Graphentheoretische Modellierung eines automatisierungstechnischen Echtzeitnetzwerks und Algorithmenentwurf zum Kommunikationsscheduling

Uwe Nowak

Universität Siegen

18.01.2007

Inhalt

Inhalt

Voraussetzungen der Arbeit

Es seien folgende Informationen gegeben:

- Die baumförmige Struktur eines Netzwerks.
- Die Art des Netzwerks (Unicast, Multicast, Broadcast).
- Ein Menge Γ von Requests zwischen Devices.

Dabei ist zu jedem Request r bekannt:

- Das Quelldevice.
- Das Zieldevice (Unicast) bzw. die Zieldevices (Multicast).
- Die Dauer $\delta(r)$ eines Requests.

Grundlegende Definitionen

Definition (Konflikt zweier Requests)

Zwei Requests haben einen Konflikt, wenn sie nicht gleichzeitig ausgeführt werden können, weil sie:

- In Halbduplex-Netzwerken Daten über dasselbe Netzwerkkabel übertragen.
- In Vollduplex-Netzwerken Daten über dasselbe Netzwerkkabel in dieselbe Richtung übertragen.

Definition (Schedule)

Ein Schedule α für eine Requestmenge R ist eine Abbildung, die jedem Request r eine Anfangszeit zuordnet, so dass keine zwei in Konflikt stehenden Requests zur selben Zeit ausgeführt werden.
Die Länge eines Schedules ist der Zeitpunkt, zu dem das letzte Request beendet ist.

Die Problemstellung

Finde einen Schedule möglichst kurzer Dauer für Γ.

Inhalt

Graphenfärbung

Definition (Knotenfärbung)

Zu einem Graphen $G = (V, E)$ ist eine Knotenfärbung eine Abbildung c der Knoten V in eine Farbmenge S, so dass keine zwei benachbarten Knoten dieselbe Farbe zugeordnet bekommen.

Definition (Kantenfärbung)

Zu einem Graphen $G = (V, E)$ ist eine Kantenfärbung eine Abbildung c der Kanten E in eine Farbmenge S, so dass keine zwei adjazenten Kanten dieselbe Farbe zugeordnet bekommen.

Definition (Chromatischer Index)

Der knotenchromatische Index χ bzw. kantenchromatische Index χ' ist die kleinste Anzahl von Farben, für die eine Knoten- bzw. Kantenfärbung existiert.

Graphenfärbung ist NP-vollständig

Problem:

- Die Bestimmung des knoten- bzw. kantenchromatischen Index ist NP-vollständig.

Lösung:

- Heuristiken.
- „Schnelle“ exakte Algorithmen.
- Betrachtung von Färbungen eingeschränkter Graphenfamilien.

Grundprinzipien der Färbungsalgorithmen

Färbungsalgorithmen basieren im Wesentlichen auf folgenden Prinzipien:

- Sequentielle Färbung (evtl. mit Umfärben).
- Unabhängige Mengen (bzw. Matchings bei Kantenfärbung).
- Die beiden Ansätze können mittels linearer Programmierung gelöst werden.

Ergebnisse für Knotenfärbung

De derzeit besten bekannten Ergebnisse für Knotenfärbung sind:

- Der beste polynomielle Algorithmus hat eine Approximationsgüte von

$$\mathcal{O}\left(\frac{|V|(\log\log|V|)^2}{(\log|V|)^3}\right).$$

- Exakte Knotenfärbung ist möglich in $\approx \mathcal{O}\left(2,4150^{|V|}\right)$.

In der Praxis:

- Heuristisch: Sequentielle Färbung nach der Degree-Saturation-Largest-First-Methode (DSATUR) in $\mathcal{O}(|V|^2)$.
- Exakt: Graphenfärbung mittels auf unabhängigen Mengen basierendem Ansatz unter Verwendung von linearer Programmierung und Spaltengenerierung.

Ergebnisse für Kantenfärbung

De derzeit besten bekannten Ergebnisse für Kantenfärbung sind:

- In $\mathcal{O}(|E|(\Delta(G)+|V|))$ lässt sich ein Graph mit $\lfloor 1,1 \cdot \chi'(G) + \frac{4}{5} \rfloor$ Farben kantenfärben. Da Graphen mit $\chi(G) \leq 2$ schneller exakt färbbar sind, lässt sich in dieser Zeit ein Graph mit maximal $\frac{4}{3}\chi'(G)$ Farben kantenfärben.
- Hypergraphen mit beschränkter Knotenanzahl lassen sich $\mathcal{O}(|E|)$ kantenfärben.
- Bipartite Graphen können in $\mathcal{O}(|E| \log \Delta(G))$ kantengefärbt werden.

Inhalt

Modellierung des Problems

- Das Netzwerk wird als baumförmiger, ungerichteter Graph $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ modelliert.
  - Die Switches sind die inneren Knoten.
  - Die Devices sind die Blätter.
  - Die Kabel sind die Kanten.
- Für ein Unicast-Request r ist die Kommunikationslinie $KL(r)$ der eindeutig bestimmte Weg von Quelldevice zum Zieldevice.
- Für ein Multicast-Request r ist der Kommunikationsbaum $KB(r)$ die Menge aller Kommunikationslinien vom Quelldevice zu den Zieldevices von r.

Ergebnisse für Requests

Sei das Netzwerk kein *VD-MC*-Netzwerk. Dann gilt

Theorem (Lokale Konfliktfreiheit äquivalent globale Konfliktfreiheit)

Enthalten zwei Requests Knoten aus einer zusammenhängenden Menge V (z.B. einen einzelnen Knoten v), so haben sie genau dann einen globalen Konflikt, wenn sie einen Konflikt in einem Knoten aus V haben.

Theorem (Konfliktknoten bilden Weg bzw. Baum)

Die Menge der Knoten und Kanten, in denen zwei Requests einen Konflikt haben, bilden in Halbduplex-Netzwerken einen Weg und in Vollduplex-Netzwerken einen Baum.

Theorem (Konflikt in benachbarten Mengen)

Es seien V und W benachbarte zusammenhängende Mengen. Enthält ein Request r Knoten aus V und ein Request s Knoten aus W, so sie genau dann einen Konflikt, wenn sie einen Konflikt in V oder W haben.

Schedules

Definition (Synchrone Schedules)

Zwei Schedules α für R und β für S heißen synchron, wenn

- sie Requests, die in beiden Mengen R und S vorkommen, dieselbe Startzeit zuordnen.
- es keine in Konflikt stehenden Requests in R und S gibt, die bezüglich α bzw. β parallel ausgeführt werden.

Definition (Synchrone Vereinigung)

Sind α und β synchrone Schedules für R und S, so kann man die synchrone Vereinigung von α und β definieren. Dieses ist ein Schedule für $R \cup S$, der

- Requests aus R zur selben Zeit wie α
- Requests aus S zur selben Zeit wie β

ausführt.

Ergebnisse für Schedules

Theorem

Die maximale Lastdauer (d.h. die maximale Belegungsdauer einer Kante) ist eine untere Schranke für die Länge eines optimalen Schedules.

Theorem

Es gebe für jeden Knoten $v \in \mathbb{V}$ einen Schedule α_v. Dann sind genau dann alle Schedules α_v (paarweise) synchron, wenn je zwei benachbarte Schedules synchron sind.

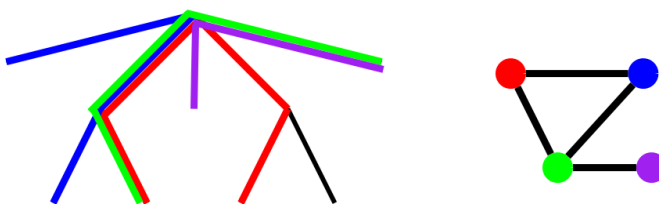
Knotenkonfliktgraph

Man versucht, die Konfliktstruktur einer Requestmenge R in einem Graphen $G = (V, E)$ abzubilden.

Definition (Knotenkonfliktgraph)

Der Knonfliktgraph enthält als Knoten gerade die Requests R und zwei Requests sind genau dann durch eine Kante verbunden, wenn sie einen Konflikt haben.

Ein solcher Graph ist für jede Requestmenge R möglich.



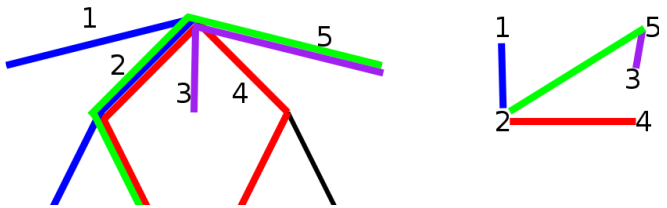
Kantenkonfliktgraph

Für Nicht-*VD*-*MC*-Netzwerke und Requests R durch einen Knoten v lässt sich der Kantenkonfliktgraph definieren.

Definition (Kantenkonfliktgraph)

Der Kantenkonfliktgraph enthält die Requests von R als Kanten und die im Netzwerk zu v inzidenten Kanten als Knoten. Dabei sind zwei Kanten im Konfliktgraph genau dann adjazent, wenn sie einem im Netzwerk zu v inzidente Kante gemeinsam (in *VD*-Netzwerken in dieselbe Richtung) enthalten.

Für *HD*-*MC*-Netzwerke ist dieses ein Hypergraph!



Erstellung der Konfliktgraphen

Theorem (Komplexität der Erzeugung der Konfliktgraphen)

- *Die Erstellung eines lokalen Knotenkonfliktgraphen für die Requests durch eine Menge v dauert* $\mathcal{O}(|V| + |E|)$.
- *Die Erstellung eines globalen Knotenkonfliktgraphen für alle Requests durch eine Menge v dauert* $\mathcal{O}(|\Gamma|^2 \cdot I(\mathcal{G}))$.

Theorem (Komplexität der Erzeugung der Konfliktgraphen)

- *Die Erstellung eines lokalen Knotenkonfliktgraphen für die Requests durch eine Menge v dauert* $\mathcal{O}(|V| + |E|)$.

Inhalt

Schedules für Requests beliebiger Länge

Für ein beliebiges Netzwerk sei der Knotenkofliktgraph für eine Requestmenge R gegeben. Dann

Theorem (First-Fit)

- *First-Fit erzeugt einen Schedule in $\mathcal{O}(|R|^2)$.*
- *Decreasing-First-Fit ist empirisch besser und erzeugt einen Schedule in $\mathcal{O}(|R|^2)$.*

Theorem (List-Scheduling)

- *List-Scheduling erzeugt einen Schedule in $\mathcal{O}(|R|^2)$.*

Theorem

List-Scheduling hat für Sterne eine Approximationsgüte von 2.

Trennlevel

Rekursiv werden den Knoten $\mathbb{V}$ Trennlevel zugewiesen.

- Wähle einen Knoten $v \in \mathbb{V}$, so dass nach dem Entfernen von v jede Zusammenhangskomponente maximal $\frac{|\mathbb{V}|}{2}$ Knoten enthält. Dieser Knoten v hat den Trennlevel 0.
- Fahre mit den so entstandenen Zusammenhangskomponenten $G_1, \ldots, G_k$ entsprechen fort. Die dort gewählten Knoten haben den Trennlevel 1.

Jeder Knoten v erhält einen Trennlevel $t(v)$ aus $\{0, \ldots, \lfloor \log_2 |\mathbb{V}| \rfloor\}$.

Level der Requests

Nun werden allen Requests Level zugewiesen:

- Das Level eines Requests r ist der minimale Trennlevel eines in r enthaltenen Knoten.
- Jedes Request enthält genau einen Knoten mit minimalem Trennlevel. Dieser wird als Wurzel bezeichnet.

Dann folgt:

- Zwei Requests mit derselben Wurzeln haben genau dann einen Konflikt, wenn sie diesen in der Wurzel haben.
- Zwei Requests mit demselben Level und unterschiedlichen Wurzeln sind konfliktfrei.
- Das Schedulen aller Requests mit derselben Wurzel entspricht dem Schedulen eines Sterns.
- Das Schedulen aller Requests desselben Levels entspricht dem Schedulen disjunkter Sterne.

Der Algorithmus List-Scheduling-Levels

Der Algorithmus folgt:

- Schedule alle Requests mit Level 0.
- Schedule alle Requests mit Level 1 und hänge diesen Schedule an den bestehenden Schedule an.
- Fahre fort bis zu den Requests mit Level $\lfloor \log_2 |\mathbb{V}| \rfloor - 1$.
- Schedule alle Requests mit Level $\lfloor \log_2 |\mathbb{V}| \rfloor$ zum Zeitpunkt 0.

Eigenschaften des Algorithmus

- Sei $l = \lfloor \log_2 |\mathbb{V}| \rfloor$ und β_k der Schedule für die Requests mit Level k. Dann gilt für die Länge $|\alpha|$ des erzeugten Schedules

$$
\begin{aligned}
|\alpha| &= \max\left(\sum_{k=0}^{l-1} |\beta_k|, \max\{\delta(r) : r \in R^l\}\right) \\
&\leq \max\left(\sum_{k=0}^{l-1} 2OPT(R^k), OPT(R^l)\right) \\
&\leq \max\left(2l \cdot \max_{0 \leq k \leq l-1} OPT(R^k), OPT(R^l)\right) \\
&\leq \max\left(2l \cdot \max_{0 \leq k \leq l-1} OPT(\Gamma), OPT(\Gamma)\right) = 2l \cdot OPT(\Gamma).
\end{aligned}
$$

  Somit hat der Algorithmus eine Approximationsgüte von $2\lfloor \log_2 |\mathbb{V}| \rfloor$.
- Der Algorithmus hat eine Komplexität von $\mathcal{O}(|\Gamma|)^2$.

Inhalt

Schedules für Requests einheitlicher Länge

Es wird vorausgesetzt, dass alle Requests die Länge 1 haben.

Definition (Klassen eines natürlichen Schedules)

Die Menge aller Requests, die in einem natürlichen Schedule gleichzeitig ausgeführt werden, bezeichnet mal als Klasse.

Theorem

Es gibt einen optimalen natürlichen Schedule.

Synchronisierbarkeit

Definition (Synchronisierbarkeit)

Zwei Schedules heißen synchronisierbar, wenn ihre Klassen so permutiert werden können (ggf. unter Verlängerung eines Schedules), so dass die permutierten Schedules synchron sind.

Theorem

In Halbduplex-Netzwerken sind je zwei Schedules für benachbarte Knotenmengen synchronisierbar.

In Vollduplex-Netzwerken ist diese Aussage leider falsch.

Eigenschaften der Synchronisierbarkeit

Theorem

Seien R und S konfliktfreie Mengen von Requests (z.B. Requests durch benachbarte zusammenhängenden Knotenmengen). Dann sind für Schedules α für R und β für S äquivalent:

- *Die Schedules α und β sind synchronisierbar.*
- *Requests aus $R \cap S$ werden genau dann gleichzeitig in α ausgeführt, wenn sie gleichzeitig in β ausgeführt werden.*
- *Die Klassen des kürzeren Schedules lassen sich so permutieren, dass der kürzere Schedule synchron zu dem längeren Schedule ist.*

Ist o.B.d.A. β der kürzere Schedule, so kann die letzte Permutation in $\mathcal{O}(|S| + |\alpha|)$ berechnet werden.

Lokale Schedules mittels Färbung

Zur Erinnerung: Sei R eine Menge von Requests. Dann gilt:

- Der Knotenkonfliktgraph C für R ist gerade so konstruiert, dass zwei Requests aus R genau dann einen Konflikt haben, wenn sie in C benachbart sind.
- Der Kantenkonfliktgraph C für R ist (falls existent) gerade so konstruiert, dass zwei Requests aus R genau dann einen Konflikt haben, wenn sie in C adjazent sind.

Somit folgt:

- Eine Knotenfärbung des Knotenkonfliktgraphen bzw. eine Kantenfärbung des Kantenkonfliktgraphen ist äquivalent zu einem Schedule für R.

Algorithmus für Halbduplex-Netzwerke

Für Halbduplex-Netzwerke ergibt sich folgender Algorithmus:

- Erzeuge für jeden Knoten den Knoten oder Kantenkonfliktgraphen.
- Erzeuge mittels Färbung einen Schedule.
  - Für Unicast-Netzwerke mit Approximationsgüte $\frac{4}{3}$ in $\mathcal{O}(|E|(\Delta(G) + |V|))$ möglich.
  - Exakt in $\mathcal{O}(|E|)$ möglich, wenn das baumförmige Netzwerk beschränkten Grad hat.
- Synchronisiere von einem beliebigen Knoten aus beginnend in Breiten- oder Tiefensuche alle Schedules und vereinige diese.

Es ergibt sich:

- Der Algorithmus hat für Unicast-Netzwerke eine Laufzeit von $\mathcal{O}(|\Gamma|^2 \cdot I(\mathcal{G}))$ und eine Approximationsgüte von $\frac{4}{3}$.
- Bei beschränktem Knotengrad hat der Algorithmus eine Laufzeit von $\mathcal{O}(|\Gamma| \cdot I(\mathcal{G}))$ und ist exakt.

Übertragung auf Vollduplex-Netzwerke nicht möglich

Es werden nur Unicast-Netzwerke betrachtet. Dann gilt:

- Der Kantenkonfliktgraph ist bipartit.
- Er lässt sich in $\mathcal{O}(|E| \log \Delta(G))$ exakt färben.

Aber:

- Die so erzeugten Schedules sind nicht synchronisierbar!

Algorithmus für Vollduplex-Netzwerke

Mögliche Lösungsalternativen:

- Färbe für jeden Knoten iterativ die Kanten (=Requests) des Kantenkonfliktgraphen so, dass bereits geschedulete Requests nicht umgefärbt werden.
- Erzeuge einen modifizierten Knotenkonfliktgraphen, der (mittels Äquivalenzklassenbildung) eine Synchronisierbarkeit der Schedules sicherstellt.

Für die Laufzeit ergibt sich:

- Im ersten Fall bei linearem Kantenfärbungsalgorithmus eine Laufzeit von $\mathcal{O}(|\Gamma| I(\mathcal{G}))$.
- Im zweiten Fall bei linearem Knotenfärbungsalgorithmus eine Laufzeit von $\mathcal{O}(|\Gamma| I(\mathcal{G}) + \sum_{v \in \mathbb{V}} \#\{\text{Konflikte in } v\})$.

In beiden Fällen wurde die Approximationsgüte nicht bestimmt.

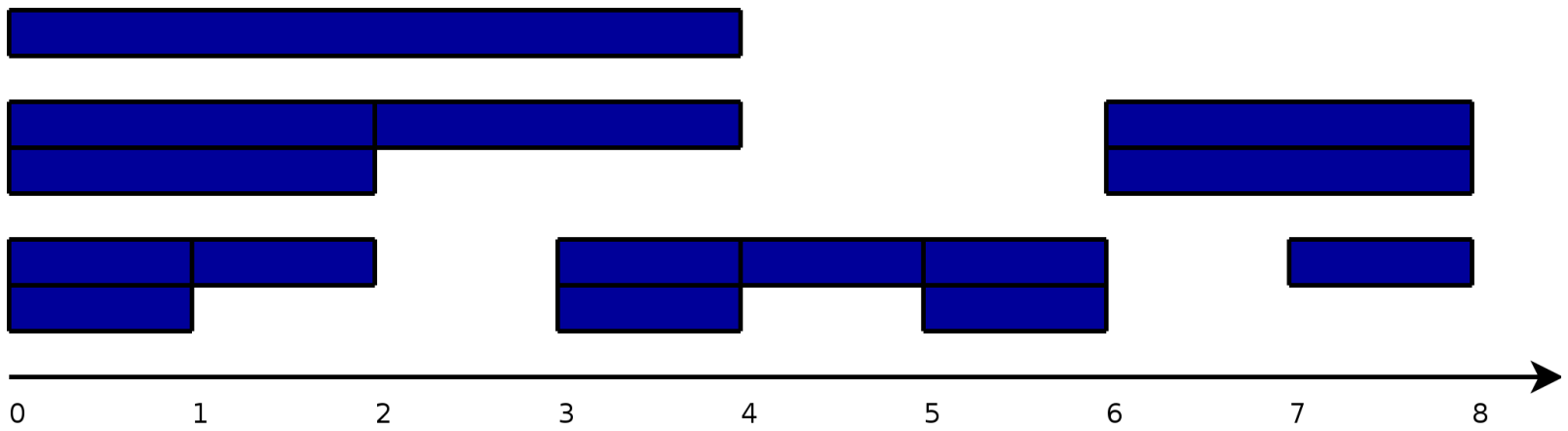
Inhalt

Schedules für Requests exponentiell gestaffelter Länge

Es wird vorausgesetzt, dass alle Requests die Länge 2^n, $n \in \{0, \ldots, N\}$ haben.

Definition (Gestaffelter Schedule)

Ein Schedule heißt gestaffelt, wenn für jedes Request die Startzeit ein vielfaches der Requestdauer ist.

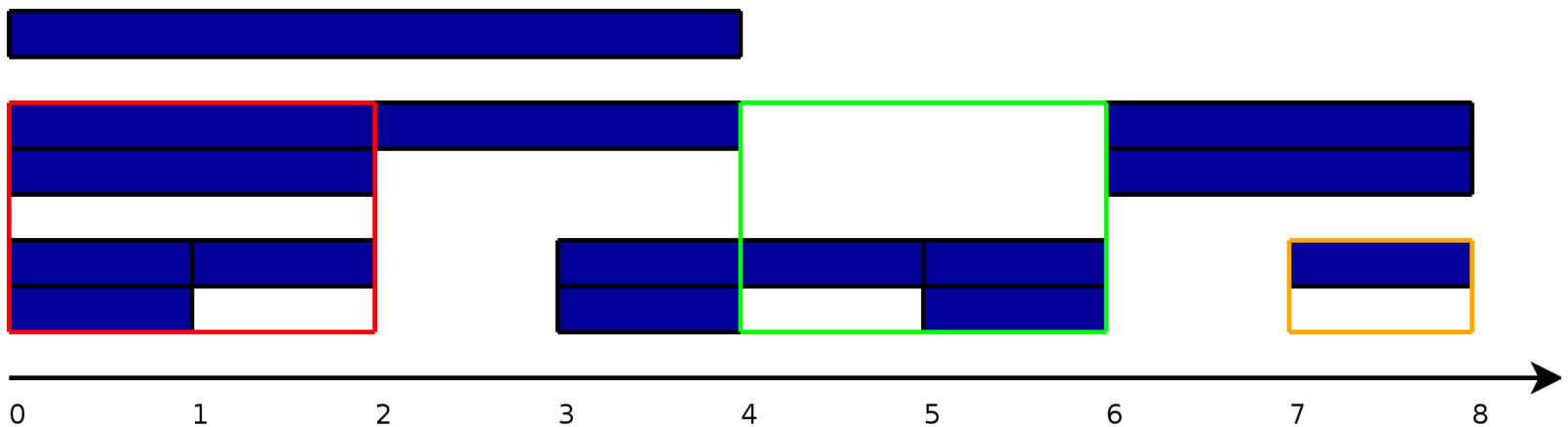


Bemerkung

Es gibt nicht immer einen optimalen gestaffelten Schedule.

Slots

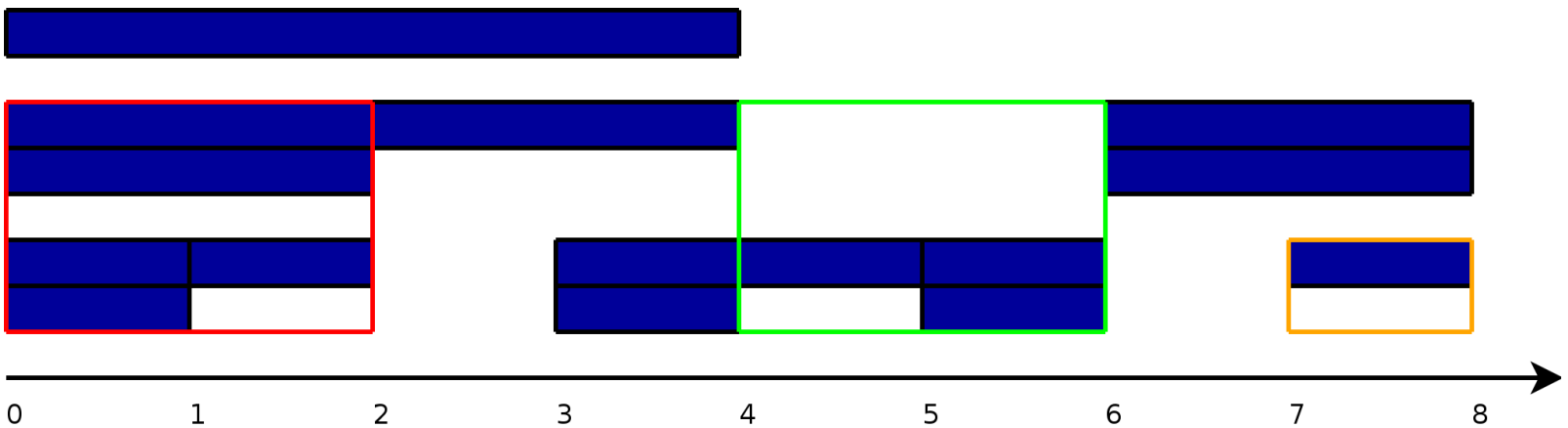
Der Begriff des Slots ist formal etwas technisch definiert. Anschaulich ist ein Slot der Länge 2^n die Menge der Requests der Länge kleiner als 2^n, die parallel ausgeführt werden.



Ein gestaffelter Schedule kann nun dadurch definiert werden, dass alle Requests in Slots eingeordnet werden.

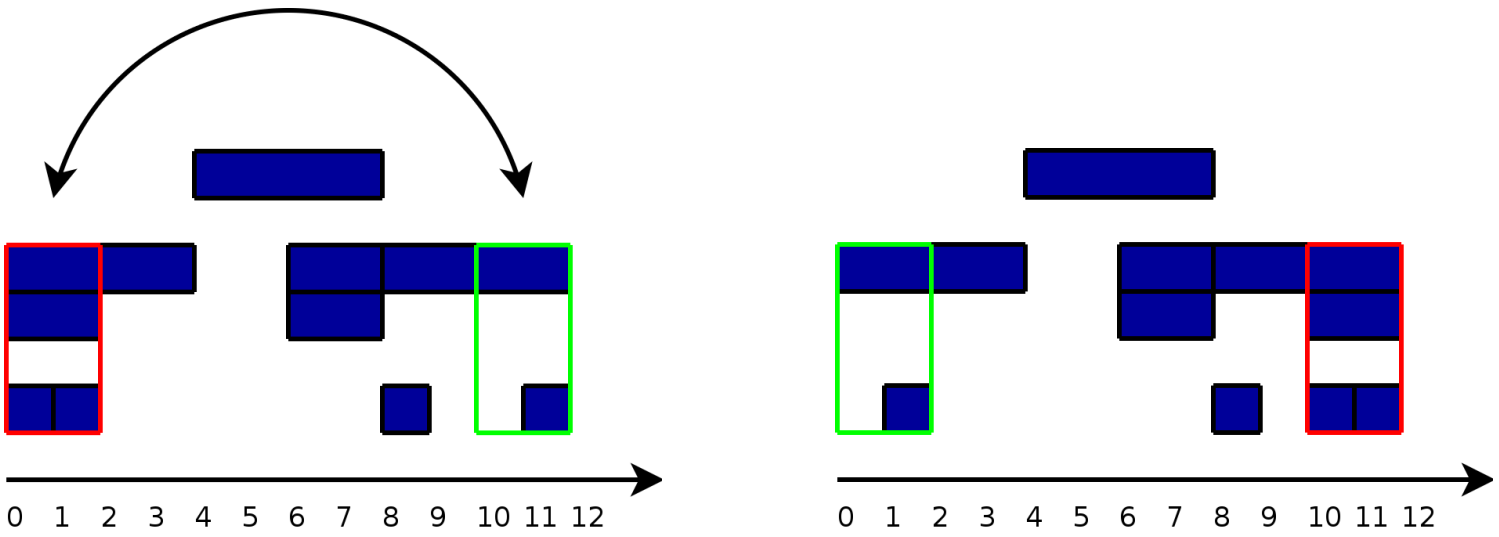
Bezeichnungen für Slots

- Ein Slot heißt leer, wenn er kein Request der Slotlänge enthält und rekursiv leer, wenn er kein Request enthält. Ein nicht leerer Slot heißt gefüllt, ein nicht rekursiv leerer Slot heißt rekursiv gefüllt.
- Ein Slot heißt frei, wenn er nicht innerhalb eines gefüllten Slots liegt.



Vertauschen freier Slots

Vertauscht man in einem Schedule zwei freie Slots gleicher Länge, so entsteht ein anderer Schedule.



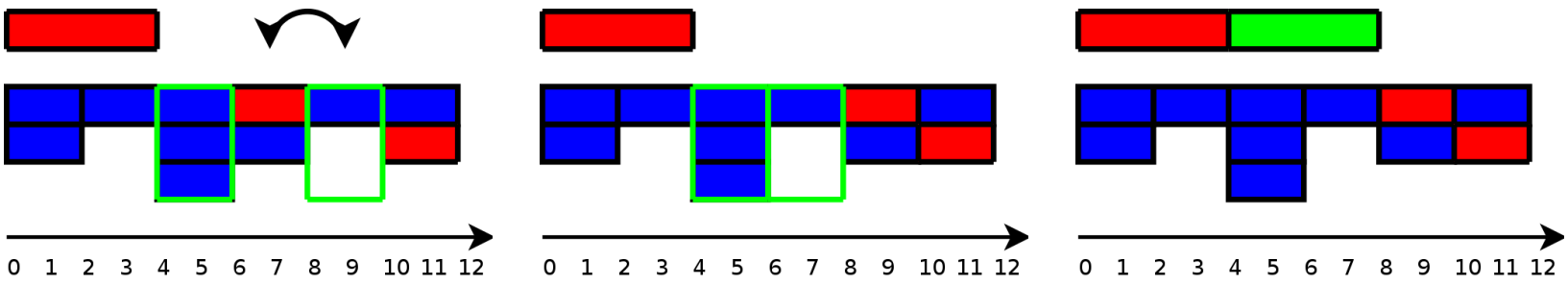
Multidimensionaler Greedy

Skizze des Algorithmus zur Erstellung eines globalen Schedules unter Verwendung eines globalen Konfliktgraphen.

- Erstelle mit den Techniken des vorherigen Kapitels einen Schedule für die Requests einer Länge 2^n
- Erweitere diesen Schedule sukzessive auf Requests der Länge $2^{n+1}, 2^{n+2}, \ldots, 2^N$.
  Ist ein Request (oder ein Block von Requests) der Länge 2^k zu Schedulen, dann:
  - Gibt es einen gefüllten Slot der Länge 2^k, in den das Request geschedulet werden kann, so tue dies.
  - Sonst, so schaue, ob es freie Slots der Länge 2^{k-1} gibt, die nicht mit dem zu schedulenden Request in Konflikt stehen. Falls ja, so tausche diese Slots nebeneinander (siehe kommende Folien).
  - Sonst schedule das Request an das Ende des Schedules.
- Erweitere diesen Schedule sukzessive mittels First-Fit auf Requests der Länge $2^{n-1}, 2^{n-2}, \ldots, 2^0$.

Erweiterung eines Schedules auf Requests größerer Länge

Ein Request r der Länge 2^k sei zu schedulen.
Die Idee: Durch das vertauschen freier Slots der Länge 2^{k-1} entsteht ein Slot der Länge 2^k, in dem kein Request einen Konflikt zu dem zu schedulenden Request hat.



Das Vorgehen:

- Erstelle einen leeren gestaffelten Schedule β.
- Fülle alle mit r in Konflikt stehende Requests in β ein.
- Suche den ersten rekursiv leeren Slot der Länge 2^k in β.

Laufzeitkomplexität

Im Folgenden sei $d(r)$ die Anzahl der mit r in Konflikt stehenden Requests.

- Das Einfügen aller zu r in Konflikt stehenden Requests in einen neuen Schedule dauert $\mathcal{O}(d(r))$.
- Durch Vertauschen der gefüllten Slots der Länge 2^k an den Anfang kann sichergestellt werden, dass das Suchen eines zu r konfliktfreien gefüllten Slots eine Komplexität von $\mathcal{O}(d(r))$ hat.
- Das Suchen zweier freier und zu r konfliktfreier Slots der Länge 2^{k-1} dauert $\mathcal{O}(d(r))$.

Summiert über alle Requests aus Γ folgt eine Laufzeit von $\mathcal{O}(|\Gamma| + \#\{\text{Konflikte}\})$.

Multidimensionaler DSATUR und TSATUR

Definition (Sättigungsgrad, Sättigungszeit)

- Der Sättigungsgrad eines Requests r der Länge 2^n ist die Anzahl der Slots der Länge 2^n, zu der ein zu r in Konflikt stehendes Request aktiv ist.
- Die Sättigungszeit eines Requests r ist die Gesamtzeit, zu der ein zu r in Konflikt stehendes Request aktiv ist.

Mit zusätzlichem Aufwand ist es in Anlehnung an die DSATUR-Färbung möglich, in jedem Schritt das Request mit maximalem Sättigungsgrad bzw. maximalem Sättigungszeit zu wählen. Die Komplexität des multidimensionalen DSATUR bzw. des multidimensionalen TSATUR ist $\mathcal{O}(|\Gamma|^2)$.