Grade \,,\ Intelligence \rightarrow SAT \,, Grade \rightarrow Letter \,\}$$

Επιπλέον μπορούμε να πούμε ότι για τον κόμβο $Grade$ ισχύουν τα ακόλουθα:

$$\mathrm{P}ar(Grade) = \{Difficulty, Intelligence\}$$

$$\mathrm{C}hild(Grade) = \{Letter\}$$

$$NonDescedants(Grade) = \{SAT\}$$

Απόδοση ξενόγλωσσων όρων

Να αναφέρουμε ότι με τους όρους *events,observations,data* εννοούμε τα *δεδομένα* που χρησιμοποιούνται ενώ με τον όρο *data streams* αναφερόμαστε σε *ροές δεδομένων*. Επιπλέον με τον όρο *accurate approximation* αναφερόμαστε σε μια "*ακριβείς" προσέγγιση* ενώ με τον όρο *joint probability distribution* αναφερόμαστε στην "*από κοινού" κατανομή* των εκάστοτε μεταβλητών. Επιπλέον με τον όρο *maintenance* αναφερόμαστε στην έννοια της "διατήρησης" ενώ με τον όρο *graphical structure* αναφερόμαστε σε μια δομή που βασίζεται σε κάποιο γράφο. Επιπροσθέτως με τον όρο *randomized counter* αναφερόμαστε σε τυχαιοκρατικούς μετρητές, με τον όρο *deterministic counter* αναφερόμαστε σε ντετερμινιστικούς μετρητές ενώ με τον όρο *approximate distributed counters* αναφερόμαστε σε *προσεγγιστικούς κατανεμημένους μετρητές*. Τέλος με τον όρο *distributed continuous model* αναφερόμαστε στο "*συνεχές" κατανεμημένο μοντέλο*.

2.2 Bayesian Networks

2.2.1 Εισαγωγή στα Bayesian Networks

Τα *Bayesian Networks* [3] αποτελούν ένα παράδειγμα σύζευξης μεταξύ του *graph theory* και του *probability theory* με σκοπό να μας βοηθήσουν να μοντελοποιήσουμε *probabilistic* και *causal* συσχετίσεις που εμφανίζονται σε *real-world* εφαρμογές [4].

Τα *Bayesian Network* μπορούμε να τα συναντήσουμε σε πολλά *πεδία*, μπορούμε να τα συναντήσουμε σε *decision-support systems*, συγκεκριμένα μπορούμε να τα δούμε ως *sensor validations systems* [5], όπου η βασική ιδέα στηρίζεται στο γεγονός ότι έχουμε στην διάθεση μας ένα σύνολο από μετρήσεις των αισθητήρων(*vector sensor readings – τα οποία αναπαρίστανται μέσω του Bayesian δικτύου*) και προσπαθούμε να εντοπίζουμε αν κάποιος αισθητήρας εμφανίζει κάποια βλάβη που έχει ως αποτέλεσμα να επηρεάζεται η συνολική λειτουργία του συστήματος που είναι ενσωματωμένος.

Επιπλέον μπορούμε να τα δούμε ως *medical diagnosis* συστήματα [4, Ch.2] , όπου ως σκοπό έχουν την διάγνωση κάποιας ασθένειας δεδομένου ενός συνόλου από "*ευρήματα*" του εκάστοτε ασθενή. Η βασική ιδέα παρουσιάζεται παρακάτω:

$$diagnosis = max_i P(D_i|E)$$

Ουσιαστικά το $P(D_i|E)$ είναι η πιθανότητα από το *disease* D_i δεδομένου του συνόλου E(*evidence)*, το οποίο αντιστοιχεί στο σύνολο που αντιπροσωπεύει τα *observed findings* του ασθενή, όπως για παράδειγμα συμπτώματα που εμφανίζει καθώς και αποτελέσματα εργαστηριακών εξετάσεων του ασθενή. Τέτοια παραδείγματα αποτελούν το *MUNIN* [6] το οποίο πρόκειται για ένα σύστημα διάγνωσης νευρομυϊκών διαταραχών ή το *HEPAR2* [7] το οποίο πρόκειται για σύστημα διάγνωσης διαταραχών του ήπατος.

Επιπροσθέτως μπορούμε να τα συναντήσουμε στο τομέα του *cybersecurity* [8]. Σε αυτήν την περίπτωση το *Bayesian Network* μπορούμε να το δούμε ως ένα *real-time security analysis* σύστημα όπου καθώς το σύστημα δέχεται δεδομένα προσπαθεί να εντοπίσει αν πρόκειται για είσοδο η οποία είναι επικίνδυνη(*malicious*) *ή* όχι(benign).

Τέλος μια ειδική κατηγορία από τα *Bayesian Networks* την οποία μπορούμε να την εντάξουμε στην κατηγορία των *Classifiers*, πρόκειται για τα γνωστά *Naïve Bayes Classifiers*(*Βλ. Κεφάλαιο 2.3*). Αντίστοιχα και τα *Naïve Bayes Classifiers* βρίσκουν εφαρμογή σε πολλές περιπτώσεις (παραθέτουμε μερικά από τα πιο γνωστά πεδία εφαρμογής τους), όπως για παράδειγμα μπορούμε να τα χρησιμοποιήσουμε ως *Documents Classifiers* [4, Ch.11] ή ακόμα και ως *Information Retrieval* συστήματα[4, Ch.12].

Καταρχάς τα "γραφικά" μοντέλα (*graphical models)* τα οποία εστιάζουμε ανήκουν στην κατηγορία των *probabilistic graphical models.* Συγκεκριμένα επικεντρωνόμαστε σε μια ευρέως γνωστή κατηγορία και δεν είναι άλλη από τα γνωστά *Bayesian Networks*. Όπως είπαμε και προηγουμένως ανήκουν στην κατηγορία των *graphical models* επομένως το βασικό χαρακτηριστικό τους είναι ότι η αναπαράσταση τους βασίζεται σε "*γράφους*"(Βλ. *Κεφάλαιο 2.1*) και συγκεκριμένα σε κάποιο *directed acyclic graph(DAG)* , όπου με την χρήση τους αυτό που επιτυγχάνουμε είναι να έχουμε μια πιο συμπαγής αναπαράστασης της

κατανομής μεταξύ όλων των τυχαίων μεταβλητών που υπάρχουν στο δίκτυο (*joint probability distribution*), όπου ορίζεται συνήθως σε κάποιο *high-dimensional* χώρο πιθανοτήτων.

Ουσιαστικά την αναπαράσταση μέσω του γράφου μπορούμε να την δούμε από δυο πτυχές . Η πρώτη πτυχή λοιπόν αναφέρεται στο γεγονός ότι την αναπαράσταση μέσω του γράφου μπορούμε να την δούμε ως ένα σύνολο από *independencies* μεταξύ του συνόλου των διαθέσιμων κόμβων που υπάρχουν στο δίκτυο ενώ η δεύτερη πτυχή αναφέρεται στο γεγονός ότι χρησιμοποιώντας τον γράφο μπορούμε να "χωρίσουμε" την αρχική κατανομή σε μικρότερα κομμάτια(*factors*) , τα οποία ορίζονται σε πολύ μικρότερο χώρο πιθανοτήτων. Με αυτόν τον τρόπο καταφέρνουμε να αποφύγουμε τον αρχικό *high-dimensional* χώρο πιθανοτήτων από το *joint probability distribution*.

Probability queries

Στόχος μας λοιπόν πέρα από την εκμάθηση των παραμέτρων(*learning parameters – Βλ. Κεφάλαιο 2.5*) που ορίζουν το *joint probability distribution* που αντιστοιχεί στο εκάστοτε *Bayesian Network* , είναι να μπορούμε να εκτιμήσουμε *ερωτήματα(queries*) που αφορούν τα *distributions* των τυχαίων μεταβλητών που έχουμε στην διάθεση μας. Η κατηγορία των *queries* που εστιάζουμε αναφέρεται ως *probability queries*. Τα *queries* αυτής της κατηγορίας αποτελούνται από δυο μέρη :

- Το πρώτο μέρος είναι το *evidence* κομμάτι, το οποίο αποτελεί ένα υποσύνολο $\boldsymbol{E}$ από τυχαίες μεταβλητές του μοντέλου μαζί με ένα στιγμιότυπο $\boldsymbol{e}$ των τιμών αυτών των μεταβλητών. Εμείς εστιάζουμε σε *queries* όπου το *evidence* ισοδυναμεί με το κενό σύνολο δηλαδή ισχύει ότι $\boldsymbol{E} = \emptyset$.
- Το δεύτερο μέρος αναφέρεται σε *queries variables*, το οποίο στην γενική περίπτωση αποτελείται από ένα υποσύνολο $\boldsymbol{Y}$ τυχαίων μεταβλητών του δικτύου. Συνήθως επικεντρωνόμαστε σε *queries* όπου τα *queries variables* αναφέρονται σε όλο το σύνολο των τυχαίων μεταβλητών του δικτύου και όχι σε κάποιο υποσύνολο δηλαδή σε *queries* που αφορούν το *joint probability distribution* του δικτύου.

$$Probabilty\ queries : P(\boldsymbol{Y} \mid \boldsymbol{E} = \boldsymbol{e})$$

Επομένως στόχος μας είναι να μπορούμε να εκτιμήσουμε *probability queries* πάνω στο *joint probability distribution* , υποθέτοντας επιπροσθέτως ότι το $\boldsymbol{E} = \emptyset$. Σε τέτοια *queries* θα αναφερόμαστε ως:

$$P(\boldsymbol{Y}) = P(Y_1, \dots, Y_n)$$

Αν υποθέσουμε τώρα ότι έχουμε ένα σύνολο από τυχαίες μεταβλητές $\boldsymbol{X} = \{X_1, \dots, X_n\}$ με την καθεμία να αποτελεί μια *binary-valued* τυχαία μεταβλητή (για παράδειγμα η κάθε τυχαία μεταβλητή X_i θα μπορούσε να εκφράζει το αποτέλεσμα της ρίψης ενός νομίσματος) και ως στόχο θέλουμε να εκτιμήσουμε το *joint probability distribution* $P(X_1, \dots, X_n)$ που ορίζεται από το σύνολο των τυχαίων μεταβλητών $\boldsymbol{X}$. Παρατηρούμε ακόμα και στην πιο απλή περίπτωση ότι για το καθορισμό του *joint probability distribution* απαιτούνται $2^n - 1$ αριθμοί , δηλαδή τους αριθμούς που αντιστοιχούν στις πιθανότητες των 2^n διαφορετικών εκχωρήσεων τιμών $x_1, \dots, x_n$ των τυχαίων μεταβλητών του συνόλου $\boldsymbol{X}$. Ακόμα και στην περίπτωση όπου το σύνολο των τυχαίων μεταβλητών είναι μικρό, βλέπουμε ότι η

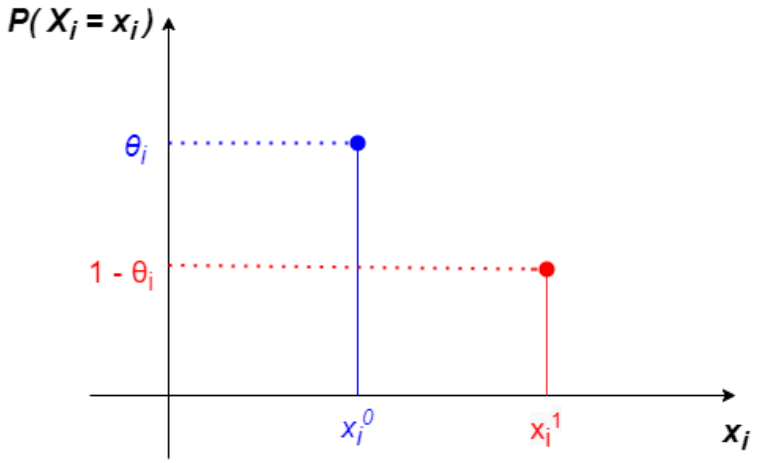
"διαχείριση" του *joint probability distribution* καθίσταται αδύνατη από κάθε άποψη είτε αναφερόμαστε στο χώρο(*heap space*) που χρειάζεται να αποθηκεύσουμε όλες τους διαφορετικούς αριθμούς που καθορίζουν την *joint* κατανομή είτε στο μέγεθος των δεδομένων(*datasets*) που απαιτούνται για να μπορέσουμε να εκτιμήσουμε τις τόσες διαφορετικές παραμέτρους που προσδιορίζουν το *joint probability distribution* , αφού παρατηρούμε ότι έχουμε *εκθετική εξάρτηση(exponential dependence)* στο αριθμό των τυχαίων μεταβλητών.

Τώρα αν υποθέσουμε ότι το σύνολο $\boldsymbol{X}$ που έχουμε στην διάθεση μας αποτελείται από τυχαίες μεταβλητές που είναι ανεξάρτητες(*marginally independent*) μεταξύ τους δηλαδή ισχύει ότι οι κατανομές από οποιοδήποτε ζεύγος τυχαίων μεταβλητών X_i, X_j είναι ανεξάρτητες ($X_i \perp X_j$) τότε χρησιμοποιώντας το *Chain Rule*(*Βλ. Κεφάλαιο 2.1*) μπορούμε να εκφράσουμε το *joint probability distribution* ως:

$$P(X_1, \dots, X_n) = P(X_1) \cdot \dots \cdot P(X_n)$$

Παρατηρούμε ότι κάνοντας χρήση της ιδιότητας του *independence* μεταξύ των τυχαίων μεταβλητών μπορούμε πλέον να προσδιορίσουμε το *joint probability distribution* χρησιμοποιώντας αποκλειστικά και μόνο τα *marginal distribution* $P(X_i)$ των τυχαίων μεταβλητών του $\boldsymbol{X}$. Εκμεταλλευόμενοι το γεγονός αυτό, αν υποθέσουμε ότι θ_{X_i} αντιστοιχεί στη παράμετρο που καθορίζει το *marginal probability distribution* $P(X_i)$ της *binary-valued* τυχαίας

$$Marginal\ probability\ distribution: P(X_i = x_i) = \theta_{x_i} = \begin{cases} \theta_i\,, & \text{όταν } x_i = x_i^1 \\ 1 - \theta_i\,, & \text{όταν } x_i = x_i^0 \end{cases}$$



Εικόνα 2: Συνάρτηση κατανομή πιθανότητας(pmf) από μια binary-valued τυχαία μεταβλητή X_i

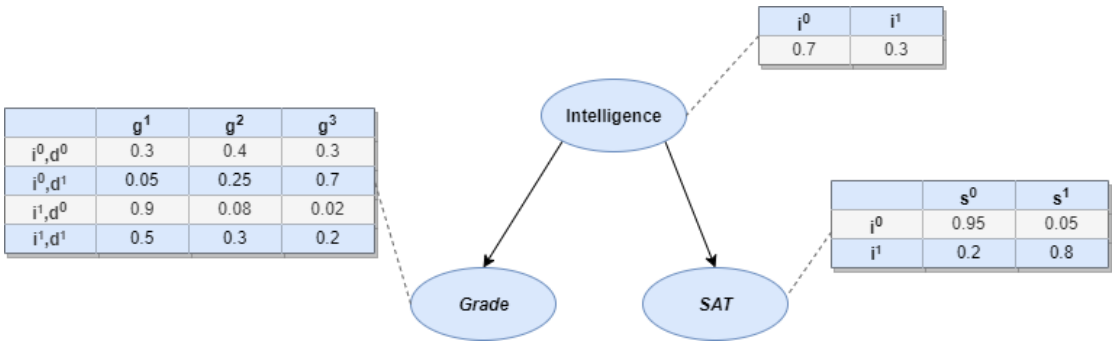
μεταβλητής X_i, πλέον το μόνο που χρειαζόμαστε για να προσδιορίσουμε το *joint probability distribution* δεν είναι τίποτα άλλο από τις παραμέτρους $\theta_{X_1}, \dots, \theta_{X_n}$ των κατανομών των τυχαίων μεταβλητών $X_1, .., X_n$. Επομένως ισχύει ότι:

$$P(x_1, \dots, x_n) = \prod_{i=1}^{n} \theta_{X_i}$$

Αυτό το οποίο καταφέραμε με αυτόν τον τρόπο είναι ότι από τον αρχικό χώρο που απαιτούνταν για το *joint probability distribution* ο οποίος αντιστοιχούσε σε κάποιο

υποχώρο(subspace) του $\mathbb{R}^{2^n}$, πλέον το μόνο που χρειαζόμαστε να είναι ένα *n-dimensional manifold* στο $\mathbb{R}^{2^n}$. Επιπλέον με αυτόν τον τρόπο επιτυγχάνουμε να έχουμε και *γραμμική εξάρτηση(linear dependence)* στο αριθμό των τυχαίων μεταβλητών.

Βέβαια αυτό το παράδειγμα αποτελεί μια ιδανική κατάσταση, αλλά ταυτόχρονα δύσκολα να μπορέσουμε να την συναντήσουμε σε *real-world* εφαρμογές , γιατί τις περισσότερες φόρες υπάρχει εξάρτηση μεταξύ των διαφόρων τυχαίων μεταβλητών που έχουμε στην διάθεση μας. Μια εναλλακτική κατάσταση που εμφανίζεται πιο "λογική" με την έννοια ότι μπορεί να προσομοιώσει σε ικανοποιητικό βαθμό αρκετές περιπτώσεις *real-world* εφαρμογών είναι να θεωρήσουμε *conditional independences* μεταξύ των διαθέσιμων τυχαίων μεταβλητών. Ουσιαστικά τα *conditional independences* μαζί με έναν *Directed Acyclic Graph(DAG)* είναι τα δυο πράγματα τα οποία ορίζουν έναν *Bayesian Network*.



Εικόνα 3: Κομμάτι από τον γράφο του Student Bayesian Network

Αν υποθέσουμε ότι έχουμε στην διάθεση το *Bayesian Network* της *Εικόνας 3* . Παρατηρούμε λοιπόν ότι το δίκτυο αποτελείται από τους κόμβους *Intelligence(I),Grade(G),SAT(S)*. Συγκεκριμένα η τυχαία μεταβλητή I μπορεί να πάρει δύο τιμές $Val(I) = \{i^1, i^0\}$, όπου το i^1 αντιστοιχεί σε *high intelligence students* ενώ το i^0 αντιστοιχεί σε *low intelligence students*. Παρομοίως η τυχαία μεταβλητή S μπορεί να πάρει δυο τιμές $Val(S) = \{s^1, s^0\}$ με το s^1 να αντιστοιχεί σε *high score* ενώ το s^0 σε *low score*. Τέλος η τυχαία μεταβλητή G μπορεί να πάρει τρεις τιμές $Val(G) = \{g^1, g^2, g^3\}$ με το g^1, g^2, g^3 να ισοδυναμούν με τους βαθμούς A, B, C αντιστοίχως. Παρατηρούμε λοιπόν ότι για το *joint probability distribution* των τριών μεταβλητών $P(I, G, S)$ απαιτούνται 12($2 \cdot 2 \cdot 3$) παράμετροι στην περίπτωση που απαριθμήσουμε όλους τους δυνατούς συνδυασμούς τιμών των τριών μεταβλητών.

Επιπλέον το γεγονός του καθολικού *independence* μεταξύ των μεταβλητών δεν μπορεί να εφαρμοστεί γιατί πλέον μπορούμε να παρατηρήσουμε *dependencies* μεταξύ των μεταβλητών (για παράδειγμα ο βαθμός κάποιου μαθητή σε κάποιο μάθημα επηρεάζεται ως έναν βαθμό και από *intelligence* του). Συγκεκριμένα υποθέτουμε ότι έχουμε το ακόλουθο *conditional independency* : $(S \perp G \mid I)$, δηλαδή η τυχαία μεταβλητή *Grade(G)* είναι ανεξάρτητη από την τυχαία μεταβλητή *SAT(S)* δεδομένου της τυχαίας μεταβλητής *Intelligence(I).*

Με βάση το τελευταίο *conditional independency* μπορούμε να εκφράσουμε το *joint probability distribution* με τον ακόλουθο τρόπο:

$$P(I, G, S) = P(S, G \mid I) \cdot P(I) = P(S \mid I) \cdot P(G \mid I) \cdot P(I)$$

Παρατηρούμε λοιπόν ότι ακόμα και με την χρήση ενός *conditional independency* , μπορούμε να χωρίσουμε το *joint probability distribution* σε μικρότερα ανεξάρτητα κομμάτια(*factors*) το

οποίο συμβάλει στο να έχουμε μια πιο *compact* αναπαράσταση από το αρχικό *distribution*. Συνέπεια αυτού είναι πλέον ότι ο αριθμός για τον προσδιορισμό του *joint distribution* είναι μικρότερος, πλέον χρειαζόμαστε 7 παραμέτρους (η διαφορά δεν είναι τόσο μεγάλη, άλλα η διαφορά μεταξύ τους αυξάνεται όσο αυξάνεται ο αριθμός τόσο των μεταβλητών όσο και των *conditional independencies* που υπάρχουν, το παράδειγμα είναι ενδεικτικό), συγκεκριμένα χρειαζόμαστε μια παράμετρο για την κατανομή $P(I)$, δυο για την κατανομή $P(S)$ και τέλος τέσσερις παραμέτρους για την κατανομή $P(G)$.

Τέλος ο διαχωρισμός του *joint probability distribution* σε μικρότερα ανεξάρτητα κομμάτια(*factors*) εμφανίζει την ιδιότητα του *modularity*, δηλαδή σε περίπτωση προσθήκης ή αφαίρεσης κάποιος τυχαίας μεταβλητής επηρεάζεται μόνο ένα συγκεκριμένο κομμάτι(*factor*) σε αντίθεση με την περίπτωση που έχουμε την "απευθείας" προσέγγιση τότε θα έπρεπε να απαριθμήσουμε ξανά όλους τους δυνατούς συνδυασμούς των τιμών του καινούργιου συνόλου των τυχαίων μεταβλητών. Επιπλέον όπως θα δούμε και στην συνέχεια τα μικρότερα ανεξάρτητα κομμάτια(*factors) του joint probability distribution* αποτελούν το κύριο χαρακτηριστικό ώστε ο αλγόριθμος του *Maximum Likelihood Estimation*(*MLE – Βλ. Κεφάλαιο 2.5*) να μπορεί να "εφαρμοστεί" σε *distributed* περιβάλλον(*Βλ. Κεφάλαιο 2.7*).

Συνοψίζοντας μπορούμε να πούμε ότι η αναπαράσταση του *Bayesian Network* μέσω ενός *directed acyclic graph(DAG)* μας προσφέρει δύο πλεονεκτήματα:

- Πρώτον μας προσφέρει μια συμπαγής αναπαράσταση του συνόλου των *conditional independences* μεταξύ των τυχαίων μεταβλητών που ορίζουν το *joint probability distribution*.
- Δεύτερον μας παρέχει ένα "μηχανισμό" για να μπορέσουμε να έχουμε μια πιο συμπαγής αναπαράσταση του *joint probability distribution(factors),* δηλαδή μέσω της *παραγοντοποίησης(factorization)* του *joint probability* καταφέρνουμε από τον αρχικό *high-dimensional* χώρο να μεταβούμε σε ένα γινόμενο από *παράγοντες(factors)* που αντιστοιχούν σε *lower-dimensional* υποχώρους.

2.2.2 Bayesian Network Semantics

Παρακάτω παραθέτουμε όλους τους ορισμούς γύρω από τα *Bayesian Networks* που χρησιμοποιούνται στην συνέχεια.

Ουσιαστικά *ένα Bayesian Network structure* $\mathcal{G}(\mathcal{X}, \mathcal{E})$ δεν είναι τίποτα άλλο από έναν *directed acyclic graph*(*DAG – Βλ. Κεφάλαιο 2.1*), με τους *κόμβους(nodes)* να αντιστοιχούν στο σύνολο των τυχαίων μεταβλητών $\mathcal{X} = \{\mathrm{X}_1, \dots, X_n\}$ και το σύνολο από τα *directed edges* $\mathcal{E}$ να αντιστοιχεί στα *conditional independences* μεταξύ των κόμβων. Συγκεκριμένα ένα οποιοδήποτε *directed edge* υποδηλώνει ότι υπάρχει *direct dependence* μεταξύ των τυχαίων μεταβλητών που το απαρτίζουν. Αν υποθέσουμε ότι $Par(X_i)$ υποδηλώνει το σύνολο των *parents* κόμβων που αντιστοιχούν στο κόμβο X_i και $NonDescedants(X_i)$ να υποδηλώνει το σύνολο των κόμβων που δεν είναι *descendants* κόμβοι από τον κόμβο X_i , τότε το $\mathcal{G}$ μπορεί να "εκφράσει" το σύνολο των *conditional independences*, τα οποία τα αποκαλούμε *local independences* και συμβολίζουμε με το $I_\ell(\mathcal{G})$, με τον ακόλουθο τρόπο:

$$Για\ κάθε\ μεταβλήτη\ X_i\ ισχύει\ οτι: (\ X_i\ \perp NonDescendants(X_i) \mid Par(X_i)\ )$$

Με το τελευταίο να υποδηλώνει ότι κάθε κόμβος X_i είναι *conditionally independent* από τους τους *non-descendants* κόμβους τους δεδομένου των *parents* κόμβων του, δηλαδή κάθε κόμβος εξαρτάται άμεσα μόνο από τους *parents* κόμβους του. Επιπλέον το σύνολο $I_\ell(\mathcal{G})$ ισοδυναμεί με το σύνολο των *directed edges* του δικτύου $\mathcal{G}(\mathcal{X}, \mathcal{E})$.

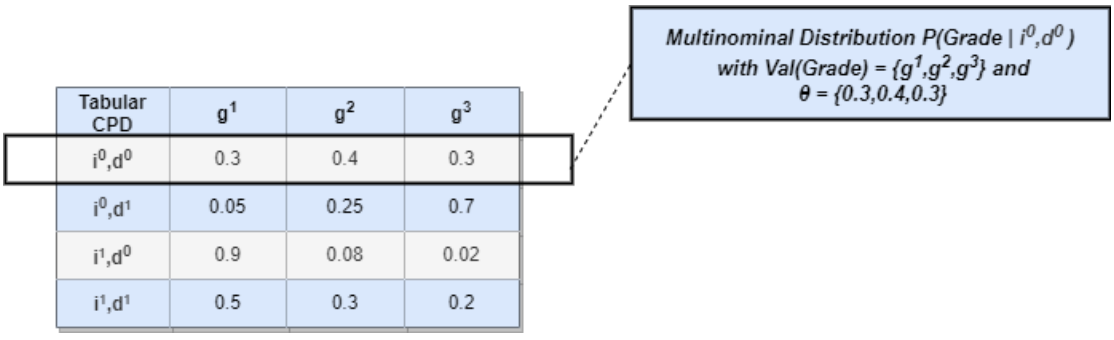
Δεδομένου ενός *Bayesian Network structure* $\mathcal{G}(\mathcal{X}, \mathcal{E})$ και το αντίστοιχου συνόλου $I_\ell(\mathcal{G})$ μπορούμε να εκφράσουμε το *joint probability distribution* $P(\mathcal{X})$ με τον ακόλουθο τρόπο:

$$P(X_1, \dots, X_n) = \prod_{i=1}^{n} P(\, X_i \mid Par(X_i)\,)$$

Η τελευταία ισότητα αντιστοιχεί σε αυτό το οποίο αποκαλούμε *chain rule από τα Bayesian Networks*. Επιπλέον ο κάθε παράγοντας του γινομένου $P(\, X_i \mid Par(X_i)\,)$ αντιστοιχεί σε αυτό το οποίο αποκαλούμε *local conditional probability distributions(CPDs)*.

Οι τυχαίες μεταβλητές που χρησιμοποιούνται στην παρούσα εργασία ανήκουν στην κατηγορία των *categorical-discrete* τυχαίων μεταβλητών με συνέπεια τα *conditional probability distributions(CPDs)* των μεταβλητών να ανήκουν στην γνωστή κατηγορία των *Tabular CPDs*. Ουσιαστικά το κάθε *CPD* $P(\, X_i \mid Par(X_i)\,)$ μπορούμε να το αναπαραστήσουμε ως έναν *πίνακα(table)* με το κάθε στοιχείο να ισοδυναμεί με την πιθανότητα που αντιστοιχεί σε έναν συνδυασμό των τιμών του X_i και του $Par(X_i)$.(*Βλ. Εικόνα 4*)

Επομένως την κάθε γραμμή από το *Tabular CPD* $P(\, X_i \mid Par(X_i) = \boldsymbol{x_i}\,)$ μπορούμε να το δούμε ως μια *multinomial κατανομή* πάνω σε $k = Val(X_i)$ *καταστάσεις(states)* με $\boldsymbol{x_i} = \in Val(Par(X_i))$ και τις αντίστοιχες πιθανότητες(παραμέτρους) $p_i \in [0,1]$ *για κάθε* $i \in Val(X_i)$ *και* $\sum_{i=1}^{Val(X_i)} p_i = 1$. Ουσιαστικά για κάθε γραμμή του *Tabular CPD* χρειαζόμαστε έναν *k-dimensional vector* παραμέτρων για να μπορέσουμε να προσδιορίζουμε την αντίστοιχη *multinomial* κατανομή. Θεωρούμε λοιπόν ότι το θ_{X_i} ισοδυναμεί με το σύνολο των παραμέτρων που απαιτούνται για τον προσδιορισμό όλων των *multinomial* κατανομών(για κάθε γραμμή) που αντιστοιχούν στο CPD $P(\, X_i \mid Par(X_i)\,)$ από την τυχαία μεταβλητή X_i. Επιπλέον θεωρούμε ότι το σύνολο $\theta = \{\theta_{X_1}, \dots, \theta_{X_n}\}$ αντιστοιχεί στις παραμέτρους από όλα τα *CPDs* του *Bayesian* δικτύου $\mathcal{G}(\mathcal{X}, \mathcal{E})$, δηλαδή σε όλες τις παραμέτρους που χρειαζόμαστε για εκτιμήσουμε το *joint probability distribution* του δικτύου.



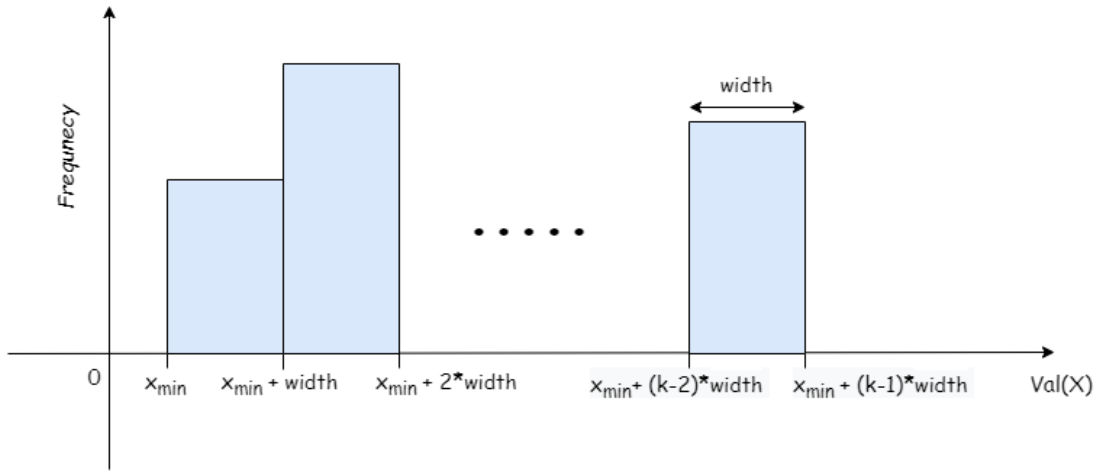
Tabular CPD	g^1	g^2	g^3
i^0,d^0	0.3	0.4	0.3
i^0,d^1	0.05	0.25	0.7
i^1,d^0	0.9	0.08	0.02
i^1,d^1	0.5	0.3	0.2

Εικόνα 4: Tabular CPD του κόμβου Grade

Τέλος αν υποθέσουμε ότι ένα *Bayesian Network* $\mathcal{G}(\mathcal{X}, \mathcal{E})$, όπου το σύνολο $\mathcal{X}$ αποτελείται από n *binary* τυχαίες μεταβλητές τότε για την "απευθείας" προσέγγιση του *joint probability distribution* απαιτούνται $2^n - 1$ παράμετροι. Στην περίπτωση όμως που παραγοντοποιήσουμε το *joint probability distribution* σύμφωνα με το $\mathcal{G}$ και υποθέσουμε ότι κάθε κόμβος έχει το πολύ k parents τότε μπορούμε να αποδείξουμε ότι ο αριθμός των παραμέτρων που απαιτούνται είναι λιγότερος από $n \cdot 2^k$. Παρατηρούμε λοιπόν ότι πρώτον καταφέραμε να αποφύγουμε την εκθετική εξάρτηση από τον αριθμό των κόμβων n, το οποίο θα ήταν και προβληματικό για μεγάλες τιμές του n και δεύτερον καταφέραμε να έχουμε εκθετική εξάρτηση με βάση το k , το οποίο αποτελεί πλεονέκτημα αφού τις περισσότερες φορές η κάθε τυχαία μεταβλητή συσχετίζεται με έναν μικρό αριθμό μεταβλητών του δικτύου και γενικά ισχύει ότι $n \ll k$. Η τελευταία ιδιότητα αποτελεί ένα από κύρια οφέλη του *Bayesian Network* αφού επιτυγχάνουμε να έχουμε *εκθετικά(exponentially)* μικρότερο αριθμό παραμέτρων για το *joint probability distribution*.

Διακριτοποίηση(Discretization)

Στην περίπτωση τώρα που στο *Bayesian Network* υπάρχουν *continuous* τυχαίες μεταβλητές, τότε το πρώτο πράγμα το οποίο θα κάνουμε για να μπορέσουμε να τις διαχειριστούμε είναι να μετατρέψουμε τις *continuous* μεταβλητές σε *discrete*. Αυτό μπορεί να γίνει μέσω της γνωστής μεθόδου του *discretization.* Ουσιαστικά βλέπουμε την διαδικασία αυτήν ως ένα *pre-processing* βήμα που εφαρμόζεται με βάση τις πληροφορίες που διαθέτουμε για τη εκάστοτε τυχαία μεταβλητή, προτού συνεχίζουμε στην εκμάθηση των παραμέτρων(*learning parameters*) του *joint probability distribution*. Για να γίνει αυτό υπάρχουν πολλοί μέθοδοι οι οποίοι παρατίθενται αναλυτικά στο [9], αλλά εστιάζουμε και υποστηρίζουμε σε μια συγκεκριμένη μέθοδο. Συγκεκριμένα υποστηρίζουμε την μέθοδο του *equal width binning*(*Εικόνα 5*) το οποίο αντιστοιχεί στο γνωστό *equal width ιστόγραμμα(histogram).*



Εικόνα 5: Equal width binning

Η βασική ιδέα του *equal width binning* είναι να χωρίζουμε το εύρος τιμών της τυχαίας μεταβλητής σε k ίσου μεγέθους διαστήματα(*buckets*) ,με το k να αποτελεί *user-input* παράμετρο. Προφανώς και αυτή η μέθοδος είναι "ευάλωτη" όταν πρόκειται για *skewed* κατανομές , βέβαια υπάρχουν και πιο *advanced* τεχνικές που το επιλύουν αλλά αποτελεί κόμματι εκτός ορίων της παρούσας εργασίας.

Τώρα αν υποθέσουμε ότι έχουμε μια *continuous* τυχαία μεταβλητή X με $Val(X) \in [x_{min}, x_{max}]$ και k να αντιστοιχεί στο αριθμό των *buckets*, τότε το *width* από το κάθε *bucket* ισούται με :

$$width = \frac{x_{max} - x_{min}}{k}$$

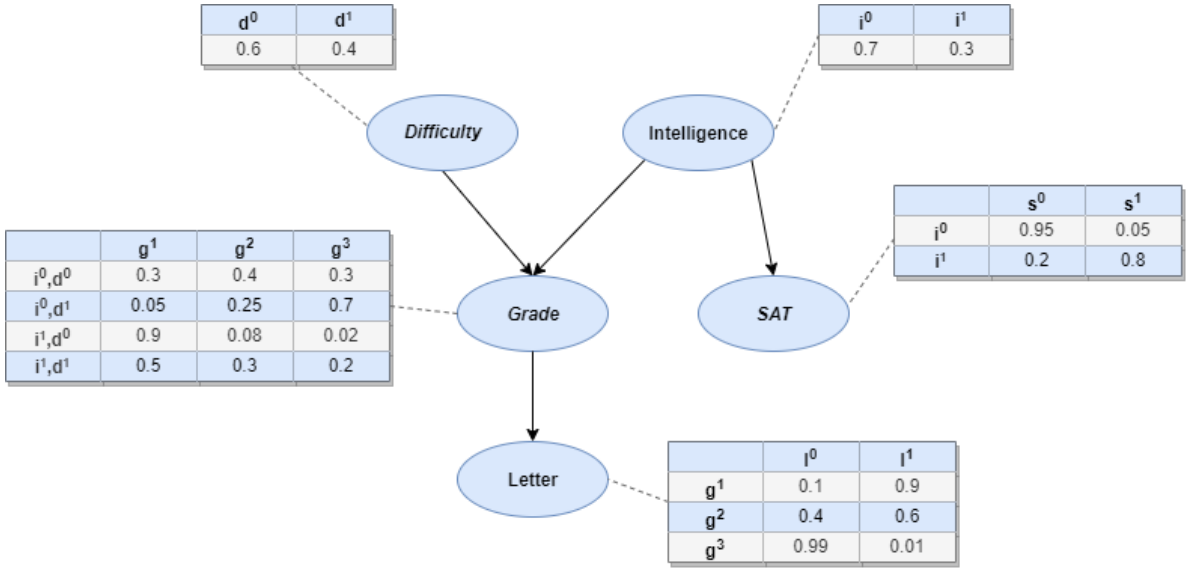
Ενώ τα όρια από τα *buckets*(*bucket boundaries*) που θα δημιουργηθούν, καθορίζονται με τον ακόλουθο τρόπο:

$$x_{min} + i \cdot width \; με \; το \; i = 1, \dots, k-1$$

Πρόκειται για μέθοδο που ανήκει στην *unsupervised* κατηγορία δηλαδή δεν λαμβάνει υπόψιν τα διαθέσιμα *class labels* που υπάρχουν. Επιπλέον ανήκει στην κατηγορία των *global* μεθόδων δηλαδή τα *partitions(buckets)* που δημιουργούνται αφορούν ολόκληρο το *continuous space* που ορίζει η τυχαία μεταβλητή και είναι ανεξάρτητα των υπόλοιπων μεταβλητών. Τέλος ανήκει στην κατηγορία των *static* μεθόδων δηλαδή απαιτούν κάποιο *user-input* το οποίο υποδηλώνει και τον μέγιστο αριθμό από *buckets-partitions* που θα δημιουργηθούν.

Τέλος να επισημάνουμε ότι την μέθοδο του *discretization* δεν την βλέπουμε συνδυαστικά, για παράδειγμα σε συνδυασμό με το *Bayesian Structure Learning* , άλλα όπως αναφέραμε και προηγουμένως την βλέπουμε αποκλειστικά ως ένα *pre-processing* βήμα.

2.2.3 *Student Bayesian Network*



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	g^1	g^2	g^3
i^0, d^0	0.3	0.4	0.3
i^0, d^1	0.05	0.25	0.7
i^1, d^0	0.9	0.08	0.02
i^1, d^1	0.5	0.3	0.2

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^3	0.99	0.01

Εικόνα 6: Student Bayesian Network μαζί με τα CPDs

Θεωρώντας όλα τα προηγούμενα πλέον μπορούμε να δούμε ένα ολοκληρωμένο παράδειγμα από ένα *Bayesian Network*. Πρόκειται για το *Student Bayesian Network*(*Εικόνα 6*). Το *Student Network* $\mathcal{G}(\mathcal{X}, \mathcal{E})$ αποτελείται από πέντε κόμβους: την *"εξυπνάδα"* του μαθητή (I), την δυσκολία του μαθήματος(D), το βαθμό του(G), την *SAT βαθμολογία* του(S) και τέλος την "ποιότητα" της συστατικής επιστολής(L), συγκεκριμένα ισχύει ότι:

$$\mathcal{X} = \{Difficulty(D), Intelligence(I), Grade(G), SAT(S), Letter(L)\}$$

Επιπροσθέτως πρόκειται για ένα δίκτυο όπου όλες οι μεταβλητές είναι *binary-valued* εκτός από την τυχαία μεταβλητή $Grade(G)$ η οποία είναι *ternary-valued*. Επιπλέον το σύνολο των *local conditional independences* $I_l(\mathcal{G})$ που ορίζεται με βάση το $\mathcal{G}(\mathcal{X}, \mathcal{E})$ φαίνεται παρακάτω :

$$Για\ τους\ κόμβους\ Intelligence, Difficulty : (I \perp G)$$

$$Για\ τον\ κόμβο\ Letter : (L \perp I, D, S \mid G)$$

$$Για\ τον\ κόμβο\ Grade : (G \perp S \mid I, D)$$

$$Για\ τον\ κόμβο\ SAT : (S \perp D, G, L \mid I)$$

Επομένως με βάση το $\mathcal{G}(\mathcal{X}, \mathcal{E})$ και το σύνολο $I_l(\mathcal{G})$ μπορούμε να εκφράσουμε το *joint probability distribution* που αντιστοιχεί στο $\mathcal{G}$ με τον ακόλουθο τρόπο(*chain rule από το Bayesian Network*) :

$$P(I, D, S, G, L) = P(I) \cdot P(D) \cdot P(G \mid I, G) \cdot P(S \mid I) \cdot P(L|G)$$

Παρατηρούμε λοιπόν ο αριθμός των παραμέτρων θ για τον προσδιορισμό του *joint probability distribution* ισούται με 15. Συγκεκριμένα χρειαζόμαστε μια παράμετρο για τις *binomial* κατανομές των κόμβων $Intelligence, Difficulty$, 2 παραμέτρους για κάθε μια από τις τέσσερις *multinomial* κατανομές του κόμβου $Grade$, μια παράμετρο για κάθε μια από τις δυο *binomial* κατανομές του κόμβου SAT και τέλος μια παράμετρο για κάθε μια από τις τρεις *binomial* κατανομές του κόμβου $Letter$. Σε αντίθεση με την περίπτωση όπου είχαμε την "απευθείας" προσέγγιση από το *joint probability distribution* θα χρειαζόμασταν 47 παραμέτρους $(2^4 \cdot 3)$.

Παράδειγμα από probability queries του Student Bayesian Network

Σκοπός μας είναι να μπορούμε να εκτιμήσουμε *probability queries* πάνω στο *joint probability distribution.* Επομένως με βάση το *Student Example* θα μπορούσε ένα *probability query* να έχει την μορφή που φαίνεται παρακάτω :

$$P(i^1, d^0, g^2, s^1, l^0) = P(i^1) \cdot P(d^0) \cdot P(g^2) \cdot P(s^1) \cdot P(l^0) = 0.004$$

Πρόκειται για ένα *query* το οποίο εκφράζει την πιθανότητα ότι πρόκειται για "έξυπνο" μαθητή; με την πιθανότητα του μαθήματος να είναι εύκολη; με την πιθανότητα ότι ένας "έξυπνος" μαθητής σε ένα εύκολο μάθημα να πάρει B; με την πιθανότητα ότι ένας "έξυπνος" μαθητής να έχει υψηλό *SAT score* και τέλος με την πιθανότητα ότι ένας μαθητής που έχει πάρει B στο μάθημα να πάρει μια "*weak*" συστατική επιστολή.

2.3 Naïve Bayes Classifiers

2.3.1 Εισαγωγή στα Naïve Bayes Classifiers

Αρχικά, πέραν από τα *Bayesian Networks* , εστιάζουμε σε μια ειδική κατηγορία τους και αυτήν δεν είναι άλλη από την ευρέως γνωστή κατηγορία των *Naïve Bayes* [10]. Συγκεκριμένα εστιάζουμε στην περίπτωση από τα *Naïve Bayes Classifier*. Για την δημιουργία των *Naïve Bayes Classifier* έχουμε την σύζευξη και πάλι του *graph theory* αλλά και του *probabilistic theory*. Συγκεκριμένα το *classification* σε αυτήν την περίπτωση γίνεται μεσώ του γνωστού κανόνα *Bayes Rule* όπου αναλύεται λεπτομερώς παρακάτω.

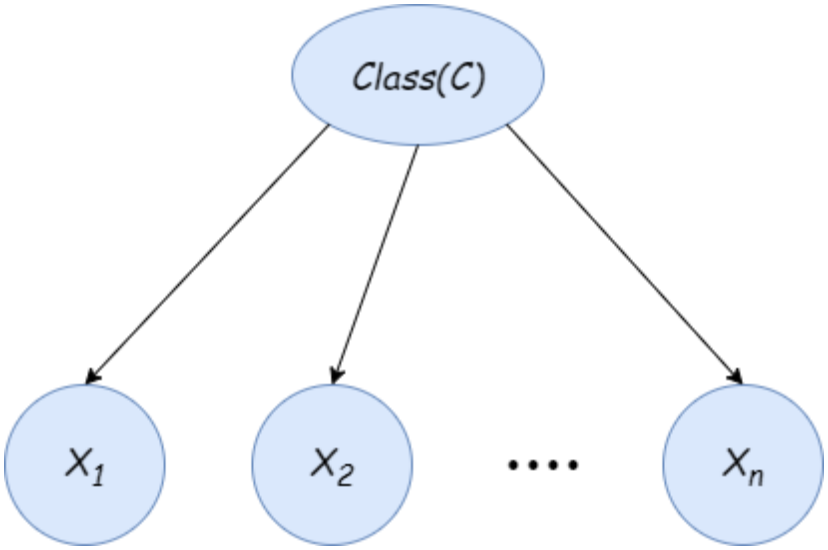
Τα *Naïve Bayes Classifiers* μπορούμε να τα συναντήσουμε σε πολλές *real word* εφαρμογές. Αρχικά μπορούμε να χρησιμοποιήσουμε ως *Documents Classifier* το οποίο αναλύεται και στην συνέχεια, κυρίως όμως τα εντοπίζουμε ως *spam filtering* από *emails*. Επιπλέον μπορούν να χρησιμοποιηθούν ως *Medical Diagnosis* συστήματα είτε ως *Credit-Card Fraud Detection* συστήματα αλλά ακόμα και στο κομμάτι του *Sentiment Analysis*. Τέλος μπορούμε να το δούμε και ως *recommender* συστήματα [11], όπου στην συγκεκριμένη περίπτωση το βλέπουμε σε συνδυασμό με την μέθοδο του *collaborative filtering*, καταφέρνοντας να επιτύχουν ένα *hybrid recommender* σύστημα το όποιο είναι *scalable* και αποδίδει καλύτερα(*accuracy*) σε σχέση τα υπόλοιπα συστήματα. Στην συγκεκριμένη περίπτωση *ο Naïve Bayes Classifier* λειτουργεί όπως και στην περίπτωση των *Documents Classifiers* μόνο που το σύνολο των *class labels* αντιστοιχεί σε ένα σύνολο από *recommendations.*

Επομένως όταν πρόκειται για την δημιουργία των *classifiers*, επικεντρωνόμαστε στην δημιουργία μια συνάρτησης που θα εκχωρεί κάποιο *class label* στα *instances* και αυτό γίνεται λαμβάνοντας υπόψιν το διαθέσιμο σύνολο χαρακτηριστικών(*attributes*) που αφορά τα διάφορα *instances*. Ουσιαστικά εστιάζουμε σε αυτό το οποίο αποκαλούμε *Classification*. Τέλος θεωρούμε ότι τόσο το σύνολο των τιμών τόσο των *attributes* όσο και του σύνολο των *class labels* είναι γνωστά εκ των προτέρων, δηλαδή βρισκόμαστε σε *supervised* περιβάλλον.

2.3.2 Naïve Bayes Classifiers Semantics

Η βασική ιδέα από τους *Naïve Bayes Classifier,* είναι ότι δημιουργούν έναν *classifier* όπου "μαθαίνει" τα *CPDs* από κάθε *feature* X_i δεδομένου του *class variable* C $P(X_i|C)$, υποθέτοντας ταυτόχρονα ότι το κάθε *feature* X_i είναι ανεξάρτητο των υπολοίπων δεδομένου του *class variable* C*(naïve Bayes assumption)*. Το *classification* γίνεται με την εφαρμογή του *Bayes Rule* και συγκεκριμένα επιλέγουμε εκείνο το *class label* c^i το οποίο έχει την μεγαλύτερη *posterior probability* δεδομένου του *input feature vector* $\boldsymbol{x}$, $P(C = c^i \mid \boldsymbol{x})$.

Παρακάτω παραθέτουμε όλους τους ορισμούς γύρω από τα *Naïve Bayes Classifiers* που χρησιμοποιούνται στην συνέχεια.



Εικόνα 7: Naïve Bayes Classifier

Ουσιαστικά *ένα Naïve Bayes Classifiers structure* $\mathcal{G}(\mathcal{X}, \mathcal{E})$ δεν είναι τίποτα άλλο από έναν *directed acyclic graph*(*Εικόνα 7*) με τους *κόμβους(nodes)* να αντιστοιχούν στο σύνολο των τυχαίων μεταβλητών $\mathcal{X} = \{\mathrm{X}_1, \dots, X_n, C\}$ και με το σύνολο από τα *directed edges* $\mathcal{E}$ να αντιστοιχεί στα *conditional independences* μεταξύ των διαθέσιμων κόμβων. Συγκεκριμένα σε αυτήν την περίπτωση μπορούμε να δούμε το σύνολο των μεταβλητών $\mathcal{X}$ ως δύο ξεχωριστά σύνολα. Το πρώτο σύνολο αναφέρεται στο σύνολο των *"χαρακτηριστικών"*(*features*) και ισοδυναμεί με το υποσύνολο $\{\mathrm{X}_1, \dots, X_n\} \subseteq \mathcal{X}$ (αν υποθέσουμε ότι αριθμός των *features* ισούται με n) , ενώ το δεύτερο σύνολο αναφέρεται στο *class variable* $C \subseteq \mathcal{X}$, όπου το σύνολο των τιμών του εκφράζει το σύνολο των *class labels* και ισοδυναμεί με το σύνολο $Val(C) = \{c^1, \dots, c^k\}$ αν υποθέσουμε ότι ο αριθμός των *class labels* ισούται με k.

Μόνο που σε αυτή την περίπτωση ισχύει αυτό το οποίο αποκαλούμε *naive Bayes assumption*, σύμφωνα με το οποίο ισχύει ότι τα *features* είναι *conditionally independent* δεδομένου του *class label.* Επιπλέον ισχύει ότι $Par(X_i) = C$ για κάθε *feature* X_i με $i \in \{1, .., n\}$. Επομένως το σύνολο των *conditional independences* $I_\ell(\mathcal{G})$ μπορούμε να το εκφράσουμε με τον ακόλουθο τρόπο :

$$\textit{Για κάθε μεταβλητή } X_i \textit{ ισχύει ότι : } (X_i \perp \boldsymbol{X_{-i}} \mid C)$$

$$\textit{όπου το σύνολο } \boldsymbol{X_{-i}} = \{X_1, .., X_n\} - \{X_i\}.$$

Δεδομένου ενός *Naïve Bayes Classifiers structure* $\mathcal{G}(\mathcal{X}, \mathcal{E})$ και το αντίστοιχου συνόλου $I_\ell(\mathcal{G})$ μπορούμε να εκφράσουμε το *joint probability distribution* $P(\mathcal{X})$ με τον ακόλουθο τρόπο:

$$P(C, X_1, \dots., X_n) = P(C) \cdot \prod_{i=1}^{n} P(X_i \mid C)$$

Ο κάθε παράγοντας του γινομένου $P(X_i \mid C)$ αντιστοιχεί σε αυτό το οποίο αποκαλούμε *local conditional probability distributions (CPDs)* από το *feature* X_i. Παρατηρούμε με βάση την προηγούμενη ισότητα , όπως και στην περίπτωση των *Bayesian Networks*, μπορούμε να εκφράζουμε το *joint probability distribution* ως γινόμενο παραγόντων(*factorization*),

συγκεκριμένα αποτελείται από το *prior distribution* $P(C)$ και από ένα σύνολο από *CPDs* $P(X_i|C)$ (με το καθένα στην γενική περίπτωση να αντιστοιχεί σε κάποια *multinomial κατανομή*). Με αυτό τον τρόπο επιτυγχάνουμε να μειώσουμε δραστικά τον αριθμό των παραμέτρων που απαιτούνται για το *joint probability distribution*.

Οι τυχαίες μεταβλητές που χρησιμοποιούνται και σε αυτήν την περίπτωση ανήκουν στην κατηγορία των *categorical-discrete* τυχαίων μεταβλητών με συνέπεια τα *conditional probability distributions(CPDs)* των μεταβλητών να ανήκουν στην γνωστή κατηγορία των *Tabular CPDs*. Σε περίπτωση *continuous μεταβλητών* χρησιμοποιούμε την μέθοδο του *discretization(Βλ. Κεφάλαιο 2.2)*. Επομένως εστιάζουμε και μπορούμε να υποστηρίξουμε *Multinomial Naïve Bayes Classifier*.

Ουσιαστικά το κάθε *CPD* $P(X_i \mid C)$ μπορούμε να το αναπαραστήσουμε ως έναν *πίνακα(table)* με το κάθε στοιχείο να ισοδυναμεί με την πιθανότητα που αντιστοιχεί σε έναν συνδυασμό των τιμών του X_i και του *class variable* C. Τέλος το κάθε *CPD* χαρακτηρίζεται από ένα σύνολο παραμέτρων θ_{X_i}.

Αν υποθέσουμε τώρα ότι έχουμε έναν *Naïve Bayes Classifier* που αποτελείται από n *binary-valued features* και το *class variable* είναι επίσης *binary-valued* , τότε μπορούμε να αποδείξουμε ότι ο αριθμός των *independent* παραμέτρων από το *factorized joint probability distribution* ισούται με $2n + 1$, δηλαδή επιτυγχάνουμε να έχουμε γραμμική εξάρτηση στο αριθμό των *features* n σε αντίθεση με την "απευθείας" προσέγγιση του *joint probability distribution* όπου έχουμε εκθετική εξάρτηση στο αριθμό των *features* ($2^n - 1$ παραμέτρους). Επομένως παρά τις ισχυρές υποθέσεις που ισχύουν για το σύνολο των *features*, εξαιτίας της απλότητας αλλά και της γραμμικής εξάρτησης στο σύνολο των *features* και κατά συνέπεια στο μικρό αριθμό παραμέτρων που απαιτούνται, μπορεί να εφαρμοστεί σε περιπτώσεις όπου έχουμε *high dimensional vector* από *features.*

Επιπλέον έχει παρατηρηθεί παρά την απλότητα και των *strong assumptions* που προδιαθέτει, παρόλο που στις περισσότερες περιπτώσεις είναι μη ρεαλιστικά, λειτουργούν αρκετά καλά συγκρίνοντας ακόμα και με πιο πολύπλοκους *classifier* όπως το *C4.5* όπως παρατίθεται αναλυτικά στο [12]. Η επιτυχία των *Naïve Bayes Classifiers* όπως παρατίθεται [13], βασίζεται στο γεγονός ότι το *classification error* δεν σχετίζεται απαραίτητα με την "ποιότητα" του *fit* που έχουμε στο *joint probability distribution (*καταλληλόλητα των *conditional independencies)*, δηλαδή *error* στην εκτίμηση του *joint probability* δεν οδηγεί απαραίτητα και σε *classification error*. Ουσιαστικά το *classification* των *class labels* δεν είναι *sensitive* σε *errors* εκτίμησης του *probability* από ένα *instance*.

Τώρα όσον αφορά την διαδικασία του *Classification*, αν υποθέσουμε αρχικώς ότι έχουμε στην διάθεση μας *ένα Naïve Bayes Classifiers structure* $\mathcal{G}(\mathcal{X}, \mathcal{E})$, με το σύνολο των *features* να ισοδυναμεί με $\{X_1, \dots, X_n\} \subseteq \mathcal{X}$ ενώ για το c*lass variable* $C \subseteq \mathcal{X}$ ισχύει ότι $Val(C) = \{c^1, \dots, c^k\}$. Στόχος λοιπόν είναι για ένα οποιοδήποτε *n-dimensional vector* $\boldsymbol{x} = [x^1, \dots, x^n]$ να μπορέσουμε να προσδιορίσουμε το *class label* στο οποίο αντιστοιχεί. Συγκεκριμένα επιλέγουμε εκείνο το *class label* το οποίο έχει την μεγαλύτερη *posteriori probability* δεδομένου του *n-dimensional vector.* Επομένως ισχύει ότι:

$$class\ label = \arg max_{c^i \in Val(C)} P(C = c^i | \boldsymbol{x})$$

Επομένως θέλουμε να μεγιστοποιήσουμε την πιθανότητα $P(C|\boldsymbol{x})$. Χρησιμοποιώντας το *Bayes rule* ισχύει ότι:

$$P\left(C = c^i|\boldsymbol{x}\right) = \frac{P(\boldsymbol{x}|C = c^i) \cdot P\left(C = c^i\right)}{P(\boldsymbol{x})} \; για \; κάθε \; c^i \in Val(C)$$

Παρατηρούμε με βάση την προηγούμενη ισότητα ότι για να προσδιορίζουμε το *class label* αρκεί να μεγιστοποιήσουμε τον αριθμητή του κλάσματος $P(\boldsymbol{x}|C = c^i) \cdot P\left(C = c^i\right)$, αφού η πιθανότητα $P(\boldsymbol{x})$ είναι ίδια για όλα τα *class labels*. Συνεπώς όσον αφορά το $P\left(\boldsymbol{x}\middle|C = c^i\right)$ μπορεί να υπολογιστεί με τον ακόλουθο τρόπο:

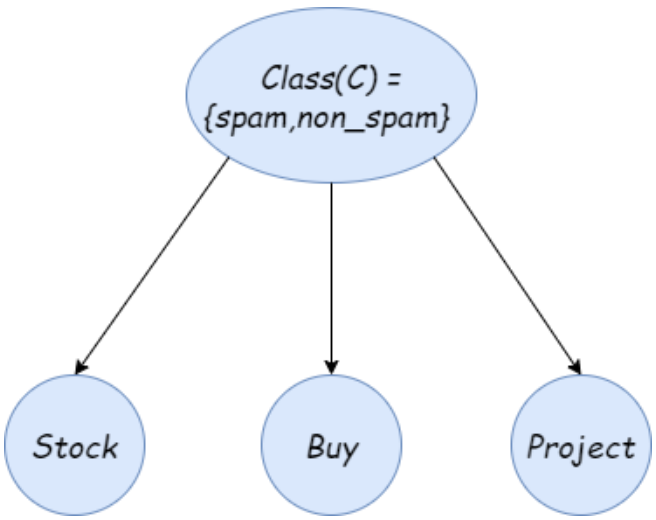
$$Naive\ Bayes\ assumption: \; P\left(\boldsymbol{x}\middle|C = c^i\right) \propto \prod_{i=1}^{n} P\left(\, X_i = x^i \,\middle|\, C = c^i \,\right)$$

Ενώ ο όρος $P\left(C = c^i\right)$ μπορεί να υπολογιστεί με βάση τα δεδομένα που έχουμε στην διάθεση μας. Συγκεκριμένα αν υποθέσουμε ότι χρησιμοποιούμε τον αλγόριθμο *MLE*(*Βλ. Κεφάλαιο 2.5*) μπορούμε να τον υπολογίσουμε με τον ακόλουθο τρόπο:

$$P\left(C = c^i\right) = \frac{count\left(C = c^i\right)}{\sum_{i=1}^{k} count(C = c^i)}$$

2.3.3 Spam classification πρόβλημα

Θεωρώντας όλα τα προηγούμενα πλέον μπορούμε να δούμε ένα ολοκληρωμένο παράδειγμα από ένα *Naïve Bayes Classifier.* Συγκεκριμένα θα εστιάζουμε στην περίπτωση που θέλουμε να εξετάσουμε αν με βάση το περιεχόμενο ενός *email*, πρόκειται για *spam* ή *non-spam(spam filtering)* , όπου το συγκεκριμένο παράδειγμα ανήκει στην ευρύτερη κατηγορία από το *Document Classification*. Επομένως σε αυτήν την περίπτωση θεωρούμε ότι το *class variable* C είναι μια *binary valued* μεταβλητή με σύνολο τιμών $Val(C) = \{spam, non\_spam\}$.



Εικόνα 8: Naïve Bayes Classifier για spam filtering

Επιπλέον το περιεχόμενο από κάθε *email* το βλέπουμε ως ένα σύνολο από λέξεις (*bag of words*), με τις *λέξεις* σε αυτήν την περίπτωση να αποτελούν τα *features*. Στο συγκεκριμένο παράδειγμα υποθέτουμε ότι έχουμε τρία *features* , τα οποία αντιστοιχούν στις λέξεις "Stock","Buy" και "Project" με το καθένα από τα *features* να συνοδεύεται από το αντίστοιχο *CPD* $P(W_i|C)$ *με* $W_i \in \{Stock, Buy, Project\}$. Το *network structure* από τον *Naïve Bayes Classifier* παρουσιάζεται στην *Εικόνα 8*. Στην γενική περίπτωση το σύνολο των *features* δηλαδή οι λέξεις είναι αρκετά μεγάλο λαμβάνοντας υπόψιν αρκετές *λέξεις-κλειδιά* για τον προσδιορισμό του *class label*.

Επομένως κάνοντας χρήση του *naïve Bayes assumption*,μπορούμε να εκφράσουμε το *joint probability distribution* με τον ακόλουθο τρόπο:

$$P(C, Stock, Buy, Project) = P(C) \cdot P(\, Stock \mid C\,) \cdot P(\, Buy \mid C\,) \cdot P(\, Project \mid C\,)$$

Ο σκοπός όπως είπαμε και προηγουμένως είναι να κάνουμε *classified* ένα *email* σε ένα από τα δυο διαθέσιμα *class label*. Αν υποθέσουμε ότι το *email* περιέχει τις λέξεις *stock,buy* και δεν περιέχει την λέξη *project*, δηλαδή το *input feature vector* ισοδυναμεί με $\boldsymbol{x} = [stock, buy, \neg project]$, τότε για να μπορέσουμε να προσδιορίζουμε το *class label* θα πρέπει να υπολογίσουμε τις *posteriori probabilities* δεδομένου του $\boldsymbol{x}$, το οποίο μπορεί να γίνει με τον ακόλουθο τρόπο :

$$Για\ το\ class\ label\ spam:$$

$$\mathrm{P}(spam \mid \boldsymbol{x}) = P(\boldsymbol{x}|spam) \cdot P(spam)$$
$$= P(stock|spam) \cdot P(buy|spam) \cdot P(\neg project|spam) \cdot P(spam)$$

$$Για\ το\ class\ label\ non\_spam:$$

$$\mathrm{P}(non\_spam \mid \boldsymbol{x}) = P(\boldsymbol{x}|\ non\_spam\ ) \cdot P(non\_spam)$$
$$= P(stock|\ non\_spam) \cdot P(buy|non\_spam) \cdot P(\neg project|non\_spam)\ \cdot P(non\_spam)$$

Επομένως οποιαδήποτε από τις πιθανότητες $\mathrm{P}(spam \mid \boldsymbol{x})$ και $\mathrm{P}(non\_spam \mid \boldsymbol{x})$ είναι μεγαλύτερη είναι και αυτή που προσδιορίζει το *class label*.

2.4 Forward Sampling

Σε αυτήν την ενότητα περιγράφουμε τον αλγόριθμο *Forward Sampling* [14] όπου χρησιμοποιήθηκε για την παραγωγή όλων των *training instances* όπου χρησιμοποιήθηκαν για την αξιολόγηση και την εκτέλεση των πειραμάτων όπου αναλύονται στην συνέχεια.

Αρχικά στοχεύομε στην δημιουργία *training instances* τα οποία είναι *full particles* δηλαδή αποτελούν στιγμιότυπα "πλήρης" εκχώρησης τιμών (*full assignments*) όλων των μεταβλητών του *Bayesian* δικτύου $\mathcal{G}(\mathcal{X}, \mathcal{E})$, το οποίο επιβεβαιώνει όπως αναφερθήκαμε και προηγουμένως ότι βρισκόμαστε σε *supervised* "καθεστώς".

Δεύτερον θεωρώντας το *joint probability distribution* $P(\mathcal{X})$ που αντιστοιχεί στο δίκτυο $\mathcal{G}$, μας ενδιαφέρει η εκτίμηση από *probabilities queries* $P(\boldsymbol{Y} = \boldsymbol{y})$, όπου $\boldsymbol{Y} \subseteq \mathcal{X}$ *και* $\boldsymbol{y} \in Val(\mathcal{Y})$. Επομένως δεν εστιάζουμε σε *conditional probability queries* , δηλαδή σε *queries* της μορφής $\boldsymbol{P}(\boldsymbol{Y} = \boldsymbol{y} \,|\boldsymbol{E} = \boldsymbol{e})$ όπου το $\boldsymbol{Y}, \boldsymbol{E} \subseteq \mathcal{X}$, $\boldsymbol{y} \in Val(\boldsymbol{Y})$ και $\boldsymbol{e} \in Val(\boldsymbol{E})$.

Τελικώς αυτό το οποίο θέλουμε είναι *full particles* με σκοπό την εκτίμηση *queries* που αφορούν το *joint probability distribution* επομένως ο αλγόριθμος που επιλέχθηκε είναι ο αλγόριθμος *Forward Sampling*. Σε αντίθεση με άλλους αλγόριθμους όπως το *Rejection Sampling* και *Likelihood Weighting* [15] όπου χρησιμοποιούνται για την δημιουργία *full particles* αλλά εστιάζουν στην εκτίμηση *conditional probability queries*.

Βέβαια υπάρχουν και άλλοι *Deterministic Search* μέθοδοι για την παραγωγή δεδομένων λαμβάνοντας υπόψιν και περιπτώσεις από *highly skewed distributions* , δηλαδή όταν έχουμε ένα μικρό αριθμό από μεταβλητές οπού έχουν μη αμελητέα κατανομή πιθανότητας , κάτι το οποίο θα είναι προβληματικό σε *sampling* μεθόδους όπως είναι και οι προαναφερθείσες μέθοδοι, αφού το σύνολο των *particles* δεν θα είναι αντιπροσωπευτικό(*biased*) του συνόλου των τυχαίων μεταβλητών που υπάρχουν. Βέβαια αυτό αποτελεί κομμάτι εκτός ορίων της παρούσας εργασίας επομένως αποφασίσαμε να χρησιμοποιήσουμε τον αλγόριθμο του *Forward Sampling*.

Η βασική ιδέα του αλγορίθμου είναι πολύ απλή και βασίζεται στην ιδέα ότι για κάθε μεταβλητή δηλαδή για κάθε κόμβο του δικτύου "παράγουμε" δεδομένα τα οποία είναι ανάλογα της συνάρτησης πιθανότητας που έχει. Το πρώτο βήμα για να λειτουργήσει ο αλγόριθμος είναι να δημιουργήσουμε ένα *topological order* του *DAG(Directed Acyclic Graph)* το οποίο αναπαριστά το *Bayesian* δίκτυο $\mathcal{G}$. Η διάταξη των κόμβων που παράγεται από το αλγόριθμο του *topological order*, εξασφαλίζει ότι για οποιανδήποτε κόμβο X_i που προηγείται από έναν οποιανδήποτε κόμβο X_j στην διάταξη , δεν υπάρχει "απευθείας" μονοπάτι από τον X_i στο X_j. Αυτό το βήμα είναι αναγκαίο γιατί χρησιμοποιώντας την διάταξη που προκύπτει από τον αλγόριθμο εξασφαλίζουμε ότι για κάθε κόμβο πρώτα "παράγουμε" δεδομένα που αντιστοιχούν στους *parents* κόμβους(αν υπάρχουν) του και μετέπειτα για αυτόν.

Επομένως με βάση την διάταξη των κόμβων που έχουμε στην διάθεση μας αυτό που κάνουμε για κάθε κόμβο είναι να κάνουμε *sampling* από το αντίστοιχο *CPD* $P(X_i \,|Par(X_i))$ του. Μόλις τελειώσει η διαδικασία για όλους τους κόμβους έχουμε και την παραγωγή του αντίστοιχου *particle*. Παρακάτω βλέπουμε το αλγόριθμο που έχει αναλυθεί προηγουμένως.

Algorithm 1: *Forward Sampling in BNs*

Input:

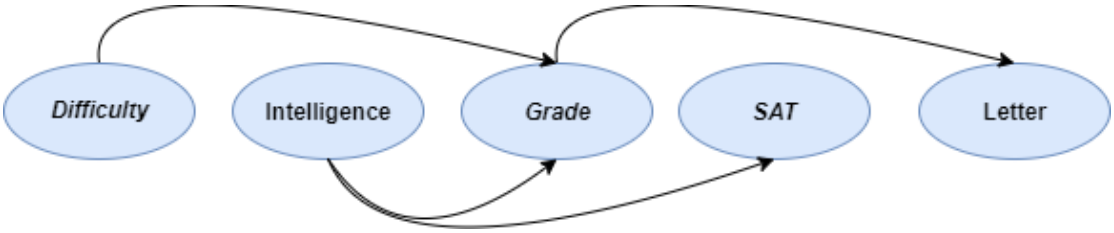
Bayesian Network Structure $\mathcal{G}(\mathcal{X}, \mathcal{E})$ *with* $\mathcal{X} = \{X_1, \ldots, X_n\}$

T *number of full particles*

Output: T full particles

1. *Let* $X_1, \ldots, X_n$ *be a topological ordering of Bayesian Network*
2. **For** 1 **to T**
3. *Initialize an empty instantiation* $\boldsymbol{t}$
4. **Foreach** *node* **from** *topological ordering* $X_1, \ldots, X_n$
5. *Get the values of parents of* X_i *from* $\boldsymbol{t}$
6. *Sample* x_i *from* $P(X_i | Par(X_i))$
7. *Add value* x_i *to* $\boldsymbol{t}$
8. **End**
9. **return** $\boldsymbol{t} = (x_1, \ldots, x_n)$
10. **End**

Τέλος παραθέτουμε την διαδικασία που απαιτείται για την δημιουργία ενός *particle* χρησιμοποιώντας τον αλγόριθμο *Forward Sampling*. Το δίκτυο το οποίο θα χρησιμοποιήσουμε είναι το *Student* (*Βλ. Κεφάλαιο 2.2.3*). Αν υποθέσουμε ότι η διάταξη των κόμβων που δημιουργείται από το αλγόριθμο του *topological order* είναι η ακόλουθη $\mathrm{D}ifficulty(D), Inteligence(I), Grade(G), SAT(S), Letter(L)$ τότε ο αλγόριθμος εκτελείται όπως φαίνεται παρακάτω:



Εικόνα 9: Topological ordering από το Student Bayesian Δίκτυο

1. Αρχικά δειγματολειπτούμε από την κατανομή $P(D)$ για τον κόμβο $\mathrm{D}ifficulty$. Ας υποθέσουμε ότι $\mathrm{D}ifficulty = d^0$.
2. Έπειτα δειγματολειπτούμε από την κατανομή $P(I)$ για τον κόμβο $Inteligence$. Ας υποθέσουμε ότι $Inteligence = i^0$.
3. Στην συνέχεια δειγματολειπτούμε από την κατανομή $P(G | I = i^0, D = d^0)$ για τον κόμβο $Grade$. Ας υποθέσουμε ότι $Grade = g^2$.
4. Ακολούθως δειγματολειπτούμε από την κατανομή $P(S | I = i^0)$ για τον κόμβο SAT. Ας υποθέσουμε ότι $SAT = s^0$.
5. Τέλος δειγματολειπτούμε από την κατανομή $P(L | G = g^2)$ για τον κόμβο $Letter$. Ας υποθέσουμε ότι $Letter = l^0$.
6. Επομένως το αποτέλεσμα που θα πάρουμε είναι $D = d^0, I = i^0, G = g^2, S = s^0, L = l^0$.

Ανάλυση του αλγορίθμου Forward Sampling

Δεδομένου ότι έχουμε ένα σύνολο από *full particles* $D = \{\xi[1], \dots, \xi[M]\}$, το οποίο έχει παραχθεί μέσω του *sampling*, χρησιμοποιώντας *convergence bounds*[1, Ch. 17] μπορούμε να εκτιμήσουμε την μέση τιμή οποιαδήποτε συνάρτησης με τον ακόλουθο τρόπο :

$$E_D(f) = \frac{1}{M}\sum_{m=1}^{M} f(\xi[m]) \;\; \text{όπου } M : \mu\acute{\varepsilon}\gamma\varepsilon\theta o\varsigma \; \alpha\pi o \; \tau o \; \sigma\acute{\upsilon}\nu o\lambda o \; \tau\omega\nu \; particles$$

Στόχος μας όπως είπαμε και προηγουμένως είναι να μπορούμε να εκτιμήσουμε *queries* της μορφής $P(\boldsymbol{Y} = \boldsymbol{y})$, επομένως το μόνο που χρειαζόμαστε είναι να γνωρίζουμε το αριθμό των *particles* όπου περιέχουν το $\boldsymbol{y}$ **,** δηλαδή ισχύει ότι:

$$P_D(\boldsymbol{y}) = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{I}\{y[m] = \boldsymbol{y}\}$$

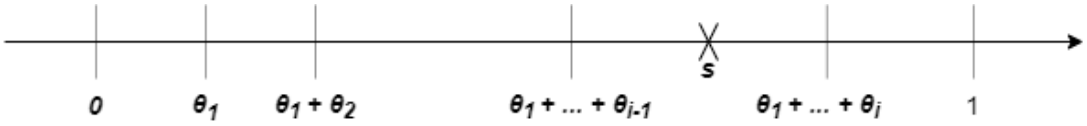
Όπου η συνάρτηση $\boldsymbol{I}$ αντιστοιχεί σε μια "*indicator*" συνάρτηση με τον ακόλουθο τύπο:

$$\begin{cases} 1 \text{ , } \alpha\nu \; \tau o \; y[m] \; \pi\varepsilon\rho\iota\acute{\varepsilon}\chi\varepsilon\iota \; \tau o \; \boldsymbol{y} \\ 0 \text{ , } \delta\iota\alpha\phi o\rho\varepsilon\tau\iota\kappa\acute{\alpha} \end{cases}$$

Τέλος αν θέλουμε να αναφερθούμε στο συνολικό κόστος δηλαδή στην πολυπλοκότητα του αλγορίθμου *Forward Sampling* , αν υποθέσουμε ότι M αντιστοιχεί στον συνολικό αριθμό από *full particles* που θα παραχθούν, $n = |\mathcal{X}|$ αντιστοιχεί στον αριθμό των κόμβων του δικτύου, $p = \max_i |Par(X_i)|$ αντιστοιχεί στο μέγιστο αριθμό από *parents* κόμβους που υπάρχουν μεταξύ των κόμβων του δικτύου και $d = \max_i |Val(X_i)|$ αντιστοιχεί στο μέγιστο αριθμό τιμών που περιέχεται μεταξύ των κόμβων του δικτύου, τότε το συνολικό κόστος δηλαδή η συνολική πολυπλοκότητα του αλγορίθμου αντιστοιχεί σε $O(M \cdot n \cdot p \cdot logd)$, θεωρώντας ότι το *sampling* οποιασδήποτε κατανομής αντιστοιχεί σε $O(logd)$ και ο χρόνος *indexing* που απαιτείται για την ανάκτηση των τιμών από τους *parents* κόμβους οποιασδήποτε κατανομής, χρησιμοποιώντας βέβαια και την κατάλληλη δομή αντιστοιχεί σε $O(p)$.

Δειγματοληψία κατανομής(Sampling distribution)

Προηγουμένως είχαμε αναφερθεί στην δειγματοληψία μιας κατανομής(*sampling distribution*) που αντιστοιχεί σε κάποια μεταβλητή χωρίς να αναφερθούμε σε περαιτέρω λεπτομέρειες. Στο πλαίσιο της παρούσας εργασίας εστιάζουμε σε *categorical* τυχαίες μεταβλητές όμως αντίστοιχα μπορούμε να διαχειριστούμε και *continuous* τυχαίες μεταβλητές χρησιμοποιώντας τον κατάλληλο τρόπο *διακριτοποίησης*. Τώρα αν υποθέσουμε ότι έχουμε μια *multinomial* κατανομή $P(X)$ με τιμές $Val(X) = \{x^i, \dots, x^k\}$, όπου καθορίζονται από τις παραμέτρους $\theta_1, \dots, \theta_k$ αντίστοιχα ,τότε η διαδικασία της δειγματοληψίας(*sampling -- Εικόνα 10*) συνοψίζεται στα τρία παρακάτω βήματα:



Εικόνα 10: Δειγματοληψία multinomial κατανομής

1. Αρχικά παράγουμε ένα δείγμα $\boldsymbol{s}$ ομοιόμορφα στο διάστημα $[0,1]$.
2. Έπειτα χωρίζουμε το διάστημα σε k υπό-διαστήματα: $[0, \theta_1), [\theta_1, \theta_1 + \theta_2)$, … δηλαδή το $\boldsymbol{i}$ υπό-διάστημα θα είναι ίσο με $[\sum_{j=1}^{i-1} \theta_j \ , \sum_{j=1}^{i} \theta_j)$
3. Αν το δείγμα $\boldsymbol{s}$ ανήκει στο $\boldsymbol{i}$ υπό-διάστημα τότε η τιμή που παράγουμε είναι το x^i(το διάστημα μπορούμε να το καθορίσουμε χρησιμοποιώντας την μέθοδο του *binary search* σε χρόνο $O(logk)$).

Ανάλυση ποιότητας εκτίμησης

Η ποιότητα της εκτίμησης που παίρνουμε μέσω της δειγματοληψίας δηλαδή το πόσο κοντά ή μακριά βρίσκεται από την *joint ground truth* κατανομή που προέρχεται από το *Bayesian Network* $\mathcal{G}$ καθορίζεται σε μεγάλο βαθμό από τον αριθμό των *particles* που παράγονται. Σε περίπτωση που θέλουμε να θέσουμε κάποια *error guarantees* της εκτίμησης που παίρνουμε, μπορούμε να το κάνουμε χρησιμοποιώντας το παρακάτω θεώρημα.

Αν υποθέσουμε ότι $P(Y = y)$ αντιστοιχεί στην *joint ground truth* κατανομή δηλαδή στην κατανομή που προέρχεται κάνοντας χρήση των *true values* των παραμέτρων , $P_D(Y = y)$ να αντιστοιχεί στην κατανομή που προέχεται από το σύνολο των *particles* που έχουν παραχθεί, η οποία αντιστοιχεί και στην κατανομή που προκύπτει από την εφαρμογή του αλγορίθμου *MLE* και με το M να αντιστοιχεί στον αριθμό των *particles*.

Θεώρημα 1[1, Corollary 12.2]: Αν υποθέσουμε ότι $M \geq \frac{3\ln\left(\frac{2}{\delta}\right)}{P(y)\varepsilon^2}$ τότε μπορούμε να έχουμε ένα *(ε, δ)-approximation* από την *joint ground truth* κατανομή, δηλαδή ισχύει ότι :

$$(1-\varepsilon) \leq \frac{P_D(y)}{P(y)} \leq (1+\varepsilon) \ με \ πιθανότητα \ 1-\delta$$

Παρατηρούμε λοιπόν ότι για να εξασφαλίζουμε *ε relative error* με *error probability* το πολύ *δ* από την *joint ground truth* κατανομή το μέγεθος των δειγμάτων εξαρτάται λογαριθμικά από την ποσότητα $\frac{1}{\delta}$, τετραγωνικά από την ποσότητα $\frac{1}{\varepsilon}$ και επιπλέον αυξάνεται γραμμικά σε σχέση με την ποσότητα $1/P(y)$. Το πρόβλημα με αυτήν την προσέγγιση είναι ότι σε ορισμένες περιπτώσεις η κατανομή $P(y)$ μπορεί να μην είναι γνωστή εκ των προτέρων επομένως δεν μπορούμε να καθορίζουμε και το μέγεθος του συνόλου των δειγμάτων ώστε να εξασφαλίζουμε μια “καλή” εκτίμηση. Προφανώς υπάρχουν και άλλα *error guarantees* [1, Ch. 17] της εκτίμησης που παίρνουμε αλλά εστιάζουμε κυρίως σε *relative error guarantees*.

Μια επέκταση του παραπάνω θεωρήματος, υποθέτοντας ότι έχουμε γνώση των παραμέτρων από τα *local CPDs* που αντιστοιχούν στο *Bayesian Network* $\mathcal{G}$ είναι το παρακάτω θεώρημα.

Θεώρημα 2[1, Corollary 17.3]: Αν υποθέσουμε ότι $P(\ X_i \mid Par(X_i)\ ) \geq \lambda\ \forall i, X_i, Par(X_i)$ και $M \geq \frac{(1+\varepsilon)^2}{2\lambda^{2(d+1)}\varepsilon^2} \log \frac{nJ^{d+1}}{\delta}$ τότε ισχύει ότι :

$$e^{-n\varepsilon} \leq \frac{P_D(y)}{P(y)} \leq e^{n\varepsilon}\ με\ πιθανότητα\ 1-\delta$$

όπου $J = max_{i=1}^{n} J_i\ με\ J_i = |Val(X_i)|, d = max_{i=1}^{n}|Par(X_i)|$ και $n = |\mathcal{X}|$.

Παρατηρούμε ότι σε αυτήν την περίπτωση ότι ο αριθμός των *particles* που πρέπει να παραχθούν για να εξασφαλίζουμε μια "καλή" εκτίμηση πέραν από την εξάρτηση από το ε, δ εξαρτάται και από τα στοιχεία J, d, n τα οποία λαμβάνουν υπόψιν την δομή που έχει το *Bayesian Network* $\mathcal{G}$.

Επομένως σε περίπτωση που θέλουμε να ελέγξουμε την ορθότητα αλλά και την "ποιότητα" της εκτίμησης που παίρνουμε, μπορούμε να περιορίσουμε όταν αυτό είναι εφικτό τις τιμές από το *joint truth probability* των *queries* που χρησιμοποιούμε (για παράδειγμα μπορούμε να ορίζουμε ότι $P(Y) \geq 0.1$ για όλα τα *queries*) και με αυτόν τον τρόπο να καθορίζουμε το κατάλληλο μέγεθος του συνόλου των *particles* ώστε να μπορούμε να ελέγξουμε την κατανομή.

2.5 Εκμάθηση παραμέτρων

Στόχος μας λοιπόν είναι εκμάθηση των παραμέτρων(*learning parameters*) από το *joint probability distribution* το οποίο αντιστοιχεί σε κάποιο *Bayesian Network Structure* $\mathcal{G}(\mathcal{X}, \mathcal{E})$. Επομένως θεωρούμε γνωστό εκ των προτέρων το *network structure*, και στην συνέχεια δεδομένου ενός συνόλου δεδομένων D προσπαθούμε να μάθουμε τις αντίστοιχες παραμέτρους. Ο αλγόριθμος που χρησιμοποιούμε και αναλύεται παρακάτω είναι ο *Maximum Likelihood Estimation*(*MLE*). Τέλος θεωρούμε ότι το σύνολο δεδομένων που έχουμε στην διάθεση μας αποτελείται από στιγμιότυπα "*πλήρους*" εκχώρησης τιμών (*instantiation* από *full assignments*), δηλαδή βρισκόμαστε σε *supervised* περιβάλλον.

Την γενική ιδέα μπορούμε να το δούμε ως την κατασκευή ενός μοντέλου M, συγκεκριμένα το μοντέλο αντιστοιχεί στην κατηγορία των *parametric models* δηλαδή χαρακτηρίζεται από ένα σύνολο παραμέτρων θ , τέτοιο ώστε το *joint probability distribution* που αντιστοιχεί στο μοντέλο να είναι "κοντά" στο *distribution* με το οποίο έχουν παραχθεί τα δεδομένα. Επομένως η επιλογή των παραμέτρων γίνεται με τέτοιο τρόπο ώστε τα διαθέσιμα *training instances* να έχουν την μεγαλύτερη πιθανότητα , δηλαδή να είναι περισσότερο πιθανά(*most probable*).

2.5.1 Maximum Likelihood Estimation (MLE)

Αρχικά θεωρούμε ότι έχουμε στην διάθεση μας ένα Bayesian Network Structure $\mathcal{G}(\mathcal{X}, \mathcal{E})$, επομένως το σύνολο των τυχαίων μεταβλητών αντιστοιχεί σε $\mathcal{X} = \{X_1, \dots, X_n\}$.

Επιπλέον θεωρούμε ότι έχουμε ένα σύνολο δεδομένων $D = \{\, \xi[1], \dots, \xi[M] \,\}$ (*training dataset*) , το όποιο αποτελείται από *independent* και *identically distributed*(*i.i.d.*) δείγματα τα οποία προήλθαν από το *joint probability distribution* $P(\mathcal{X})$ που αντιστοιχεί στο $\mathcal{G}$ (*Βλ. Κεφάλαιο 2.4*). Επίσης θεωρούμε ότι το $\xi[i]\ για\ κάθε\ i \in \{1, \dots, M\}$ αποτελεί ένα *instantiation* από *full assignments* του συνόλου των τυχαίων μεταβλητών $\mathcal{X}$. Τέλος θέλουμε να προσδιορίσουμε το σύνολο των παραμέτρων θ το οποίο καθορίζει το σύνολο των παραμέτρων που απαιτούνται για τα *CPDs* των τυχαίων μεταβλητών.

Επομένως το επόμενο βήμα είναι να ορίζουμε για το *likelihood function* του συνόλου των παραμέτρων θ δεδομένου του συνόλου δεδομένων D , το οποίο μπορούμε να το κάνουμε τον ακόλουθο τρόπο :

$$likelihood : L(\theta : D) = P(D|\theta)$$

Ενώ αντίστοιχα μπορούμε να ορίζουμε και το *log-likelihood* , που είναι και αυτό το οποίο χρησιμοποιείται τις περισσότερες φορές(συνήθως υπολογίσουμε το $\log \mathrm{P}(\xi|\theta)$ για να αποφύγουμε προβλήματα *overflow*) , με τον ακόλουθο τρόπο:

$$log - likelihood : l(\theta : D) = logP(D|\theta)$$

Επιπλέον αξίζει να σημειωθεί ότι το *negated form* από το *log-likelihood* αντιστοιχεί σε αυτό το οποίο αποκαλούμε *loss function ,* δηλαδή μετράει το *loss* που έχουμε με βάση των σύνολο των παραμέτρων που μάθαμε για ένα οποιοδήποτε *instance* ξ.

$$loss\ function :\ loss(\xi|\theta) = -logP(\xi|\theta)$$

Επιπροσθέτως, δεδομένου ότι το σύνολο $D = \{\, \xi[1], \dots, \xi[M] \,\}$ αποτελείται από *i.i.d instances*, τότε μπορούμε να εκφράσουμε το *likelihood function* με τον ακόλουθο τρόπο:

$$L(\theta : D) = P(D|\theta) = \prod_{i=1}^{M} P(\, \xi[i] : \theta)$$

Τέλος όπως είπαμε και προηγουμένως ενδιαφερόμαστε στο μεγιστοποιήσουμε το *likelihood function* δηλαδή να επιλέξουμε τις παραμέτρους θ που μεγιστοποιούν το *likelihood* function. Αν υποθέσουμε ότι $\hat{\theta}$ αντιστοιχεί στο σύνολο το οποίο μεγιστοποιεί το *likelihood function* τότε ισχύει ότι:

$$\text{Maximum Likelihood Estimation} : L\big(\hat{\theta} : D\big) = max_{\theta \epsilon \Theta} L(\theta : D)$$
$$\mu\varepsilon\ \Theta\ \nu\alpha\ \alpha\nu\tau\iota\sigma\tau\sigma\iota\chi\varepsilon\acute{\iota}\ \sigma\tau\sigma\ hypothesis\ space$$

Αξιοποιώντας τώρα το *Bayesian network structure* μπορούμε να εκφράσουμε το *likelihood function* με τον παρακάτω τρόπο:

$$L(\theta : D) = P(D|\theta) = \prod_{i=1}^{M} P_G(\xi[i] : \theta)$$

$$= \prod_{i=1}^{M} \prod_{j=1}^{n} P(x_i[m] \mid Par(X_i)[m] : \theta)$$

$$= \prod_{i} \left[\prod_{m} P(x_i[m] \mid Par(X_i)[m] : \theta) \right]$$

Παρατηρούμε λοιπόν ότι ο κάθε όρος μέσα στις αγκύλες αντιστοιχεί στο *condition likelihood function* από την τυχαία μεταβλητή X_i δεδομένου του $Par(X_i)$. Αν υποθέσουμε ότι θ_{X_i} αντιστοιχεί στο σύνολο των παραμέτρων που καθορίζουν το *CPD* $P(X_i|Par(X_i))$, τότε ισχύει ότι:

$$L(\theta : D) = \prod_{i} L_i(\theta_{X_i} : D)$$

Όπου ο κάθε παράγοντας του γινομένου αντιστοιχεί στο *local likelihood* από την τυχαία μεταβλητή X_i και ισοδυναμεί με:

$$L_i\big(\theta_{X_i} : D\big) = \prod_{m} P(x_i[m] \mid Par(X_i)[m] : \theta_{X_i})$$

Η τελευταία ιδιότητα είναι γνωστή ως *global decomposability,* όπου αν υποθέσουμε ότι το κάθε θ_{X_i} είναι ανεξάρτητο από θ_{X_j} για όλα τα $i \neq j$ με $i, j \in \{1, \dots, n\}$, τότε μπορούμε να μεγιστοποιήσουμε το κάθε *local likelihood function* ανεξάρτητα από τα υπόλοιπα και στην συνέχεια μπορούμε να τα συνδυάσουμε για να πάρουμε το *MLE*. Πράγμα που το καθιστά ιδανικό(σε συνδυασμό με το *local decomposability*) για την εφαρμογή του σε *distributed περιβάλλον*.

Επομένως αν υποθέσουμε ότι έχουμε ένα σύνολο δεδομένων $D = \{\, \xi[1], \dots, \xi[M] \,\}$ και ένα *Bayesian Network Structure* $\mathcal{G}(\mathcal{X}, \mathcal{E})$, τότε αν υποθέσουμε ότι θ_{X_i} είναι ανεξάρτητο από θ_{X_j} για όλα τα $i \neq j$ με $i, j \in \{1, \dots, n\}$ και επιπλέον ισχύει ότι $\hat{\theta}_{X_i}$ μεγιστοποιεί το *local likelihood*

function $L_i(\theta_{X_i} : D)$ της τυχαίας μεταβλητής X_i , τότε το σύνολο παραμέτρων $\hat{\theta} = \{\hat{\theta}_{X_1}, \dots, \hat{\theta}_{X_n}\}$ είναι αυτό το οποίο μεγιστοποιεί και το *likelihood function* $L(\theta : D)$.

Πέραν από την ιδιότητα του *global decomposability* από το *likelihood* ενός *Bayesian Network* έχουμε και την ιδιότητα του *local decomposability*. Συγκεκριμένα αν υποθέσουμε ότι έχουμε το *tabular CPD* $P(X_i \mid Par(X_i))$ (όπου αποτελεί και την κατηγορία που εστιάζουμε) της τυχαία μεταβλητή X_i με $Par(X_i) = \boldsymbol{U}$, τότε μπορούμε να ξαναγράψουμε το *local likelihood function* με τον ακόλουθο τρόπο:

$$\begin{aligned} L_i(\theta_{X_i} : D) &= \prod_m P(x_i[m] \mid Par(X_i)[m] : \theta_{X_i}) \\ &= \prod_m \theta_{x_i[m] \mid Par(X_i)[m]} \\ &= \prod_{\boldsymbol{u} \in Val(\boldsymbol{U})} \left[\prod_{x \in Val(X_i)} \theta_{x_i \mid \boldsymbol{u}}^{M[\boldsymbol{u}, x_i]} \right] \end{aligned}$$

Όπου το $M[\boldsymbol{u}, x_i]$ αντιστοιχεί στο αριθμό των φορών που εμφανίζεται ο συνδυασμός τιμών $X_i = x_i$ και $Par(X_i) = \boldsymbol{u}$ στο D.

Επομένως με βάση την προηγουμένη ισότητα επιτυγχάνουμε να συγκεντρώσουμε μαζί όλες τις αναφορές για την παράμετρο $\theta_{x_i|\boldsymbol{u}}$ για όλα τα $x_i \in Val(X_i)$. Επομένως προσπαθούμε να μεγιστοποιήσουμε της παραμέτρους $\theta_{x_i|\boldsymbol{u}}$ λαμβάνοντας υπόψιν τον περιορισμό ότι $\sum \theta_{x_i|\boldsymbol{u}} = 1$ για όλα τα $\boldsymbol{u} \in Val(\boldsymbol{U})$. Η τελευταία ισότητα αντιστοιχεί σε αυτό το οποίο αποκαλούμε *local decomposability* γιατί μπορούμε πλέον να μεγιστοποιήσουμε τις παραμέτρους $\theta_{x_i|\boldsymbol{u}}$ για κάθε $\boldsymbol{u} \in Val(\boldsymbol{U})$ ανεξάρτητα την μια από την άλλη.

Maximum Likelihood Estimate

Τέλος μπορούμε να αποδείξουμε ότι το *maximum likelihood estimate* $\hat{\theta}_{X_i}$ από το θ_{X_i}(προϋποθέτει βέβαια ότι το *likelihood function* είναι *παραγωγίσιμο*) το οποίο αντιστοιχεί στο *tabular CPD* $P(X_i \mid Par(X_i))$ ισούται:

$$Maximum\ likelihood\ estimate : \hat{\theta}_{X_i} = \frac{C_i(x_i, x_i^{par})}{C_i(x_i^{par})}$$
$$για\ κάθε\ x_i \in Val(X_i), x_i^{par} \in Par(X_i)$$

Όπου το $C_i(x_i, x_i^{par})$ ισούται με τον αριθμό των *instances* $(X_i = x_i, Par(X_i) = x_i^{par})$ που βρίσκονται στο D και το $C_i(x_i^{par})$ να ισούται με τον αριθμό των *instances* $(Par(X_i) = x_i^{par})$ που βρίσκονται στο D. Επομένως το μόνο πράγμα που χρειαζόμαστε είναι το $C_i(x_i, x_i^{par})$ και το $C_i(x_i^{par})$ για κάθε τυχαία μεταβλητή X_i του δικτύου και για κάθε $x_i \in Val(X_i)$ και $x_i^{par} \in Val(Par(X_i))$. Στον *Αλγόριθμο 2* βλέπουμε τον αλγόριθμο *MLE*.

Algorithm 2: *Maximum likelihood Estimation (MLE)*

Input:
Bayesian Network Structure $\mathcal{G}(\mathcal{X}, \mathcal{E})$
$\boldsymbol{D} = \{\, \xi[1], \dots, \xi[M] \,\}$ *of full assignments*
Output:
$\hat{\theta} = \{\widehat{\theta_{X_1}}, \dots, \widehat{\theta_{X_n}}\}$

1. // Count
2. **Foreach** $\xi[1]$ **from** $\boldsymbol{D}$
3. **Foreach** X_i **from** $\mathcal{X}$
4. Increment count(x_i, x_i^{par})
5. Increment count(x_i^{par})
6. **End**
7. **End**
8.
9. // Estimation
10. **Foreach** X_i **from** $\mathcal{X}$
11. **Return** $\widehat{\theta_{X_i}} = F_i(x_i, x_i^{par})$ / $F_i(x_i^{par})$
12. **End**

Επομένως δεδομένου ενός *Bayesian Network Structure* $\mathcal{G}(\mathcal{X}, \mathcal{E})$, το *MLE* από το *joint probability distribution* μπορούμε να το εκφράσουμε με τον ακόλουθο τρόπο:

$$P(\mathcal{X}) = \prod_{i=1}^{n} \theta_{X_i} = \prod_{i=1}^{n} \frac{C_i(x_i, x_i^{par})}{C_i(x_i^{par})}$$

MLE consistency

Επιπλέον μπορεί να αποδειχθεί ότι ο αλγόριθμος *MLE* είναι *consistency* [1, Theorem 16.1]. Δεδομένου μια ακολουθίας από *i.i.d* δείγματα $D_M = \{\xi[1], .., \xi[M]\}$ από την κατανομή $P(\mathcal{X})$, τότε ισχύει ότι:

$$\lim_{M \to \infty} P_{D_M}(A) = P(A)$$

Δηλαδή για αρκετά μεγάλο σύνολο δεδομένων η εμπειρική κατανομή(*empirical distribution*) $P_{D_M}(A)$ που ισοδυναμεί με την κατανομή που παίρνουμε από *MLE* αλγόριθμο , θα είναι αρκετά κοντά στην αρχική κατανομή $P(\mathcal{X})$ με μεγάλη πιθανότητα, συγκεκριμένα η πιθανότητα συγκλίνει στο 1 καθώς το $M \to \infty$.

Data fragmentation and overfitting

Τέλος ένα περιοριστικός παράγοντας για την εκμάθηση των παραμέτρων(*learning parameters*) από ένα *Bayesian Network* δεδομένου ενός συνόλου δεδομένων(*training instances*), αποτελεί το γεγονός ότι καθώς αυξάνεται ο αριθμός των *parents nodes* από ένα

κόμβο ενώ παράλληλα έχουμε εκθετική αύξηση στα *parents assignments* του κόμβου ταυτόχρονα περιμένουμε ότι θα έχουμε και εκθετική μείωση στο αριθμό των δεδομένων από το κάθε ένα *parent assignment*. Το φαινόμενο αυτό ονομάζεται *data fragmentation* δηλαδή το σύνολο των δεδομένων χωρίζεται σε ένα μεγάλο αριθμό από μικρά υποσύνολα. Επομένως στην περίπτωση που αριθμός των δεδομένων είναι μικρός τότε κάποιο υποσύνολο των δεδομένων που έχουμε για μια παράμετρο θα είναι μικρό, επομένως υπάρχουν περιπτώσεις όπου μπορεί να μας οδηγήσει σε *overfitting*(μεγάλος αριθμός από μηδενικά στην κατανομή).

2.6 Laplace Smoothing

Ένα πρόβλημα που παρατηρείται κατά την διαδικασία του αλγορίθμου *MLE* τόσο στα *Bayesian Network* όσο και σε *Naïve Bayes Classifier* είναι ότι σε ορισμένες περιπτώσεις και συγκεκριμένα όταν το μέγεθος των *training instances* είναι αρκετά μικρό (βέβαια αυτό μπορεί να συμβεί και όταν το μέγεθος των *training instances* είναι μεγάλο σε συνδυασμό *highly skewed distributions*), να μην είναι αντιπροσωπευτικό του *joint probability distribution* δηλαδή να μην περιέχει δεδομένα για όλες τις τιμές των κόμβων του δικτύου και κυρίως για εκείνες τις τιμές για τις οποίες η παράμετρος θ_{X_i} που καθορίζει την τιμή τους είναι αρκετά μικρή.

Επομένως σε ορισμένες περιπτώσεις μπορεί να μην έχουμε καθόλου δεδομένα για συγκεκριμένες τιμές των κόμβων του δικτύου, με αποτέλεσμα κατά την διαδικασία *εκτίμησης*(*estimation*) των *queries* στο *joint probability distribution* να θεωρήσουμε τις παραμέτρους που αντιστοιχούν σε αυτές τις τιμές μηδενικές(*zero parameter*), κάτι που στην πραγματικότητα μπορεί να μην ισχύει, γιατί δεν σημαίνει απαραίτητα επειδή δεν έχουμε "εντοπίσει" τιμές συγκεκριμένων κόμβων ότι και πρακτικά δεν υπάρχουν. Συνεπώς το πρόβλημα προκύπτει όταν δεν έχουμε εντοπίσει τιμές των κόμβων των οποίων η παράμετρος θ που αντιστοιχεί στην συγκεκριμένη τιμή είναι μη μηδενική(*non-zero parameter*).

Σε αυτές τις περιπτώσεις λοιπόν, όπου δεν έχουμε καθόλου δεδομένα για κάποια παράμετρο και κληθούμε να απαντήσουμε σε *queries* που περιέχουν τιμές ή συνδυασμό τιμών κόμβων που αντιστοιχεί σε μικρή πιθανότητα δηλαδή σε μικρό θ (συνήθως παρατηρείται όταν τα *queries* αντιστοιχούν σε *highly unlikely queries* δηλαδή σε *queries* με αρκετά μικρό *joint probability distribution* δηλαδή είναι "*σπάνια*" να συμβούν) , τότε το αποτέλεσμα που θα πάρουμε δεν θα είναι ιδανικό αφού μη έχοντας καθόλου δεδομένα για τιμές ή συνδυασμό τιμών κόμβων, θα οδηγηθούμε σε *μηδενικό joint probability distribution*, αφού για όσες παραμέτρους δεν υπάρχουν δεδομένα θα τις "*θεωρήσουμε*" *μηδενικές* ενώ στην πραγματικότητα κάτι τέτοιο μπορεί να μην ισχύει.

Ένας τρόπος για να αντιμετωπίσουμε τέτοιες περιπτώσεις δηλαδή *zero* παραμέτρους είναι να εφαρμόζουμε την ενδεδειγμένη τεχνική του *Laplace smoothing.* Να σημειωθεί ότι η τεχνική του *Laplace smoothing* μπορεί να εφαρμοστεί σε όλες τις παραμέτρους δηλαδή *zero* και *non-zero* παραμέτρους αλλά εστιάζουμε όπως έχει αναφερθεί σε *zero* παραμέτρους. Η βασική ιδέα είναι ότι για κάθε *zero* παράμετρο δηλαδή για κάθε παράμετρο που δεν έχουμε καθόλου δεδομένα, να προσθέτουμε λ $(pseudocount)$ αναφορές της τιμής που αντιστοιχεί στην παράμετρο ανεξαρτήτως αν έχουν παρατηρηθεί ή όχι. Επειδή η παράμετρος θ_{X_i}

αντιστοιχεί σε κάποια πιθανότητα χρειάζεται επιπροσθέτως να κανονικοποιήσουμε την τιμή της και αυτό μπορεί να γίνει διαιρώντας με τον συνολικό αριθμό των τιμών της μεταβλητής X_i πολλαπλασιασμένο με την τιμής της παραμέτρου λ. Επομένως αν υποθέσουμε ότι η παράμετρος θ_{X_i} αντιστοιχεί στο *CPD* $P(X = x_i \mid Par(X_i))$ της τυχαίας μεταβλητής X_i, τότε με την εφαρμογή του *Laplace Smoothing* θα έχουμε ότι:

$$Maximum\ likelihood\ estimate\ with\ Laplace\ Smoothing$$

$$P(X = x_i \mid Par(X_i)) = \frac{count\big(X = x_i\ \ and\ Par(X_i)\big) + \lambda}{count\big(Par(X_i)\big) + \lambda \cdot Val(X_i)}\ \ \mu\varepsilon\ \lambda > 0$$

Μπορούμε να δούμε την εφαρμογή της τεχνικής του *Laplace Smoothing* με ένα μικρό παράδειγμα. Αν υποθέσουμε ότι έχουμε ένα *biased coin* με δυο όψεις $\{Head(H), Tail(T)\}$, και αποφασίσουμε να κάνουμε μια ρίψη, έστω ότι το αποτέλεσμα που παίρνουμε είναι $Head$. Έπειτα αποφασίζουμε να κάνουμε την εκτίμηση των παραμέτρων χρησιμοποιώντας το αποτέλεσμα της παραπάνω ρίψης και τον αλγόριθμο *MLE*, τότε προκύπτει ότι:

$$Maximum\ Likelihood\ estimate$$

$$p(H) = \frac{1}{1} = 1\,, \qquad p(T) = \frac{0}{1} = 0$$

Παρατηρούμε λοιπόν ότι και οι δύο εκτιμήσεις που παίρνουμε από τον αλγόριθμο *MLE* δεν είναι "*λογικές*". Μάλιστα στην συγκεκριμένη περίπτωση έχουμε και ένα παράδειγμα από *overfit* του *MLE* αλγορίθμου κάτι το οποίο επιβεβαιώνει και το γεγονός ότι όταν το μέγεθος των δεδομένων που έχουμε στην διάθεση είναι αρκετά μικρό και ο αριθμός των παραμέτρων αρκετά μεγάλος τότε ο αλγόριθμος του *MLE* θα κάνει *overfit* (πολλές από τις παραμέτρους θα τις θεωρήσει μηδενικές κάτι που στην πραγματικότητα δεν ισχύει*)*.

Κάνοντας τώρα εφαρμογή της τεχνικής του *Laplace Smoothing* και θεωρώντας ότι $\lambda = 1$ τότε προκύπτει ότι:

$$Maximum\ Likelihood\ estimate\ with\ Laplace\ Smoothing$$

$$p(H) = \frac{1 + 1}{1 + 2} = \frac{2}{3}\,, \qquad p(T) = \frac{0 + 1}{1 + 2} = \frac{1}{3}$$

Παρατηρούμε λοιπόν η εκτίμηση των παραμέτρων που έχουμε έπειτα από την εφαρμογή του *Laplace Smoothing* τείνει περισσότερο στην πραγματικότητα αλλά ταυτόχρονα έχουμε και την αποφυγή του *overfit* ακόμα και με μικρό αριθμό δεδομένων.

Επομένως μεταβάλλοντας την τιμή του λ ,ουσιαστικά καθορίζουμε το *smoothing-regularization* που προσθέτουμε στην παράμετρο. Όσο μεγαλύτερη είναι η τιμή του λ τόσο περισσότερο η κατανομή προσεγγίζει την ομοιόμορφη κατανομή. Συγκεκριμένα σε περιπτώσεις που αφορούν *zero* παραμέτρους δηλαδή τόσο ο αριθμητής όσο και ο παρονομαστής της προηγούμενης σχέσης είναι μηδενικός, ανεξαρτήτως από την τιμή του λ έχουμε την ακριβή προσέγγιση της ομοιόμορφης κατανομής.

Συνήθως αυτό το οποίο ισχύει τις περισσότερες φορές είναι ο αριθμητής της σχέσης να είναι μηδενικός . Όπως αποδείχθηκε και από πειραματικό κομμάτι η τιμή του λ που πετυχαίνουμε τα καλύτερα δυνατά αποτελέσματα όσον αφορά την "*ποιότητα*" της εκτίμησης που παίρνουμε, επιτυγχάνεται όταν το $\lambda = 1$.

Επιπλέον ένα άλλο πλεονέκτημα αυτής της προσέγγισης είναι ότι όσο περισσότερα γίνονται τα δεδομένα τόσο λιγότερη γίνεται και η επίδραση του λ , πράγμα το οποίο είναι επιθυμητό δεδομένου ότι όσο περισσότερα δεδομένα έχουμε τόσο περισσότερο μπορούμε να τα "*εμπιστευτούμε*". Στην περίπτωση μας δηλαδή όταν έχουμε μια *zero* παράμετρο, όταν αρχίζουμε να "*εντοπίζουμε*" δεδομένα για την συγκεκριμένη παράμετρο δηλαδή όταν μεταπίπτει σε μια *non-zero* παράμετρο τότε παύει και η χρήση του *Laplace Smoothing* , όπως είπαμε και προηγουμένως την τεχνική του *Laplace Smoothing* την εφαρμόζουμε αποκλειστικά και μόνο σε *zero* παραμέτρους τόσο στην περίπτωση των *Bayesian Networks* όσο και στους *Naïve Bayes Classifier,* σε διαφορετική περίπτωση ο υπολογισμός της τιμής των παραμέτρων γίνεται αποκλειστικά από την χρήση των δεδομένων.

2.7 Distributed Continuous Model

Το μοντέλο το οποίο χρησιμοποιούμε δεν είναι άλλο από το *Continuous Distributed Monitoring Model* [16]. Συγκεκριμένα μπορούμε να το δούμε ως ένα συνδυασμό από το *communication model* όπου υπάρχουν $k \geq 2$ *sites* και ως στόχο θέλουμε *το one-shot computation* από κάποια συνάρτηση και του *data stream model* όπου έχουμε ένα *site*($k = 1$) , και ως στόχο θέλουμε το *continuously monitoring* κάποιας συνάρτησης.

Με το μοντέλο αυτό μπορούμε να προσομοιώσουμε τα περισσότερα προβλήματα τα οποία παρουσιάζονται σε *learning tasks* πάνω σε "*big data streaming"* συστήματα , δηλαδή σε συστήματα με όγκο δεδομένων που είναι αρκετά μεγάλος(της τάξης *Petabytes*), με δεδομένα τα οποία είναι *streaming* και *distributed* ενώ ταυτόχρονα πρέπει να μπορούμε να ανταποκριθούμε σε *real-time απαντήσεις*.

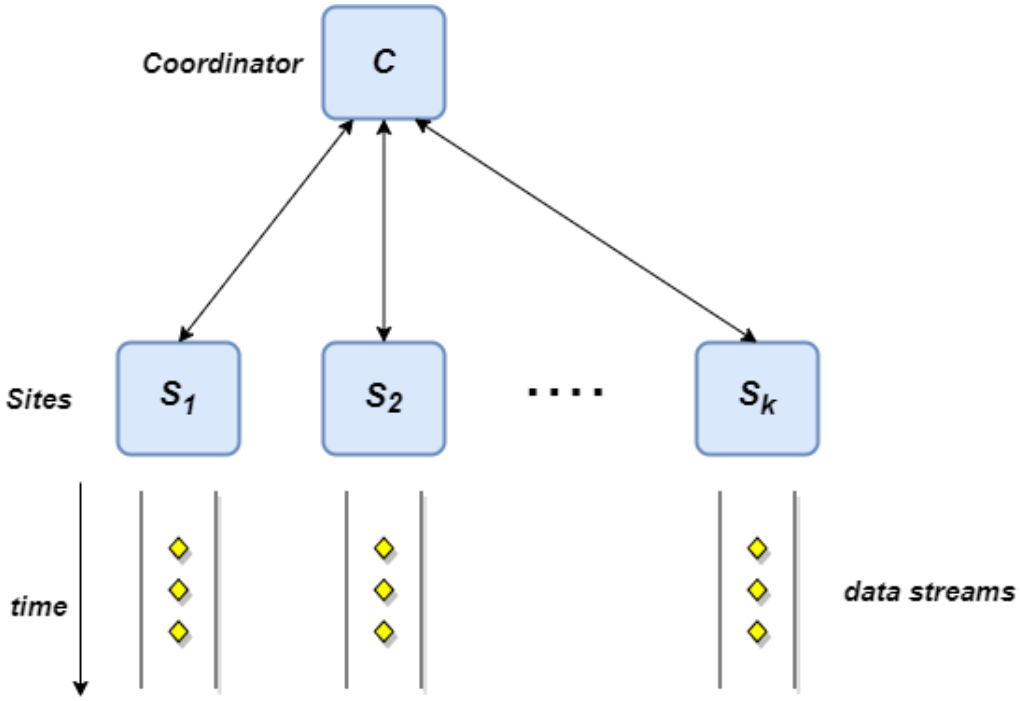
Γενικά το *Continuous Distributed Monitoring Model* μπορούμε να το δούμε ως ένα σύνολο από *παρατηρητές*(*observers*) όπου ο καθένας βλέπει μόνο ένα κομμάτι από τις παρατηρήσεις, με στόχο να συνεργαστούν ώστε να εκτιμήσουν κάποιο *function* πάνω στην ένωση των παρατηρήσεων , όπου αυτό πρέπει να γίνεται για κάθε χρονική στιγμή.

Το *function* μπορεί για παράδειγμα να αντιστοιχεί στο *counting* του συνόλου των παρατηρήσεων (*count-tracking πρόβλημα* – το οποίο είναι και αυτό το οποίο εστιάζουμε), το οποίο αντιστοιχεί σε ένα *linear* και *monotonic increasing function*. Βέβαια μπορεί να περιέχει και κάποιο περισσότερο *complex function* όπως θα μπορούσε να είναι το *second frequency moment* F_2 το οποίο βρίσκει εφαρμογή σε πολλές περιπτώσεις (για παράδειγμα σε *join sizes*).

Ως επιπλέον στόχο αποτελεί η προσπάθεια να χρησιμοποιήσουμε όσον τον δυνατόν λιγότερο *communication cost* μεταξύ των παρατηρητών(*communication-efficient*) γίνεται. Όπως θα δούμε και στην συνέχεια για να το επιτύχουμε αυτό συνήθως απαιτείται να έχουμε μια *approximate* εκτίμηση του εκάστοτε *function* (*approximate answers*).

Τέλος θεωρούμε ότι το σύνολο των παρατηρήσεων που δέχεται ο κάθε παρατηρητής, πρόκειται για *high-speed,high-volume* και *high-throughput data streams*, επομένως προσεγγίσεις όπως το *periodic polling* είτε το *centralization* των *streams* καθίσταται μη εφικτές.

Συγκεκριμένα θεωρούμε ότι το μοντέλο αποτελείται από k *sites*(ή *workers*) δηλαδή έχουμε ένα σύνολο από *sites* $S = \{S_1, .., S_k\}$. Το κάθε S_i λαμβάνει ένα *stream* από παρατηρήσεις, με ρυθμό που πιθανόν μεταβάλλεται. Υποθέτουμε ότι το $A_i(t)$ αποτελεί το *multiset* των στοιχείων(*bag of elements*) που έχουν ληφθεί από το *site* S_i μέχρι την χρονική στιγμή t και ισχύει ότι το $A(t) = \uplus_i A_i(t)$ με $i \in \{1, \dots, k\}$, δηλαδή το $A(t)$ αποτελεί την ένωση των *streams* από όλα τα *sites*, με το $\uplus$ να υποδηλώνει *multiset addition*.



Εικόνα 11: Distributed Continuous Model

Επιπλέον έχουμε και ακόμα ένα *site* , ο οποίος αντιστοιχεί στον *Coordinator*. Συγκεκριμένα ο *Coordinator* μπορεί να επικοινωνεί απευθείας με τον κάθε ένα site S_i. Επομένως υπάρχει ένα *two-way communication channel* μεταξύ του *Coordinator* με κάθε ένα από τα k *sites* για να εξυπηρετεί τον σκοπό αυτόν(*Εικόνα 11*). Επιπλέον θεωρούμε ότι δεν υπάρχει "*απευθείας*" επικοινωνία μεταξύ των *sites*. Βέβαια αυτό δεν αποτελεί κάποιο περιορισμό γιατί μπορεί να γίνει χρησιμοποιώντας ως ενδιάμεσο κόμβο τον *Coordinator*, στέλνοντας δηλαδή μηνύματα μέσω του *Coordinator*. Τέλος θεωρούμε ότι αριθμός των μηνυμάτων από τα *sites* προς το *Coordinator* αντιστοιχεί στο *downstream communication cost* ενώ ο αριθμός των μηνυμάτων από τον *Coordinator* προς τα *sites* αντιστοιχεί *upstream communication cost*.

Όπως είπαμε και προηγουμένως θέλουμε να υπολογίσουμε κάποιο *function*(*monitoring function*) πάνω στην ένωση των επιμέρους *streams* , συγκεκριμένα αυτό το οποίο θέλουμε είναι ο *Coordinator* για οποιαδήποτε χρονική στιγμή t να "διατηρεί" το $f\big(A(t)\big)$ από κάποιο *function* f. Για παράδειγμα στην περίπτωση του *count-tracking* προβλήματος ισχύει ότι το $f\big(A(t)\big) = |A(t)|$.

Επιπλέον εστιάζουμε στην περίπτωση του *value monitoring* , όπου θέλουμε ανά πάσα χρονική στιγμή t να έχουμε μια εκτίμηση $\hat{f}(A(t))$ από την συνάρτηση $f(A(t))$, παρέχοντας ταυτόχρονα εγγυήσεις σφάλματος(*error guarantees*) για την τιμή της εκτίμησης.

Συγκεκριμένα θέλουμε να έχουμε ένα *ε,δ-approximation* από την συνάρτηση $f(A(t))$ ανά πάσα χρονική στιγμή, δηλαδή να ισχύει ότι:

$$(1-\varepsilon)\cdot f\big(A(t)\big) \leq \hat{f}\big(A(t)\big) \leq (1+\varepsilon)\cdot f\big(A(t)\big) \ \mu\varepsilon\ \pi\iota\theta\alpha\nu\acute{o}\tau\eta\tau\alpha\ 1-\delta\ ,\forall\ t$$

Το βασικό στοιχείο το οποίο θέλουμε και προκύπτει από την ανάλυση του αλγορίθμου *MLE* είναι να μπορούμε να διατηρήσουμε τα διαθέσιμα *counters* που χρειάζονται για το κάθε *CPD* του δικτύου, μόνο που αυτό πρέπει να γίνει σε συνδυασμό με το *Continuous Distributed Model*. Επομένως πλέον αναφερόμαστε σε *distributed counters* όπου αναλύονται στην συνέχεια(*Βλ. Κεφάλαιο 4.1.1*). Επιπλέον θέλουμε να διατηρήσουμε τα *distributed counters* χρησιμοποιώντας όσο το δυνατόν λιγότερη επικοινωνία μεταξύ των *sites* και του *Coordinator* (*communication efficient*). Το πρόβλημα αυτό είναι γνωστό ως *count-tracking* πρόβλημα και έχει αναλυθεί σε αρκετές περιπτώσεις όπως παρατίθεται στο [17] και στο [20]. Τέλος το *count-tracking* πρόβλημα μπορούμε να το δούμε και ως το F_1 *frequency moment*.

Κεφάλαιο 3:

Δήλωση του προβλήματος

3.1 Ορισμός του προβλήματος

Η γενική ιδέα του προβλήματος το όποιο εστιάζουμε είναι να σχεδιάσουμε ένα σύστημα το οποίο θα μπορεί να διατηρήσει μια συλλογή από *distributed counters* χρησιμοποιώντας όσον τον δυνατόν λιγότερο αριθμό μηνυμάτων γίνεται (*communication-efficient*) ενώ ταυτόχρονα να παρέχει εγγυήσεις(*error guarantees*) από το μοντέλο το οποίο προσπαθούμε να διατηρήσουμε(*maintenance*). Ωστόσο τα συστήματα με τα οποία ασχολούμαστε πρόκειται για συστήματα τα οποία αποτελούνται από *massive,dynamic,high-throughput* και *distributed data sources*, επομένως η απλοϊκή λύση του *centralization* των δεδομένων καθίσταται μη αποδοτική(*inefficient*) και μη εφικτή(*infeasible*) γιατί πολύ απλά το *communication cost* θα είναι αρκετά μεγάλο.

Επομένως αυτό το οποίο θέλουμε είναι να σχεδιάσουμε μια μέθοδο για το *communication-efficient maintenance* ενός γραφικού μοντέλου πάνω σε *distributed* και *streaming* δεδομένα. Συγκεκριμένα θέλουμε να έχουμε ένα *accurate approximation* από το *MLE* πάνω στο *distributed continuous model* ανά πάσα χρονική στιγμή.

Όπως έχουμε δει και προηγουμένως το μόνο πράγμα το οποίο χρειαζόμαστε για να έχουμε το *maximum likelihood estimate* είναι να μπορούνε να έχουμε στην διάθεση μας τους *counters* που απαιτούνται για να ορίσουμε τις παραμέτρους από το κάθε *CPD* του δικτύου. Επομένως πρέπει να μπορούμε να διατηρήσουμε μια συλλογή από *distributed counters* που απαιτούνται για την εκτίμηση του *MLE*. Δεδομένου ότι είμαστε σε *distributed* περιβάλλον το πρώτο πρόβλημα που παρατηρείται είναι ότι σε περίπτωση που αποφασίσουμε να διατηρήσουμε τους *EXACT counters*(*Βλ. Κεφάλαιο 4.1.1*) , δηλαδή *counters* όπου για κάθε *event* που λαμβάνει κάποιο *site* να στέλνουν και το αντίστοιχο *update* μήνυμα στον *Coordinator* , τότε το *communication cost* θα γινόταν γρήγορα αρκετά μεγάλο με αποτέλεσμα να ήταν και *bottleneck* ολόκληρου του συστήματος.

Επομένως για να αποφύγουμε αυτό το πρόβλημα μπορούμε να κάνουμε ένα *tradeoff* ανάμεσα στην απόδοση(*efficiency*) και την ποιότητα(*quality*) της εκτίμησης. Συγκεκριμένα αντί να έχουμε το *Exact MLE* το οποίο συνοδεύεται και με υψηλό *communication cost* , είναι προτιμότερο να έχουμε ένα *accurate approximate* από το *MLE* δηλαδή ένα *approximate* το οποίο είναι αρκετά κοντά στο αρχικό, το οποίο σε αυτήν την περίπτωση συνοδεύεται με αρκετά μικρότερο *communication cost*. Συγκεκριμένα αυτό μπορεί να γίνει αξιοποιώντας το γεγονός ότι οι παράμετροι που χρειαζόμαστε είναι ανεξάρτητες μεταξύ τους, επομένως αυτό το οποίο χρειάζεται να κάνουμε είναι να βρούμε τον τρόπο με τον οποίο μπορούμε να χωρίσουμε το διαθέσιμο "*error budget*" μεταξύ των διαφορετικών και ανεξάρτητων παραμέτρων(*slack allocation problem*), εξασφαλίζοντας πρώτον ότι το *error* από το *joint probability distribution* είναι στα επιθυμητά όρια και δεύτερον το *communication cost* να είναι όσον τον δυνατό λιγότερο, υποθέτοντας πάντα ότι το *Bayesian Network Structure* είναι γνωστό εκ των προτέρων.

Αυτό μπορεί να γίνει χρησιμοποιώντας τόσο την πρώτη προσέγγιση(*Βλ. Κεφάλαιο 4.1*) δηλαδή να έχουμε μια συλλογή από *approximate distributed counters* , όπου το καθένα τον βλέπουμε ως ξεχωριστή οντότητα όσο και με την προσέγγιση που προτείναμε δηλαδή χρησιμοποιώντας το *Functional Geometric Protocol(FGM)*. Αυτό πρέπει να γίνει βέβαια σε συνδυασμό με τους τρεις αλγορίθμους *BASELINE,UNIFORM,NON_UNIFORM* [18], οι οποίοι καθορίζουν τον τρόπο με τον οποίο μπορούμε να χωρίσουμε με κατάλληλο τρόπο το διαθέσιμο "*error budget*" και αναλύονται στην συνέχεια.

Δεδομένου ενός *Bayesian Network structure* $\mathcal{G}(\mathcal{X},\mathcal{E})$, υποθέτουμε ότι $\hat{P}(\mathcal{X})$ υποδηλώνει το *EXACTMLE* από το *joint probability distribution* που αντιστοιχεί στο $\mathcal{G}$. Επιπλέον δεδομένου ενός *approximation factor* ε με $\varepsilon \in [0,1]$, τότε ένα *ε-approximation* από το *MLE* είναι το *joint probability distribution* $\tilde{P}(\mathcal{X})$ για κάθε $\boldsymbol{x} \in Val(\mathcal{X})$, αν και μόνο αν ισχύει ότι:

$$\varepsilon - approximation\ of\ MLE$$

$$e^{-\varepsilon} \leq \frac{\tilde{P}(\boldsymbol{x})}{\hat{P}(\boldsymbol{x})} \leq e^{\varepsilon}$$

Αντίστοιχα δεδομένου μια επιπρόσθετης παραμέτρους δ με $\delta \in [0,1]$, τότε το $\tilde{P}(\mathcal{X})$ είναι *(ε,δ)-approximation* αν και μόνο αν ισχύει ότι:

$$(\varepsilon,\delta) - approximation\ of\ MLE$$

$$e^{-\varepsilon} \leq \frac{\tilde{P}(\boldsymbol{x})}{\hat{P}(\boldsymbol{x})} \leq e^{\varepsilon}\ με\ πιθανότητα\ 1-\delta$$

Στόχος μας είναι ανά πάσα χρονική στιγμή να έχουμε ένα *(ε,δ)-approximation* από το *MLE*. Δεδομένου ενός *Bayesian Network structure* $\mathcal{G}(\mathcal{X},\mathcal{E})$, το *EXACTMLE* από το *joint probability distribution* $\tilde{P}(\mathcal{X})$ (*Βλ. Κεφάλαιο 2.5*) μπορούμε να το εκφράσουμε με τον ακόλουθο τρόπο:

$$\tilde{P}(\mathcal{X}) = \prod_{i=1}^{n} \theta_{X_i} = \prod_{i=1}^{n} \frac{C_i\left(x_i, x_i^{par}\right)}{C_i\left(x_i^{par}\right)}$$

Με το $C_i\left(x_i, x_i^{par}\right)$ να αντιστοιχεί στον αριθμό των *events* $(X_i = x_i, Par(X_i) = x_i^{par})$ και αντίστοιχα το $C_i(x_i^{par})$ να ισούται με τον αριθμό των *events* $(Par(X_i) = x_i^{par})$.

Τώρα αν υποθέσουμε ότι $A_i\left(x_i, x_i^{par}\right)$, $A_i\left(x_i^{par}\right)$ αντιστοιχούν σε *approximate distributed counters* των $C_i\left(x_i, x_i^{par}\right)$, $C_i(x_i^{par})$ αντίστοιχα, τότε για οποιοδήποτε *input vector* $\boldsymbol{x} = [x_1, \dots, x_n]$, δεδομένου ενός *approximation factor ε*, θέλουμε να ισχύει ότι:

$$e^{-\varepsilon} \leq \frac{\tilde{P}(\boldsymbol{x})}{\hat{P}(\boldsymbol{x})} = \prod_{i=1}^{n} \left(\frac{A_i\left(x_i, x_i^{par}\right)}{C_i\left(x_i, x_i^{par}\right)} \cdot \frac{C_i\left(x_i^{par}\right)}{A_i\left(x_i^{par}\right)} \right) \leq e^{\varepsilon}$$

Ορισμός του Classification προβλήματος

Όπως αναφέραμε και σε προηγούμενο κεφάλαιο πέρα από τα *Bayesian Networks* εστιάζουμε και σε *Naïve Bayes Classifier*. Δεδομένου ότι πλέον στην διάθεση μας έχουμε *approximate distributed counters* θα πρέπει να ορίσουμε ξανά και το πρόβλημα του *Classification*. Δεδομένου ενός *Naïve Bayes Classifier* $\mathcal{G}(\mathcal{X}, \mathcal{E})$ και ενός *approximation factor* ε, για οποιοδήποτε *input vector* $\boldsymbol{x}$(ή αλλιώς *evidence*) θα ισχύει ότι το *class label* c^i μπορεί να επιλύσει το *Bayesian classification* πρόβλημα με ε *error* αν:

$$\hat{P}\left(C = c^i \middle| \boldsymbol{x}\right) \geq (1-\varepsilon) \cdot max_{c^i \in Val(C)} \hat{P}\left(C = c^i \middle| \boldsymbol{x}\right)$$

Δηλαδή επιλέγουμε εκείνο το *class label* c^i το όποιο έχει *conditional probability* που βρίσκεται κοντά στη μέγιστη πιθανότητα. Δεδομένου ότι έχουμε ότι έχουμε *(ε,δ)-approximation* από το *MLE* του *joint probability distribution* του *Naïve Bayes Classifier* τότε μπορούμε να βρούμε το *class label* c^i το οποίο μπορεί να επιλύσει το *Bayesian Classification* πρόβλημα με ε *error* [18, Lemma 13].

Συνεπώς αυτό το οποίο θέλουμε είναι να έχουμε ανά πάσα χρονική στιγμή(*continuously*) ένα *(ε,δ)-approximation* από *MLE* του *joint probability distribution* που αντιστοιχεί στο *Bayesian Network* πάνω στο *distributed continuous model* , χρησιμοποιώντας ταυτόχρονα όσο τον δυνατόν λιγότερο *communication cost* γίνεται(*communication efficient*).

3.2 Η γενική προσέγγιση

Παρακάτω βλέπουμε την γενική προσέγγιση που χρησιμοποιείται τόσο από την πρώτη προσέγγιση δηλαδή να έχουμε μια συλλογή από *approximate distributed counters* τα οποία τα βλέπουμε ως ξεχωριστές οντότητες(*Βλ. Κεφάλαιο 4.1*) όσο και από την προσέγγιση την οποία προτείναμε (*Βλ. Κεφάλαιο 4.2*).

Την γενική προσέγγιση του προβλήματος μπορούμε να την συνοψίζουμε με τους παρακάτω τέσσερις αλγορίθμους.

Ο πρώτος αλγόριθμος είναι ο αλγόριθμος *INITIALIZATION* (Αλγόριθμος 1). Πρόκειται για έναν αλγόριθμο που χρησιμοποιείται σε όλα τα *sites*, συμπεριλαμβανομένου και του *coordinator*. Ο αλγόριθμος αυτός εκτελείται πρώτος και συγκεκριμένα είναι υπεύθυνος για την *αρχικοποίηση(initialization)* όλων των διαθέσιμων *approximated distributed counters* που απαιτούνται από το κάθε *CPD* του *Bayesian Network*, ώστε να μπορούμε να εκτιμήσουμε το *MLE* από το *joint probability distribution* του δικτύου. Συγκεκριμένα για την αρχικοποίηση χρειαζόμαστε τρεις παραμέτρους. Η πρώτη παράμετρο αναφέρεται στο *Bayesian Network* που θα χρησιμοποιήσουμε, η δεύτερη παράμετρος αναφέρεται στον τύπο από το *approximate distributed counter*(*type_counter*) που θα χρησιμοποιήσουμε όσον αφορά την πρώτη προσέγγιση ενώ στην περίπτωση της δεύτερης προσέγγισης αναφέρεται στον τύπο από το *frequency vector* που μπορούμε να χρησιμοποιήσουμε. Τέλος η τελευταία παράμετρο αναφέρεται στο *schema_error*, δηλαδή στον αλγόριθμο που θα χρησιμοποιήσουμε για να καθορίσουμε την κατάλληλη τιμή του *approximation factor* ε για καθένα από τα διαθέσιμα *counters* και αντιστοιχεί σε έναν από τους τρεις αλγορίθμους *BASELINE,UNIFORM* και *NON_UNIFORM*.

Algorithm 1: *INITIALIZATION*

Input:

type_counter: corresponds to one of the four available algorithms (RANDOMIZED, DETERMINISTIC, EXACT, CONITNUOUS)
schema_error: corresponds to one of the three available algorithms (BASELINE, UNIFORM, NON_UNIFORM)
Bayesian Network Structure $\mathcal{G}(\mathcal{X}, \mathcal{E})$ *, where* $\mathcal{X} = \{X_1, .., X_n\}$ *are the nodes of Bayesian Network*

1. **Foreach** X_i **from** $\mathcal{X}$
2. *// Generate all the distributed counters from node* X_i *based on type of counter*
3. *GENERATE_DIST_COUNTERS* (*type_counter*, X_i)
4.
5. *// Initialize the distributed counters based on error schema*
6. **Foreach** $A_i\left(x_i, x_i^{par}\right)$ **from** $X_i \cap Par(X_i)$
7. *INITIALIZE_DIST_COUNTER* ($A_i\left(x_i, x_i^{par}\right)$, *schema_error*)
8. **Foreach** $A_i\left(x_i^{par}\right)$ **from** $Par(X_i)$
9. *INITIALIZE_DIST_COUNTER* ($A_i\left(x_i^{par}\right)$, *schema_error*)
10. **End**

Ο αλγόριθμο ο οποίος είναι υπεύθυνος ώστε να διαχειρίζεται τα *events* είναι ο αλγόριθμος *UPDATE(Αλγόριθμος 2)*. Αυτός ο αλγόριθμος χρησιμοποιείται αποκλειστικά από όλα τα διαθέσιμα *sites* με σκοπό το καθένα να μπορεί να διαχειριστεί το *data stream* από *training instances* που του αναλογεί. Σκοπός του είναι να διαχειρίζεται το εκάστοτε *input vector* κάνοντας παράλληλα όλες τις απαραίτητες αυξήσεις στα *distributed counters*.

Algorithm 2: UPDATE

Input: $\boldsymbol{x} = < x_1, \ldots, x_n >$ *is a full particle of nodes from Bayesian Network*

1. **Foreach** x_i **from** $\boldsymbol{x}$
2.
3. **If**(X_i *has parents*) **then**
4. *// Get the counters correspond to the parents' nodes of* X_i
5. *GET_COUNTER* (x_i^{par})
6. *INCREMENT_DIST_COUNTER* (x_i^{par})
7. **End**
8.
9. *// Get the counters correspond to the* X_i *node*
10. *GET_COUNTER* (x_i)
11.
12. *// Increment the distributed counters for the* X_i *node*
13. *INCREMENT_DIST_COUNTER* (x_i)

Επιπλέον ο αλγόριθμος ο οποίος είναι υπεύθυνος για να διαχειρίζεται τα μηνύματα μεταξύ των *sites* και του *Coordinator*(και το ανάποδο) είναι ο αλγόριθμος *HANDLE_MESSAGE (Αλγόριθμος 3)*. Συγκεκριμένα χρησιμοποιείται και από τις δύο πλευρές, δηλαδή τόσο από την πλευρά των *sites* που δέχονται μηνύματα από το *Coordinator* όσο και από την πλευρά του *Coordinator* που δέχεται μηνύματα από τα *sites*(*two-way communication channel*).

Ανάλογα με τον τύπο του μηνύματος(*type_message*) εκτελείται και η αντίστοιχή ενέργεια που συνήθως είτε πρόκειται για την αποστολή κάποιου μηνύματος (για παράδειγμα θα μπορούσε να αποτελούσε το *broadcast* κάποιας τιμής από το *Coordinator* προς τα *sites*) είτε για το *update* της τιμής κάποιας μεταβλητής. Τόσο ο τύπος των μηνυμάτων όσο η μετέπειτα ακολουθία ενεργειών καθορίζεται από την εκάστοτε προσέγγιση.

Algorithm 3: *HANDLE_MESSAGE*

Input:

type_message: corresponds to one of the available types of messages
message: corresponds to a message

1. *// Handle the message based on the type of it*
2. *HANDLE_MESSAGE* (*type_message*)
3. *UPDATE_COUNTERS ()*

Τέλος ο αλγόριθμος ο οποίος είναι υπεύθυνος για την εκτίμηση των *probability queries* είναι ο αλγόριθμος *ESTIMATE(Αλγόριθμος 4)* και χρησιμοποιείται αποκλειστικά από το *Coordinator*. Επομένως καθώς δέχεται *queries* ο *coordinator*, αυτό το οποίο χρειάζεται να κάνει είναι να υπολογίσει την τιμή από το *joint probability distribution* που αντιστοιχεί στο *query*.

Algorithm 4: *ESTIMATE*

Input: $\boldsymbol{x} = < x_1, \dots, x_n >$ *is a full particle of nodes from Bayesian Network*
Output: *Return the estimated probability from event*

1. **Foreach** x_i **from** $\boldsymbol{x}$
2. *// Get the counters for the parameter* θ_{X_i}
3. *GET_COUNTER*$\left(x_i, x_i^{par}\right)$
4. *GET_COUNTER*$\left(x_i^{par}\right)$
5.
6. *// Estimate the parameter for the specific* X_i
7. *ESTIMATE_PAR* **(**θ_{X_i}**)**
8. *UPDATE_EST_PROBABILITY* (θ_{X_i})
9. **End**
10. **Return** *EST_PROBABILITY*

Κεφάλαιο 4:

Ανάλυση του προβλήματος

4.1 Μια πρώτη προσέγγιση

Η πρώτη προσέγγιση λοιπόν για αυτό πρόβλημα είναι να έχουμε μια συλλογή από *approximate distributed counters* , όπου το καθένα τον βλέπουμε ως ξεχωριστή οντότητα. Αυτό το οποίο θέλουμε είναι να ορίσουμε το κατάλληλο *approximation factor ε* για κάθε ένα από τους *approximate counter*s που έχουμε στην διάθεση μας, με σκοπό να επιτύχουμε ένα *ε-approximation* από το *MLE* του *joint probability distribution* χρησιμοποιώντας ταυτόχρονα όσον το δυνατόν λιγότερο *communication cost* γίνεται.

Αυτό μπορεί να γίνει με τους τρεις αλγορίθμους *BASELINE, UNIFORM, NON_UNIFORM* [18] όπου καθορίζουν τον τρόπο με τον οποίο θα ορίσουμε το *approximation factor ε* σε συνδυασμό με τον κατάλληλο *approximate distributed counter*. Συγκεκριμένα τα *approximate distributed counters* που χρησιμοποιούνται είναι ο *RANDOMIZED counter* και *DETERMINISTIC counter* και αναλύονται στην συνέχεια.

EXACT counters

Ο πρώτος τρόπος για μπορέσουμε να έχουμε *continuous maintenance* από το *MLE*,το όποιο απαιτεί και το *continuous maintenance* από τα αντίστοιχα *counters*, είναι να χρησιμοποιήσουμε *EXACT counters*(*Αλγόριθμος 1,2*). Με αυτόν τον τρόπο μπορούμε να έχουμε πάντα το *EXACTMLE* από το *joint probability distribution* αλλά όπως θα δούμε και στην συνέχεια το *communication cost* γρήγορα θα αποτελεί το *bottleneck* ολόκληρου του συστήματος.

Algorithm 1: *Exact Counter* (*EXACT*) - *Worker*

Input: $\boldsymbol{x} = < x_1, \dots, x_n >$ *is a full particle of nodes from Bayesian Network*

1. **If** (*received = `INPUT`*)
2. **Foreach** x_i **from** $\boldsymbol{x}$
3. **If**(X_i *has parents*) **then**
4. *// Get the counters correspond to the parents' nodes of* X_i
5. *GET_COUNTER* (x_i^{par})
6. *INCREMENT_ COUNTER* (x_i^{par})
7. *SEND_MESSAGE* (x_i^{par},*`INCREMENT`*)
8. **End**
9.
10. *GET_COUNTER* (x_i)
11. *INCREMENT_COUNTER* (x_i)
12. *SEND_MESSAGE* (x_i ,*`INCREMENT`*)
13. **End**
14. **End**

Η βασική ιδέα από τον κάθε *EXACT counter* που έχουμε είναι ότι για κάθε *increment* που γίνεται στο *counter* στέλνεται ταυτόχρονα και το αντίστοιχο μήνυμα στο *Coordinator* (*Αλγόριθμος 1*) για να ενημερώσει την τιμή από τον *counter*(*Αλγόριθμος 2*). Με αυτόν τον τρόπο χάνουμε οποιοδήποτε όφελος από το *distributed* περιβάλλον ενώ ταυτόχρονα το *communication cost* γίνεται γρήγορα υψηλό.

Συγκεκριμένα αν υποθέσουμε ότι έχουμε ένα Bayesian Network Structure $\mathcal{G}(\mathcal{X}, \mathcal{E})$, το οποίο αποτελείται από n κόμβους , τότε στην περίπτωση που χρησιμοποιήσουμε *EXACT counters* προκύπτει ότι το συνολικό *communication cost* για να μπορέσουμε να διατηρήσουμε με συνεχή τρόπο(*continuously maintenance*) το *MLE* του $\mathcal{G}$, είναι ίσο με $O(m \cdot n)$ δεδομένου ότι έχουμε m *observations,* δηλαδή αντιστοιχεί στο να στείλουμε m μηνύματα μεγέθους n. Τέλος μπορούμε να παρατηρήσουμε ότι με αυτόν τον τρόπο το *communication cost* αυξάνεται γραμμικά με τον αριθμό των *observations* από το *data stream* , επομένως θέλουμε να αποφύγουμε την γραμμική εξάρτηση στο αριθμό των *observations*.

Algorithm 2: Exact Counter (*EXACT*) - *Coordinator*

Input:

type_message: corresponds to one of the available types of messages
message: corresponds to a message

1. **If** (*type_message = `INCREMENT`*)
2. *GET_COUNTER_MSG (message)*
3. *UPDATE_COUNTER*

4.1.1 Approximated Distributed counters

Για να αποφύγουμε το πρόβλημα της γραμμικής εξάρτησης στον αριθμό των *training instances* πλέον χρειάζεται να μπορούμε να διατηρήσουμε ένα *(ε,δ)-approximation* από το *MLE* που αντιστοιχεί στο Bayesian Network $\mathcal{G}(\mathcal{X}, \mathcal{E})$. Για να γίνει αυτό βέβαια πρέπει να χρησιμοποιήσουμε τα κατάλληλα *approximate distributed counters* , τα οποία ταυτόχρονα συνοδεύονται με το μικρότερο δυνατό *communication cost.* Συγκεκριμένα ο τύπος των *approximate distributed counters* που χρησιμοποιούμε είναι ο *RANDOMIZED* και ο *DETERMINISTIC* και αναλύονται στην συνέχεια. Τέλος ο *Πίνακας 2* περιέχει το *space* και το *communication cost* από τον κάθε *counter* ενώ ο *Πίνακας 1* περιέχει όλο τη σημειογραφία(*notation*) που χρησιμοποιείται τόσο από τον *RANDOMIZED* όσο και από τον *DETERMINISTIC counter*.

Count-Tracking πρόβλημα

Θεωρούμε λοιπόν ότι το κάθε *site* S_i με $i \in \{1, \dots, k\}$ διατηρεί ένα *local counter* n_i , όπου αρχικά είναι μηδέν . Καθώς δέχεται δεδομένα το κάθε *site* S_i το μόνο που χρειάζεται να κάνει είναι να αυξήσει το *local counter* $n_i(t)$, με το $n_i(t)$ ισοδυναμεί με την τιμή του *local counter* n_iτην χρονική στιγμή t . Αν υποθέσουμε ότι το κάθε *update* για κάθε *site* S_i είναι της μορφής $\langle i, c \rangle$, τότε το κάθε site S_i το μόνο που χρειάζεται να κάνει είναι να αυξήσει το *local counter* $n_i(t)$ κατά c (*Cash register model*). Υποθέτουμε ότι το $c > 0$ και συγκεκριμένα ισχύει ότι το

$c = 1$,δηλαδή έχουμε μόνο *insertions*. Επομένως το κάθε *local counter* n_i μπορούμε να το δούμε ως μια *linear,monotonic increasing* συνάρτηση(αντίστοιχα ισχύει και για το συνολικό *counter*).

Στόχος μας είναι ο *Coordinator* να έχει ένα *(ε,δ)-approximation* από το συνολικό *counter* $n(t) = \sum_{i=1}^{k} n_i(t)$ ανά πάσα χρονική στιγμή δεδομένου ότι $0 \leq \varepsilon, \delta \leq 1$, δηλαδή να ισχύει ότι:

$$(1-\varepsilon)\cdot n(t) \leq \hat{n}(t) \leq (1+\varepsilon)\cdot n(t) \text{ με πιθανότητα } 1-\delta$$

Μόνο που αυτό πρέπει να γίνει χρησιμοποιώντας όσο το δυνατόν λιγότερο *communication cost*. Επιπλέον όπως παρατίθεται στο [19], έχει αποδειχθεί ότι το ελάχιστο *communication cost* που απαιτείται για να διατηρήσουμε έναν τέτοιο *counter* είναι ίσο με $O(k/\varepsilon \cdot logN)$, με το N αντιστοιχεί στην τελική τιμή του συνολικού *counter* n ενώ το k αντιστοιχεί στον αριθμό των *sites*.

Συμβολισμός	*Περιγραφή*
$\boldsymbol{n_i}$	*Ο "τοπικός" counter του* S_i
$\boldsymbol{n}$	*Ο counter πάνω σε όλα τα sites*
$\boldsymbol{\bar{n}_i}$	*Η τελευταία τιμή που στάλθηκε από το* S_i *στον Coordinator*
$\boldsymbol{p}$	*Πιθανότητα* $p \in [0,1)$. *Αρχικά* $p = 1$
$\boldsymbol{\hat{n}_i}$	*Η εκτίμηση του* n_i
$\boldsymbol{\hat{n}}$	*Η εκτίμηση πάνω σε όλα sites (το άθροισμα από τα* $\hat{n}_i$*)*
$\boldsymbol{\bar{n}}$	*Η τελευταία τιμή που στάλθηκε από τον Coordinator*
$\boldsymbol{\lfloor x \rfloor_2}$	*Η επόμενη δύναμη του 2 που είναι μικρότερη από το* x
$\boldsymbol{n_i'}$	*Η τελευταία ενημέρωση της τιμής του* n_i *στον Coordinator*
$\boldsymbol{n'}$	*Το άθροισμα από τα* n_i'
$\boldsymbol{N}$	*Η τελική τιμή του counter*
$\boldsymbol{\varepsilon}$	*Approximation factor*
$\boldsymbol{\delta}$	*Delta, αναφέρεται μόνο σε randomized counters*
$\boldsymbol{k}$	*Ο αριθμός των sites*

Πίνακας 1: Περίληψη σημειογραφίας

4.1.1.1 Randomized Counters

Ο πρώτος *counter* που υλοποιήσαμε και αναλύουμε στην συνέχεια είναι ο *RANDOMIZED counter*. Επιπλέον έχει αποδειχθεί ότι ο *RANDOMIZED counter* αποτελεί και το *lower communication cost* που μπορούμε να επιτύχουμε στο *count-tracking* πρόβλημα, όπως παρατίθενται στο [19]. Το *communication cost* που απαιτείται ισούται με $O(\sqrt{k}/\varepsilon \cdot logN)$ και για να το επιτύχουμε αυτό όπως προκύπτει και από το όνομα του, είναι να χρησιμοποιήσουμε κάποιον *randomized* αλγόριθμο. Συγκεκριμένα χρησιμοποιώντας το *randomization* επιτυγχάνουμε να μειώσουμε το *communication cost* κατά $\sqrt{k}$ σε σχέση με το *communication cost* από οποιοδήποτε *deterministic* μέθοδο.

Η βασική ιδέα βασίζεται στο γεγονός ότι χρησιμοποιώντας το *randomization* μπορούμε να παράγουμε έναν *unbiased estimator* για το n_i, δηλαδή να ισχύει ότι $E[\hat{n}_i] = n_i$ ενώ το *variance* του *estimator* να ισούται με $Var[\hat{n}_i] = (\varepsilon n)^2/k$.

Δεδομένου ότι το *variance* από το κάθε *estimator* είναι ίσο με $(\varepsilon n)^2/k$, τότε μπορούμε να αποδείξουμε ότι το συνολικό *variance* ισούται με $(\varepsilon n)^2$, δηλαδή ισχύει ότι (*independence of* $n_i's$):

$$Var[\hat{n}] = Var\left[\sum_{i=1}^{k} \hat{n}_i\right] = \sum_{i=1}^{k} Var[\hat{n}_i] = k \cdot Var[\hat{n}_i] = (\varepsilon n)^2$$

Το οποίο με τη σειρά του είναι αρκετό για να παράγουμε έναν *estimator* $\hat{n}$ με error εn με πιθανότητα $1-\delta$. Με το τελευταίο να προκύπτει χρησιμοποιώντας το *Chebyshev's inequality*.

Αντίστοιχα, δεδομένου ότι το *variance* από κάθε *unbiased estimator* $\hat{n}_i$ ισούται με $(\varepsilon n)^2/k$ τότε είναι αρκετό για να υποστηρίξουμε *error* το οποίο ισούται με $\varepsilon n/\sqrt{k}$, σε αντίθεση με $\varepsilon n/k$ που απαιτείται από κάθε *deterministic* μέθοδο. Επομένως με αυτόν τον τρόπο μπορούμε να επιτύχουμε μια βελτίωση κατά $\sqrt{k}$. Το τελευταίο προκύπτει χρησιμοποιώντας το *Chebyshev's inequality*, συγκεκριμένα πρέπει να ισχύει ότι:

$$\frac{Var[\hat{n}_i]}{(error)^2} \leq 1 \Rightarrow error \geq \varepsilon n/\sqrt{k}$$

Επομένως κάθε *site* S_i πλέον όποτε λαμβάνει κάποιο *event* δηλαδή έχουμε μια αύξηση στο *local counter* n_i, τότε στέλνει την τελευταία τιμή του n_i στον Coordinator με πιθανότητα p. Όσον αφορά το *estimation* που έχει ο *coordinator* για κάθε *local counter* n_i, ισχύει ότι:

$$\hat{n}_i = \begin{cases} \bar{n}_i - 1 - 1/p\,, & \alpha\nu\ \tau o\ \ \bar{n}_i\ \upsilon\pi\acute{\alpha}\rho\chi\varepsilon\iota \\ 0\,, & \delta\iota\alpha\varphi o\rho\varepsilon\tau\iota\kappa\acute{\alpha} \end{cases}$$

Με το $\bar{n}_i$ να αντιστοιχεί στην τελευταία τιμή(*last update*) που έχε ο *Coordinator* από το *local counter* n_i ενώ το συνολικό *estimation* από το n που έχει ο *Coordinator* ισούται με $\hat{n} = \sum_{i=1}^{k} \hat{n}_i$.

Με βάση και τα προηγούμενα αρκεί να αποδείξουμε ότι το $\hat{n}_i$ είναι ένας *unbiased estimator* με $Var[\hat{n}_i] \leq 1/p^2$. Συνέπεια αυτού το συνολικό *estimation* $\hat{n}$ να είναι ένας *unbiased estimator* με $Var[\hat{n}] \leq k/p^2$. Επομένως θέτοντας κατάλληλα την πιθανότητα , συγκεκριμένα αρκεί $p = \Theta(\sqrt{k}/\varepsilon n)$ τότε προκύπτει ότι $Var[\hat{n}] = (\varepsilon n)^2$, το οποίο όπως είπαμε και προηγουμένως είναι αρκετό για να έχουμε οποιαδήποτε χρονική στιγμή έναν *estimator* $\hat{n}$ με *error* εn το οποίο ισχύει με πιθανότητα $1-\delta$. Η απόδειξη παρατίθεται αναλυτικά στο [19].

Δεδομένου ότι το n δεν είναι προσβάσιμο από τα *sites* και ισχύει ότι $p = \Theta(\sqrt{k}/\varepsilon n)$, αρκεί το κάθε *site* S_i να έχει κάποιο *constant-factor approximation* από το n σε κάθε *round*. Αυτό μπορεί να γίνει με τον ακόλουθο τρόπο , πέραν από το ότι κάθε *site* S_i για κάθε *increment* στέλνει ένα μήνυμα στον *Coordinator* με πιθανότητα p(*INCREMENT*), πλέον το κάθε site S_i στέλνει ένα επιπλέον μήνυμα κάθε φορά που το *local counter* n_i διπλασιάζεται(*DOUBLES*). Αντίστοιχα ο *Coordinator* όταν λάβει το αντίστοιχο μήνυμα *DOUBLES* τότε ενημερώνει την μεταβλητή $n' = \sum_{i=1}^{k} n_i'$, με το n_i' να αντιστοιχεί στο τελευταίο *update* που έχει ο *Coordinator* για τον *local counter* n_i. Όταν το n' διπλασιαστεί (συγκεκριμένα όταν αλλάξει κατά ένα παράγονται ανάμεσα στο 2 και το 4) , τότε ο *Coordinator* κάνει *broadcast* το n' σε όλα τα *sites* και ένα καινούργιο *round* ξεκινάει. Με αυτό τον τρόπο καταφέρνουμε να έχουμε ένα *constant-factor approximation* από το n σε κάθε *round* , το οποίο μπορούν να το

χρησιμοποιήσουν τα *sites* ώστε να ορίσουν το p. Υποθέτουμε ότι το $\bar{n}$ ισούται με την τελευταία τιμή του n' που έγινε *broadcast*.

Συγκεκριμένα όσο το $\bar{n} \leq \sqrt{k}/\varepsilon$ τότε ισχύει ότι $p = 1$. Επομένως τα πρώτα $O(\sqrt{k}/\varepsilon)$ στοιχεία θα σταλθούν στον *Coordinator* αφού ισχύει ότι $p = 1$. Όταν το $\bar{n} > \sqrt{k}/\varepsilon$ τότε θέτουμε $p = 1/\lfloor \varepsilon\bar{n}/\sqrt{k} \rfloor_2$. Επομένως μπορούμε να παρατηρήσουμε ότι καθώς το $\bar{n}$ αυξάνεται σε κάθε *round* αντίστοιχα και η πιθανότητα θα μειώνεται(συγκεκριμένα υποδιπλασιάζεται σε κάθε *round*).

Στην αρχή κάθε *round* όπως είπαμε και πριν η πιθανότητα μειώνεται , επομένως πρέπει να προσαρμόσουμε τα $\bar{n}_i$ που έχει ο *Coordinator* προκειμένου να φαίνεται ότι ολόκληρο το σύστημα έτρεχε με την καινούργια πιθανότητα p. Αυτό μπορεί να γίνει με τον ακόλουθο τρόπο, συγκεκριμένα κάθε *site* S_i αποφασίσει με πιθανότητα $p = 1/2$ αν το $\bar{n}_i$ παραμείνει το ίδιο. Αν αποφασίσει να αλλάξει τότε κάθε *site* S_i κάνει *flip* ένα *coin* με πιθανότητα $1/p$ (το καινούργιο p) επαναλαμβανόμενα. Για κάθε *failed coin flip* μειώνει το $\bar{n}_i$ κατά 1. Αυτό επαναλαμβάνεται μέχρι να έχουμε ένα *successful coin flip* ή το $\bar{n}_i = 0$. Στο τέλος το *site* S_i στέλνει μήνυμα στο *Coordinator* με την καινούργια τιμή του $\bar{n}_i$. Το τελευταίο βήμα είναι απαραίτητο γιατί με αυτόν τον τρόπο καταφέρνουμε να αποφύγουμε το *bias* που θα δημιουργούταν στον *estimator* $\hat{n}_i$ με την αλλαγή του p. Όλη η διαδικασία που έχει περιγράφει προηγουμένως συνοψίζεται στον *Αλγόριθμο 3* και *4*.

Algorithm 3: *Randomized Counter (RC) - Worker*

Input: *epsilon, delta*
type_message: *corresponds to one of the available types of messages*

1. **If** (n_i *increment by one*)
2. *SEND_MESSAGE* (n_i,p,`*INCREMENT*`)
3. *UPDATE* ($\bar{n}_i$)
4. **If** (n_i *doubles*)
5. *SEND_MESSAGE* (n_i', p=1, `*DOUBLES*`)
6. *UPDATE* ($\bar{n}_i$)
7. **If** (*type_message = `UPDATE`*)
8. *// New round begins*
9. *UPDATE* ($\bar{n}$)
10. **If** ($\bar{n} < \sqrt{k}/\varepsilon$) *Set p=1*
11. **Else**
12. *Set* $p = 1/\lfloor \varepsilon\bar{n}/\sqrt{k} \rfloor_2$
13. **If** (*p halved*)
14. *// Adjust the* n_i
15. *SEND_MESSAGE* (n_i,$p = \frac{1}{2}$,`*INCREMENT*`)
16. **If** (*loose flip*)
17. *While (Flip until succeed)*
18. *SEND_MESSAGE* (**n_i-num_of_failures,$p = 1$,`*INCREMENT*`**)
19. *UPDATE* ($\bar{n}_i$)

Τέλος όσον αφορά το *communication cost* για τον *RANDOMIZED counter*, ισχύει ότι το *communication cost* για ένα *round* αποτελείται από το *communication cost* που απαιτείται για το *broadcast* της τιμής $\bar{n}$ από τον *Coordinator*(*upstream communication cost*) το οποίο ισούται με $O(k)$ και το *communication cost* που απαιτείται να στείλουν τα *sites* στον *coordinator*(*downstream communication cost*) το οποίο ισούται με $O(np)$. Επομένως το *communication cost* που απαιτείται για ένα *round* ισούται με $O(k + np) = O(k + \sqrt{k}/\varepsilon) =$

$O(\sqrt{k}/\varepsilon)$. Δεδομένου ότι έχουμε $logN$ rounds, προκύπτει ότι το συνολικό *communication cost* ισούται με $O(\sqrt{k}/\varepsilon \cdot logN)$.

Algorithm 4: *Randomized Counter (RC) - Coordinator*

Input: *type_message corresponds to one of the available types of messages*

1. **If** (*type_message = `INCREMENT`*)
2. *UPDATE ($\bar{n}_i$)*
3. *CALCULATE ($\hat{n}_i$)*
4. **If** (*type_message = `DOUBLES`*)
5. *UPDATE (n_i')*
6. *UPDATE (n')*
7. **If (**n'*doubles***)** *// Change by factor between 2 and 4*
8. *// New round begins*
9. *Set $\bar{n} = n'$*
10. *BROADCAST($\bar{n}$,`UPDATE`)*

4.1.1.2 Deterministic Counters

Ο δεύτερος *counter* που υλοποιήσαμε και αναλύουμε στην συνέχεια είναι ο *DETERMINISTIC counter*. Παρόμοια προσέγγιση έχει χρησιμοποιηθεί τόσο από το [17] όσο και από το [20]. Όπως προκύπτει και από το όνομα σε αυτήν την περίπτωση εστιάζουμε σε *deterministic* αλγόριθμο και πλέον επιδιώκουμε να έχουμε ένα *ε-approximation* του *counter* , δηλαδή θέλουμε για κάθε χρονική στιγμή t να ισχύει ότι:

$$(1-\varepsilon)\cdot n(t) \leq \hat{n}(t) \leq (1+\varepsilon)\cdot n(t)$$

Την βασική ιδέα του αλγορίθμου μπορούμε να την δούμε με τον ακόλουθο τρόπο. Δεδομένου ότι θέλουμε να έχουμε *ε-approximation* του *counter* , το *error* το οποίο μπορούμε να υποστηρίξουμε σε κάθε χρονική στιγμή είναι ίσο με εn . Υποθέτοντας ότι ο αριθμός των "*ανεξάρτητων*" *sites* που έχουμε ισούται με k, είναι εύκολο να δούμε ότι το μέγιστο *error* που μπορεί να έχει το κάθε *site* ώστε να έχουμε ένα *ε-approximation* πρέπει να ισούται με $\varepsilon n/k$.

Ο τρόπος με το οποίο μπορεί να γίνει αυτό συνοψίζεται στους *Αλγορίθμους 4,5* και αναλύεται στην συνέχεια. Δεδομένου ότι κάθε *site* είναι απίθανο να γνωρίζει για κάθε χρονική στιγμή την τιμή του n , μπορούμε αντί αυτού το κάθε *site* να έχει ένα *constant factor approximation* του n μέσω του $\bar{\boldsymbol{n}}$ για κάθε *round*. Το $\bar{\boldsymbol{n}}$ ισοδυναμεί με την τελευταία τιμή που έγινε *broadcast* από τον *Coordinator*. Αρχικά θεωρούμε ότι το $\bar{\boldsymbol{n}} = 0$. Κάθε *site* S_i πλέον στέλνει ένα μήνυμα στο *Coordinator* κάθε φόρα που το $n_i > \varepsilon\bar{n}/k$(*INCREMENT*). Όταν ο *Coordinator* λάβει k τέτοια μηνύματα , δηλαδή έχουμε φτάσει στο μέγιστο δυνατό *error* τότε κάνει *broadcast* ένα μήνυμα σε όλα τα *sites* προκειμένου να συγκεντρώσει τα *local counters* από το κάθε *site(DRIFT)*. Στην συνέχεια για κάθε τέτοιο *DRIFT* μήνυμα που λαμβάνει ενημερώνει την τιμή του $\bar{\boldsymbol{n}}$**,** μόλις λάβει k τέτοια μηνύματα δηλαδή έχει συγκεντρώσει το *local counter* από κάθε *site* τότε κάνει *broadcast* ένα *UPDATE* μήνυμα σε όλα τα *sites* με την καινούργια τιμή του $\bar{\boldsymbol{n}}$**.** Αυτό το σημείο αντιστοιχεί στο ξεκίνημα ενός καινούργιο *round*. Τέλος μπορούμε να παρατηρήσουμε ότι τα πρώτα $O(k/\varepsilon)$ στοιχεία θα σταλθούν στον *Coordinator* αφού το $\varepsilon\bar{n}/k \leq 1$.

Algorithm 4: *Deterministic Counter (DC) - Worker*

Input: *epsilon*
type_message corresponds to one of the available types of messages

1. **If** (*type_message = `INPUT`*)
2. *Increment n_i // local drift*
3. **If** ($n_i > \varepsilon\bar{n}/k$)
4. *SEND_MESSAGE (n_i,$p = 1$,`INCREMENT`)*
5. *Reset the local counter n_i // local drift*
6.
7. **If** (*type_message = `DRIFT`*)
8. *SEND_MESSAGE (n_i,$p = 1$,`DRIFT`)*
9. *Reset the local counter n_i // local drift*
10.
11. **If** (*type_message = `UPDATE`*)
12. *// New round begins*
13. *UPDATE ($\bar{n}$)*

Τέλος όσον αφορά το *communication cost* για τον *DETERMINISTIC counter*, ισχύει ότι το *communication cost* για ένα *round* αποτελείται από το *communication cost* που απαιτείται για το *broadcast* της τιμής $\bar{n}$ όσο και ενός *DRIFT* μηνύματος από τον *Coordinator*(*upstream communication cost*) το οποίο ισούται με $O(k)$ και το *communication cost* που απαιτείται να στείλουν τα *sites* στον *Coordinator*(*downstream communication cost*) το οποίο ισούται με $O(k + k/\varepsilon)$. Επομένως το *communication cost* που απαιτείται για ένα *round* ισούται με $O(k + k/\varepsilon) = O(k/\varepsilon)$. Δεδομένου ότι ο *Coordinator* κάνει *broadcast* κάθε φορά που το $\bar{n}$ αυξηθεί κατά ένα παράγοντα που ισούται με $1 + \sum_{i=1}^{k} \varepsilon/k = 1 + \varepsilon$ τότε ο αριθμός των *rounds* φράζεται από το $O(\log_{1+\varepsilon} N)$, επομένως προκύπτει ότι το συνολικό *communication cost* ισούται με $O(k/\varepsilon \cdot logN)$.

Algorithm 5*: Deterministic Counter (DC) – Coordinator*

Input: *type_message corresponds to one of the available types of messages*

1. **If** (*type_message = `INCREMENT`*)
2. *UPDATE ($\bar{n}_i$)*
3. *CALCULATE ($\hat{n}_i$)*
4. **If** (num_messages_inc = k)
5. *BROADCAST(`DRIFT`)*
6.
7. **If** (*type_message = `DRIFT`*)
8. *UPDATE ($\bar{n}$)*
9. **If** (*num_messages_drift = k*)
10. *// New round begins*
11. *BROADCAST($\bar{n}$,`UPDATE`)*

Count-Tracking πρόβλημα	*Space (ανά site)*	*Communication cost*
EXACT	$O(1)$	$O(N)$
RANDOMIZED	$O(logN)$	$O(\sqrt{k}/\varepsilon \cdot logN)$
DETERMINISTIC	$O(logN)$	$O(k/\varepsilon \cdot logN)$

Πίνακας 2: Space και communication cost από τον κάθε counter

4.1.1.3 Σύγκριση μεταξύ RANDOMIZED και DETERMINISTIC

Η μόνη διαφορά ανάμεσα στον *DETERMINISTIC* και *RANDOMIZED counter* εστιάζεται στην εξάρτηση από τον αριθμό των *sites* k. Συγκεκριμένα ο *RANDOMIZED counter* είναι καλύτερος κατά ένα παράγοντα $\sqrt{k}$. Με το τελευταίο να επιβεβαιώνεται και στο πειραματικό στάδιο, όπου καθώς μεταβάλλαμε τον αριθμό των *sites* k, η μεταβολή του *communication cost* του *RANDOMIZED counter* ήταν πιο μικρή σε σχέση με του *DETERMINISTIC counter.*

Ο λόγος που αποφασίσαμε να υλοποιήσουμε τον *DETERMINISTIC counter* πρώτον αφορά λόγους πληρότητας δηλαδή να έχουμε μια γενική εικόνα από κάθε τύπο *approximate counter* που μπορούμε να χρησιμοποιήσουμε και δεύτερον οφείλεται στο γεγονός ότι όταν το k είναι μικρό, ισχύει το ανάποδο δηλαδή ο *DETERMINISTIC counter* εμφανίζεται να έχει μικρότερο *communication cost*.

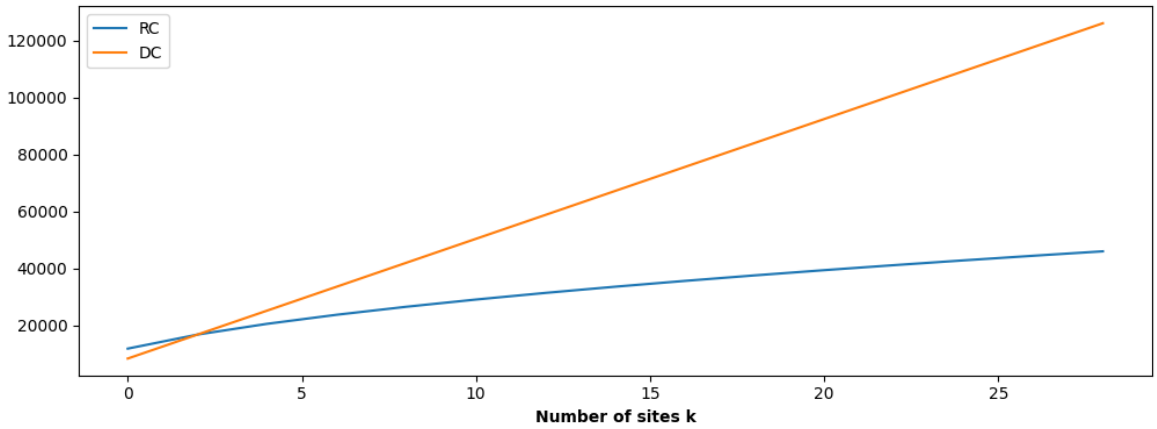
Όπως είπαμε προηγουμένως το στοιχείο που καθορίζει κάθε πότε στέλνουμε στοιχεία , για το *DETERMINISTIC counter* είναι το $\varepsilon \cdot \bar{n}_{DC}/k$ ενώ για τον *RANDOMIZED counter* είναι το $p = 1/\lfloor \varepsilon \cdot \bar{n}_{RC}/\sqrt{k} \rfloor_2$. Επομένως μπορούμε να παρατηρήσουμε ότι ο αριθμός των στοιχείων που στέλνονται από κάθε τύπο καθορίζεται από την τιμή $\bar{n}_{DC}$ και την τιμή $\bar{n}_{RC}$ αντίστοιχα.

Σχεδόν σε όλες τις περιπτώσεις ισχύει ότι το $\bar{n}_{RC} < \bar{n}_{DC}$ επομένως αναμένουμε να σταλθούν και περισσότερα μηνύματα από τον *DETERMINISTIC counter*. Αν υποθέσουμε ότι ο *DETERMINISTIC counter* στέλνει κάθε στοιχείο με πιθανότητα p' που καθορίζεται αποκλειστικά από την ποσότητα $\varepsilon \cdot \bar{n}_{DC}/k$, παρόλο που το $\bar{n}_{RC} < \bar{n}_{DC}$ δηλαδή απαιτούνται περισσότερα μηνύματα να σταλθούν για να έχουμε την πιθανότητα p', αυτό το οποίο ισχύει μετέπειτα είναι ότι το $p' > p$. Άρα παρόλο που ο *DETERMINISTIC counter* στέλνει περισσότερα μηνύματα για να φτάσει την τιμή του $\bar{n}_{DC}$ μετά αποκτά πλεονέκτημα γιατί ισχύει ότι το $p' > p$.

Επομένως όσο μικρότερη είναι η διαφορά ανάμεσα στο $\bar{n}_{RC}$ και στο $\bar{n}_{DC}$ η οποία καθορίζεται αποκλειστικά από τον αριθμό των *sites* k, δηλαδή όσο μικρότερος είναι ο αριθμός των *sites* τόσο μικρότερη είναι και η μεταξύ τους διαφορά, τόσο περισσότερο και το πλεονέκτημα που αποκτά ο *DETERMINISTC counter*. Ενώ όταν ισχύει το ανάποδο δηλαδή όσο μεγαλύτερη είναι η διαφορά ανάμεσα στο $\bar{n}_{RC}$ και στο $\bar{n}_{DC}$, το οποίο συνεπάγεται ότι έχουμε μεγάλο αριθμό από *sites* , τότε ο *DETERMINISTIC counter* χάνει το οποιοδήποτε πλεονέκτημα γιατί ο αριθμός των μηνυμάτων για να φτάσει την τιμή του $\bar{n}_{DC}$ είναι πολύ μεγαλύτερος από τον αριθμό των μηνυμάτων που απαιτούνται για την τιμή του $\bar{n}_{RC}$, επομένως το μετέπειτα πλεονέκτημα από το p' εξανεμίζεται εξαιτίας της μεγάλης διαφοράς ανάμεσα στο $\bar{n}_{DC}$ και στο $\bar{n}_{RC}$, αφού πλέον ισχύει ότι το $\bar{n}_{DC} \gg \bar{n}_{RC}$.

Συνολικά, όπως αποδείχθηκε και στο πειραματικό κομμάτι είναι προτιμότερο ότι όταν ο αριθμός των *sites* είναι μικρός να χρησιμοποιούμε τον *DETERMINISTC counter* ενώ όταν ο αριθμός των *sites* είναι μεγάλος είναι καλύτερο να προτιμήσουμε το *RANDOMIZED counter*.

Τέλος στην *Εικόνα 12* μπορούμε να δούμε τις τιμές που έχουν το $\bar{n}_{RC}$ και το $\bar{n}_{DC}$(άξονας y) καθώς μεταβάλλουμε τον αριθμό των *sites* k, υποθέτοντας ότι $\varepsilon = 4 \times 10^{-4}, \delta = 0.25$ και $k \in [2{,}30]$. Παρατηρούμε λοιπόν ότι καθώς αυξάνεται ο αριθμός των *sites* τόσο μεγαλύτερη είναι και η διαφορά μεταξύ του $\bar{n}_{RC}$ και του $\bar{n}_{DC}$.



Εικόνα 12: Τιμές του $\bar{n}_{RC}$ και του $\bar{n}_{DC}$για διάφορες τιμές του k

4.1.2 Ανάλυση των αλγορίθμων BASELINE, UNIFORM και NON_UNIFORM

Πέραν από την γενική προσέγγιση και τα *approximated distributed counters* που έχουν αναλυθεί προηγουμένως, το μόνο πράγμα το οποίο έμεινε να δούμε είναι πως θα χωρίσουμε το διαθέσιμο *"error budget"* μεταξύ των *approximate distributed counter* $A_i(x_i, x_i^{par})$, $A_i(x_i^{par})$, με τέτοιο τρόπο ώστε ανά πάσα στιγμή να έχουμε *(ε,δ)-approximation* από *MLE*.

Παρακάτω παραθέτουμε και αναλύουμε τους τρεις βασικούς αλγορίθμους *BASELINE, UNIFORM, NON_UNIFORM* οι οποίο καθορίζουν τον τρόπο με τον οποίο μπορούμε να χωρίσουμε το διαθέσιμο *"error budget"*.

Η ανάλυση των τριών αλγορίθμων γίνεται με βάση τον *RANDOMIZED counter*. Ο *Πίνακας 3* περιέχει τα αποτελέσματα για κάθε ένα από τους τρεις αλγορίθμους με βάση τον *RANDOMIZED counter*. Όσον αφορά τον *DETERMINISTIC counter,* η διαφορά στο συνολικό *communication cost* εντοπίζεται στις παραμέτρους δ και k . Συγκεκριμένα για το *DETERMINISTIC counter* δεν υπάρχει εξάρτηση από το δ ενώ η εξάρτηση στον αριθμό των *sites* k είναι γραμμική.

Αρχικά θεωρούμε ότι έχουμε ένα Bayesian Network Structure $\mathcal{G}(\mathcal{X}, \mathcal{E})$, με το σύνολο των κόμβων να αντιστοιχεί στο $\mathcal{X} = \{X_1, \dots, X_n\}$, δηλαδή έχουμε ένα *Bayesian Network* με n κόμβους. Επιπλέον συμβολίζουμε με J_i το *cardinality* του συνόλου τιμών του κόμβο X_i δηλαδή ισχύει ότι $J_i = |Val(X_i)|$ για κάθε $i \in \{1, \dots, n\}$ ενώ το K_i αντιστοιχεί στο *cardinality* του συνόλου τιμών από τους *parents* κόμβους του κόμβου X_i δηλαδή ισχύει ότι $K_i = |Val(Par(X_i))|$ για κάθε $i \in \{1, \dots, n\}$. Τέλος το k υποδηλώνει τον αριθμό των *sites*.

Επιπροσθέτως θεωρούμε ότι το $J = max_{i=1}^{n} J_i$ δηλαδή αντιστοιχεί στο κόμβο με το μεγαλύτερο σύνολο τιμών και ότι το $d = max_{i=1}^{n} |Par(X_i)|$ αντιστοιχεί στον μέγιστο αριθμό από *parents* κόμβους που έχει κάποιος(ή κάποιοι) κόμβος στο δίκτυο. Τέλος θεωρούμε ότι έχουμε στην διάθεση μας δυο παραμέτρους $0 \leq \varepsilon, \delta \leq 1$.

BASELINE

Η πρώτη προσέγγιση είναι ο *BASELINE* αλγόριθμος. Συγκεκριμένα ο αλγόριθμος *BASELINE* για κάθε *approximate distributed counter* $A_i(x_i, x_i^{par})$, $A_i(x_i^{par})$ ορίζει το *approximation factor* ίσο με $\frac{\varepsilon}{3n}$. Ο αλγόριθμος *BASELINE* εξασφαλίζει ότι ανά πάσα χρονική στιγμή έχουμε ένα *(ε,δ)-approximation* από το *MLE* του *joint probability distribution* που αντιστοιχεί στο $\mathcal{G}$. Η απόδειξη παρατίθεται αναλυτικά στο [18, Section IV-C]. Παρατηρούμε ότι με αυτήν την προσέγγιση το *error budget* μοιράζεται ομοιόμορφα μεταξύ των *counters*.

Το συνολικό *communication cost* που απαιτείται δεδομένου *m training instances* ισούται με:

$$O\left(\frac{n^2 \cdot J^{d+1}\sqrt{k}}{\varepsilon} \cdot \log\frac{1}{\delta} \cdot \log \mathrm{m}\right)$$

Το *communication cost* από κάθε *approximate distributed counter* σε αυτήν την περίπτωση ισούται με $O(\frac{n\cdot\sqrt{k}}{\varepsilon} \cdot \log\frac{1}{\delta} \cdot \log \mathrm{m})$. Επιπλέον για κάθε $i \in \{1, \dots, n\}$ δηλαδή για κάθε κόμβο X_i υπάρχουν το πολύ J^{d+1} *counters* της μορφής $A_i(x_i, x_i^{par})$ και το πολύ J^d *counters* της μορφής $A_i(x_i^{par})$, για κάθε $x_i \in Val(X_i)$ και $x_i^{par} \in Val(Par(X_i))$. Συνδυάζοντας τα δύο προηγούμενα προκύπτει ότι το συνολικό *communication cost* ισούται με $O(\frac{n^2 \cdot J^{d+1}\sqrt{k}}{\varepsilon} \cdot \log\frac{1}{\delta} \cdot \log \mathrm{m})$. Τέλος μπορούμε να παρατηρήσουμε ότι ακόμα και την πιο απλή περίπτωση δηλαδή τον αλγόριθμο *BASELINE* καταφέραμε να αποφύγουμε την γραμμική εξάρτηση στον αριθμό των *training instances m*, πλέον έχουμε λογαριθμική εξάρτηση.

UNIFORM

Μια προσεκτική ανάλυση οδηγεί στον αλγόριθμο *UNIFORM*. Συγκεκριμένα ο αλγόριθμος *UNIFORM* για κάθε *approximate distributed counter* $A_i(x_i, x_i^{par})$, $A_i(x_i^{par})$ ορίζει το *approximation factor* ίσο με $\frac{\varepsilon}{16\sqrt{n}}$. Χρησιμοποιώντας αυτό το *approximation factor* για κάθε *counter* και πάλι ο αλγόριθμος *UNIFORM* εξασφαλίζει ότι ανά πάσα χρονική στιγμή έχουμε ένα *(ε,δ)-approximation* από το *MLE* του *joint probability distribution* που αντιστοιχεί στο $\mathcal{G}$. Η απόδειξη παρατίθεται αναλυτικά στο [18, Section IV-C]. Παρατηρούμε και σε αυτήν την περίπτωση το *error budget* μοιράζεται ομοιόμορφα μεταξύ *counters*.

Το συνολικό *communication cost* που απαιτείται δεδομένου *m training instances* ισούται με:

$$O\left(\frac{n^{3/2} \cdot J^{d+1}\sqrt{k}}{\varepsilon} \cdot \log\frac{1}{\delta} \cdot \log \mathrm{m}\right)$$

Σε αυτήν την περίπτωση το *communication cost* από κάθε *approximate distributed counter* ισούται με $O(\frac{\sqrt{nk}}{\varepsilon} \cdot \log\frac{1}{\delta} \cdot \log \mathrm{m})$ δεδομένου ότι το *approximation factor* ισούται με $\frac{\varepsilon}{16\sqrt{n}}$. Επιπλέον για κάθε $i \in \{1, \dots, n\}$ δηλαδή για κάθε κόμβο X_i υπάρχουν το πολύ J^{d+1} counters της μορφής $A_i(x_i, x_i^{par})$ και το πολύ J^d counters της μορφής $A_i(x_i^{par})$, για κάθε $x_i \in Val(X_i)$ και $x_i^{par} \in Val(Par(X_i))$. Συνδυάζοντας τα δύο προηγούμενα προκύπτει ότι το συνολικό *communication cost* ισούται με $O(\frac{n^{3/2} \cdot J^{d+1}\sqrt{k}}{\varepsilon} \cdot \log\frac{1}{\delta} \cdot \log \mathrm{m})$. Τέλος και σε αυτήν την περίπτωση καταφέραμε να έχουμε λογαριθμική εξάρτηση στο αριθμό των *training instances* και επιπλέον καταφέραμε να μειώσουμε το *communication cost* κατά $\sqrt{n}$ σε σχέση με τον αλγόριθμο *BASELINE*.

NON_UNIFORM

Μέχρι στιγμής και οι δυο προηγούμενοι αλγόριθμοι χώριζαν το "*error budget"* ομοιόμορφα σε όλους τους διαθέσιμους *counters*. Κανένας από τους δυο προηγούμενους αλγορίθμους δεν λάμβανε στοιχεία που αφορούν την δομή που έχει ο κάθε κόμβος , δηλαδή στοιχεία όπως για παράδειγμα το *cardinality* του συνόλου των τιμών του, τον αριθμό των *parents* κόμβων που έχει και αυτό έρχεται να το κάνει ο αλγόριθμος *NON_UNIFORM*. Σε αυτήν την περίπτωση το "*error budget"* δεν μοιράζεται ομοιόμορφά σε όλα τα διαθέσιμα *counters* αλλά το *approximation factor* για κάθε *approximate distributed counter* $A_i(x_i, x_i^{par})$, $A_i(x_i^{par})$ είναι πλέον συναρτήσει τόσο του $J_i = |Val(X_i)|$ όσο και του $K_i = |Val(Par(X_i))|$.

Σε αυτήν την περίπτωση υπάρχει διαχωρισμός για τους *counters*. Συγκεκριμένα για κάθε *approximate distributed counter* $A_i(x_i, x_i^{par})$ το *approximation factor* ορίζεται με τον ακόλουθο τρόπο:

$$v_i = \frac{(J_i K_i)^{\frac{1}{3}} \cdot \varepsilon}{16a} \text{, όπου } \alpha = \left(\sum_{i=1}^{n} (J_i K_i)^{\frac{2}{3}} \right)^{1/2}$$

Ενώ για κάθε *approximate distributed counter* $A_i(x_i^{par})$ το *approximation factor* ορίζεται με τον ακόλουθο τρόπο:

$$\mu_i = \frac{(K_i)^{\frac{1}{3}} \cdot \varepsilon}{16\beta} \text{, όπου } \beta = \left(\sum_{i=1}^{n} (K_i)^{\frac{2}{3}} \right)^{\frac{1}{2}}$$

Χρησιμοποιώντας το v_i, μ_i για κάθε *counter* και πάλι μπορούμε να εξασφαλίσουμε ότι ανά πάσα χρονική στιγμή έχουμε ένα *(ε,δ)-approximation* από το *MLE* του *joint probability distribution* που αντιστοιχεί στο $\mathcal{G}$. Η απόδειξη παρατίθεται αναλυτικά στο [18, Section IV-E].

Επιπλέον, μπορούμε να παρατηρήσουμε ότι σε αυτήν την περίπτωση το *approximation factor* για κάθε *approximate distributed counter* της μορφής $A_i(x_i, x_i^{par})$ εξαρτάται τόσο από το J_i όσο και από το K_i δηλαδή είναι ανάλογο τόσο του συνόλου τιμών του X_i όσο και του *cardinality* του συνόλου τιμών από τους *parents* κόμβους του X_i . Αντίστοιχα το *approximation factor* για κάθε *approximate distributed counter* της μορφής $A_i(x_i^{par})$

εξαρτάται αποκλειστικά από το K_i, δηλαδή είναι ανάλογο μόνο από το *cardinality* του συνόλου τιμών από τους *parents* κόμβους του X_i.

Όσον αφορά το *communication cost* που απαιτείται δεδομένου *m training instances* ισούται με:

$$O(\Gamma \cdot \frac{\sqrt{k}}{\varepsilon} \cdot \log \frac{1}{\delta} \cdot \log \mathrm{m}) \text{, όπου } \Gamma = \left(\sum_{i=1}^{n} (J_i K_i)^{\frac{2}{3}}\right)^{3/2} + \left(\sum_{i=1}^{n} (K_i)^{\frac{2}{3}}\right)^{3/2}$$

Σε αυτήν την περίπτωση το *communication cost* από κάθε *approximate distributed counter* της μορφής $A_i\left(x_i, x_i^{par}\right)$ ισούται με $O(\frac{\sqrt{k}}{v_i} \cdot \log \frac{1}{\delta} \cdot \log \mathrm{m})$. Δεδομένου τώρα ότι τα *counters* $A_i\left(x_i, x_i^{par}\right)$ για τον κόμβο X_i καθορίζονται τόσο από το J_i όσο και από το K_i , ισχύει ότι για κάθε X_i έχουμε $J_i \cdot K_i$ *counters*. Επομένως το συνολικό *communication cost* για όλα τα *counters* $A_i\left(x_i, x_i^{par}\right)$ από κάθε κόμβο X_i , ισούται με:

$$M_1 = \sum_{i=1}^{n} \frac{J_i \cdot K_i \cdot \sqrt{k}}{v_i} \cdot \log \frac{1}{\delta} \cdot \log \mathrm{m}$$

Κάνοντας αντικατάσταση το v_i , μπορούμε να εκφράσουμε το M_1 με τον ακόλουθο τρόπο:

$$M_1 = \left(\sum_{i=1}^{n} (J_i \cdot K_i)^{2/3}\right)^{3/2} \cdot \frac{\sqrt{k}}{\varepsilon} \cdot \log \frac{1}{\delta} \cdot \log \mathrm{m}$$

Αντίστοιχα για κάθε κόμβο X_i έχουμε K_i *counters* της μορφής $A_i\left(x_i^{par}\right)$ και το *communication cost* που απαιτείται για τον καθένα ισούται με $O(\frac{\sqrt{k}}{\mu_i} \cdot \log \frac{1}{\delta} \cdot \log \mathrm{m})$, επομένως το συνολικό *communication cost* για όλα τους κόμβους ισούται με (κάνοντας και αντικατάσταση του μ_i):

$$M_2 = \left(\sum_{i=1}^{n} K_i^{2/3}\right)^{3/2} \cdot \frac{\sqrt{k}}{\varepsilon} \cdot \log \frac{1}{\delta} \cdot \log \mathrm{m}$$

Επομένως προκύπτει ότι το συνολικό κόστος ισούται με $M_1 + M_2 = O(\Gamma \cdot \frac{\sqrt{k}}{\varepsilon} \cdot \log \frac{1}{\delta} \cdot \log \mathrm{m})$. Επιπλέον και σε αυτήν την περίπτωση καταφέραμε να έχουμε λογαριθμική εξάρτηση στο αριθμό των *training instances*.

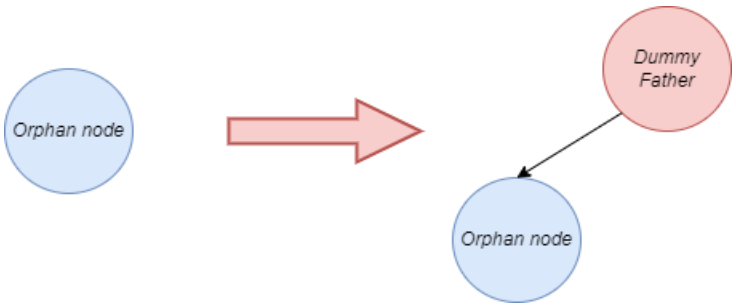
	Approximation Factor	*Communication Cost*
BASELINE $A_i(x_i, x_i^{par}), A_i(x_i^{par})$	$\frac{\varepsilon}{3n}$	$O\left(\frac{n^2 \cdot J^{d+1}\sqrt{k}}{\varepsilon} \cdot \log\frac{1}{\delta} \cdot \log \mathrm{m}\right)$
UNIFORM $A_i(x_i, x_i^{par}), A_i(x_i^{par})$	$\frac{\varepsilon}{16\sqrt{n}}$	$O\left(\frac{n^{3/2} \cdot J^{d+1}\sqrt{k}}{\varepsilon} \cdot \log\frac{1}{\delta} \cdot \log \mathrm{m}\right)$
NON_UNIFORM $A_i(x_i, x_i^{par})$	$v_i = \frac{(J_i K_i)^{\frac{1}{3}} \cdot \varepsilon}{16a}$ $\alpha = \left(\sum_{i=1}^{n} (J_i K_i)^{\frac{2}{3}}\right)^{1/2}$	$O(\Gamma \cdot \frac{\sqrt{k}}{\varepsilon} \cdot \log\frac{1}{\delta} \cdot \log \mathrm{m})$ $\Gamma = \left(\sum_{i=1}^{n} (J_i K_i)^{\frac{2}{3}}\right)^{3/2} + \left(\sum_{i=1}^{n} (K_i)^{\frac{2}{3}}\right)^{3/2}$
$A_i(x_i^{par})$	$\mu_i = \frac{(K_i)^{\frac{1}{3}} \cdot \varepsilon}{16\beta}$ $\beta = \left(\sum_{i=1}^{n} (K_i)^{\frac{2}{3}}\right)^{\frac{1}{2}}$	

Πίνακας 3: Approximation factor και Communication Cost για όλους τους αλγορίθμους

4.1.2.1 Dummy Father

Ένα πρόβλημα το οποίο παρατηρήσαμε στο αλγόριθμο *NON_UNIFORM* όπου το κάθε *approximation factor ε* είναι ανάλογο τόσο του J_i όσο και του K_i , είναι όταν κάποιο από τους κόμβους είναι *orphan* δηλαδή ισχύει ότι το $Par(X_i) = \emptyset$. Σε αυτήν την περίπτωση με βάση τον αλγόριθμο *NON_UNIFORM* το *approximation factor ε* για κάθε τέτοιο κόμβο θα ήταν μηδέν αφού και το K_i του κόμβου θα ήταν μηδέν. Αυτό έχει ως συνέπεια ότι για κάθε τέτοιο κόμβο θα έπρεπε να διατηρήσουμε το *EXACT counter* για κάθε $x_i \in Val(X_i)$. Επομένως στην περίπτωση που το δίκτυο είχε πολλούς *orphan* κόμβους αυτό θα είχε ως αποτέλεσμα το *communication cost* από τέτοιους κόμβους να ήταν αρκετά μεγάλο.

Για να αποφύγουμε αυτό το πρόβλημα, προτείναμε για κάθε *orphan* κόμβο X_i την εισαγωγή ενός "*εικονικού*" *parent* κόμβου(*Dummy father -- Εικόνα 13*). Συγκεκριμένα για κάθε τέτοιο κόμβο θεωρούμε ότι το $K_i = 1$. Επομένως με αυτό τον τρόπο επιτυγχάνουμε να αποφύγουμε να διατηρούμε τον *EXACT counter* για κάθε *counter* της μορφής $A_i(x_i, x_i^{par})$ που αντιστοιχεί στον *orphan* κόμβο X_i.



Εικόνα 13: Dummy Father

Με αυτόν αυτό τον τρόπο επιπλέον επιτυγχάνουμε να αυξήσουμε και την τιμή του $\alpha = \left(\sum_{i=1}^{n} (J_i K_i)^{\frac{2}{3}} \right)^{1/2}$, το όποιο οδηγεί και σε "*ελαφρώς*" μεγαλύτερα *approximation factor* για τους υπόλοιπους κόμβους , το όποιο με την σειρά του οδηγεί και σε "*ελαφρώς*" μικρότερο *communication cost*. Το βασικό πλεονέκτημα και πάλι είναι ότι μπορούμε να αποφύγουμε τους *EXACT counters* και κατά κύριο λόγο είναι αυτό το οποίο οδηγεί και στο ποσοστό μείωσης που επιτυγχάνουμε.

Όσον αφορά το πειραματικό κομμάτι, τον αλγόριθμο του *NON_UNIFORM* το χρησιμοποιούμε σε συνδυασμό με την μέθοδο του *Dummy Father*. Με αυτόν τον τρόπο καταφέραμε να μειώσουμε το *communication cost* σε ένα ποσοστό που κυμαίνεται από $0 - 50\%$ σε σχέση τον αλγόριθμο *NON_UNIFORM*(*Βλ. Κεφάλαιο 6.3.5*). Τέλος να επισημάνουμε ότι αυτό το ποσοστό μείωσης που επιτυγχάνουμε εξαρτάται σε μεγάλο βαθμό από την τοπολογία του ίδιου του *Bayesian Network*, δηλαδή αν περιέχει *orphan* κόμβους ή όχι και για αυτό τον λόγο παρατηρούμε ότι το ποσοστό μείωσης μεταβάλλεται μεταξύ των διαφόρων *Bayesian Networks* που χρησιμοποιήσαμε.

4.1.2.2 Σύγκριση μεταξύ των αλγορίθμων BASELINE, UNIFORM, NON_UNIFORM

Αν θέλουμε να συγκρίνουμε τους προηγούμενους αλγορίθμους στην γενική περίπτωση ισχύει ότι *BASELINE < UNIFORM < NON_UNIFORM*. Τώρα αν επικεντρωθούμε στους δυο αλγορίθμους δηλαδή στον *BASELINE* και στον *UNIFORM* , αυτό το οποίο ισχύει είναι ότι ο αλγόριθμος του *UNIFORM* είναι καλύτερος από το αλγόριθμο *BASELINE* αφού ισχύει ότι $\frac{\varepsilon}{16\sqrt{n}} > \frac{\varepsilon}{3n}$, επομένως το *communication cost* από κάθε *approximate distributed counter* του αλγορίθμου *UNIFORM* είναι μικρότερο. Υπάρχουν όμως περιπτώσεις που ισχύει το ανάποδο. Παρατηρήσαμε ότι όταν έχουμε μικρά *Bayesian Networks* δηλαδή όταν έχουμε *Bayesian Networks* όπου ο αριθμός των κόμβων n είναι μικρότερος του 29 τότε ισχύει το ανάποδο. Σε αυτήν την περίπτωση ισχύει το ανάποδο γιατί ισχύει ότι $\frac{\varepsilon}{16\sqrt{n}} < \frac{\varepsilon}{3n}$ για $n \leq 29$, επομένως το *communication cost* από κάθε *approximate distributed counter* του αλγορίθμου *BASELINE* είναι μικρότερο, άρα και το συνολικό κόστος. Τέλος τόσο ο αλγόριθμος *BASELINE* όσο και ο αλγόριθμος *UNIFORM* έχουν την ίδια εξάρτηση πάνω στους παραμέτρους k, ε, δ και m, εκεί που διαφέρουν είναι η παράμετρος n, συγκεκριμένα ο αλγόριθμος *UNIFORM* είναι κατά $\sqrt{n}$ καλύτερος.

Όσον αφορά τους αλγορίθμους *UNIFORM* και *NON_UNIFORM*, αυτό το οποίο ισχύει είναι ότι ο αλγόριθμος *NON_UNIFORM* είναι τουλάχιστον όσο καλός είναι και ο αλγόριθμος *UNIFORM* δεδομένου ότι ο αλγόριθμος *NON_UNIFORM* χρησιμοποιεί περισσότερη πληροφορία. Γενικά τόσο ο αλγόριθμος *UNIFORM* όσο και ο αλγόριθμος *NON_UNIFORM* έχουν την ίδια εξάρτηση πάνω στους παραμέτρους k, ε, δ και m, εκεί που διαφέρουν είναι στην εξάρτηση από τις παραμέτρους J_i, K_i αντίστοιχα. Στην περίπτωση που αυτές οι παράμετροι διαφέρουν μεταξύ των διαθέσιμων κόμβων τότε ο αλγόριθμος του *NON_UNIFORM* είναι καλύτερος ενώ σε αντίθεση περίπτωση όπου τόσο τα J_i όσο και τα K_i είναι κοντά μεταξύ των διαθέσιμων κόμβων τότε οι δυο αλγόριθμοι βρίσκονται αρκετά κοντά, με τον αλγόριθμο του *UNIFORM* να εμφανίζεται λίγο καλύτερος, το οποίο επιβεβαιώθηκε και στο πειραματικό κομμάτι γιατί όλα τα *Bayesian Networks* που χρησιμοποιήσαμε δεν εμφανίζουν σημαντικές διαφορές στις παραμέτρους J_i, K_i μεταξύ των διαθέσιμων κόμβων.

4.1.3 Communication cost από Naïve Bayes Classifier

Τέλος παραθέτουμε το *communication cost* για κάθε ένα από τους τρεις αλγορίθμους, στην περίπτωση των *Naïve Bayes Classifier*.

Δεδομένου ενός *Naïve Bayes Classifier* $\mathcal{G}(\mathcal{X}, \mathcal{E})$, με το σύνολο των κόμβων να ισούται με $\mathcal{X} = \{\mathrm{X}_1 , \dots , X_n, C\}$. Σε αυτήν την περίπτωση ισχύει ότι το K_i για κάθε κόμβο X_i ισούται με το *cardinality* του *συνόλου* από το *class variable* C, δηλαδή ισχύει ότι το $K_i = J_c$ για κάθε $X_i \in \mathcal{X} - \{C\}$. Επιπλέον ισχύει ότι ο μέγιστος αριθμός από *parents* κόμβους που έχει κάποιος(ή κάποιοι) κόμβος στο δίκτυο ισούται με 1 , δηλαδή ισχύει ότι $d = 1$.

Με βάση τον *Πίνακα 3* , μπορούνε να δούμε ότι το συνολικό *communication cost* από τον *BASELINE* αλγόριθμο στην περίπτωση *Naïve Bayes Classifier* ισούται με:

$$BASELINE\ με \text{ Naïve Bayes Classifier: } O\left(\frac{n^2 \cdot J^2\sqrt{k}}{\varepsilon} \cdot \log\frac{1}{\delta} \cdot \log \mathrm{m}\right)$$

Ενώ για τον *UNIFORM* αλγόριθμο το συνολικό *communication cost* ισούται με:

$$UNIFORM\ με \text{ Naïve Bayes Classifier: } O\left(\frac{n^{3/2} \cdot J^2\sqrt{k}}{\varepsilon} \cdot \log\frac{1}{\delta} \cdot \log \mathrm{m}\right)$$

Όσον αφορά το *NON_UNIFORM* αλγόριθμο για κάθε *approximate distributed counter* $A_i\left(x_i, x_i^{par}\right)$ το *approximation factor* ορίζεται με τον ακόλουθο τρόπο:

$$v_i = \frac{(J_i)^{\frac{1}{3}} \cdot \varepsilon}{16a} \text{ , όπου } \alpha = \left(\sum_{i=1}^{n} (J_i)^{\frac{2}{3}}\right)^{1/2}$$

Ενώ για κάθε *approximate distributed counter* $A_i\left(x_i^{par}\right)$ το *approximation factor* ορίζεται με τον ακόλουθο τρόπο:

$$\mu_i = \frac{(K_i)^{\frac{1}{3}} \cdot \varepsilon}{16\beta} = \frac{\varepsilon}{16\sqrt{n}} \text{ , όπου } \beta = \sqrt{n \cdot (J_c)^{2/3}}$$

Επομένως το συνολικό *communication cost* που απαιτείται για τον *NON_UNIFORM* αλγόριθμο στην περίπτωση *Naïve Bayes Classifier* δεδομένου *m training instances* ισούται με:

$$NON\_UNIFORM\ με \text{ Naïve Bayes Classifier}$$

$$O\left(J_c \cdot \left(\sum_{i=1}^{n} (J_i)^{\frac{2}{3}}\right)^{3/2} \cdot \frac{\sqrt{k}}{\varepsilon} \cdot \log\frac{1}{\delta} \cdot \log \mathrm{m}\right)$$

Τέλος μπορούμε να δούμε ότι και στους τρεις αλγορίθμους και στην περίπτωση των *Naïve Bayes Classifier* επιτυγχάνουμε να έχουμε λογαριθμική εξάρτηση στο αριθμό των *training instances*.

4.2 Μια δεύτερη προσέγγιση

Η δεύτερη προσέγγιση που προτείναμε για την επίλυση του προβλήματος , δηλαδή να έχουμε ένα *(ε,δ)-approximation* από το *MLE* του *joint probability distribution* που αντιστοιχεί στο *Bayesian Network Structure* $\mathcal{G}(\mathcal{X}, \mathcal{E})$ πάνω στο *Distributed Continuous Model*, ήταν πλέον να μην βλέπουμε τα *approximate distributed counters* που πρέπει να διατηρήσουμε ως ξεχωριστές οντότητες(*individually*) αλλά να τα βλέπουμε "*συλλογικά*" , δηλαδή πλέον να τα διαχειριζόμαστε ως *vectors* και συγκεκριμένα ως *frequency vectors*. Με αυτόν τον τρόπο μπορούμε να εφαρμόσουμε πιο γενικές τεχνικές για την επίλυση του προβλήματος οι οποίες όπως θα δούμε και στην συνέχεια οδηγούν σε αρκετά καλύτερα αποτελέσματα. Συγκεκριμένα αυτό το οποίο προτείναμε είναι η εφαρμογή του *Functional Geometric Monitoring* [21] σε συνδυασμό με τους αλγορίθμους *BASELINE* και *UNIFORM*, οι οποίοι χρησιμοποιούνται αποκλειστικά για τον καθορισμό της κατάλληλης τιμής του *approximation factor ε* στα διαθέσιμα *counters*.

Τέλος να επισημάνουμε ότι δεν χρησιμοποιούμε τον αλγόριθμο *NON_UNIFORM* γιατί όπως αναλύσαμε και προηγουμένως το *approximation factor ε* που ορίζεται από τον αλγόριθμο *NON_UNIFORM* εξειδικεύεται για κάθε *approximate counter* ενώ εμείς πλέον χρησιμοποιούμε *frequency vectors*. Επομένως θέλουμε αλγορίθμους που να μπορούν να εφαρμοστούν καθολικά, δηλαδή για όλα τα *approximate counters* και αυτός είναι ο λόγος που επιλέξαμε τους αλγορίθμους *BASELINE* και *UNIFORM*.

Όπως είπαμε και προηγουμένως, μια τέτοιο μέθοδο αποτελεί το *Functional Geometric Monitoring(FGM)*. Η συγκεκριμένη μέθοδος επιλέχθηκε γιατί έχει σημαντικά οφέλη τόσο στο *communication cost* όσο στο *scalability* σε σχέση με οποιαδήποτε άλλη μέθοδο αλλά κυρίως με μια προηγούμενη χρονικά μέθοδο, του *Geometric Monitoring(GM)* [22], το *FGM* αποτελεί προπομπό του *GM*.

Αρχικά το *FGM* αποτελεί μια καθολικά εφαρμόσιμη μέθοδο δηλαδή παρέχει μια μέθοδο που είναι ανεξάρτητη από το πρόβλημα το οποίο κάνουμε *monitoring* και αυτό γίνεται μέσω μιας *problem-specific οικογένειας συναρτήσεων* που ονομάζονται *safe functions* και αναλύουμε στην συνέχεια. Αυτό έχει ως αποτέλεσμα η μέθοδο του *FGM* να μπορεί να ενσωματωθεί εύκολα σε οποιοδήποτε *stream processing framework* , όπως για παράδειγμα το *Apache Flink* [23] και *Apache Spark* [24], παρέχοντας ένα "γενικό" εργαλείο για το *monitoring* πολλών προβλημάτων.

Επιπλέον η μέθοδος του *FGM* έχει αποδειχθεί ότι μπορεί να προσαρμοστεί σε "δυσμενείς" συνθήκες που αφορούν το εκάστοτε *monitoring* πρόβλημα, όπως για παράδειγμα στην περίπτωση που έχουμε παρά πολύ *tight bounds*(το οποίο αποτελεί και η δικιά μας περίπτωση). Επιπλέον η μέθοδος του *FGM* μπορεί να προσαρμοστεί εύκολα ακόμα και στην περίπτωση παρουσίας *skew* στο *distribution* των δεδομένων μεταξύ των *sites*.

4.2.1 Functional Geometric Monitoring (FGM)

Έστω ότι έχουμε στην διάθεση μας k sites. Πλέον κάθε *site* S_i με $i \in \{1, \dots, k\}$, στην διάθεση του έχει ένα *frequency vector* από *counters* $\boldsymbol{S_i}(t)$(*local state vector*) τα οποίο ενημερώνει με κατάλληλο τρόπο(*Βλ. Κεφάλαιο 3.2*) καθώς δέχεται δεδομένα. Επιπλέον ισχύει ότι $\boldsymbol{S}(t) = \sum_{i=1}^{k} \boldsymbol{S_i}(t)$, δηλαδή το $\boldsymbol{S}(t)$ αντιστοιχεί στο *global state* vector.Υποθέτουμε δηλαδή ότι το *global state* αντιστοιχεί στο άθροισμα των επιμέρους *local states*. Εστιάζουμε στην περίπτωση του *continuous query (Q)*, δηλαδή για κάθε χρονική στιγμή θέλουμε ο *Coordinator* να έχει ένα *ε-approximation* από την τιμή του *query* που αντιστοιχεί στο το *global state vector* $S(t)$, το οποίο μπορούμε να το εκφράσουμε με τον ακόλουθο τρόπο:

$$Q\big(\boldsymbol{S}(t)\big) \in (1 \pm \varepsilon) Q(\boldsymbol{E}(t))$$

Με το $\boldsymbol{E}(t) = \sum_{i=1}^{k} \boldsymbol{E_i}(t)$, όπου το $\boldsymbol{E_i}$ αντιστοιχεί στο *estimated vector* που έχει ο *Coordinator* για κάθε *site* S_i. Για να ισχύει η προηγούμενη σχέση θα πρέπει κάθε *site* S_i με περιοδικό τρόπο συγκεκριμένα σε κάθε *round*, να στέλνει ένα μήνυμα στο *Coordinator* το οποίο να τον ενημερώνει για το πόσο έχει μεταβληθεί το *local state vector* $\boldsymbol{S_i}(t)$ του από το *local data stream* που έχει λάβει. Συγκεκριμένα ο *Coordinator* έχει ένα *estimator vector* $\boldsymbol{E_i}$ για κάθε *site* S_i, επομένως κάθε φορά που το *site* στέλνει ένα μήνυμα στο *Coordinator* το οποίο τον ενημερώνει για το πόσο έχει μεταβληθεί το *local state vector* του , δηλαδή το *site* S_i στέλνει το *drift vector* του $\boldsymbol{X_i}(t) = \boldsymbol{S_i}(t) - \boldsymbol{E_i}$, τότε ο *Coordinator* το μόνο που χρειάζεται να κάνει είναι να ενημερώσει το $\boldsymbol{E_i}$, προσθέτοντας το $\boldsymbol{X_i}$ (*vector addition*). Παραπάνω λεπτομέρειες παρατίθενται αναλυτικά στο [21].

Το *Functional Geometric Monitoring*(FGM) για να εξασφαλίζει το *safety* ολόκληρου του συστήματος, δηλαδή η τιμή από το *monitoring query* να βρίσκεται στα επιθυμητά όρια χρησιμοποιεί ένα *safe function* φ , το οποίο αντιστοιχεί σε ένα *real function* και καθορίζεται αναλόγως με το *monitoring* πρόβλημα. Συγκεκριμένα το *safe function* φ διοχετεύεται σε κάθε *site* S_i το οποίο το μόνο που χρειάζεται να κάνει είναι να υπολογίζει το $\varphi(\boldsymbol{X_i})$ καθώς μεταβάλλεται το *drift vector* $\boldsymbol{X_i}$. Αυτό είναι αρκετό γιατί μπορεί να αποδειχθεί ότι αν το άθροισμα $\psi = \sum_{i=1}^{k} \varphi(\boldsymbol{X_i}) > 0$, τότε εξασφαλίζουμε και του *safety* ολόκληρου του συστήματος. Παραπάνω λεπτομέρειες όσον αφορά το *safe zone composition* όσο και "*quality*" κριτήρια για τα *safe zones* , παρατίθενται αναλυτικά στο [25].

Συγκεκριμένα εστιάζουμε στον *monitoring* των *frequency moments* F_p , αλλά όπως είπαμε και προηγουμένως και προκύπτει και από την ανάλυση του αλγορίθμου *MLE*, μας ενδιαφέρει να μπορούμε να διατηρήσουμε *counters* της μορφής $A_i\big(x_i, x_i^{par}\big)$, $A_i\big(x_i^{par}\big)$ δεδομένου ενός *Bayesian Network Structure* $\mathcal{G}(\mathcal{X}, \mathcal{E})$ όπου το σύνολο των κόμβων ισούται με $\mathcal{X} = \{X_1, \dots, X_n\}$. Επομένως εστιάζουμε στην περίπτωση *monitoring* του F_1 *frequency moment* από το *vector* $\boldsymbol{x} = [x_1, \dots, x_n]$, όπου το F_1 *moment* ισούται με το l^1 $norm$ δηλαδή ισχύει ότι:

$$\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i|$$

Στην περίπτωση μας το κάθε x_i από το *frequency vector* $\boldsymbol{x}$ αντιστοιχεί σε κάποιο από τα *counters* της μορφής $A_i\big(x_i, x_i^{par}\big)$, $A_i\big(x_i^{par}\big)$. Επομένως το $\boldsymbol{x}$ περιέχει όλα τα *counters* που αντιστοιχούν για κάθε κόμβο X_i του δικτύου $\mathcal{G}$. Συγκεκριμένα για κάθε *counter* $A_i\big(x_i, x_i^{par}\big)$

και $A_i(x_i^{par})$ κρατάμε το *frequency* του δηλαδή το πόσες φορές εμφανίζεται στο *data stream*.

Επομένως εστιάζοντας στο *monitoring* του F_1 *frequency moment*, τα *safe functions* που θα χρησιμοποιήσουμε θα είναι της μορφής:

$$\|\boldsymbol{x} + \boldsymbol{E}\|_1 - T$$

Όπως είπαμε και προηγουμένως εστιάζουμε στην περίπτωση του *continuous query* επομένως το *threshold* T σε κάθε *round* θα μεταβάλλεται και θα ισοδυναμεί με $(1+\varepsilon)\|\boldsymbol{E}\|_1$, άρα το *safe function* θα έχει την ακόλουθη μορφή:

$$\|\boldsymbol{x} + \boldsymbol{E}\|_1 - (1+\varepsilon)\|\boldsymbol{E}\|_1$$

Στόχος μας είναι να έχουμε ένα *ε-approximation* από το *MLE* που αντιστοιχεί στο *Bayesian Network* $\mathcal{G}(\mathcal{X}, \mathcal{E})$. Αυτό πρώτον προϋποθέτει να μοιράσουμε το διαθέσιμο *error budget* στα *counters* με τέτοιο τρόπο ώστε να εξασφαλίσουμε ότι για κάθε χρονική στιγμή έχουμε ένα *ε-approximation* από το *MLE*, δεδομένου ότι πλέον τα *counters* τα βλέπουμε ως *frequency vectors* και το *safe function* έχει την παραπάνω μορφή είναι προφανές ότι το *approximation factor ε* θα πρέπει να είναι το ίδιο για όλους τους *counters*. Αυτό συνεπάγεται αυτομάτως ότι πλέον μπορούμε να χρησιμοποιήσουμε μόνο τους αλγορίθμους *BASELINE* και *UNIFORM*. Δεύτερον δεδομένου ότι μοιράζαμε το *error budget* με κατάλληλο τρόπο, προϋποθέτει ανά πάσα χρονική στιγμή να έχουμε ένα *ε-approximation* για κάθε *counter* $A_i(x_i, x_i^{par})$ και $A_i(x_i^{par})$ από κάθε X_i.

Χρησιμοποιώντας την παραπάνω μορφή μπορούμε να παρατηρήσουμε ότι το μέγιστο *error* που έχουμε σε κάθε *round* είναι ανάλογο του $\varepsilon\|\boldsymbol{E}\|_1$, το οποίο όπως μπορούμε να δούμε δεν είναι αρκετό για να μας εξασφαλίσει ότι θα έχουμε ένα *ε-approximation* για κάθε *counter* $A_i(x_i, x_i^{par})$, $A_i(x_i^{par})$. Αυτό οφείλεται στο γεγονός ότι *error* που έχουμε είναι ανάλογο του αθροίσματος των τιμών των επιμέρους *counters* και επομένως δεν είναι αντιπροσωπευτικό για το κάθε ένα *counter*. Για παράδειγμα αν έχουμε ένα *frequency vector* το οποίο περιέχει έναν "*μικρό*" *counter* ενώ οι υπόλοιποι αντιστοιχούν σε "*μεγάλους*" *counter* μιλώντας αριθμητικά, τότε σε αυτήν την περίπτωση το *error* που θα διαθέσουμε στο εκάστοτε *round* θα επηρεαστεί και από τις τιμές των "μεγάλων" *counters*. Αυτό θα έχει ως συνέπεια στην συνέχεια να μην μπορούμε να ανιχνεύσουμε μεταβολές που αντιστοιχούν στον "*μικρό*" *counter* γιατί πολύ απλά οι μεταβολές του θα είναι πολύ μικρότερες του *error*.

Επομένως προτείναμε να χρησιμοποιούμε και πάλι ένα *safe function* της μορφής $\|\boldsymbol{x} + \boldsymbol{E}\|_1 - T$, μόνο που το *threshold* T θα ορίζεται με τον ακόλουθο τρόπο:

$$T = \|\boldsymbol{E}\|_1 + minViolatedValue$$

$$\text{όπου } minViolatedValue = \varepsilon \cdot \min_{\mathrm{i}} \boldsymbol{E}$$

Δηλαδή το *minViolatedValue* καθαρίζεται από την τιμή του ε και από την ελάχιστη τιμή του *counter* που υπάρχει στο E. Με αυτόν τον τρόπο μπορούμε να εξασφαλίσουμε ότι θα έχουμε ένα *ε-approximation* για κάθε *counter* $A_i(x_i, x_i^{par})$ και $A_i(x_i^{par})$ για κάθε X_i , γιατί το *error* σε κάθε *round* πλέον υπολογίζεται με βάση την μικρότερη τιμή μεταξύ των *counters*, δηλαδή αντιστοιχεί στο ελάχιστο δυνατό που μπορούμε να έχουμε. Επομένως εξασφαλίζοντας ότι δεν θα έχουμε κάποιο *violation* για τον μικρότερο αριθμητικά *counter* αυτομάτως

εξασφαλίζουμε ότι και υπόλοιποι *counters* δεν θα έχουν κάποιο *violation* γιατί για να έχουν θα πρέπει να μεταβληθούν περισσότερο από αυτό το οποίο έχουμε ορίσει. Βέβαια αυτό θα οδηγήσει σε περισσότερα *rounds* (με μερικά να είναι και αχρείαστα), αλλά το γεγονός ότι βλέπουμε "συνολικά" τους *counters* και σε κάθε μήνυμα ο *Coordinator* λαμβάνει πληροφορία από όλα τα διαθέσιμα *counters* συμβάλλει στο να μειωθεί το *communication cost* αρκετά.

Συνοψίζοντας μπορούμε να πούμε ότι παρόλο που το *error* ορίζεται με βάση την μικρότερη τιμή του *counter* (το οποίο θα είναι και αρκετά μικρό), το πλεονέκτημα του *FGM* οφείλεται στο γεγονός ότι ο *Coordinator* για κάθε *update* μήνυμα που λαμβάνει από το *site* S_i ταυτόχρονα λαμβάνει και πληροφορία για όλα τα *counters* , επομένως βλέποντας "συνολικά" τα *counters* αποδίδει καλύτερα στην πράξη το οποίο επιβεβαιώθηκε και στο πειραματικό κομμάτι. Συγκεκριμένα όσον αφορά το *communication cost* πετύχαμε μια βελτιστοποίηση της τάξης του *10x* σε σχέση με την πρώτη προσέγγιση δηλαδή το να έχουμε μια συλλογή από *distributed counters* όπου το καθένα το βλέπουμε ξεχωριστά, ενώ πετύχαμε και μια βελτιστοποίηση της τάξης του *100x* σε σχέση με το *EXACTMLE*. Τέλος το *FGM* αποτελεί μια *scalable* και *high-throughput* προσέγγιση για το πρόβλημα το οποίο επιβεβαιώθηκε και στο πειραματικό κομμάτι.

Rebalancing

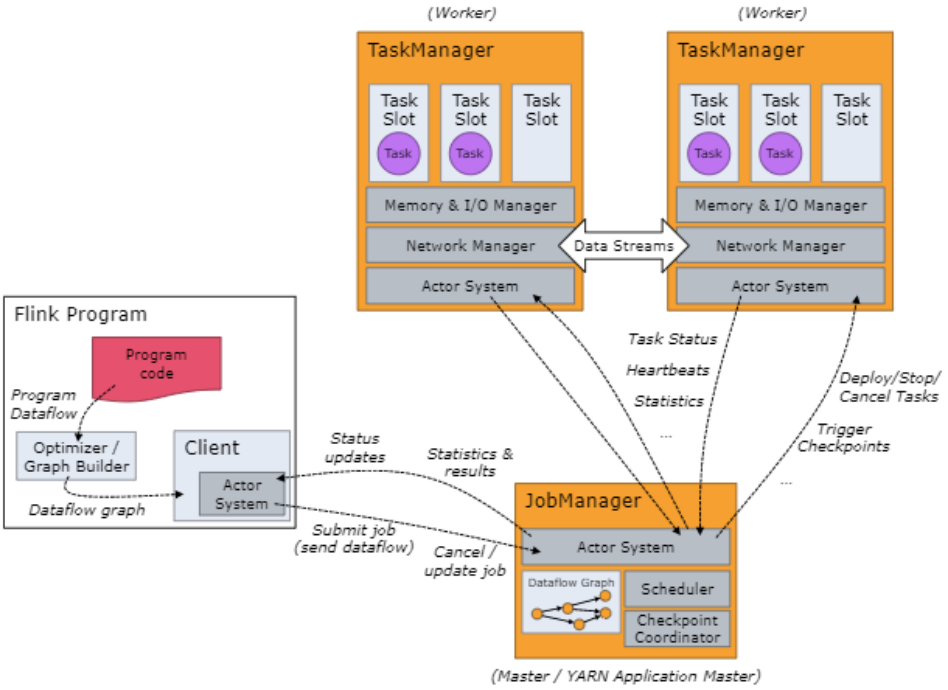
Επιπλέον υλοποιήσαμε και την μέθοδο του *rebalancing* που στοχεύει στο να κάνει την διάρκεια του κάθε *round* μεγαλύτερη, το οποίο είναι επιθυμητό γιατί πρώτον τόσο καλύτερα τα *local streams* συνοψίζονται στα *local state vectors* και δεύτερον μπορούμε να αποφύγουμε περισσότερες φορές το *upstream communication cost* του *estimated vector* $\boldsymbol{E}$ στην αρχή του κάθε *round*. Η βασική ιδέα προκύπτει από το γεγονός ότι η συνθήκη $\psi = \sum_{i=1}^{k} \varphi(\boldsymbol{X_i}) > 0$ δεν υπονοεί απαραίτητα ότι και το *global state* $\boldsymbol{S}(t)$ έχει μετακινηθεί αρκετά μακριά από το $\boldsymbol{E}$, επομένως το *safe function* μπορεί ακόμα να είναι χρήσιμο με συνέπεια σε ορισμένες περιπτώσεις να μπορέσουμε να αποφύγουμε το *upstream communication cost* μεταφοράς ενός καινούργιου *safe function* στα *sites*. Αντί αυτού ο *Coordinator* κάνει *broadcast* ένα *scaling factor* λ, το οποίο χρησιμοποιείται από το *sites* ώστε να προσαρμόσουν με κατάλληλο τρόπο τα *drift vectors* τους. Επομένως όταν συμβαίνει επιτυγχάνουμε να μειώσουμε το *communication cost* γιατί χρειάζεται να κάνουμε *broadcast* ενός *scaling factor* ενώ σε αντίθετη περίπτωση θα χρειαζόταν να κάνουμε *broadcast* το *estimated vector* .Τέλος στο πειραματικό κομμάτι το *FGM* χρησιμοποιείται σε συνδυασμό με την μέθοδο του *rebalancing*.

Κεφάλαιο 5:

Σχεδίαση και Υλοποίηση του Συστήματος

5.1 Apache Flink

Το *Apache Flink* [23] αποτελεί μια *open-source* πλατφόρμα που χρησιμοποιείται για *distributed stream data processing*. Συγκεκριμένα πυρήνας του *Flink* αποτελεί το *streaming engine* το οποίο παρέχει μας παρέχει την δυνατότητα να εκτελούμε *stateful* υπολογισμούς πάνω σε *unbounded data streams* ενώ ταυτόχρονα εγγυάται *in-memory* ταχύτητες εκτέλεσης των υπολογισμών. Επιπλέον το *Flink* είναι σχεδιασμένο με τέτοιο τρόπο ώστε να μπορεί να ενσωματωθεί σε οποιοδήποτε *cluster environment* (όπως για παράδειγμα το *Hadoop Yarn,Kubernetes*).

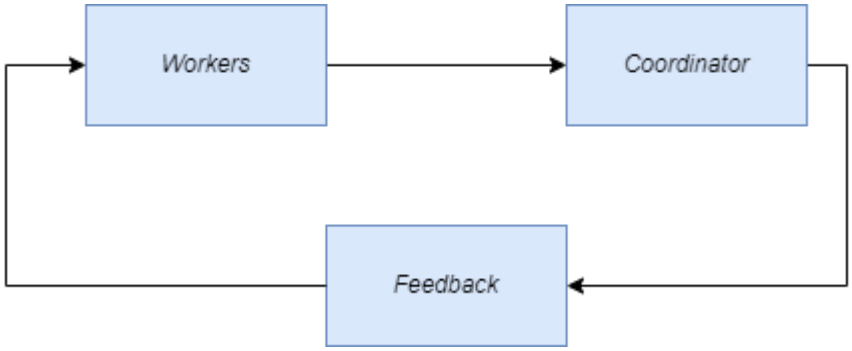


Εικόνα 14: Apache Flink

Το *Apache Flink* αποτελείται από δυο *process*, το πρώτο αφορά το *Job Manager* και το δεύτερο αφορά τους *TaskMangers*(*Εικόνα 14*). Συγκεκριμένα ο *Job Manger* είναι υπεύθυνος για τον συντονισμό ολόκληρης της διαδικασίας εκτέλεσης της εκάστοτε εφαρμογής ενώ ο κάθε *TaskManager* είναι υπεύθυνος για την εκτέλεση των *tasks(operators)* του *dataflow*, όπως επίσης για το *buffering* και το *exchange* των *data streams*. Στην δικιά μας περίπτωση τόσο το κάθε *site*(ή *worker*) όσο και ο *Coordinator* αντιστοιχούν σε κάποιο *task slot* από τους διαθέσιμους *TaskManagers*.

Επιπλέον το *Flink* μας παρέχει πληθώρα επιλογών από *APIs* τα οποία μπορούμε να τα συνδυάσουμε με σκοπό την ανάπτυξη της δικιάς μας εφαρμογής. Ένα από τα πιο σημαντικά *API* είναι αυτό το οποίο περιέχει τα *stateful operators*. Πρόκειται για *operators* τα οποία μας παρέχουν την δυνατότητα να αποθηκεύσουμε τα *states* τους παρέχοντας παράλληλα την

επεξεργασία των δεδομένων. Το *state* μπορούμε να το δούμε ως ένα *snapshot* του *operator* μέχρι την δεδομένη χρονική στιγμή το οποίο θυμάται πληροφορίες σχετικά με τον operator. Στην περίπτωση μας τόσο το κάθε *site* όσο και ο *Coordinator* αντιστοιχούν σε έναν *stateful operator*.



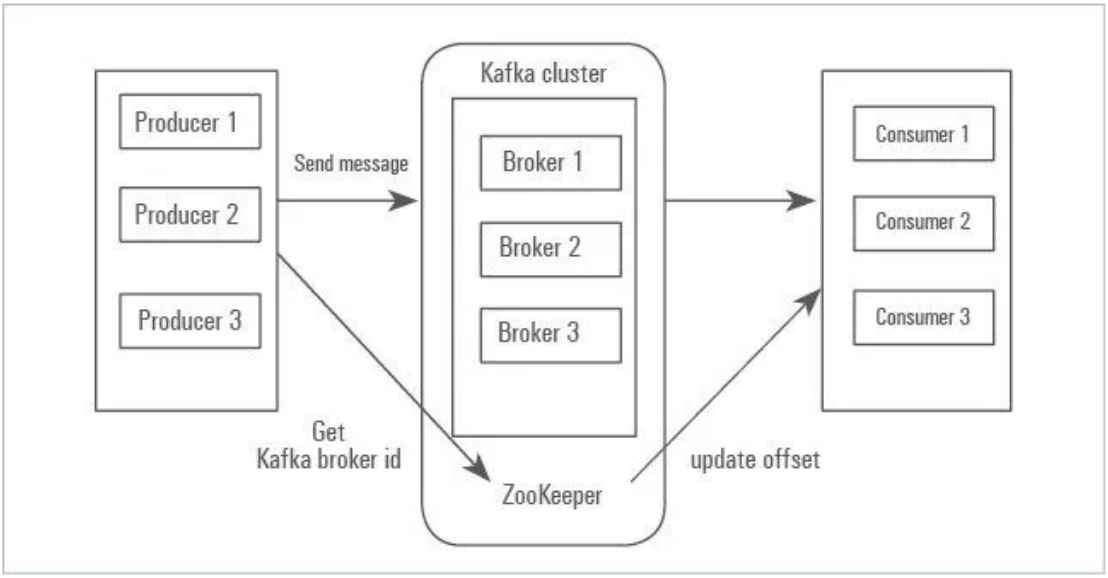
Εικόνα 15: Feedback loop

Όπως αναφέραμε και προηγουμένως μεταξύ των *sites* και του *Coordinator* θα πρέπει να υπάρχει μια ανάδραση(*feedback loop*), δηλαδή η δυνατότητα επικοινωνίας μεταξύ τους. Το *feedback loop* μπορούμε να το δούμε ως *redirecting* της εξόδου του ενός *operator* σε κάποιον προηγούμενο *operator*(*Εικόνα 15*). Επιπλέον το *feedback loop* το συναντάμε σε περιπτώσεις που θέλουμε να έχουμε *continuously maintenance* κάποιου μοντέλου, το οποίο ισοδυναμεί με την περίπτωση μας. Το *Apache Flink* μας παρέχει την δυνατότητα των *Iterative Streams* για να μπορούμε να υλοποιήσουμε το *feedback loop*, δεδομένου όμως ότι είναι ακόμα σε πρώιμο στάδιο (αυτό οφείλεται και στο *version* του *Flink* που χρησιμοποιήσαμε – *Βλ. Κεφάλαιο 6*) αποφασίσαμε να μην το χρησιμοποιήσουμε, αντί αυτού πλέον χρησιμοποιούμε *Kafka topics* για να υλοποιήσουμε το *feedback loop*.

5.2 Apache Kafka

Το *Apache Kafka* [26] πρόκειται για μια *open-source* πλατφόρμα που μας παρέχει την δυνατότητα του *distributed event streaming*. Το *Kafka* αποτελεί προπομπό των *traditional messaging* συστημάτων αφού αποτελεί ένα σύστημα το οποίο μας παρέχει χαρακτηριστικά όπως *high-throughput,high-reliability,scalability* και *durability* τα οποία αποτελούν χαρακτηριστικά τα οποία είναι αναγκαία για την επεξεργασία *distributed streaming data sources*. Στο *Apache Kafka* τα δεδομένα αποθηκεύονται σε *topics*. Τα *topics* με την σειρά τους χωρίζονται σε *partitions* το οποίο είναι σημαντικό τόσο για το *scalability* όσο και για το *reliability* του συστήματος. Επιπλέον τα *partitions* από τα *topics* διαχειρίζονται από τον *Kafka broker*. Στην γενική περίπτωση έχουμε περισσότερα από ένα *Kafka broker* τα οποία μπορούμε να τα δούμε ως *Kafka cluster*. Επιπλέον το *Kafka* μας παρέχει δυο *services*, το πρώτο *service* χρησιμοποιείται για να διαβάσουμε δεδομένα από κάποιο *topic* και το

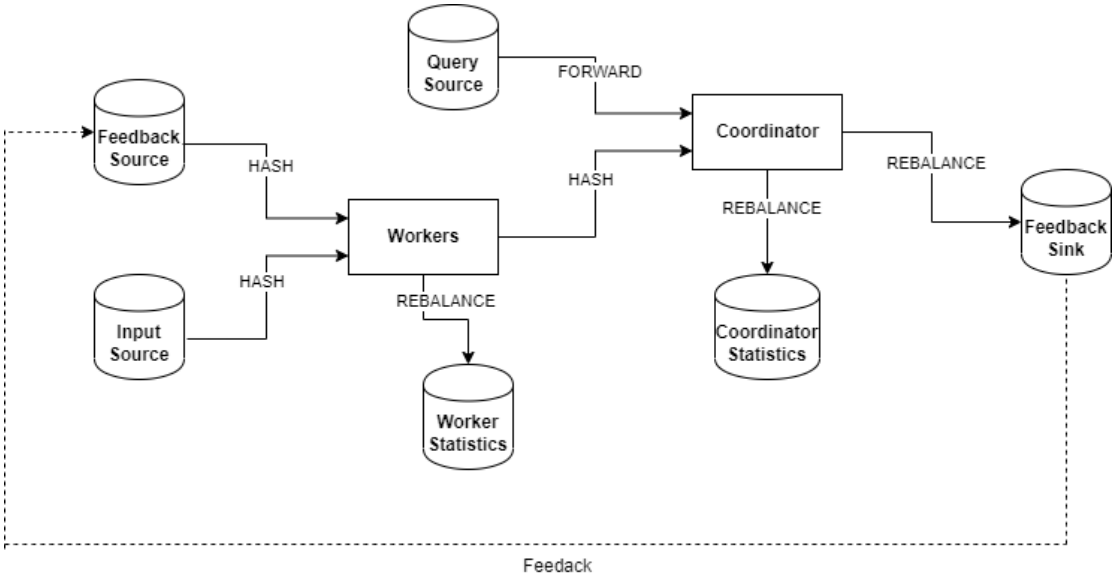
αποκαλούμε *Kafka consumer* ενώ το δεύτερο το χρησιμοποιούμε για να παράγουμε δεδομένα σε κάποιο *topic* και το αποκαλούμε *Kafka producer*.



Εικόνα 16: Apache Kafka

Στην περίπτωση μας το *Kafka* το χρησιμοποιούμε πρώτον για διαβάσουμε τα *data streams* τα οποία διοχετεύονται στα *sites* και δεύτερον για την υλοποίηση του *feedback loop*. Όσον αφορά την υλοποίηση του *feedback loop* χρησιμοποιούμε ένα *Kafka topic* το οποίο συμπεριφέρεται ως *buffer* ανάμεσα σε έναν *producer* και έναν *consumer*. Στην περίπτωση μας ο *producer* αντιστοιχεί στον *Coordinator* ο οποίος παράγει όλα τα "*control"* μηνύματα τα οποία πρέπει να διαβαστούν από το *consumer*, με το *consumer* στην περίπτωση μας να αντιστοιχεί στα *sites*.

5.3 Αρχιτεκτονική του Συστήματος



Εικόνα 17: Αρχιτεκτονική του συστήματος

Για τον σχεδιασμό του συστήματος που υλοποιήσαμε , χρησιμοποιήσαμε τόσο το *Apache Flink* όσο και το *Apache Kafka*. Η *Εικόνα 17* περιλαμβάνει την υλοποίηση που προτείναμε και αναλύουμε στην συνέχεια.

Αρχικά το *Input Source operator* είναι υπεύθυνο να διαβάζει τα δεδομένα από το αντίστοιχο *Kafka topic*(*input topic*) και στην συνέχεια να διαμορφώνει τα δεδομένα ώστε να έχουν το κατάλληλο *format*. Συγκεκριμένα το *format* που έχουν τα δεδομένα, έχει την ακόλουθη μορφή:

$$< value, worker_{id} >$$

Όπου το $value$ αντιστοιχεί στον *input feature vector* του εκάστοτε *Bayesian Network*(ή *Naïve Bayes Classifier*) ενώ το $worker_{id}$ μπορούμε να το δούμε ως *pseudo-key* το οποίο χρησιμοποιείται ώστε τα δεδομένα να κατανέμονται μεταξύ των *Workers*(*horizontal partitioning*). Συγκεκριμένα το *Input Source operator* παράγει τα $worker_{id}$ με τέτοιο τρόπο ώστε τα δεδομένα να κατανέμονται με ομοιόμορφο τρόπο στους *Workers*.

Ο κάθε *Worker* λαμβάνει δυο εισόδους, επομένως μπορούμε να τον δούμε ως ένα *keyed co-process operator*. Η πρώτη είσοδος αφορά τα δεδομένα που πρέπει να επεξεργαστεί και προέρχονται από τον *Input Source Operator* ενώ η δεύτερη είσοδος προέρχεται από το *Feedback Source*. Στην περίπτωση μας το *Feedback Source* περιλαμβάνει όλα τα "*control*" μηνύματα που προέρχονται από τον *Coordinator*. Τέλος το *Feedback Source* και το *Feedback Sink* αφορούν το ίδιο *topic*(*feedback topic*), όπου στην προκειμένη περίπτωση συμπεριφέρεται ως *buffer* ανάμεσα στο *Coordinator* και τους *Workers*, συγκεκριμένα τον *Coordinator* μπορούμε να τον δούμε ως *Kafka Producer* στο *Feedback Sink* ενώ τους *Workers* μπορούμε να τους δούμε ως *Kafka Consumers* από το *Feedback Source*, επομένως με αυτόν τρόπο επιτυγχάνουμε να έχουμε ένα *two-way communication channel* μεταξύ του *Coordinator* με τον κάθε *Worker*(*feedback loop*).

Αντίστοιχα και ο *Coordinator* λαμβάνει δυο εισόδους, επομένως μπορούμε να τον δούμε και αυτόν ως ένα *keyed co-process operator*. Η πρώτη είσοδος αφορά τους *Workers*, δηλαδή περιλαμβάνει όλα τα μηνύματα που στέλνονται από τους *Workers*. Η δεύτερη είσοδος προέρχεται από το *Query Source*. Το *Query Source* στην περίπτωση μας δρα είτε ως *testing source* το οποίο περιέχει *queries* τα οποία χρησιμοποιούνται για την αξιολόγηση της ποιότητας της εκτίμησης του *joint probability distribution* ή ως *query source* όπου μπορεί να χρησιμοποιηθεί από τον χρήστη για να υποβάλει *queries* που αφορούν το *joint probability distribution* του *Bayesian Network*.

Επιπλέον τα μηνύματα τα οποία ανταλλάζονται μεταξύ του *Coordinator* και των *Workers* έχουν το *format* που φαίνεται παρακάτω:

$$< value, worker_{id}, type\_message >$$

Όπου το $value$ αντιστοιχεί στην τιμή που περιέχει το μήνυμα η οποία μπορεί να αναφέρεται στην τιμής από έναν *approximate distributed counter* είτε σε κάποιο *frequency vector* στην περίπτωση του *FGM*. Το $worker_{id}$ στην περίπτωση μηνυμάτων από τους *Workers* προς τον *Coordinator* αναφέρεται στο $cord_{id}$ ενώ στην περίπτωση μηνυμάτων από τον *Coordinator* προς τους *Workers* το $worker_{id}$ αναφέρεται στο *Worker* που πρέπει να μεταφερθεί το μήνυμα. Τέλος $type\_message$ καθορίζει τον τύπο του μηνύματος.

Επιπλέον το κάθε μήνυμα πέρα από τα τρία πεδία ειδικά στην περίπτωση όπου τα *approximate distributed counters* τα βλέπουμε ως ξεχωριστές οντότητες(*individually*), θα

πρέπει το μήνυμα να περιέχει και πληροφορία που αφορά τον *counter* για τον οποίο αναφέρεται τον μήνυμα, βέβαια τον ακόλουθο τρόπο τον χρησιμοποιούμε και ως μηχανισμό ανάκτησης των *counters* κατά την διαδικασία της ενημέρωσης των τιμών τους(*update*). Συγκεκριμένα κάθε *counter* στην περίπτωση του *Bayesian Network* χαρακτηρίζεται μοναδικά από την ακόλουθη πληροφορία:

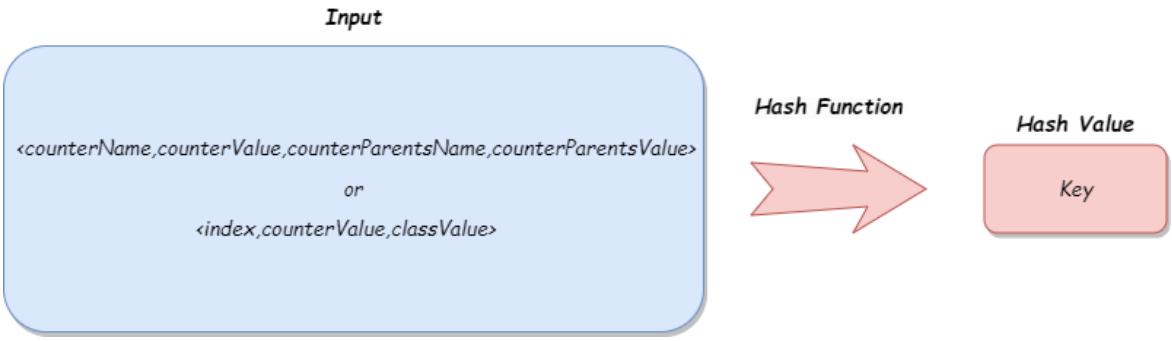
$$< counterName, counterValue, counterParentsName, counterParentsValue >$$

Ενώ στην περίπτωση των *Naïve Bayes Classifier* ο κάθε *counter* χαρακτηρίζεται μοναδικά από την ακόλουθη πληροφορία:

$$< index, counterValue, classValue >$$

Όπου το *index* αντιστοιχεί στη θέση(*position*) που βρίσκεται η τιμή του *counter* στο *input feature vector*.

Επειδή αν το κάθε μήνυμα περιλαμβάνει όλη την προηγούμενη πληροφορία το μέγεθος του μηνύματος θα ήταν αρκετά μεγάλο, πλέον το κάθε *counter* χαρακτηρίζεται από ένα *unique key* το οποίο αντιστοιχεί σε ένα *long* αριθμό. Ο αριθμός αυτός προκύπτει κάνοντας *hashing* την πληροφορία που χαρακτηρίζει μοναδικά τον κάθε *counter* χρησιμοποιώντας το κατάλληλο *hash function*(*Εικόνα 18*).



Εικόνα 18: Hashing

Η διαδικασία του *hashing* προφανές και δεν γίνεται κάθε φορά που θέλουμε να στείλουμε ένα μήνυμα, αλλά γίνεται μόνο κατά την διαδικασία του *Initialization* όπου αντιστοιχούμε το κάθε *counter* με ένα *unique key* και χρησιμοποιείται καθόλη την διάρκεια. Επομένως με αυτόν τον τρόπο πρώτον επιτυγχάνουμε να μειώσουμε το μέγεθος του εκάστοτε μηνύματος αλλά και δεύτερον μπορούμε να το χρησιμοποιήσουμε για να δημιουργήσουμε ένα γρήγορο μηχανισμό ανάκτησης των *counters* χρησιμοποιώντας την κατάλληλη δομή(*Hash Map)*.

Τέλος τόσο ο *Worker* όσο και ο *Coordinator* έχει από μια "πλευρική" έξοδο(*side-output*) που μπορούμε να το δούμε ως ένα "*logging*" μηχανισμό που χρησιμοποιείται τόσο για τους *Workers* όσο και για τον *Coordinator*. Συγκεκριμένα χρησιμοποιείται ώστε να μπορούν να καταγράφονται πληροφορίες οι οποίες είναι χρήσιμές για το σύστημα όπως για παράδειγμα το *throughput* που έχει το σύστημα, το *communication cost* την δεδομένη χρονική στιγμή κ.α.

Παραπάνω λεπτομέρειες όσον αφορά το *project structure* και το *setup του project* παρατίθενται στο *Παράρτημα*. Τέλος όλος ο κώδικας είναι διαθέσιμος στο [27].

Κεφάλαιο 6:

Πειραματική Αξιολόγηση

Η πειραματική ανάλυση μας περιέχει πληθώρα πειραμάτων που αφορά τόσο την πρώτη προσέγγιση(*Βλ. Κεφάλαιο 4.1*) όσο και την προσέγγιση την οποία προτείναμε(*Βλ. Κεφάλαιο 4.2*). Συγκεκριμένα η πειραματική ανάλυση βασίζεται σε τρεις τομείς. Ο πρώτος τομέας επιβεβαιώνει την λειτουργικότητα της προσέγγισης που προτείναμε, ο δεύτερος αναφέρεται σε πειράματα τα οποίο εστιάζουν τόσο στην ικανότητα του συστήματος να κάνει *scale* όσο και στη ικανότητα του συστήματος να διαχειριστεί μεγάλο όγκο δεδομένων ενώ ο τρίτος τομέας παρέχει μια σύγκριση μεταξύ των δυο προσεγγίσεων. Τέλος όλα τα πειράματα αναφέρονται σε *Bayesian Networks*, αντίστοιχα αποτελέσματα ισχύουν και στην περίπτωση των *Naïve Bayes Classifiers*.

Συνοψίζοντας έχουμε πληθώρα πειραμάτων που αφορά την πρώτη προσέγγιση, συγκεκριμένα έχουμε πειράματα τα οποία αναφέρονται και στους δυο *approximate distributed counters* που υλοποιήσαμε, δηλαδή τον *RANDOMIZED* και το *DETERMINISTIC counter* σε συνδυασμό με τους αλγορίθμους *BASELINE*,*UNIFORM* και *NON_UNIFORM*. Όσον αφορά την δεύτερη προσέγγιση, δηλαδή το *FGM* και σε αυτήν την περίπτωση έχουμε πληθώρα πειραμάτων του *FGM* σε συνδυασμό με τους δυο αλγορίθμους *BASELINE* και *UNIFORM*. Τέλος σε όλα τα πειράματα θεωρούμε γνωστό εκ των προτέρων το *structure* από το κάθε *Bayesian Network* και ως στόχο έχουμε την εκμάθηση των παραμέτρων(*learning parameters*).

Testing Setup

Όλα τα πειράματα τα οποία παρουσιάζονται και αναλύονται στην συνέχεια εκτελέστηκαν στο *SoftNet Cluster* του *SoftNet lab* [28]. Το *SoftNet Cluster* αποτελείται από *12 Quad Core Xeon X3323 2.5GHz,8 GB DDR3 RAM*. Το *Apache Flink version* είναι το *1.10.0* και το *Apache Kafka Connector version* είναι το *1.9.3* .

6.1 Datasets

Όλα τα *Bayesian Network* που χρησιμοποιήσαμε πρόκειται για *real-world Bayesian Networks* τα οποία είναι διαθέσιμα στο *repository* [29] και πρόκειται για *Bayesian Networks* τα οποία έχουν χρησιμοποιηθεί και σε προηγούμενες μελέτες. Έχουμε επιλέξει ένα *Bayesian Network* από κάθε κατηγορία. Συγκεκριμένα έχουμε επιλέξει το *ALARM* [30] ως *small size network*(≤ 40 *nodes*), το *HEPAR2* [7] ως *medium size network*(≤ 100 *nodes*) ενώ έχουμε επιλέξει το *LINK* [31] ως *large size network*(≤ 1000 *nodes*). Τέλος όλα τα *Bayesian Networks* που χρησιμοποιήσαμε ανήκουν στην κατηγορία των *Discrete Bayesian Network*.

Όσον αφορά το *ALARM* [30] πρόκειται για ένα δίκτυο το οποίο χρησιμοποιείται σε ασθενής υπό εντατική παρακολούθηση με σκοπό την διάγνωση κάποιας ηπατικής διαταραχής και την διάγνωση κάποιου είδους καρκίνου του οισοφάγου. Το *HEPAR II* [7] πρόκειται για ένα δίκτυο το οποίο χρησιμοποιείται για την διάγνωση διαταραχών του ήπατος και στοχεύει στην μείωση του αριθμού των βιοψιών που πρέπει να γίνουν στον εκάστοτε ασθενή. Το *LINK* [31]

πρόκειται για ένα δίκτυο που χρησιμοποιείται σε αυτό το οποίο αποκαλούμε *linkage analysis*, το πρόβλημα εκτίμησης των σχετικών θέσεων των γονιδίων στο χρωμόσωμα. Τέλος όλα τα *Bayesian Networks* που χρησιμοποιούμε ανήκουν στο χώρο του *biomedical engineering* αλλά όπως είπαμε και προηγουμένως βρίσκουν εφαρμογή σε πολλά *real-world* προβλήματα [4].

Ο *Πίνακας 4* περιέχει μια επισκόπηση των *Bayesian Networks* που χρησιμοποιήσαμε στα πειράματα. Συγκεκριμένα περιέχει τον αριθμό των *nodes* από κάθε *Bayesian Network* , που αντιστοιχεί και στο μέγεθος από το *input feature vector*. Επιπλέον περιέχει τον αριθμό των *edges* που αντιστοιχούν στον συνολικό αριθμό των *conditional dependencies* που υπάρχουν μεταξύ των κόμβων στο κάθε *Bayesian Network*. Επιπροσθέτως περιέχει τον συνολικό αριθμό των παραμέτρων που απαιτούνται για τον καθορισμό του *joint probability distribution* για το κάθε *Bayesian Network* ενώ η τελευταία στήλη του πίνακα περιέχει τον συνολικό αριθμό των *counters* που χρειάζονται και πρέπει να διατηρήσουμε έτσι ώστε να μπορούμε να εκτιμήσουμε τις παραμέτρους από το κάθε *Bayesian Network*.

Dataset	*Number of nodes*	*Number of edges*	*Number of parameters*	*Number of counters*
ALARM	*37*	*46*	*509*	*983*
HEPAR II	*70*	*123*	*1453*	*2816*
LINK	*724*	*1125*	*14211*	*26609*

Πίνακας 4: Bayesian Networks

Training Data

Όσον αφορά τα *training dataset*, για κάθε *network* δημιουργήσαμε δεδομένα πάνω στο *joint probability distribution* χρησιμοποιώντας την μέθοδο του *Forward Sampling*(*Βλ. Κεφάλαιο 2.4*) Συγκεκριμένα για κάθε *Bayesian Network* δημιουργήσαμε *dataset* όπου το μέγεθος τους κυμαίνεται από $5K$ έως $50M$.Τα δεδομένα αυτά χρησιμοποιούνται αποκλειστικά για την εκμάθηση των παραμέτρων(*learning parameters*) του εκάστοτε *Bayesian Network*.

Testing Data

Όσον αφορά το *testing dataset* που χρησιμοποιήσαμε στα πειράματα δεν είναι τίποτα άλλο από *queries* που αφορούν συγκεκριμένα *events*. Στόχος μας είναι να αξιολογήσουμε την το *accuracy* που παρέχεται από το σύστημα. Συγκεκριμένα θέλουμε να μετρήσουμε την ικανότητα του *trained network* να μπορεί να εκτιμήσει με ακρίβεια(*error guarantees*) την πιθανότητα από διάφορα *events*. Για αυτό το σκοπό δημιουργήσαμε *1000 events* πάνω στο *joint probability distribution* του εκάστοτε *Bayesian Network* και για κάθε ένα τέτοιο *event* θέλουμε την εκτίμηση της πιθανότητας του, χρησιμοποιώντας όμως τις *learning* παραμέτρους που μας παρέχει το αντίστοιχο *trained network*. Όλα τα *events* αντιστοιχούν σε *probability queries* αποκλειστικά πάνω στο *joint probability distribution*. Τέλος δεν υπάρχει κάποιος περιορισμός όσον αφορά την πιθανότητα από τα *events* που χρησιμοποιούμε, δηλαδή το *testing dataset* αποτελείται τόσο από *queries* με μικρή πιθανότητα(*highly unlikely*) όσο και queries με μεγάλη πιθανότητα(*highly likely*), ο μόνος περιορισμός που πρέπει να ικανοποιείται είναι το βασικό αξίωμα πιθανοτήτων δηλαδή η πιθανότητα του *event* να αντιστοιχεί σε κάποιο μη-αρνητικό αριθμό.

6.2 Μετρικές απόδοσης

Communication cost

Αρχικά το *communication cost* που εμφανίζεται παρακάτω αναφέρεται στον συνολικό αριθμό των *bytes* των μηνυμάτων. Συγκεκριμένα αποτελείται από το άθροισμα του αριθμού των *bytes* του *upstream communication cost* με τον αριθμό των *bytes* του *downstream communication cost*.

Error to Ground Truth (GT)

Για κάθε *testing event* υπολογίζουμε την πιθανότητα του χρησιμοποιώντας τις *learning parameters* που προέρχονται από το *approximate model*, δηλαδή από το εκάστοτε *trained network*. Στην συνέχεια το συγκρίνουμε με την *ground truth* πιθανότητα του *event* που προέρχεται κάνοντας χρήση των *ground truth* παραμέτρων του δικτύου, όπως είπαμε και προηγουμένως το *structure* του δικτύου και κατά συνέπεια και οι παράμετροι του είναι γνωστά εκ των προτέρων. Επομένως η πρώτη μετρική που χρησιμοποιούμε αφορά το *error* από την *ground truth(GT)* πιθανότητα και θα αναφερόμαστε σε αυτό ως *Error to GT*.

Error to EXACTMLE

Αντίστοιχα για τους αλγορίθμους *BASELINE*,*UNIFORM* και *NONUNIFORM* υπολογίσουμε το *error* που προκύπτει συγκρίνοντας με τα αποτελέσματα του *EXACTMLE* και στην συνέχεια θα αναφερόμαστε σε αυτό ως *Error to EXACTMLE*.

6.3 Πειραματικά αποτελέσματα

Αρχικά όλα τα παρακάτω πειραματικά αποτελέσματα προκύπτουν έπειτα από την εκτέλεση πέντε ανεξάρτητων *runs*, με τα αποτελέσματα των τιμών που χρησιμοποιούνται παρακάτω να αντιστοιχούν στον μέσο όρο των αποτελεσμάτων των πέντε *runs*. Επιπλέον όλα τα *Bayesian Networks* που χρησιμοποιήσαμε δεν έχουν εμφανίζουν μεγάλες διαφορές στα J_i, K_i μεταξύ των κόμβων με αποτέλεσμα να μην έχουμε σημαντικές διαφορές μεταξύ των αλγορίθμων *BASELINE*,*UNIFORM* και *NONUNIFORM* και στις δυο προσεγγίσεις. Συγκεκριμένα εστιάζουμε στις διαφορές που εμφανίζονται μεταξύ των δυο προσεγγίσεων. Τέλος ισχύει ότι ο αριθμός των *workers* ισούται με 16, δηλαδή $k = 16$, το $\varepsilon = 0.1$ και $\delta = 0.1$, εκτός αν ορίζεται διαφορετικά.

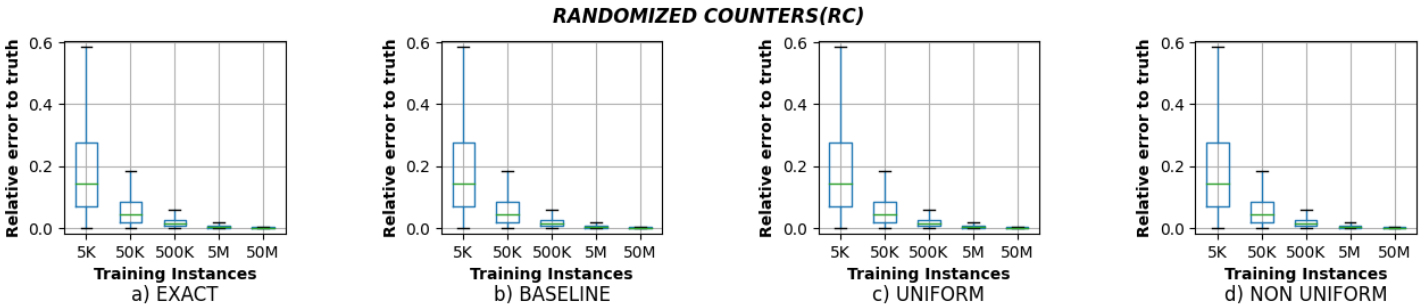
Έχουμε τέσσερις κατηγορίες πειραμάτων, η πρώτη αφορά το *communication cost* μεταβάλλοντας το μέγεθος των *training instances* , η δεύτερη κατηγορία αφορά το *communication cost* μεταβάλλοντας το *approximation factor ε* , η τρίτη κατηγορία αφορά το *communication cost* μεταβάλλοντας τον αριθμό των *workers* ενώ η τελευταία κατηγορία αναφέρεται στην ικανότητα του συστήματος να κάνει *scale*. Και στις τέσσερις περιπτώσεις συγκρίνουμε τόσο τις δυο προσεγγίσεις μεταξύ τους όσο και την καθεμιά προσέγγιση με την περίπτωση των *EXACT counters* δηλαδή με το *EXACTMLE*.

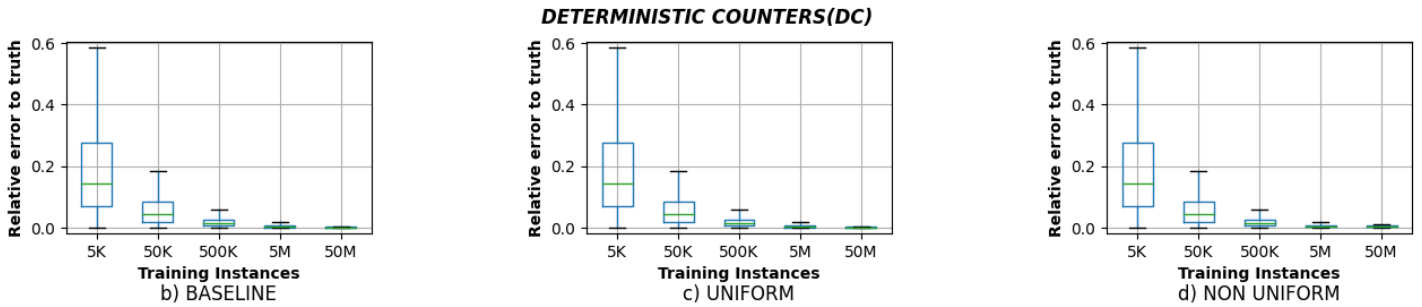
6.3.1 Communication cost σε σχέση με τον αριθμό των training instances

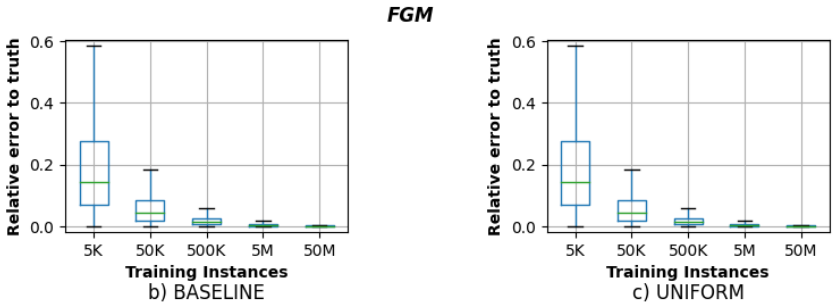
Η πρώτη κατηγορία αφορά το *communication cost* καθώς μεταβάλλουμε το μέγεθος των *training instances*. Συγκεκριμένα το μέγεθος των *training instances* κυμαίνεται από $5K$ έως $50M$, στην συγκεκριμένη περίπτωση το *dataset* είναι το *HEPAR II*.

Αρχικά συγκρίνουμε το *Error to GT* συναρτήσει των *training instances* για καθεμιά από τις δύο προσεγγίσεις, δηλαδή την πρώτη προσέγγιση χρησιμοποιώντας τόσο τους *RANDOMIZED counters*(*RC*) όσο και τους *DETERMINISTIC counters*(*DC*) και την δεύτερη προσέγγιση χρησιμοποιώντας το *FGM* πρωτόκολλο. Συγκεκριμένα το *Error to GT* αντιστοιχεί στο *absolute error* από τα *probability queries*.

Παρατηρούμε λοιπόν ότι καθώς μεταβάλλουμε τον αριθμό των *training instances* σε όλες τις περιπτώσεις , το *error* μειώνεται καθώς αυξάνεται το μέγεθος των *instances* το οποίο είναι αναμενόμενο γιατί όσο περισσότερα δεδομένα έχουμε στην διάθεση μας τόσο περισσότερη πληροφορία για τις παραμέτρους έχουμε, επομένως έχουμε και καλύτερη εκτίμηση αυτών. Επιπλέον μπορούμε να παρατηρήσουμε ότι καθώς αυξάνεται ο αριθμός των δεδομένων, υπάρχει συρρίκνωση του *interquartile range* το οποίο σημαίνει ότι και το *variance* του *error* των *probability queries* μικραίνει καθώς αυξάνεται ο αριθμός των δεδομένων. Επιπροσθέτως μπορούμε να παρατηρήσουμε ότι έχουμε αρκετά καλό *accuracy* και στις δυο προσεγγίσεις αφού για παράδειγμα με τα $5M$ *instances* το *median error* είναι πάντα λιγότερο από το 1% για κάθε προσέγγιση. Τέλος στο *Παράρτημα* υπάρχει και το διάγραμμα *mean error to GT* για το δίκτυο *HEPAR II* και για τις δύο προσεγγίσεις.

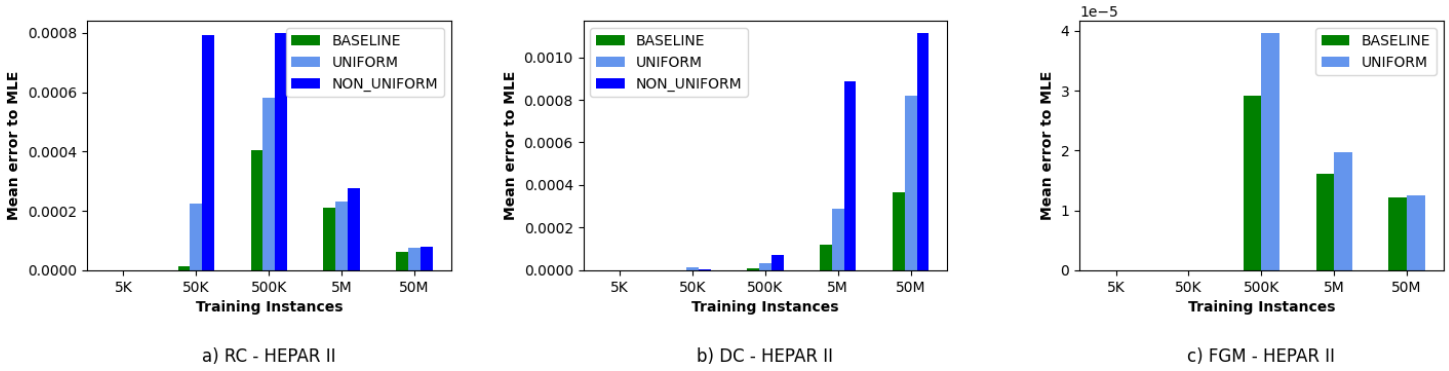






Εικόνα 19: Error to GT σε σχέση με τον αριθμό των training instances και για τις δυο προσεγγίσεις, για το dataset HEPAR II

Στην συνέχεια συγκρίνουμε το *Error to EXACTMLE* συναρτήσει των *training instances* για καθεμιά από τις δύο προσεγγίσεις. Συγκεκριμένα το *Error to EXACTMLE* σε αυτήν την περίπτωση αντιστοιχεί στο *mean absolute error* των *probability queries*. Το *error* αυτό προέρχεται από δύο πηγές, πρώτον περιλαμβάνει το *statistical error* το οποίο είναι έμφυτο εξαιτίας του αριθμού των *training instances* που έχουμε δει, δηλαδή το *error* σε σχέση με τις *ground truth τιμές* των παραμέτρων και δεύτερον περιλαμβάνει το *approximation error*, δηλαδή το *error* ανάμεσα στο μοντέλο το οποίο κάνουμε *maintenance* και στο μοντέλο που προκύπτει χρησιμοποιώντας *EXACT counters*, με το τελευταίο να προκύπτει από την χρήση του *approximation factor ε*.



Εικόνα 20: Error to EXACTMLE σε σχέση με τον αριθμό των training instances και για τις δυο προσεγγίσεις, για το dataset HEPAR II

Σκοπός μας όπως είπαμε είναι να μπορούμε να ελέγχουμε το *approximation error* προς το *EXACTMLE* δηλαδή το *error* σε σχέση με την περίπτωση που χρησιμοποιούσαμε *EXACT counters*. Μπορούμε να παρατηρήσουμε ότι σε όλες τις προσεγγίσεις το *error* το οποίο έχουμε παραμένει κατά προσέγγιση το ίδιο καθώς μεταβάλλουμε τον αριθμό των *training instances* κάτι το οποίο επιβεβαιώνει και την λειτουργικότητα της προσέγγισης της οποίας προτείναμε, δηλαδή εγγυόμαστε σε όλες τις προσεγγίσεις σε συνδυασμό με τους αλγορίθμους *BASELINE,UNIFORM* και *NON_UNIFORM* ότι το *error* που έχουμε είναι *bounded*(*error guarantees*), όπου τα *error guarantees* καθορίζονται σε σχέση πάντα με το *approximation factor ε,δ*. Στην περίπτωση μικρού αριθμού *training instances*($\leq 50K$) το *error* το οποίο έχουμε είναι αρκετά μικρό , δηλαδή κοντά στο μηδέν και αυτό οφείλεται στο γεγονός ότι έχουμε *tight bounds* για κάθε *counter* αφού οι τιμές των *counters* είναι αρκετά μικρές, μιλώντας αριθμητικά. Για παράδειγμα στην περίπτωση μας, όπου το $\varepsilon = 0.1$ το

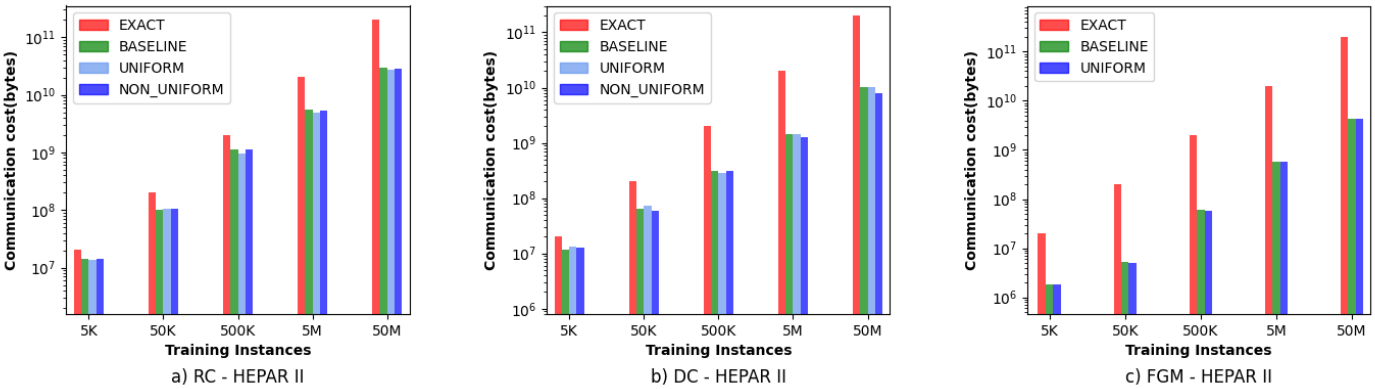
approximation factor κάθε *counter* είναι περίπου της τάξης του 10^{-4} , επομένως μπορούμε να καταλάβουμε ότι τα *bounds* είναι αρκετά *tight* για μικρές τιμές των *counters*.

Επιπλέον ισχύει ότι το μικρότερο *error* δηλαδή το καλύτερο *accuracy* σε σχέση με το *EXACTMLE* το εμφανίζει ο αλγόριθμος του *BASELINE* και αυτό οφείλεται στο γεγονός ότι ο αλγόριθμος *BASELINE* παρουσιάζει πιο *tight bounds* σε κάθε *counter* σε σχέση με τους άλλους δυο αλγορίθμους, μετέπειτα ακολουθούν ο αλγόριθμος *UNIFORM* και *NON_UNIFORM* που έχουν παρόμοιο *accuracy*. Τέλος σκοπός αυτού του πειράματος είναι να αποδείξει την λειτουργικότητα και των δύο προσεγγίσεων και όχι για να βγάλουμε κάποιο συμπέρασμα μεταξύ των αλγορίθμων *BASELINE,UNIFORM* και *NON_UNIFORM*.

Τέλος συγκρίνουμε το *communication cost* συναρτήσει των *training instances* τόσο μεταξύ των δύο προσεγγίσεων όσο και της καθεμίας προσέγγισης σε σχέση με το *communication cost* από το *EXACTMLE*. Να σημειώσουμε ότι ο *άξονας y* βρίσκεται σε λογαριθμική κλίμακα.

Αρχικά σημαντικές διαφορές μεταξύ των αλγορίθμων *BASELINE,UNIFORM* και *NON_UNIFORM* δεν παρουσιάζονται γιατί όπως είπαμε και προηγουμένως δεν εμφανίζονται μεγάλες διαφορές στα J_i, K_i μεταξύ των κόμβων του δικτύου *HEPAR II* όσο και των υπολοίπων. Γενικά ισχύει ότι ο αλγόριθμος *UNIFORM* και *NON_UNIFORM* είναι αρκετά κοντά μεταξύ τους ενώ και οι δυο είναι καλύτεροι του αλγορίθμου *BASELINE* καθώς αυξάνεται ο αριθμός των *training instances*.

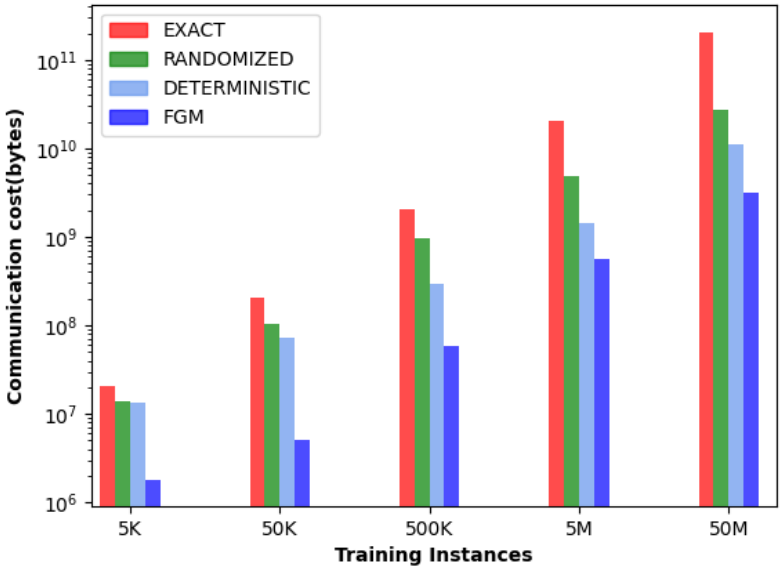
Επιπλέον μπορούμε να παρατηρήσουμε ότι όσο αυξάνεται ο αριθμός των *training instances* τόσο μεγαλύτερη είναι η διαφορά σε σχέση με τον *EXACTMLE* το οποίο είναι επιθυμητό καθώς αποδεικνύει ότι και οι δυο προσεγγίσεις μπορούν να διαχειριστούν μεγάλο αριθμό δεδομένων με ικανοποιητικό τρόπο δηλαδή χρησιμοποιώντας όσον τον δυνατόν λιγότερο *communication cost*. Συγκεκριμένα παρατηρούμε τόσο στην περίπτωση των *RANDOMIZED* όσο και των *DETERMINISTIC counters* ότι έχουμε ως και μια τάξη μεγέθους λιγότερο *communication cost* σε σχέση με το *EXACTMLE* , το οποίο πρακτικά σημαίνει ότι στέλνουν *10 φορές* λιγότερα μηνύματα σε σχέση με το *EXACTMLE*. Όσον αφορά την περίπτωση της υλοποίησης που προτείναμε δηλαδή το *FGM* , παρατηρούμε ότι έχουμε ως δυο τάξεις μεγέθους λιγότερο *communication cost* σε σχέση με το *EXACTMLE* , το οποίο πρακτικά σημαίνει ότι στέλνει *100 φορές* λιγότερα μηνύματα σε σχέση με το *EXACTMLE*.



Εικόνα 21: Communication cost σε σχέση με τον αριθμό των training instances και για τις δυο προσεγγίσεις, για το dataset HEPAR II

Επιπλέον μπορούμε να παρατηρήσουμε ότι η αύξηση του *communication cost* είναι σχεδόν λογαριθμική σε σχέση με τον αριθμό των *training instances*. Αυτό φαίνεται καλύτερα στην περίπτωση που αφορά το *dataset LINK*. Αποτελέσματα τόσο από το *dataset LINK* όσο και από το *dataset ALARM* υπάρχουν στο *Παράρτημα*.

Τέλος στην παρακάτω εικόνα μπορούμε να δούμε μια σύγκριση μεταξύ των δύο προσεγγίσεων. Για κάθε προσέγγιση μεταξύ των αλγορίθμων *BASELINE,UNIFORM* και *NON_UNIFORM* χρησιμοποιείται εκείνος ο αλγόριθμος που έχει το μικρότερο c*ommunication cost* για κάθε προσέγγιση. Παρατηρούμε λοιπόν ότι η προσέγγιση που προτείναμε είναι καλύτερη από την πρώτη προσέγγιση δηλαδή την περίπτωση να *χρησιμοποιούμε individually approximate distributed counters*(είτε *RANDOMIZED* είτε *DETERMINISTIC*), έως και μια τάξη μεγέθους. Όσον αφορά την περίπτωση σύγκρισης μεταξύ των *RANDOMIZED* και *DETERMINISTC* counters, παρατηρούμε ότι ο *DETERMINISTC* χρησιμοποιεί έως 50% λιγότερο *communication cost* σε σχέση με τον *RANDOMIZED*(*Βλ. Κεφάλαιο 4.1.1.3*).

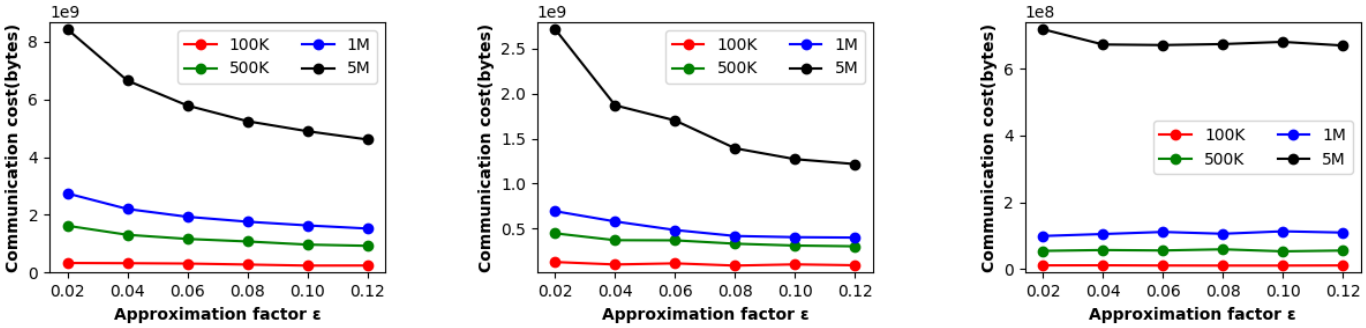


Εικόνα 22: Communication cost σε σχέση με τον αριθμό των training instances και για τις δυο προσεγγίσεις για το dataset HEPAR II-Best results

6.3.2 Communication cost σε σχέση με το approximation factor ε

Σε αυτήν την περίπτωση συγκρίνουμε το *communication cost* συναρτήσει του *approximation factor ε*. Συγκεκριμένα μεταβάλλουμε το *approximation factor ε* στο διάστημα $[0.02,0.12]$ με βήμα 0.02 και μετράμε το *communication cost* και για τις δυο προσεγγίσεις. Τα μεγέθη των *datasets* που χρησιμοποιήσαμε είναι $100K, 500K, 1M, 5M$ αντίστοιχα. Το *dataset* που χρησιμοποιούμε είναι το *HEPAR II*. Παρακάτω βλέπουμε τα αποτέλεσμα των προσεγγίσεων σε συνδυασμό με τον αλγόριθμο *UNIFORM*, τα αποτελέσματα του *communication cost* συναρτήσει του *approximation factor ε* για τους άλλους δυο αλγορίθμους βρίσκονται στο *Παράρτημα*.

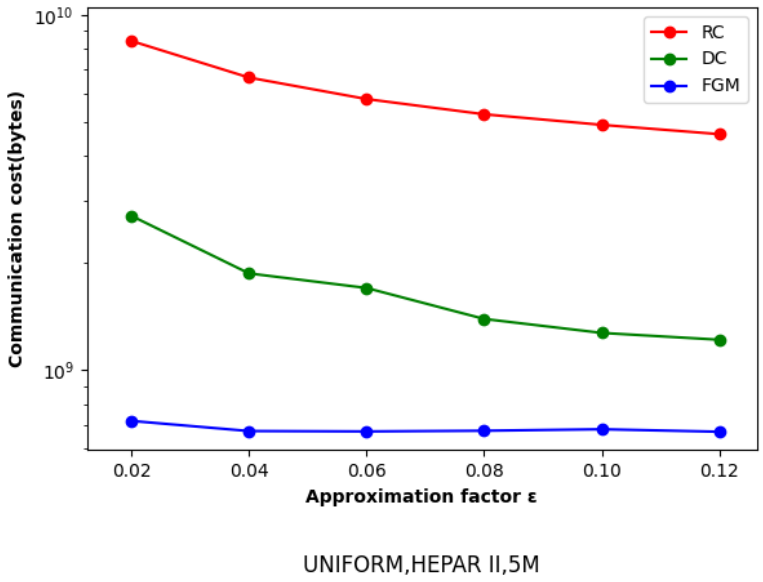
Παρατηρούμε λοιπόν ότι και στις δυο προσεγγίσεις για όλους τους αλγορίθμους *BASELINE,UNIFORM* και *NON_UNIFORM* το *communication cost* δεν έχει την ίδια μεταβολή σε σχέση με την μεταβολή του *approximation factor ε*, κάτι που κάνει και τις δυο προσεγγίσεις να μην είναι *sensitive* σε μεταβολές του *approximation factor ε*. Ακόμα και στην αρχή, δηλαδή στην μετάβαση του *approximation factor ε* από το 0.02 στο 0.04, όσον αφορά την πρώτη προσέγγιση (τόσο στην περίπτωση του *RANDOMIZED* όσο και στην περίπτωση του *DETERMINISTIC*) για όλα τα μεγέθη των *datasets* έχουμε ένα ποσοστό μείωσης ανάμεσα στο $15\% - 20\%$ που σταδιακά εξασθενεί καθώς μεταβάλλουμε το *approximation factor ε*. Όσον αφορά την δεύτερη προσέγγιση το *communication cost* παραμένει σχεδόν σταθερό καθώς μεταβάλλουμε το *approximation factor ε* , ουσιαστικά η δεύτερη προσέγγιση δεν έχει σχεδόν καμιά επίδραση στο *communication cost* με την μεταβολή του *approximation factor ε*.



a) RC - UNIFORM b) DC - UNIFORM c) FGM - UNIFORM

Εικόνα 23: Communication cost σε σχέση με το approximation factor ε, για το dataset HEPAR II και τον αλγόριθμο UNIFORM

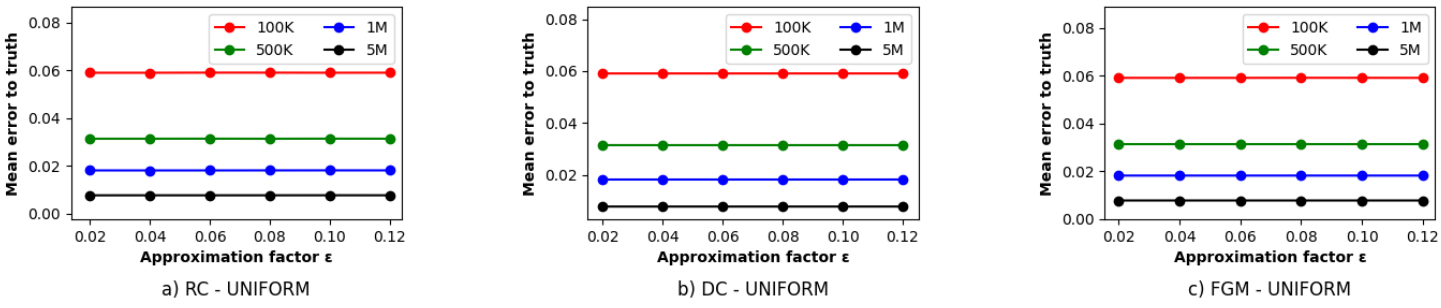
Συνοψίζοντας αν θέλουμε να μιλήσουμε συγκριτικά μεταξύ των προσεγγίσεων την καλύτερη εξάρτηση σε σχέση με το *approximation factor ε* την έχει η προσέγγιση που προτείναμε δηλαδή το *FGM*. Τέλος αν θέλουμε να συγκρίνουμε τους δυο *counters* της πρώτης προσέγγισης μπορούμε να πούμε ότι έχουν περίπου την ίδια εξάρτηση σε σχέση με το *approximation factor* ε. Να σημειώσουμε ότι ο *άξονας y* της *Εικόνα 24* βρίσκεται σε λογαριθμική κλίμακα.



Εικόνα 24: Communication cost σε σχέση με το approximation factor ε και για τις δύο προσεγγίσεις, για το dataset HEPAR II και τον αλγόριθμο UNIFORM

Τέλος παρακάτω παρουσιάζουμε το διάγραμμα *Error to GT* καθώς μεταβάλλουμε το *approximation factor ε* και για τις δυο προσεγγίσεις με βάση με τον αλγόριθμο *UNIFORM*, αντίστοιχα είναι και τα αποτελέσματα για τους άλλους δυο αλγορίθμους. Σε αυτήν την περίπτωση το *error* αντιστοιχεί στο *mean absolute error* των *probability queries*.

Παρατηρούμε ότι καθώς μεταβάλλουμε το *approximation factor ε* το *error* δεν μεταβάλλεται σημαντικά (διαφορές υπάρχουν απλώς δεν είναι αρκετά μεγάλες), αυτό οφείλεται στο γεγονός ότι το *approximation factor ε* επηρεάζει μόνο το *approximation error* το οποίο στην προκειμένη περίπτωση είναι αρκετά μικρότερο σε σχέση με το *statistical error*(συγκεκριμένα έως και δυο τάξεις μεγέθους μικρότερο), επομένως έχει ως αποτέλεσμα να συρρικνώνει την επίδραση του *approximation error* και για αυτό τον λόγο δεν βλέπουμε σημαντικές αλλαγές καθώς μεταβάλλεται το *approximation factor ε*. Μπορούμε όμως να παρατηρήσουμε μεταβολές στο *error* για τα διάφορα μεγέθη των *datasets*, συγκεκριμένα το *error* μικραίνει καθώς αυξάνεται το μέγεθος των *datasets*. Αυτό οφείλεται στο γεγονός ότι το *statistical error* μικραίνει καθώς αυξάνεται το μέγεθος των *datasets*. Τέλος το *accuracy* το οποίο έχουμε είναι αρκετά καλό, συγκεκριμένα είναι μικρότερο του 6% για όλες τις προσεγγίσεις και για κάθε μέγεθος του *dataset*.

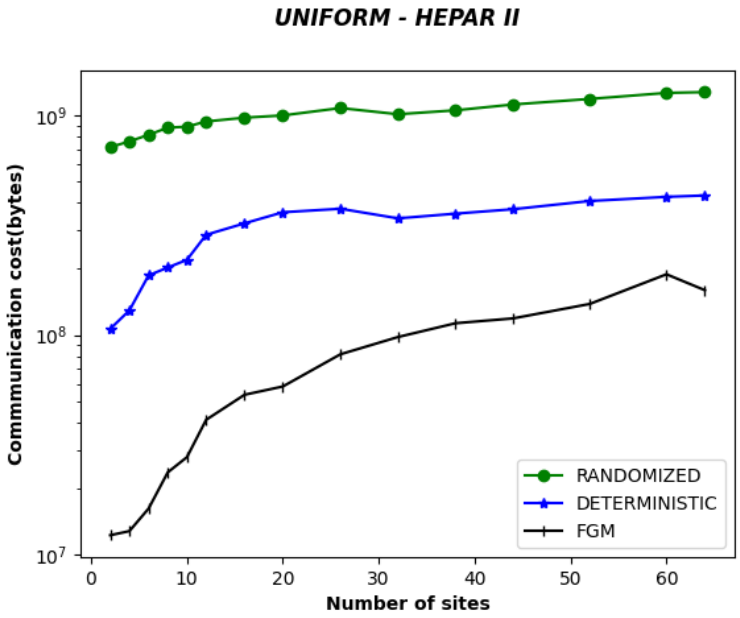


Εικόνα 25: Error to GT σε σχέση με το approximation factor ε για τις δύο προσεγγίσεις, για το dataset HEPAR II και τον αλγόριθμο UNIFORM

6.3.3 Communication cost σε σχέση με τον αριθμό των workers

Σε αυτήν την περίπτωση συγκρίνουμε το *communication cost* συναρτήσει του αριθμού των *sites*. Συγκεκριμένα μεταβάλλουμε τον αριθμό των *sites* στο διάστημα $[2,64]$ και μετράμε το *communication cost* και για τις δυο προσεγγίσεις. Το μέγεθος του *dataset* που χρησιμοποιήσαμε είναι $500K$ και αναφέρεται στο *HEPAR II*. Παρακάτω βλέπουμε το *communication cost* από τις δυο προσεγγίσεις σε συνδυασμό με τον αλγόριθμο *UNIFORM*, για τους υπόλοιπους αλγόριθμους τα αντίστοιχα διαγράμματα βρίσκονται στο *Παράρτημα*. Τέλος να σημειώσουμε ότι ο *άξονας y* βρίσκεται σε *λογαριθμική κλίμακα*.

Μπορούμε να παρατηρήσουμε ότι το *communication cost* αυξάνεται με *sub-linearly* τρόπο σε σχέση με τον αριθμό των *sites* και στις δυο προσεγγίσεις. Όσον αφορά την καλύτερη προσέγγιση δηλαδή την προσέγγιση με την καλύτερη εξάρτηση σε σχέση με τον αριθμό των *sites*, παρόλο που υπάρχει διαφορά μεταξύ των *communication cos*t των προσεγγίσεων (για αυτό επιλέχθηκε και *λογαριθμική κλίμακα*), είναι η πρώτη προσέγγιση σε συνδυασμό τους *RANDOMIZED counters* , το οποίο επιβεβαιώνεται και από τους θεωρητικούς υπολογισμούς. Μετέπειτα ακολουθά η δεύτερη προσέγγιση δηλαδή το *FGM* και στο τέλος έρχεται η πρώτη προσέγγιση σε συνδυασμό με τους *DETERMINISTIC counters.*



Εικόνα 26: Communication cost σε σχέση με το αριθμό των sites για τις δύο προσεγγίσεις, για το dataset HEPAR II και τον αλγόριθμο UNIFORM

6.3.4 Scalability

Μέχρι στιγμής εστιάσαμε στην εκτίμηση του *communication cost* για διάφορες παραμέτρους του συστήματος μας. Σε αυτήν την περίπτωση εστιάζουμε στην απόδοση που έχει το σύστημα μας χρησιμοποιώντας καθεμία από τις δυο προσεγγίσεις. Για την εκτίμηση της απόδοσης χρησιμοποιούμε δυο μετρικές. Η πρώτη αφορά το *throughput*(*events/sec*) του συστήματος μας δηλαδή τον μέσο αριθμό από *training instances* που μπορεί να διαχειριστεί το σύστημα ανά δευτερόλεπτο ενώ η δεύτερη αφορά το *runtime(sec)* το οποίο αντιστοιχεί στο συνολικό χρόνο από το πρώτο μήνυμα που λαμβάνει ο *Coordinator* μέχρι και το τελευταίο κατά την διάρκεια της διαδικασίας του *training*.

Επομένως για να μετρήσουμε την απόδοση χρησιμοποιώντας τις δυο προαναφερθείσες μετρικές κάναμε δυο είδη πειραμάτων. Στο πρώτο πείραμα μεταβάλλουμε τον αριθμό των *sites* και υπολογίσουμε τις δυο μετρικές ενώ στο δεύτερο πείραμα μεταβάλλουμε τον *παραλληλισμό* δηλαδή τον αριθμό των *subtasks* που μπορεί να χρησιμοποιήσει το σύστημα μας (μπορούμε να το δούμε και ως το μέγεθος των *resources* που χρησιμοποιεί το σύστημα) και αντίστοιχα υπολογίσουμε τις δυο μετρικές.

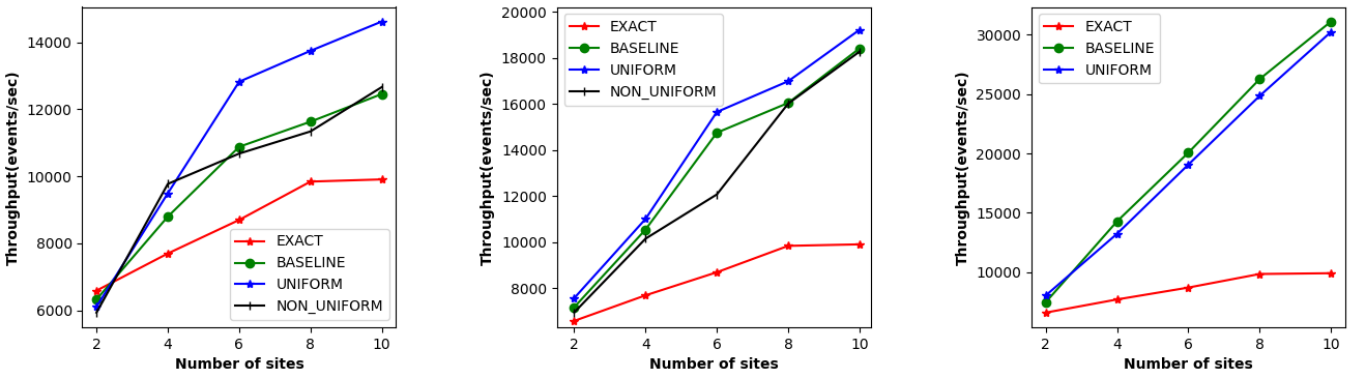
Σε κάθε πείραμα συγκρίνουμε τις προσεγγίσεις τόσο μεταξύ τους όσο και την καθεμία με την περίπτωση του *EXACTMLE*. Τέλος όλα τα παρακάτω πειράματα αναφέρονται στο *dataset HEPAR II* ενώ το μέγεθος ισούται με $500K$. Παρακάτω βλέπουμε μόνο τα αποτελέσματα από το *throughput*, τα αποτελέσματα από τον *runtime* βρίσκονται στο *Παράρτημα*. Να αναφέρουμε επειδή τα μεγέθη του *throughput* και του *runtime* είναι ανάλογα μεταξύ τους ότι ισχύει για την περίπτωση του *throughput* ισχύει και για την περίπτωση του *runtime* μόνο που στην περίπτωση του *throughput* επιδιώκουμε την αύξηση του ενώ στην περίπτωση του *runtime* επιδιώκουμε την μείωση του.

Στην πρώτη περίπτωση λοιπόν υπολογίσουμε τόσο *throughput(event/sec)* όσο και το *runtime(sec)* συναρτήσει του αριθμού των *sites*. Συγκεκριμένα μεταβάλλουμε τον αριθμό των *sites* στο διάστημα $[2,10]$ με βήμα 2. Τέλος να επισημάνουμε ότι στον αριθμό των *sites* δεν συμπεριλαμβάνεται ο *Coordinator*.

Όσον αφορά την πρώτη προσέγγιση τόσο στην περίπτωση του *RANDOMIZED* και του *DETERMINISTIC counter*, μπορούμε να παρατηρήσουμε ότι καθώς αυξάνουμε τον αριθμό των *sites* δεν υπάρχει και η ανάλογη αύξηση στο *throughput* , συγκεκριμένα αρχίζει να συμβαίνει όταν ο αριθμός των *sites* είναι μεγαλύτερος από έξι. Αυτό οφείλεται στο γεγονός ότι καθώς αυξάνεται ο αριθμός των *sites* αυξάνεται και το *communication cost*, συγκεκριμένα την μεγαλύτερη αύξηση στο *communication cost* την παρατηρούμε σε μικρό αριθμό από *sites(Εικόνα 26)* και για αυτό τον λόγο επιλέξαμε ο αριθμός των *sites* να βρίσκεται στο διάστημα $[2,10]$.

Επομένως η αύξηση του *communication cost*, τουλάχιστον στο αρχικό στάδιο(*initial state*) δηλαδή στο αρχικό σημείο όπου όλοι οι *counters* εμφανίζουν *violations*, έχει ως αποτέλεσμα ο όγκος των μηνυμάτων που φτάνουν στον *Coordinator* να είναι αρκετά μεγάλος με συνέπεια να δημιουργείται *backpressure* στα *sites*. Το *backpressure* εμφανίζεται μόνο στο αρχικό στάδιο(*initial state*) δηλαδή εκεί που έχουμε αρκετά μικρούς *counters* με πολύ *tight* bounds, στην συνέχεια όπως μπορούμε να δούμε και παρακάτω εξαλείφεται. Τέλος μπορούμε να παρατηρήσουμε ότι το *throughput* από το *EXACTMLE* παραμένει σχεδόν το ίδιο με την αύξηση του αριθμού των *sites* , το οποίο επιβεβαιώνει ότι το *communication cost* από το

EXACTMLE είναι αρκετά μεγάλο με αποτέλεσμα το *backpressure* που εμφανίζεται στα *sites* να είναι διαρκές και επομένως δεν έχουμε και το κατάλληλο *scaling* όσον αφορά την περίπτωση του *EXACTMLE*.

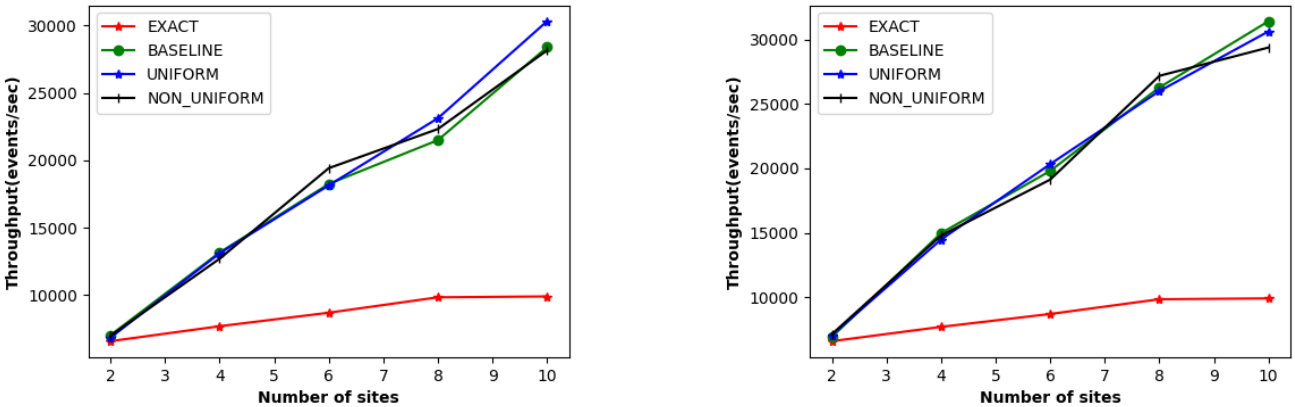


b) RC - HEPAR II b) DC - HEPAR II c) FGM - HEPAR II

Εικόνα 27: Throughput(events/sec) σε σχέση με το αριθμό των sites για τις δύο προσεγγίσεις, για το dataset HEPAR II

Όσον αφορά την δεύτερη προσέγγιση μπορούμε να δούμε ότι η αύξηση στο *throughput* είναι γραμμική σε σχέση με τον αριθμό με των *sites* , το οποίο επιβεβαιώνει ότι δεν εμφανίζεται κάποιο *backpressure* στα *sites* με συνέπεια η προσέγγιση που προτείναμε να έχει το ιδανικό *scaling*. Η διαφορά μεταξύ τους οφείλεται στο γεγονός ότι το *communication cost* της προσέγγισης που υλοποιήσαμε είναι αρκετά μικρότερο από το *communication cost* της πρώτης προσέγγισης. Τέλος όποιες διαφορές παρουσιάζονται τόσο στο *throughput* όσο και στο *runtime* μεταξύ των αλγορίθμων *BASELINE,UNIFORM* και *NON_UNIFORM* οφείλονται στην διαφορά του *communication cost* που υπάρχει μεταξύ τους και είναι ανάλογη της διαφοράς αυτής.

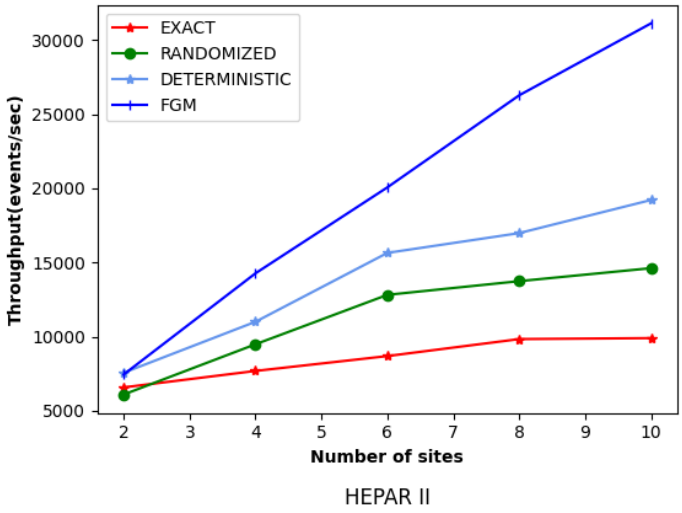
Στην συνέχεια μετρήσαμε το *throughput* όσο και το *runtime* της πρώτης προσέγγισης μετά το αρχικό στάδιο(*initial state*) δηλαδή μετά το σημείο όπου όλοι οι *counters* εμφανίζουν *violations* και μπορούμε να παρατηρήσαμε ότι το *throughput* έχει ανάλογη αύξηση σε σχέση με τον αριθμό των *sites* ενώ αντίστοιχα το *runtime* έχει την ανάλογη μείωση σε σχέση με τον αριθμό των *sites*, το οποίο επιβεβαιώνει και το γεγονός ότι το *backpressure* μετά το αρχικό στάδιο(*initial state*) αρχίζει να εξαλείφεται με αποτέλεσμα και η πρώτη προσέγγιση και για τα δυο *counters* μετά το αρχικό στάδιο να έχει το ιδανικό *scaling*.



a) RC - HEPAR II, after initial state b) DC - HEPAR II, after initial state

Εικόνα 28: Throughput(events/sec) σε σχέση με το αριθμό των sites για την πρώτη προσέγγιση, για το dataset HEPAR II μετά το αρχικό στάδιο

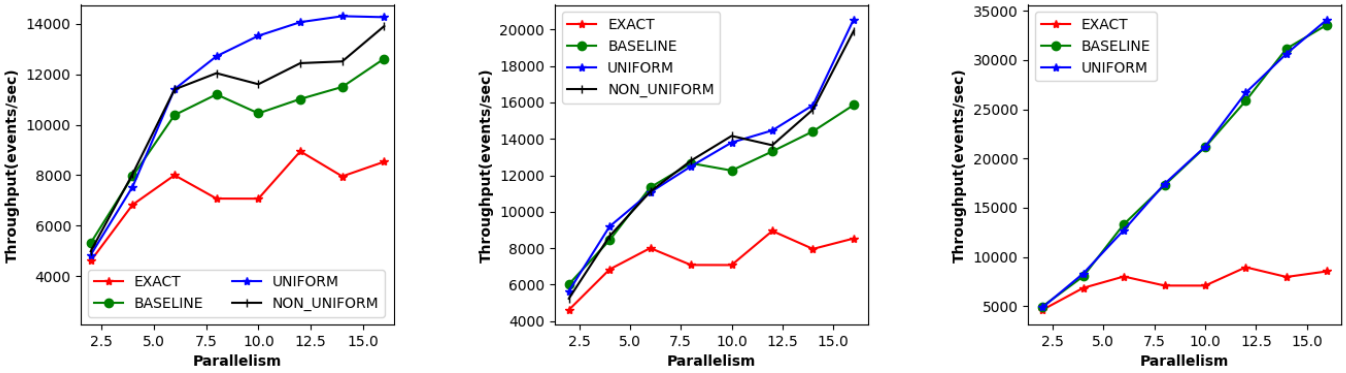
Τέλος στην παρακάτω εικόνα μπορούμε να δούμε μια σύγκριση μεταξύ των δύο προσεγγίσεων. Για κάθε προσέγγιση μεταξύ των αλγορίθμων *BASELINE,UNIFORM* και *NON_UNIFORM* χρησιμοποιείται εκείνος ο αλγόριθμος που εμφανίζει το μεγαλύτερο *throughput* για κάθε προσέγγιση(ενώ στην περίπτωση του *runtime* χρησιμοποιείται ο αλγόριθμος με το μικρότερο *runtime*). Όσον αφορά το *throughput* της πρώτης προσέγγισης αντιστοιχεί στο *throughput* χωρίς να θεωρήσουμε κάποιο αρχικό στάδιο(*initial state*). Μπορούμε να παρατηρήσουμε ότι η προσέγγιση που υλοποιήσαμε είναι αρκετά καλύτερη και αυτό οφείλεται στην διαφορά του *communication cost* που εμφανίζεται μεταξύ τους, δηλαδή η διαφορά στο *throughput* είναι ανάλογη και της διαφοράς στο *communication cost*. Τέλος μπορούμε να παρατηρήσουμε ότι το *throughput* τόσο της πρώτης προσέγγισης όσο και της δεύτερης προσέγγισης είναι αρκετά καλύτερο από το *throughput* του *EXACTMLE* το οποίο επιβεβαιώνει και την διαφορά στο *communication cost* που έχουν μεταξύ τους.



Εικόνα 29: Throughput(events/sec) σε σχέση με το αριθμό των sites για τις δύο προσεγγίσεις, για το dataset HEPAR II-Best results

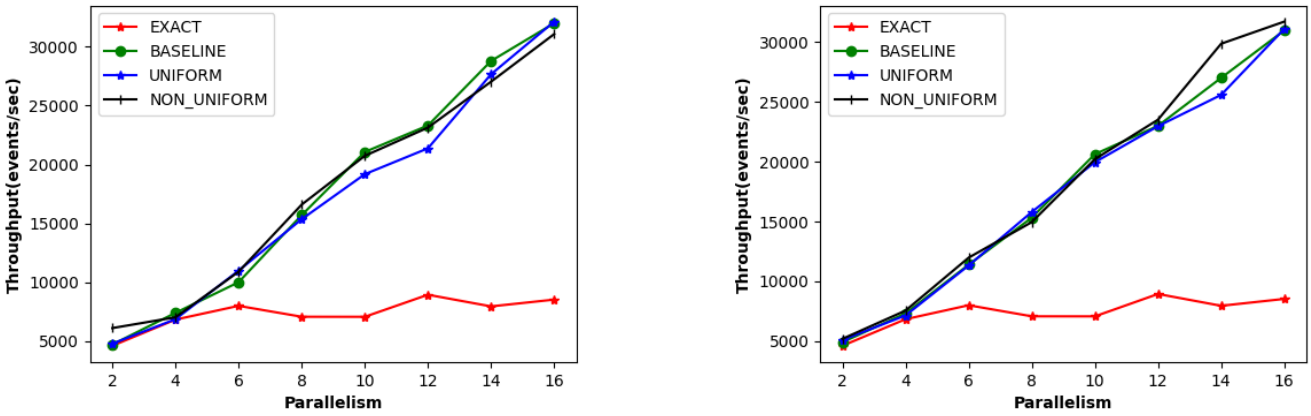
Στην δεύτερη περίπτωση λοιπόν υπολογίσουμε τόσο το *throughput(event/sec)* όσο και το *runtime(sec)* συναρτήσει του αριθμού του παραλληλισμού. Συγκεκριμένα μεταβάλλουμε τον αριθμό του παραλληλισμού στο διάστημα $[2,16]$ με βήμα 2 ενώ ο αριθμός των *sites* είναι ίσος με 16.

Ότι ισχύει στην πρώτη περίπτωση εμφανίζεται και σε αυτήν την περίπτωση. Πάλι μπορούμε να παρατηρήσουμε ότι στην πρώτη προσέγγιση καθώς αυξάνεται ο αριθμός του παραλληλισμού τόσο *throughput* όσο και το *runtime* δεν έχουν την ανάλογη μεταβολή με την μεταβολή του αριθμού του παραλληλισμού.



Εικόνα 30: Throughput(events/sec) σε σχέση με το αριθμό του παραλληλισμού για τις δύο προσεγγίσεις, για το dataset HEPAR II

Αυτό οφείλεται στο γεγονός ότι καθώς αυξάνεται ο αριθμός του παραλληλισμού ο ρυθμός με τον οποίο φτάνουν τα μηνύματα στο *Coordinator* να είναι μεγαλύτερος από τον ρυθμό με τον οποίο μπορεί να διαχειριστεί τα μηνύματα, με αποτέλεσμα να δημιουργείται και πάλι *backpressure* στα *sites*. Το *backpressure* όσον αφορά την πρώτη προσέγγιση εμφανίζεται μόνο στο αρχικό στάδιο (*initial state*) δηλαδή εκεί που έχουμε αρκετά μικρούς *counters* με πολύ *tight* bounds και επομένως πολλά *violations* με συνέπεια το *communication cost* να είναι αρκετά μεγάλο. Στην συνέχεια όπως μπορούμε να δούμε και παρακάτω εξαλείφεται, συγκεκριμένα αν μετρήσουμε το *throughput* μετά το αρχικό στάδιο(*initial state*) παρατηρούμε ότι και η πρώτη προσέγγιση έχει την ανάλογη αύξηση στο *throughput*(*Εικόνα 31*) σε σχέση με την μεταβολή στον αριθμό του παραλληλισμού.

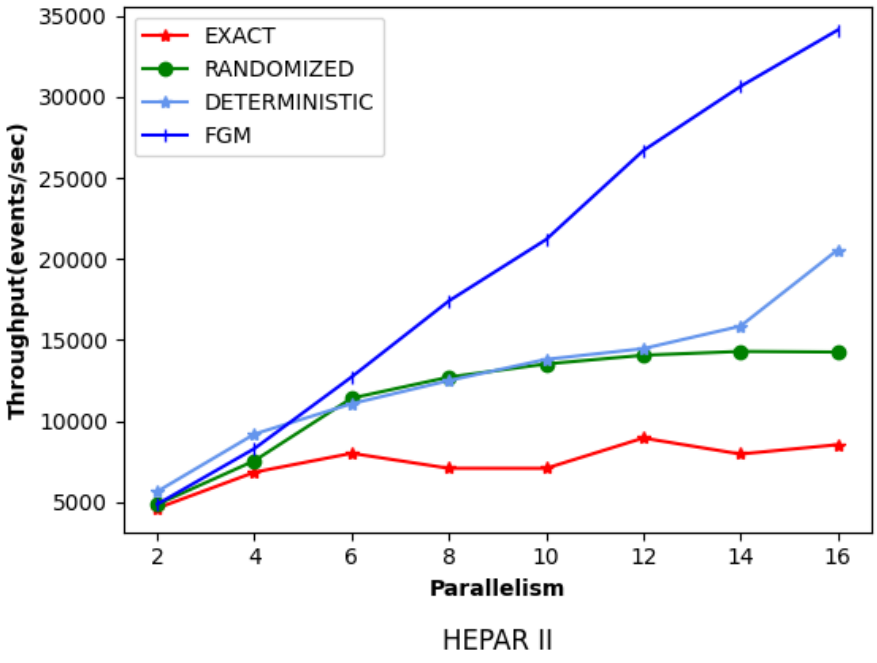


Εικόνα 31: Throughput(events/sec) σε σχέση με το αριθμό του παραλληλισμού για την πρώτη προσέγγιση, για το dataset HEPAR II μετά το αρχικό στάδιο

Όσον αφορά την δεύτερη προσέγγιση μπορούμε να δούμε ότι η αύξηση στο *throughput* είναι γραμμική σε σχέση με τον αριθμό του παραλληλισμού , το οποίο επιβεβαιώνει ότι δεν εμφανίζεται κάποιο *backpressure* στα *sites* με συνέπεια η προσέγγιση που προτείναμε να έχει το ιδανικό *scaling* όσον αφορά και τον αριθμό του παραλληλισμού δηλαδή σε σχέση με τα *resources* που μπορεί να χρησιμοποιήσει.

Επιπλέον μπορούμε να παρατηρήσουμε ότι το *throughput* από το *EXACTMLE* παραμένει σχεδόν το ίδιο με την αύξηση του αριθμού του παραλληλισμού, το οποίο επιβεβαιώνει ότι ο ρυθμός με τον οποίο φτάνουν τα μηνύματα στο *Coordinator* είναι αρκετά μεγάλος. Αυτό οφείλεται στο γεγονός ότι το *communication cost* από το *EXACTMLE* είναι αρκετά μεγάλο με αποτέλεσμα το *backpressure* που εμφανίζεται στα *sites* να διαρκές. Επομένως στη περίπτωση του *EXACTMLE* δεν έχουμε το κατάλληλο *scaling* όσον αφορά τα *resources* που μπορεί να χρησιμοποιήσει.

Τέλος στην παρακάτω εικόνα μπορούμε να δούμε μια σύγκριση μεταξύ των δύο προσεγγίσεων. Για κάθε προσέγγιση μεταξύ των αλγορίθμων *BASELINE,UNIFORM* και *NON_UNIFORM* χρησιμοποιείται εκείνος ο αλγόριθμος που εμφανίζει το μεγαλύτερο *throughput* για κάθε προσέγγιση (ενώ στην περίπτωση του *runtime* χρησιμοποιείται ο αλγόριθμος με το μικρότερο *runtime*). Μπορούμε να παρατηρήσουμε και πάλι ότι η προσέγγιση που υλοποιήσαμε είναι καλύτερη σε σχέση με την πρώτη προσέγγιση και επομένως μπορεί να διαχειριστεί με καλύτερο τρόπο τα διαθέσιμα *resources* εξασφαλίζοντας ταυτόχρονα και το κατάλληλο *scaling* ολόκληρου του συστήματος. Η διαφορά αυτή οφείλεται και πάλι στην διαφορά του *communication cost* που υπάρχει μεταξύ τους. Επιπροσθέτως μπορούμε να παρατηρήσουμε ότι το *throughput* τόσο της πρώτης όσο και της δεύτερης προσέγγισης είναι αρκετά καλύτερο από το *throughput* του *EXACTMLE* το οποίο επιβεβαιώνει και πάλι την διαφορά στο *communication cost* που υπάρχει μεταξύ τους.

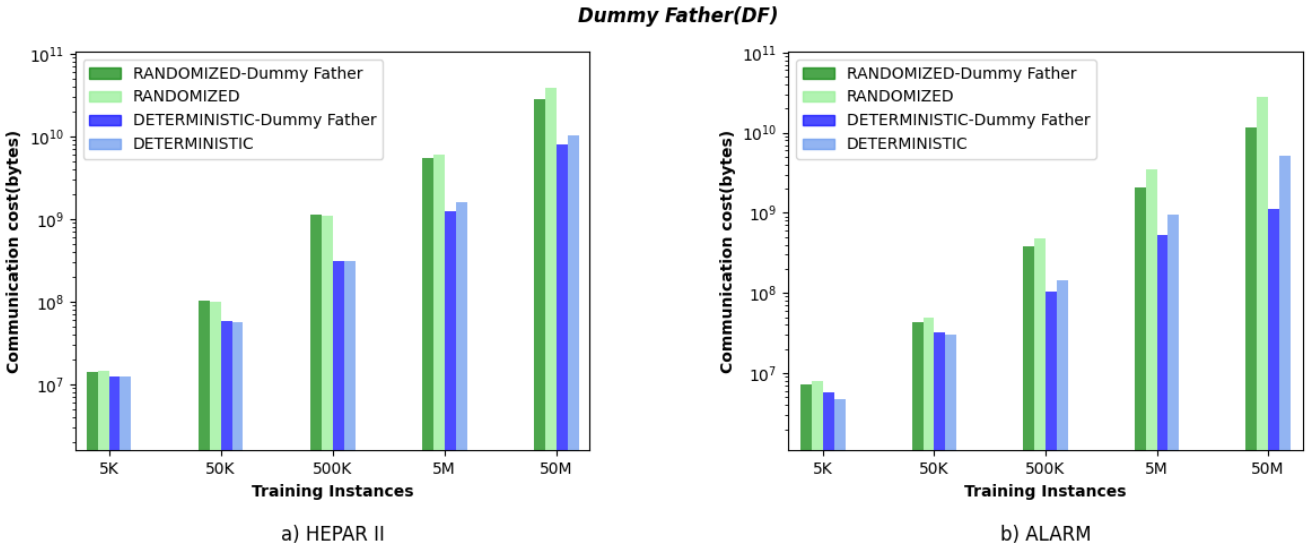


Εικόνα 32: Throughput(events/sec) σε σχέση με το αριθμό του παραλληλισμού για τις δύο προσεγγίσεις, για το dataset HEPAR II - Best results

6.3.5 Dummy Father

Τέλος παρακάτω παραθέτουμε τα αποτέλεσμα από την μέθοδο του *Dummy Father(DF)* σε συνδυασμό με τον αλγόριθμο *NON_UNIFORM*. Τα παρακάτω αποτέλεσμα όπως να αναφέραμε και προηγουμένως αφορούν αποκλειστικά την πρώτη προσέγγιση(*Βλ. Κεφάλαιο 4.2.1*).

Συγκεκριμένα μετράμε το *communication cost* του αλγορίθμου *NON_UNIFORM* σε συνδυασμό με την μέθοδο του *Dummy Father(DF)* και το συγκρίνουμε με το *communication cost* του αλγορίθμου *NON_UNIFORM*, χρησιμοποιώντας τόσο τους *RANDOMIZED* όσο και τους *DETERMINISTIC counters* της πρώτης προσέγγισης. Τα *datasets* που χρησιμοποιήσαμε κυμαίνονται από $5K$ έως $50M$ και αφορούν τα *dataset HEPAR II* και *ALARM* αντίστοιχα. Τέλος να επισημάνουμε ότι ο *άξονας y* βρίσκεται σε *λογαριθμική κλίμακα*.



Εικόνα 33: Communication cost σε συνδυασμό με την μέθοδο του Dummy Father(DF), για τα datasets HEPAR II,ALARM

Παρατηρούμε λοιπόν ότι όσο αυξάνεται ο αριθμός των *training instances* τόσο μεγαλύτερη αρχίζει να είναι και η διαφορά μεταξύ τους και αυτό οφείλεται στο γεγονός ότι τόσο λιγότερα μηνύματα απαιτούνται να σταλθούν για τον κάθε *counter*. Συγκεκριμένα έχουμε ένα ποσοστό μείωσης του *communication cost* της τάξης του 40% για τους *RANDOMIZED counters* ενώ ένα ποσοστό μείωσης του *communication cost* της τάξης του 50% για τους *DETERMINISTIC counters*. Να υπενθυμίσουμε και πάλι ότι η διαφορά στο *communication cost* οφείλεται καθαρά στην αποφυγή του να διατηρήσουμε τους *EXACT counters* στην περίπτωση κάποιου *orphan* κόμβου. Επομένως οι διαφορές ανά δίκτυο θα είναι διαφορετικές, συγκεκριμένα στην περίπτωση του *dataset ALARM* αποφεύγουμε *26 EXACT counters* ενώ στην περίπτωση του *dataset HEPAR II* αποφεύγουμε *18 EXACT counters*.

Κεφάλαιο 7:

Επίλογος

7.1 Συμπεράσματα

Συνοψίζοντας καταφέραμε να προτείνουμε μια διαφορετική προσέγγιση για την συνεχή εκμάθηση των παραμέτρων(*continuously learning parameters*) από ένα *Bayesian Network* πάνω στο *continuous distributed model* , χρησιμοποιώντας την μέθοδο του *FGM*. Συγκεκριμένα καταφέραμε να μειώσουμε το *communication cost* που απαιτείται συγκριτικά με την μέθοδο του *EXACTMLE* ενώ ταυτόχρονα μπορούμε να εγγυηθούμε *error guarantees* στην εκτίμηση *probabilities queries* πάνω στο *joint probability distribution* του *Bayesian Network* όπως επιβεβαιώνεται και από την πειραματική ανάλυση. Επιπλέον παρουσιάζουμε μια εμπεριστατωμένη ανάλυση της πρώτης προσέγγισης χρησιμοποιώντας τόσο *RANDOMIZED* όσο και *DETERMINISTIC approximate distributed counters* σε συνδυασμό πάντα με τους αλγορίθμους *BASELINE, UNIFORM* και *NON_UNIFORM*. Τέλος καταφέραμε να ενσωματώσουμε και τις δυο προσεγγίσεις σε ένα από τα πιο γνωστά *distributed streaming framework* , το *Apache Flink*.

7.2 Μελλοντικές Επεκτάσεις

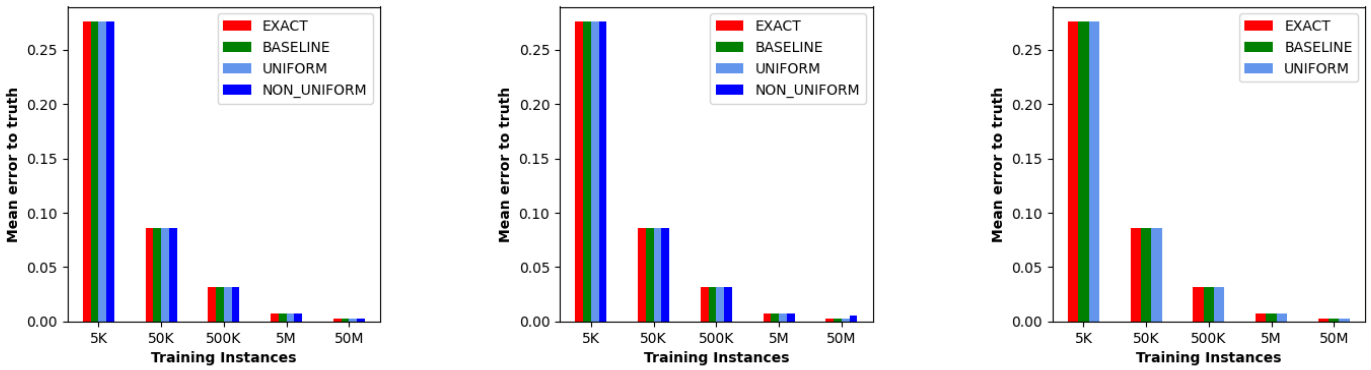
Υπάρχουν αρκετές επεκτάσεις που μπορούμε να δούμε. Μια ενδιαφέρον επέκταση που αφορά και τις δυο προσεγγίσεις είναι το *learning* από το *structure* του *Bayesian* Network (το οποίο το θεωρούσαμε δεδομένου καθόλη την διάρκεια της ανάλυσης) καθώς φτάνουν δεδομένα στα *sites* , δηλαδή το *learning structure* του *Bayesian Network* προσαρμοσμένο σε *distributed streaming* περιβάλλον. Επιπλέον μια ενδιαφέρον επέκταση όσον αφορά την δεύτερη προσέγγιση είναι πρώτον η αναζήτηση *safe functions* με σκοπό την περαιτέρω μείωση του *communication cost* και δεύτερον την προσαρμογή των *Graphical Model sketches* [32] με σκοπό και πάλι την μείωση του *communication cost* , συγκεκριμένα αντί για *frequency vectors* να χρησιμοποιούμε *graphical sketches* τα οποία θα αναπαριστούν με *compact* τρόπο ολόκληρο του *joint probability distribution*(*space efficient*) παρέχοντας ταυτόχρονα τα καταλληλά *error guarantees* που απαιτούνται.

Παράρτημα

Πειραματικά αποτελέσματα

Communication cost σε σχέση με τον αριθμό των training instances

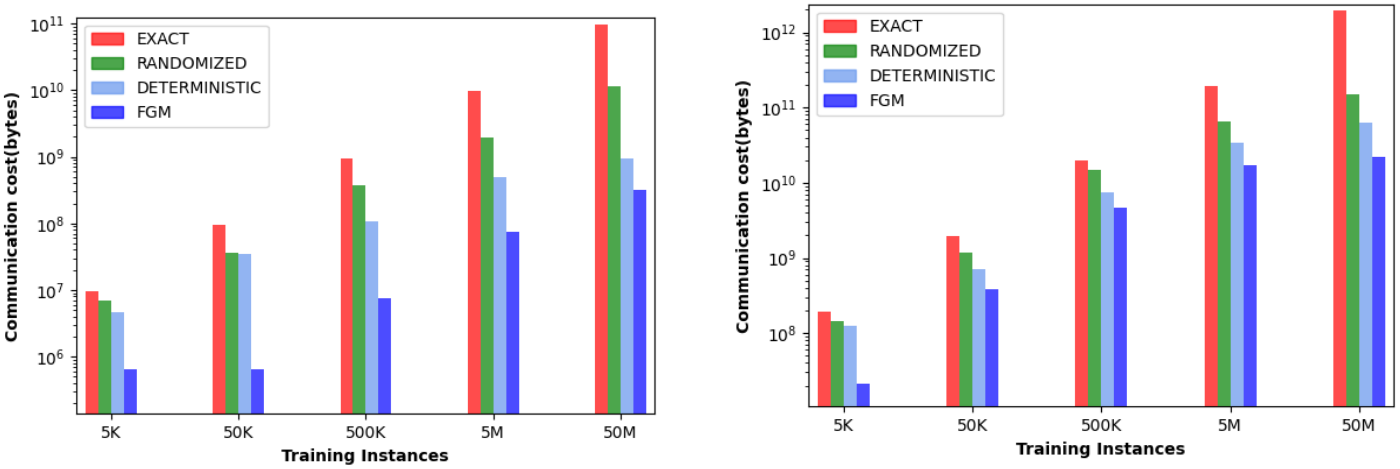
Παρακάτω βλέπουμε το *mean error to GT* για καθεμιά από τις δυο προσεγγίσεις. Να επισημάνουμε ότι διαφορές μεταξύ των αλγορίθμων *BASELINE,UNIFORM* και *NON_UNIFORM* δεν παρατηρούνται γιατί σε αυτήν την περίπτωση το *statistical error* είναι αρκετά μεγαλύτερο από το *approximation error* επομένως έχει και ως αποτέλεσμα το *statistical error* να είναι κυρίαρχο, επομένως δεν παρατηρούνται και διαφορές μεταξύ των αλγορίθμων. Επιπλέον το *error* μικραίνει καθώς αυξάνεται ο αριθμός των *training instances* αφού ταυτόχρονα μειώνεται το *statistical error*. Τέλος μπορούμε να παρατηρήσουμε ότι έχουμε αρκετά καλό *accuracy* και στις δυο προσεγγίσεις. Συγκεκριμένα για *datasets* μεγαλύτερα από $50K$ έχουμε λιγότερο *error* λιγότερο του 10% για κάθε προσέγγιση, το οποίο είναι αρκετά καλό δεδομένου και των *probability queries*.



a) RC - HEPAR II b) DC - HEPAR II c) FGM - HEPAR II

Εικόνα 34: Mean error to GT σε σχέση με τον αριθμό των training instances και για τις δυο προσεγγίσεις, για το dataset HEPAR II

Παρακάτω βλέπουμε το *communication cost* συναρτήσει των *training instances* και για τις δυο προσεγγίσεις για τα *datasets LINK* και *ALARM*. Για κάθε προσέγγιση μεταξύ των αλγορίθμων *BASELINE,UNIFORM* και *NON_UNIFORM* χρησιμοποιείται εκείνος ο αλγόριθμος που έχει το μικρότερο c*ommunication cost* για κάθε προσέγγιση στο κάθε δίκτυο.

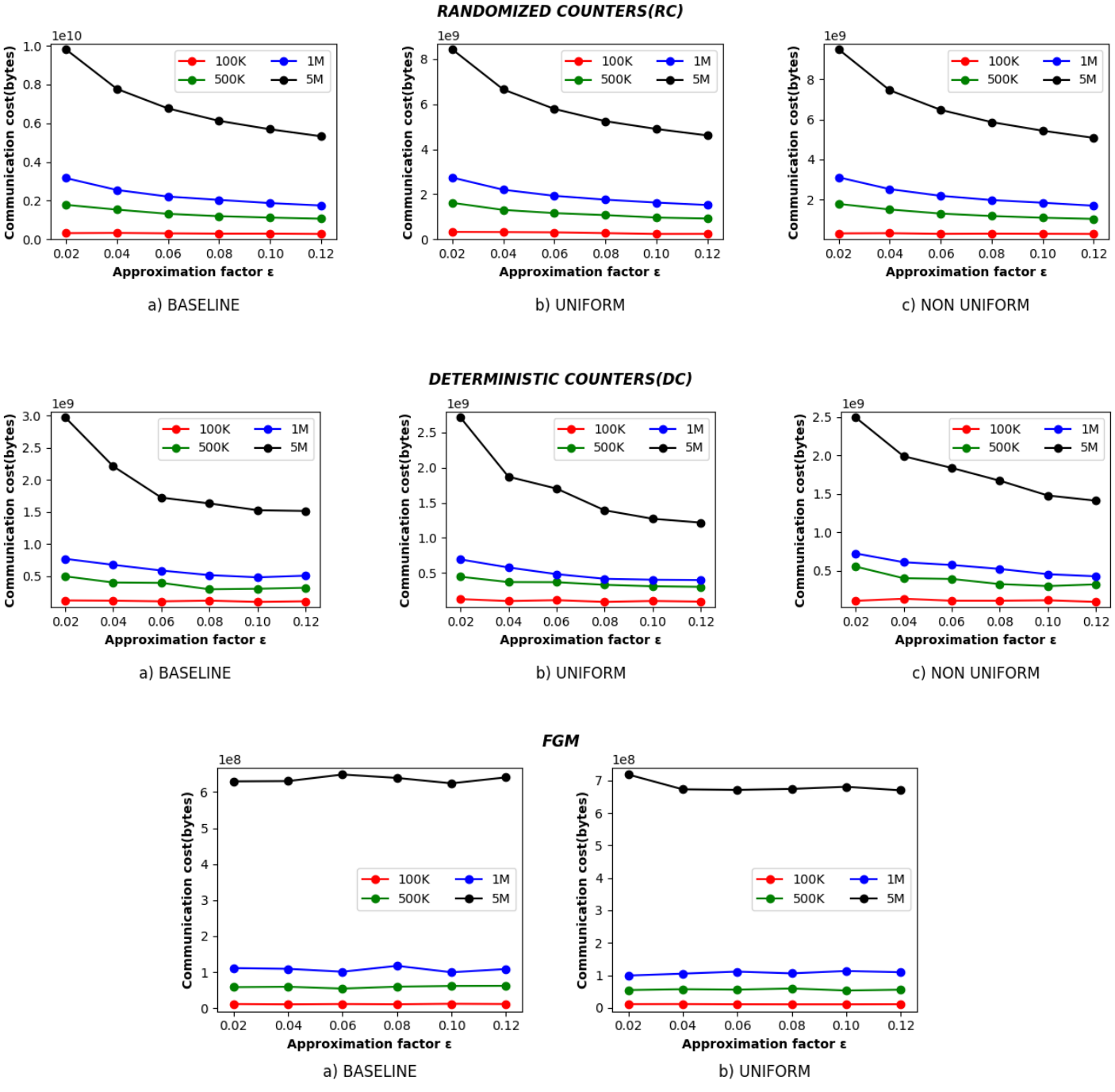


a) ALARM b) LINK

Εικόνα 35: Communication cost σε σχέση με τον αριθμό των training instances για τα datasets LINK,ALARM

Communication cost σε σχέση με το approximation factor ε

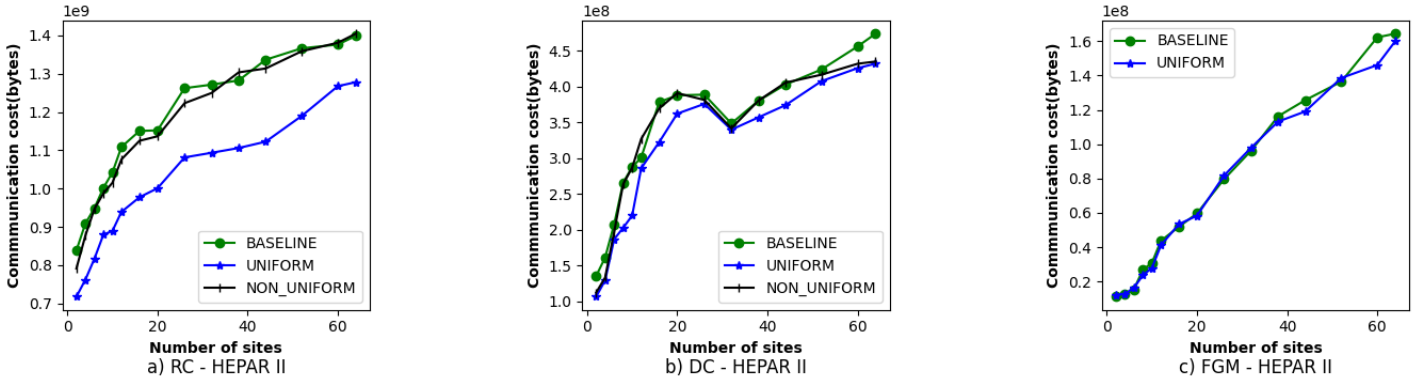
Παρακάτω βλέπουμε το *communication cost* συναρτήσει του *approximation factor ε* για όλους τους αλγορίθμους *BASELINE,UNIFORM* και *NON_UNIFORM* για κάθε προσέγγιση.



Εικόνα 36: Communication cost σε σχέση με το approximation factor ε για τις δύο προσεγγίσεις, για το dataset HEPAR II

Communication cost σε σχέση με τον αριθμό των workers

Παρακάτω βλέπουμε το *communication cost* συναρτήσει του αριθμού των *sites* για όλους τους αλγορίθμους *BASELINE,UNIFORM* και *NON_UNIFORM* για κάθε προσέγγιση.

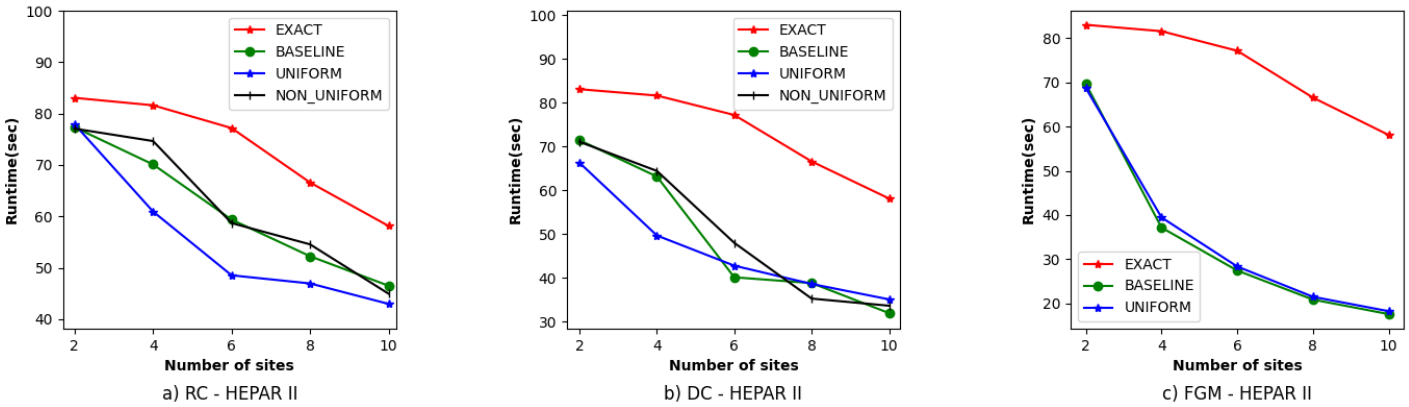


Εικόνα 37: Communication cost σε σχέση με το αριθμό των sites για τις δύο προσεγγίσεις, για το dataset HEPAR II

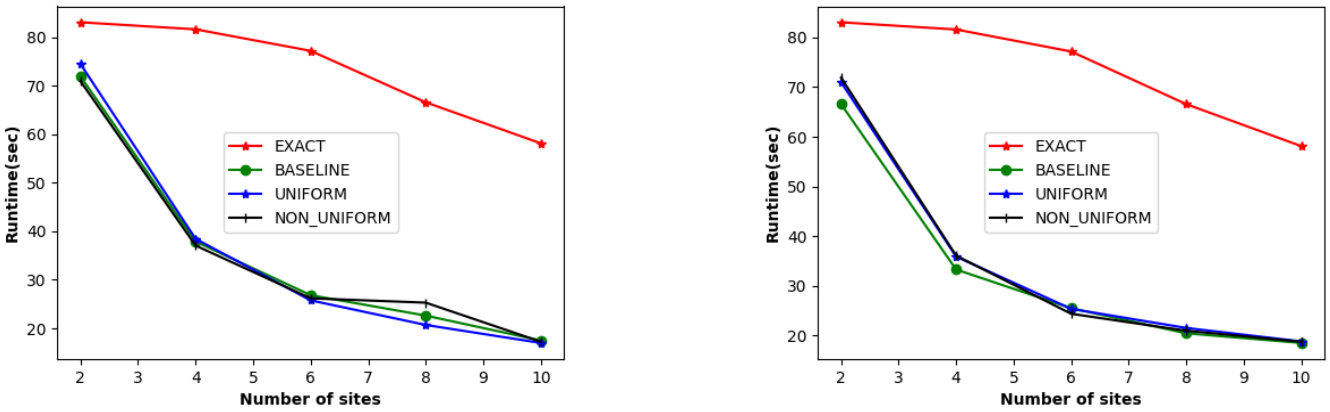
Scalability

Runtime σε σχέση με τον αριθμό των sites

Παρακάτω βλέπουμε το *runtime(sec)* συναρτήσει του αριθμού των *sites* για όλους τους αλγορίθμους *BASELINE,UNIFORM* και *NON_UNIFORM* για κάθε προσέγγιση.



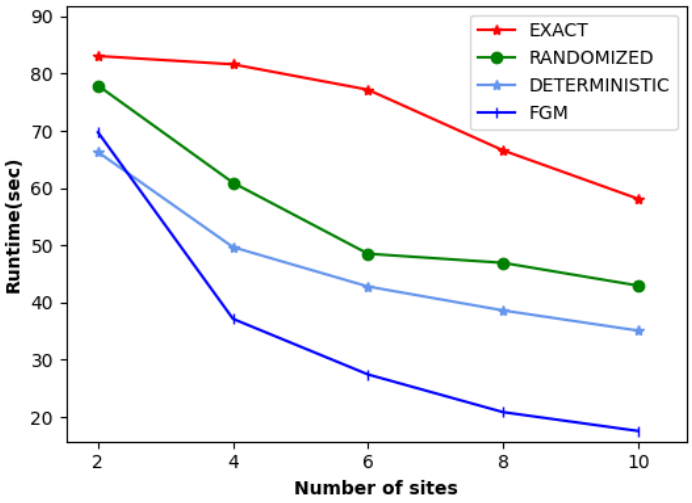
Εικόνα 38: Runtime(sec) σε σχέση με το αριθμό των sites για τις δύο προσεγγίσεις, για το dataset HEPAR II



a) RC - HEPAR II, after initial state

b) DC - HEPAR II, after initial state

Εικόνα 39: Runtime(sec) σε σχέση με το αριθμό των sites για την πρώτη προσέγγιση, για το dataset HEPAR II μετά το αρχικό στάδιο

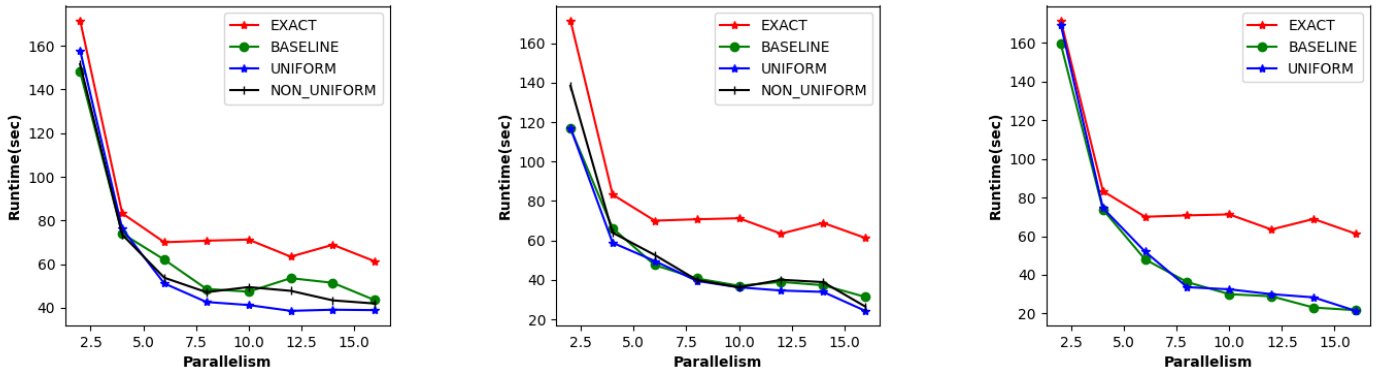


HEPAR II

Εικόνα 40: Runtime(sec) σε σχέση με το αριθμό των sites για τις δύο προσεγγίσεις, για το dataset HEPAR II-Best results

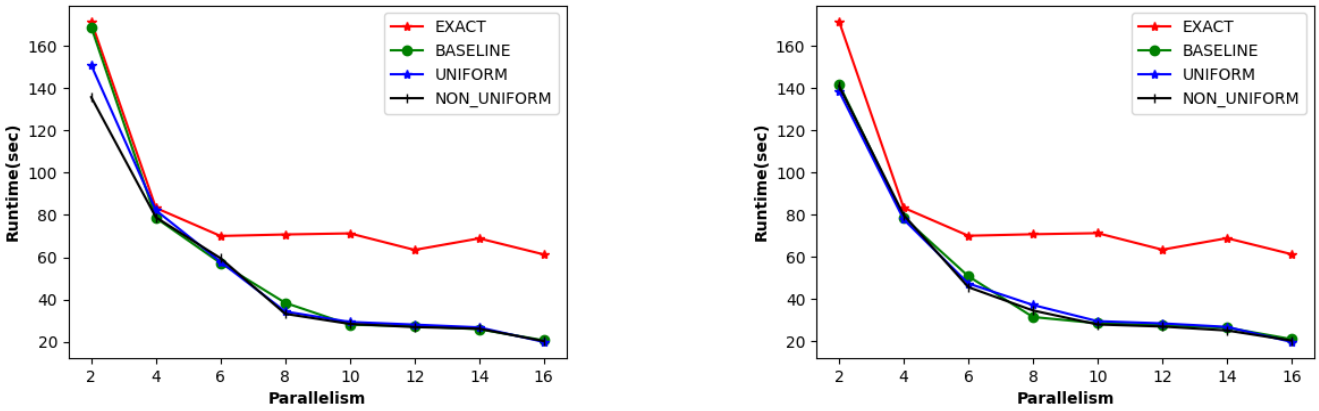
Runtime σε σχέση με τον αριθμό του παραλληλισμού

Παρακάτω βλέπουμε το *runtime(sec)* συναρτήσει του αριθμού του παραλληλισμού για όλους τους αλγορίθμους *BASELINE,UNIFORM* και *NON_UNIFORM* για κάθε προσέγγιση.



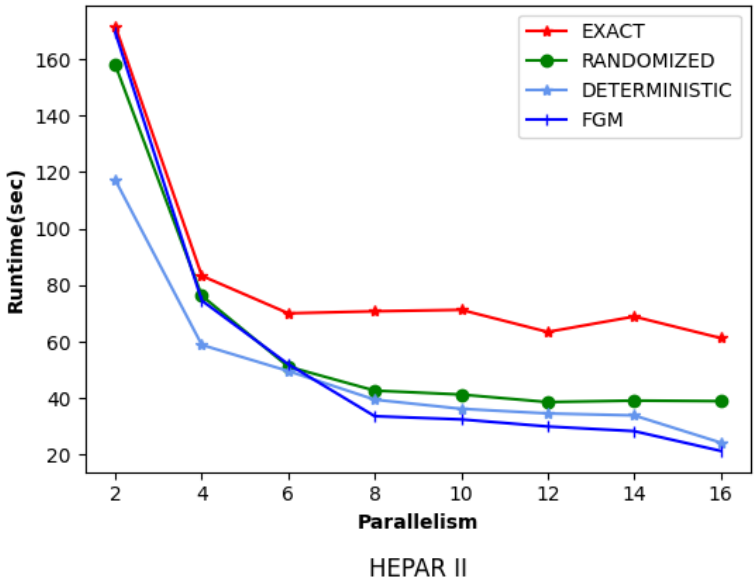
a) RC - HEPAR II b) DC - HEPAR II c) FGM - HEPAR II

Εικόνα 41: Runtime(sec) σε σχέση με το αριθμό του παραλληλισμού για τις δύο προσεγγίσεις, για το dataset HEPAR II



a) RC - HEPAR II, after initial state b) DC - HEPAR II, after initial state

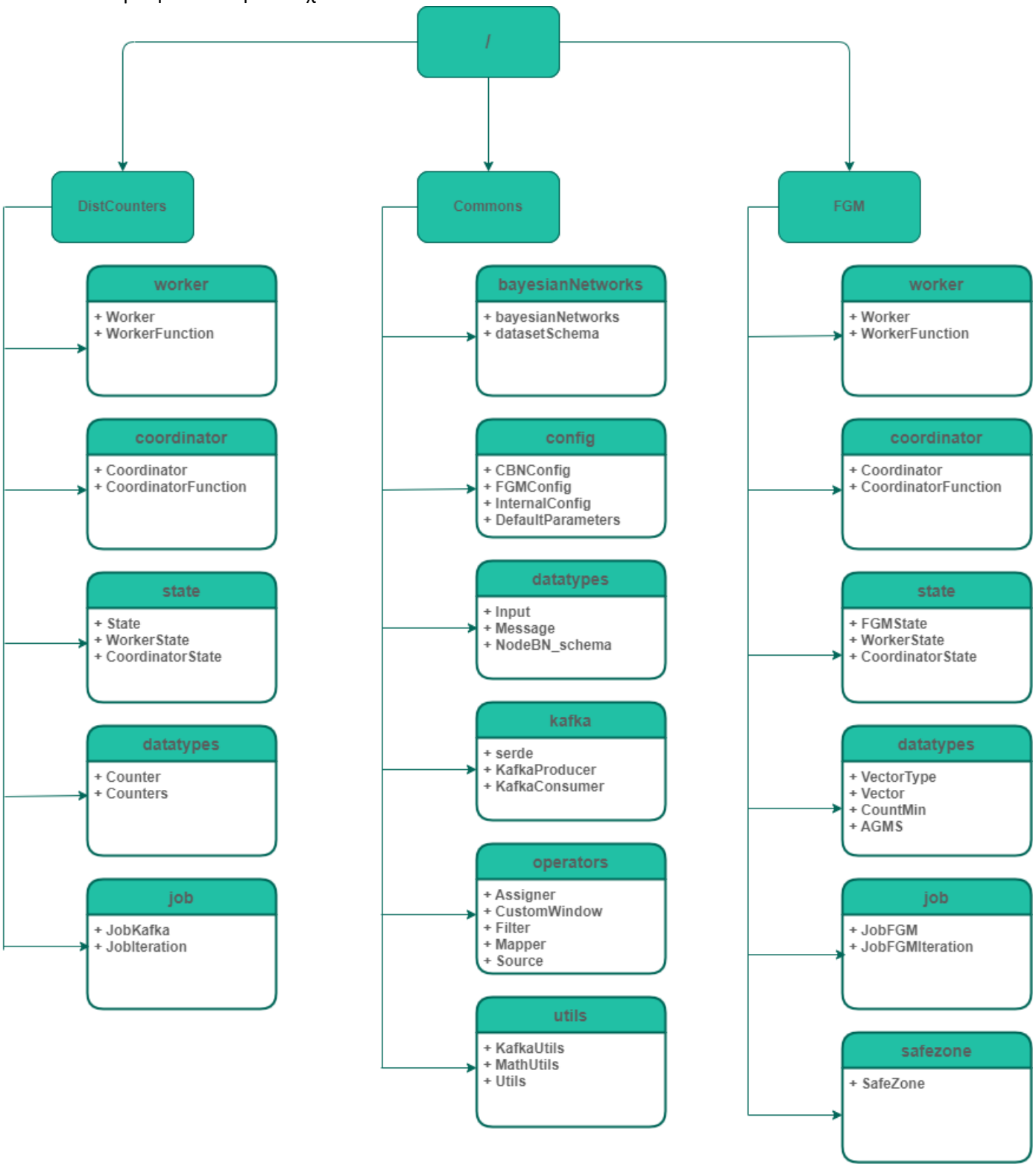
Εικόνα 42: Runtime(sec) σε σχέση με το αριθμό του παραλληλισμού για την πρώτη προσέγγιση, για το dataset HEPAR II μετά το αρχικό στάδιο



Εικόνα 43: Runtime(sec) σε σχέση με το αριθμό των sites για τις δύο προσεγγίσεις, για το dataset HEPAR II - Best results

Project Structure

Στην παρακάτω *εικόνα* παρουσιάζεται η γενικευμένη μορφή από την δομή του *project* που υλοποιήσαμε, δηλαδή το πως οργανώνεται σε επιμέρους πακέτα τα οποία αναλύονται περιληπτικά στην συνέχεια.



Εικόνα 44: Δομή και οργάνωση του project

Όπως βλέπουμε και στην εικόνα το structure του *project* χωρίζεται σε τρεις μεγάλες κατηγορίες , η πρώτη κατηγορία είναι το *DistCounters* η οποία όπως προκύπτει και από το όνομα αναφέρεται στα *Approximate Distributed Counters* , η δεύτερη κατηγορία αναφέρεται στην δεύτερη προσέγγιση που προτείναμε και αναφέρεται εξ ολοκλήρου στο *FGM* πρωτόκολλο. Τέλος η κατηγορία *Commons* αναφέρεται στα επιμέρους στοιχεία που χρησιμοποιούνται και από τις δυο προηγούμενες προσεγγίσεις. Στην συνέχεια αναλύουμε περιληπτικά τα πιο σημαντικά και επιμέρους στοιχεία από κάθε ενότητα.

DistCounters-FGM

- ***worker****:* Η κατηγορία αυτή αποτελείται από δυο επιμέρους στοιχεία. Το πρώτο αναφέρεται στο *Worker* το οποίο περιέχει την βασική λογική που πρέπει να έχει ο κάθε *Worker* δηλαδή παρέχει την λογική που απαιτείται ώστε ο *Worker* να μπορεί να διαχειριστεί τόσο τα *inputs*(*Input Source*) όσο και τα *control* μηνύματα που δέχεται από τον *Coordinator*(*Feedback Source*). Η δεύτερη κατηγορία *WorkerFunction* περιέχει όλη την λειτουργικότητα που βρίσκεται πίσω από τον κάθε *Worker* και καθορίζεται από την εκάστοτε προσέγγιση που χρησιμοποιούμε.
- ***coordinator****:* Η κατηγορία αυτή αποτελείται επίσης από δυο επιμέρους στοιχεία. Το πρώτο στοιχείο είναι ο *Coordinator* και αναφέρεται στην λογική που χρησιμοποιείται από αυτόν, συγκεκριμένα περιέχει την λογική που απαιτείται για την διαχείριση των μηνυμάτων από τους *Workers* όσο και την λογική που απαιτείται για την διαχείριση των *probability queries(Query Source)*.Το δεύτερο στοιχείο είναι το *CoordinatorFunction* που περιέχει όλη την λειτουργικότητα που βρίσκεται πίσω από την λογική του *Coordinator* και καθαρίζεται αναλόγως με την προσέγγιση που χρησιμοποιούμε.
- ***state:*** Σε αυτήν την κατηγορία υπάρχουν τρία επιμέρους στοιχεία αλλά σε αυτήν την περίπτωση υπάρχει μια ιεραρχική σχέση μεταξύ τους. Το στοιχείο *State* είναι αυτό που περιέχει όλα τα κοινά στοιχεία που χρειάζεται να περιέχονται και στις δύο πλευρές(*Worker-Coordinator*) και κληρονομούνται τόσο από το *WorkerState* όσο και από το *CoordinatorState*. Συγκεκριμένα το *State* περιέχει στοιχεία που αφορούν το *setup* ολόκληρου του *pipeline* και περιγράφονται αναλυτικά στην παρακάτω ενότητα. Το δεύτερο στοιχείο είναι το *WorkerState* που περιέχει όλα τα στοιχεία από το στοιχείο *State* και επιπλέον το *state* που απαιτείται από τον κάθε *Worker* , όπως για παράδειγμα στην περίπτωση των *approximate distributed counter* αντιστοιχεί στο σύνολο των *approximate distributed counters* που πρέπει να διατηρήσει ο κάθε *Worker* ενώ στην περίπτωση του *FGM* πρωτοκόλλου αναφέρεται στα *frequency vectors* που πρέπει να διατηρήσει ο κάθε *Worker*. Στην ίδια λογική κυμαίνεται και το *CoordinatorState* δηλαδή περιέχει τα στοιχεία από το *State* και επιπλέον το *state* που πρέπει να διατηρεί ο Coordinator το οποίο καθορίζεται από την προσέγγιση που χρησιμοποιούμε (*approximate distributed counters* ή *frequency vectors*).
- ***datatypes:*** Η κατηγορία αυτή περιέχει τα βασικά *datatypes* τα οποία χρησιμοποιούνται τόσο από το *WorkerState* όσο και από το *CoordinatorState*. Όσον αφορά την πρώτη προσέγγιση περιέχει το βασικό στοιχείο που χρησιμοποιείται εκατέρωθεν και από τις δυο πλευρές και δεν είναι άλλο από το *Counter* το οποίο ισοδυναμεί σε ένα *approximated distributed counter*. Αντίστοιχα όσον αφορά την δεύτερη προσέγγιση περιέχει το βασικό στοιχείο το οποίο δεν είναι άλλο από το *Vector* το οποίο αντιστοιχεί σε ένα *frequency vector*. Επιπλέον όπως είπαμε και προηγουμένως το *FGM* αποτελεί μια μέθοδο που είναι ανεξάρτητη του *monitoring*

προβλήματος και για αυτό τον λόγο υλοποιήσαμε ακόμα ένα βασικό *datatype* που δεν είναι άλλο από τα *sketches*. Συγκεκριμένα υλοποιήσαμε δυο βασικά και ευρέως γνωστά *sketches*, το πρώτο αναφέρεται στο *Fast-AGMS sketch* [33] ενώ το δεύτερο αναφέρεται στο *CountMin sketch* [34]. Τέλος τα δυο *sketches* μπορούν να χρησιμοποιηθούν για οποιοδήποτε πρόβλημα θέλουμε να κάνουμε *monitoring* σε συνδυασμό με το κατάλληλο *safe function*.

- ***safezone:*** Η κατηγορία αναφέρεται αποκλειστικά στο *FGM* και αναφέρεται στα *safe functions* που χρησιμοποιούνται από το *FGM*. Επομένως αναλόγως με το *monitoring* πρόβλημα το μόνο που χρειάζεται να κάνουμε είναι να ορίσουμε το κατάλληλο *safe function* που απαιτείται, συνδυάζοντας με το κατάλληλο *datatype* που μας παρέχει το σύστημα.
- ***job:*** Αυτή η κατηγορία περιέχει το τελικό *pipeline* που προκύπτει συνδυάζοντας όλα τα επιμέρους στοιχεία. Το πρώτο αφορά το *pipeline* που προκύπτει χρησιμοποιώντας ως *feedback loop* κάποιο *Kafka topic* ενώ το δεύτερο αφορά το *pipeline* που προκύπτει χρησιμοποιώντας ως *feedback loop* το *Iterative Stream operator* που μας παρέχει το *Apache Flink*.

Commons

- ***bayesianNetworks:*** Η κατηγορία αυτή περιέχει όλα τα διαθέσιμα *Bayesian Networks* του συστήματος και τα αντίστοιχα *schemas* από το *dataset*, δηλαδή το *format* που έχει το κάθε *input feature vector* του *Bayesian Network*. Επομένως σε περίπτωση προσθήκης κάποιου νέου *Bayesian Network* ή *Naïve Bayes Classifier* το μόνο που χρειάζεται να κάνουμε είναι να ορίσουμε το αντίστοιχο *bayesiaNetwork* σε συνδυασμό με το *datasetSchema*.
- ***config***: Η κατηγορία αυτή αποτελείται από τέσσερα επιμέρους στοιχεία. Το πρώτο στοιχείο είναι το *InternalConfig* και είναι αυτό που περιέχει τις από κοινού παραμέτρους που χρησιμοποιούνται και από τις δύο μεθόδους (*DistCounter - FGM*) και χρειάζεται να γίνουν *setup* για να καθοριστεί ολόκληρη η λειτουργεία του *pipeline*. Το δεύτερο στοιχείο είναι το *CBNConfig* και είναι αυτό που περιέχει όλες τις παραμέτρους που χρησιμοποιούνται από το *DistCounter* ενώ η άλλη κατηγορία είναι το *FGMConfig* και περιέχει όλες τις παραμέτρους που χρησιμοποιούνται από το *FGM* πρωτόκολλο. Όλες οι παράμετροι αναλύονται παρακάτω και συγκεκριμένα στην ενότητα *Project Setup*.
- ***datatypes:*** Σε αυτήν την κατηγορία ανήκουν τα βασικά *datatypes* που χρησιμοποιούνται από κοινού και από τις δύο μεθόδους. Συγκεκριμένα περιέχει το *format* που πρέπει να έχε το *Input Source* και το *format* που πρέπει να έχουν τα μηνύματα(*Message*) που ανταλλάσσονται μεταξύ του *Coordinator* και των *Workers*.

Project Setup

Αυτή η ενότητα περιέχει λεπτομέρειες για όλες τις διαθέσιμες παραμέτρους που χρειάζονται να οριστούν ώστε ο χρήστης να μπορέσει να τρέξει ένα παράδειγμα του συστήματος που υλοποιήσαμε.

Distributed Counters Configuration

Παράμετρος: *typeCounter*
Περιγραφή: Η παράμετρος αυτή καθορίζει τον τύπο του *approximate distributed counter* που θέλουμε να χρησιμοποιήσουμε, συγκεκριμένα μπορεί να αντιστοιχεί σε έναν από τους τέσσερις διαθέσιμους τύπος: *RANDOMIZED, DETERMINISTIC, CONITNUOUS* και *EXACT*.

FGM Configuration

Παράμετρος: *typeState*
Περιγραφή*:* Η παράμετρος αυτή ορίζει τον τύπο του *state* που θα χρησιμοποιηθεί και στις δύο πλευρές (*Workers-Coordinator*), συγκεκριμένα μπορεί να αντιστοιχεί σε έναν από τους τρεις διαθέσιμους τύπους *state*: *VECTOR, AGMS* και *COUNTMIN*.

Παράμετρος: *enableRebalancing*
Περιγραφή: Η παράμετρος αυτή ενεργοποιεί/απενεργοποιεί το μηχανισμό του *rebalancing* του *FGM* πρωτοκόλλου.

Παράμετρος: *width,depth*
Περιγραφή: Αυτές οι παράμετροι είναι έγκυρες μόνο όταν ο τύπος από το *state* που θα χρησιμοποιηθεί αντιστοιχεί σε έναν από τα διαθέσιμα *sketches*. Συγκεκριμένα ορίζουν το *width* και το *depth* από το *sketch*.

Common Configuration

Παράμετρος: *typeNetwork*
Περιγραφή: Αυτή η παράμετρος ορίζει τον τύπο του *network* που θα χρησιμοποιηθεί, συγκεκριμένα υπάρχουν δυο διαθέσιμοι τύποι: *BAYESIAN, NAÏVE*.

Παράμετρος: *BNSchema*
Περιγραφή: Αυτή η παράμετρος καθορίζει το *Bayesian Network* που θα χρησιμοποιήσουμε. Υπάρχουν αρκετές επιλογές από *Bayesian Networks* που έχουμε ήδη ενσωματώσει τα οποία είναι διαθέσιμα στο [29]. Παρακάτω αναφέρουμε μερικά από αυτά: *SACHS, ALARM, HEPAR2, LINK, MUNIN, EARTHQUAKE*.

Παράμετρος: *datasetSchema*
Περιγραφή: Αυτή η παράμετρος καθορίζει το *schema* από το *dataset* που θα χρησιμοποιηθεί. Χρησιμοποιείται το ίδιο ακριβώς όνομα με αυτό το οποίο ορίζει και το *schema* από το *Bayesian Network*. Παρακάτω παραθέτουμε μερικές από τις διαθέσιμες επιλογές: *SACHS, ALARM, HEPAR2, LINK, MUNIN, EARTHQUAKE* που χρησιμοποιούνται σε συνδυασμό με το ομώνυμο *Bayesian Network*.

Παράμετρος: *errorSetup*
Περιγραφή: Αυτή η παράμετρος καθορίζει τον αλγόριθμο που ορίζει με κατάλληλο τρόπο το *approximation factor ε*. Υπάρχουν τρεις διαθέσιμε επιλογές: *BASELINE, UNIFORM* και *NON_UNIFORM*.

Παράμετρος: *querySize*
Περιγραφή: Αυτή η παράμετρος καθορίζει τον αριθμό των *probability queries* που θα εκτιμηθούν από τον *Coordinator*.

Παράμετρος: *workers*
Περιγραφή: Αυτή η παράμετρος καθορίζει τον αριθμό από *workers/sites* (χωρίς να εμπεριέχεται ο *Coordinator*) που θα χρησιμοποιηθούν.

Παράμετρος: *parallelism*
Περιγραφή: Αυτή η παράμετρος καθορίζει τον αριθμό του παραλληλισμού δηλαδή των αριθμό των *subtasks* που θα χρησιμοποιήσει ο κάθε *operator* του *pipeline*.

Παράμετρος: *eps*
Περιγραφή: Αυτή η παράμετρος ορίζει το *approximation factor ε* που καθορίζει το *accuracy* που θέλουμε να έχουμε στα *estimated probabilities*.

Παράμετρος: *delta*
Περιγραφή: Αυτή η παράμετρος καθορίζει το *likelihood* από το *estimated probability*, δηλαδή ορίζει την τιμή του *approximation factor δ* και χρησιμοποιείται αποκλειστικά σε συνδυασμό με τον *RANDOMIZED counter*.

Παράμετρος: enableSmoothing
Περιγραφή: Αυτή η παράμετρος ενεργοποιεί/απενεργοποιεί την μέθοδο του *Laplace Smoothing* κατά την διαδικασία εκτίμησης των *probability queries*.

Παράμετρος: *inputTopic*
Περιγραφή: Αυτή η παράμετρος ορίζει το όνομα από το *Kafka topic* που θα χρησιμοποιηθεί ως *input* στους *Workers*.

Παράμετρος: *feedbackTopic*
Περιγραφή: Αυτή η παράμετρος ορίζει το όνομα από τα *Kafka topic* που θα χρησιμοποιηθεί ως *feedback* ανάμεσα στους *Workers* και τον *Coordinator*.

Παρακάτω βλέπουμε ένα παραδείγματα χρήσης των παραμέτρων για την προσέγγιση των *Distributed Counters*.

Το παρακάτω παράδειγμα αναφέρεται στο *dataset HEPAR II* χρησιμοποιώντας ως *inputTopic* το *sourceHEPAR2* και ως *feedbackTopic* το *feedbackHEPAR2*. Ο αριθμός των *workers* που θα χρησιμοποιηθεί ισούται με 8 ενώ ο παραλληλισμός θα ισούται με 4.Το *approximation factor* ε θα είναι ίσο με 0.1 ενώ το *approximate factor* δ θα ισούται με 0.25. Ο τύπος από το *approximate distributed counter* θα είναι ο *RANDOMIZED* σε συνδυασμό με τον αλγόριθμο *UNIFORM*. Τέλος ο αριθμός των *queries* ισούται με 1000 με το *queryTopic* να αντιστοιχεί στο *queryHEPAR2*.

--inputTopic sourceHEPAR2 --feedbackTopic feedbackHEPAR2 --workers 8 --parallelism 4 --eps 0.1 --delta 0.25 --errorSetup UNIFORM --typeCounter RANDOMIZED --queriesSize 1000 --bn HEPAR2 --datasetSchema HEPAR2

Η μόνη αλλαγή που απαιτείται ώστε το παραπάνω παράδειγμα να μπορεί να εφαρμοστεί και για την δεύτερη προσέγγιση δηλαδή για το *FGM* είναι το *typeCounter* να αντικατασταθεί με το *typeState* το οποίο συνοδεύεται με το κατάλληλο τύπο που μας παρέχει το σύστημα(π.χ. *VECTOR*).

Βιβλιογραφία

[1] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. Cambridge, MIT Press, 2009.

[2] J. Joyce, "Bayes' Theorem," *Stanf. Encycl. Philos.*, Jun. 2003, [Online]. Available:https://plato.stanford.edu/archives/spr2019/entries/bayes-theorem/

[3] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[4] O. Pourret, P. Naïm, and B. Marcot, *Bayesian Networks: A Practical Guide to Applications*. John Wiley & Sons, 2008.

[5] O. J. Mengshoel, A. Darwiche, and S. Uckun, "Sensor Validation using Bayesian Networks," Feb. 2008.

[6] S. Andreassen *et al.*, "MUNIN: an expert EMG assistant," in *Computer-Aided Electromyography and Expert Systems*, J. E. Desmedt, Ed. Pergamon Press, 1989, pp. 255–277.

[7] A. Onisko, M. J. Druzdzel, and H. Wasyluk, "A Bayesian Network Model for Diagnosis of Liver Disorders," Mar. 2003.

[8] P. Xie, J. H. Li, X. Ou, P. Liu, and R. Levy, "Using Bayesian networks for cyber security analysis," in *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*, Jun. 2010, pp. 211–220.

[9] J. Dougherty, R. Kohavi, and M. Sahami, "Supervised and Unsupervised Discretization of Continuous Features," 1995, pp. 194–202.

[10] P. Langley, W. Iba, and, and K. Thompson, "An analysis of Bayesian classifiers," in *Proceedings of the tenth national conference on Artificial intelligence*, San Jose, California, Apr. 1992, pp. 223–228.

[11] M. Ghazanfar and A. Prügel-Bennett, "An Improved Switching Hybrid Recommender System Using Naive Bayes Classifier and Collaborative Filtering," Apr. 2010.

[12] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian Network Classifiers," *Mach. Learn.*, vol. 29, no. 2, pp. 131–163, Nov. 1997.

[13] T. J. Watson, "An empirical study of the naive Bayes classifier," 2001.

[14] M.Henrion, "Propagating Uncertainty in Bayesian Networks by Probabilistic Logic Sampling," in *Machine Intelligence and Pattern Recognition*, vol. 5, J. F. Lemmer and L. N. Kanal, Eds. North-Holland, 1988, pp. 149–163.

[15] R. Fung and K.-C. Chang, "Weighing and Integrating Evidence for Stochastic Simulation in Bayesian Networks," 1990, vol. 10, pp. 209–219.

[16] G. Cormode, "The continuous distributed monitoring model," *ACM SIGMOD Rec.*, vol. 42, no. 1, pp. 5–14, May 2013.

[17] K. Yi and Q. Zhang, "Optimal Tracking of Distributed Heavy Hitters and Quantiles." arXiv, Nov. 30, 2008.

[18] Y. Zhang, S. Tirthapura, and G. Cormode, "Learning Graphical Models from a Distributed Stream," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, Paris, Apr. 2018, pp. 725–736.

[19] Z. Huang, K. Yi, and Q. Zhang, "Randomized Algorithms for Tracking Distributed Count, Frequencies, and Ranks." arXiv, Dec. 02,2011.Available:http://arxiv.org/abs/1108.3413

[20] G. Cormode, S. Muthukrishnan, and K. Yi, "Algorithms for distributed functional monitoring," *ACM Trans. Algorithms*, vol. 7, no. 2, pp. 1–20, Mar. 2011.

[21] V. Samoladas and M. Garofalakis, "Functional Geometric Monitoring for Distributed Streams," 2019, p. 12.

[22] I. Sharfman, A. Schuster, and D. Keren, “A geometric approach to monitoring threshold functions over distributed data streams,” *ACM Trans. Database Syst.*, vol. 32, no. 4, p. 23, Nov. 2007.
[23] “Apache Flink.” https://flink.apache.org/ (accessed Feb. 11, 2023).
[24] “Apache Spark.” https://spark.apache.org/ (accessed Feb. 10, 2023).
[25] A. Lazerson, I. Sharfman, D. Keren, A. Schuster, M. Garofalakis, and V. Samoladas, “Monitoring distributed streams using convex decompositions,” *Proc. VLDB Endow.*, vol. 8, no. 5, pp. 545–556, Jan. 2015.
[26] “Apache Kafka.” https://kafka.apache.org/ (accessed Feb. 11, 2023).
[27] “Nikolaos Tzimos.” https://github.com/NikolasTz (accessed Feb. 11, 2023).
[28] “Software Technology and Network Applications Laboratory | SoftNet.” https://www.softnet.tuc.gr/en/ (accessed Feb. 11, 2023).
[29] M. Scutari, “bnlearn – Bayesian Network Repository.” https://www.bnlearn.com/bnrepository/ (accessed Feb. 11, 2023).
[30] I. A. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper, “The ALARM Monitoring System: A Case Study with two Probabilistic Inference Techniques for Belief Networks,” in *AIME 89*, Berlin, Heidelberg, 1989, pp. 247–256.
[31] C. S. Jensen and A. Kong, “Blocking Gibbs Sampling for Linkage Analysis in Large Pedigrees with Many Loops,” *Am. J. Hum. Genet.*, vol. 65, no. 3, pp. 885–901, Sep. 1999.
[32] B. Kveton, H. Bui, M. Ghavamzadeh, G. Theocharous, S. Muthukrishnan, and S. Sun, “Graphical Model Sketch.” arXiv, Jul. 18, 2016.
[33] G. Cormode and M. Garofalakis, “Sketching Streams Through the Net: Distributed Approximate Query Tracking,” presented at the Very Large Data Bases Conference, Aug. 2005.
[34] G. Cormode and S. Muthukrishnan, “An Improved Data Stream Summary: The Count-Min Sketch and Its Applications,” Berlin, Heidelberg, 2004, vol. 2976, pp. 29–38.