Quantitatives Frequent Pattern Mining in drahtlosen Sensornetzen

Daniel Klan, Thomas Rohe, Kai-Uwe Sattler
FG Datenbanken & Informationssysteme, TU Ilmenau
{*first.last*}@tu-ilmenau.de

Abstract: In drahtlosen Sensornetzen ist die effiziente und möglichst sensorlokale Analyse der Daten ein wichtiger Ansatz zur Verlängerung der Batterielaufzeit der Sensoren. Neben den klassischen primitiven Analyseverfahren, wie zum Beispiel Filtern oder Aggregation, werden zunehmend auch komplexe Verfahren teilweise oder vollständig innerhalb der Grenzen der Sensornetze verarbeitet. Die vorliegende Arbeit widmet sich einem dieser Verfahren – der Erkennung von häufig auftretenden Mustern (Frequent Itemsets) über Attributen mit kontinuierlichen Werten. Schwerpunkt ist die teilweise Auslagerung des vorgestellten Verfahrens auf die Knoten des Sensornetzes.

1 Einleitung

Sensoren zum Erfassen und Überführen physischer oder chemischer Eigenschaften in auswertbare Größen finden sich heute in vielen Bereichen des täglichen Lebens. In den letzten Jahren sind insbesondere drahtlose Sensornetze (*wireless sensor network* - WSN) als Mittel zur großflächigen, weitestgehend autarken Überwachung von Regionen immer stärken in den Fokus von Industrie und Forschung getreten. Bei jedem der Knoten dieser Netzwerke handelt es für sich genommen um einen Computer mit stark beschränkten Ressourcen. Im Regelfall verfügen die einzelnen Sensorknoten über eine Sensorik zur Erfassung der Realweltdaten (zum Beispiel zum Messen von Luftfeuchtigkeit, Temperatur, etc.), einer leistungsschwachen CPU, einem oftmals stark beschränkten Speicher für Daten und Programme, einer Batterie zur Energieversorgung und einem drahtlosen, zumeist funkbasierten Kommunikationsmodul. Es hat sich gezeigt, dass eine Reduktion der Kommunikation zwischen den einzelnen Sensorknoten zu einer drastischen Reduktion der notwendigen Energie führt, was sich wiederum in längeren Lebenszeiten des Sensornetzes äußert. Aus diesem Grund versucht man zusehends Verarbeitungsprozesse auf die einzelnen Sensorknoten im Netzwerk (*In-Network Processing*) auszulagern.

Einmal von den Sensoren erzeugt, werden die Daten im Allgemeinen einer Vielzahl an Analysen unterzogen. Eines der interessantesten Probleme in diesem Zusammenhang ist das Finden von bisher nicht bekannten Zusammenhängen zwischen verschiedenen Sensormessgrößen oder -werten. Auf Grundlage dieser lassen sich anschließend zum Beispiel Regelmengen bestimmen, welche zur Prozessoptimierung eingesetzt werden können. Im Bereich der Gebäudeautomation können so zum Beispiel Heizperioden automatisch bestimmt werden und Verkehsleitsysteme können in Abhängigkeit von wiederkehrend auftretenden Stoßzeiten automatisch verkehrsflussleitende Regeln bestimmen.

Das Problem der Detektion solcher Assoziationsregeln findet sich in einer Vielzahl von Anwendungen wieder und zählt heute zu den klassischen Data-Mining-Aufgaben. Die Detektion der Regelmengen basiert dabei oftmals auf dem Problem der Erkennung von häufigen Mustern (*Frequent Pattern Mining*). Viele der in den letzten Jahren vorgestellten Lösungsverfahren setzen für die Analyse dabei kategorische Attribute voraus. Insbesondere im Kontext von Sensordaten ist dies eine Einschränkung, welche den tatsächlichen Gegebenheiten oftmals nicht genügt. Vielmehr handelt es sich bei den von den Sensoren bereitgestellten Größen um Daten über einem stetigen Wertebereich.

Die Übertragung existierender Frequent-Pattern-Mining-Verfahren auf solche Daten führt hierbei oftmals nicht zu den gewünschten Ergebnissen und der resultierende Berechnungsaufwand ist nicht vertretbar. Basierend auf dieser Feststellung wurden verschiedene Verfahren entwickelt, welche sowohl mit kategorischen Attributen als auch mit kontinuierlichen Attributen umgehen können. Die meisten der präsentierten Verfahren setzen dabei auf einen Apriori-Ansatz mit Kandidatengenerierung, welcher für das Finden von häufigen Mustern alle möglichen Kombinationen von Teilmustern prüft. Sowohl der verwendete Apriori-Ansatz, als auch das Prüfen aller möglicher Teilmuster disqualifizieren die vorgestellten Verfahren für eine Analyse über kontinuierlichen Datenströmen.

In der vorliegenden Arbeit wird ein neuer Ansatz für das Problem des quantitativen Frequent Pattern Mining über Datenströmen mit kontinuierlichen Wertebereichen präsentiert. Schwerpunkt ist dabei die Vorverarbeitung der von den Sensoren eines WSN erzeugten Daten auf den Sensorknoten selbst.

Zunächst wird ein kurzer Überblick über Vorarbeiten im Bereich des Frequent Pattern Mining gegeben. Anschließend werden in Abschnitt 3 für das weitere Verständnis notwendige Definitionen vorgenommen. In Abschnitt 4 folgt eine Beschreibung des Verfahrens für das Finden der häufigen Itemsets. Basierend auf diesem werden in Abschnitt 5 zwei Ansätze vorgestellt, welche ein Vorverarbeitung im Sensornetz zur effizienten Verarbeitung einsetzen. Das beschriebene Verfahren und die im Sensornetz verteilten Lösungen werden in Abschnitt 6 evaluiert. Der Beitrag schließt mit einer Zusammenfassung und einem Überblick über zukünftige Arbeiten.

2 Verwandte Arbeiten

Eines der ersten Verfahren zum Finden von häufigen Mustern war das von Agrawal et al. präsentierte *Apriori*-Verfahren [AIS93, AS94]. Die Grundlage des Ansatzes bildet die Annahme, dass sich häufige Muster ausschließlich aus häufigen Teilmustern zusammensetzen. Basierend auf dieser Heuristik erzeugt das Verfahren alle möglichen $k-Itemset$-Kandidaten aus der Menge aller $(k-1)-Itemset$-Kandidaten. Aufgrund der wiederholten Analyse der Daten handelt es sich hierbei um ein *Multi-Pass*-Verfahren, welches seine Anwendung insbesondere in der Verarbeitung von endlichen Datenmengen findet. Auf Basis des Apriori-Algorithmus wurde eine Vielzahl an weiteren Algorithmen entwickelt, welche unter anderem eine Steigerung der Performance oder die Adaption des Algorithmus an Datenströme zum Ziel hatten (z.B. [PCY95, FMMT96].

Han et. al präsentieren in [HPY00] mit dem FP-Tree (*frequent pattern tree*) einen Präfixbaum zur effizienten Speicherung häufiger Muster. Zusätzlich zu dieser Datenstruktur stellen sie mit dem FP-Growth-Algorithmus ein Verfahren zum Finden häufiger Muster auf Basis des FP-Tree vor. Im Gegensatz zum Apriori-Verfahren verzichtet das FP-Growth-Verfahren vollständig auf die Generierung von Kandidaten.

Frequent Pattern Mining über Datenströmen stellt eine besondere Herausforderung dar. Neben dem von Gianella in [GHRL03] beschriebenen FP-Stream existiert auch eine Reihe weiterer Lösungen. So verwendet der in [TAJL08] vorgestellte Compact Pattern Tree (CPT) gleitende Fenster (ähnlich dem hier vorgestellem FP^2-Stream und dem DSTree [LK06]). Die Effizienz des Verfahrens wird dabei durch eine zusätzliche Restrukturierungsphase erreicht. Während dieser Phase wird der CPT, in Abhängigkeit von der Itemset-Häufigkeiten, mit dem Ziel einer hohen Kompaktheit umsortiert. In [MM02] beschreiben die Autoren, neben dem Sticky-Sampling und dem Lossy-Counting mit BTS (*Buffer-Trie-SetGen*), ein Verfahren, welches die Häufigkeiten von Itemsets mittels einer Gitter-Struktur speichert.

Die erste Arbeit die sich mit dem Problem der quantitativen Frequent-Itemset-Mining beschäftigte stammt von Srikant und Aggrawal [SA96]. Das vorgestellte Verfahren basiert dabei im Wesentlichen auf einem Apriori-Verfahren, welches die partitionierten Wertebereiche der einzelnen Attribute in geeigneter Weise kombiniert. Die Autoren in [FMMT96] beschreiben ein Verfahren, welche Techniken aus der algorithmischen Geometrie verwenden, um die optimalen Intervallgrenzen zu finden. Der wesentliche Nachteil des vorgestellten Verfahrens liegt in der Beschränkung auf Regeln, welche nur aus 2 Items bestehen. Die Autoren in [AL03] betrachten quantitativ-kategorische Assoziationsregeln. Das Interessantheitsmaß der Autoren basiert dabei auf dem Mittelwert von Transaktionen. Eine Regel wird dann als signifikant angesehen, wenn sich der Mittelwert über eine Teilmenge an Transaktionen in Relation zur Menge aller Transaktionen, die nicht dieser Regel folgen, als signifikant herausstellt. Zur Berechnung des Signifikanzlevels ziehen sie dabei den Steiger-Z-Test heran. Als Null-Hypothese nehmen sie an, dass die Mittelwerte beider Teilmengen gleich sind. Wird diese Null-Hypothese abgelehnt (mit einer Konfidenz von 95%), so wird die gefundene Regel als signifikant unterschiedlich von der Restmenge der Regeln angenommen.

In der Literatur finden sich auch Ansätze, welche versuchen, das Problem des quantitativen Association Rule Mining mit nicht-statistischen Verfahren zu lösen. So verwenden die Autoren in [MAR02] zum Beispiel einen genetischen Algorithmus zur Problemlösung. Ein Individuum entspricht dabei einem k-Itemset, welches sich aus den Items und deren minimalen und maximalen Werten zusammensetzt. Mittels Crossover, Selektion und Mutation werden diese Items anschließend rekombiniert bzw. angepasst und deren Güte über eine Fitnessfunktion bestimmt. Einen ähnlichen Ansatz verfolgen die Autoren in [SAVCN07].

Einen ersten Ansatz zur In-Network-Detektion häufiger Muster wird in [Röm08] präsentiert. Für die Analyse nutzt der Autor dabei eine Datenstromversion des Apriori-Algorithmus (der Datenstrom wird in Batches zerlegt und diese werden anschließend getrennt von einander durch das Apriori-Verfahren ausgewertet). Das Verfahren setzt dabei auf die Analyse kategorischer Attribute.

3 Vorbetrachtungen

Zunächst wollen wir verschiedene Begrifflichkeiten einführen, welche für das Verständnis des im Weiteren präsentierten Verfahrens hilfreich sind.

Im Folgenden bezeichnet A_i ein Attribut im Datenstrom und y einen einzelnen Attributwert. $A_i\langle y\rangle$ entspreche den zum Attribut A_i gehörenden Wert y. Der Ausdruck $A_i(l,r)$ bezeichne ein Item über dem Attribut A_i. Das Item ist entweder ein *quantitatives Attribut* mit einem Wert im Intervall $[l,r]$, oder ein *kategorisches Attribut* mit dem Wert $l = r$. $\mathcal{I}$ bezeichne die Menge aller Items. Die Menge $X = \{A_1(l,r), ..., A_k(l',r')\} \subseteq \mathcal{I}$, repräsentiert ein Itemset (oder auch k-Itemset), wobei die A_i mit $1 \leq i \leq k$ paarweise disjunkt sind.

D bezeichne eine Menge von Transaktionen. Eine Transaktion $d_{\Delta t} \in D$ ist eine Menge von Attributwerten $A_i\langle y\rangle$ im Zeitintervall Δt. Jede Transaktion $d_{\Delta t} = \{A_i\langle y\rangle, ..., A_j\langle y'\rangle\}$ kann auf ein Itemset $\mathcal{I}$ abgebildet werden, d.h. für jeden Wert $A_i\langle y\rangle$ existiert ein Item $A_i(l,r)$ in $\mathcal{I}$ mit $y \in [l,r]$.

Die Häufigkeit $freq(X,D)$ eines Itemsets X entspricht der Anzahl der Transaktionen $d_{\Delta t} \in D$ im Zeitintervall Δt, die auf das Itemset abgebildet werden können. Der Support $supp(X,D)$ eines Itemsets ist definiert als der prozentuale Anteil an Transaktionen $d_{\Delta t} \in D$, welche auf das Itemset abgebildet werden können. Üblicherweise sind nur häufige Itemsets von Interesse. Ein Itemset wird als häufig bezeichnet, wenn dessen Häufigkeit einen vordefinierten Schwellwert $minsup$ überschreitet.

Signifikanz und Informationsdichte Ziel des quantitativen Frequent Itemset Mining ist das Finden von zusammen auftretenden Items, deren Informationsgehalt sich *signifikant* von dem aller anderen Items unterscheidet. Ein quantitatives Item wird genau dann als signifikant bezeichnet, wenn dessen Informationsdichte größer ist, als die Informationsdichte der alternativen Items. Die Informationsdichte eines Items $A_i(l,r)$ ist dabei wie folgt definiert,

$$density(A_i(l,r)) = \frac{freq(A_i(l,r))}{|l-r|} \tag{1}$$

wobei $|l-r|$ der Distanz zwischen den beiden Intervallgrenzen des Items entspricht. Entsprechend dieser Definition wird ein Itemset $\mathcal{I}$ genau dann als signifikant bezeichnet, wenn alle Items dieses Itemsets signifikant sind.

Generalisierung Ein Itemset $\hat{X}$ wird genau dann als *Generalisierung* eines Itemsets X bezeichnet, wenn $\hat{X}$ dieselben Attribute enthält wie X und es gilt:

$$\forall A_i(l,r) \in X \wedge A_i(l',r') \in \hat{X} : l' \leq l \leq r \leq r'$$

D.h. alle Transaktion, welche sich auf das Itemset X abbilden lassen, können ebenso auf das Itemset $\hat{X}$ abgebildet werden.

	A_1	A_2		A_1	A_2
t_1	22	100	t_6	21.5	0
t_2	22.5	-	t_7	21	-
t_3	-	100	t_8	-	100
t_4	22.5	-	t_9	22	-
t_5	21.5	0	t_{10}	22.5	100

Tabelle 1: Beispiel Transaktionen

itemset	$freq$	$supp$
$\{A_1(20, 21.5]\}$	3	0.3
$\{A_1(20, 23]\}$	8	0.8
$\{A_2(0, 50]\}$	4	0.4
$\{A_2(50, 100]\}$	4	0.4
$\{A_1(20, 21.5], A_2(0, 50]\}$	2	0.2
$\{A_1(20, 23], A_2(0, 50]\}$	2	0.2
$\{A_1(20, 23], A_2(50, 100]\}$	2	0.2

Tabelle 2: Beispiel Itemsets

Im Folgenden soll ein kurzes Beispiel die eben beschriebenen Zusammenhänge verdeutlichen.

Beispiel 1 Gegeben sei ein Datenstrom, welcher Werte zweier Attribute A_1 (zum Beispiel Temperatur) und A_2 (zum Beispiel Bewegung) enthält. Im Zeitintervall Δt wurden die in Tabelle 1 dargestellten 10 Transaktionen beobachtet. Seien weiterhin die folgenden Items gegeben:

$$A_1(20, 21.5], A_1(20, 23], A_2(0, 50] \text{ und } A_2(50, 100]$$

Basierend auf diesen Items lassen sich die in Tabelle 1 abgebildeten Transaktionen auf die in Tabelle 2 dargestellten Itemsets abbilden. Die Tabelle zeigt die entsprechenden Häufigkeiten und den Support, den die einzelnen Itemsets aufweisen.

Es lassen sich die folgenden Dichten für die Items ermitteln: $density(A_1(20, 21.5]) = 2$ und $density(A_1(20, 23]) \approx 2.7$. Bei dem einelementigen Itemset $\{A_1(20, 23]\}$ handelt es sich um eine Generalisierung des Itemsets $\{A_1(20, 21.5]\}$. ■

4 Basis FP^2-Stream

Der FP^2-Stream entspricht einem Präfixbaum (ähnlich dem FP-Tree von Han et al. [HPY00]), in welchem die häufigsten Itemsets in einer kompakten Form gespeichert werden. Eine Header-Tabelle enthält Verweise auf alle Itemsets, welche sich gegenwärtig im Präfixbaum befinden. Jeder Pfad von der Wurzel bis zu einem Knoten im Präfixbaum repräsentiert ein Itemset. Zusätzlich zu den Itemsetinformationen sind in jedem Knoten des Baumes die Häufigkeiten der durch den Pfad des Knotens repräsentierten Itemsets in den letzten k Zeiträumen vermerkt. Bei den Zeiträumen handelt es sich um gleitende Zeitfenster.

Weiterhin sind alle Knoten, welche das gleiche Item repräsentieren, untereinander über Zeiger verbunden. Der entsprechende Eintrag der Header-Tabelle verweist dabei auf das erste Element der so erzeugten Liste. Jeder Eintrag der Header-Tabelle enthält zusätzlich ein *Equi-Width*-Histogramm, welches einen approximativen Überblick über die Häufigkeitsverteilung der zuletzt eingefügten Werte in den Items gibt. Abbildung 1 zeigt einen Beispiel-FP^2-Stream, dem die in Tabelle 1 beschriebenen Transaktionen zugrunde liegen.

Im weiteren Verlauf dieses Abschnittes soll ein kurzer Überblick der für die Verwaltung des Präfixbaumes notwendigen Operationen gegeben werden. Eine ausführliche Beschrei-

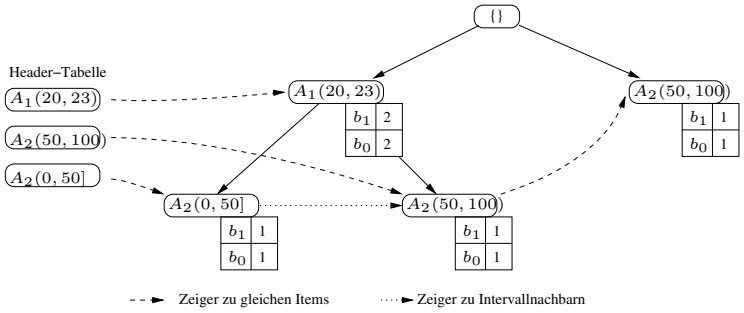


Abbildung 1: Beispiel FP^2-Stream

bung findet sich in [KRS10]. Die Verwaltung der einzelnen Items im FP^2-Stream folgt im wesentlichen einem *Top-Down*-Ansatz, d.h., die einzelnen Items (und somit die daraus gebildeten Itemsets) werden durch geeignete Split- und Merge-Operationen so lange verfeinert, bis sich Itemsets bilden, welche eine annähernd gleiche Informationsdichte aufweisen. Den generellen Ablauf des Verfahrens zeigt Algorithmus 1.

Einfügen neuer Transaktionen Das Einfügen neuer Transaktionen in den FP^2-Stream erfolgt batchweise. Ein Batch B bezeichnet dabei eine Menge von $|B|$ aufeinanderfolgenden Transaktionen. Neue Transaktionen werden nicht direkt in den FP^2-Stream integriert, sondern zuvor in einen sogenannten FP^2-Tree, bei dem es sich ebenfalls um einen Präfixbaum handelt, eingefügt. Der FP^2-Tree entspricht dabei in wesentlichen Teilen dem FP^2-Stream, wobei die Knoten jedoch lediglich die zum aktuellen Batch gehörenden Häufigkeiten verwalten.

Ähnlich wie beim FP-Stream [GHRL03] wird das Einfügen des ersten Batches B_0 in den FP^2-Stream getrennt von der Behandlung aller weiteren Batches betrachtet. Zunächst wird für jedes Attribut A_i des Datenstromes ein Item $A_i(min, max)$ angelegt, wobei min dem minimalen Wert von A_i in B_0 und max dem maximalen Wert von A_i in B_0 entspricht. Die Items werden anschließend absteigend nach der Häufigkeit ihres Auftretens im ersten Batch sortiert und in den FP^2-Tree eingefügt.

Alle weiteren Batches werden anschließend gleich behandelt. Jede Transaktion im Batch wird nach folgendem Schema zunächst in den FP^2-Tree eingefügt: Existiert bereits ein Knoten, welcher das Itemset repräsentiert, so wird die Frequenz des entsprechenden Knotens im FP^2-Tree um 1 inkrementiert. Existiert noch kein Knoten, welcher das Itemset repräsentiert, so muss ein neuer Knoten im FP^2-Tree angelegt werden.

Wurde die für den Batch vorgegebene Transaktionszahl erreicht, dann kann dieser in den FP^2-Stream integriert werden. Hierzu wird jedem Knoten im FP^2-Stream ein Eintrag für den neuen Batch im gleitenden Zeitfenster hinzugefügt. Sollten im FP^2-Stream Knoten vorhanden sein, welche durch die Integration des FP^2-Tree keine Aktualisierung erfahren, so sind deren Häufigkeiten für diesen Batch gleich 0. Sind während der Verarbeitung eines Batches neue Knoten im FP^2-Tree hinzugekommen, so müssen diese ebenfalls in den

Input : Menge von Transaktionen D
Output : Menge von häufigen Itemsets

```
erzeuge leeren FP²-Tree;
stelle für jedes Attribut A_i im Datenstrom dessen minimalen und maximalen Wert min und max fest und lege
  entsprechende Itemknoten im FP²-Tree an;
füge die Transaktionen von Batch B_0 in den FP²-Tree ein;
übernimm Knotengrenzen und Häufigkeiten aus dem FP²-Tree in den FP²-Stream;
while Batch B_i, i > 0 do
    initialisiere FP²-Tree mit den Knoten und Intervallgrenzen des FP²-Stream;
    sortiere alle Transaktionen des aktuellen Batch in den FP²-Tree ein (falls notwendig, füge neue Knoten
      hinzu bzw. erweitere existierende Knoten);
    übertrage alle Knoten aus dem FP²-Tree in den FP²-Stream;
    führe eventuell notwendige Split-Operationen auf dem FP²-Stream aus;
    führe eventuell notwendige Merge-Operationen auf dem FP²-Stream durch;
    Lies alle häufigen Itemsets aus dem FP²-Stream aus (FP²-Growth);
return häufige Itemsets;
```

Algorithmus 1 : Prinzipieller Ablauf FP^2-Stream

FP^2-Stream übernommen werden. Die Zeitfenster der neuen Knoten enthalten dabei nur den Häufigkeitswert des aktuellen Batches.

Wurden alle Transaktionen eines Batches in den FP^2-Stream eingefügt, dann wird geprüft, inwieweit sich Items verfeinern lassen. Ziel ist es dabei, eine möglichst hohe Informationsdichte pro Item zu erhalten, welche gleichmäßig über das ganze Item ist. Der Prozess der Verfeinerung ist hierbei ein zweistufiger Prozess. In einem ersten Schritt werden diejenigen Items verfeinert, deren Füllgrad zu hoch ist. Im zweiten werden Items, die sich generalisieren lassen, gemischt. Beide Prozesse werden im Weiteren genauer beschrieben.

Aufspaltung von Items Als wesentliches Kriterium für die Güte der Approximation eines Items wird die Häufigkeitsverteilung der ihm zugehörigen Attributwerte im letzten Batch herangezogen. Sind die in ein Item eingefügten Werte ungleichmäßig verteilt, so wurden die Grenzen für dieses Item schlecht gewählt. Der Grad der Ungleichverteilung wird über die Schiefe der in der Header-Tabelle mitgeführten *Equi-Width*-Histogramme bestimmt. Es existiert dabei für jeden Batch und jedes Item ein eigenes Histogramm.

Die Schiefe als Maß für die Güte eines Items $A_i(l, r)$ wird wie folgt bestimmt

$$skew(A_i(l,r)) = \frac{\max_{1 \leq j \leq d}\{hist(A_i(l,r),j)\} - \min_{1 \leq j \leq d}\{hist(A_i(l,r),j)\}}{\sum_{j=1}^{d} hist(A_i(l,r),j)},$$

wobei d die Anzahl der Buckets im Histogramm bezeichnet. $hist(A_i(l,r),j)$ bezeichnet die Häufigkeit im j-ten Bucket des Histogramms für das Item $A_i(l,r)$.

Überschreitet die Schiefe für ein gegebenes Item $A_i(l,r)$ einen vorab definierten Schwellwert $maxskew$, d.h. $skew(A_i(l,r)) > maxskew$, dann wird das Item in zwei Items mit disjunkten Intervallen gesplittet. O.B.d.A. werden im Folgenden Items immer in Items mit links offenem Intervall zerlegt.

Für den Teilungspunkt wird der Median über die in dem letzten Batch eingefügten Werte herangezogen. Hierzu wird der Median über die Buckets des Häufigkeitshistogramms bestimmt. Die Aufspaltung des Itemintervalls erfolgt anschließend auf Basis des Mittelpunktes des Median-Buckets. Das Ausgangsitem $A_i(l, r)$ wird dabei in die beiden Items $A_i(l, median/2]$ und $A_i(median/2, r)$ zerlegt, wobei $median/2$ den Mittelpunkt des Median-Buckets bezeichnet.

Bei einer Itemaufspaltung müssen alle Knoten im Baum, welche dieses Item repräsentieren, ebenfalls aufgespalten werden. Befinden sich unterhalb eines Knotens, welcher ein aufzuspaltenes Item repräsentiert, weitere Knoten, dann werden diese Teilbäume kopiert und als neue Teilbäume an die entstehenden Knoten angehangen. Die Häufigkeit des mit den aufgespalteten Knoten assoziierten Itemsets wird beim Überführen in die neuen Itemsets halbiert. Die genaue Verteilung der Daten in den einzelnen Knoten ist unbekannt. Daher wird zur Vereinfachung eine Gleichverteilung angenommen wird.

Zusätzlich zu den Knoten müssen auch deren Fenster angepasst werden. Jeder der beiden neu entstandenen Knoten erhält die Hälfte der Häufigkeitswerte des Original-Items. Um später bei der Itemextraktion auf einfache Weise das bisherige Aufspaltungsverhalten von Items nachvollziehen zu können, wird bei einer Aufspaltung in jedem Zeitfensterslot ein Verweis auf das Fenster angelegt, das beim Aufspalten abgeteilt wurde. Um weitere Aufspaltungen bzw. das Zusammenführen möglichst effizient gestalten zu können, verweist das letzte Fenster in der Liste auf das erste Listenelement, so dass ein Ring entsteht. Aufgrund der Verwendung von gleitenden Fenstern, wird mit jedem neu eingefügten Batch ein alter Batch aus den Fenstern verdrängt, so dass der damit verbundene Ring entfernt wird. Das kontinuierliche Verdrängen alter Werte hat zur Folge, dass sich die exakten Häufigkeitswerte in den verfeinerten Intervallen über die Zeit durchsetzen.

Verschmelzen von Items Durch das Aufspalten von Items wird der FP^2-Stream kontinuierlich vergrößert. So resultiert zum Beispiel der Split des Wurzelknotens in einer Verdoppelung aller Knoten in der Datenstruktur. Um den Präfixbaum möglichst kompakt zu halten, werden benachbarter Knoten mit nahezu gleicher Informationsdichte verbunden. Die beiden Items $A_i(l, r]$ und $A_i(l', r']$ werden genau dann zu einem neuen Item $A_i(l, r']$ zusammengefasst, wenn für alle Knoten im FP^2-Stream, die dieses Item repräsentieren, die folgenden Bedingungen erfüllt sind:

- Die Items $A_i(l, r]$ und $A_i(l', r']$ sind *direkte Nachbarn*, d.h. es gilt $r = l'$.
- Die Items $A_i(l, r]$ und $A_i(l', r']$ besitzen den gleichen Präfix, d.h. sie verfügen über den gleichen Elternknoten im FP^2-Stream.
- Die Informationsdichte des Items $A_i(l, r]$ entspricht nahezu der Informationsdichte des $A_i(l', r']$, d.h. $density(A_i(l, r]) \sim density(A_i(l', r'])$.

Die erste Bedingung stellt sicher, dass es sich bei den Verbundkandidaten auch wirklich um Intervallnachbarn handelt. Ausschließlich direkte Nachbarn können miteinander verbunden werden. Die zweite Bedingung garantiert, dass alle von der Item-Verschmelzung betroffenen Itemsets ebenfalls verbunden werden können. Die letzte Bedingung stellt sicher,

dass die zu verbindenden Items über einen nahezu gleichen Informationsgehalt verfügen. Es kommt damit beim Verschmelzen zu keinem Informationsverlust bzw. der Informationsverlust bewegt sich in einem vom Nutzer definierten Bereich.

Pruning und Reorganisation Weitere Möglichkeiten, die Datenstruktur kompakt zu halten, sind das Pruning und die Reorganisation der Datenstruktur. Beim Prunning werden nach [GHRL03] nicht notwendige Itemsets und Teilfenster aus der Datenstruktur entfernt. Der hier beschriebene Ansatz des FP^2-Stream verwendet gleitende Fenster, was ein kontinuierliches Pruning zur Folge hat.

Eines der wesentlichen Probleme des FP^2-Stream bzw. des original FP-Stream-Verfahrens ist die Abhängigkeit von den Häufigkeitswerten des ersten Batches. Die richtige Wahl hat einen entscheidenden Einfluss auf die Struktur des Präfixbaumes und damit die Performance des Verfahrens. Dieses Problem tritt insbesondere beim FP^2-Stream zu Tage. Jede Knotenverfeinerung hat die Verdoppelung der Anzahl seiner Kindknoten zur Folge. Wurde die initiale Struktur schlecht gewählt, so resultiert dies im schlimmsten Fall in einem exponentiellen Wachstum der Knotenanzahl. Entsprechend steigen Speicher und Wartungsaufwand der Datenstruktur. Eine kontinuierliche Überwachung und Re-Optimierung ist somit unumgänglich.

Ein einfacher Ansatz zur Optimierung der Datenstruktur ist die Extraktion aller Itemsets aus dem aktuellen FP^2-Stream, die Ordnung dieser nach ihren Häufigkeiten und das anschließende Neubefüllen eines leeren FP^2-Stream. In [TAJL08, TAJL09] wurde eine entsprechende Restrukturierungsphase beschrieben, welche das gleiche Problem für einen anderen Präfixbaum löst.

Rekonstruktion von Items und Itemset Extraktion Zwar werden im FP^2-Stream häufige Itemsets in einer kompakten Darstellung gespeichert, es wird aber dadurch nicht garantiert, dass diese auch effizient extrahiert werden können. Han et al. präsentieren in [HPY00] mit dem FP-Growth ein Verfahren zur effizienten Extraktion aller im FP-Tree gespeicherten häufigen Muster. Im Folgenden wird ein für den FP^2-Stream angepasstes Growth-Verfahren zum Extrahieren der häufigen Muster beschrieben.

Bevor die häufigen Itemsets aus dem FP^2-Stream extrahiert werden können, sind zwei Vorverarbeitungsschritte durchzuführen. In einem ersten Schritt wird aus dem FP^2-Stream ein FP^2-Tree extrahiert. Aufgrund des kontinuierlichen Teilens von Knoten und dem damit verbundenen Aufteilen von Häufigkeitswerten, handelt es sich bei den Häufigkeiten in den Knoten des FP^2-Stream zumeist nur um approximierte Werte. Lediglich Knoten, welche zuvor nicht geteilt wurden, weisen genaue Häufigkeitswerte auf. In der gegenwärtigen Umsetzung des FP^2-Stream werden nur Items mit exakten Häufigkeitswerten in den FP^2-Tree integriert. Aufgeteilte Knoten mit approximierten Werten werden so lange zusammengefasst bis ein Knoten entsteht, welcher genaue Häufigkeiten enthält.

Der durch den Extraktionsprozess erzeugte FP^2-Tree enthält im Allgemeinen quantitative Items, deren Support nicht dem geforderten minimalen Support $minsup$ genügt. In einem nächsten Schritt werden daher durch Rekombination benachbarter Intervalle Items er-

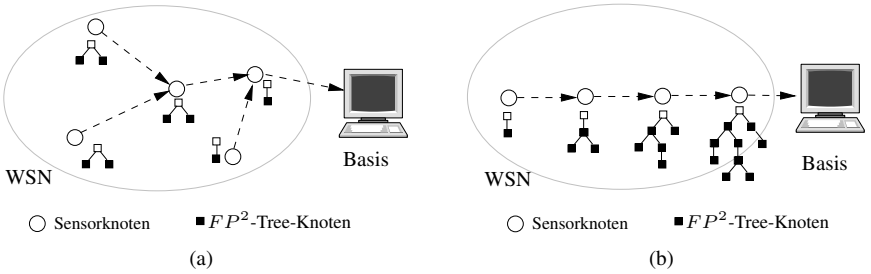


Abbildung 2: Verteilte Verarbeitung des FP^2-Stream: (a) batchweise Verarbeitung und (b) Präfixbaum-batchweise Verarbeitung

zeugt, welche dem minimalen Support genügen. Während der Rekombinationsphase wird das Item mit der höchsten Informationsdichte so lange mit benachbarten Items (d.h., dem jeweils direkt benachbarten Item mit der höchsten Informationsdichte) verbunden, bis der geforderte minimale Support erreicht wird. Soll mehr als ein häufiges Item pro Datenstromattribut ermittelt werden, so wird das beschriebene Verfahren auf die verbliebenen nicht-häufigen Items angewandt.

Nachdem der FP^2-Tree erzeugt wurde, dessen Items den definierten Bedingungen genügen, können aus diesem häufige Itemsets mit den oben erwähnten Verfahren extrahiert werden. Aus Gründen der effizienten Auswertung wird hierzu beim FP^2-Stream auf das *Top-Down*-Verfahren nach [WTHL02] zurückgegriffen.

5 Verteilter FP^2-Stream

Basierend auf dem eben vorgestellten Basisverfahren sollen im Weiteren zwei angepasste Verfahren vorgestellt werden, welche in Teilen in einem WSN bearbeitet werden können. Die beiden präsenierten Verfahren nutzen dabei konsequent die batchweise Verarbeitung des Verfahrens aus. Zudem verbleiben die Operationen zur Reorganisation der Datenstruktur vollständig auf der zentralen Instanz.

Batchweise Verarbeitung Die Auswertung der im FP^2-Stream gespeicherten Häufigkeiten erfolgt immer erst am Ende eines Batches. Während der Einfügephase wird die Datenstruktur lediglich mit den neu eintreffenden Werten befüllt. Die Daten eines einzelnen Batches müssen erst zum Ende des Batches auf der zentralen Instanz zur Verfügung stehen. Das nachstehende Verfahren macht von dieser Eigenschaft Gebrauch.

Jeder Sensorknoten verwaltet den Teil des FP^2-Tree, den er selbst mit Daten befüllt. M.a.W. auf einem Sensorknoten liegen genau diejenigen Knoten, welche von diesem mit Werten versorgt werden. Zusätzlich liegen noch die Histogramme der Header-Tabelle der betroffenen Items verteilt im Netzwerk vor. Ein Histogramm befindet sich genau dann auf einem Sensorknoten, wenn dieser die zum Histogramm gehörenden Items verwaltet.

Damit der zeitliche Zusammenhang zwischen den Messwerten der Sensoren nicht verloren geht, müssen die Messwerte am Ende eines Batches an die zentrale Instanz weitergeleitet werden. Dadurch, dass die Sensoren wissen, welche Items es im FP^2-Stream gibt, können sie die Messwerte in Form der von ihnen verwalteten Knoten versenden. Zusätzlich zu diesen Informationen werden am Ende eines Batches die in den Sensorknoten verwalteten Histogramme ausgewertet. Lediglich die Schiefe der Histogramme sowie der Median werden an die zentrale Instanz übermittelt (siehe Abbildung 2(a)). Der Transfer der Daten am Ende eines Batches erfolgt dabei auf Basis des vom Sensorknotenbetriebssystems zur Verfügung gestellten Ad-Hoc-Netzwerkes. Ein Knotenausfall führt zum Verlust der Informationen des betroffenen Knotens. Insofern möglich Routen die im Netzwerk verbleibenden Knoten die Ergebnisse über alternative Pfade zum Basisknoten.

Die zentrale Instanz fügt die aus dem WSN erhaltenen Batches in den zentral verwalteten FP^2-Stream ein. Anschließend folgen die Schritte der Item-Verfeinerung und der Itemsetextraktion. Eventuelle Veränderungen an den Itemgrenzen müssen anschließend durch die zentrale Instanz an die Sensorknoten propagiert werden.

Präfixbaum-batchweise Verarbeitung Im eben beschriebenen Ansatz sind für das Update der zentralen Datenstruktur am Ende eines jeden Batches, mindestens $h \cdot n$ Nachrichten notwendig (h bezeichnet die mittlere Hop-Entfernung der Knoten von der Basisstation und n die Anzahl der Knoten im WSN). Der folgende Ansatz, welcher eine zusätzliche Aggregation der Item-Information entlang einer Kettenstruktur, ähnlich dem *chained-based aggregation* Ansatz in [LRS02] vornimmt, soll diesem Umstand entgegen wirken.

Bei diesem Ansatz verwaltet jeder Knoten im WSN einen Teilbaum des FP^2-Tree. Der von der Basisstation am weitesten entfernte Knoten verwaltet die Wurzel und alle von ihm bedienten Items des FP^2-Tree. Der nächste Knoten in Richtung Basisstation verwaltet eine vollständige Kopie der Datenstruktur seines Vorgängers in der Kette. Zusätzlich beinhaltet der von ihm gespeicherte Teil-FP^2-Tree noch alle Knoten, welche durch das Hinzufügen seiner eigenen Daten erzeugt werden können. Dieser sukzessive Aufbau der Datenstruktur setzt sich in Richtung Basisstation fort. Der letzte Knoten in der Kette, welcher zum Ergebnis beiträgt, verwaltet somit eine vollständige Kopie des FP^2-Tree.

Am Ende eines Batches werden die Änderungen, beginnend vom letzten Element der Kette, in Richtung Basisstation propagiert. Die Updates werden dabei von einem zum anderen Knoten in der Kette weitergereicht, wobei jeder Knoten zunächst mit Hilfe der Zwischenergebnisse seine lokale Kopie des FP^2-Tree anpasst. Anschließend werden die Änderungen des Knotens und seiner Vorgänger in aggregierter Form an den Nachfolgeknoten weitergereicht. Dieser passt die Datenstruktur seinerseits an usw. (Abbildung 2(b)). Die Auswertung des vollständigen Baumes und dessen Anpassungen werden am Batchende auf der zentralen Instanz vorgenommen.

Bei der verwendeten Kettenstruktur handelt es sich um eine zusätzliche logische Ebene, welcher nicht zwangsläufig mit der tatsächlichen physischen Routingstruktur übereinstimmen muss. So kann es sein, dass zwischen zwei Knoten der logischen Ebene eine Vielzahl an Knoten auf der physischen Ebene liegen, welche für den Transfer verantwortlich sind. Im Fall eines Knotenausfalls auf der physischen Ebene kann dies

durch alternative Pfade eventuell kompensiert werden. Ein Knotenausfall auf der logischen Ebene hat die selben Folgen, wie bei anderen Kettenbasierten Ansätzen. Es kommt zum vollständigen Verlust der Informationen hinter dem ausgefallen Knoten.

Beim eben vorgestellten Ansatz werden zwischen zwei Sensorknoten im Mittel mehr Daten ausgetauscht als im vorhergegangenen Fall. Diese werden aber lediglich zum direkten Nachfolger gesendet werden. Somit entfällt das aufwendige Routing zur zentralen Instanz. In Abhängigkeit von der zugrundeliegenden Topologie (für die batchweise Verarbeitung) und den FP^2-Tree-Eigenschaften (Anzahl Attribute und Knoten je Attribut) sollte dieser Ansatz energetisch günstiger sein als die zuvor vorgestellte Lösung.

Neben diesen beiden teilweise im WSN verteilten Lösungen existiert noch der triviale Ansatz. Bei diesem erfassen die Sensoren lediglich ihre Werte und senden diese an die zentrale Instanz, welche im Anschluss für die weitere Verarbeitung verantwortlich ist.

6 Evaluierung

Im folgenden Abschnitt sollen zwei wesentliche Aspekte der vorgestellten Lösungen gezeigt werden:

1. Das vorgestellte FP^2-Stream-Verfahren ist zur Detektion von häufigen Mustern über quantitativen Attributen geeignet und
2. die präsentierten verteilten Ansätze können gegenüber einer rein auf der Basisstation stattfindenden Verarbeitung zu einer Reduktion des Energieverbrauches in WSN führen.

Die Evaluierung erfolgt auf Basis des Systems *AnduIN* [KHKS10]. Dieses stellt sowohl eine zentrale Optimierungs- und Verarbeitungskomponente, als auch einen In-Network-Komponente zur Verfügung. Im nachfolgenden Abschnitt sollen die wesentlichen Eigenschaften des verwendeten Systems kurz vorgestellt werden.

Anschließend folgt eine kurze Evaluierung des präsentierten Basisverfahrens. Eine ausführliche Evaluierung findet sich in [KRS10]. Abschließend wird in diesem Abschnitt auf die beiden verteilten Verfahren eingegangen. Hier steht insbesondere der Vergleich zur trivialen Lösung einer vollständig zentralen Verarbeitung im Mittelpunkt.

AnduIN Ziel des Projekts *AnduIN* ist eine Verknüpfung der Vorteile eines In-Network-Query-Prozessors (INQP) (Energieeffizienz) mit denen eines Datenstrom-Managementsystems (DSMS) (Unterstützung komplexer Datenstromoperatoren). Im Folgenden sollen die wesentlichen Teile des Systems kurz vorgestellt werden.

Im Gegensatz zu monolitischen INQP-Lösungen, wie zum Beispiel TinyDB [MFHH05], wird die Laufzeitumgebung der Sensorknoten bei *AnduIN* abhängig von einer Menge konkreter Anfragen entwickelt und anschließend im WSN verteilt. Dieser modulare Aufbau erlaubt die Integration prinzipiell beliebiger Funktionalität in die Laufzeitumgebung,

sofern die auf den Sensorknoten zur Verfügung stehenden Ressourcen (zum Beispiel Speicherbedarf) nicht überschritten werden. Der INQP-Teil von Anfragen bzw. die notwendigen, miteinander verknüpften Operatoren werden in den INQP-Anfragepool der Laufzeitumgebung integriert. Die Laufzeitumgebung kann anschließend (manuell oder automatisch) auf die Sensorknoten übertragen werden.

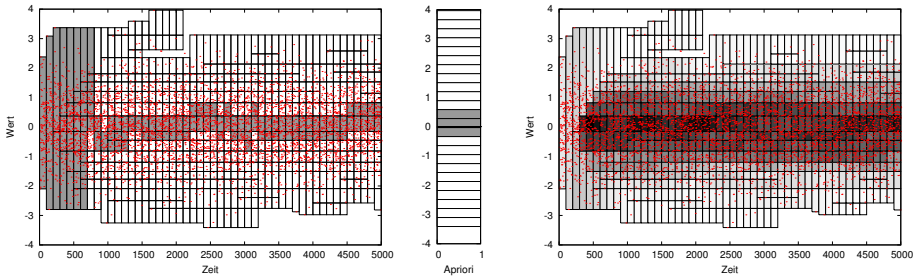
Der DSMS-Teil des System enthält alle wesentlichen Verbeitungskomponenten für die initiale Anfrageverarbeitung. Vom Anwender definierte Anfragen werden von der DSMS-Komponente analysiert, optimiert und anschließend bei Bedarf für die weitere Verarbeitung partitioniert. Die Arbeitsweise der Anfrageanalyse entspricht dabei zum großen Teil der von traditionellen DBMS. Der wesentliche Unterschied findet sich in der Anfrageoptimierung. Hier hat der Optimierer bei der Überführung logischer Operatoren in ihre physischen Äquivalente die Möglichkeit, zwischen Operatoren zu wählen, welche ausschließlich lokal von der DSMS-Komponente ausgeführt werden und solchen, welche partiell oder vollständig vom INQP bearbeitet werden. Entsprechende Kostenmodelle für die Wahl des besten Ausführungsplanes wurden ebenfalls integriert.

Enthält der ausgewählte Ausführungsplan Teile, welche von der INQP-Komponente bearbeitet werden sollen, dann werden die entsprechenden Operatoren identifiziert und anschließend aus dem Anfrageplan herausgelöst werden. Der physische Anfrageplan wird also in einen im DSMS und einen vom INQP auszuführenden Teil zerlegt. Der für den INQP bestimmte Teil des Anfrageplanes wird dabei in eine XML-Spezifikation überführt und anschließend an einen Codegenerator übergegeben. Dieser übernimmt das finale Erzeugen der Sensorknoten-Laufzeitumgebung. Der vom DSMS zu bearbeitende Teil wird an den zentralen Anfragepool weitergeleitet.

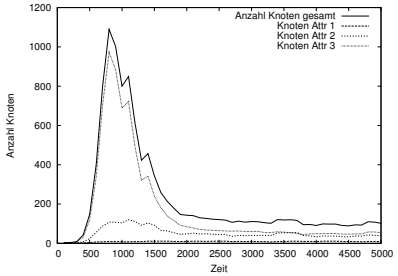
Im Fall des präsentierten Verfahrens bedeutet dies konkret, das fünf verschiedene Operatoren in *AnduIN* integriert wurden: ein vollständig vom DSMS zu verarbeitender Operator und die beiden teilweise verteilten Varianten, wobei bei diesen jeweils ein DSMS Operator als auch ein INQP Operator implementiert wurden.

Basis FP^2-Stream Es wurden 3 Datenströme zu je 5000 Datenpunkten (aus $\mathbb{R}$) erzeugt. Ein Datenstrom lieferte Werte über einem Attribut, d.h. die Analyse erfolgte im Weiteren über insgesamt 3 Attributen. Die erzeugten Werte waren standardnormalverteilt, was weitestgehend dem Verhalten von realen Sensordaten entspricht (je nach Genauigkeit der eingesetzten Technik schwanken die erfassten Werte mehr oder weniger stark um den realen Wert).

Aus diesen drei Datenströmen sollten nun häufige Itemsets extrahiert werden, wobei pro Attribut (Datenstrom) ein Item gefunden werden sollte. In einem ersten Test sollten Items bzw. Itemsets mit einem minimalen Support von $minsup = 0.25$ extrahiert werden. Es wurde mit einer Batchgröße von $|B|$ = 100, einer Fenstergröße von $w = 5$, Histogrammen mit $d = 10$ Buckets und einer maximalen Schiefe von $maxskew = 0.2$ getestet. Eine vertikale Reihe Boxen in Abbildung 3 (a) und (b) entspricht dabei den Knoten im FP^2-Stream, die über diesem Attribut zum gegebenen Zeitpunkt existieren. Weiße Boxen repräsentieren Knoten welche nicht häufig sind. Graue Balken, die sich über mehrere Boxen erstrecken können, stellen Items dar, die im FP^2-Stream über mehrere Knoten verteilt



(a) Erkanntes Item (b) Informationsdichte



(c) Knotenanzahl im FP^2-Stream

Abbildung 3: Standardnormalverteilte Daten

sind. Pro Zeitschritt wird jeweils ein solches extrahiert.

Zum Vergleich ist in Abbildung 3(a) (rechts) das Item dargestellt, welches über dem selben Attribut durch den Apriori-Ansatz von Srikant [SA96] nach dem Durchlauf des kompletten Datensatzes erzeugt wird. Der direkte Vergleich zeigt, dass das vorgestellte Verfahren das Item in den nahezu gleichen Grenzen findet. Die Schwankungen beim FP^2-Stream kommen durch das Rauschen der Daten zustande. Aufgrund der Betrachtung einer viel kürzeren Vergangenheit, kommen die Schwankungen entsprechend stärker zum Tragen.

In Abbildung 3(a) ist die Initialisierungsphase des Verfahrens sehr gut zu erkennen. Zu Beginn der Analyse wird der Wertebereich durch das Item vollständig überdeckt. Anschließend wird alles überdeckende Knoten in mehrere Teilknoten zerlegt. Aufgrund des Zeitfensters von 5 Batches setzt sich diese Verfeinerung allerdings erst nach 600 Zeiteinheiten durch. Anschließend schwankt das erzeugte Item um den Mittelwert der standardnormalverteilten Daten.

Elementar für die Effektivität und die Effizienz des Verfahrens ist das verwendete Maß der Informationsdichte. Abbildung 3(b) zeigt für den betrachteten Beispieldatenstrom die Dichte für die einzelnen Items im FP^2-Stream. Die Itemgrenzen entsprechen denen aus Abbildung 3(a). Die Abbildung zeigt sehr gut, wie die Informationsdichte mit zunehmender Knotenverfeinerung steigt, und dass sie erwartungsgemäß um den Mittelwert am größten ist. Im Fall, dass keines der Items den geforderten minimalen Support aufweist,

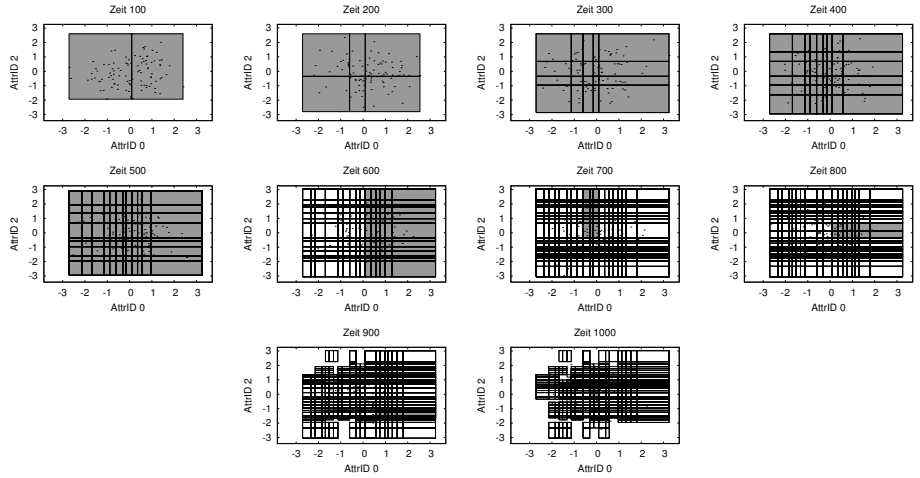


Abbildung 4: Entwicklung des 2-elementigen Itemsets über den Attributen 0 und 2

startet die Rekombination während der FP^2-Growth-Phase am Item mit der höchsten Informationsdichte.

Der FP^2-Stream ist eine Datenstruktur zur effizienten Speicherung von Transaktionen. Das kontinuierliche Verfeinern von Knoten hat allerdings einen starken Einfluss auf die Größe der Baumstruktur. So führt zum Beispiel eine Verfeinerung der Wurzel zu einer Verdoppelung aller abhängigen Knoten. Während der initialen Phase muss sich das Verfahren erst an die Daten anpassen. Dies kann eine Vielzahl an Split- und Verschmelzungs-Operationen zur Folge haben. In Abbildung 3(c) ist die zeitliche Entwicklung der Knotenanzahl zu obigem Beispiel dargestellt. Sehr gut zu erkennen ist die Spitze zu Beginn des Verarbeitungsprozesses. An dieser Stelle übersteigt die Knotenanzahl, sogar die Anzahl der im FP^2-Stream gespeicherten Transaktionen (500 Transaktionen). Nach dieser initialen Phase fällt die Anzahl an Knoten im Baum auf ca. 100-150. Da sich keine wesentlichen Änderungen in der Datenverteilung ergeben, bleibt die Knotenanzahl auf diesem Level in etwa konstant.

Abbildung 3(c) zeigt außerdem die Anzahl der Knoten pro Attribut im Baum. Erwartungsgemäß steigt diese Zahl mit der Tiefe des Baumes. Das Attribut, welches sich auf Blattebene befindet (im Beispiel Attribut 3), wird also durch die höchste Anzahl an Knoten repräsentiert.

Durch das Rekombinieren von Items innerhalb des FP^2-Stream werden automatisch auch die entsprechenden Itemsets zusammengeführt. Während der Itemset-Exktraktion werden alle Items aus der Datenstruktur herausgelöst, welche dem minimalen Support genügen. Im Beispiel sind dies neben den einelementigen Itemsets zusätzlich 2- und vereinzelt auch 3-Itemsets. Die Bilder in Abbildung 4 zeigen die zeitliche Entwicklung des 2-elementigen Itemsets über den Attributen 0 und 2. Jedes der einzelnen Bilder entspricht dabei dem am Ende eines Batches extrahiertem Itemset. Die einzelnen Boxen entsprechen wieder den

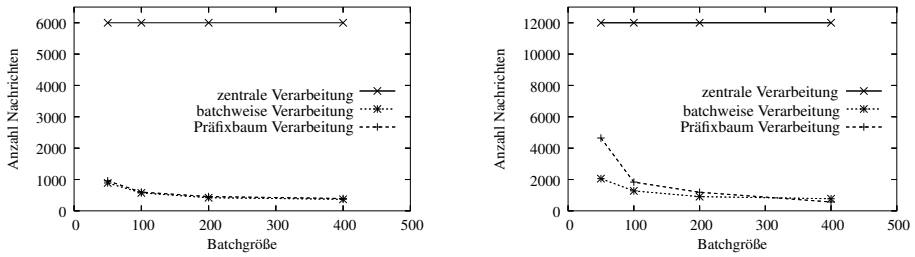


Abbildung 5: Anzahl der versandten Nachrichten: (a) Sensorknoten mit ID 2, (b) Sensorknoten mit ID 1

Items im FP^2-Stream. Das extrahierte Itemset ist grau hinterlegt.

In den Abbildungen ist die initiale Phase wiederum sehr gut zu erkennen. Zunächst überdeckt das Itemset den vollständigen Wertebereich in beiden Dimensionen. Anschließend werden die beiden Attribute durch Splits verfeinert. Aufgrund der verwendeten Fenstergröße 5 setzen sich diese Anpassungen erst ab dem Zeitpunkt 600 durch. Im weiteren Verlauf pendelt sich das Itemset um die Mittelwerte in beiden Dimensionen ein. Sehr gut zu erkennen ist auch, dass Itemsets, die den Rand des Wertebereichs repräsentieren, infolge zu weniger Werte in diesen Regionen mit der Zeit entfernt werden. Dies ist einer der Gründe, warum sich die Anzahl an Knoten im FP^2-Stream nach dem initialen Anstieg auf einem deutlich niedrigeren Level einpendelt.

In [KRS10] wurde das vorgestellte Basis-Verfahren ausführlich evaluiert. Im Weiteren soll auf die vorgestellte verteilte Implementierung eingegangen werden.

Verteilter FP^2-Stream Im Weiteren sollen die vorgestellten Varianten: vollständig auf der zentralen Komponente durchgeführte Berechnung, batchweise In-Network-Verarbeitung und Präfixbaum batchweise In-Network-Verarbeitung des FP^2-Stream miteinander verglichen werden. Hierbei stehen drei wesentliche Aspekte im Vordergrund: (i) der Einfluss der Batchgröße auf die Anzahl der Nachrichten, die versandt werden müssen, (ii) der Speicherbedarf auf den Sensoren und (ii) der Energieverbrauch auf den Sensoren. Bei den Messungen wurde mit jeweils 3 Sensorknoten gearbeitet, wobei ein Sensorknoten mit der Basisstation verbunden war. Der Sensorknoten auf der Basisstation erfasste Temperaturwerte, die beiden batteriebetriebenen Sensorknoten jeweils Temperatur und Luftfeuchtigkeit. Die Experimente erfolgten mit 6000 Messpunkten pro physikalischem Sensor. In allen Tests waren die Sensorknoten in einer logischen Kette angeordnet. Der Basisknoten bildete dabei den Beginn der Kette und hatte die ID 0. Der zweite Knoten in der Kette hatte die ID 1. Der Knoten mit ID 2 bildet das Kettenende.

In einem ersten Test wurde die Anzahl der zu versendenden Nachrichten in Abhängigkeit von der verwendeten Batchgröße gemessen. Die Evaluierung erfolgte auf Basis einer eigens für den INQP von *AnduIN* entwickelten Simulationsumgebung, da in dieser ein einfaches Überwachen der versandten Nachrichten möglich ist. Das Verfahren wurde mit 4

	238 Knoten	895 Knoten	1387 Knoten
Sensorknoten ID0			
batchweise-Ansatz	1176	1182	1416
Präfixbaum-Ansatz	16234	42362	64658
Sensorknoten ID1			
batchweise-Ansatz	2647	2687	2891
Präfixbaum-Ansatz	12383	19197	23555
Sensorknoten ID2			
batchweise-Ansatz	2279	2495	2731
Präfixbaum-Ansatz	6747	7299	7535

Tabelle 3: Speicherbedarf des FP^2-Tree auf den Sensorknoten (in Byte)

verschiedenen Batchgrößen evaluiert: 60. 100. 200 und 400 Datensätzen.

Die Abbildung 5(a) und 5(b) zeigen die Gesamtanzahl der verwandten Nachrichten für die beiden drahtlos angebundenen Sensorknoten mit ID 1 und ID 2. Die Nachrichtenanzahl umfasst dabei die von dem jeweiligen Knoten gesendeten Nachrichten (sowohl in Richtung Basisstation als auch über die Feedback-Komponente in Richtung Kettenende). Sehr gut zu erkennen ist die konstant hohe Anzahl an Nachrichten im Fall des reinen Sendens an die zentrale Instanz. Die beiden verteilten Verfahren benötigen jeweils deutlich weniger Nachrichten, wobei die Anzahl erwartungsgemäß mit steigender Batchgröße abnimmt. Ebenfalls sehr gut zu erkennen ist, dass der Knoten am Kettenende im Fall des Präfixbaum-Ansatzes nahezu die selbe Nachrichtenanzahl versendet, wie im Fall des einfachen batchbasierten Ansatzes. Anders verhält sich dies auf dem zweiten Knoten in der Kette, welcher die Ergebnisse vom Kettenende erhält, diese in seinen eigenen lokalen FP^2-Tree einarbeitet und dann die Gesamtergebnisse weiterleitet. Hier zeigt sich der Präfixbaum-Ansatz insbesondere bei kleineren Batchgrößen deutlich nachrichtenintensiver ist. Da der Knoten am Kettenende zwei Attribute erfasst, verwaltet dieser im Präfixbaum-Ansatz bereits einen einfachen FP^2-Tree. In Abhängigkeit der Verfeinerung der entsprechenden Intervalle kann dieser FP^2-Tree verhältnismäßig komplex sein, was im Weiteren eine entsprechend hohe Anzahl an zu versendenden Nachrichten zum Nachfolgeknoten nach sich zieht. Dementsprechend steigt auch die Anzahl der vom zweiten Knoten zum Knoten der Basisstation gesendeten Nachrichten. Erst bei einer Batchgröße von 400 Datensätzen wird der Präfix-Ansatz effizienter als der einfache batchweise Ansatz. Aufgrund der großen Batchgröße kommt es vermutlich zu einer geringeren Verfeinerung, so dass weniger Knoten entstehe. Infolgedessen müssen weniger Nachrichten an die Folgeknoten in der Kette versandt werden.

Im nächsten Test soll gezeigt werden, wie sich der Speicherverbrauch der beiden verteilten Verfahren auf den Sensorknoten entwickelt. Hierfür wurden drei verschiedene komplexe FP^2-Trees erzeugt. Die Anzahl der Knoten bewegte sich dabei zwischen 238 und 1387 Knoten. Die Bestimmung des für die Verwaltung des FP^2-Trees benötigten Speichers pro Sensorknoten erfolgte wiederum auf Basis der Simulationsumgebung.

In Tabelle 3 ist der Speicherbedarf pro Knoten dargestellt. Im Fall des batchweisen Ansatzes steigt der Speicherbedarf mit der Komplexität der Datenstruktur nur langsam an. Sehr gut ist auch der Unterschied zwischen dem Knoten mit der ID0 und den anderen

Operator	Zeit in ms	elektrische Stromstärke in mA	konsumierte Energie in μJ
Einfügen eine Wertes			
batchweise-Ansatz	11.9	6.6	259.2
Präfixbaum-Ansatz	12.3	7.0	284.1
Batchende			
batchweise-Ansatz	65.8	7.7	1672.0
Präfixbaum-Ansatz	55.2	7.6	1384.4
Daten Senden (Initialisierung)	2.3	7.9	61.7
Daten Senden (62Byte - 1 Push)	3.6	8.4	99.7
Daten Senden (Initialisierung + 73xPush)	271.0	-	7344.8

Tabelle 4: Energieverbrauch für die beiden FP^2-Stream-In-Network-Operatoren

beiden Knoten zu erkennen. Da dieser lediglich ein Attribut überwacht benötigt er auch nur halb so viel Speicher wie die anderen beiden Knoten. Die Tabelle zeigt auch wie sich der Speicherbedarf im Fall des Präfixbaum-Ansatzes mit jedem zusätzlichen Knoten dramatisch erhöht. Hier erhöht sich der Speicherbedarf für eigene FP^2-Tree-Knoten um den Speicherbedarf für die FP^2-Tree-Knoten der Vorgänger in der Kettenstruktur. Auch die Größe des FP^2-Tree hat einen dramatischen Einfluss auf die Größe des benötigten Speichers. Im Fall des großen FP^2-Tree auf dem Sensorknoten mit der ID 0 überschreitet der benötigt Speicherbedarf bereits 64kB, was mehr ist als viele Sensorknoten zu Verfügung stellen. Die Tabelle zeigt weiterhin, dass – sofern vorab abgeschätzt werden kann wie sich die Größe des FP^2-Tree entwickelt – auch eine Speicherabschätzung auf Basis von Erfahrungswerten möglich ist. Im Fall des batchbasierten Ansatzes entwickelt sich der benötigte Speicherplatz nahezu linear mit der Anzahl an Attributen und der Größe des FP^2-Tree.

Hinsichtlich der Einsatzmöglichkeiten der vorgestellten FP^2-Stream-Varianten als In-Network-Operatoren ist insbesondere der Energieverbrauch der einzelnen Implementierungen interessant. Im folgenden soll dieser bestimmt werden. Für die Messungen kamen Sensorknoten mit foljgender Ausstattung zum Einsatz: MSBA2-Boards mit ARM LPC2387 CPU, CC1100 Funk-Modul, 512KB Flashspeicher und 98 KB RAM, sowie SHT11-Sensoren (Luftfeuchtigkeit und Temperatur). Da die beiden verteilten Varianten batchbasiert sind, muss zwischen den Kosten für das Einfügen eines Wertes in den FP^2-Stream und der Kosten für die Auswertung am Batchende unterschieden werden. In Tabelle 4 sind die jeweiligen Energieverbräuche auf den Sensorknoten für einen mittelgroßen FP^2-Tree (895 Knoten) dargestellt. Zum Vergleich enthält die Tabelle zusätzlich die Kosten für das Versenden eines 62 Byte Nachrichtenpaketes (6 Byte Header). Statt der gemessenen $7344.8\mu J$ für das Senden jedes Datensatzes sind bei der In-Network-Verarbeitung des FP^2-Stream je nach Variante nur noch zwischen $259.2\mu J$ und $284.1\mu J$ notwendig. Erst am Batchende entstehen durch das Auslesen der Daten zusätzliche Kosten, die aber immer noch deutlich geringer ausfallen, als die für das Versenden eines Datensatzes. Durch das deutliche Verringern der Nachrichtenanzahl ergibt sich somit eine drastische Reduktion des Gesamtenergieverbrauches bei den beiden vorgestellten in-Network-Varianten des präsentierten Verfahrens.

Die Experimente zeigen zwei wichtige Ergebnisse: (i) Der Ansatz der batchweisen Verarbeitung arbeitet in jedem Fall effizienter als die zentrale Variante und ist auch in Zusam-

menhang mit großen FP^2-Trees einsetzbar. (ii) Der Präfixbaum-Ansatz ist zwar ebenfalls energieeffizienter als die zentrale Variante, seine Einsatzmöglichkeiten sind aber durch die Größe des FP^2-Tree beschränkt. Erst im Fall einer relativ kleinen Indexstruktur macht sich der erwartete Vorteil durch die geringere Nachrichtenanzahl bemerkbar. Die notwendige Einschwingphase führt bei diesem Ansatz außerdem dazu, dass er zu Beginn sehr kostenintensiv ist, so dass die vorhandenen Speicherressourcen auf den Sensorknoten unter Umständen aufgebraucht werden und es zu entsprechenden Verarbeitungsabbrüchen kommt. Hat sich die Anzahl der Knoten hingegen auf einem niedrigen Niveau eingependelt, so kann dieser Ansatz durchaus energieeffizienter sein als der primitive batchweise Ansatz. Für das Problem des Einschwingens sind mehrere Lösungen denkbar. Eine generelle Reduktion der Knotenanzahl im FP^2-Stream wäre eine Möglichkeit. Ein anderer Ansatz wäre eine hybride Lösung aus den beiden vorgestellten verteilten Ansätzen. Während der Einschwingphase wird zunächst mit dem batchweisen Ansatz begonnen. Nachdem sich die Knotenanzahl auf einem entsprechend niedrigen Niveau eingependelt hat, könnte anschließend auf den Präfixbaum-Ansatz gewechselt werden.

7 Zusammenfassung

Der effiziente Umgang mit den beschränkten Energieressourcen ist eine der Schlüsselaufgaben in drahtlosen Sensornetzen. Neben der Integration einfacher Verfahren zum Beispiel zur Datenaggregation und -filterung werden zunehmend auch komplexere Verarbeitungsprozesse auf die einzelnen Sensorknoten ausgelagert.

In der vorliegenden Arbeit wurde ein neuartiger Ansatz zur Extraktion von häufigen Mustern aus Datenströmen über quantitativen Attributen präsentiert. Das Verfahren kombiniert dabei bekannte Techniken aus dem Frequent Pattern Mining mit Lösungen aus dem Bereich der multidimensionalen Histogramme. Für eine energieeffiziente In-Network-Verarbeitung des Verfahrens wurden zusätzlich zum trivialen Ansatz einer vollständig zentralen Verarbeitung zwei weitere Lösungsansätze präsentiert, welche die Daten teilweise im Sensornetz vorverarbeiten. In der vorgestellten Evaluierung konnte gezeigt werden, dass das präsentierte Verfahren für die Detektion häufiger Muster über quantitativen Attributen geeignet ist. Zudem wurde gezeigt, dass die beiden vorgestellten Lösungsansätze zu einer signifikanten Reduktion der Nachrichtenanzahl und damit zu einer Reduktion des Gesamtenergieverbrauches des WSN führen kann.

Die Ergebnisse haben allerdings auch noch Schwachstellen des Verfahrens aufgezeigt. So ist gerade die initiale Einschwingphase und die Fixierung des Verfahrens an die erste Konfiguration im Zusammenhang mit der Gesamtanzahl an Elementen in der Datenstruktur problematisch. Für beide Probleme wurden entsprechende Lösungsansätze vorgeschlagen, welche in zukünftigen Arbeiten weiter verfolgt werden sollen.

Literatur

[AIS93] R. Agrawal, T. Imielinski und A. N. Swami. Mining Association Rules between Sets of Items in Large Databases. In *SIGMOD 1993*, Seiten 207–216, 1993.

[AL03] Y. Aumann und Y. Lindell. A Statistical Theory for Quantitative Association Rules. *Journal of Intelligent Information Systems*, 20(3):255–283, 2003.

[AS94] R. Agrawal und R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *VLDB 1994*, Seiten 487–499, 1994.

[FMMT96] T. Fukuda, Y. Morimoto, Sh. Morishita und T. Tokuyama. Mining optimized association rules for numeric attributes. In *PODS 1996*, Seiten 182–191, 1996.

[GHRL03] C. Giannella, J. Han, E. Robertson und C. Liu. Mining Frequent Itemsets over Arbitrary Time Intervals in Data Streams. Bericht, Indiana University, 2003.

[HPY00] J. Han, J. Pei und Y. Yin. Mining Frequent Patterns without Candidate Generation. In *SIGMOD 2000*, Seiten 1–12, 2000.

[KHKS10] D. Klan, K. Hose, M. Karnstedt und K. Sattler. Power-Aware Data Analysis in Sensor Networks. In *ICDE 2010*, Seiten 1125–1128, 2010.

[KRS10] D. Klan, Th. Rohe und K. Sattler. Quantitatives Frequent-Pattern Mining über Datenströmen. In *KDML 2010*, 2010.

[LK06] C. K.-S. Leung und Q. I. Khan. DSTree: A Tree Structure for the Mining of Frequent Sets from Data Streams. In *ICDM 2006*, Seiten 928–932, 2006.

[LRS02] St. Lindsey, C. Raghavendra und K. M. Sivalingam. Data Gathering Algorithms in Sensor Networks Using Energy Metrics. *IEEE Trans. Parallel Distrib. Syst.*, 13(9):924–935, 2002.

[MAR02] J. Mata, J.L. Alvarez und J.C. Riquelme. An Evolutionary Algorithm to Discover Numeric Association Rules. In *SAC 2002*, Seiten 590–594, 2002.

[MFHH05] S. Madden, M. J. Franklin, J. M. Hellerstein und W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Trans. on Database Systems*, 30(1):122–173, 2005.

[MM02] G. S. Manku und R. Motwani. Approximate Frequency Counts over Data Streams. In *VLDB 2002*, Seiten 346–357, 2002.

[PCY95] J. S. Park, M.-S. Chen und P. S. Yu. An Effective Hash-based Algorithm for Mining Association Rules. In *SIGMOD '95*, Seiten 175–186, 1995.

[Röm08] K. Römer. Discovery of Frequent Distributed Event Patterns in Sensor Networks. In *EWSN 2008*, Seiten 106–124, 2008.

[SA96] R. Srikant und R. Agrawal. Mining quantitative association rules in large relational tables. *SIGMOD Rec.*, 25(2):1–12, 1996.

[SAVCN07] A. Salleb-Aouissi, C. Vrain, C. und Nortet. QuantMiner: a genetic algorithm for mining quantitative association rules. In *IJCAI 2007*, Seiten 1035–1040, 2007.

[TAJL08] S. K. Tanbeer, Ch. F. Ahmed, B.-S. Jeong und Y.-K. Lee. Efficient Frequent Pattern Mining over Data Streams. In *CIKM '08*, Seiten 1447–1448, 2008.

[TAJL09] S. K. Tanbeer, Ch. F. Ahmed, B.-S. Jeong und Y.-K. Lee. Efficient single-pass frequent pattern mining using a prefix-tree. *Inf. Sci.*, 179(5):559–583, 2009.

[WTHL02] K. Wang, L. Tang, J. Han und J. Liu. Top Down FP-Growth for Association Rule Mining. In *PAKDD '02*, Seiten 334–340, 2002.