Verteilte Sensornetzwerke

Mit Datenaggregation und Sensorfusion

PD Stefan Bosse

Universität Bremen - FB Mathematik und Informatik

Algorithmen in DSN (Teil 1)

Welche Algorithmen finden in DSN Anwendung?

Welche Besonderheiten existieren in DSN?

Wie wirkt sich Verteilte Berechung und Koordination auf Kommunikation aus?

Algorithmenklassen

Algorithmen in Verteilten Sensornetzwerken können unterschieden werden in folgende Klassen:

1. Nachbarschaftsexploration von Sensorknoten, Lokalisation
2. Graphenalgorithmen (Gruppenbildung, Netzwerkbildung, Pfadfindung usw., Routing)
3. Sensoraggregation und Sensorfusion (Lokaler vs. globaler Zustand)
4. Konsens und Mehrheitsfindung, Leader Wahl
5. Verteilter Mutualer Ausschluss
6. Verteiltes Rechnen - Das DSN als Big Virtual Machine
7. Verteilte Services (Sensor Cloud)

Lokaler vs. Globaler Zustand

In einem großen verteilten System kann kein Knoten jederzeit eine globale Ansicht des gesamten Systems haben. Stattdessen hat jeder Knoten nur eine lokale Ansicht des Systems und muss seine Entscheidungen auf diesen lokalen Informationen treffen.

- Viele Aufgaben können vollständig lokal gelöst werden, zum Beispiel kann ein Knoten die niedrigste gemessene Temperatur in seiner Nachbarschaft ermitteln, indem er einfach mit seinen Nachbarn kommuniziert.
- Viele andere Aufgaben sind von Natur aus global, zum Beispiel die Ermittlung der Mindesttemperatur im gesamten Sensornetzwerk. Um solche globalen Probleme zu lösen, müssen einige Informationen das gesamte Netzwerkdiagramm durchlaufen.

Verteilte Sensornetzwerke unterscheiden sich von generischen verteilten Systemen wesentlich durch deren Eingabedaten die bereits verteilt sind (keine oder nur kurzreichweitige Kommunikation)! Einzig (reduzierte) Ausgabedaten müssen über eine größere Reichweite kommuniziert werden.

- Die Beschränkung der Entfernung, aus der Informationen gesammelt werden, impliziert, dass
  1. Verteilte Sensoralgorithmen schnell sind, da die Kommunikation typischerweise der zeitaufwändigste Teil ist, und
  2. Änderungen der Eingabe oder Topologie lokal behandelt werden können, d.h. nur ein Teil der Knoten muss ihre Ausgabe neu berechnen.

In einem großen System stören Fehler oder Rekonfigurationen nur einen kleinen Teil des Systems. Sie können aus globaler Sicht korrekt funktionieren, selbst wenn einige der Knoten ausfallen, fehlerhaftes Verhalten zeigen oder sogar absichtlich versucht wird, das System als Ganzes zu stören.

Taxonomie der Verteilten Algorithmen

Globaler oder Zentraler Algorithmus

Der globale Algorithmus arbeitet direkt auf dem gesamten Netzwerk und verfügt über einen vollständigen Sensordatenzustand D des Netzwerks

Verteilter Algorithmus

In einem verteilten Algorithmus führt jeder Sensorknoten seinen eigenen Algorithmus aus. Ein Knoten kennt nur den eigenen Zustand. Um mehr über den Rest des Netzwerks zu erfahren, tauschen benachbarte Knoten wiederholt Nachrichten miteinander aus (Erweiterung des Datenzustandes eines Knotens).

Verteilter und Lokalisierter Algorithmus

Ein lokalisierter Algorithmus ist ein Sonderfall eines verteilten Algorithmus. Zu Beginn hat ein Knoten nur Informationen über seinen eigenen Zustand. Um mehr über den Rest des Netzwerks zu erfahren, müssen Nachrichten ausgetauscht werden.

- In einem k-lokalisierten Algorithmus darf jeder Knoten mit einem gegebenen konstanten Wert k höchstens k Mal mit seinen Nachbarn kommunizieren (Upper Boundary Limit).
- Ein Knoten kann jedoch seine Kommunikation verzögern; Beispielsweise kann ein Knoten warten, um Nachrichten zu senden, bis alle seine Nachbarn mit größeren Bezeichnern einen bestimmten Ausführungszustand erreicht haben.

- Lokalisierte Algorithmen können vorteilhaft sein, wenn sie nur eine kleine Anzahl von Nachrichten benötigen.
- Lokalisierte Algorithmen können aber auch langsam sein: Ein Knoten u muss möglicherweise warten, bis ein Nachbar v alle seine Nachrichten sendet, während Knoten v wiederum auf seinen Nachbarn w warten muss und so weiter.
    - Tatsächlich kann es eine lineare Kausalitätskette geben, wobei zu jeder Zeit nur ein Knoten aktiv ist. Dies ergibt eine Worst-Case-Ausführungszeit von $\Theta(n)$, wobei n die Anzahl der Knoten ist.
- Lokalisierte Algorithmen haben synchronen Nachrichtenaustausch

Verteilter und Lokaler Algorithmus

Auch hier kennt jeder Knoten zu Beginn nur seinen eigenen Zustand. In einem k-lokalen Algorithmus kann jeder Knoten mit einem konstanten Wert k höchstens k Mal mit seinen Nachbarn kommunizieren. Im Gegensatz zu k-lokalisierten Algorithmen können Knoten ihre Nachrichten nicht verzögern. Insbesondere verarbeiten alle Knoten k synchronisierte Phasen, und die Operationen eines Knotens in Phase i hängen von den Informationen ab, die während der Phasen 1 bis $i-1$ empfangen werden.

- Ein lokaler Algorithmus ist ein verteilter Algorithmus, der unabhängig von der Anzahl der Knoten im Netzwerk in einer konstanten Anzahl synchroner Kommunikationsrunden abläuft.

Die effizientesten lokalen Algorithmen sind oft randomisiert, d.h. die Anzahl der Runden k kann variieren.

Aufgaben in Verteilten Systemen

- Koordination von Aufgaben
- Datenverteilung (Nachbarschaft, global)
- Datenzusammenführung (Nachbarschaft, global)
- Daten- und Sensorauswahl (Gruppierung, Netzwerkbildung)
- Wettbewerb auflösen (Scheduling und Mutualer Ausschluss)
- Planung von Messaufgaben

Verteilter Brechnungsgraph

Ein verteiltes System wird als Graph $G = (V, E)$ modelliert, dessen Knotensatz V (mit $n := |V|$) die Recheneinheiten sind und dessen Kantensatz E definieren, welche Knoten direkt miteinander kommunizieren dürfen.

- Man konzentriert sich typischerweise auf die Lokalität eines Problems, d.h. auf die maximale (Hop-) Entfernung, von der Knoten Informationen als Funktion der Anzahl der Knoten n erhalten müssen.
- Die Berechnung wird i.A. in Runden iterativ durchgeführt, wobei in jeder Runde alle Knoten gleichzeitig Nachrichten austauschen.

```
Each node v performs the following actions concurrently with all other nodes:
repeat
  send (possibly different) messages to neighbours
  receive neighbours' messages
  perform some local computations to prepare for the next round
until termination;
```

Alg. 1. Grundgerüst eines verteilten Algorithmus mit lokalen Daten

Nachrichten und Transmissionsenergie

Die von einem Knoten verbrauchte Energie wird durch die Summe über alle seine Übertragungen berechnet. Somit hat die Energie, die zum Senden einer Nachricht benötigt wird, die Form $c \cdot d^{\alpha}$, wobei d der Abstand zwischen Sender und Empfänger ist, α der Pfadverlustexponent (normalerweise $\alpha > 2$) ist und c eine Konstante ist.

Netzwerkpartitionierung und Gruppen

- Sowohl für die verteilte Berechnung als auch für Sensorgruppierung sind Graphenalgorithmen wichtig

Häufiges Problem in dynamischen ad-hoc und mobilen Sensornetzwerken ist Gruppenbildung der Sesnorknoten unter Randbedingungen

- Bekanntes Problem: Partitionierung eines Graphens in unabhängige Knotenmengen

MIS

Maximal Independent Set Problem: Wie kann ein Netzwerkgraph aus Knoten V so mit Kanten (Kommunikation) E verbunden werden, so dass ein Knoten, der zu einer Klasse $c_i \in C$ gehört, immer nur mit Knoten einer anderen Klasse c_j mit $j \neq i$ verbunden ist?

Zum Beispiel ist für eine Sensorfusion immer ein Tripel aus räumlich verteilten Sensoren (T,H,V) erforderlich (T: Temperatur, H: Luffeichtigkeit, V: Luftbewegung)

[yaroslavvb.blogspot.com/2011/03/linear-programming-for-maximum.html]

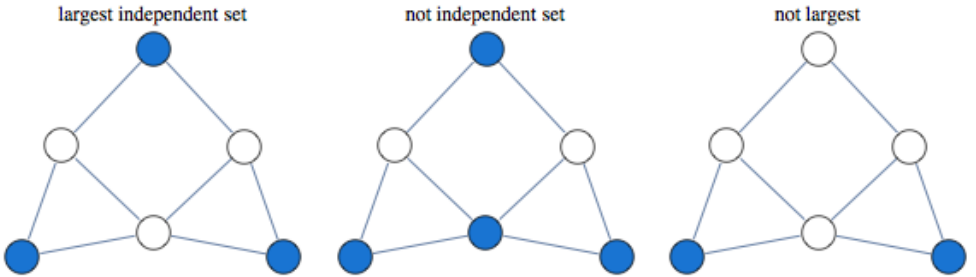

Abb. 1. Beispiele für Netzwerkkonfigurationen

MDS

Minimal Dominating Set Problem: Die Berechnung der kleinsten unabhängigen Partitionen von Netzwerkknoten (Gegenteil zu MIS).

Betrachten wir einen einfachen verbundenen ungerichteten Graphen $G = (V, E)$, wobei V eine Menge von Knoten und E eine Menge von Kanten ist. Für jeden Prozess p und q definieren wir $||p, q||$, den Abstand von p nach q, als die Länge des kürzesten Pfades in G von p nach q.

Bei einer nicht negativen ganzen Zahl k ist eine Teilmenge von Prozessen D eine k-dominierende Menge von G, wenn jeder Prozess, der nicht in D ist, höchstens k von einem Prozess in D entfernt ist (*Begrenzung der Kommunikation in Sensornetzwerken*).

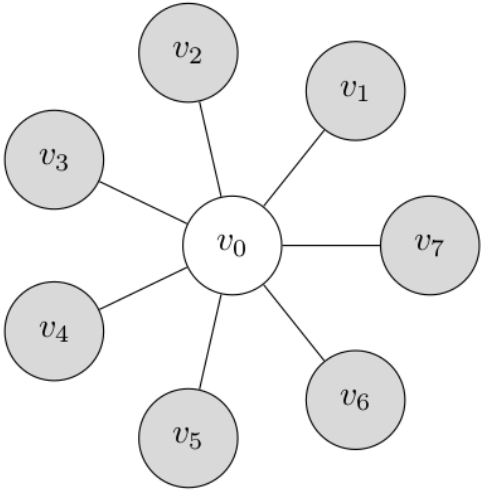

Abb. 2. Beispiel für ein minimales 1-dominierendes Netzwerk

- Die Berechnung einer dominanten Menge (DS) ist eine grundlegende Operation in Sensornetzwerken. Eine solche Menge kann beispielsweise zum Erstellen von Knotenclustern verwendet werden.
- Darüber hinaus kann es als Grundlage für den Aufbau von Backbone-Netzwerken dienen, die typischerweise ein verbundenes DS bilden.

[9]

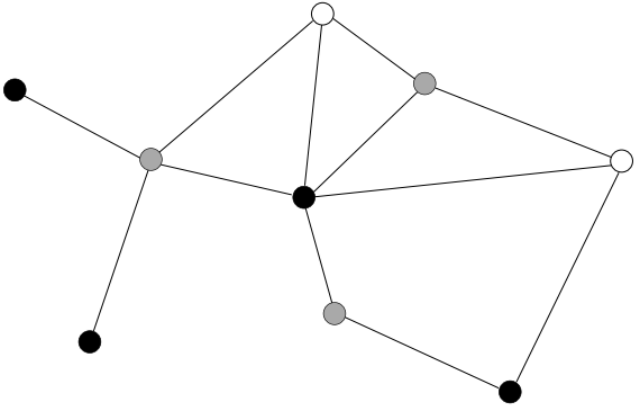

Abb. 3. MIS: Partitionierung eines Netzwerkgraphen in unabhängige Knotenmengen (mit 3er Färbung)

[7]

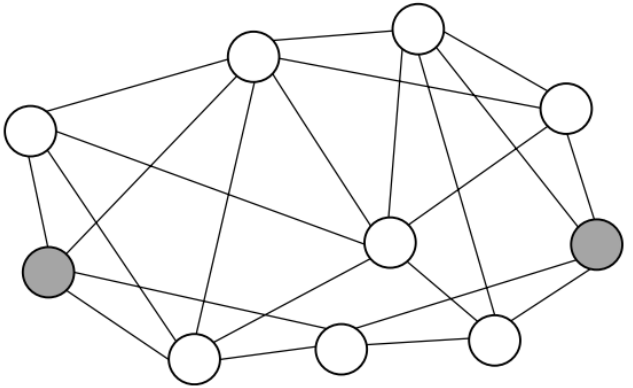

Abb. 4. MDS: Eine Minimum dominierende Menge mit zwei Dominatoren

MDS

Ein einfacher Ansatz ist der folgende:

1. Zuerst initialisieren wir die Menge der Dominatoren mit der leeren Menge, $D := \{\}$.
2. Wir nennen Knoten in D schwarz ("Dominatoren"), Knoten die mit Knoten in D verbunden sind grau ("dominierte Knoten"), und alle anderen Knoten weiß.
3. Sei $w(v)$ die Anzahl der weißen Knoten neben Knoten v, einschließlich v selbst.
4. Dann iterieren wir in jedem Schritt über alle Knoten v (globale Operation!) und berechnen die Zahl $w(v)$ der weißen Nachbarn von v, wobei vermerkt wird, dass ein Knoten x dieser Menge die größte Anzahl hat.
5. Am Ende jedes Schrittes fügen wir die Knoten x zu D hinzu.
   - Das heißt, wir wählen den Knoten als Dominator, der die meisten neuen Knoten abdeckt und "gierig" die Anzahl der verbleibenden Knoten so weit wie möglich reduziert.

[7]

```
1: D := {};
2: while ∃ white nodes do
3:   x := {x|w(x) = max_v{w(v)}};
4:   D := D ∪ x;
5: od;
```

Alg. 2. Globaler und "gieriger" MDS Algorithmus

Verteilter Globaler MDS: Jeder Sensorknoten sendet seine eigene Kennung plus die Kennungen aller seiner Nachbarn an alle anderen Knoten im Netzwerk. Daher erhält jeder Knoten ein Bild des gesamten Konnektivitätsdiagramms. Dann berechnet der Knoten mit dem größten Bezeichner das MDS unter Verwendung des globalen Algorithmus

Dieser Algorithmus ist tatsächlich verteilt und liefert kleine dominierende Mengen: Die Größe der Mengen ist wiederum asymptotisch optimal. Aufgrund des Versendung von Broadcastnachrichten ist die Nachrichtenkomplexität sehr groß. Darüber hinaus skaliert der Algorithmus nicht auf eine große Anzahl von Sensorknoten.

Lokaler und Lokalisierter Algorithmus

- Knoten in einem k-lokalen Algorithmus können nur Informationen über Knoten in ihrer k-Nachbarschaft sammeln
- Bei einigen lokalen Algorithmen kann eine beliebig kleine Konstante k gewählt werden (auf Kosten eines geringeren Nachbarschaftsverhältnisses)
- Lokale Algorithmen eignen sich daher besonders für Szenarien, in denen sich die Umgebung der Knoten häufig ändert, da sie sich ständig an die neuen Gegebenheiten anpassen können.

Lokaler Algorithmus: Eine dominierende Menge kann auch mit einem lokalen Algorithmus berechnet werden: Jeder Knoten u fragt seine Nachbarn mit höherer Priorität/ID (in Bezug auf Knotengrad und Bezeichner) nach ihren Nachbarn. Wenn diese Nachbarn mit höherer Priorität verbunden sind und alle Nachbarn von u abdecken, tritt u nicht der dominierenden Menge bei und wird andernfalls zu einem Dominator.

Lokalisierter Algorithmus: Jeder Knoten *v* wartet, bis alle seine Nachbarn, die einen größeren Grad (oder im Falle desselben Grades eine größere Kennung) als *v* haben, entschieden haben, ob sie der dominierenden Menge beitreten sollen oder nicht. Wenn einer dieser Knoten ein Dominator ist, entscheidet *v*, sich nicht der dominierenden Menge anzuschließen. Andernfalls wird *v* zum Dominator. Daher muss jeder Knoten höchstens zweimal mit seinen Nachbarn kommunizieren: einmal, um ihren Grad herauszufinden und einmal, um ihnen von seiner Entscheidung mitzuteilen.

[7]

```
1: (* Code executed by node v *)
2: send degree and ID to all neighbors;
3: receive messages from neighbors;
4: while (∃ undecided neighbor w with prio(w) > prio(v)) do
5:   wait();
6: od;
7: (* Decision *)
8: if (∃ dominator in neighborhood) then
9:   D := D;
10:  send "I am dominated!" to neighbors;
11: else
12:  D := D ∪ v;
13:  send "I am a dominator!" to neighbors;
14: fi;
```

Alg. 3. Verteilter und Lokalisierter MDS Algorithmus

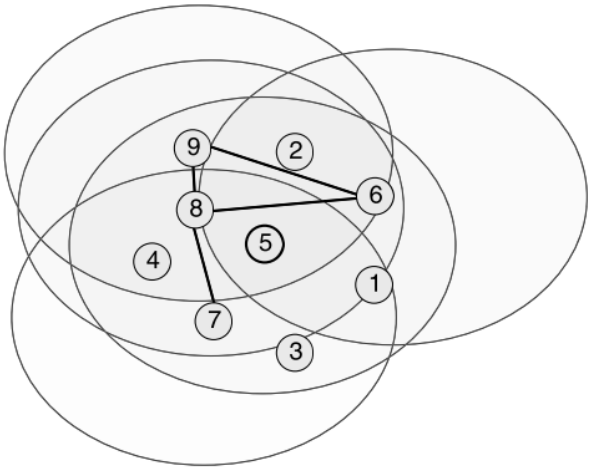

Abb. 5. Illustration des Local Dominating Set Algorithmus: Da alle Knoten mit größeren IDs miteinander verbunden sind und alle anderen Nachbarn von Knoten 5 abdecken, tritt Knoten 5 der dominierenden Menge nicht bei — unabhängig von den Entscheidungen

MIS Problemdefiniton

Gegeben ein Graph $G = (V, E)$, eine unabhängige Menge ist eine Teilmenge $S \subseteq V$ von Knoten, so dass keine zwei benachbarten Knoten in S, d.h., für jede Kante $\{v, w\} \in E$, gilt dass $v \notin S$ und $w \notin S$. Ein maximales S ist eine unabhängige Menge wenn dir Ausschlussbedingung gilt: $S \cup \{v\}$ ist nicht unabhängig für jedes $v \in V \setminus S$.

Def. 1. Maximale unabhängige Knotenmenge

Algorithmen

- Das MIS ist ein grundlegendes Symmetrieproblem, das für einen zentralisierten Algorithmus trivial ist.
- Beginnend mit $S := \emptyset$ geht man einfach nacheinander über die Knoten und fügt einen beliebigen Knoten ohne Nachbarn in S zu S.
  - Wenn Knoten jedoch gleichzeitig agieren, können sie S in derselben Runde beitreten; daher muss sichergestellt sein:
  - Keine zwei benachbarten Knoten dürfen in derselben Runde S beitreten.
  - Unter der Annahme, *dass Knoten eindeutige und relationale Bezeichner haben*, wird die triviale sequentielle MIS-Lösung in einen einfachen verteilten Algorithmus umgesetzt.

- Der MIS Algorithmus wird iterativ in *Runden* durchgeführt bis alle Knoten in Teilmengen von V verteilt sind und obige Bedingungen erfüllen
  - In jeder Runde wird ein Knoten mit der höchstrangingen (auf S bezogenen) lokalen Identifikationsnummer ausgewählt und hinzugefügt.

[9]

input : each node $v \in V$ has a unique identifier $id(v)$
output: each node v knows whether it is in the MIS
Each node v performs the following actions concurrently with all other nodes:
send $id(v)$ to neighbours
receive neighbours' identifiers
Memorise neighbours with higher priority as $A := \{w \in \mathcal{N}_v \,|\, id(w) > id(v)\}$
repeat
  if $A = \emptyset$ **then**
    send 'join' to neighbours $\mathcal{N}_v$
    join MIS and terminate
  if *receive 'join' message* **then**
    send 'not join' to neighbours $\mathcal{N}_v$
    do not join MIS and terminate
  receive 'not join' messages
  $A = A \setminus \{w \in \mathcal{N}_v \,|$ received 'not join' from $w\}$
until *termination*;

Alg. 4. Naiver MIS Algorithmus

Bei der Ausführung von dem naiven MIS Algorithmus tritt jeder noch aktive Knoten mit einer lokal maximalen Kennung in der nächsten Runde der unabhängigen Menge bei.

- Da es niemals zwei Nachbarn geben kann, die eine lokal maximale Kennung haben, kann dies niemals zu einem Konflikt führen.
- Alle Nachbarn solcher Knoten enden in der nachfolgenden Runde, um sicherzustellen, dass auch später keine Konflikte auftreten.
- Für die Knoten ist sichergestellt dass sie terminieren und dann mindestens einen Nachbarn in der unabhängigen Menge haben, d.h. sie können nicht Teil einer unabhängigen Menge sein, die bereits verbundene Knoten enthält.
- Die Laufzeit dieses Algorithmus wird durch den längsten Pfad von Knoten mit absteigender Kennung angegeben. Dieser ist höchstens $D+1$, wobei D den Durchmesser des Graphen bezeichnet, d.h. die maximale Länge eines kürzesten Pfades zwischen einem beliebigen Knotenpaar.

Laufzeiten von lokalisierten Algorithmen

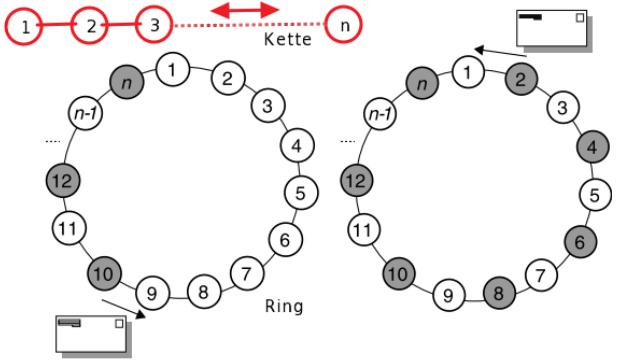

Abb. 6. Lokalisierte Algorithmen können hohe Ausführungszeiten besitzen, gerade bei Kettentopologien die nicht geschlossen sind

- N_V: Nachbarschaft eines Knotens, d.h., $N_V = \{w \in V | \{v,w\} \in E\}$

D kann groß sein — ein einfaches Netzwerk ist das Liniennetzwerk (auch als verknüpfte Liste oder Kette bezeichnet), das bereits den Durchmesser $n - 1$ hat! Im schlechtesten Fall nicht besser als eine sequentielle Lösung — ein einfaches lokal definierbares Problem wie das Finden eines MIS sollte eine Lösung ohne eine globale Kausalitätskette haben.

Kausalitäten sind Abhängigkeiten und Führen Synchronisation mit Blockierung ein, d.h. nicht alle Knoten rechnen.

Verteilte Algorithmen für Sensornetzwerke werden üblicherweise hinsichtlich ihrer zeitlichen Ausführungskomplexität, ihrer räumlichen Komplexität und ihrer Nachrichtenkomplexität bewertet.

Dies macht lokale Algorithmen besonders geeignet für Szenarien, in denen sich die Umgebung der Knoten häufig ändert, da sie sich ständig an die neuen Umstände anpassen können.

Verbesserter Algorithmus

Laufzeitverbesserung durch Randomisierung und Zulassung von Kollisionen (also fehlerhaften Verhalten). Beispiel ist auch Random Walk um einen Pfad A-B in einem Netzwerk durch zufällige Knotenauswahl in der Nachbarschaft zu erzeilen.

- Randomisierte Algorithmen terminieren prinzipiell immer. Und der mittlerer Fall ist meistens besser als bei deterministischen Algortihmen.
- Ansatz: Die Identifikationsnummern werden jetzt randomisiert ausgewählt
- Wenn die Zufallszahlen eine logarithmische Größe haben, ist die Wahrscheinlichkeit, dass zwei Zahlen gleich sind, vernachlässigbar. Es kann gezeigt werden, dass in jeder Iteration die erwartete Anzahl von Kanten neben Abschlussknoten ein konstanter Bruchteil der Gesamtzahl der verbleibenden Kanten ist.

- Da dies für jeden Graphen gilt, insbesondere für die durch die aktiven Knoten in jeder Iteration induzierten Untergraphen, impliziert dies eine erwartete Laufzeit von O(log n).

output: each node $v \in V$ knows whether is in the MIS [9]
Each node v performs the following actions concurrently with all other nodes:
repeat
  let $r(v)$ be a randomly chosen number
  send $r(v)$ to neighbours
  receive random numbers from all (non-terminated) neighbours $\mathcal{N}_v$
  if $r(v) > r(w)$ *for all* $w \in \mathcal{N}_v$ **then**
    send 'join' to $\mathcal{N}_v$
    join MIS and terminate
  if *receive 'join' message* **then**
    send 'not join' to $\mathcal{N}_v$
    do not join MIS and terminate
until *termination*;

Alg. 5. Randomisierter "best hope" MIS Algorithmus

Einfärbung (Colouring)

Bei einem Graphen $G = (V, E)$ ist eine k-Färbung von G eine Funktion $c : V \rightarrow \{1,..,k\}$, so dass für jede Farbe $i \in \{1,..,k\}$ die Menge $c^{-1}(i)$ unabhängig ist. D.h. zwei benachbarte Knoten v und w haben nicht die gleiche Farbe, d.h. $c(v) \neq c(w)$.

Def. 2. Färbungsproblem

- Es ist da Ziel, die Anzahl der Farben zu minimieren: Eine k-Färbung erlaubt jedem Knoten in k Runden zu agieren, ohne Nachbarn zu stören oder gestört zu werden
- Dies ist in Sensornetzwerken von großem Interesse, da es möglich ist, einen Zeitplan einzurichten, in dem alle Knoten einmal übertragen können, ohne Konflikte zu verursachen.
- Es ist schwierig, die Mindestanzahl von Farben zu bestimmen, die zum Einfärben eines Graphen erforderlich sind: Es ist NP-hartes Problem!

[9]

input : each node $v \in V$ has a unique identifier $id(v)$
output: each node v knows its colour $c(v) \in \{1, 2, 3\}$
Each node v performs the following actions concurrently with all other nodes:
$c(v) := id(v)$
for $\log^* n + \mathcal{O}(1)$ *rounds* **do**
  send $c(v)$ to counter-clockwise neighbour u
  receive $c(w)$ from clockwise neighbour w
  $i :=$ index of least significant differing bit of $c(v)$ and $c(w)$
  set the new colour to $c(v) := 2 \cdot i + c(v)[i]$
repeat
  send $c(v)$ to the two neighbours $\mathcal{N}_v = \{u, w\}$
  receive $c(u), c(w)$ from neighbours $\mathcal{N}_v$
  if $c(v) > \max\{c(u), c(w)\}$ **then**
    $c(v) := \min\{1, 2, 3\} \setminus \{c(u), c(w)\}$
    terminate and output $c(v)$
until *termination*;

Alg. 6. Einfärbung von Netzwerkknoten innerhalb einer Ringanordnung

Laufzeit und Kosten

Lokalisierte (nicht unbedingt lokale) verteilte Algorithmen zeigen geringe Kommunikationskomplexität, aber können eine große Laufzeit durch Nachrichtenvermittlung haben, gerade bei Ringstrukturen!

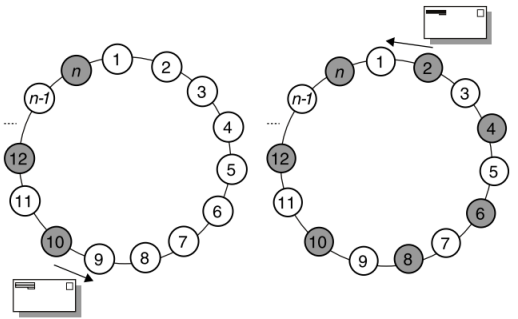

Abb. 7. Lokalisierter Algorithmus in einer Ring Struktur (z.B. vorheriger Färbungsalgorithmus)
→ Lange Laufzeiten!

Es ist zu beachten, dass lokale Algorithmen aufgrund der synchronen Phasen möglicherweise höhere Anforderungen an die Netzwerkschicht (Medium Access Layer) stellen als lokalisierte Algorithmen.

- Insbesondere in unzuverlässigen drahtlosen Netzwerken scheint es kostspielig zu sein, ein Medienzugriffskontrollschema zu implementieren, das synchrone Übertragungen ermöglicht, da Nachrichten aufgrund von Interferenzen (widersprüchlichen gleichzeitigen Übertragungen) oder Mobilität verloren gehen (selbst wenn die Knoten selbst nicht mobil sind, ist die Umgebung typischerweise dynamisch und aktiviert / deaktiviert vorübergehend Links).

Knotenidentifikation

Verteilte Algorithmen benötigen häufig unterscheidbare Knoten. Gibt es diese nicht können i.A. nur statistische Verfahren eingesetzt werden.

Knoten-ID

- Typischerweise kann davon ausgegangen werden, dass Knoten eindeutige Bezeichner haben. IDs könnten beispielsweise während der Bereitstellung mithilfe eines Zufallszahlengenerators generiert werden. Da RFID-Tags bereits über IDs verfügen, kann man annehmen, dass Sensorknoten während des Produktionsprozesses eine eindeutige ID erhalten.

Bestimmte Aufgaben können von keinem verteilten Algorithmus gelöst werden, wenn keine Bezeichner vorhanden sind, da Symmetrien zwischen den Knoten nicht durchbrochen werden kann.

- Ähnlich wie bei der Knotenverteilung im Raum sind die gebräuchlichsten Modelle für ID-Verteilungen Zufallsverteilungen und Worst-Case-Verteilungen.
- Manchmal kommt es auch darauf an, aus welchem Bereich die Bezeichner ausgewählt werden. Auch hier sind viele Variationen möglich. Beispielsweise kann jeder der n Knoten eine eindeutige 128-Bit-Kennung haben (Bereich 0,.., $2^{128}-1$). In einem restriktiveren Fall können die Knoten aufeinanderfolgende Nummern haben (z.B. Bereich 1,.., n).

Alternativ oder zusätzlich können Knoten IDs Standortinformationen enthalten - zum Beispiel, wenn die Knoten mit einem Global Positioning System (GPS) oder einem Galileo-Satellitenempfänger Gerät ausgestattet sind.
Standortinformationen können die Leistung bestimmter Vorgänge steigern: Beispielsweise kann ein Routing-Algorithmus geografische Informationen nutzen, um die Nachricht an einen Nachbarn weiterzuleiten, der in Richtung des Ziels der Nachricht liegt (Greedy Routing).

Zusammenfassung

- Verteilte Algorithmen in Sensornetzwerken können global, lokalisiert, und lokal ausgeführt werden
- Verteilte Algorithmen benutzen Nachrichten über uni- oder bidirektionale Kanäle
- Verteilte Algorithmen dienen der Koordination und Kooperation, Konsensfindung, Auflösung von Wettbewerbskonflikten, und der Netzwerk- und Gruppenbildung (Graphenalgorithmen)
- Wichtige Algorithmen: MIS/MDS/Colouring
- Viele Algorithmen nehmen an:
  1. Eindeutige Knotenkennungen (IDs)
  2. Synchrone Kommunikation ohne Ausfälle und Verfälschung
  3. Eventuell Zeitstempel ↠ Uhrensynchronisation!