Karlsruher Institut für Technologie

Scheduling im $SINR_G$-Modell

Algorithmen für Sensornetzwerke - Christof Doll

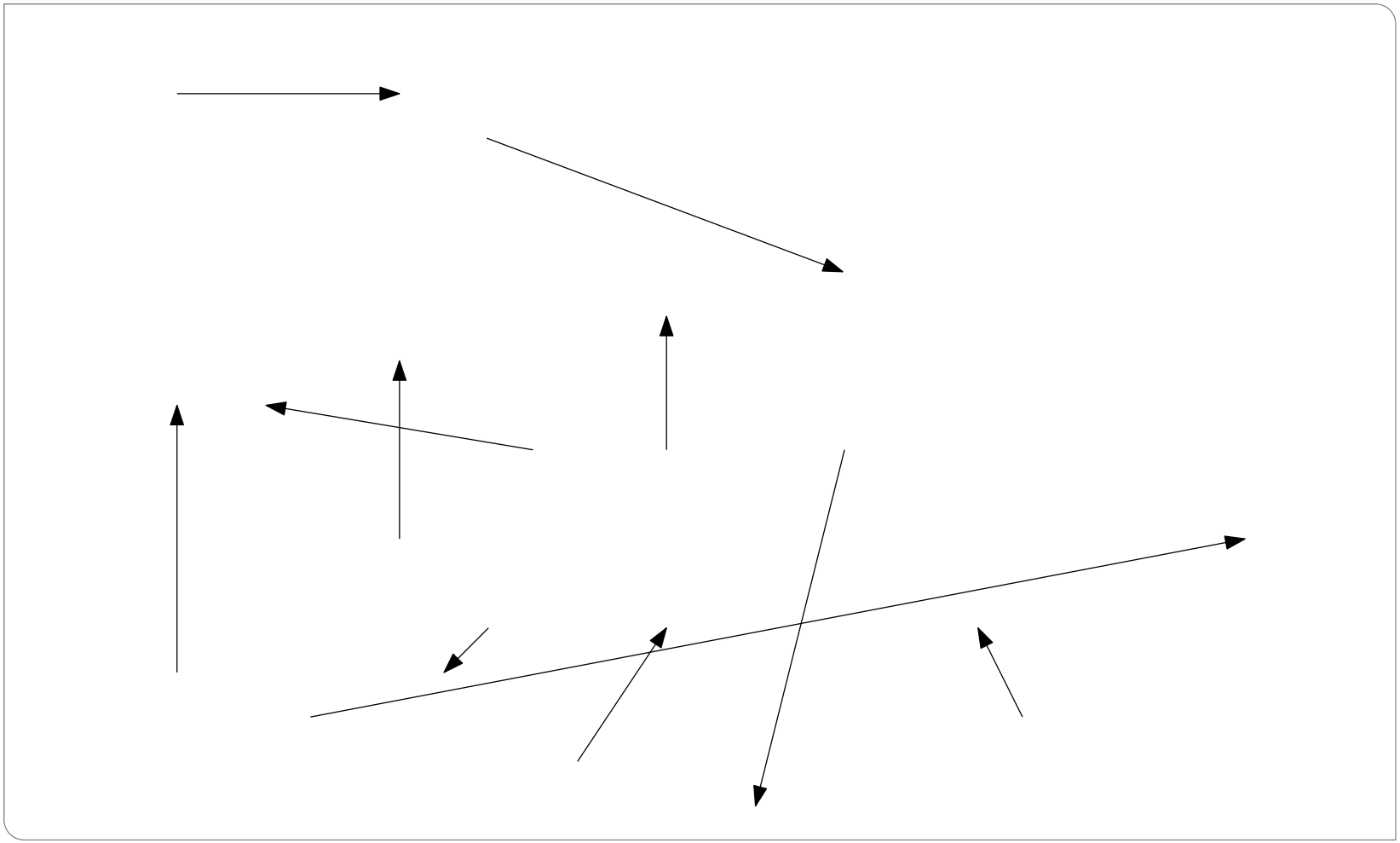
Das Modell $SINR_G$

Definition der Probleme

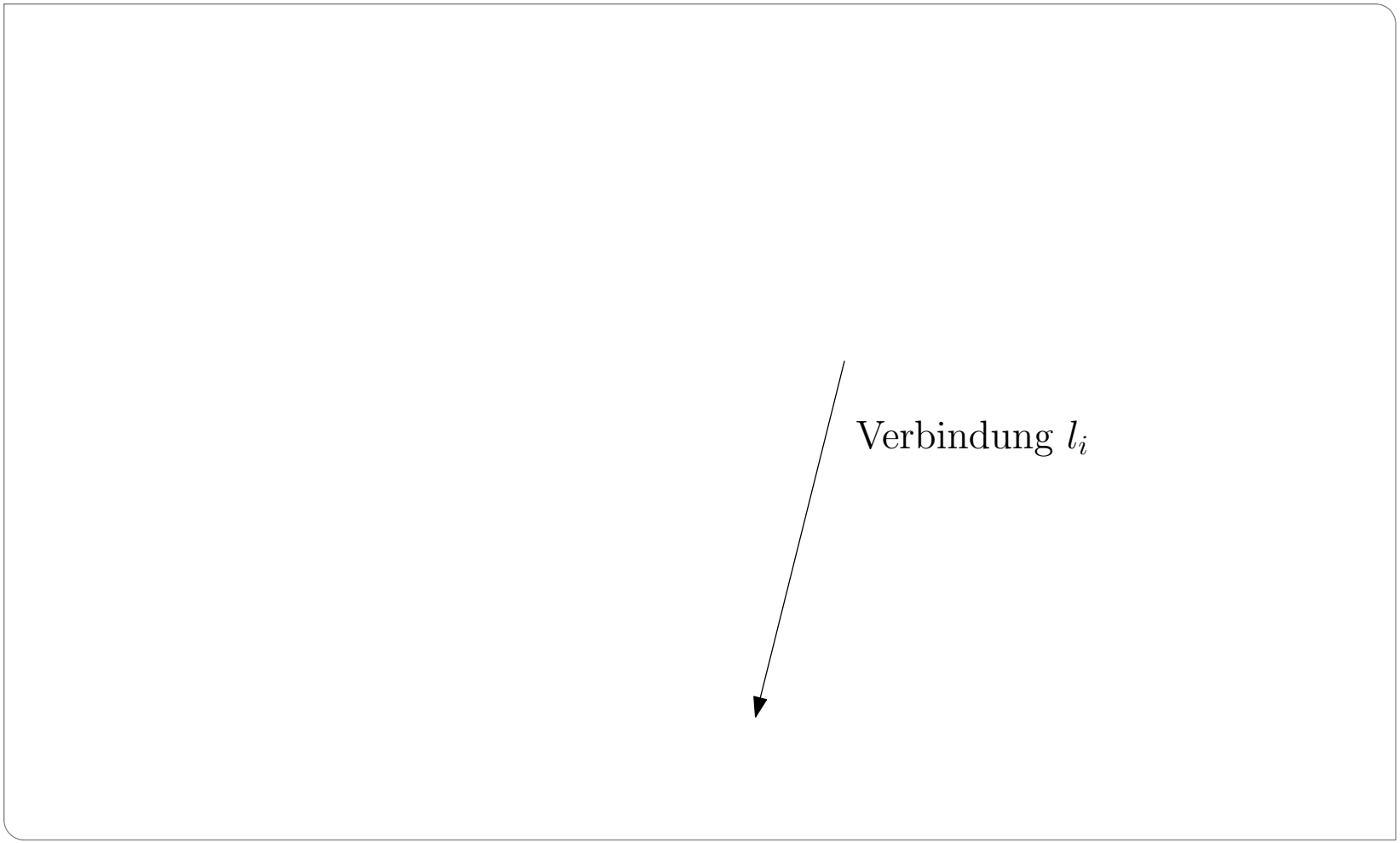
$\mathcal{NP}$-Schwere-Beweis

Approximationsalgorithmen

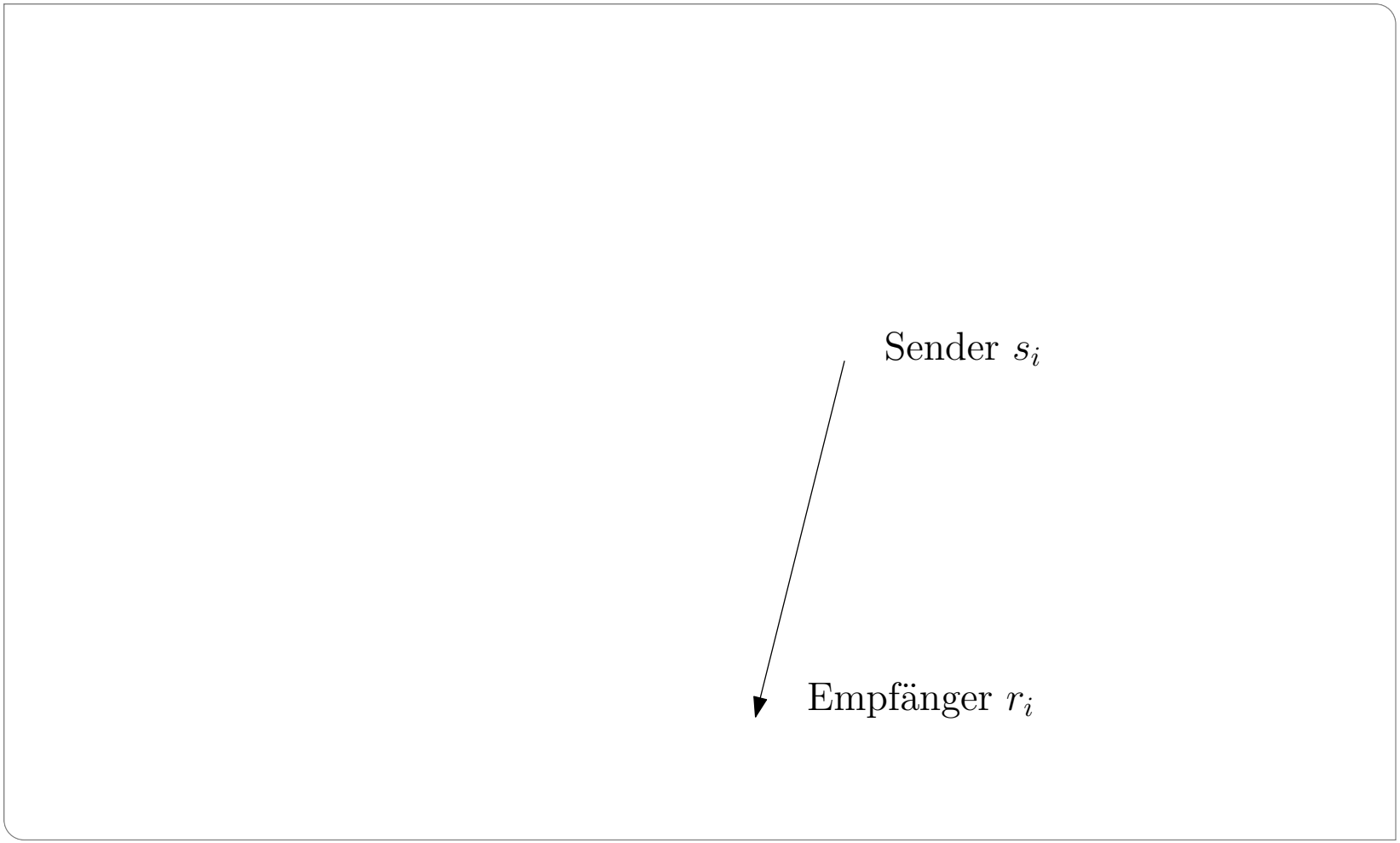
Das Modell $SINR_G$



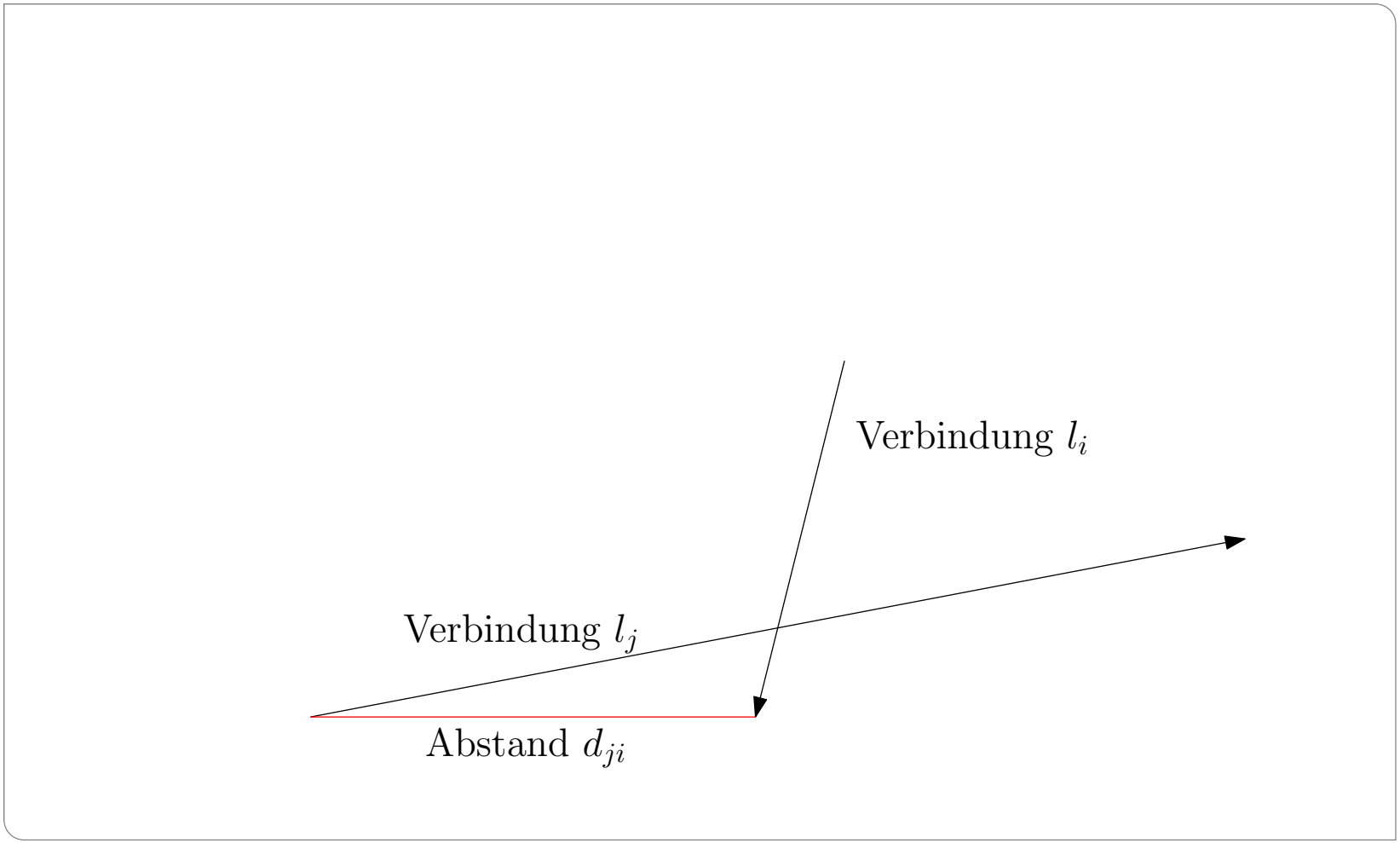
Das Modell $SINR_G$



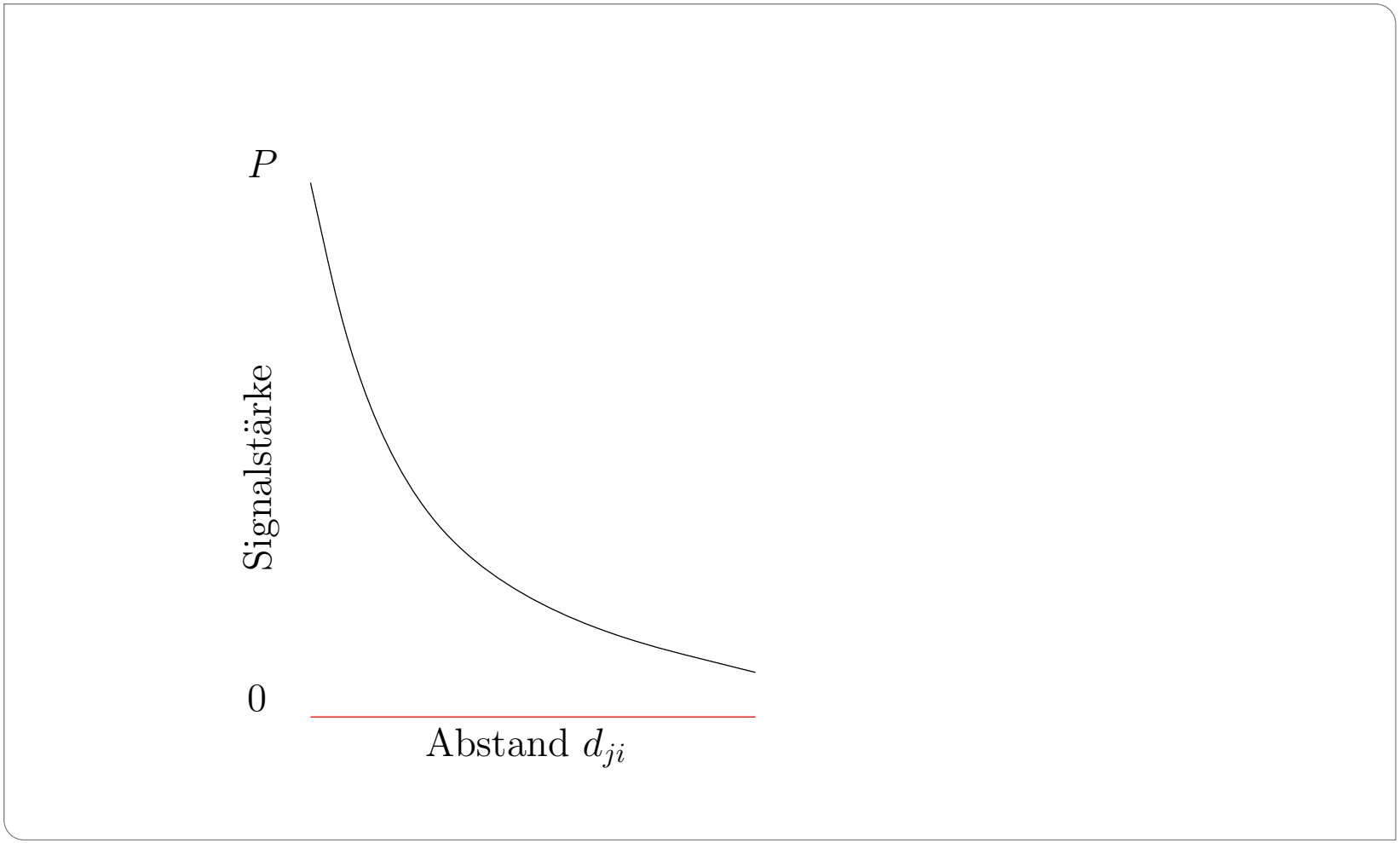
Das Modell $SINR_G$



Das Modell $SINR_G$



Das Modell $SINR_G$



Das Modell $SINR_G$

Definition

- Empfangsstärke $P_{ji} := \frac{P}{d_{ji}^{\alpha}}$ mit Pfadverlust-Exponent $\alpha > 2$
- Rauschen N (hier $N = 0$)
- $SINR_G(r_i) := \frac{P_{ii}}{\sum_{j \neq i} P_{ji} + N}$
- $SINR_G(r_i) \geq \beta \Leftrightarrow$ Übertragung l_i erfolgreich
- Schwellenwert $\beta > 1$ abhängig von der Hardware

Das Modell $SINR_G$

Vorteile

- Signalstärke hängt von Distanz ab
- Modellierung verschiedener Szenarien durch Pfadverlust-Exponent

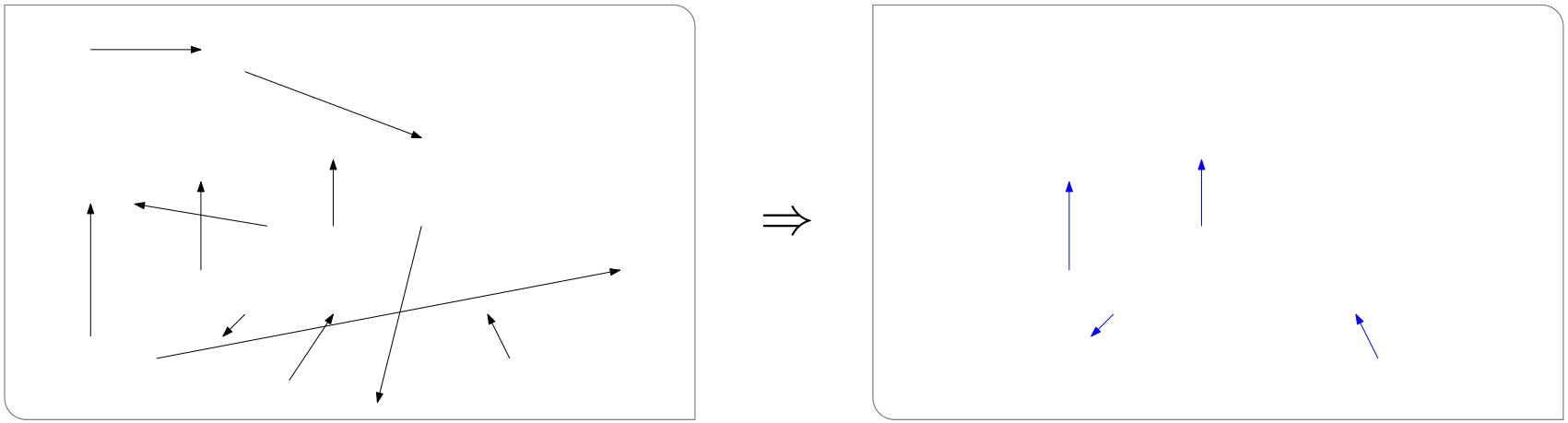
Nachteile

- Keine Modellierung von Wänden u.ä.
- Interferenz auf beliebig große Distanz
- Oft sind die Distanzen nicht bekannt

Scheduling-Probleme

One-Slot-Scheduling

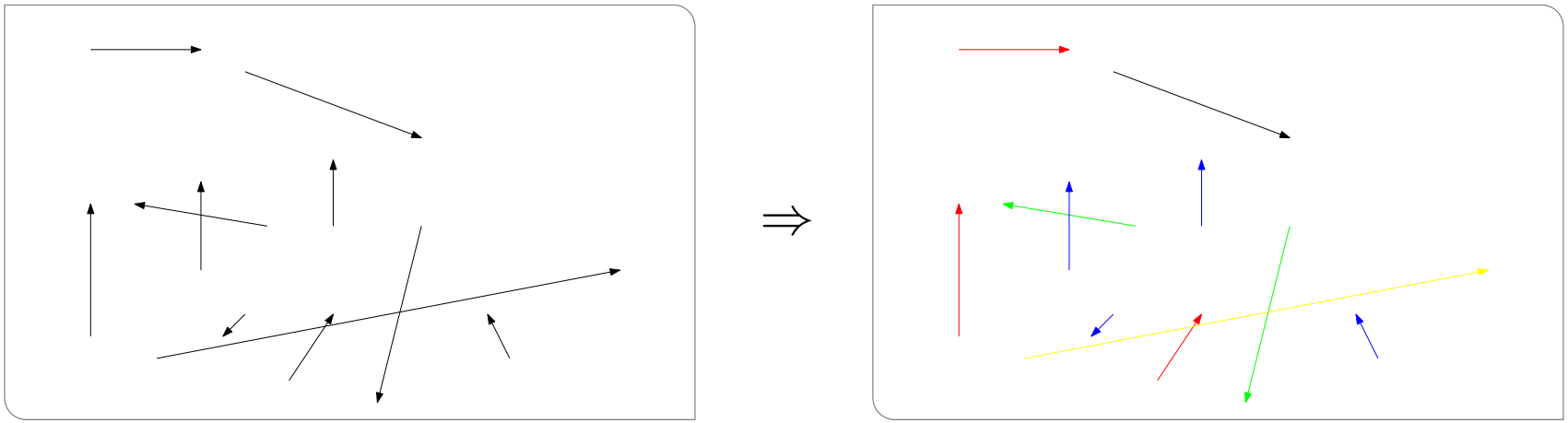
- $L = \{l_1, \ldots, l_n\}$
- Finde $L' \subseteq L$:
  - $\forall l \in L' : SINR_G(l) \geq \beta$
  - $|L'| \stackrel{!}{=} \max$



Scheduling-Probleme

SCHEDULING

- $L = \{l_1, \ldots, l_n\}$
- Finde Schedule $S = \{S_1, \ldots, S_T\}$
  - $S_1 \cup \ldots \cup S_T = L$
  - $\forall t : \forall l \in S_t : SINR_G(l) \geq \beta$
  - $T \stackrel{!}{=} \min$



SCHEDULING ist $\mathcal{NP}$-schwer

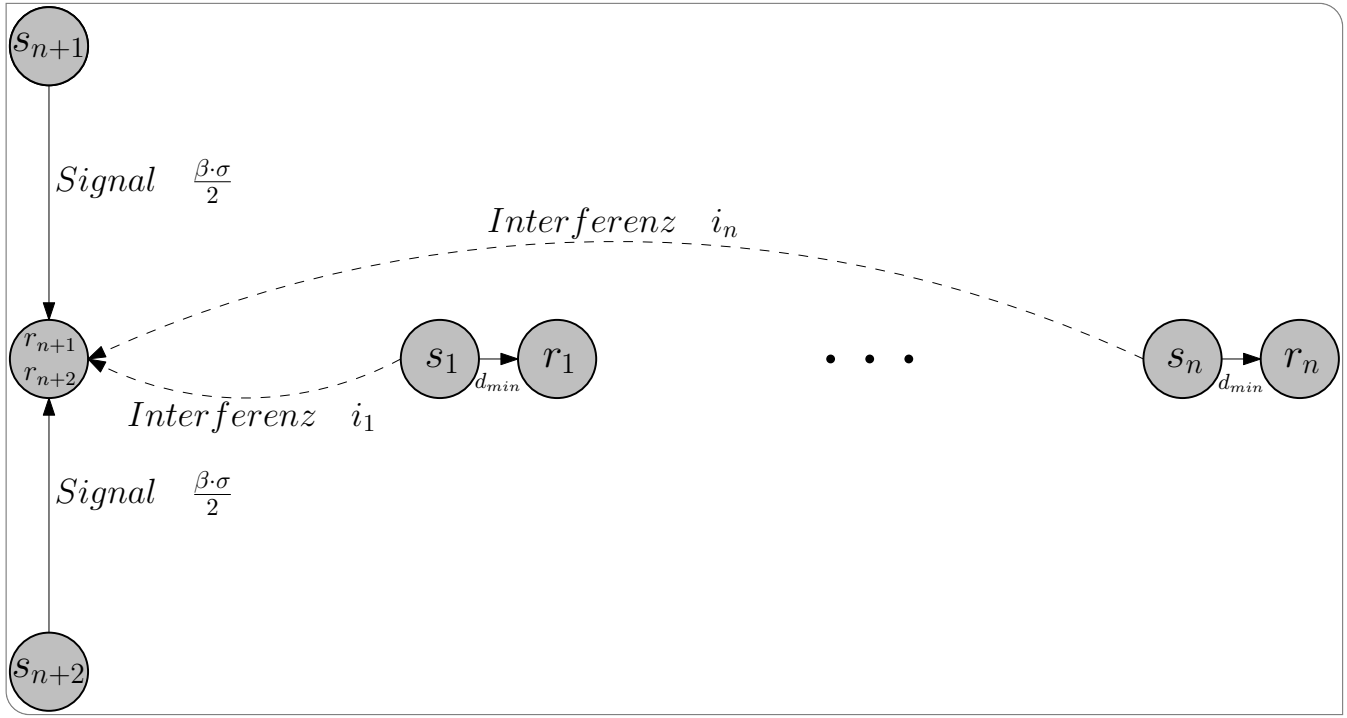
SCHEDULING ist $\mathcal{NP}$-schwer

- PARTITION $\propto$ SCHEDULING

Definition

- $\mathcal{I} = \{i_1, \ldots, i_n\}$
- Finde $\mathcal{I}_1, \mathcal{I}_2 \subseteq \mathcal{I}$
  - $\mathcal{I}_1 \cap \mathcal{I}_2 = \emptyset$
  - $\mathcal{I}_1 \cup \mathcal{I}_2 = \mathcal{I}$
  - $\sum_{i \in \mathcal{I}_1} i = \sum_{i \in \mathcal{I}_2} i = \frac{1}{2}\sigma$ mit $\sigma = \sum_{i \in \mathcal{I}} i$

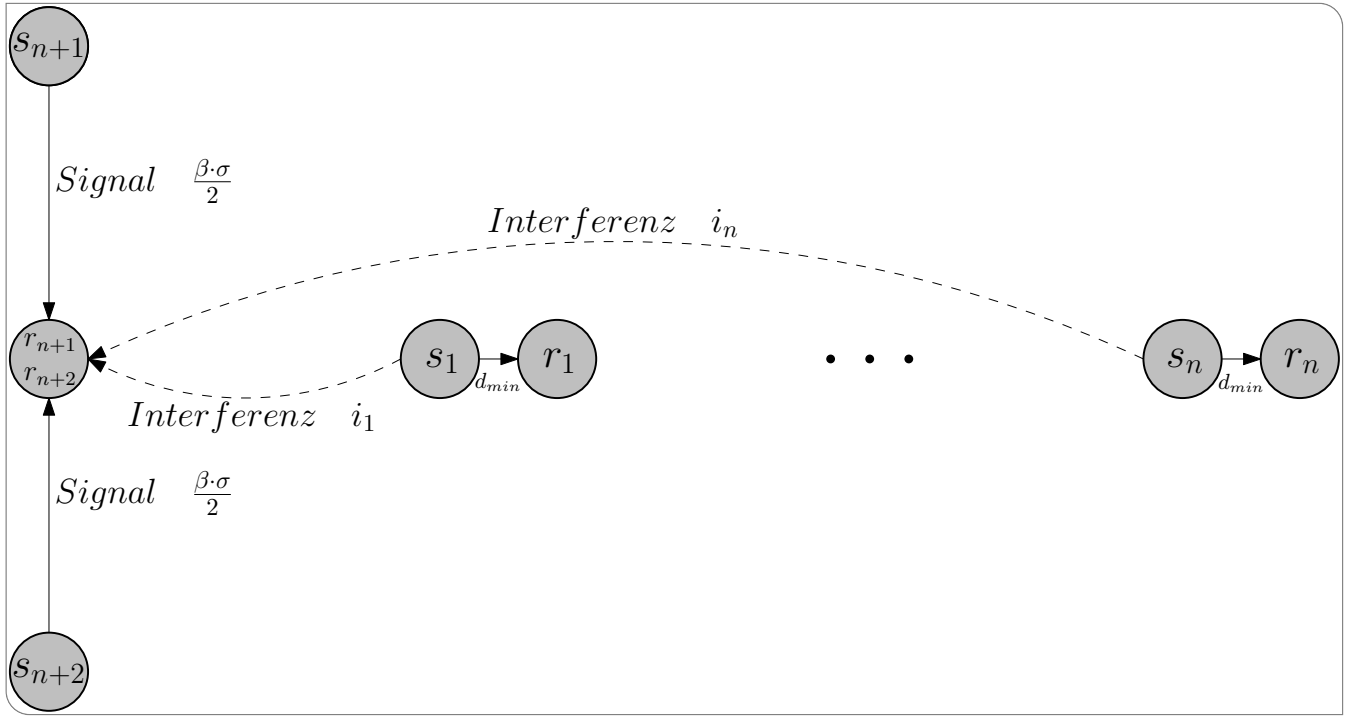
PARTITION ∝ SCHEDULING



Transformation

- $\forall j \in \{1, \ldots, n\} : s_j = ((\frac{P}{i_j})^{\frac{1}{\alpha}}, 0), \quad r_j = s_j + (d_{min}, 0)$

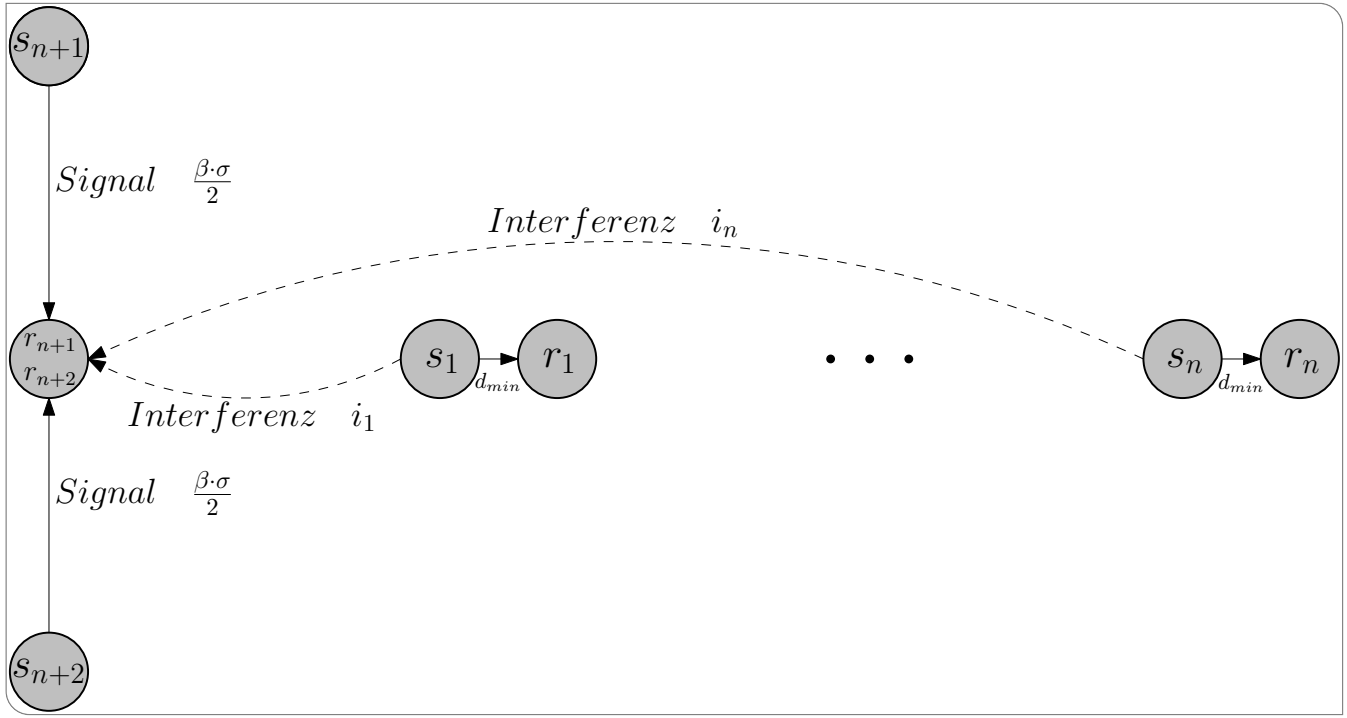
PARTITION ∝ SCHEDULING



Transformation

- $r_{n+1} = (0,0), \quad s_{n+1} = (0, (\frac{2P}{\beta\sigma})^{\frac{1}{\alpha}})$

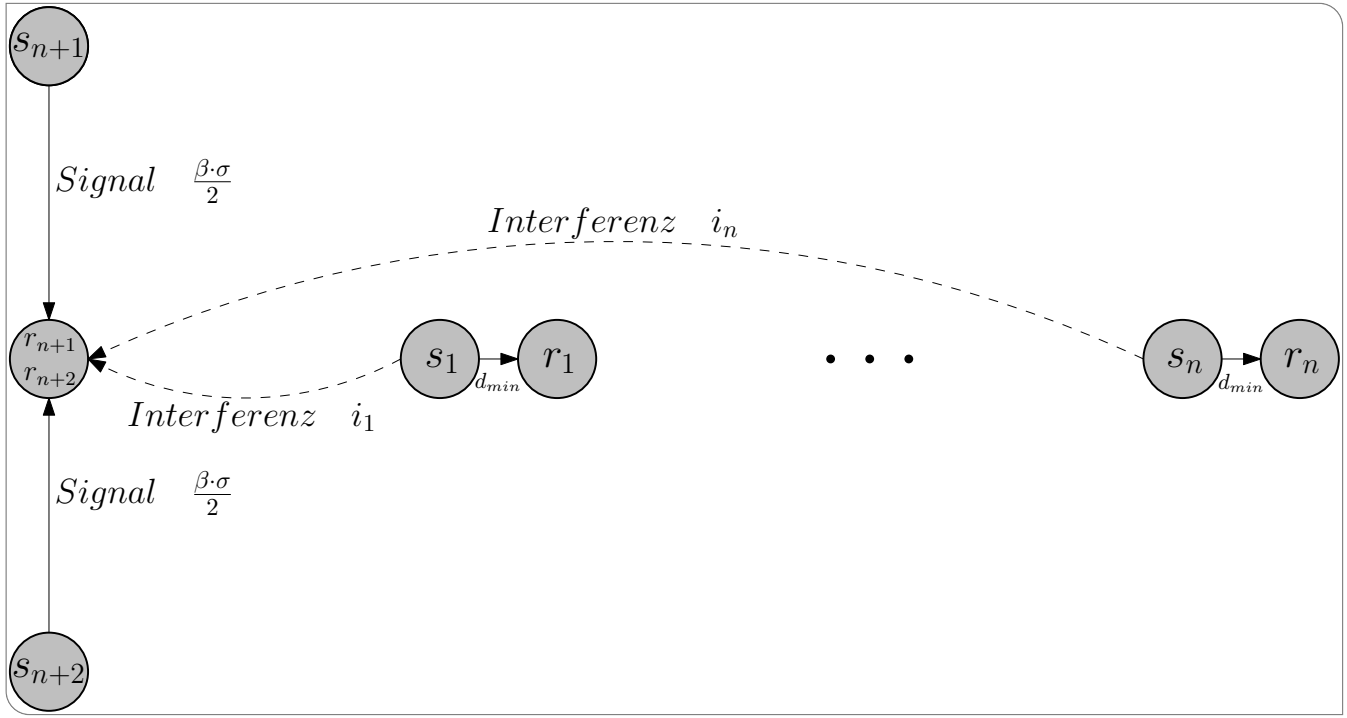
PARTITION ∝ SCHEDULING



Transformation

- $r_{n+2} = (0, 0), \quad s_{n+2} = (0, -(\frac{2P}{\beta\sigma})^{\frac{1}{\alpha}})$

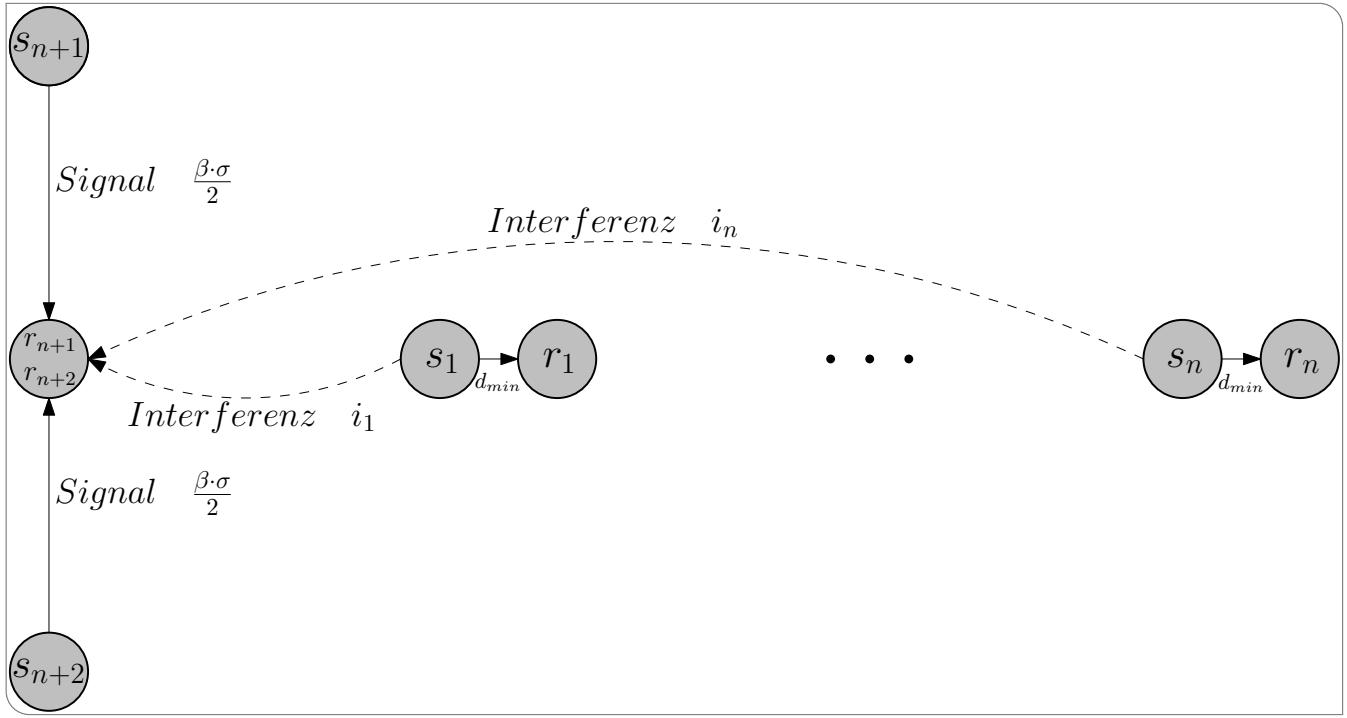
PARTITION ∝ SCHEDULING



Beweis

- $\exists\, \mathcal{I}_1, \mathcal{I}_2$ wie gefordert $\Leftrightarrow \exists$ Schedule mit genau zwei Slots

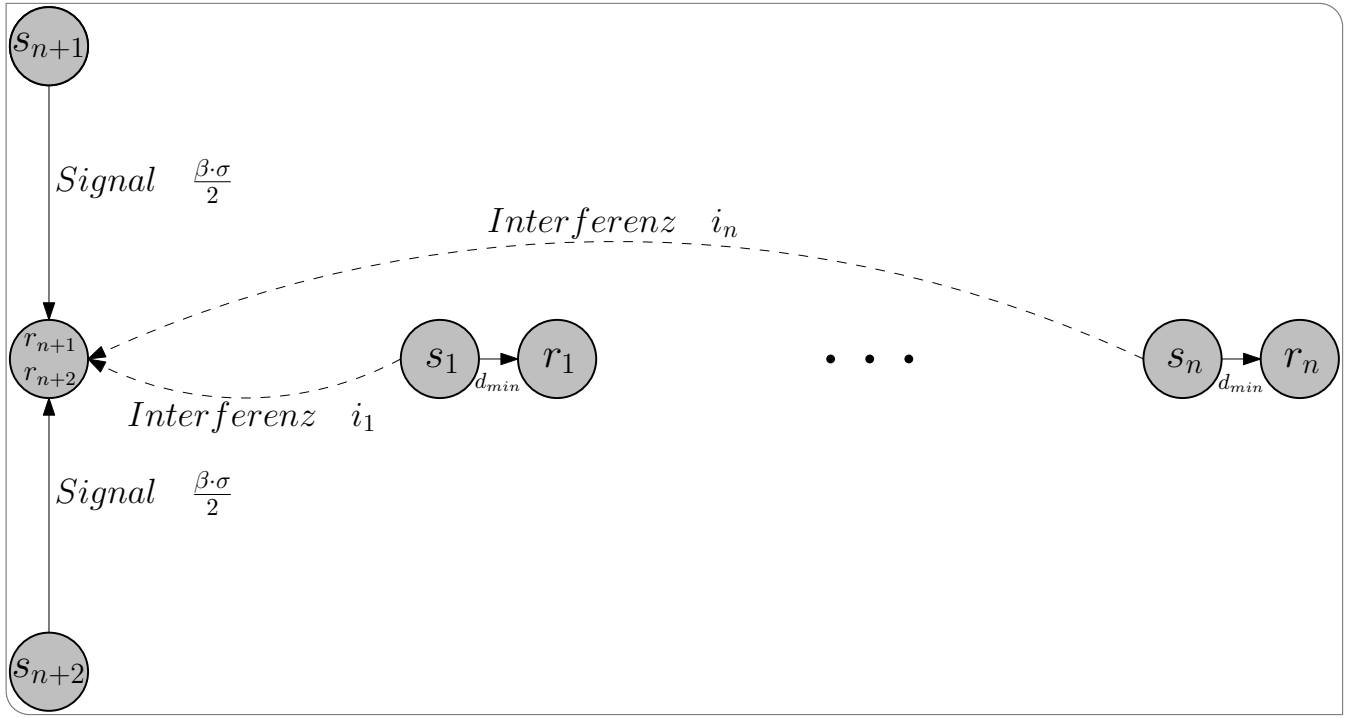
PARTITION ∝ SCHEDULING



Beweis

- l_{n+1} und l_{n+2} können nicht gleichzeitig realisiert werden

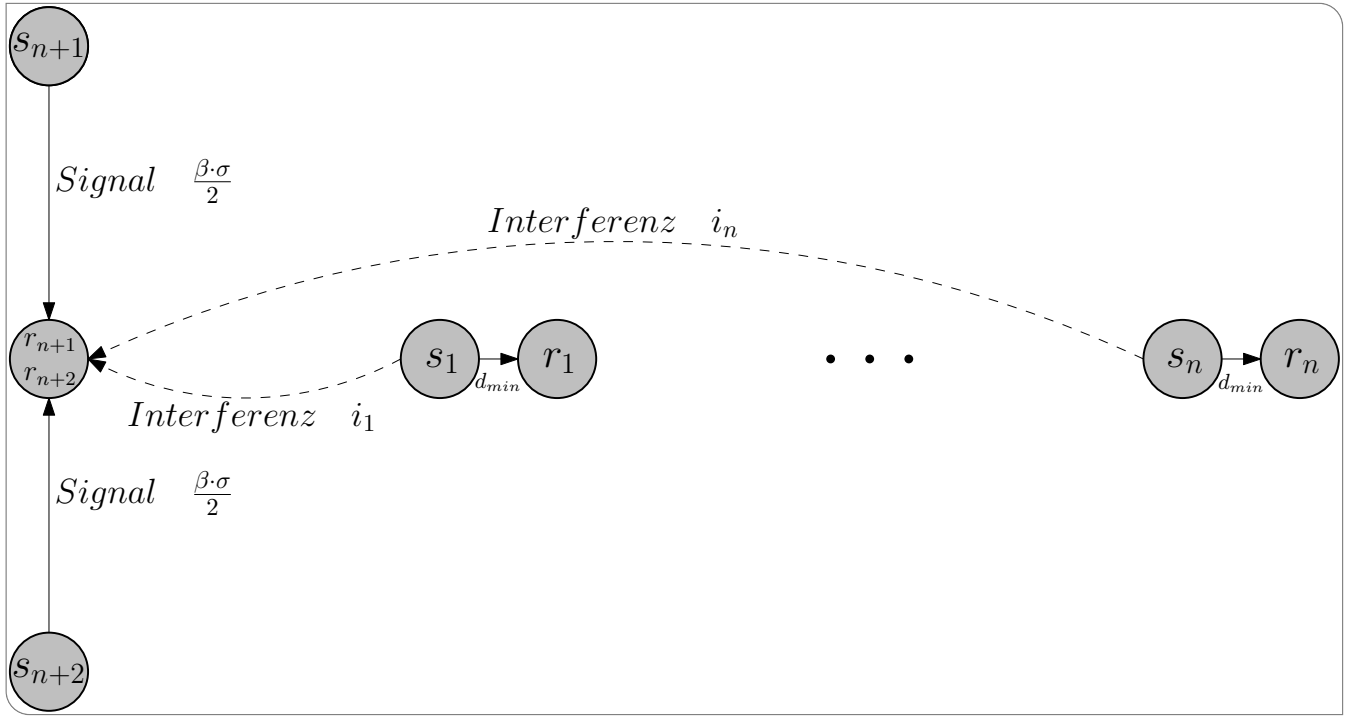
PARTITION ∝ SCHEDULING



Beweis

- l_i wird durch die Interferenz von $\{l_1, \ldots, l_{n+1}\} \setminus \{l_i\}$ nicht behindert

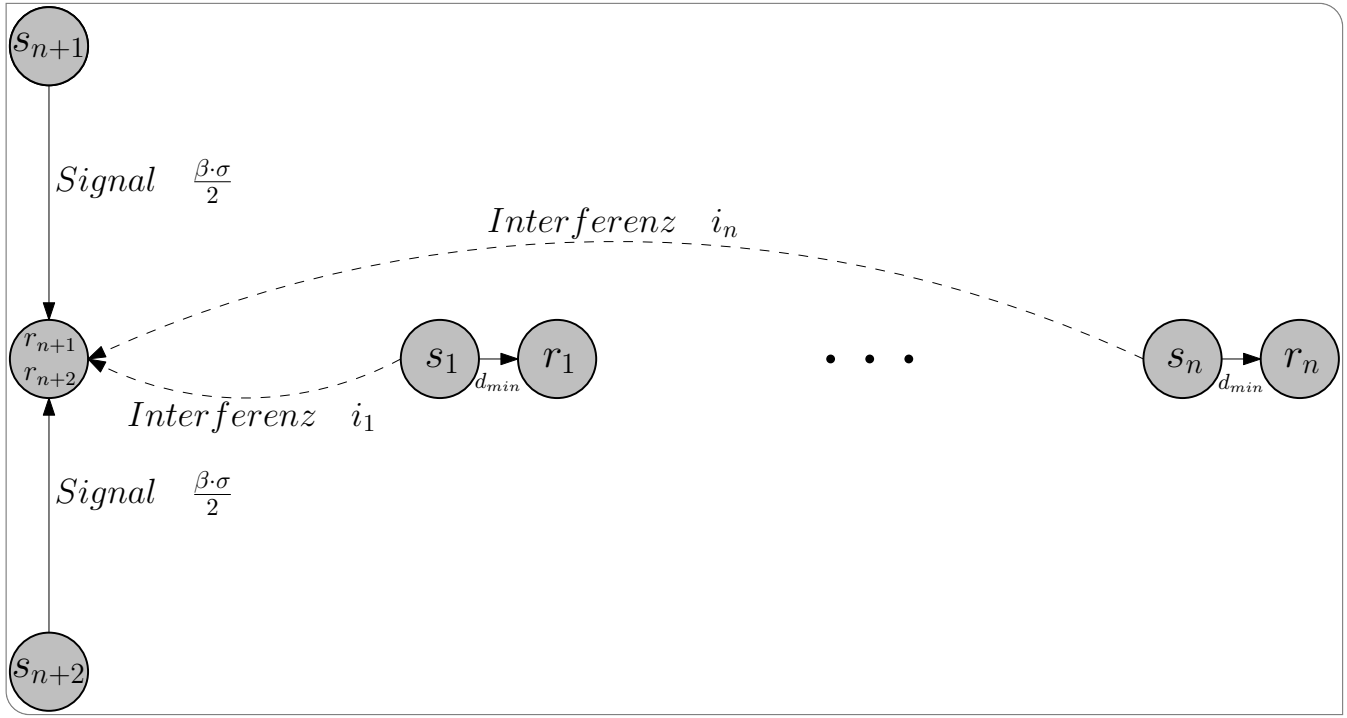
Partition $\propto$ Scheduling



Beweis

- $\exists\, \mathcal{I}_1, \mathcal{I}_2$ wie gefordert $\Rightarrow \exists$ Gültiger Schedule (S_1, S_2)
  - $S_1 = \{l_{n+1}\} \cup \{l_j : i_j \in \mathcal{I}_1\}$
  - $S_2 = \{l_{n+2}\} \cup \{l_j : i_j \in \mathcal{I}_2\}$

PARTITION ∝ SCHEDULING



Beweis

- $\nexists\, \mathcal{I}_1, \mathcal{I}_2$ wie gefordert $\Rightarrow SINR_G(r_{n+1}) < \beta$ oder $SINR_G(r_{n+2}) < \beta$

Algorithmen

Anwendung

- Zentrale Verfahren
- Ineffizient für “interaktive” Planung
- Schedules können vorberechnet und einprogrammiert werden

Allgemein

- Master-Slave-Approximations-Strategie

Vorbemerkung ONE-SLOT-SCHEDULING

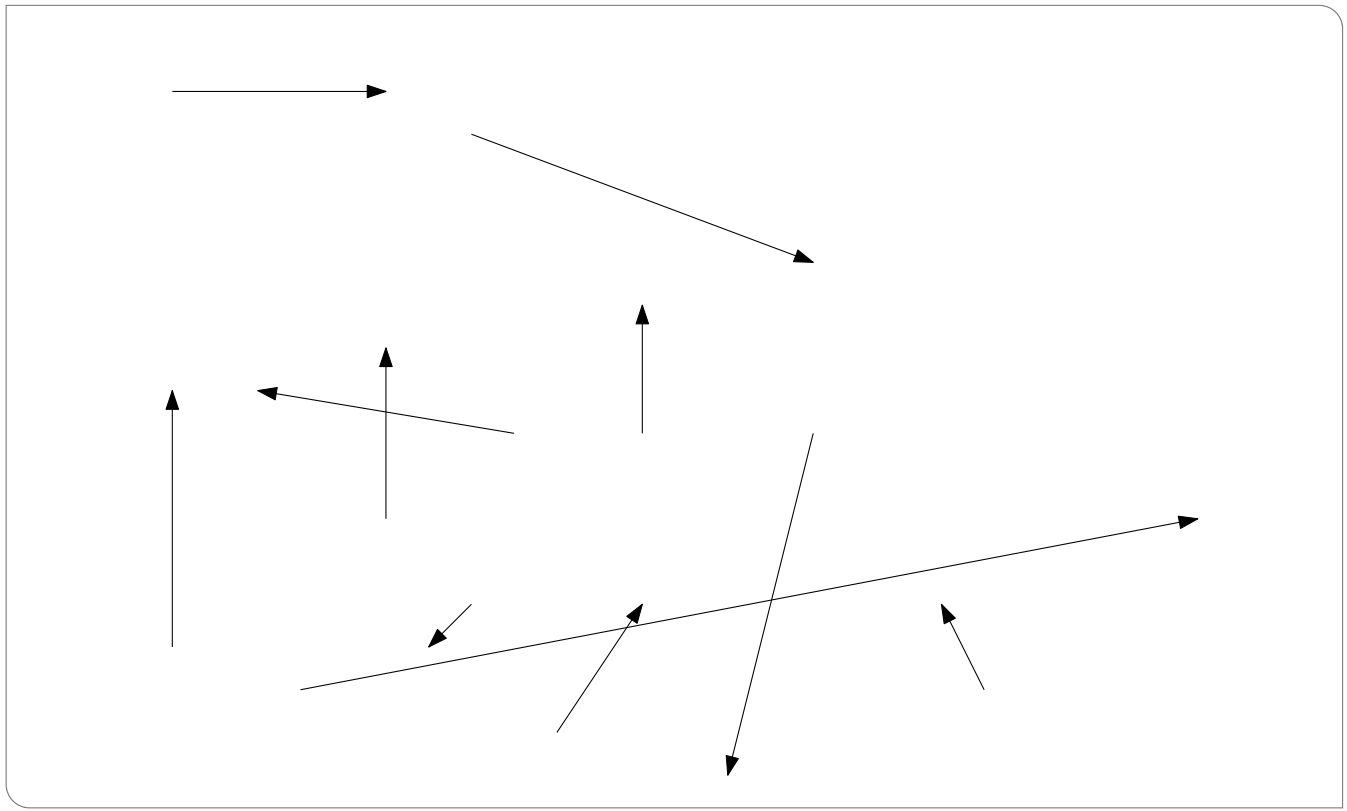
- Affectedness $a_S(l_i) := \frac{1}{SINR_G(r_i)} \cdot \beta$
- $a_S(l_i) \leq 1 \Leftrightarrow SINR_G(r_i) \geq \beta$

ONE-SLOT-SCHEDULING

Algorithmus

1: **procedure** ONE-SLOT SCHEDULING($L = \{l_1, \ldots, l_n\}$)
2: Setze c //Konstante abhängig von α und β
3: **repeat**
4: Kürzeste Verbindung l_v in S einfügen
5: Lösche $l_u \in L : d_{uv} \leq c \cdot d_{vv}$
6: Lösche $l_w \in L : a_S(l_w) \geq \frac{2}{3}$
7: **until** $L = \emptyset$
8: **return** S

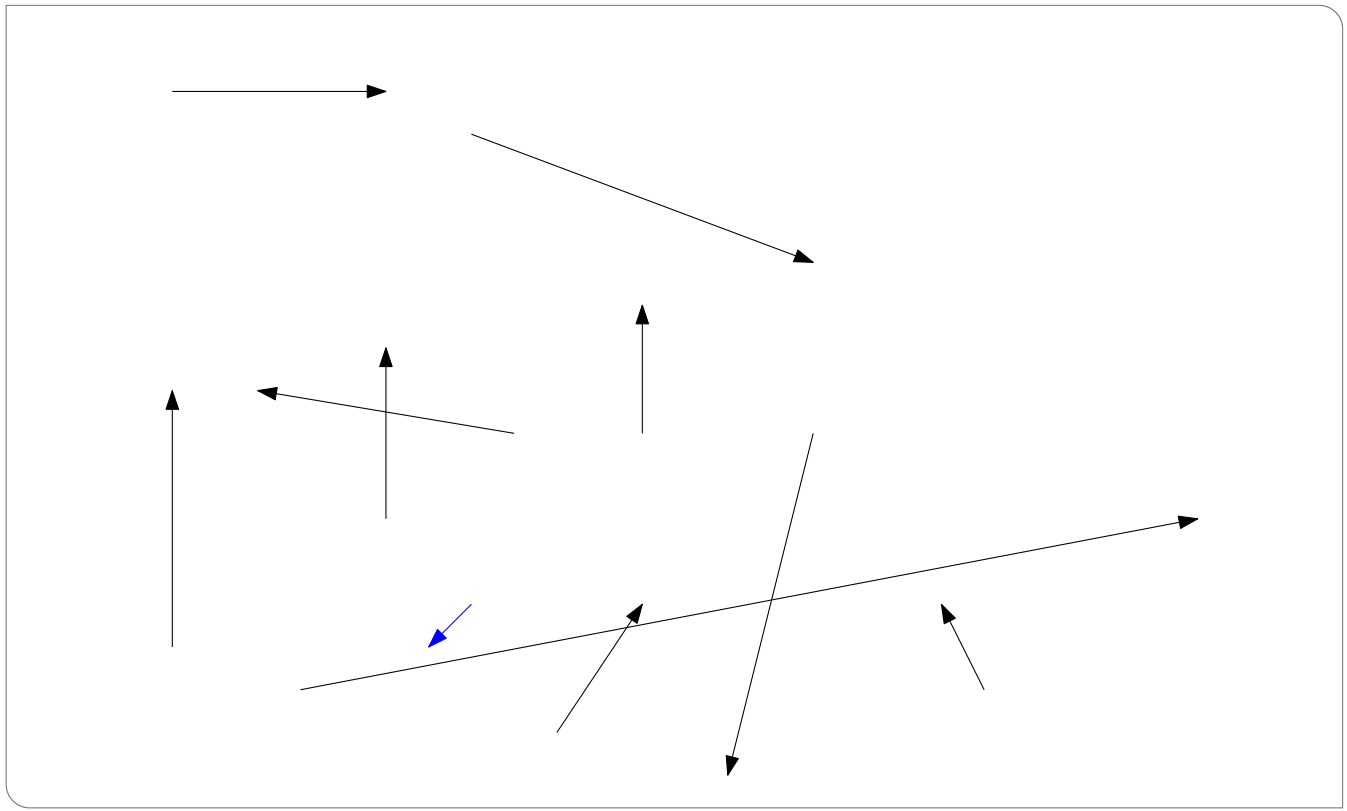
ONE-SLOT-SCHEDULING



Algorithmus

1: Setze c //Konstante abhängig von α und β

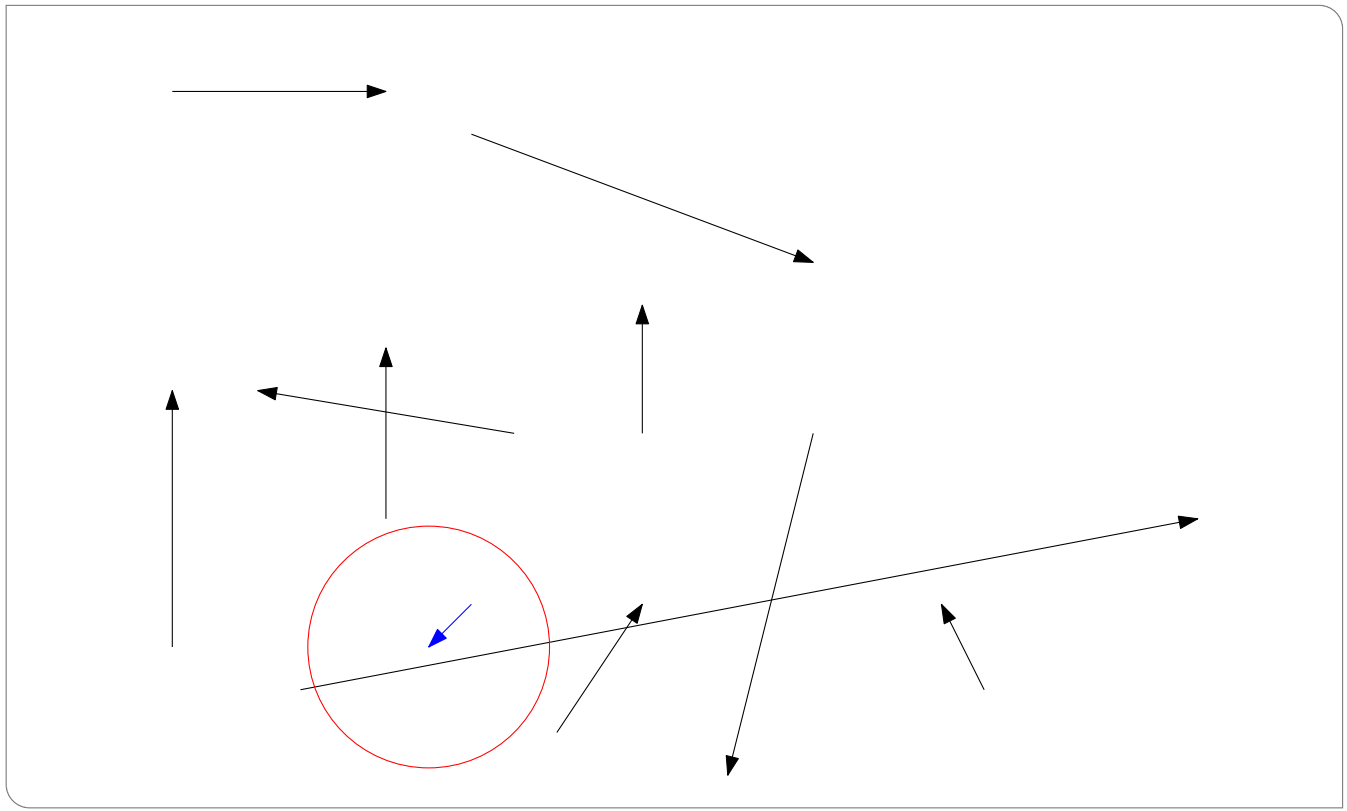
ONE-SLOT-SCHEDULING



Algorithmus

1: Kürzeste Verbindung l_v in S einfügen

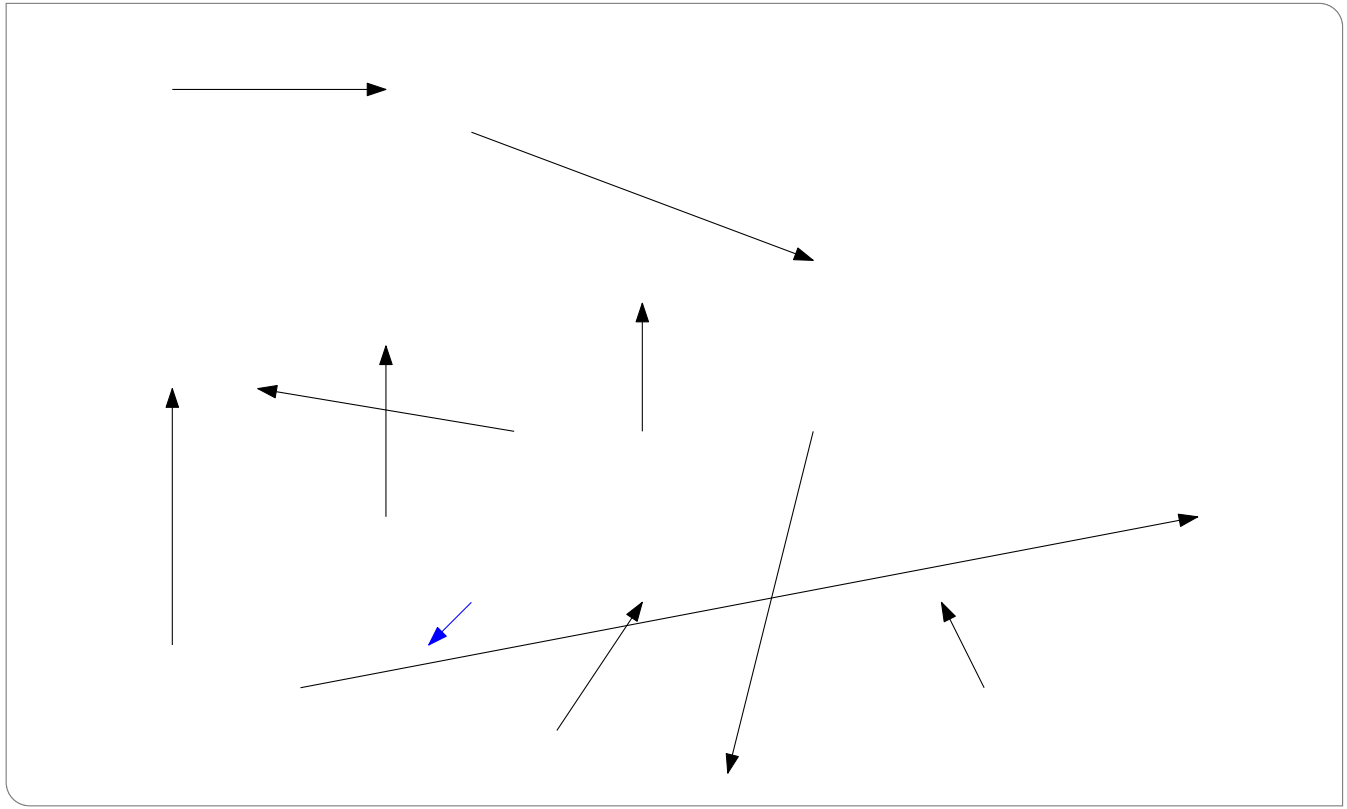
ONE-SLOT-SCHEDULING



Algorithmus

1: Lösche $l_u \in L : d_{uv} \leq c \cdot d_{vv}$

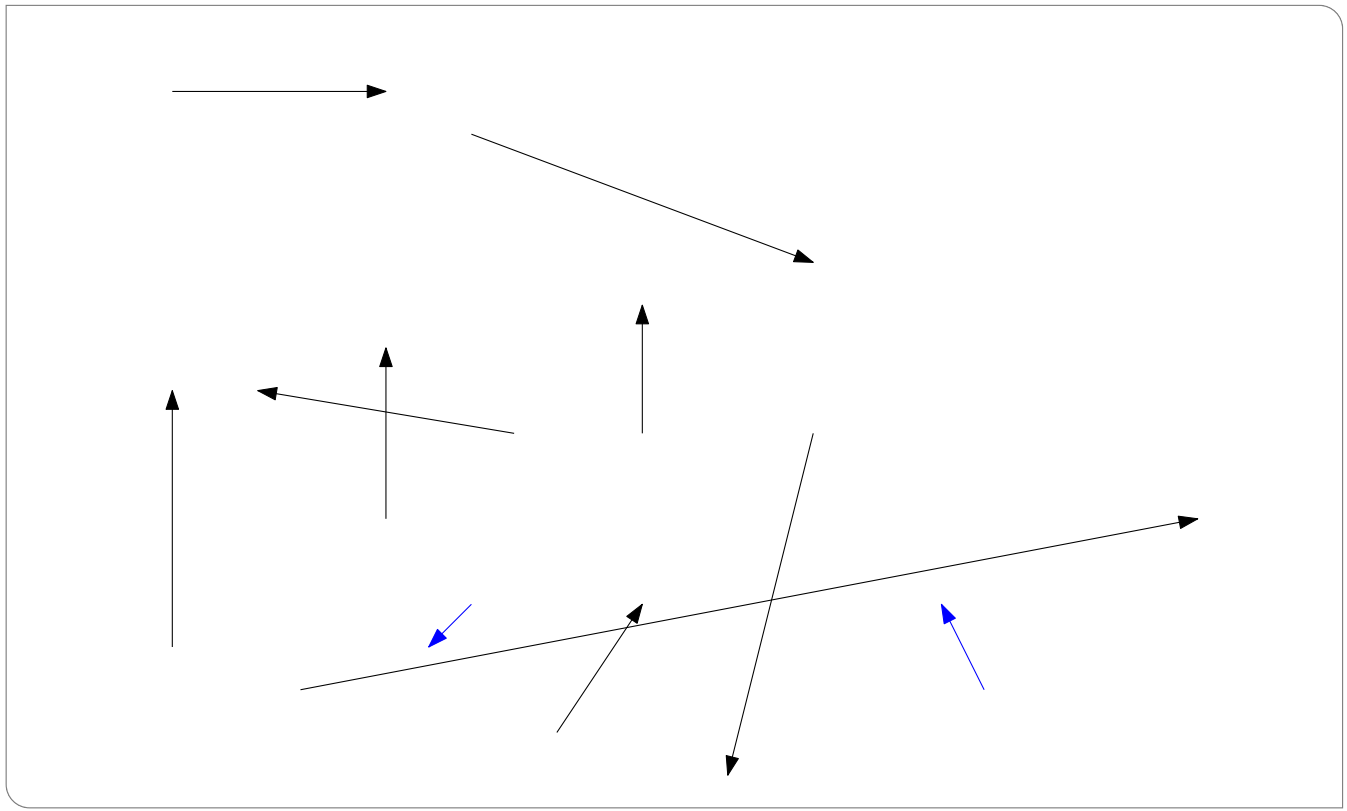
ONE-SLOT-SCHEDULING



Algorithmus

1: Lösche $l_w \in L : a_S(l_w) \geq \frac{2}{3}$

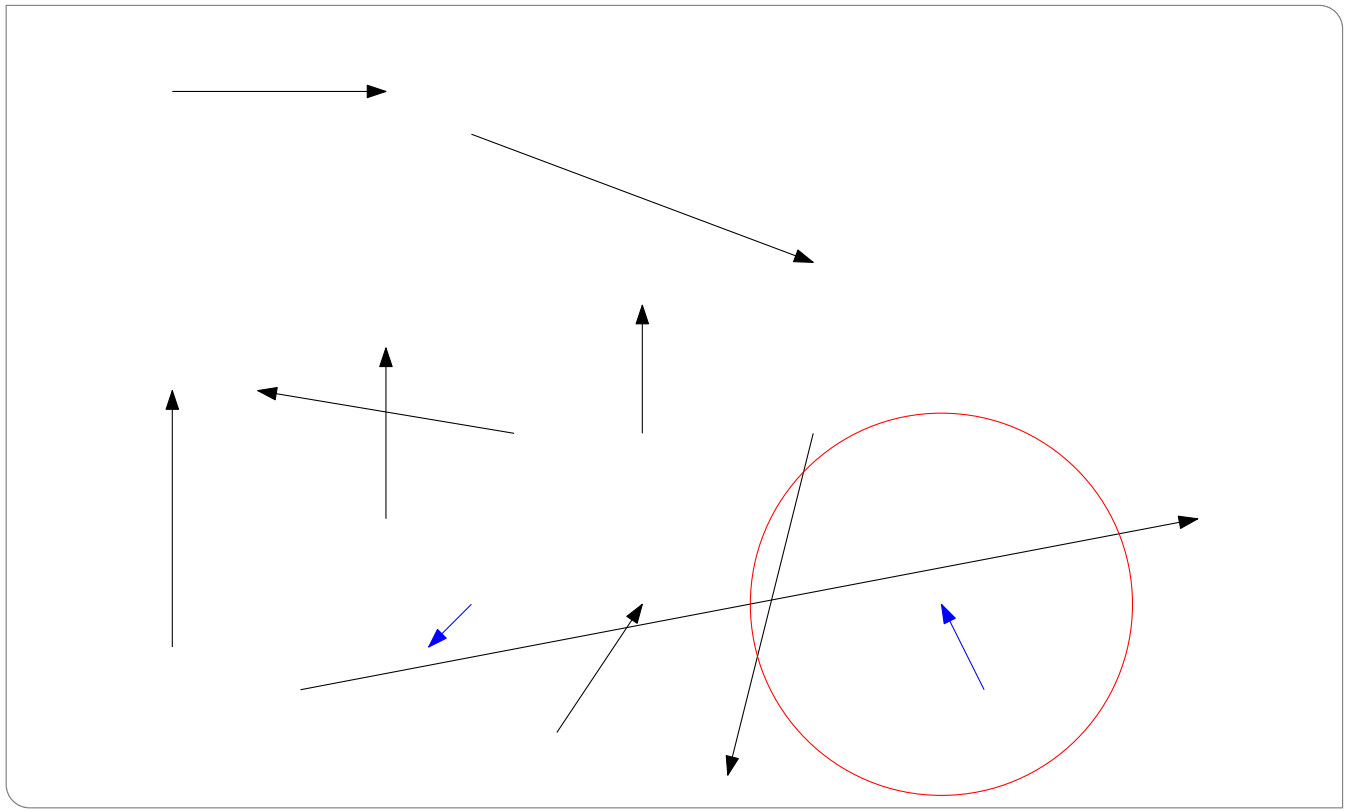
ONE-SLOT-SCHEDULING



Algorithmus

1: Kürzeste Verbindung l_v in S einfügen

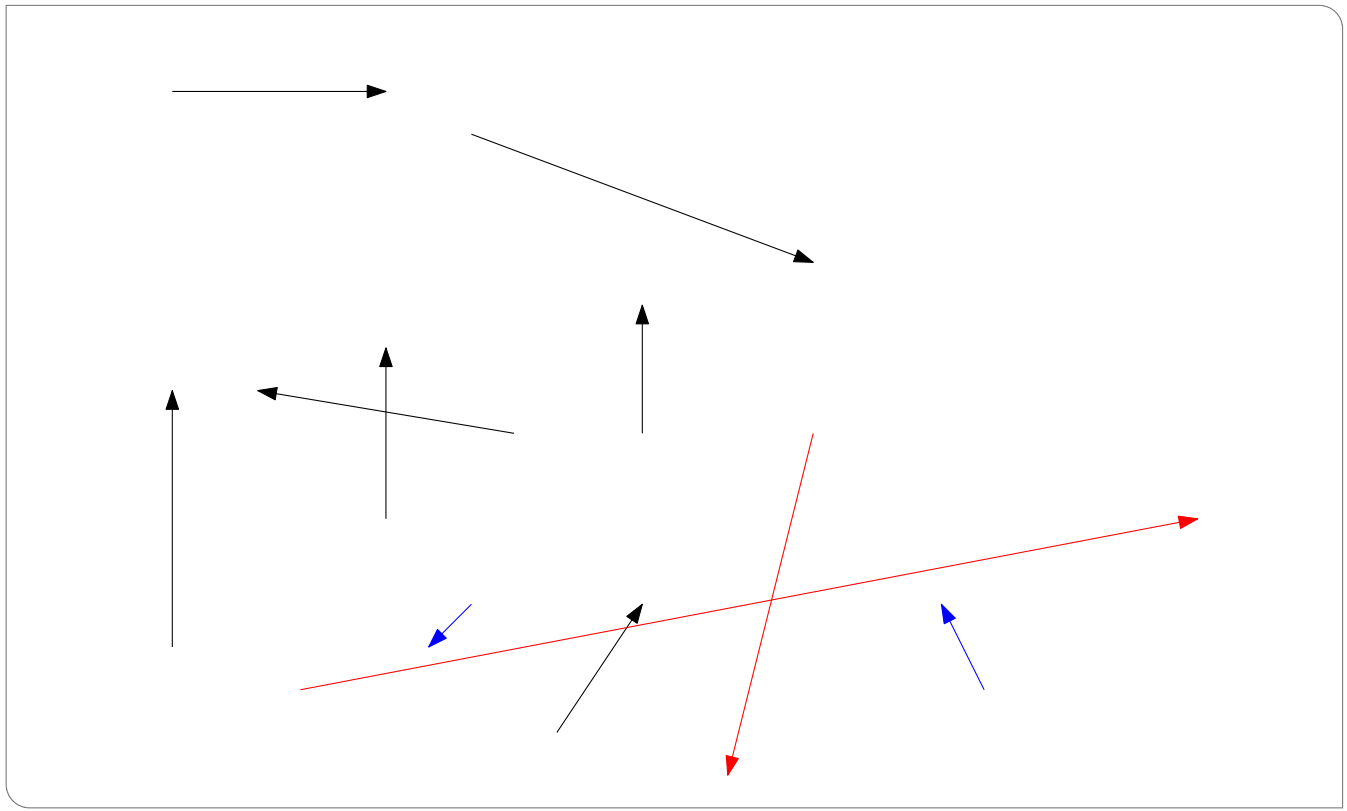
ONE-SLOT-SCHEDULING



Algorithmus

1: Lösche $l_u \in L : d_{uv} \leq c \cdot d_{vv}$

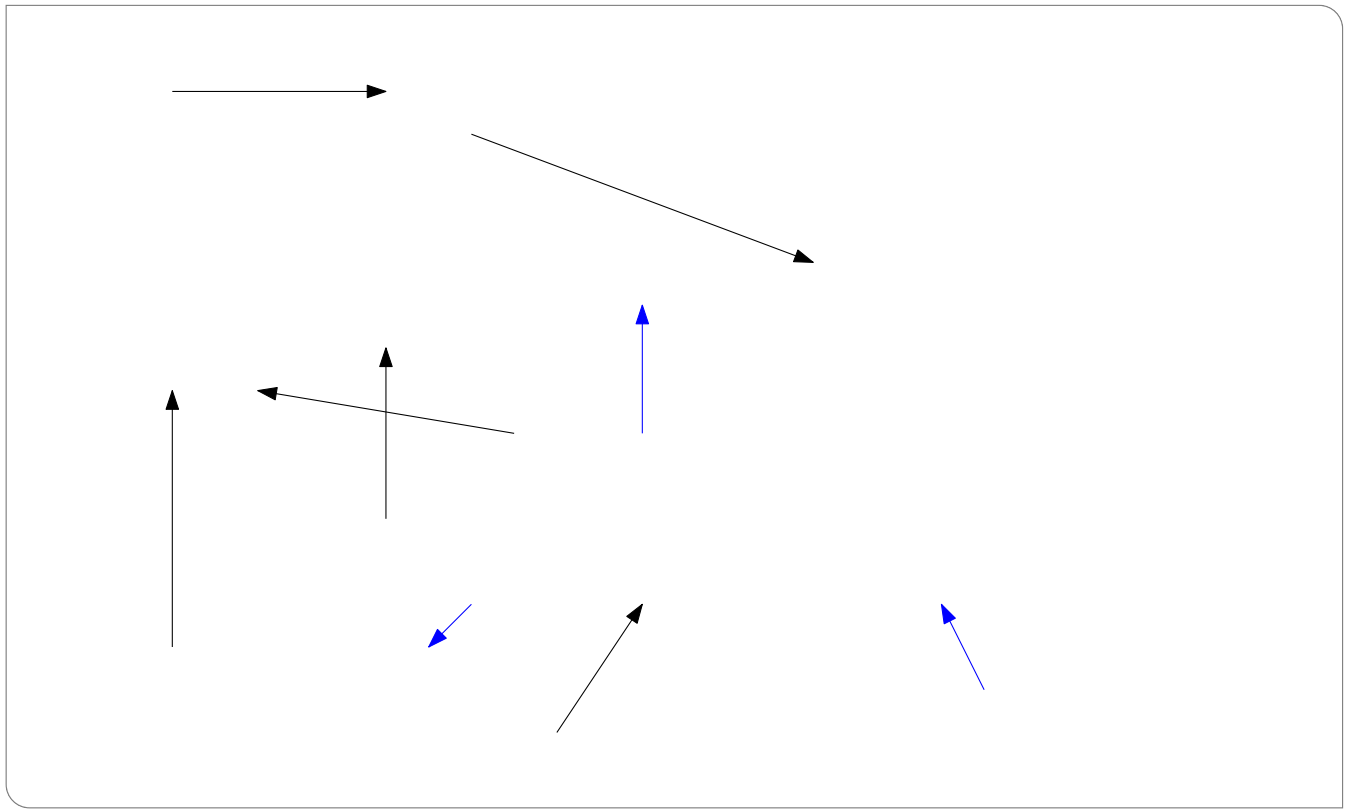
ONE-SLOT-SCHEDULING



Algorithmus

1: Lösche $l_w \in L : a_S(l_w) \geq \frac{2}{3}$

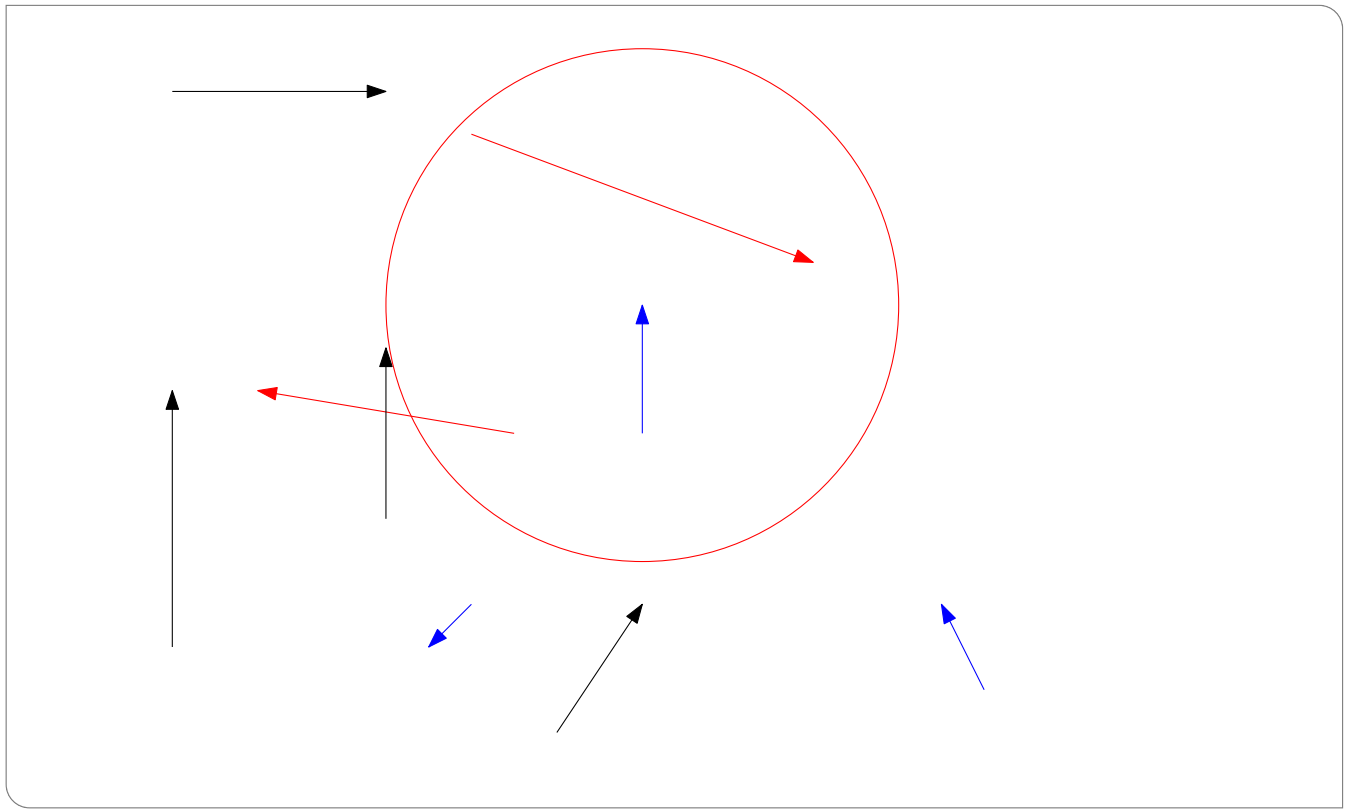
ONE-SLOT-SCHEDULING



Algorithmus

1: Kürzeste Verbindung l_v in S einfügen

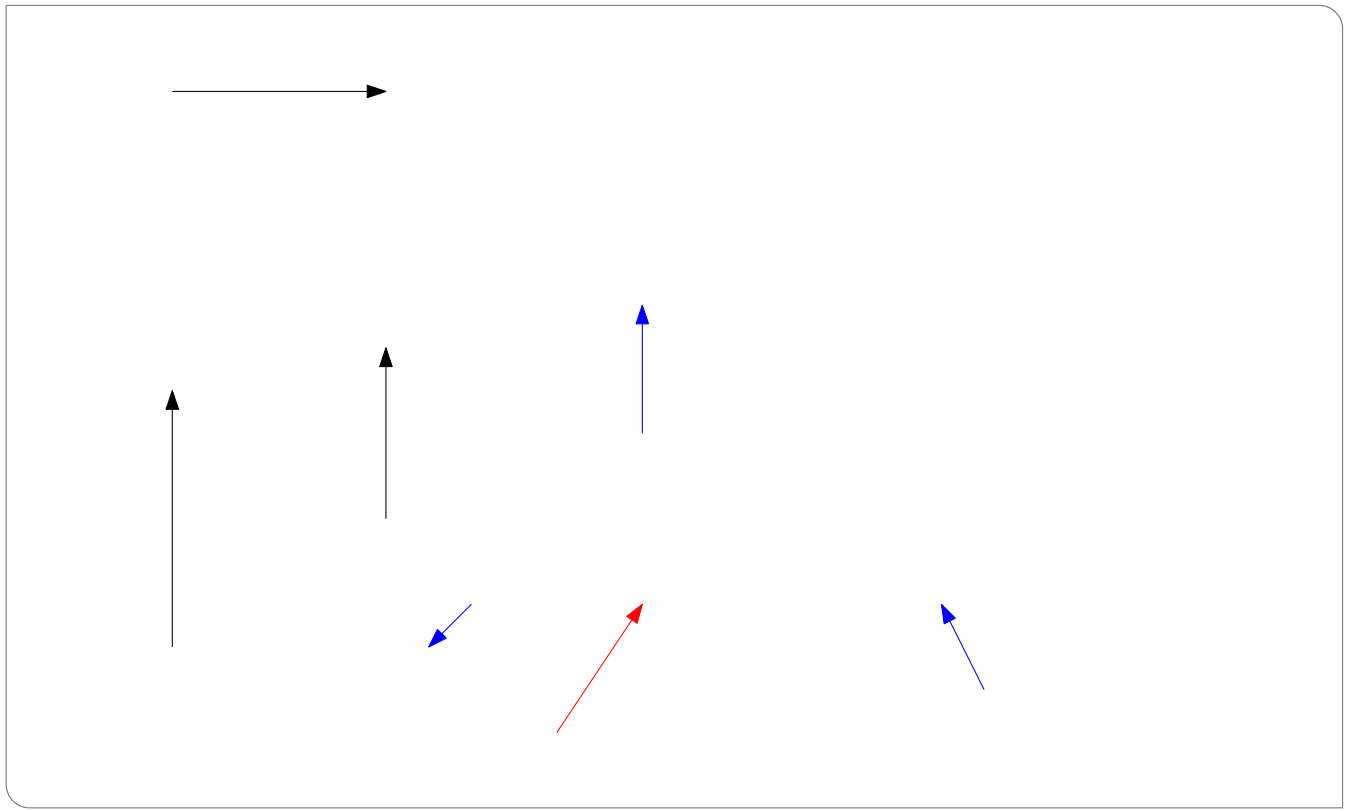
ONE-SLOT-SCHEDULING



Algorithmus

1: Lösche $l_u \in L : d_{uv} \leq c \cdot d_{vv}$

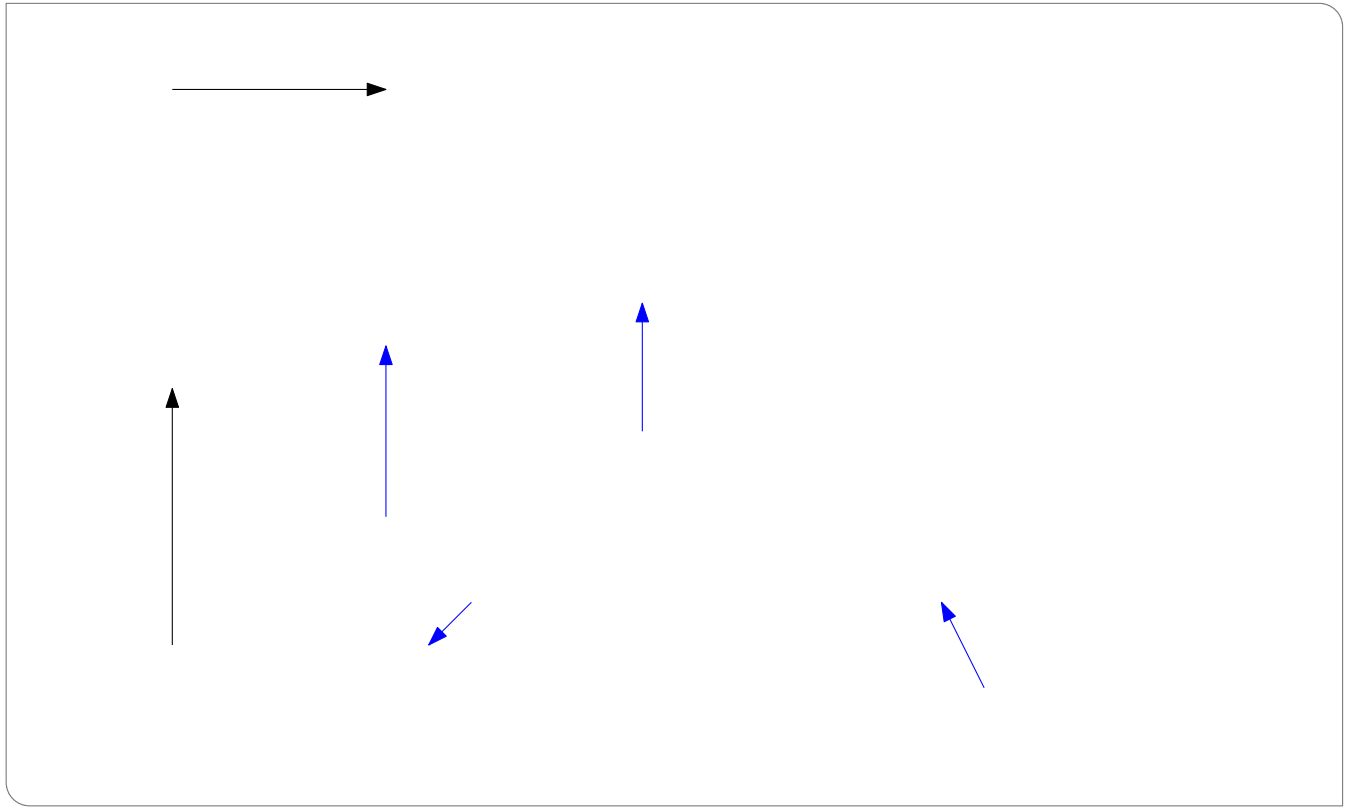
ONE-SLOT-SCHEDULING



Algorithmus

1: Lösche $l_w \in L : a_S(l_w) \geq \frac{2}{3}$

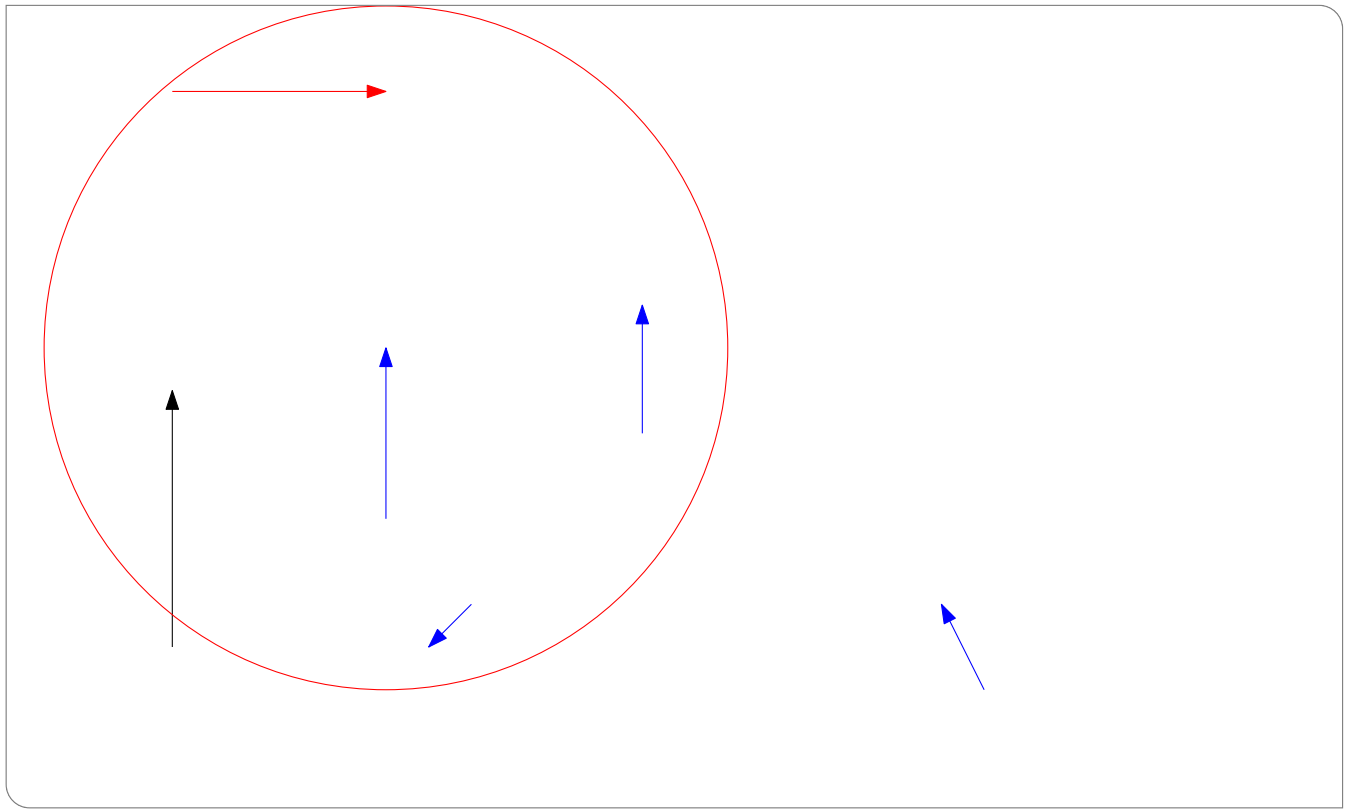
ONE-SLOT-SCHEDULING



Algorithmus

1: Kürzeste Verbindung l_v in S einfügen

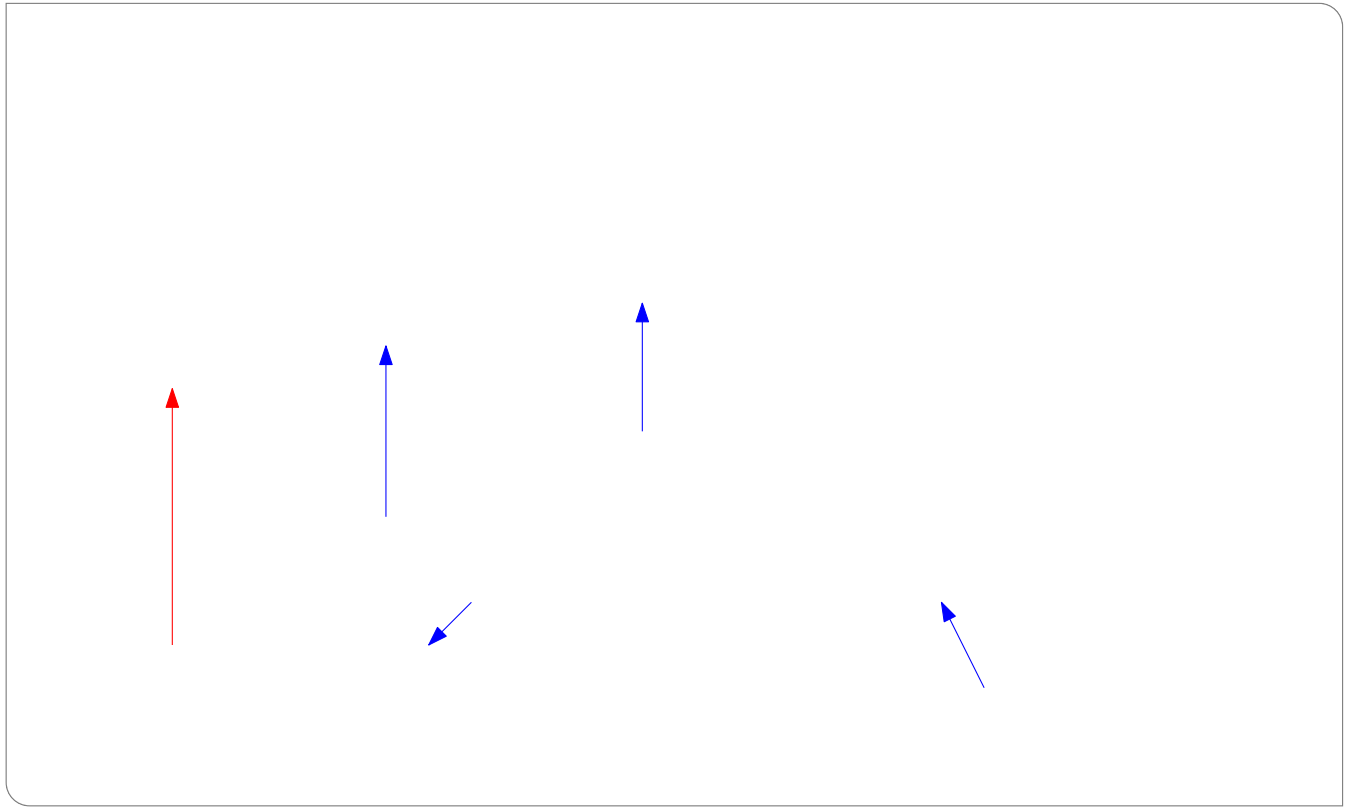
ONE-SLOT-SCHEDULING



Algorithmus

1: Lösche $l_u \in L : d_{uv} \leq c \cdot d_{vv}$

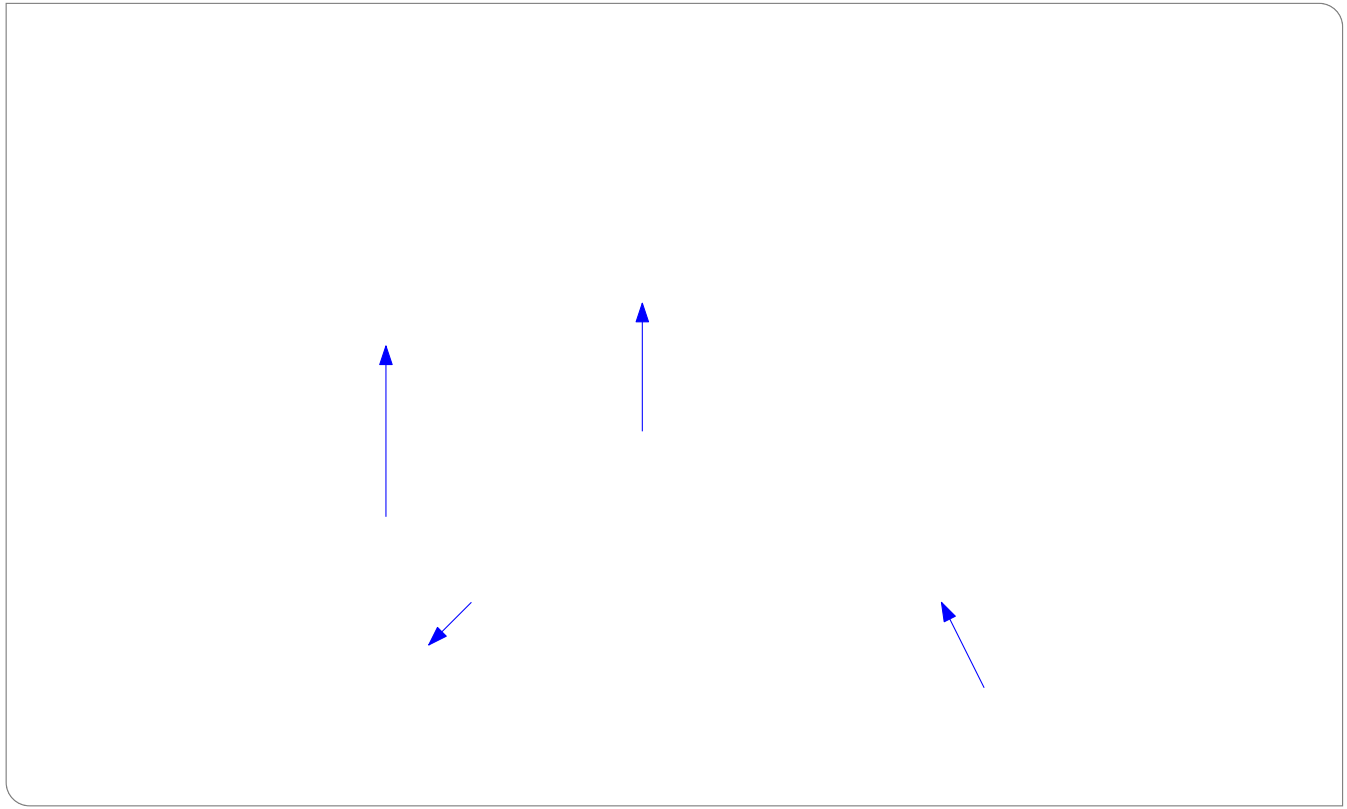
ONE-SLOT-SCHEDULING



Algorithmus

1: Lösche $l_w \in L : a_S(l_w) \geq \frac{2}{3}$

ONE-SLOT-SCHEDULING



Algorithmus

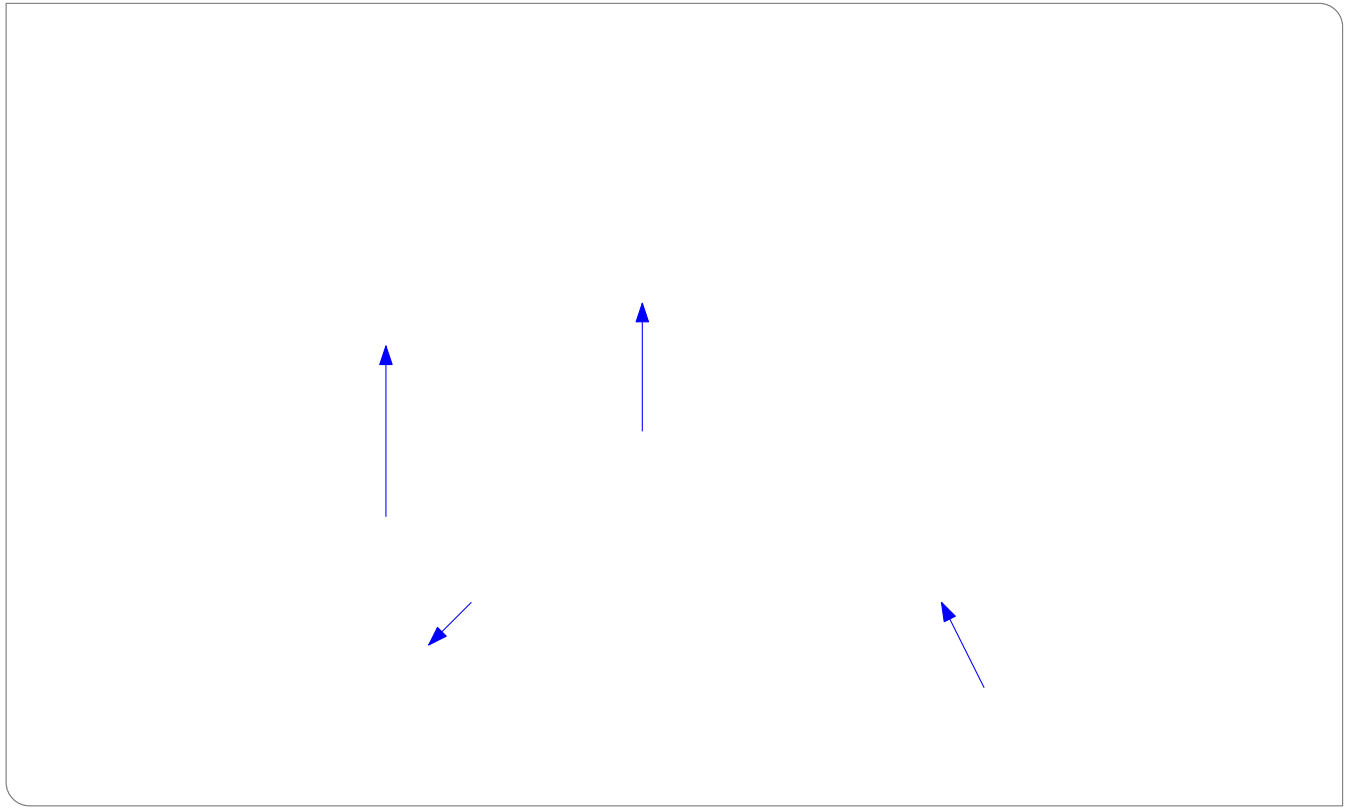
1: **return** S

ONE-SLOT-SCHEDULING

Algorithmus

1: **procedure** ONE-SLOT SCHEDULING($L = \{l_1, \ldots, l_n\}$)
2: Setze c //Konstante abhängig von α und β
3: **repeat**
4: Kürzeste Verbindung l_v in S einfügen
5: Lösche $l_u \in L : d_{uv} \leq c \cdot d_{vv}$
6: Lösche $l_w \in L : a_S(l_w) \geq \frac{2}{3}$
7: **until** $L = \emptyset$
8: **return** S

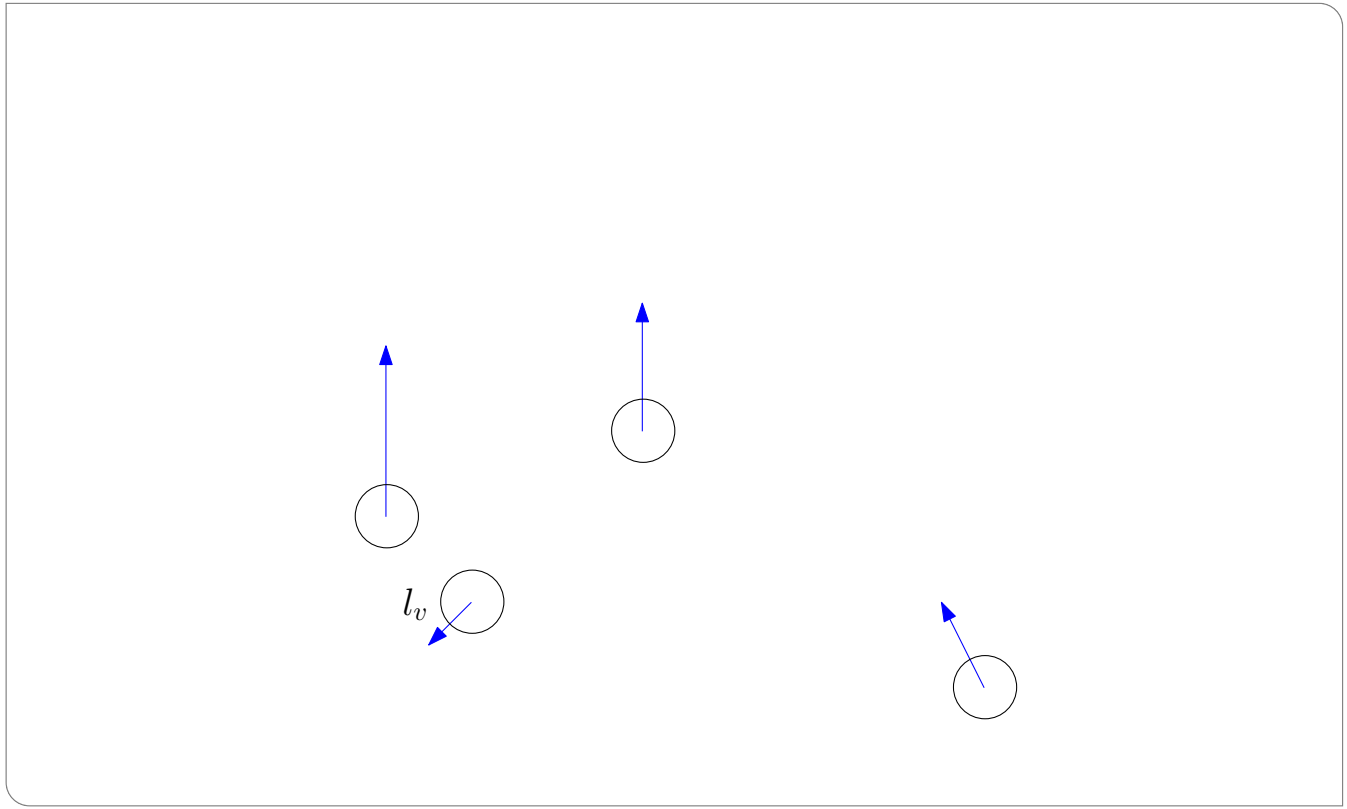
ONE-SLOT-SCHEDULING



Korrektheit

- Nach l_v eingefügte Verbindungen erhöhen $a_S(l_v)$ um maximal $1/3$

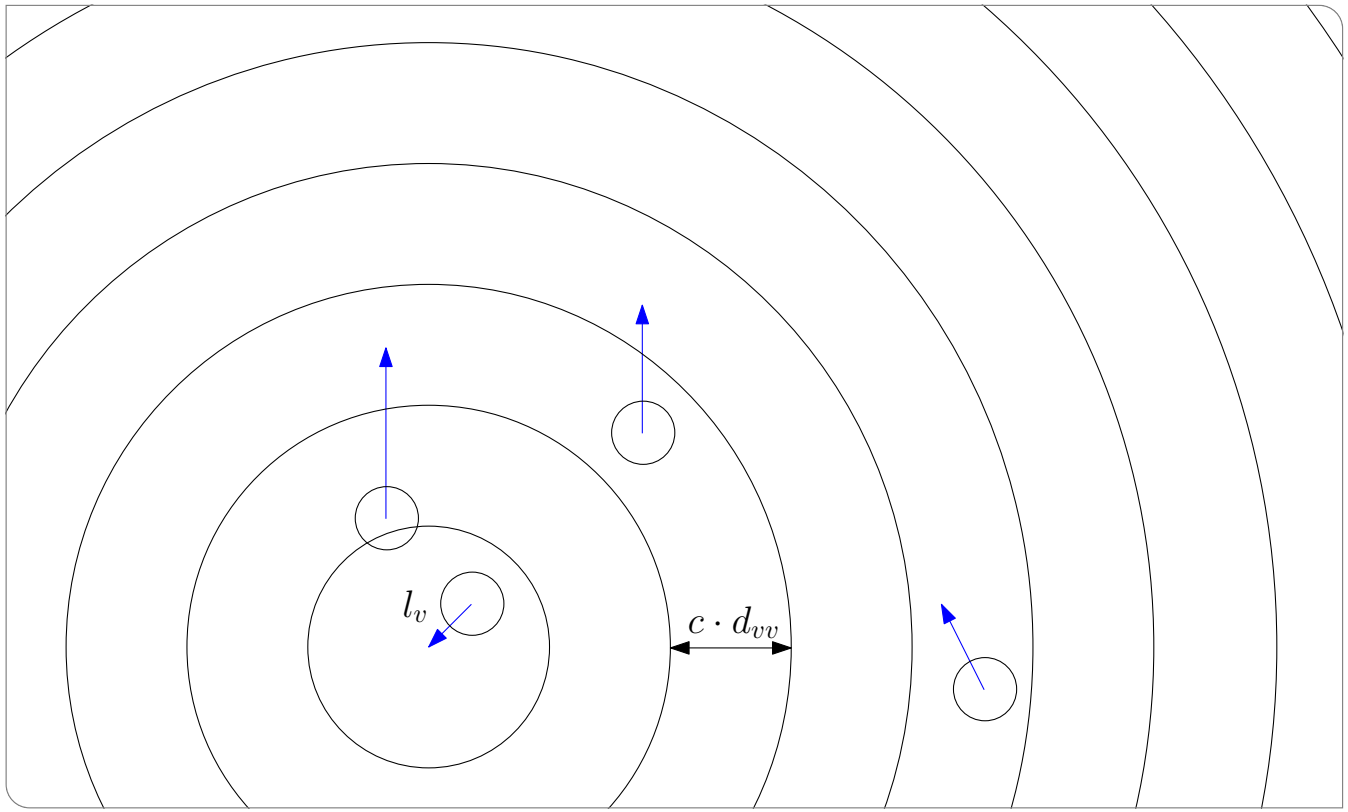
ONE-SLOT-SCHEDULING



Korrektheit

- Disjunkte Scheiben um die Sender mit Radius $d_{vv} \cdot \frac{(c-1)}{2}$

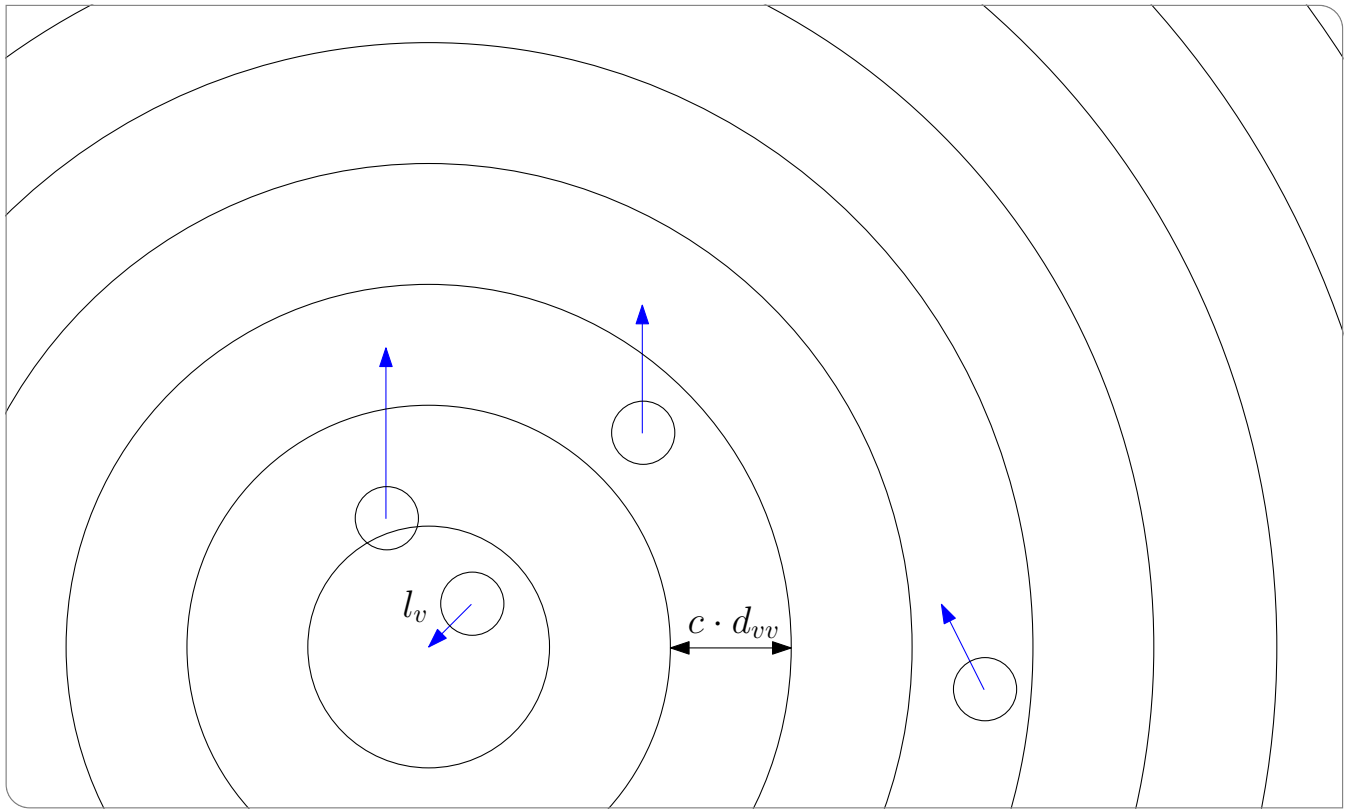
ONE-SLOT-SCHEDULING



Korrektheit

- Teile die Ebene in Ringe R_k um r_v mit Breite $c \cdot d_{vv}$

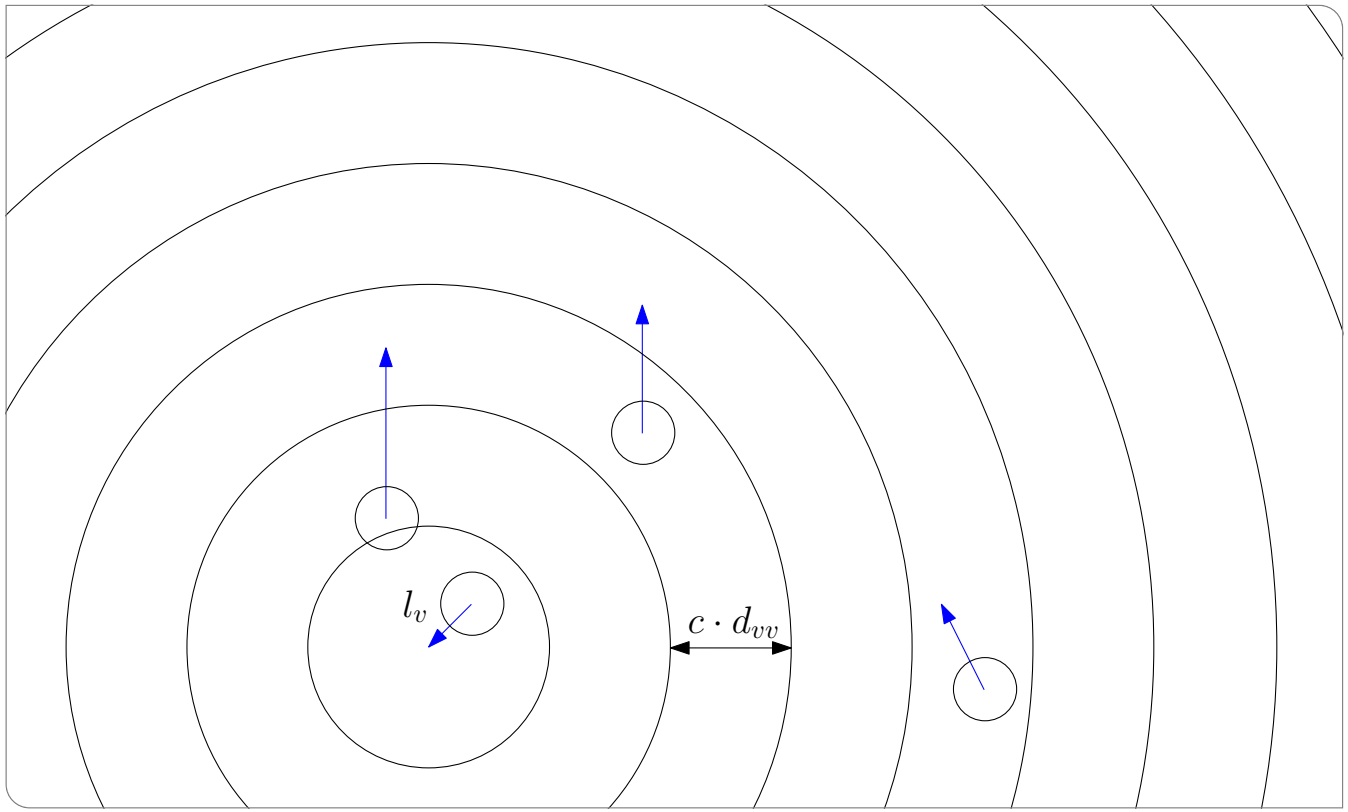
ONE-SLOT-SCHEDULING



Korrektheit

- Anzahl der Sender in Ring $R_k \leq \frac{\text{Fläche } R_k}{\text{Fläche Scheibe}}$

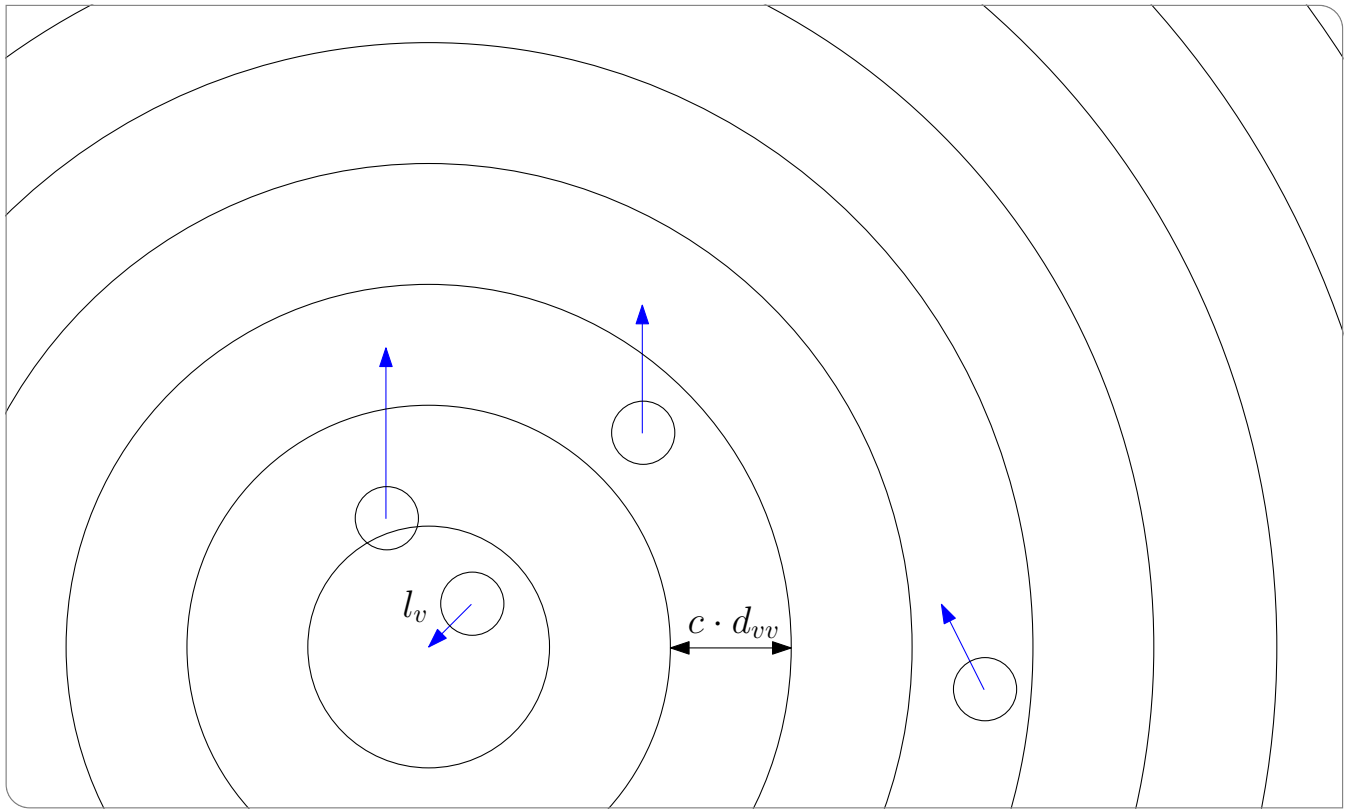
ONE-SLOT-SCHEDULING



Korrektheit

- Abstand der Sender in Ring $R_k \geq k \cdot c \cdot d_{vv}$

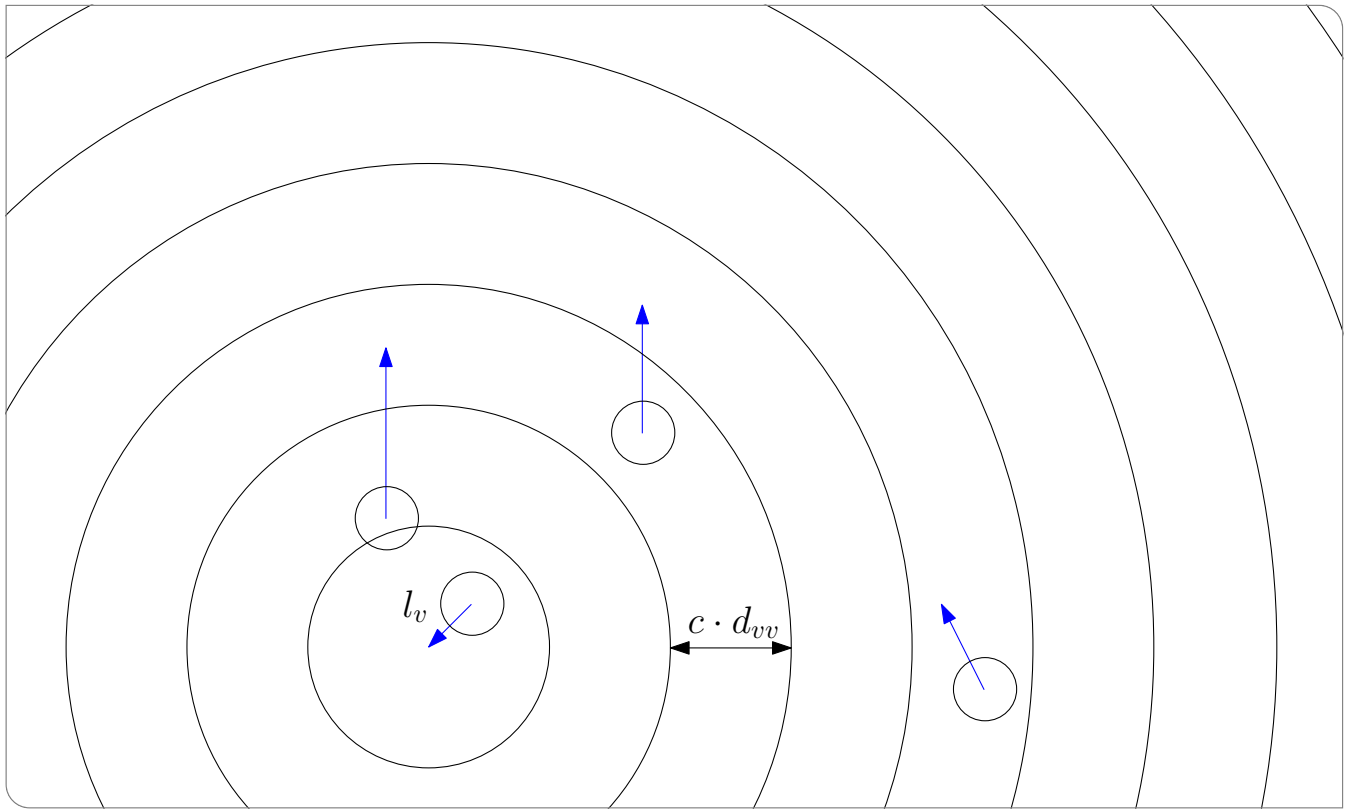
One-Slot-Scheduling



Korrektheit

- Obere Schranke für die Affectedness durch Sender in einem Ring

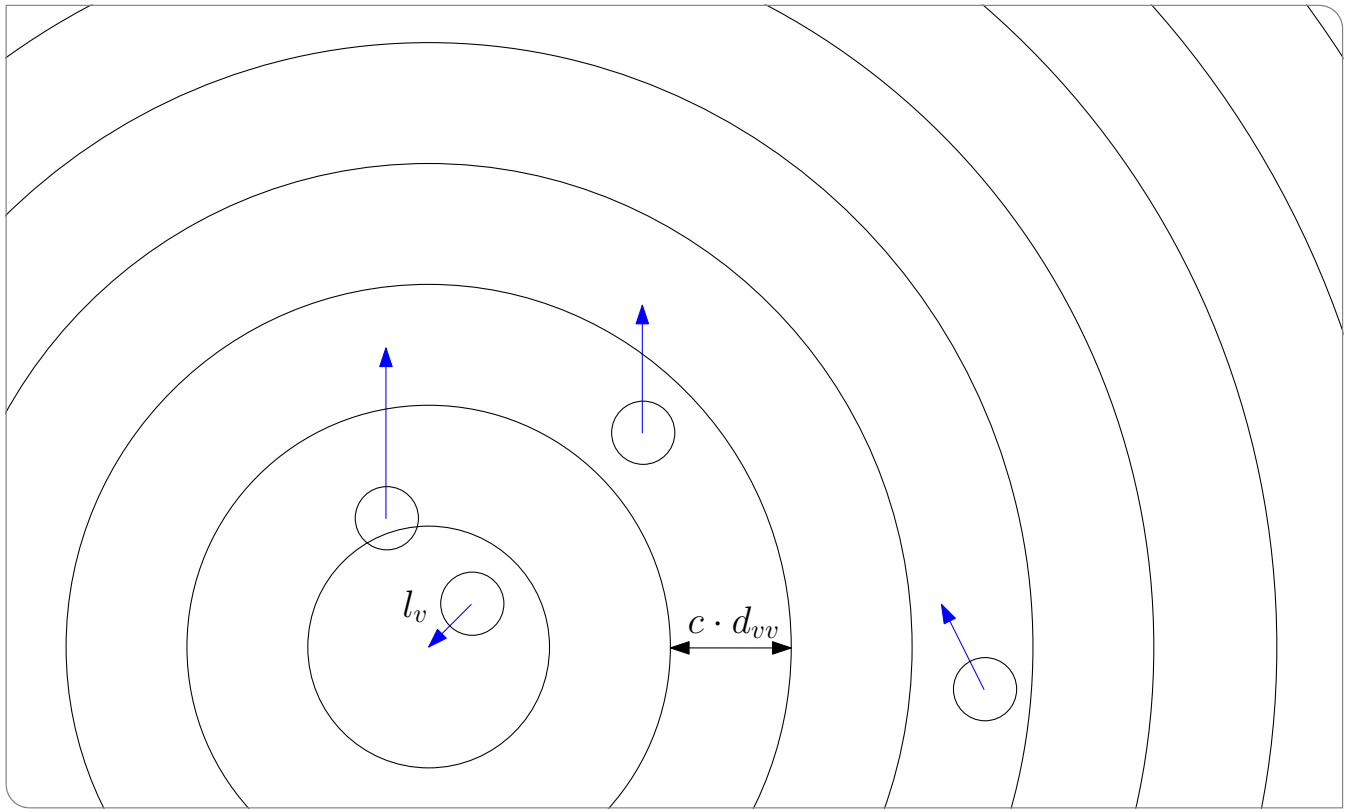
ONE-SLOT-SCHEDULING



Korrektheit

- Summiere über unendlich viele Ringe auf

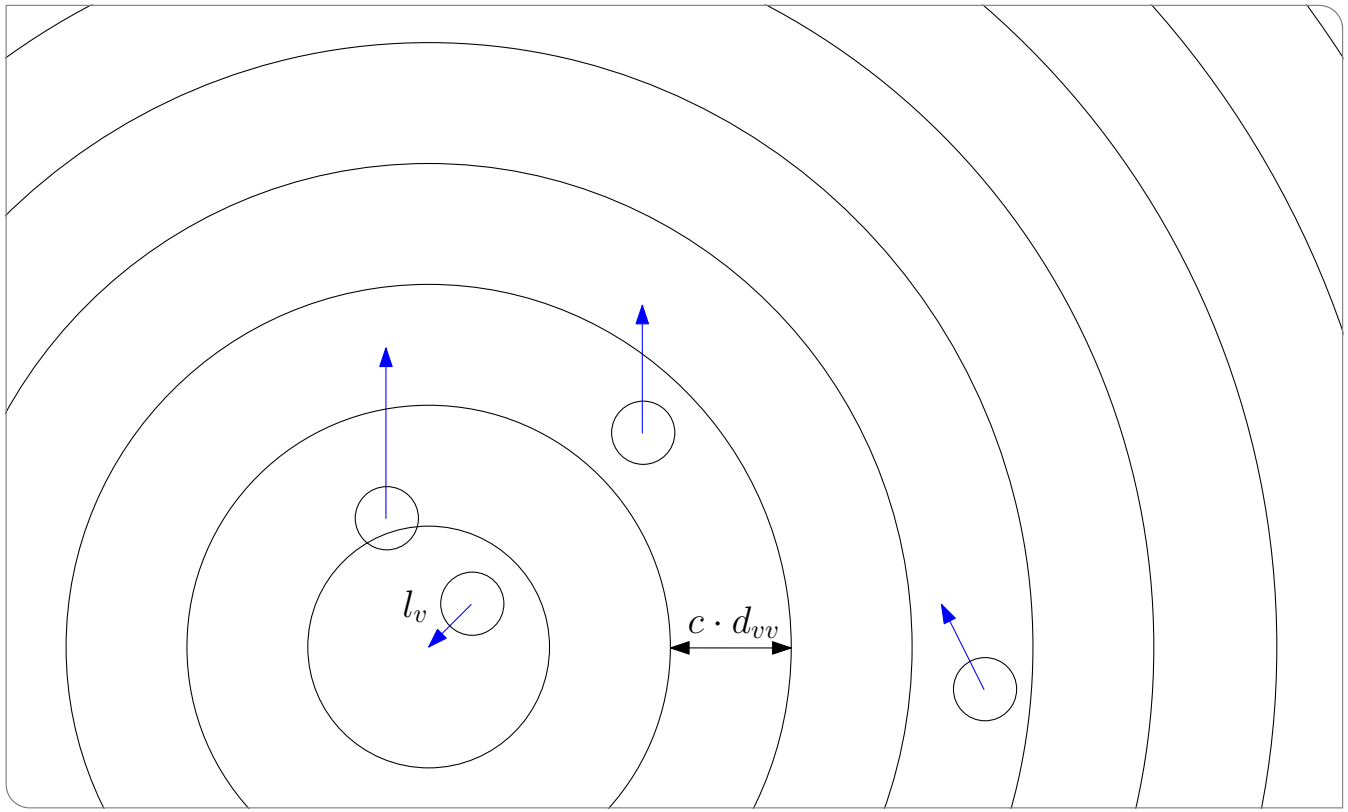
ONE-SLOT-SCHEDULING



Korrektheit

- Affectedness durch später eingefügte Verbindungen $\leq \frac{1}{3}$ bei entsprechender Wahl von c

ONE-SLOT-SCHEDULING



Korrektheit

- $a_S(l_v) \leq 1$

ONE-SLOT-SCHEDULING

Approximations-Rate

- ONE-SLOT-SCHEDULING hat konstante Approximations-Rate ρ
- Für realistische Parameter α und β gilt $\rho > 10000$
- Simplere Algorithmen sind in der Praxis erfolgreicher

SCHEDULING

Algorithmus

```
1: procedure SCHEDULING(L = {l_1, ..., l_n})
2:     t := 0
3:     repeat
4:         t := t + 1
5:         S_t := ONE-SLOT-SCHEDULING(L)
6:         L := L \ S_t
7:     until L = ∅
8:     return (S_1, ..., S_t)
```

SCHEDULING

Korrektheit

- Folgt aus der Korrektheit von ONE-SLOT-SCHEDULING

Approximationsrate von SCHEDULING

Schritt 1

- Argumentation über die Kosteneffektivität
- $c(S) = 1$ für alle S
- Die Kosteneffektivität ist $e(S) = \frac{c(S)}{|S|_{\text{erstmalig überdeckt}}} = p(l)$ für $l \in S$
- $e(S)$ steigt monoton im Laufe des Algorithmus

Schritt 2

- $l_1, \ldots, l_n$ sei die Einfügereihenfolge der Verbindungen
- Es kann ein Schedule mit Kosten *OPT* erstellt werden
- Die verbleibenden l_i können stets mit Kosten *OPT* geplant werden

Approximationsrate von SCHEDULING

Schritt 3

- Zu jedem Zeitpunkt i existiert ein S mit $e(S) = \frac{1}{|S|} \leq \frac{OPT}{n-i+1} \Leftrightarrow |S| \geq \frac{n-i+1}{OPT}$
- Annahme: S existiert nicht
- $\forall S : |S| < \frac{n-i+1}{OPT} \Rightarrow OPT \cdot |S| < n - i + 1$
- Verbleibende Verbindungen sind nicht mit Kosten OPT planbar ↯

Schritt 4

- $p(l_i) \leq \rho \cdot \frac{OPT}{n-i+1}$
- $\sum_{i=1}^{n} p(l_i) \leq (1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}) \cdot \rho \cdot OPT = \mathcal{O}(log(n)) \cdot OPT$

Scheduling im $SINR_G$-Modell

Zusammenfassung

- $SINR_G$, SCHEDULING und ONE-SLOT-SCHEDULING
- SCHEDULING ist $\mathcal{NP}$-schwer
- Algorithmus für ONE-SLOT-SCHEDULING hat eine konstante Approximationsrate
- Algorithmus für SCHEDULING hat eine logarithmische Approximationsrate

Scheduling im $SINR_G$-Modell

Vielen Dank für die Aufmerksamkeit