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 sich 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. Auf Grundlage dieser lassen sich anschließend zum Beispiel Regelmengen bestimmen, welche zur Prozessoptimierung eingesetzt werden können.

Das Problem der Detektion von 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. Mit dem FP^2-Stream wurde in [KRS10] ein neuer Ansatz für das Problem des quantitativen Frequent Pattern Mining über Datenströmen mit kontinuierlichen Wertebereichen präsentiert. Schwerpunkt der vorliegenden Arbeit ist die Anpassung des in [KRS10] vorgestellten Verfahrens zur Verarbeitung innerhalb eines WSN mit dem Ziel einer energieeffizienten Vorverarbeitung.

Im Weiteren wird ein kurzer Überblick über Vorarbeiten im Bereich des Frequent Pattern Mining gegeben. In Abschnitt 3 folgt eine kurze Beschreibung des in [KRS10] präsentierten Verfahrens für das Finden der häufigen Itemsets über quantitativen Attributen. Anschließend werden in Abschnitt 4 zwei Ansätze vorgestellt, welche dieses Verfahren für eine teilweise Berechnung im Sensornetz partitioniert. In Abschnitt 5 folgt eine Evaluierung der vorgestellten Lösung. Der Beitrag schließt mit einer Zusammenfassung.

2 Verwandte Arbeiten

Eines der ersten Verfahren zum Finden von häufigen Mustern war das von Agrawal et al. präsentierte *Apriori*-Verfahren [AIS93]. 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 [PCY95,FMMT96]. Han et. al präsentieren in [HPY00] mit dem FP-Tree (*frequent pattern tree*) einen Präfixbaum zur effizienten Speicherung häufiger Muster. Zusätzlich stellen sie mit dem FP-Growth-Algorithmus ein Verfahren zum Finden häufiger Muster auf Basis des FP-Tree vor.

Die erste Arbeit die sich mit dem Problem des quantitativen Frequent-Itemset-Minings beschäftigte stammt von Srikant und Agrawal [SA96]. Das vorgestellte Verfahren basiert dabei im Wesentlichen auf dem 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.

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

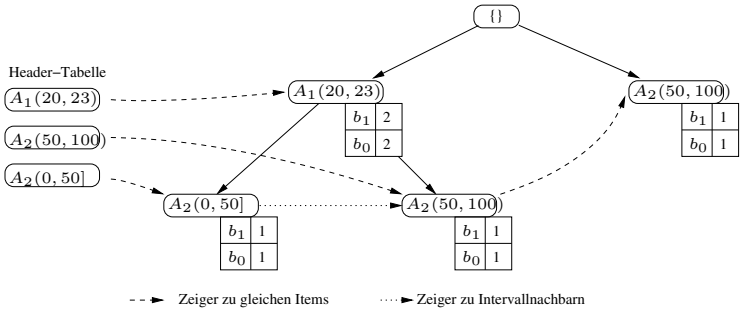


Abbildung 1: Beispiel FP^2-Stream

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 Basis FP^2-Stream

Mit dem FP^2-Stream wurde in [KRS10] eine Datenstruktur zur kompakten Repräsentation quantitativer Itemsets vorgestellt. Die Idee basiert dabei auf dem datenstromgeeigneten FP-Stream von Gianella et al. [GHRL03]. Beim batchbasierten FP-Stream werden in den Knoten (schiefe) Zeitfenster eingebettet, welche Häufigkeiten einzelner Batches zu den jeweiligen Zeitpunkten enthalten.

Statt eines kategorischen Wertes überdeckt jeder Knoten im FP^2-Stream ein Intervall. Die Intervallgrenzen der einzelnen Items werden durch kontinuierliche Split- und Merge-Schritte am Ende der Verarbeitung eines Batches an die aktuelle Datenverteilung im Datenstrom angepasst. Die Entscheidung ob und welche Items verfeinert werden, erfolgt dabei auf Grundlage der Datenverteilung der letzten Batches. Hierzu wird für jedes Item in der Headertabelle ein entsprechendes Histogramm verwaltet. Um ein altern von Itemsets zu ermöglichen, wurden statt schiefer Zeitfenster gleitende Fenster in die einzelnen Knoten eingebettet. Alte, weniger interessante Muster werden somit automatisch aus der Datenstruktur entfernt. Ein Zugriff auf beliebig alte Daten ist somit aber nicht mehr möglich.

Für eine effiziente Verwaltung der Datenstruktur mussten zudem zusätzliche Verweise zwischen benachbarten Fenstern bzw. benachbarten Itemsets eingeführt werden. Abbildung 1 zeigt einen einfachen Beispiel-FP^2-Stream. In [KRS10] konnte gezeigt werden, dass das präsentierte Verfahren für die Detektion von häufigen Itemsets über quantitativen Itemsets geeignet ist.

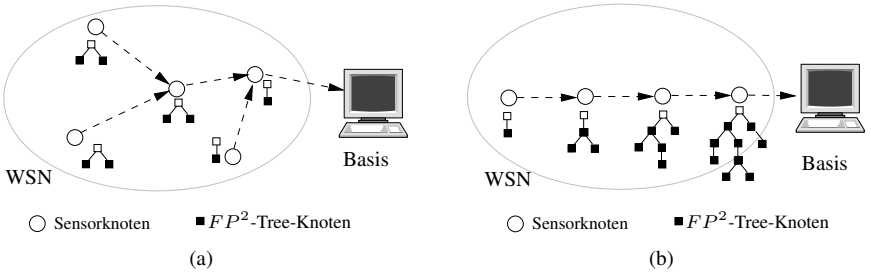


Abbildung 2: Verteilte Verarbeitung des FP^2-Stream: (a) batchweise Verarbeitung und (b) Präfixbaum-batchweise Verarbeitung

4 Verteilter FP^2-Stream

Basierend auf dem Basisverfahren sollen im Weiteren zwei erweiterte Varianten vorgestellt werden, welche teilweise in einem WSN bearbeitet werden können. Die beiden präsentierten Verfahren nutzen dabei konsequent die batchweise Verarbeitung des Verfahrens aus. Zudem verbleiben die Operationen zur Reorganisation der Datenstruktur (Split und Merge) 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. D.h., 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. Ergebnisse anderer Knoten werden über alternative Pfade zum Basisknoten geroutet.

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.

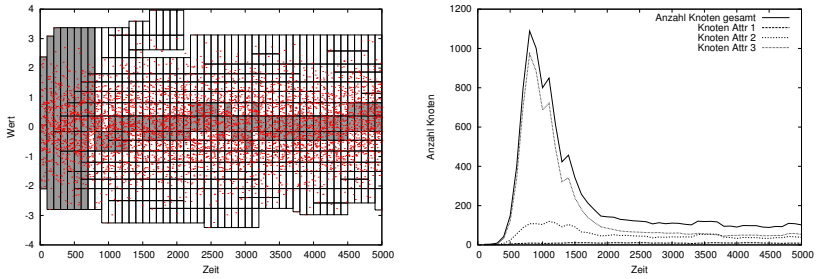
Bei diesem Ansatz verwaltet jeder Knoten im WSN einen Teilbaum des FP^2-Tree. Der von der Basisstation am weitesten entfernte Knoten verwaltet die Wurzel und alle von ihm bedienten Items des FP^2-Tree. Der nächste Knoten in Richtung Basisstation verwaltet eine vollständige Kopie der Datenstruktur seines 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.



(a) Erkanntes Item (b) Knotenanzahl im FP^2-Stream

Abbildung 3: Standardnormalverteilte Daten

5 Evaluierung

Im folgenden Abschnitt sollen die beiden präsentierten verteilten Ansätze gegenüber einer rein auf der Basisstation stattfindenden Verarbeitung evaluiert werden. Die Evaluierung wurde anhand von *AnduIN* [KHKS10], einem kombinierten Datenstrom-/In-Network-Query-Prozessor, vorgenommen.

Basis FP^2-Stream Zunächst sollen kurz wesentliche Eigenschaften des FP^2-Stream an einigen Untersuchungsergebnissen wiedergegeben werden. Hierzu 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 wurden nun häufige Itemsets mit einem minimalem Support von 0.25 extrahiert. Weiterhin 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 (siehe [KRS10]). Abbildung 3(a) zeigt beispielhaft die zeitliche Entwicklung eines der über dem Datenstrom extrahierten Items. Eine vertikale Reihe Boxen in Abbildung 3(a) 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 sind. Pro Zeitschritt wird jeweils ein solches extrahiert.

Das kontinuierliche Verfeinern von Knoten hat allerdings einen starken Einfluss auf die Größe der Baumstruktur. 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(b) ist die zeitliche Entwicklung der Knotenanzahl zu obigem Beispiel dargestellt. Sehr gut zu erkennen ist die Spitze zu Beginn des Verarbeitungsprozesses. 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(b) zeigt außerdem die Anzahl

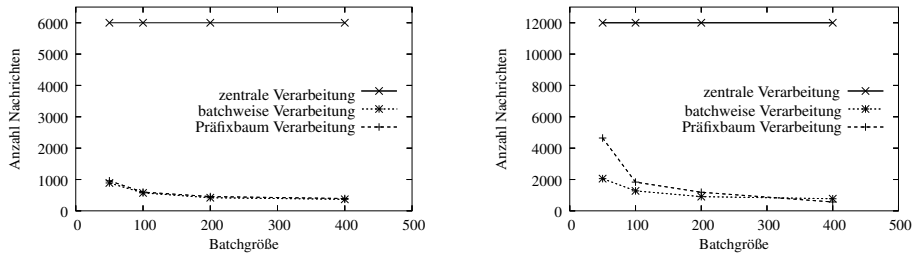


Abbildung 4: Anzahl versandter Nachrichten: (a) Sensorknoten mit ID 2, (b) Sensorknoten mit ID 1

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.

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

Die Abbildungen 4(a) und 4(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-

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

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 1 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 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 signifikanten 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 Erfah-

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

rungswerten 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 folgender 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 2 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 Zusammenhang 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 Datenstruktur 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.

6 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 der in [KRS10] wurden Lösungsansätze präsentiert, welche eine Verarbeitung des in [KRS10] vorgestellten FP^2-Stream teilweise im Sensornetz ermöglicht. In der vorgestellten Evaluierung konnte gezeigt werden, 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.

[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.

[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.

[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.