DIPLOMARBEIT

Schneller Algorithmus für kürzeste Wege in irregulären Gittergraphen

Angefertigt am
Forschungsinstitut für Diskrete Mathematik

Vorgelegt der
Mathematisch-Naturwissenschaftlichen Fakultät der
Rheinischen Friedrich-Wilhelms-Universität Bonn

März 2009

von

Jesco Humpola

aus Melle

Inhaltsverzeichnis

Einleitung

Eine elektronische Schaltung bestehend aus Kondensatoren, Transistoren, Widerständen und anderen Bauteilen, die auf kleinstem Raum vollständig auf einem einzigen Halbleitersubstrat untergebracht sind, wird als integrierter Schaltkreis (*Chip*) bezeichnet. Wenn er aus Millionen von Bauelementen besteht, geschehen der Entwurf und die Realisierung im VLSI[1]-Design-Prozess. Für anwendungsspezifische Chips, wie sie zum Beispiel in Mobiltelefonen benötigt werden, existiert als Teilgebiet das ASIC[2]-Design.

Der Designprozess lässt sich in drei Phasen einteilen. Auf die *funktionale Spezifikation* folgt das *Logical Design*. Dabei werden logische Bauteile (*Circuits*) logisch verknüpft. Das Resultat ist die *Netzliste*. Im *Physical Design* wird diese Netzliste realisiert. Wenn eine disjunkte Anordnung (*Placement*) der Circuits auf der Chipfläche gefunden ist, wird das zeitliche Verhalten verbessert (*Timing-Optimierung, Clocktree-Synthese*). Schließlich ist eine Verdrahtung (*Routing*) gesucht, die exakt den logischen Verknüpfungen entspricht.

Das Forschungsinstitut für Diskrete Mathematik der Universität Bonn entwickelt in Kooperation mit IBM seit mehr als 20 Jahren die BonnTools (Korte et al. [2007]), ein Programmpaket für das *Physical Design*. Mit Hilfe dieses Pakets werden weltweit selbst besonders komplexe Chips erfolgreich entworfen. Die BonnTools enthalten das Programm BonnRoute®, um ein Routing für einen Chip zu bestimmen. Mit diesem Programm werden unter anderem Millionen von kürzesten Wegen in einem Graphen mit Milliarden von Knoten berechnet. Der dafür verwendete Algorithmus wird in Hetzel [1998] beschrieben. Dieser bestimmt mit Hilfe einer Modifikation des Algorithmus von Dijkstra [1959] zielgerichtet kürzeste Pfade, indem Intervalle anstatt einzelner Knoten gelabelt werden.

Im Rahmen dieser Arbeit wird der genannte Algorithmus vorgestellt und verallgemeinert. Diese Verallgemeinerung wird als Prozedur formuliert und mit `Suche_Pfad` bezeichnet. Mit dieser Prozedur können kürzeste Wege in einem verallgemeinerten Subgraphen eines dreidimensionalen Gittergraphen bestimmt werden, der als „Track-Graph“ bezeichnet wird. Außerdem können verallgemeinerte Kantengewichte verwendet werden. Damit ist der Algorithmus auch für zukünftige Technologien, 45 nm und darüber hinaus, effizient einsetzbar.

Zur Bestimmung eines kürzesten Pfades in diesem Track-Graph wird eine Laufzeitschranke angegeben. Diese ist im Unterschied zu Hetzel [1998] auch für mehrere Zielknoten gültig. Für nur einen Zielknoten ist diese Schranke zudem schärfer im Vergleich zu Hetzel [1998].

Mit den theoretischen Ergebnissen dieser Arbeit wurde die entsprechend Hetzel [1998] vorhandene Implementierung vom Autor erweitert. Als Beispiel für die Suche eines kürzesten Pfades ist am Ende dieser Arbeit eine Pfadsuche visualisiert.

[1] Very Large Scale Integration
[2] Application-Specific Integrated Circuit

An dieser Stelle möchte ich mich ganz herzlich bei Herrn Prof. Dr. Bernhard Korte und Herrn Prof. Dr. Jens Vygen für die hervorragende Betreuung beim Erstellen dieser Arbeit und die sehr guten Arbeitsbedingungen am Forschungsinstitut für Diskrete Mathematik der Universität Bonn bedanken.

Besonders danke ich auch Dirk Müller, Jun. Prof. Dr. Tim Nieberg, Christian Panten, Dr. Sven Peyer und Christian Schulte, die alle meine Fragen stets engagiert und motivierend beantwortet haben.

Ebenfalls möchte ich mich bei allen anderen Mitarbeitern des Arithmeums für viele Anregungen und eine herzliche Kollegialität bedanken.

Abschließend danke ich meinen Eltern und allen Freunden, die mich während meines Studiums und bei der Erstellung dieser Diplomarbeit in vielfältiger Hinsicht unterstützt haben.

Kapitel 1

Pfadsuche im Detailed Routing

Die Theorie der Probleme im VLSI-Design wird in Vygen [2001] beschrieben. Wie in der Einleitung erwähnt, beschäftigt sich das Routing mit der Bestimmung der Verdrahtung eines Chips. Eine detaillierte Beschreibung dazu findet sich in Rohe [2001] und Peyer [2007].

Die Chipfläche $\mathcal{F}$ eines Chips ist üblicherweise durch einen achsenparallelen Quader gegeben, dessen Seitenlängen durch $x_{\max}, y_{\max}, z_{\max} \in \mathbb{Z}_{\geq 0}$ definiert sind:

$$\mathcal{F} := \left[0, x_{\max}\right] \times \left[0, y_{\max}\right] \times \left[0, z_{\max}\right] \subseteq \mathbb{R}^3_{\geq 0}$$

Der Suchraum für die Bestimmung einer Verdrahtung wird in BonnRoute® innerhalb dieser Chipfläche $\mathcal{F}$ durch einen Graphen modelliert. Hierbei handelt es sich um einen sogenannten „Track-Graph" $\mathcal{G}$, der in Abschnitt 1.1 zusammen mit einer Kantenkostenfunktion c definiert wird. Mit diesem Track-Graph muss eine Menge von Pfaden kürzestmöglich bzgl. c realisiert werden. Dazu nutzt das Programm BonnRoute® ein zweistufiges Verfahren. Für jeden Pfad wird im *Global Routing* ein sogenannter „Korridor" berechnet, eine Teilmenge der Knotenmenge des Track-Graph (vgl. Albrecht [2001], Müller [2002] und Müller und Vygen [2008]). Für einen Pfad mit Startknoten S und Zielknoten T induziert die entsprechende Teilmenge einen Subgraphen $\mathcal{G}'$ von $\mathcal{G}$, in dem im *Detailed Routing* ein kürzester S-T-Pfad bzgl. c bestimmt wird.

Diese kürzesten Pfade werden mit dem Algorithmus von Dijkstra [1959] berechnet. Um die Berechnung schnell auszuführen, wird in Abschnitt 1.2 gezeigt, wie der Algorithmus von Dijkstra so modifiziert werden kann, dass er zielorientiert kürzeste Pfade mit Hilfe eines zulässigen Potenzials bestimmt. Zusätzlich werden in Abschnitt 1.3 zwei Funktionen vorgestellt, die hierzu als zulässiges Potenzial in BonnRoute® genutzt werden und in Hetzel [1995] und Peyer [2007] angegeben sind. In Abschnitt 1.4 werden diese Modifikationen des Algorithmus von Dijkstra schließlich als Prozedur formuliert, die durch Hetzel [1995] und Peyer et al. [2006] motiviert ist.

In Abschnitt 1.5 wird kurz auf einen anderen Ansatz zur Bestimmung der Verdrahtung eines Chips eingegangen, bei dem der Suchraum nicht durch einen Track-Graph modelliert ist.

1.1 Definition des Track-Graph $\mathcal{G}$

Der Track-Graph $\mathcal{G}$ ist ein gerichteter Graph. Die verwendete graphentheoretische Notation orientiert sich im Folgenden an Korte und Vygen [2008].

1.1.1 Knotenmenge des Track-Graph $\mathcal{G}$

Die Knotenmenge $\mathrm{V}(\mathcal{G})$ ist Teilmenge der ganzzahligen Punkte, die in der Chipfläche $\mathcal{F}$ enthalten sind. Es gilt mit den nachfolgend definierten Mengen X, Y und Z:

$$\mathrm{V}(\mathcal{G}) \subseteq \mathrm{X} \times \mathrm{Y} \times \mathrm{Z} = \mathcal{F} \cap \mathbb{Z}^3_{\geq 0}$$

Definition 1.1.1:
Die Mengen X, Y *und* Z *sind definiert als:*

$$\mathrm{X} := \{0, \ldots, x_{max}\}$$
$$\mathrm{Y} := \{0, \ldots, y_{max}\}$$
$$\mathrm{Z} := \{0, \ldots, z_{max}\}$$

Im Folgenden wird die Knotenmenge $\mathrm{V}(\mathcal{G})$ als Vereinigung disjunkter Mengen definiert, die jeweils die Knoten einer sogenannten Verdrahtungsebene enthalten. Jede Verdrahtungsebene ist mit einem Wert $z \in \mathrm{Z}$ assoziiert und für jede dieser Ebenen ist eine Vorzugsrichtung bestimmt.

Definition 1.1.2:
Die Vorzugsrichtung $\mathrm{VR}(z)$ *einer Verdrahtungsebene* $z \in \mathrm{Z}$ *ist entweder als horizontal* $[\leftrightarrows]$ *oder als vertikal* $[\updownarrow]$ *definiert.*

Wie in der Praxis üblich, wird im Folgenden angenommen, dass die Vorzugsrichtungen von zwei benachbarten Ebenen unterschiedlich definiert sind:

$$\forall z, z' \in \mathrm{Z}: \left(|z' - z| = 1 \Rightarrow \mathrm{VR}(z) \neq \mathrm{VR}(z')\right)$$

In Abhängigkeit von der Vorzugsrichtung ist für jede Verdrahtungsebene $z \in \mathrm{Z}$ eine Menge T_z angegeben, deren Elemente als sogenannte „Track-Koordinaten“ bezeichnet werden. Für diese Mengen gilt:

$$\forall z \in \mathrm{Z}: \ \mathrm{T}_z \subseteq \begin{cases} \mathrm{Y} & \text{falls } \mathrm{VR}(z) = \text{horizontal } [\leftrightarrows] \\ \mathrm{X} & \text{falls } \mathrm{VR}(z) = \text{vertikal } [\updownarrow] \end{cases}$$

Zur leichteren Notation im weiteren Verlauf sind folgende Spezialfälle definiert:

$$\mathrm{T}_{-1} := \emptyset \qquad \mathrm{T}_{z_{\max}+1} := \emptyset$$

Diese Mengen von Track-Koordinaten definieren nun folgendermaßen die x- bzw. y-Koordinaten der Knoten der Verdrahtungsebenen und somit die Knotenmenge $\mathrm{V}(\mathcal{G})$:

Definition 1.1.3:
Die Knotenmenge $\mathrm{V}(\mathcal{G})$ *des Track-Graph* $\mathcal{G}$ *ist definiert als:*

$$\mathrm{V}(\mathcal{G}) := \bigcup_{z \in \mathrm{Z}}^{\cdot} \mathrm{V}_z$$

Die Menge V_z *ist für eine Ebene* $z \in \mathrm{Z}$ *definiert durch:*

$$\mathrm{V}_z := \begin{cases} (\mathrm{T}_{z-1} \cup \mathrm{T}_{z+1}) \times \mathrm{T}_z \times \{z\} & \textit{falls } \mathrm{VR}(z) = \textit{horizontal } [\leftrightarrows] \\ \mathrm{T}_z \times (\mathrm{T}_{z-1} \cup \mathrm{T}_{z+1}) \times \{z\} & \textit{falls } \mathrm{VR}(z) = \textit{vertikal } [\updownarrow] \end{cases}$$

Zur Veranschaulichung dieser Definition ist ein Beispiel für einen Track-Graph $\mathcal{G}$ in Abbildung 1.1.1 auf Seite 4 angegeben.

Da die Knotenmenge $\mathrm{V}(\mathcal{G})$ als Teilmenge von $\mathbb{Z}^3_{\geq 0}$ angegeben ist, sind die Komponenten eines Knotens $v \in \mathrm{V}(\mathcal{G})$ als v_1, v_2 und v_3 definiert:

Definition 1.1.4:
Für einen Knoten $v = (x, y, z) \in \mathrm{V}(\mathcal{G})$ sind die Komponenten von v definiert durch

$$v_1 := x \qquad v_2 := y \qquad v_3 := z.$$

1.1.2 Kantenmenge des Track-Graph $\mathcal{G}$

Die Definition der Kantenmenge $\mathrm{E}(\mathcal{G})$ erfolgt mit Hilfe der beiden Funktionen $\mathcal{X}_z$ und $\mathcal{Y}_z$, die für jede Verdrahtungsebene $z \in \mathrm{Z}$ folgendermaßen definiert sind:

Definition 1.1.5:
Für eine Verdrahtungsebene $z \in \mathrm{Z}$ gelte mit $x_1 < \ldots < x_n$ und $y_1 < \ldots < y_m$:

$$\mathrm{V}_z = \{x_1, \ldots, x_n\} \times \{y_1, \ldots, y_m\} \times \{z\}$$

Die Funktionen $\mathcal{X}_z$ und $\mathcal{Y}_z$ sind jeweils als Projektion auf den Index definiert:

$$\begin{array}{ccc} \mathcal{X}_z : \{x_1, \ldots, x_n\} & \to & \{1, \ldots, n\} \\ x_i & \mapsto & i \end{array} \qquad \begin{array}{ccc} \mathcal{Y}_z : \{y_1, \ldots, y_m\} & \to & \{1, \ldots, m\} \\ y_j & \mapsto & j \end{array}$$

Die Umkehrfunktionen sind mit Definitionsbereich $\mathbb{Z}$ folgendermaßen definiert:

$$\mathcal{X}_z^{-1}(i) := \begin{cases} \infty & \text{falls } i > n \\ -\infty & \text{falls } i < 1 \\ x_i & \text{sonst} \end{cases} \qquad \mathcal{Y}_z^{-1}(j) := \begin{cases} \infty & \text{falls } j > m \\ -\infty & \text{falls } j < 1 \\ y_j & \text{sonst} \end{cases}$$

Die Kantenmenge des Track-Graph $\mathcal{G}$ ist nun durch folgende Definition bestimmt:

Definition 1.1.6:
Zwischen zwei Knoten $(x, y, z) \in \mathrm{V}(\mathcal{G})$ und $(x', y', z') \in \mathrm{V}(\mathcal{G})$ existiert eine Kante

$$\big((x, y, z), (x', y', z')\big) \in \mathrm{E}(\mathcal{G})$$

genau dann, wenn

$$\Big\| \big(\mathcal{X}_z(x), \mathcal{Y}_z(y)\big) - \big(\mathcal{X}_z(x'), \mathcal{Y}_z(y')\big) \Big\|_1 = 1 \qquad (\text{falls } z = z')$$

$$\big\|(x, y, z) - (x', y', z')\big\|_1 = 1 \qquad (\text{falls } z \neq z').$$

Da die Funktionen $\mathcal{X}_z$ und $\mathcal{Y}_z$ für jede Ebene $z \in \mathrm{Z}$ streng monoton wachsend und damit insbesondere injektiv sind, ist eine Kante $e = \big((x, y, z), (x', y', z')\big) \in \mathrm{E}(\mathcal{G})$ von einem der folgenden drei Typen:

Typ 1:	$x \neq x'$	$y = y'$	$z = z'$
Typ 2:	$x = x'$	$y \neq y'$	$z = z'$
Typ 3:	$x = x'$	$y = y'$	$z \neq z'$

Definition 1.1.7:
*Eine Kante vom Typ 1 ist eine **horizontale** Kante, eine vom Typ 2 eine **vertikale** Kante und eine vom Typ 3 wird als **Via** bezeichnet.*

Entsprechend der Vorzugsrichtung einer Verdrahtungsebene $z \in \mathrm{Z}$ kann eine Kante e mit $e \in \mathrm{E}(\mathcal{G}[\mathrm{V}_z])$ als Kante in oder entgegen der Vorzugsrichtung bezeichnet werden:

Definition 1.1.8:
Für eine Verdrahtungsebene $z \in \mathrm{Z}$ mit horizontaler $[\leftrightarrows]$ Vorzugsrichtung ist eine horizontale Kante $e \in \mathrm{E}(\mathcal{G}[\mathrm{V}_z])$ eine Kante in Vorzugsrichtung. Dementsprechend ist eine vertikale Kante $e \in \mathrm{E}(\mathcal{G}[\mathrm{V}_z])$ eine Kante entgegen der Vorzugsrichtung.
Die Definition für eine Ebene $z \in \mathrm{Z}$ mit vertikaler $[\downarrow\uparrow]$ Vorzugsrichtung erfolgt analog.

Ein Beispiel für den Track-Graph $\mathcal{G}$ eines Chips mit drei Verdrahtungsebenen $\mathrm{Z} = \{0, 1, 2\}$ zeigt nachfolgende Abbildung 1.1.1. Da der Track-Graph $\mathcal{G}$ für jede Kante (v, w) auch die Rückwärtskante (w, v) enthält, zeigt das Beispiel einen ungerichteten Graphen. Jedes Via ist gepunktet dargestellt. Die Track-Koordinaten sind grau eingefärbt und es gilt:

$$\begin{aligned}
&\mathrm{VR}(2) = \text{vertikal } [\downarrow\uparrow] && |\mathrm{T}_2| = 5 && |\mathrm{T}_1 \cup \mathrm{T}_3| = 4\\
&\mathrm{VR}(1) = \text{horizontal } [\leftrightarrows] && |\mathrm{T}_1| = 4 && |\mathrm{T}_0 \cup \mathrm{T}_2| = 9\\
&\mathrm{VR}(0) = \text{vertikal } [\downarrow\uparrow] && |\mathrm{T}_0| = 7 && |\mathrm{T}_{-1} \cup \mathrm{T}_1| = 4
\end{aligned}$$

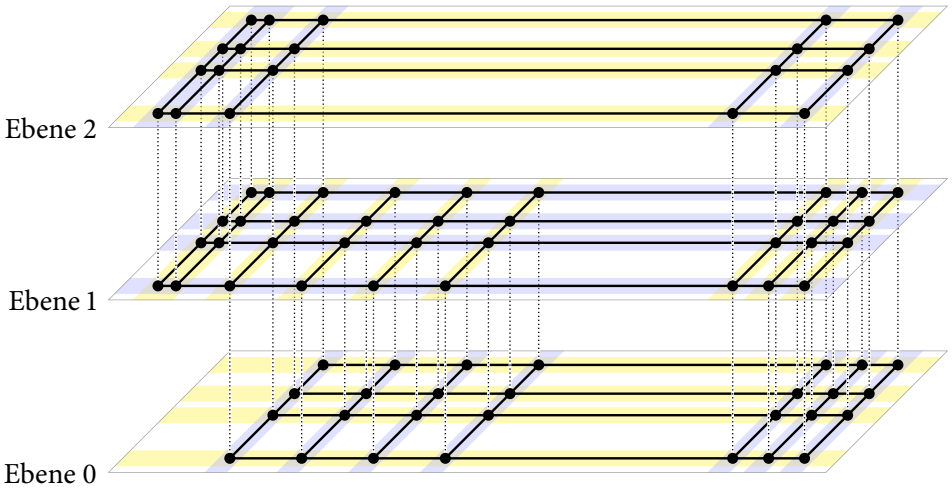


Abbildung 1.1.1: Beispiel für einen Track-Graph $\mathcal{G}$

1.1.3 Kantenkosten des Track-Graph $\mathcal{G}$

Zur Definition der Kantenkostenfunktion $c : \mathrm{E}(\mathcal{G}) \to \mathbb{R}$ des Track-Graph $\mathcal{G}$ ist für jede Verdrahtungsebene $z \in \mathrm{Z}$ ein Vektor c_z angegeben. Für diesen gilt mit einer Ebene $z \in \mathrm{Z}$ und $\mathrm{T}_z = \{t_1, \ldots, t_n\}$:

$$c_z = \left(c_{z,+}, c_{z,-},\ c_{z,\parallel},\ c_{z,t_1,t_2}, \ldots, c_{z,t_{n-1},t_n},\ c_{z,t_n,t_{n-1}}, \ldots, c_{z,t_2,t_1}\right) \in \mathbb{Z}_{>0}^{2n+1}$$

Die Vektoren sind aus numerischen Gründen rational und deshalb ohne Einschränkung ganzzahlig gewählt. Die Einträge dieser Vektoren bestimmen die Proportionalität zwischen den Kosten einer Kante $e = (v, w) \in \mathrm{E}(\mathcal{G})$ und ihrer Länge $\|v - w\|_1$ entsprechend folgender Definition:

Definition 1.1.9:
Die Kantenkostenfunktion $c : \mathrm{E}(\mathcal{G}) \to \mathbb{Z}_{>0}$ *ist für eine Kante*

$$e = \left((x, y, z), (x', y', z')\right) \in \mathrm{E}(\mathcal{G})$$

folgendermaßen definiert:

- *Falls* $z \neq z'$*:*

$$c(e) := \left\|(x, y, z) - (x', y', z')\right\|_1 \cdot \begin{cases} c_{z,+} & \text{falls } z < z' \\ c_{z,-} & \text{falls } z > z' \end{cases}$$

- *Falls* $z = z'$ *und* $\mathrm{VR}(z) =$ *horizontal* $[\leftrightarrows]$*:*

$$c(e) := \left\|(x, y, z) - (x', y', z')\right\|_1 \cdot \begin{cases} c_{z,\parallel} & \text{falls } x \neq x' \\ c_{z,y,y'} & \text{falls } y \neq y' \end{cases}$$

- *Falls* $z = z'$ *und* $\mathrm{VR}(z) =$ *vertikal* $[\updownarrow]$*:*

$$c(e) := \left\|(x, y, z) - (x', y', z')\right\|_1 \cdot \begin{cases} c_{z,x,x'} & \text{falls } x \neq x' \\ c_{z,\parallel} & \text{falls } y \neq y' \end{cases}$$

Für eine Verdrahtungsebene $z \in \mathrm{Z}$ mit horizontaler $[\leftrightarrows]$ Vorzugsrichtung gilt im Allgemeinen $c_{z,y,y'} > c_{z,\parallel}$ für alle $y, y' \in \mathrm{T}_z$ mit $|\mathcal{Y}_z(y) - \mathcal{Y}_z(y')| = 1$. Analog dazu gilt im Allgemeinen $c_{z,x,x'} > c_{z,\parallel}$ für eine Verdrahtungsebene $z \in \mathrm{Z}$ mit vertikaler $[\updownarrow]$ Vorzugsrichtung und $x, x' \in \mathrm{T}_z$ mit $|\mathcal{X}_z(x) - \mathcal{X}_z(x')| = 1$. Daraus folgt, dass ein kürzester Pfad zwischen zwei Knoten $v, w \in \mathrm{V}(\mathcal{G})$ im Allgemeinen größtenteils aus Kanten in Vorzugsrichtung besteht.

In BonnRoute® gilt für die Kosten einer Kante $e = (v, w) \in E(\mathcal{G})$ bislang:

$$c(e) = \begin{cases} \|v - w\|_1 & \text{falls } e \text{ ein Kante in Vorzugsrichtung ist.} \\ \|v - w\|_1 \cdot 4 & \text{falls } e \text{ ein Kante entgegen der Vorzugsrichtung ist.} \\ \|v - w\|_1 \cdot 2600 & \text{falls } e \text{ ein Via ist } (\|v - w\|_1 = 1). \end{cases}$$

Dementsprechend sind die Komponenten eines Vektors c_z für eine Ebene $z \in Z$ folgendermaßen definiert:

- $c_{z,\parallel} = 1$
- $c_{z,+} = c_{z,-} = 2600$
- $c_{z,y,y'} = 4$ für alle $y, y' \in T_z$ mit $|\mathcal{Y}_z(y) - \mathcal{Y}_z(y')| = 1$, falls $VR(z) =$ horizontal [⇆]
- $c_{z,x,x'} = 4$ für alle $x, x' \in T_z$ mit $|\mathcal{X}_z(x) - \mathcal{X}_z(x')| = 1$, falls $VR(z) =$ vertikal [⇵]

1.1.4 Subgraph $\mathcal{G}'$ einer Pfadsuche

Um für zwei Mengen $S, T \subseteq V(\mathcal{G})$ einen kürzesten S-T-Pfad zu bestimmen, steht oftmals nur ein Subgraph $\mathcal{G}'$ von $\mathcal{G}$ zur Verfügung, wie in der Einleitung von Kapitel 1 beschrieben. Dieser muss kein induzierter Subgraph sein. Als Beispiel für einen solchen Graphen $\mathcal{G}'$ ist die nachfolgende Abbildung angegeben. Der gezeigte Subgraph enthält keine Kanten entgegen der Vorzugsrichtung. Dies wird allerdings nicht gefordert. Der Track-Graph $\mathcal{G}$ ist für dieses Beispiel in Abbildung 1.1.1 dargestellt.

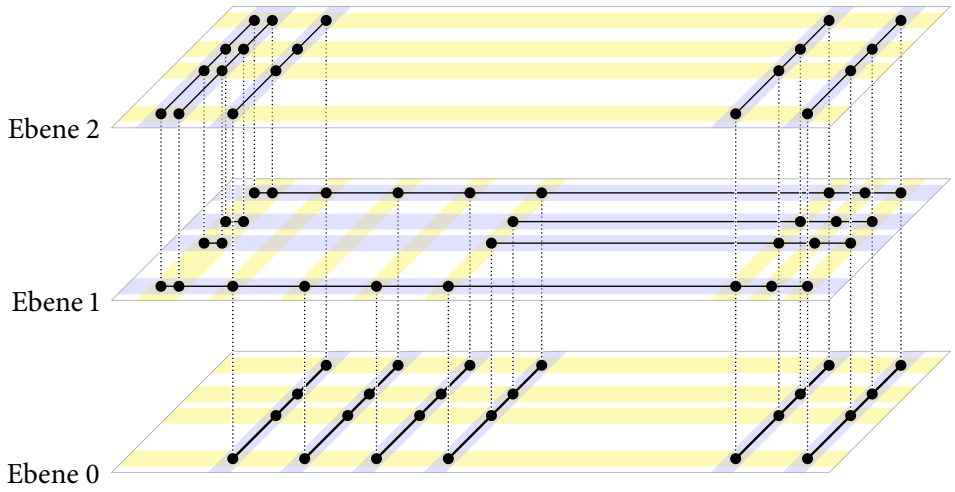


Abbildung 1.1.2: Beispiel für einen Subgraphen $\mathcal{G}'$ von $\mathcal{G}$

Für die nachfolgenden Kapitel werden der Track-Graph $\mathcal{G}$, ein Subgraph $\mathcal{G}'$, die zwei Mengen $S, T \subseteq V(\mathcal{G}')$ und die Kantenkostenfunktion c als gegeben angenommen.

1.2 Beschleunigung des Algorithmus von Dijkstra

Nach Definition ist die Kantenkostenfunktion c nicht negativ. Deshalb kann zur Bestimmung eines kürzesten S-T-Pfades in $\mathcal{G}'$ bzgl. c der Algorithmus von Dijkstra [1959] genutzt werden. In diesem Abschnitt wird gezeigt, dass der Algorithmus mit Hilfe eines zulässigen Potenzials beschleunigt werden kann. Als Resultat wird die Prozedur `Dijkstra_fc` angegeben, die zielgerichtet eine Labelfunktion l berechnet, mit dere Hilfe ein kürzester S-T-Pfad angegeben werden kann.

1.2.1 Reduzierte Kantenkosten

Eine Funktion $\pi : \mathrm{V}(\mathcal{G}') \to \mathbb{R}$, welche die Bedingung

$$\forall\, (v,w) \in \mathrm{E}(\mathcal{G}') :\ c\left((v,w)\right) + \pi\,(v) - \pi\,(w) \geq 0$$

erfüllt, wird als zulässiges Potenzial in $\mathcal{G}'$ bezeichnet. Mit einem zulässigen Potenzial π werden reduzierte Kantenkosten definiert:

Definition 1.2.1:
Die reduzierten Kantenkosten sind für eine Kante $(v,w) \in \mathrm{E}(\mathcal{G}')$ definiert als:

$$c_\pi\left((v,w)\right) := c\left((v,w)\right) + \pi\,(v) - \pi\,(w)$$

Mit der folgenden Definition gilt der nachfolgende Satz, der für zwei Knoten $v, w \in \mathrm{V}(\mathcal{G}')$ den Zusammenhang zwischen einem kürzesten v-w-Pfad bzgl. c und c_π angibt.

Definition 1.2.2:
Für einen Graphen G *und zwei Knotenmengen* $\mathrm{A}, \mathrm{B} \subseteq \mathrm{V}\,(\mathrm{G})$ *wird die Menge aller* A-B-*Pfade in* G *mit* $\mathcal{P}_{\mathrm{G}}\,(\mathrm{A},\mathrm{B})$ *bezeichnet.*

Satz 1.2.3:
Für zwei Knoten $v, w \in \mathrm{V}(\mathcal{G}')$ gilt folgender Zusammenhang zwischen der Länge eines kürzesten v-w-Pfades in $\mathcal{G}'$ bzgl. c und c_π:

$$\mathrm{dist}_{\mathcal{G}',c_\pi}\,(v,w) = \mathrm{dist}_{\mathcal{G}',c}\,(v,w) + \pi\,(v) - \pi\,(w)$$

Beweis:
Falls $\mathcal{P}_{\mathcal{G}'}\left(\{v\},\{w\}\right) = \emptyset$, dann gilt:

$$\mathrm{dist}_{\mathcal{G}',c_\pi}\,(v,w) = \infty = \mathrm{dist}_{\mathcal{G}',c}\,(v,w)$$

Falls $\mathcal{P}_{\mathcal{G}'}\left(\{v\},\{w\}\right) \neq \emptyset$, so gilt:

$$\begin{aligned}
\mathrm{dist}_{\mathcal{G}',c_\pi}\,(v,w) &= \min\left\{ \sum_{e\in\mathrm{E}(\mathrm{P})} c_\pi\,(e) \;\middle|\; \mathrm{P} \in \mathcal{P}_{\mathcal{G}'}\left(\{v\},\{w\}\right) \right\} \\
&= \min\left\{ \sum_{e\in\mathrm{E}(\mathrm{P})} c\,(e) + \pi\,(v) - \pi\,(w) \;\middle|\; \mathrm{P} \in \mathcal{P}_{\mathcal{G}'}\left(\{v\},\{w\}\right) \right\} \\
&= \mathrm{dist}_{\mathcal{G}',c}\,(v,w) + \pi\,(v) - \pi\,(w)
\end{aligned}$$

□

Bemerkung 1.2.4:
Damit folgt für zwei Knoten $v, w \in \mathrm{V}(\mathcal{G}')$ *und einen Pfad* $\mathrm{P} \in \mathcal{P}_{\mathcal{G}'}\left(\{v\}, \{w\}\right)$*:*

$$c_\pi(\mathrm{P}) = \mathrm{dist}_{\mathcal{G}', c_\pi}(v, w) \quad \Rightarrow \quad c(\mathrm{P}) = \mathrm{dist}_{\mathcal{G}', c}(v, w)$$

Aus numerischen Gründen wird π als rationale Funktion gefordert, die o.B.d.A. ganzzahlig ist. Betrachte nun das folgende zulässige Potenzial:

Definition 1.2.5:
Eine Funktion $\pi : \mathrm{V}(\mathcal{G}') \to \mathbb{Z}_{\geq 0}$ *wird als* ***future cost*** *bezeichnet, wenn folgendes gilt:*

- $-\pi$ *ist ein zulässiges Potenzial für* c*:*

$$\forall\, (v, w) \in \mathrm{E}(\mathcal{G}') : c\left((v, w)\right) + \left(-\pi\,(v)\right) - \left(-\pi\,(w)\right) \geq 0$$

- π *verschwindet auf den Zielknoten:*

$$\forall\, t \in \mathrm{T} : \pi\,(t) = 0$$

Nach Hart et al. [1968] und Goldberg und Harrelson [2005] gilt folgender Satz:

Satz 1.2.6:
Gilt für eine future cost π_1 *und eine andere future cost* π_2

$$\forall v \in \mathrm{V}(\mathcal{G}') : \ \pi_1(v) \leq \pi_2(v),$$

so folgt für $\mathrm{S} = \{s\}$ *und* $\mathrm{T} = \{t\}$*:*

$$\left\{ v \in \mathrm{V}(\mathcal{G}') \;\middle|\; \mathrm{dist}_{\mathcal{G}', c_{-\pi_2}}(s, v) \leq \mathrm{dist}_{\mathcal{G}', c_{-\pi_2}}(s, t) \right\}$$
$$\subseteq$$
$$\left\{ v \in \mathrm{V}(\mathcal{G}') \;\middle|\; \mathrm{dist}_{\mathcal{G}', c_{-\pi_1}}(s, v) \leq \mathrm{dist}_{\mathcal{G}', c_{-\pi_1}}(s, t) \right\}$$

Da $c_{-\pi}(e) \geq 0$ für alle Kanten $e \in \mathrm{E}(\mathcal{G}')$ gilt, kann für die Bestimmung eines kürzesten S-T-Pfades bzgl. $c_{-\pi}$ der Algorithmus von Dijkstra genutzt werden. Die gleiche Aussage gilt für die Kantenkostenfunktion c. Für $|\mathrm{S}| = 1$ und $|\mathrm{T}| = 1$ ist ein kürzester S-T-Pfad bzgl. $c_{-\pi}$ nach Bemerkung 1.2.4 auch ein kürzester S-T-Pfad bzgl. c, allerdings kann der Algorithmus von Dijkstra nach Satz 1.2.6 früher terminieren, wenn ein kürzester S-T-Pfad bzgl. $c_{-\pi}$ bestimmt wird.

Die Bezeichnung *future cost* begründet sich durch folgenden Satz:

Satz 1.2.7:
Für einen Knoten $v \in \mathrm{V}(\mathcal{G}')$ *und eine future cost* π *ist* $\pi\,(v)$ *eine untere Schranke für einen kürzesten* v*-*T*-Pfad in* $\mathcal{G}'$ *bzgl.* c*:*

$$\forall v \in \mathrm{V}(\mathcal{G}') : \pi\,(v) \leq \mathrm{dist}_{\mathcal{G}', c}\,(v, \mathrm{T})$$

Beweis:
Betrachte einen Knoten $v \in \mathrm{V}(\mathcal{G}')$ und einen Knoten $t \in \mathrm{T}$. Für einen kürzesten v-t-Pfad P in $\mathcal{G}'$ gilt:

$$c_{-\pi}(\mathrm{P}) = \sum_{e \in \mathrm{E}(\mathrm{P})} c(e) + \big(-\pi(v)\big) - \big(-\pi(t)\big) \geq 0$$
$$\Rightarrow \quad \pi(v) - \pi(t) \leq \sum_{e \in \mathrm{E}(\mathrm{P})} c(e) = \mathrm{dist}_{\mathcal{G}',c}(v,t)$$

Für jedes $t \in \mathrm{T}$ gilt nach Voraussetzung $\pi(t) = 0$.
Falls es keinen v-T-Pfad P gibt, gilt $\mathrm{dist}_{\mathcal{G}',c}(v,\mathrm{T}) = \infty$. □

Für $|\mathrm{S}| > 1$ oder $|\mathrm{T}| > 1$ kann ein kürzester S-T-Pfad nicht wie zuvor angegeben bestimmt werden. Deshalb betrachte den Graphen $\bar{\mathcal{G}}'$, der wie folgt definiert ist:

$$\mathrm{V}(\bar{\mathcal{G}}') := \mathrm{V}(\mathcal{G}') \cup \{\bar{s}\} \cup \{\bar{t}\}$$
$$\mathrm{E}(\bar{\mathcal{G}}') := \mathrm{E}(\mathcal{G}') \cup \big\{(\bar{s},v) \;\big|\; v \in \mathrm{S}\big\} \cup \big\{(v,\bar{t}) \;\big|\; v \in \mathrm{T}\big\}$$

Der Graph $\bar{\mathcal{G}}'$ entsteht also aus $\mathcal{G}'$, indem ein Knoten $\bar{s}$ und eine Kante $(\bar{s},v)$ für jeden Startknoten $v \in \mathrm{S}$ zu $\mathcal{G}'$ und ein Knoten $\bar{t}$ und eine Kante $(v,\bar{t})$ für jeden Zielknoten $v \in \mathrm{T}$ zu $\mathcal{G}'$ hinzugefügt wird. Mit einer *future cost* π definiere für diesen neuen Graphen $\bar{\mathcal{G}}'$ das Potenzial $\bar{\pi}$

$$\bar{\pi}(v) := \begin{cases} 0 & \text{falls } v \in \{\bar{s},\bar{t}\} \\ \pi(v) & \text{falls } v \in \mathrm{V}(\mathcal{G}') \end{cases}$$

und die Kantenkosten $\bar{c}$

$$\bar{c}(e) := \begin{cases} 0 & \text{falls } e \notin \mathrm{E}(\mathcal{G}') \\ c(e) & \text{falls } e \in \mathrm{E}(\mathcal{G}'). \end{cases}$$

Mit $\bar{c}_{-\bar{\pi}}\big((v,w)\big) := c\big((v,w)\big) - \bar{\pi}(v) + \bar{\pi}(w)$ für eine Kante $(v,w) \in \mathrm{E}(\bar{\mathcal{G}}')$ ist dann ein kürzester $\bar{s}$-$\bar{t}$-Pfad $\bar{\mathrm{P}}$ bzgl. $\bar{c}_{-\bar{\pi}}$ nach gleicher Argumentation wie oben auch ein kürzester $\bar{s}$-$\bar{t}$-Pfad bzgl. $\bar{c}$. Der Pfad $\mathrm{P} := \bar{\mathrm{P}}[\mathrm{V}(\mathcal{G}')]$ ist nach Defintion von $\bar{c}$ schließlich ein kürzester S-T-Pfad bzgl. c.

Bemerkung 1.2.8:
Das Potenzial $-\bar{\pi}$ ist zulässig für $\bar{\mathcal{G}}'$, da für die zusätzlichen Kanten gilt:

$$\forall v \in \mathrm{S}: \; \underbrace{\bar{c}\big((\bar{s},v)\big)}_{=0} - \underbrace{\bar{\pi}(\bar{s})}_{=0} + \underbrace{\bar{\pi}(v)}_{\geq 0} \geq 0$$

$$\forall v \in \mathrm{T}: \; \underbrace{\bar{c}\big((v,\bar{t})\big)}_{=0} - \underbrace{\bar{\pi}(v)}_{=0} + \underbrace{\bar{\pi}(\bar{t})}_{=0} \geq 0$$

1.2.2 Prozedur `Dijkstra_fc`

In dem Graphen $\bar{\mathcal{G}}'$ soll mit dem Algorithmus von Dijkstra [1959] ein kürzester $\bar{s}$-$\bar{t}$-Pfad in $\bar{\mathcal{G}}'$ bzgl. $\bar{c}_{-\bar{\pi}}$ bestimmt werden. Dazu wird die nachfolgende Prozedur `Dijkstra_fc` genutzt, die mit dem Graphen $\mathcal{G}'$, einer *future cost* π und reduzierten Kantenkosten $c_{-\pi}$ die Labelfunktion

$$l : \mathrm{V}(\mathcal{G}') \to \mathbb{Z}_{\geq 0}$$

bestimmt.

Prozedur `Dijkstra_fc(S, T)`

① ***Initialisierung***
- **für alle** $w \in \mathrm{V}(\mathcal{G}')$ **tue**
  - **wenn** $w \in \mathrm{S}$ **dann** $l(w) \leftarrow \pi(w)$ **sonst** $l(w) \leftarrow \infty$
- $\delta \leftarrow \min\left\{ l(w) \;\middle|\; w \in \mathrm{V}(\mathcal{G}') \right\}$

② ***Setze Labels***
- **solange** $\delta < \infty$ **und** $\nexists v \in \mathrm{T} : l(v) < \delta$ **tue**
  - **für alle** $v \in \mathrm{V}(\mathcal{G}') : l(v) = \delta$ **tue**
    - **für alle** $w \in \Gamma^+(v)$ **tue**
      - $l(w) \leftarrow \min\left\{ l(w), \delta + c_{-\pi}\left((v,w)\right) \right\}$
  - $\delta \leftarrow \min\left\{ l(v) \;\middle|\; v \in \mathrm{V}(\mathcal{G}') : l(v) > \delta \right\}$

Das Ausführen der Operation

für alle $w \in \Gamma^+(v)$ **tue**
- $l(w) \leftarrow \min\left\{ l(w), \delta + c_{-\pi}\left((v,w)\right) \right\}$

für einen Knoten $v \in \mathrm{V}(\mathcal{G}')$ wird im Folgenden als „permanent labeln“ von v zu Kosten δ bezeichnet. Die Prozedur `Dijkstra_fc` operiert nicht auf dem Graphen $\bar{\mathcal{G}}'$ sondern auf $\mathcal{G}'$, sodass in den ersten beiden Zeilen der Initialisierung ① der Knoten $\bar{s}$ zu Kosten 0 permanent gelabelt wird. Hierbei gilt für die Kanten $(\bar{s}, w) \in \delta^+(\bar{s})$:

$$\bar{c}_{-\bar{\pi}}\left((\bar{s},w)\right) = \underbrace{\bar{c}\left((\bar{s},w)\right)}_{=0} - \underbrace{\bar{\pi}(\bar{s})}_{=0} + \bar{\pi}(w) = \bar{\pi}(w) = \pi(w)$$

Bezeichnet δ_k den Wert δ in Iteration k der äußersten Schleife in ②, dann ist die Anzahl n der Iterationen implizit dadurch gegeben, dass $\delta_n = \infty$ gilt oder $n-1$ der kleinste Index ist, für den ein Zielknoten $v \in \mathrm{T}$ zu Kosten δ_{n-1} permanent gelabelt wurde. Wegen $\bar{c}_{-\bar{\pi}}(e) = 0$ für alle Kanten $e \in \delta^-(\bar{t})$, gilt $\mathrm{dist}_{\bar{\mathcal{G}}', \bar{c}_{-\bar{\pi}}}(\bar{s}, \bar{t}) = \min\left\{ l(v) \;\middle|\; v \in \mathrm{T} \right\}$. Deshalb ergibt sich ein kürzester S-T-Pfad nach Terminierung der Prozedur `Suche_Pfad` wie folgt:

Bemerkung 1.2.9:
Für einen Knoten $v \in \mathrm{T}$ *mit* $l(v) < \infty$ *folgt* $\mathcal{P}_{\bar{\mathcal{G}}'}(\bar{s}, v) \neq \emptyset$. *Wird für jeden Knoten* $w \in \mathrm{V}(\mathcal{G}')$ *mit* $l(w) < \infty$ *ein beliebiger Knoten*

$$p(w) \in \left\{ v' \in \Gamma^-(w) \;\middle|\; l(v') + \bar{c}_{-\bar{\pi}}\big((v', w)\big) = l(w) \right\}$$

gewählt, kann mit der Funktion p *leicht ein Pfad* $\mathrm{P} \in \mathcal{P}_{\bar{\mathcal{G}}'}(\bar{s}, v)$ *mit*

$$\bar{c}_{-\bar{\pi}}(\mathrm{P}) = l(v) = \mathrm{dist}_{\bar{\mathcal{G}}', \bar{c}_{-\bar{\pi}}}(\bar{s}, v)$$

bestimmt werden. Da $\bar{c}_{-\bar{\pi}}(e) = 0$ *und* $\bar{c}(e) = 0$ *für alle Kanten* $e \in \delta^-(\bar{t})$, *ist der Pfad* P *dann ein kürzester* $\bar{s}$-T-*Pfad in* $\bar{\mathcal{G}}'$ *bzgl.* $\bar{c}$ *und somit* $\mathrm{P}[\mathrm{V}(\mathcal{G}')]$ *ein kürzester* S-T-*Pfad in* $\mathcal{G}'$ *bzgl.* c.

Die Prozedur `Dijkstra_fc` kann früher terminieren:

Bemerkung 1.2.10:
Nach Definition des Algorithmus von Dijkstra kann die Abfrage

$$\nexists v \in \mathrm{T} : l(v) < \delta$$

in der äußersten Schleife von Abschnitt ② durch

$$\nexists v \in \mathrm{T} : l(v) \leq \delta$$

ersetzt werden und Bemerkung 1.2.9 ist weiterhin gültig. Wird die Laufzeit der Prozedur `Dijkstra_fc` *analysiert, muss davon ausgegangen werden, dass ein Knoten* $v \in \mathrm{T}$ *der letzte zu Kosten* δ_{n-1} *permanent gelabelte Knoten ist, falls* $\mathcal{P}_{\bar{\mathcal{G}}'}(\bar{s}, \mathrm{T}) \neq \emptyset$. *Deshalb soll die obige Abfrage nicht ersetzt werden.*

1.3 Future Cost

In diesem Kapitel werden zwei Funktionen vorgestellt, die als *future cost* π in der Prozedur `Dijkstra_fc` genutzt werden können, wenn ein Wert $c_\perp \in \mathbb{Z}_{>0}$ existiert, sodass für die Kosten einer Kante $e = (v, w) = \big((x, y, z), (x', y', z')\big) \in \mathrm{E}(\mathcal{G})$ gilt:

$$c(e) = \begin{cases} \|v - w\|_1 \cdot c_\parallel & \text{falls } e \text{ ein Kante in Vorzugsrichtung ist} \\ \|v - w\|_1 \cdot c_\perp & \text{falls } e \text{ ein Kante entgegen der Vorzugsrichtung ist} \\ \|v - w\|_1 \cdot c_{z,+} & \text{falls } z < z' \\ \|v - w\|_1 \cdot c_{z,-} & \text{falls } z > z' \end{cases}$$

Beide Funktionen sind dadurch motiviert, dass der Wert $\pi(v)$ nach Satz 1.2.7 für jeden Knoten $v \in \mathrm{V}(\mathcal{G}')$ eine untere Schranke für die Länge eines kürzesten v-T-Pfades bzgl. c in $\mathcal{G}'$ ist.

1.3.1 Future Cost π_H nach Hetzel [1995]

In Hetzel [1995] wird eine Funktion vorgeschlagen, die durch die L_1-Norm motiviert ist und als *future cost* genutzt werden kann. Diese Funktion ist auf einem induzierten Subgraphen G des dreidimensionalen Gittergraphen mit Knotenmenge $\mathbb{Z}^3$ definiert. Es gilt $\mathrm{V}(\mathrm{G}) = \mathbb{Z}^3 \cap \mathcal{F}$ und zwischen zwei Knoten $(x, y, z) \in \mathrm{V}(\mathrm{G})$ und $(x', y', z') \in \mathrm{V}(\mathrm{G})$ existiert genau dann eine Kante, wenn $|x - x'| + |y - y'| + |z - z'| = 1$ gilt.

Zur Definition von π_H werden zunächst die minimale und maximale z-Koordinate der Zielknoten mit $z^\mathrm{T}_\mathrm{min}$ und $z^\mathrm{T}_\mathrm{max}$ bezeichnet:

$$z^\mathrm{T}_\mathrm{min} := \min\{v_3 \mid v \in \mathrm{T}\} \qquad z^\mathrm{T}_\mathrm{max} := \max\{v_3 \mid v \in \mathrm{T}\}$$

Die Menge T der Zielknoten soll durch eine Menge minimaler Kardinalität $\mathcal{T}$ von disjunkten Rechtecken der Form

$$\mathrm{R} = \big[x, x'\big] \times \big[y, y'\big] \times \{z\} \text{ mit } (x, y, z), (x', y', z) \in \mathrm{T} : x \le x', y \le y'$$

dargestellt sein, sodass gilt:

$$\mathrm{T} = \bigdot\bigcup_{\mathrm{R} \in \mathcal{T}} \big(\mathrm{R} \cap \mathrm{V}(\mathcal{G}')\big)$$

Mit dieser Definition wird die Menge $\mathrm{T}'(z_0)$ für eine Verdrahtungsebene $z_0 \in \mathrm{Z}$ definiert:

$$\mathrm{T}'(z_0) := \left\{ (x, y, z_0) \in \mathrm{V}(\mathrm{G}) \;\middle|\; \exists z : (x, y, z) \in \bigdot\bigcup_{\mathrm{R} \in \mathcal{T}} \mathrm{R} \right\}$$

Mit

$$\mathrm{T}' := \bigdot\bigcup_{z_0 \in \{z^\mathrm{T}_\mathrm{min}, \ldots, z^\mathrm{T}_\mathrm{max}\}} \mathrm{T}'(z_0)$$

ist der Wert $\pi_H(v)$ für einen Knoten $v \in V(G)$ durch die Länge eines kürzesten v-T'-Pfades in G bzgl. c' definiert:

$$\begin{aligned} \pi_H : V(G) &\to \mathbb{Z}_{\geq 0} \\ v &\mapsto \mathrm{dist}_{G,c'}(v, T') \end{aligned}$$

Die Kantenkosten c' sind dabei für eine Kante $e = \big((x,y,z),(x',y',z')\big) \in E(G)$ gegeben durch:

$$c'(e) := \big\|(x,y,z) - (x',y',z')\big\|_1 \cdot \begin{cases} \min\{c_\parallel, c_\perp\} & \text{falls } x \neq x' \\ \min\{c_\parallel, c_\perp\} & \text{falls } y \neq y' \\ \min\{c_{z,-}, c_{z',+}\} & \text{falls } z > z' \\ \min\{c_{z,+}, c_{z',-}\} & \text{falls } z < z' \end{cases}$$

Da der Graph G für jede Kante $(v,w) \in E(G)$ auch die Rückwärtskante (w,v) enthält und für die Kantenkosten $c'\big((v,w)\big) = c'\big((w,v)\big)$ gilt, folgt:

$$\forall v \in V(G) : \ \mathrm{dist}_{G,c'}(v, T') = \mathrm{dist}_{G,c'}(T', v)$$

Deshalb ist der Wert $\pi_H(v)$ ebenfalls durch $\mathrm{dist}_{G,c'}(T', v)$ gegeben und entspricht damit der Länge eines kürzesten T'-v-Pfades in G bzgl. c'.

Zu zeigen ist, dass π_H als *future cost* in $\mathcal{G}'$ genutzt werden kann. Es folgt nach Definition von c' und da T' Obermenge der Zielknoten T ist, dass $\pi_H(v)$ eine untere Schranke für die Länge eines kürzesten v-T-Pfades in $\mathcal{G}$ bzgl. c ist:

$$\pi_H(v) = \mathrm{dist}_{G,c'}(v, T') \leq \mathrm{dist}_{G,c'}(v, T) \leq \mathrm{dist}_{\mathcal{G},c}(v, T)$$

Somit ist eine notwendige Bedingung erfüllt, damit π_H als *future cost* genutzt werden kann. Um die hinreichende Bedingung zu zeigen, betrachte eine Kante $(v,w) \in E(\mathcal{G})$. Für diese gilt:

$$\pi_H(v) - \pi_H(w) = \mathrm{dist}_{G,c'}(v, T') - \mathrm{dist}_{G,c'}(w, T') \leq c\big((v,w)\big)$$

Damit ist die Funktion $-\pi_H$ ein zulässiges Potenzial in $\mathcal{G}$. Weil $\pi_H(v) = 0$ für alle $v \in T$ und $\mathcal{G}'$ ein Subgraph von $\mathcal{G}$ ist, folgt deshalb, dass

$$\pi_H\Big|_{V(\mathcal{G}')} : V(\mathcal{G}') \to \mathbb{Z}_{\geq 0}$$

eine *future cost* nach Definition 1.2.5 ist.

In der Praxis sollte die Laufzeit des Algorithmus von Dijkstra eine obere Schranke für die Summe der Laufzeiten zur Berechnung einer *future cost* und der Prozedur `Dijkstra_fc` sein. Deshalb wird die Laufzeit zur Berechnung der Funktion π_H angegeben.

Für $|T| = 1$ kann π_H in konstanter Zeit bestimmt werden. Für $|T| > 1$ gilt nach Hetzel [1995] unter Verwendung sogenannter „Slab-Methoden“ der Satz:

Satz 1.3.1:
Es kann in Zeit

$$\mathcal{O}\left(|\mathcal{T}|^2\right)$$

eine Datenstruktur erzeugt werden, die das Bestimmen von $\pi_H(v)$ für $v \in \mathrm{V}(\mathcal{G}')$ in

$$\mathcal{O}\left(\log|\mathcal{T}|\right)$$

erlaubt.

1.3.2 Future Cost π_P nach Peyer et al. [2006]

In Peyer et al. [2006] wird ein Algorithmus beschrieben, der kürzeste Pfade in einem induzierten Subgraphen G' des dreidimensionalen Gittergraphen mit Knotenmenge $\mathbb{Z}^3$ bestimmt. Hierbei gilt $\mathrm{V}(\mathrm{G}') \subseteq \mathcal{F}$ und $\mathrm{V}(\mathcal{G}') \subseteq \mathrm{V}(\mathrm{G}')$. Für diesen Subgraphen ist eine Kantenkostenfunktion c' folgendermaßen definiert:

$$c'(e) = \begin{cases} \|v-w\|_1 \cdot c_{\|} & \text{falls } e \text{ ein Kante in Vorzugsrichtung ist} \\ \|v-w\|_1 \cdot c_{\perp} & \text{falls } e \text{ ein Kante entgegen der Vorzugsrichtung ist} \\ \|v-w\|_1 \cdot c_{z,+} & \text{falls } z < z' \\ \|v-w\|_1 \cdot c_{z,-} & \text{falls } z > z' \end{cases}$$

Mit

$$\mathrm{T}'' := \bigcup_{\mathrm{R} \in \mathcal{T}}^{\cdot} \left(\mathrm{R} \cap \mathrm{V}(\mathrm{G}')\right)$$

hat der Subgraph G' die Eigenschaft:

$$\forall v \in \mathrm{V}(\mathcal{G}') : \ \mathrm{dist}_{\mathrm{G}',c'}\left(v, \mathrm{T}''\right) \leq \mathrm{dist}_{\mathrm{G}',c'}\left(v, \mathrm{T}\right) \leq \mathrm{dist}_{\mathcal{G}',c}\left(v, \mathrm{T}\right)$$

Der Graph G' wird so gewählt, dass $\mathrm{V}(\mathcal{G}') \subseteq \mathrm{V}(\mathrm{G}')$ gilt, damit Knotenmengen zu Rechtecken zusammengefasst werden können. Auf diesen Rechtecken propagiert der Algorithmus zweidimensionale Funktionen. Das Ergebnis ist die Funktion

$$\begin{aligned} \pi_{\mathrm{P}} : \mathrm{V}(\mathrm{G}') &\to \mathbb{Z}_{\geq 0} \\ v \quad &\mapsto \mathrm{dist}_{\mathrm{G}',c'}\left(v, \mathrm{T}''\right). \end{aligned}$$

Nach gleicher Argumentation wie in Abschnitt 1.3.1 impliziert diese die *future cost*

$$\pi_{\mathrm{P}}\Big|_{\mathrm{V}(\mathcal{G}')} : \mathrm{V}(\mathcal{G}') \to \mathbb{Z}_{\geq 0}.$$

Außerdem folgt für die *future cost* π_{H}:

$$\forall v \in \mathrm{V}(\mathcal{G}') : \ \pi_{\mathrm{H}}(v) \leq \pi_{\mathrm{P}}(v)$$

Daher terminiert die Prozedur `Dijkstra_fc` entsprechend den Ausführungen von Abschnitt 1.2 früher, wenn die Funktion π_{P} anstatt π_{H} als *future cost* genutzt wird.

Wie in Abschnitt 1.1.9 beschrieben, gilt innerhalb von BonnRoute® $c_{\parallel} = 1$, $c_{\perp} = 4$ und $c_{z,+} = c_{z,-} = 2600$ für jede Verdrahtungsebene $z \in Z$. Für diesen Fall ist die empirisch ermittelte durchschnittliche Laufzeit zur Berechnung von π_P und anschließender Bestimmung eines kürzesten S-T-Pfades mit dem im nächsten Kapitel angegebenen Algorithmus `Suche_Pfad` in Peyer [2007] angegeben. Hier wird außerdem ein Hybridverfahren getestet, bei dem zu Beginn der Prozedur `Suche_Pfad` entschieden wird, ob die *future cost* π_H oder π_P genutzt werden soll. Entsprechend Satz 1.2.6 bewirkt die Nutzung von π_P weniger Labeloperationen der Prozedur `Dijkstra_fc`, im Vergleich zu π_H dauert die Berechnung von π_P im Allgemeinen jedoch länger.

1.4 Einteilung in Blöcke

Der Algorithmus, der die in Abschnitt 1.3.2 vorgestellte *future cost* π_P bestimmt, definiert zudem die sogenannte Blockmenge $\mathcal{B}$. In diesem Abschnitt wird diese Menge $\mathcal{B}$ beschrieben. Anschließend wird die Prozedur `Dijkstra_fc` mit Hilfe der Blockmenge $\mathcal{B}$ so modifiziert, dass sich die Prozedur `Block_Dijkstra_fc` ergibt. Diese ist die Grundlage für die im nächsten Kapitel angegebene Prozedur `Suche_Pfad`.

1.4.1 Blockmenge $\mathcal{B}$

Die Blockmenge $\mathcal{B} \subseteq 2^{V(\mathcal{G}')}$ enthält sogenannte Blöcke und es gilt mit $\pi = \pi_P$:

1. Jeder Knoten $v \in V(\mathcal{G}')$ ist Element von genau einem Block:

$$\bigcup_{B \in \mathcal{B}}^{\cdot} B = V(\mathcal{G}')$$

2. Jeder Block enthält mindestens einen Knoten:

$$\forall B \in \mathcal{B} : B \neq \emptyset$$

3. Ein Block $B \in \mathcal{B}$ enthält Knoten von nur einer Verdrahtungsebene:

$$\forall B \in \mathcal{B}\ \exists z \in Z : B \subseteq V_z$$

4. Es existiert eine Funktion $c_{\mathcal{B}} : \mathcal{B} \times \mathcal{B} \to \mathbb{R}$, für die gilt:

$$\begin{aligned} &B, B' \in \mathcal{B} :\ B \subseteq V_z, B' \subseteq V_{z'} \text{ mit } z \neq z' \\ \Rightarrow\quad &\forall (v,w) \in E(B : B') : c_{\pi}\big((v,w)\big) = c_{\mathcal{B}}(B, B') \end{aligned}$$

5. Es existiert eine Ordnung $\mathcal{B} = \{B_0, \ldots, B_n\}$, sodass gilt:

$$B_i, B_j \in \mathcal{B} : i < j \quad \Rightarrow \quad c_{\mathcal{B}}(B_j, B_i) > 0$$

Da die z-Koordinaten aller Knoten eines Blocks $B \in \mathcal{B}$ übereinstimmen, wird folgende Definition angegeben:

Definition 1.4.1:
Für einen Block $B \in \mathcal{B}$ *mit* $B \subseteq V_z$ *definiere* $z(B) := z$. *Die Menge der Track-Koordinaten* T_B *des Blocks* B *ist definiert als:*

$$T_B := \begin{cases} \{v_2 \mid v \in B\} & \textit{falls } VR(z) = \textit{ horizontal } [\leftrightarrows] \\ \{v_1 \mid v \in B\} & \textit{falls } VR(z) = \textit{ vertikal } [\updownarrow] \end{cases}$$

1.4.2 Prozedur `Block_Dijkstra_fc`

Mit Hilfe der Blockmenge $\mathcal{B}$ ist es möglich, die in der Prozedur `Dijkstra_fc` für einen Knoten $v \in \mathrm{V}(\mathcal{G}')$ ausgeführte Operation

$$l(w) \leftarrow \min\left\{l(w), \delta + c_{\text{-}\pi}\left((v,w)\right)\right\} \tag{1.4.1}$$

nach Peyer et al. [2006] für einige Knoten $w \in \Gamma^+(v)$ nicht sofort, sondern zu einem späteren Zeitpunkt auszuführen. In der folgenden Prozedur `Block_Dijkstra_fc` wird dies realisiert:

Prozedur `Block_Dijkstra_fc(S,T)`

---

① ***Initialisierung***

für alle $\mathrm{B} \in \mathcal{B}, \Delta \in \mathbb{Z}_{\geq 0}$ **tue**
  $\mathcal{R}(\mathrm{B}, \Delta) \leftarrow \emptyset$

für alle $w \in \mathrm{V}(\mathcal{G}')$ **tue**
  wenn $w \in \mathrm{S}$ **dann** $l(w) \leftarrow \pi(w)$ **sonst** $l(w) \leftarrow \infty$

$\delta \leftarrow \min\left\{l(w) \mid w \in \mathrm{V}(\mathcal{G}')\right\}$

② ***Setze Labels***

solange $\delta < \infty$ **und** $\nexists v \in \mathrm{T} : l(v) < \delta$ **tue**
  für alle $\mathrm{B}_i \in \mathcal{B}$ **tue**
    Prozessiere Registrierte
    für alle $w \in \left\{w \in \mathrm{R} \mid \mathrm{R} \in \mathcal{R}(\mathrm{B}_i, \delta)\right\}$ **tue**
      $l(w) \leftarrow \min\left\{l(w), \delta\right\}$
    Prozessiere Block
    für alle $v \in \mathrm{B}_i : l(v) = \delta$ **tue**
      für alle $w \in \Gamma^+(v)$ **tue**
        wenn $w_3 = v_3$ **dann**
          $l(w) \leftarrow \min\left\{l(w), \delta + c_{\text{-}\pi}\left((v,w)\right)\right\}$
        sonst
          Sei $\mathrm{B} \in \mathcal{B}$ der Block mit $w \in \mathrm{B}$.
          $\mathcal{R}\left(\mathrm{B}, \delta + c_{\mathcal{B}}(\mathrm{B}_i, \mathrm{B})\right) \leftarrow \mathcal{R}\left(\mathrm{B}, \delta + c_{\mathcal{B}}(\mathrm{B}_i, \mathrm{B})\right) \cup \left\{\{w\}\right\}$
  Aktualisiere δ
  $\delta \leftarrow \inf\left\{\left\{l(w) \mid w \in \mathrm{V}(\mathcal{G}') : l(w) > \delta\right\} \cup \left\{\Delta > \delta \mid \exists \mathrm{B} \in \mathcal{B} : \mathcal{R}(\mathrm{B}, \Delta) \neq \emptyset\right\}\right\}$

Die Knoten $v \in \mathrm{V}(\mathcal{G}')$, für die die Operation (1.4.1) ausgeführt werden soll, werden nicht in beliebiger Reihenfolge prozessiert, sondern es werden nacheinander die Knoten eines Blocks betrachtet. Die Operation (1.4.1) wird dann mit einem Knoten $v \in \mathrm{B}_i \in \mathcal{B}$ nur für

die Knoten $w \in \Gamma^+(v)$ ausgeführt, für die sich der Block $B \in \mathcal{B}$ mit $w \in B$ auf der gleichen Ebene befindet: $w_3 = v_3$. Falls $w_3 \neq v_3$, wird der Knoten w als Menge $\{w\}$ zu der Menge der registrierten Knoten

$$\mathcal{R}\left(B, \delta + c_{\text{-}\pi}\left((v,w)\right)\right) = \mathcal{R}\big(B, \delta + c_{\mathcal{B}}(B_i, B)\big)$$

hinzugefügt. Damit sind alle in $\mathcal{R}(B,\Delta)$ mit $B \in \mathcal{B}, \Delta \in \mathbb{Z}_{\geq 0}$ enthaltenen Mengen einelementig. Der Grund dafür, dass der Knoten w als Menge $\{w\}$ hinzugefügt wird, wird in Abschnitt 2.5 deutlich. Zu einem späteren Zeitpunkt, wenn ein Knoten des Blocks B zu Kosten $\delta' := \delta + c_{\mathcal{B}}(B_i, B)$ permanent gelabelt werden soll, wird zuerst die Operation

$$l(w) \leftarrow \min\left\{l(w), \delta'\right\}$$

für alle Knoten $w \in B$ mit $\{w\} \in \mathcal{R}(B, \delta')$ ausgeführt werden.

Bezeichnet δ_1 den Wert δ und l_1 die Labelfunktion l nach Terminierung der Prozedur `Dijkstra_fc` und δ_2 den Wert δ und l_2 die Labelfunktion l nach Terminierung der Prozedur `Block_Dijkstra_fc`, so folgt mit obigem Punkt 5 aus Abschnitt 1.4.1:

- $\delta_1 = \delta_2$
- $\left\{v \in V(\mathcal{G}') \;\middle|\; l_1(v) < \delta_1\right\} = \left\{v \in V(\mathcal{G}') \;\middle|\; l_2(v) < \delta_2\right\}$

Damit ist die Korrektheit der Prozedur `Block_Dijkstra_fc` gezeigt, da mit der Labelfunktion l_2 der gleiche kürzeste S-T-Pfad bestimmt werden kann, wie mit der Labelfunktion l_1.

Bemerkung 1.4.2:
Die Prozedur `Block_Dijkstra_fc` *kann entsprechend Bemerkung 1.2.10 früher terminieren, ohne die zuvor bewiesene Korrektheit zu beeinflussen, wenn die Abfrage*

$$\nexists v \in T : l(v) < \delta$$

durch die Abfrage

$$\nexists v \in T : l(v) \leq \delta$$

ersetzt wird.

1.4.3 Blockmenge $\mathcal{B}$ für π_H

Für die in Abschnitt 1.3.1 vorgestellte *future cost* π_H kann ebenfalls eine Blockmenge $\mathcal{B}$ angegeben werden. Mit den dort definierten Werten z^T_{min} und z^T_{max} ist eine Einteilung in Blöcke mit $\mathcal{B} = \{B_0, \ldots, B_{z_{max}}\}$ mit

$$\mathcal{B}' = \left\{V_0, \ldots, V_{z^T_{min}}, V_{z_{max}}, \ldots, V_{z^T_{max}}, V_{z^T_{min}+1}, \ldots, V_{z^T_{max}-1}\right\}$$

gegeben durch:

$$\mathcal{B} := \left\{B'_i \cap V(\mathcal{G}') \;\middle|\; B'_i \in \mathcal{B} : B_i \cap V(\mathcal{G}') \neq \emptyset\right\}$$

Die bzgl. $\mathcal{B}$ in Abschnitt 1.4.1 formulierten Aussagen 1–3 gelten nach Definition von $V(\mathcal{G})$, die Aussagen 4–5 nach Definition von π_H.

1.5 Pfadsuche nach Xing und Kao [2002]

Eine andere Möglichkeit die Verdrahtung eines Chips zu bestimmen, wird in Xing und Kao [2002] vorgestellt. Im Gegensatz zu dem bisher beschriebenen Ansatz, wird der Suchraum nicht durch einen Graphen modelliert. Stattdessen wird ein Verfahren zur Berechnung von achsenparallelen Liniensegmenten angegeben, deren Aneinanderreihung zwei in der Chipfläche $\mathcal{F}$ enthaltene Punkte S und T garantiert kürzestmöglich miteinander verbinden. Um die Berechnung dieser Liniensegmente auszuführen, wird ein Teilgraph eines Hanan-Gitters betrachtet, das durch die in der Chipfläche $\mathcal{F}$ enthaltenen Objekte induziert ist. Auf den Kanten dieses Teilgraphen werden dann stückweise lineare Funktionen propagiert. Die Kantengewichte sind in Abhängigkeit dieses Teilgraphen definiert.

Ein Vorteil dieses Verfahrens besteht offensichtlich darin, dass der Suchraum nicht durch einen Track-Graph modelliert wird und entsprechend eingeschränkt ist. In der Praxis müssen allerdings Millionen von kürzesten Pfaden berechnet werden, sodass in der Regel viele Objekte in der Chipfläche $\mathcal{F}$ enthalten sind. Die angegebenen Laufzeittests wurden auf Testinstanzen ausgeführt, die aus einer einzigen Ebene eines VLSI-Designs generiert wurden. Diese Ebene enthält maximal 7000 Objekte. Gerade wenn ein Großteil der Verdrahtung berechnet wurde, ist die Anzahl der Objekte deutlich größer. Dies impliziert, dass die Teilgraphen der Hanan-Gitter aus vielen Kanten bestehen und deshalb nicht schnell für jede Pfadsuche neu berechnet werden können. Die Laufzeitangabe erfolgt zudem so, dass die Zeit zur Berechnung dieser Teilgraphen nicht enthalten ist.

Kapitel 2

Schnelle Pfadsuche

In diesem Kapitel wird die in BonnRoute® genutzte Prozedur `Suche_Pfad` vorgestellt. Der Subgraph $\mathcal{G}'$ wird durch sogenannte Intervalle dargestellt. Diese ermöglichen es, die Prozedur `Block_Dijkstra_fc` durch das Labeln von Intervallen so zu modifizieren, dass die schnellere Prozedur `Suche_Pfad` resultiert.

In Abschnitt 2.1 wird zunächst die Darstellung von $\mathcal{G}'$ durch Intervalle erklärt. Anschließend wird in Abschnitt 2.2 die Prozedur `Suche_Pfad` vorgestellt. Diese wird in Subprozeduren unterteilt, die in den nachfolgenden Abschnitten optimiert und auf Laufzeit untersucht werden. Die Gesamtlaufzeit der Prozedur `Suche_Pfad` wird schließlich in Abschnitt 2.7 angegeben. Das Ergebnis eines Aufrufes von `Suche_Pfad` ist in Abschnitt 2.8 visualisiert.

Die Funktion $\pi : \mathrm{V}(\mathcal{G}') \to \mathbb{Z}_{\geq 0}$ ist im Folgenden eine *future cost*. In Abhängigkeit von dieser Funktion π soll entsprechend den Ausführungen in Abschnitt 1.3 die Blockmenge $\mathcal{B}$ angegeben sein. Falls π nicht durch eine der in Abschnitt 1.3 beschriebenen Funktionen gegeben ist, muss die Blockmenge $\mathcal{B}$ entsprechend berechnet werden. Die Blockmenge $\mathcal{B}$ soll so gespeichert werden, dass für einen Block $\mathrm{B}_i \in \mathcal{B}$ die Menge $\left\{\mathrm{B} \in \mathcal{B} \setminus \{\mathrm{B}_i\} \;\middle|\; \mathrm{E}(\mathrm{B}_i : \mathrm{B}) \neq \emptyset\right\}$ der „benachbarten Blöcke von B_i“ in Zeit $\mathcal{O}(\log|\mathcal{I}|)$ bestimmt werden kann. Außerdem sollen für jedes Element dieser Menge die Kosten $c_{\mathcal{B}}(\mathrm{B}_i, \mathrm{B})$ ebenfalls in Zeit $\mathcal{O}(\log|\mathcal{I}|)$ bestimmt werden können. Weiterhin wird vorausgesetzt, dass die Funktionen $\mathcal{X}_z$ und $\mathcal{Y}_z$ sowie $\mathcal{X}_z^{-1}$ und $\mathcal{Y}_z^{-1}$ für jede Verdrahtungsebene $z \in \mathrm{Z}$ in konstanter Zeit zur Verfügung stehen. Die Kantenkostenfunktion c soll ebenfalls in konstanter Zeit zur Verfügung stehen. Dies kann jeweils zum Beispiel mit Hilfe eines Arrays ermöglicht werden.

2.1 Intervalle

Der Graph $\mathcal{G}'$ kann üblicherweise durch eine Adjazenzmatrix oder eine Adjazenzliste dargestellt werden. Die Knotenmenge $\mathrm{V}(\mathcal{G}')$ kann in der Praxis allerdings 10^{11} Knoten enthalten. Deshalb ist eine solche Darstellung ineffizient und soll im Folgenden nicht genutzt werden. In diesem Abschnitt wird beschrieben, wie der Graph $\mathcal{G}'$ in der Praxis durch Intervalle dargestellt werden kann und welche Eigenschaften die entstehenden Intervalle besitzen.

2.1.1 Darstellung von $\mathcal{G}'$ mit Intervallen

Es wird zunächst die Intervallmenge $\mathcal{I}$ definiert.

Definition 2.1.1:
Für eine Ebene $z \in \mathrm{Z}$ und eine Track-Koordinate $t \in \mathrm{T}_z$ ist die Intervallmenge $\mathcal{I}_z(t) \subseteq 2^{\mathrm{V}(\mathcal{G}')}$ als Menge minimaler Kardinalität definiert, sodass die folgenden fünf Bedingungen erfüllt sind:

$$\bigcup_{\mathrm{I} \in \mathcal{I}_z(t)}^{\cdot} \mathrm{I} = \mathrm{V}(\mathcal{G}') \cap \begin{cases} (\mathrm{T}_{z-1} \cup \mathrm{T}_{z+1}) \times \{t\} \times \{z\} & \textit{falls } \mathrm{VR}(z) = \textit{horizontal } [\leftrightarrows] \\ \{t\} \times (\mathrm{T}_{z-1} \cup \mathrm{T}_{z+1}) \times \{z\} & \textit{falls } \mathrm{VR}(z) = \textit{vertikal } [\downuparrows] \end{cases} \tag{2.1.1a}$$

$$\forall \mathrm{I} \in \mathcal{I}_z(t)\ \exists \mathrm{B} \in \mathcal{B} : \mathrm{I} \subseteq \mathrm{B} \tag{2.1.1b}$$

$$\forall \mathrm{I} \in \mathcal{I}_z(t) : \left(\mathrm{I} \cap (\mathrm{S} \cup \mathrm{T}) \neq \emptyset \ \Rightarrow\ |\mathrm{I}| = 1\right) \tag{2.1.1c}$$

$$\forall \mathrm{I} \in \mathcal{I}_z(t)\ \exists \alpha_\mathrm{I} \in \{c_\parallel, 0, -c_\parallel\} \forall v, w \in \mathrm{I} :\ \pi(v) - \pi(w) = \alpha_\mathrm{I} \cdot \left(\sum_{i=1}^{2} (v_i - w_i)\right) \tag{2.1.1d}$$

$$\forall \mathrm{I} \in \mathcal{I}_z(t) : \mathcal{G}'[\mathrm{I}] \textit{ ist zusammenhängend} \tag{2.1.1e}$$

Ein Element von $\mathcal{I}_z(t)$ wird als Intervall bezeichnet. Die Menge der Intervalle einer Ebene $z \in \mathrm{Z}$ und die Menge aller Intervalle sind definiert durch:

$$\mathcal{I}(z) := \bigcup_{t \in \mathrm{T}_z}^{\cdot} \mathcal{I}_z(t) \qquad \mathcal{I} := \bigcup_{z \in \mathrm{Z}}^{\cdot} \mathcal{I}(z)$$

Mit dieser Definition kann die Menge der Intervalle eines Blocks $\mathrm{B} \in \mathcal{B}$ angegeben werden:

Definition 2.1.2:
Für eine Verdrahtungsebene $z \in \mathrm{Z}$ und einen Block $\mathrm{B} \in \mathcal{B}$ mit $\mathrm{B} \subseteq \mathrm{V}_z$ ist die Menge der Intervalle einer Track-Koordinate $t \in \mathrm{T}_\mathrm{B}$ von B definiert als:

$$\mathcal{I}_\mathrm{B}(t) := \left\{\mathrm{I} \in \mathcal{I}_z(t) \,\middle|\, \mathrm{I} \subseteq \mathrm{B}\right\}$$

Die Menge aller Intervalle eines Blocks ist definiert als:

$$\mathcal{I}(\mathrm{B}) := \bigcup_{t \in \mathrm{T}_\mathrm{B}}^{\cdot} \mathcal{I}_\mathrm{B}(t)$$

Ein Intervall $\mathrm{I} \in \mathcal{I}$ ist nach Definition eine zusammenhängende Teilmenge der Knotenmenge $\mathrm{V}(\mathcal{G}')$ und jede Kante $e \in \mathrm{E}(\mathcal{G}'[\mathrm{I}])$ ist eine Kante in Vorzugsrichtung. Als Beispiel für eine Menge $\mathcal{I}$ ist die nachfolgende Abbildung 2.1.1 angegeben. Intervalle, die mehr als einen Knoten enthalten, sind rot eingezeichnet. Der Graph $\mathcal{G}'$ entspricht dem in Abbildung 1.1.2 auf Seite 6 dargestellten Subgraphen. Die Menge T ist durch den grün eingezeichneten Knoten gegeben. Die *future cost* π ist durch die in Abschnitt 1.3.1 beschriebene Funktion π_H gegeben. Damit ist die Blockmenge $\mathcal{B}$ gegeben durch:

$$\mathcal{B} = \left\{\mathrm{V}_0 \cap \mathrm{V}(\mathcal{G}'), \mathrm{V}_2 \cap \mathrm{V}(\mathcal{G}'), \mathrm{V}_1 \cap \mathrm{V}(\mathcal{G}')\right\}$$

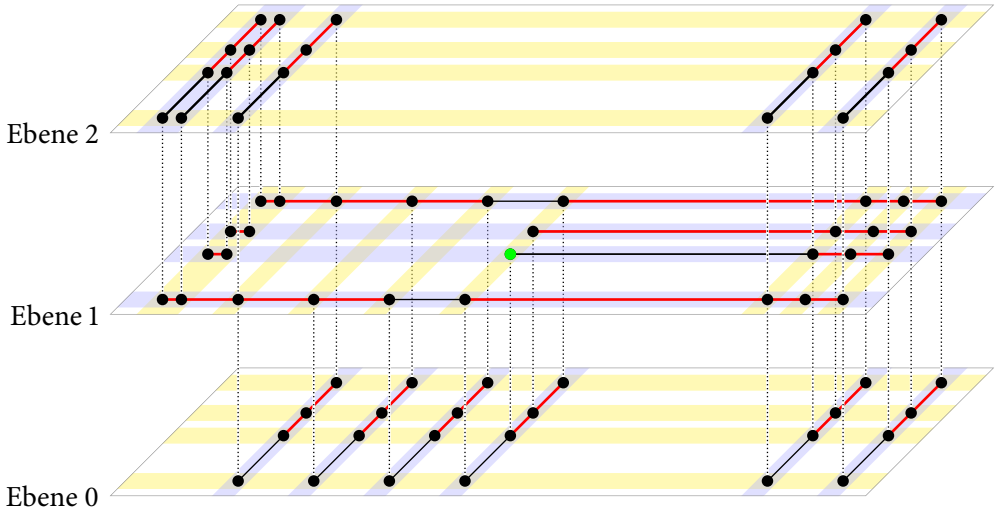


Abbildung 2.1.1: Beispiel für die Intervalle (rot) eines Subgraphen $\mathcal{G}'$

In der Praxis kann die Menge $\mathcal{I}_z(t)$ für eine Verdrahtungsebene $z \in \mathrm{Z}$ und eine Track-Koordinate $t \in \mathrm{T}_z$ als Suchbaum implementiert werden. Folglich kann für einen Knoten $v \in \mathrm{V}(\mathcal{G}')$ in Zeit $\mathcal{O}(\log|\mathcal{I}|)$ das Intervall $\mathrm{I} \in \mathcal{I}$ mit $v \in \mathrm{I}$ bestimmt werden.

Um die Menge $\Gamma^+(v)$ der Nachbarn eines Knotens $v = (x, y, z) \in \mathrm{V}(\mathcal{G}')$ angeben zu können, bezeichne zunächst die Mengen potenzieller Nachbarknoten von v in $\mathcal{G}$ folgendermaßen mit $\mathrm{N_X}(v)$, $\mathrm{N_Y}(v)$ und $\mathrm{N_Z}(v)$:

- potenzielle Nachbarn in x-Richtung
$$\mathrm{N_X}(v) := \left\{\mathcal{X}_z^{-1}\left(\mathcal{X}_z(x)-1\right), \mathcal{X}_z^{-1}\left(\mathcal{X}_z(x)+1\right)\right\} \times \{y\} \times \{z\}$$
- potenzielle Nachbarn in y-Richtung
$$\mathrm{N_Y}(v) := \{x\} \times \left\{\mathcal{Y}_z^{-1}\left(\mathcal{Y}_z(y)-1\right), \mathcal{Y}_z^{-1}\left(\mathcal{Y}_z(y)+1\right)\right\} \times \{z\}$$
- potenzielle Nachbarn in z-Richtung
$$\mathrm{N_Z}(v) := \{x\} \times \{y\} \times \{z-1, z+1\}$$

Nach Definition von $\mathcal{G}$ gilt für einen Knoten $w \in \mathrm{V}(\mathcal{G})$:

$$(v, w) \in \mathrm{E}(\mathcal{G}) \quad \Rightarrow \quad w \in \mathrm{N_X}(v) \cup \mathrm{N_Y}(v) \cup \mathrm{N_Z}(v)$$

Da $\mathcal{G}'$ Subgraph von $\mathcal{G}$ ist, folgt für einen Knoten $v \in \mathrm{V}(\mathcal{G}')$ und das Intervall $\mathrm{I} \in \mathcal{I}$ mit $v \in \mathrm{I}$:

$$(v, w) \in \mathrm{E}\left(\mathcal{G}'\right)$$
$$\Leftrightarrow$$
$$w \in \mathrm{N_X}(v) \cup \mathrm{N_Y}(v) \cup \mathrm{N_Z}(v) \quad \text{und} \quad \exists \mathrm{J} \in \mathcal{I} : w \in \mathrm{J} \wedge \mathrm{E}(\mathrm{I} : \mathrm{J}) \neq \emptyset$$

Die Funktionen $\mathcal{X}_z$, $\mathcal{X}_z^{-1}$ und $\mathcal{Y}_z$, $\mathcal{Y}_z^{-1}$ stehen nach Voraussetzung für alle Ebenen $z \in \mathrm{Z}$ in konstanter Zeit zur Verfügung. Außerdem kann in Zeit $\mathcal{O}(\log|\mathcal{I}|)$ für einen potenziellen Nachbarknoten $w \in \mathrm{N_X}(v) \cup \mathrm{N_Y}(v) \cup \mathrm{N_Z}(v)$ entschieden werden, ob es ein Intervall $\mathrm{J} \in \mathcal{I}$ mit $w \in \mathrm{J}$ gibt. Für dieses Intervall soll es dann möglich sein, in konstanter Zeit zu entscheiden, ob $\mathrm{E}(\mathrm{I} : \mathrm{J}) = \emptyset$ gilt. Damit kann die Menge $\Gamma^+(v)$ in Zeit $\mathcal{O}(\log|\mathcal{I}|)$ bestimmt werden.

2.1.2 Future Cost auf Intervallen

Im weiteren Verlauf werden folgende Definitionen benötigt:

Definition 2.1.3:
Für eine Knotenmenge $\mathrm{I} \subseteq \mathrm{V}(\mathcal{G}')$ *definiere:*

$$\mathrm{X}(\mathrm{I}) := \{v_1 \mid v \in \mathrm{I}\} \qquad \mathrm{Y}(\mathrm{I}) := \{v_2 \mid v \in \mathrm{I}\}$$

Mit $\mathrm{I} \subseteq \mathrm{V}_z$ *definiere* $z(\mathrm{I}) := z$. *Falls* $\mathrm{X}(\mathrm{I}) = \{x\}$ *definiere* $x(\mathrm{I}) := x$ *und falls* $\mathrm{Y}(\mathrm{I}) = \{y\}$ *definiere* $y(\mathrm{I}) := y$.

Die Bedingung (2.1.1d) der Definition der Intervallmenge $\mathcal{I}$ ermöglicht es, den Wert $\pi(v)$ für jeden Knoten $v \in \mathrm{I}$ zu berechnen, wenn $\pi(\bar{v})$ für ein festes $\bar{v} \in \mathrm{I}$ bekannt ist. Deshalb sollen für jedes Intervall $\mathrm{I} \in \mathcal{I}$ die Werte α_I und $\gamma_\mathrm{I} := \pi(\bar{v})$ mit

$$\bar{v} := \left(\min \mathrm{X}(\mathrm{I}), \min \mathrm{Y}(\mathrm{I}), z(\mathrm{I})\right)$$

gespeichert werden. Wegen

$$\forall v \in \mathrm{I}: \ \pi(v) = \gamma_\mathrm{I} + \alpha_\mathrm{I} \cdot (v_1 - \bar{v}_1 + v_2 - \bar{v}_2)$$

wird folgende Funktion für ein Intervall $\mathrm{I} \in \mathcal{I}$ angegeben:

Definition 2.1.4:
Sei $z \in \mathrm{Z}$ *und* $t \in \mathrm{T}_z$. *Für* $\mathrm{I} \in \mathcal{I}_z(t)$ *definiere:*

- *Falls* $\mathrm{VR}(z)$ = *horizontal* $[\leftrightarrows]$*:*

$$\begin{aligned} f_{cost}(\cdot, \mathrm{I}) : \mathrm{X}(\mathrm{I}) &\to \mathbb{Z}_{\geq 0} \\ x &\mapsto \gamma_\mathrm{I} + \alpha_\mathrm{I} \cdot \left(x - \min \mathrm{X}(\mathrm{I})\right) \end{aligned}$$

- *Falls* $\mathrm{VR}(z)$ = *vertikal* $[\downarrow\uparrow]$*:*

$$\begin{aligned} f_{cost}(\cdot, \mathrm{I}) : \mathrm{Y}(\mathrm{I}) &\to \mathbb{Z}_{\geq 0} \\ y &\mapsto \gamma_\mathrm{I} + \alpha_\mathrm{I} \cdot \left(y - \min \mathrm{Y}(\mathrm{I})\right) \end{aligned}$$

Für ein Intervall $\mathrm{I} \in \mathcal{I}$ soll der Wert $\pi(v)$ für einen Knoten $v \in \mathrm{I}$ im Folgenden stets mit Hilfe der oben definierten Funktion $f_{cost}(\cdot, \mathrm{I})$ bestimmt werden.

2.1.3 Reduzierte Kantenkosten auf Intervallen

Betrachte ein Intervall $I \in \mathcal{I}$ und eine Kante $e = (v, w) \in E(\mathcal{G}'[I])$. Im Folgenden wird gezeigt, dass der Wert α_I ausreicht, um die reduzierten Kantenkosten $c_{\text{-}\pi}(e)$ anzugeben. Die Kante e ist nach Definition der Intervallmenge $\mathcal{I}$ eine Kante in Vorzugsrichtung und für die Kantenkosten $c(e)$ gilt:

$$c(e) = c_{\parallel} \cdot \|v - w\|_1 = c_{\parallel} \cdot |v_1 - w_1 + v_2 - w_2|$$

Außerdem gilt mit (2.1.1d):

$$\pi(v) - \pi(w) = \alpha_I \cdot (v_1 - w_1 + v_2 - w_2)$$

Wegen $\alpha_I \in \{c_{\parallel}, 0, -c_{\parallel}\}$ folgt damit für die reduzierten Kantenkosten:

$$\begin{aligned} c_{\text{-}\pi}(e) &= c(e) - \big(\pi(v) - \pi(w)\big) \\ &= c_{\parallel} \cdot |v_1 - w_1 + v_2 - w_2| - \alpha_I \cdot (v_1 - w_1 + v_2 - w_2) \in \big\{0, c(e), c(e) \cdot 2\big\} \end{aligned}$$

Die folgende Tabelle gibt für ein Intervall $I \in \mathcal{I}$ mit $z := z(I)$ die reduzierten Kantenkosten $c_{\text{-}\pi}(e)$ in Abhängigkeit von α_I und der Richtung der Kante $e = (v, w) \in E(\mathcal{G}'[I])$ an:

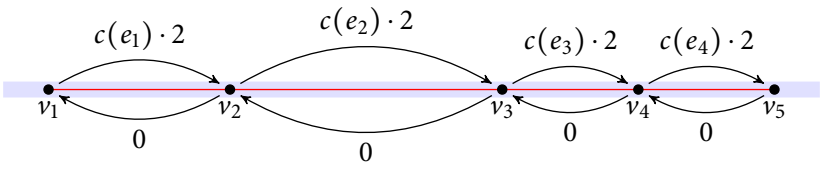
$\mathrm{VR}(z) = \text{horizontal } [\leftrightarrows]$				
		$\alpha_I = c_{\parallel}$	$\alpha_I = 0$	$\alpha_I = -c_{\parallel}$
$(v_1 > w_1)$	$\leftarrow$	0	$c(e)$	$c(e) \cdot 2$
$(v_1 < w_1)$	$\rightarrow$	$c(e) \cdot 2$	$c(e)$	0

$\mathrm{VR}(z) = \text{vertikal } [\downarrow\uparrow]$				
		$\alpha_I = c_{\parallel}$	$\alpha_I = 0$	$\alpha_I = -c_{\parallel}$
$(v_2 > w_2)$	$\downarrow$	0	$c(e)$	$c(e) \cdot 2$
$(v_2 < w_2)$	$\uparrow$	$c(e) \cdot 2$	$c(e)$	0

Als Beispiel für die reduzierten Kantenkosten der Kanten eines Intervalls betrachte ein Intervall $I = \{v_1, v_2, v_3, v_4, v_5\}$ mit $\alpha_I = c_{\parallel}$ und

$$E(\mathcal{G}'[I]) = \big\{(v_i, v_{i+1}) \;\big|\; i \in \{1, \ldots, 4\}\big\} \cup \big\{(v_i, v_{i-1}) \;\big|\; i \in \{2, \ldots, 5\}\big\}$$

Mit $e_i := (v_i, v_{i+1})$ für $i \in \{1, \ldots, 4\}$ sind die reduzierten Kantenkosten folgender Abbildung zu entnehmen:



Für ein Intervall $I \in \mathcal{I}$ können die reduzierten Kantenkosten $c_{\text{-}\pi}(e)$ einer Kante $e = (v, w)$ mit $e \in E(\mathcal{G}'[I])$ also ohne Bestimmung der Differenz $\pi(v) - \pi(w)$ angegeben werden, wenn der Wert α_I bekannt ist.

2.1.4 Anzahl der Intervalle

In BonnRoute® wird die *future cost* $\pi = \pi_H$ bzw. $\pi = \pi_P$ genutzt. In Peyer [2007] ist die Anzahl an Intervallen $|\mathcal{I}|$ im Verhältnis zur Anzahl an Knoten $|V(\mathcal{G}')|$ empirisch ermittelt. Es folgt, dass die Anzahl der Intervalle üblicherweise deutlich kleiner als die Anzahl der Knoten ist:

$$\left|\mathcal{I}\right| \ll \left|V(\mathcal{G}')\right|$$

Die Ergebnisse zeigen, dass die Anzahl der Intervalle für Chips der 130 nm und 90 nm Technologie etwa 23 mal geringer als die Anzahl der Knoten ist.

2.1.5 Prozedur `Label_Intervall`

Im Folgenden wird die Intervallmenge $\mathcal{I}$ und für jedes Intervall $I \in \mathcal{I}$ die Funktion $f_{cost}(\cdot, I)$ und der Wert α_I als gegeben vorausgesetzt. Mit der Darstellung von $\mathcal{G}'$ durch Intervalle wird eine schnelle Realisierung der Prozedur `Block_Dijkstra_fc` im nächsten Abschnitt angegeben, die mit `Suche_Pfad` bezeichnet wird. Diese Prozedur nutzt die Prozedur `Label_Intervall`(Δ, w, J). Der Aufruf erfolgt für einen Wert $\Delta \in \mathbb{Z}_{\geq 0}$, ein Intervall $J \in \mathcal{I}$ und einen Knoten $w \in J$ und bewirkt die folgenden Operationen:

$l(w) \leftarrow \min\left\{l(w), \Delta\right\}$
für alle $\bar{w} \in \Gamma^+(w) \cap J$ **mit** $l(w) + c_{-\pi}\left((w, \bar{w})\right) < l(\bar{w})$ **tue**
  `Label_Intervall`$\left(l(w) + c_{-\pi}\left((w, \bar{w})\right), \bar{w}, J\right)$

In Abschnitt 2.3 wird eine schnelle Realisierung der Prozedur `Label_Intervall` angegeben, die die obigen Operationen nicht explizit ausführt.

2.2 Prozedur `Suche_Pfad`

Die in Abschnitt 1.4 angegebene Prozedur `Block_Dijkstra_fc` wird mit Hilfe der Intervallmenge $\mathcal{I}$ modifiziert. Die neue Prozedur wird mit `Suche_Pfad` bezeichnet und bewirkt folgende Operationen:

① ***Initialisierung***

für alle $B \in \mathcal{B}, \Delta \in \mathbb{Z}_{\geq 0}$ **tue**
  $\mathcal{R}(B, \Delta) \leftarrow \emptyset$
für alle $w \in V(\mathcal{G}')$ **tue**
  Sei $J \in \mathcal{I}$ das Intervall mit $w \in J$.
  wenn $w \in S$ **dann**
    `Label_Intervall`$(\pi(w), w, J)$
  sonst
    `Label_Intervall`(∞, w, J)
$\delta \leftarrow \min\left\{l(w) \mid w \in V(\mathcal{G}')\right\}$

② ***Setze Labels***

solange $\delta < \infty$ **und** $\nexists v \in T : l(v) < \delta$ **tue**
  für alle $B_i \in \mathcal{B}$ **tue**
    Prozessiere Registrierte
    für alle $J \in \mathcal{I}(B_i)$ **tue**
      für alle $w \in \left\{w \in J \cap R \mid R \in \mathcal{R}(B_i, \delta)\right\}$ **tue**
        `Label_Intervall`(δ, w, J)
    Prozessiere Block
    für alle $v \in B_i : l(v) = \delta$ **tue**
      für alle $w \in \Gamma^+(v)$ **tue**
        wenn $w_3 = v_3$ **dann**
          Sei $J \in \mathcal{I}$ das Intervall mit $w \in J$.
          `Label_Intervall`$\left(\delta + c_{-\pi}((v, w)), w, J\right)$
        sonst
          Sei $B \in \mathcal{B}$ der Block mit $w \in B$.
          $\mathcal{R}\left(B, \delta + c_{\mathcal{B}}(B_i, B)\right) \leftarrow \mathcal{R}\left(B, \delta + c_{\mathcal{B}}(B_i, B)\right) \cup \left\{\{w\}\right\}$
  Aktualisiere δ
  $\delta \leftarrow \inf\left\{\left\{l(w) \mid w \in V(\mathcal{G}') : l(w) > \delta\right\} \cup \left\{\Delta > \delta \mid \exists B_i \in \mathcal{B} : \mathcal{R}(B_i, \Delta) \neq \emptyset\right\}\right\}$

Diese Prozedur benutzt die Prozedur `Label_Intervall`, um den Wert $l(w)$ für einen Knoten $w \in V(\mathcal{G}')$ zu modifizieren. Die Anzahl der Aufrufe von `Label_Intervall` kann reduziert werden, wie in den nachfolgenden Abschnitten gezeigt. Daher impliziert eine schnelle Realisierung der Prozedur `Label_Intervall` eine schnelle Realisierung von `Suche_Pfad`.

2.2.1 Korrektheit von `Suche_Pfad`

Bezeichnet l^1 die Labelfunktion l nach Terminierung von `Block_Dijkstra_fc` aus Abschnitt 1.4 und l^2 die Labelfunktion l nach Terminierung der Prozedur `Suche_Pfad`, so wird in diesem Abschnitt gezeigt, dass mit der Funktion l^2 der gleiche kürzeste S-T-Pfad bestimmt werden kann, wie mit der Labelfunktion l^1. Dazu werden zunächst die Unterschiede beider Prozeduren betrachtet, von denen es insgesamt zwei gibt. Der erste Unterschied besteht darin, dass die Labeloperation

$$l(w) \leftarrow \min\left\{l(w), \Delta\right\}$$

für einen Knoten $w \in \mathrm{J}$ durch den Aufruf der Prozedur `Label_Intervall`(Δ, w, J) ersetzt ist. Wie in Abschnitt 2.1.5 angegeben, bewirkt diese Prozedur folgende Operationen:

$l(w) \leftarrow \min\left\{l(w), \Delta\right\}$
für alle $\bar{w} \in \Gamma^+(w) \cap \mathrm{J}$ **mit** $l(w) + c_{-\pi}\left((w, \bar{w})\right) < l(\bar{w})$ **tue**
  `Label_Intervall`$\left(l(w) + c_{-\pi}\left((w, \bar{w})\right), \bar{w}, \mathrm{J}\right)$

Diese Vorgehensweise wird als **„Labeln von Intervallen“** bezeichnet.

Der zweite Unterschied betrifft den mit *Prozessiere Registrierte* bezeichneten Abschnitt der Prozedur `Block_Dijkstra_fc`. Die Operationen

für alle $w \in \left\{w \in \mathrm{R} \;\middle|\; \mathrm{R} \in \mathcal{R}(\mathrm{B}_i, \delta)\right\}$ **tue**
  $l(w) \leftarrow \min\left\{l(w), \delta\right\}$

sind ersetzt durch:

für alle $\mathrm{J} \in \mathcal{I}(\mathrm{B}_i)$ **tue**
  für alle $w \in \left\{w \in \mathrm{J} \cap \mathrm{R} \;\middle|\; \mathrm{R} \in \mathcal{R}(\mathrm{B}_i, \delta)\right\}$ **tue**
    `Label_Intervall`(δ, w, J)

Dies bedeutet allerdings nur, dass die Knoten $w \in \left\{w \in \mathrm{R} \;\middle|\; \mathrm{R} \in \mathcal{R}(\mathrm{B}_i, \delta)\right\}$ in einer vorgegebenen Reihenfolge prozessiert werden, denn nach Definition der Intervallmenge $\mathcal{I}$ gilt:

$$\dot{\bigcup_{\mathrm{J} \in \mathcal{I}(\mathrm{B}_i)}} \mathrm{J} = \mathrm{B}_i$$

Der Unterschied der Prozeduren `Block_Dijkstra_fc` und `Suche_Pfad`, der $l^1(v) \neq l^2(v)$ für einen Knoten $v \in \mathrm{V}(\mathcal{G}')$ implizieren kann, besteht also darin, dass Intervalle gelabelt werden.

Im Folgenden wird gezeigt, dass mit der Labelfunktion l^2 der gleiche kürzeste S-T-Pfad bestimmt werden kann, wie mit der Labelfunktion l^1. Die Prozedur `Block_Dijkstra_fc` wird dazu mit Superskript 1 und die Prozedur `Suche_Pfad` mit Superskript 2 bezeichnet.

Für die Prozedur `Block_Dijkstra_fc` ist δ_k^1 durch den Wert δ der k-ten Iteration und l_k^1 als Labelfunktion l nach der k-ten Iteration der Schleife in Teil ② von `Block_Dijkstra_fc` definiert. Damit ist folgende Menge definiert:

$$N_k^1 := \left\{ v \in V(\mathcal{G}') \;\middle|\; l_k^1(v) = \delta_k \right\}$$

Außerdem bezeichne $\mathcal{R}^1(B_i, \Delta)$ die Mengen $\mathcal{R}(B_i, \Delta)$ mit $B_i \in \mathcal{B}$, $\Delta \in \mathbb{Z}_{\geq 0}$ und $n_{\max}^1$ die Anzahl der Iterationen der Schleife in ②. Analog für die Prozedur `Suche_Pfad`. Der Graph $\bar{\mathcal{G}}'$, der Knoten $\bar{s}$ und die reduzierten Kantenkosten $\bar{c}_{-\bar{\pi}}$ sind entsprechend den Ausführungen in Abschnitt 1.2.1 gegeben.

Bemerkung 2.2.1:
Nach Definition der Prozeduren gilt für jeden Knoten $v \in V(\mathcal{G}')$:

- $\forall k \in \left\{0, \ldots, n_{\max}^1\right\} : \; l_k^1(v) \geq \mathrm{dist}_{\bar{\mathcal{G}}', \bar{c}_{-\bar{\pi}}}(\bar{s}, v)$
- $\forall k \in \left\{0, \ldots, n_{\max}^2\right\} : \; l_k^2(v) \geq \mathrm{dist}_{\bar{\mathcal{G}}', \bar{c}_{-\bar{\pi}}}(\bar{s}, v)$

Bemerkung 2.2.2:
Für alle Knoten $v \in V(\mathcal{G}')$ und alle $k < n_{\max}^1$ gilt wegen der Korrektheit der Prozedur `Block_Dijkstra_fc`*:*

$$\mathrm{dist}_{\bar{\mathcal{G}}', \bar{c}_{-\bar{\pi}}}(\bar{s}, v) = \delta_k^1 \;\Leftrightarrow\; v \in N_k^1$$

Mit obigen Definitionen gilt der Satz:

Satz 2.2.3:
Es gilt $n_{\max}^1 = n_{\max}^2$ und $N_k^1 = N_k^2$ für alle $k \in \left\{0, \ldots, n_{\max}^1\right\}$.

Beweis:
Zeige durch vollständige Induktion:

$$\forall k \in \left\{0, \ldots, \min\left\{n_{\max}^1, n_{\max}^1\right\}\right\} : \; \delta_k^1 = \delta_k^2 \;\wedge\; N_k^1 = N_k^2$$

Daraus folgt $n_{\max}^1 = n_{\max}^2$.

Induktionsanfang:
Nach Definition der Prozedur `Label_Intervall` folgt:

$$\left\{ v \in V(\mathcal{G}') \;\middle|\; l_0^1(v) < \infty \right\} \subseteq \left\{ v \in V(\mathcal{G}') \;\middle|\; l_0^2(v) < \infty \right\}$$

Allerdings gilt $\pi(w) < \pi(v)$ für eine Kante $(v, w) \in E(\mathcal{G}')$ mit $c_{-\pi}\big((v, w)\big) = 0$. Dies folgt aus $\pi(v) - \pi(w) = c\big((v, w)\big)$ und $c\big((v, w)\big) > 0$ nach Definition der Kantenkosten. Daher gilt $\delta_0^1 = \min\left\{\pi(v) \;\middle|\; v \in S\right\} = \min\left\{\pi(v) \;\middle|\; v \in S\right\} = \delta_0^2$ und

$$\left\{ v \in V(\mathcal{G}') \;\middle|\; l_0^1(v) = \delta_0^1 \right\} = \left\{ v \in V(\mathcal{G}') \;\middle|\; l_0^2(v) = \delta_0^2 \right\}.$$

Für Iteration $k = 0$ gilt also $N_0^1 = N_0^2$ und $\delta_0^1 = \delta_0^2$.

Induktionsschluss:
Es gelte $\mathrm{N}_k^1 = \mathrm{N}_k^2$ und $\delta_k^1 = \delta_k^2$ für alle $k \in \{0, \ldots, n\}$ mit $0 \leq n \leq \min\left\{n_{\max}^1, n_{\max}^2\right\}$. Für $n = \min\left\{n_{\max}^1, n_{\max}^2\right\}$ ist nichts zu zeigen, deshalb sei $n < \min\left\{n_{\max}^1, n_{\max}^2\right\}$.
Zu zeigen ist $\mathrm{N}_{n+1}^1 = \mathrm{N}_{n+1}^2$ und $\delta_{n+1}^1 = \delta_{n+1}^2$. Dies wird durch Widerspruch gezeigt.

Fall 1: Angenommen, es gilt $\mathrm{N}_{n+1}^1 \neq \mathrm{N}_{n+1}^2$ und $\delta_{n+1}^1 = \delta_{n+1}^2$.
(1.1) Durch Widerspruch wird $\mathrm{N}_{n+1}^1 \subseteq \mathrm{N}_{n+1}^2$ gezeigt.
Sei $w \in \mathrm{N}_{n+1}^1 \setminus \mathrm{N}_{n+1}^2$. Dann existiert ein Knoten $v \in \Gamma^-(w)$, sodass gilt:

$$l_{n+1}^1(w) = l_{n+1}^1(v) + c_{\text{-}\pi}\left((v,w)\right)$$

1. Fall: $c_{\text{-}\pi}\left((v,w)\right) > 0$. Wegen $l_{n+1}^1(v) < \delta_{n+1}^1$ existiert ein Index $k \in \{0, \ldots, n\}$, sodass $v \in \mathrm{N}_k^1 = \mathrm{N}_k^2$ gilt. Dann folgt:

$$l_{n+1}^2(w) \leq l_{n+1}^2(v) + c_{\text{-}\pi}\left((v,w)\right) = l_{n+1}^1(v) + c_{\text{-}\pi}\left((v,w)\right) = l_{n+1}^1(w)$$

2. Fall: $c_{\text{-}\pi}\left((v,w)\right) = 0$. Bezeichne $\left(v_k^1\right)_{k\in\mathrm{K}}$ die Folge der Knoten, die in dem mit *Prozessiere Block* von `Block_Dijkstra_fc` gekennzeichneten Abschnitt ausgewählt werden. Da die Funktion $c_{\text{-}\pi}$ der reduzierten Kantenkosten nicht negativ ist, gilt:

$$\forall v' \in \left(\mathrm{V}(\mathcal{G}') \setminus \dot{\bigcup_{k\in\{1,\ldots,n\}}} \mathrm{N}_k^2\right) : \; l_{n+1}^2(v') \geq \delta_{n+1}^2$$

Deshalb können die ersten $|\mathrm{K}|$ Knoten in dem mit *Prozessiere Block* von `Suche_Pfad` gekennzeichneten Abschnitt in genau der gleichen Reihenfolge ausgewählt werden wie in `Block_Dijkstra_fc`. Es folgt:

$$l_{n+1}^2(w) \leq l_{n+1}^1(w)$$

Aus $l_{n+1}^1(w) = \mathrm{dist}_{\bar{\mathcal{G}}',\bar{c}_{-\bar{\pi}}}(\bar{s}, w) \leq l_{n+1}^2(w)$ folgt $l_{n+1}^2(w) = l_{n+1}^1(w) = \delta$. ↯

(1.2) Durch Widerspruch wird $\mathrm{N}_{n+1}^1 \supseteq \mathrm{N}_{n+1}^2$ gezeigt.
Sei $w \in \mathrm{N}_{n+1}^2 \setminus \mathrm{N}_{n+1}^1$. Dann gilt nach Induktionsvoraussetzung:

$$w \notin \dot{\bigcup_{k\in\{1,\ldots,n\}}} \mathrm{N}_k^2 = \dot{\bigcup_{k\in\{1,\ldots,n\}}} \mathrm{N}_k^1$$

Mit der Korrektheit von `Block_Dijkstra_fc` nach Bemerkung 2.2.2 folgt:

$$\mathrm{dist}_{\bar{\mathcal{G}}',\bar{c}_{-\bar{\pi}}}(\bar{s}, w) \underset{\text{Bem. 2.2.2}}{>} l_{n+1}^2(w) \underset{\text{Bem. 2.2.1}}{\geq} \mathrm{dist}_{\bar{\mathcal{G}}',\bar{c}_{-\bar{\pi}}}(\bar{s}, w) \qquad \text{↯}$$

Also gilt $\mathrm{N}_{n+1}^1 = \mathrm{N}_{n+1}^2$ im Widerspruch zur Annahme. Damit ist bewiesen, dass Fall 1 nicht möglich ist.

Fall 2: Angenommen, es gilt $\delta^1_{n+1} \neq \delta^2_{n+1}$.
Es gilt nach der n-ten Iteration $\mathcal{R}^1(\Delta, \mathrm{B}_i) = \mathcal{R}^2(\Delta, \mathrm{B}_i)$ für alle $\mathrm{B}_i \in \mathcal{B}, \Delta \in \mathbb{Z}_{\geq 0}$. Deshalb gilt nach der n-ten Iteration:

$$\inf\left\{\Delta > \delta^1_n \;\middle|\; \exists \mathrm{B}_i \in \mathcal{B} : \mathcal{R}^1(\mathrm{B}_i, \Delta) \neq \emptyset\right\} = \inf\left\{\Delta > \delta^2_n \;\middle|\; \exists \mathrm{B}_i \in \mathcal{B} : \mathcal{R}^2(\mathrm{B}_i, \Delta) \neq \emptyset\right\}$$

Da, wie oben beschrieben, die Knoten in dem mit *Prozessiere Block* bezeichneten Teil von `Suche_Pfad` in der gleichen Reihenfolge ausgewählt werden können, wie in der Prozedur `Block_Dijkstra_fc`, folgt:

$$\inf\left\{l^1_n(v) \;\middle|\; v \in \mathrm{V}(\mathcal{G}') : l^1_n(v) > \delta^1_n\right\} > \inf\left\{l^2_n(v) \;\middle|\; v \in \mathrm{V}(\mathcal{G}') : l^2_n(v) > \delta^2_n\right\}$$

Deshalb muss $\delta^1_{n+1} > \delta^2_{n+1}$ gelten. Wegen der Korrektheit von `Block_Dijkstra_fc` nach Bemerkung 2.2.2 impliziert dies allerdings:

$$\exists v \in \mathrm{V}(\mathcal{G}') \setminus \dot{\bigcup_{k\in\{1,\ldots,n\}}} \mathrm{N}^2_k : \mathrm{dist}_{\bar{\mathcal{G}}', \bar{c}_{-\bar{\pi}}}(\bar{s}, v) \underset{\text{Bem. 2.2.1}}{\leq} l^2_{n+1}(v) = \delta^2_{n+1} \underset{\text{Bem. 2.2.2}}{<} \mathrm{dist}_{\bar{\mathcal{G}}', \bar{c}_{-\bar{\pi}}}(\bar{s}, v) \;\lightning$$

Also gilt $\delta^1_{n+1} = \delta^2_{n+1}$ im Widerspruch zur Annahme. Damit ist bewiesen, dass Fall 2 nicht möglich ist. □

Für $\delta := \delta^1_{n_{\max}} = \delta^2_{n_{\max}}$ gilt demnach:

$$\left\{v \in \mathrm{V}(\mathcal{G}') \;\middle|\; l^1(v) < \delta\right\} = \left\{v \in \mathrm{V}(\mathcal{G}') \;\middle|\; l^2(v) < \delta\right\}$$

Damit ist die Korrektheit der Prozedur `Suche_Pfad` gezeigt, da mit der Labelfunktion l^2 der gleiche kürzeste S-T-Pfad bestimmt werden kann, wie mit der Labelfunktion l^1.

2.2.2 Aufteilung in Prozeduren

Die mit *Prozessiere Registrierte* und *Prozessiere Block* bezeichneten Abschnitte der oben angegebenen Prozedur werden im Folgenden als Prozeduren definiert. Mit diesen beiden Prozeduren wird die Prozedur `Suche_Pfad` angegeben.

Die Prozedur `Prozessiere_Registrierte`$(\mathcal{R}(\mathrm{B}_i, \delta), \mathrm{B}_i, \delta)$ ist definiert durch diese Operationen:

für alle $\mathrm{J} \in \mathcal{I}(\mathrm{B}_i)$ **tue**
  für alle $w \in \left\{w \in \mathrm{J} \cap \mathrm{R} \;\middle|\; \mathrm{R} \in \mathcal{R}(\mathrm{B}_i, \delta)\right\}$ **tue**
    `Label_Intervall`(δ, w, J)

In Abschnitt 2.6 wird eine schnelle Realisierung dieser Prozedur vorgestellt. Weiterhin wird die Prozedur `Prozessiere_Block`(B_i, δ) wie folgt definiert:

für alle $v \in \mathrm{B}_i : l(v) = \delta$ **tue**
  für alle $w \in \Gamma^+(v)$ **tue**
    wenn $w_3 = v_3$ **dann**
      Sei $\mathrm{J} \in \mathcal{I}$ das Intervall mit $w \in \mathrm{J}$.
      `Label_Intervall`$\left(\delta + c_{-\pi}\left((v,w)\right), w, \mathrm{J}\right)$
    sonst
      Sei $\mathrm{B} \in \mathcal{B}$ der Block mit $w \in \mathrm{B}$.
      $\mathcal{R}\left(\mathrm{B}, \delta + c_{\mathcal{B}}(\mathrm{B}_i, \mathrm{B})\right) \leftarrow \mathcal{R}\left(\mathrm{B}, \delta + c_{\mathcal{B}}(\mathrm{B}_i, \mathrm{B})\right) \cup \left\{\{w\}\right\}$

Eine schnelle Realisierung dieser Prozedur wird in Abschnitt 2.5 angegeben. Mit diesen beiden Prozeduren ist eine schnelle Realisierung der Prozedur `Suche_Pfad` gegeben durch:

Prozedur `Suche_Pfad(S, T)`

---

① ***Initialisierung***
  für alle $\mathrm{B} \in \mathcal{B}, \Delta \in \mathbb{Z}_{\geq 0}$ **tue**
    $\mathcal{R}(\mathrm{B}, \Delta) \leftarrow \emptyset$
  für alle $w \in \mathrm{V}(\mathcal{G}')$ **tue**
    Sei $\mathrm{J} \in \mathcal{I}$ das Intervall mit $w \in \mathrm{J}$.
    wenn $w \in \mathrm{S}$ **dann**
      `Label_Intervall`$(\pi(w), w, \mathrm{J})$
    sonst
      `Label_Intervall`(∞, w, J)
  $\delta \leftarrow \min\left\{l(w) \;\middle|\; w \in \mathrm{V}(\mathcal{G}')\right\}$

② ***Permanentlabeln der Knoten mit potenziellen Kosten*** δ
  solange $\delta < \infty$ **und** $\nexists v \in \mathrm{T} : l(v) < \delta$ **tue**
    für alle $\mathrm{B}_i \in \mathcal{B}$ **tue**
      `Prozessiere_Registrierte`$(\mathcal{R}(\mathrm{B}_i, \delta), \mathrm{B}_i, \delta)$
      `Prozessiere_Block`(B_i, δ)
    $\delta \leftarrow \inf\left\{\left\{l(w) \;\middle|\; w \in \mathrm{V}(\mathcal{G}') : l(w) > \delta\right\} \cup \left\{\Delta > \delta \;\middle|\; \exists \mathrm{B}_i \in \mathcal{B} : \mathcal{R}(\mathrm{B}_i, \Delta) \neq \emptyset\right\}\right\}$

Die Prozedur kann entsprechend Bemerkung 1.4.2 und der zuvor bewiesenen Korrektheit früher terminieren, wenn die Abfrage $\nexists v \in \mathrm{T} : l(v) < \delta$ durch die Abfrage $\nexists v \in \mathrm{T} : l(v) \leq \delta$ ersetzt wird.

Nach Definition der Prozedur `Label_Intervall` impliziert das Labeln von Intervallen, dass jeder Aufruf von `Label_Intervall`$\left(l(v) + c_{-\pi}\left((v,w)\right), w, \mathrm{J}\right)$ innerhalb der Prozedur `Suche_Pfad` für ein Intervall $\mathrm{J} \in \mathcal{I}$ und eine Kante $(v,w) \in \mathrm{E}(\mathcal{G}'[\mathrm{J}])$ entfallen kann. Diese Aussage wird später benötigt, um eine schnelle Realisierung der Prozedur `Suche_Pfad` anzugeben, die darauf basiert, die Anzahl der Aufrufe von `Label_Intervall` zu reduzieren. Sie wird deshalb als Satz formuliert:

Satz 2.2.4:
Innerhalb der Prozedur `Suche_Pfad` *kann der Aufruf von*

$$\texttt{Label\_Intervall}\big(l(v) + c_{-\pi}\big((v,w)\big), w, \mathrm{J}\big)$$

für ein Intervall $\mathrm{J} \in \mathcal{I}$ *und eine Kante* $(v,w) \in \mathrm{E}(\mathcal{G}'[\mathrm{J}])$ *entfallen.*

Um die Laufzeiten eines Aufrufes der Prozeduren `Prozessiere_Registrierte` bzw. `Prozessiere_Block` in den nächsten Abschnitten abzuschätzen, werden an dieser Stelle die folgenden beiden Sätze formuliert. Die Labelfunktion l nach Terminierung der Prozedur $\texttt{Suche\_Pfad}(\mathrm{S},\mathrm{T})$ wird mit $\bar{l}$ bezeichnet. Außerdem bezeichnet δ_k den Wert δ in Iteration k der äußeren Schleife in ②.

Satz 2.2.5:
Betrachte einen Aufruf von $\texttt{Prozessiere\_Block}(\mathrm{B}_i, \delta_k)$. *Bezeichnet* $\hat{l}$ *die Labelfunktion* l *nach Terminierung dieses Aufrufes, dann gilt:*

$$\forall w \in \mathrm{B}_i : \big(\hat{l}(w) = \delta_k \;\Rightarrow\; \bar{l}(w) = \delta_k\big)$$

Beweis:
Der Beweis erfolgt durch Widerspruch.
Angenommen, es existiert ein Knoten $w \in \mathrm{B}_i$ mit $\hat{l}(w) = \delta_k$ und $\bar{l}(w) < \delta_k$. Dann müsste in Iteration $k' \geq k$ für einen Knoten $v \in \Gamma^-(w)$ und den zugehörigen Block $\mathrm{B}_j \in \mathcal{B}$ mit $v \in \mathrm{B}_j$ die Prozedur $\texttt{Label\_Intervall}(\delta_{k'} + c_{\mathcal{B}}(\mathrm{B}_j, \mathrm{B}_i), w, \mathrm{J})$ aufgerufen werden.
Für $j > i$ gilt $k' \geq k$. Es folgt:

$$\delta_{k'} + \underbrace{c_{\mathcal{B}}(\mathrm{B}_j, \mathrm{B}_i)}_{>0} > \delta_k \quad \lightning$$

Für $j \leq i$ gilt $k' > k$. Es folgt:

$$\delta_{k'} + c_{\mathcal{B}}(\mathrm{B}_j, \mathrm{B}_i) \geq \delta_{k'} > \delta_k \quad \lightning$$

□

Satz 2.2.6:
Nach Terminierung eines Aufrufes von $\texttt{Prozessiere\_Block}(\mathrm{B}_i, \delta)$ *gilt für jeden Knoten* $v \in \mathrm{B}_i$ *mit* $l(v) = \delta$*:*

$$\forall w \in \Gamma^+(v) \cap \mathrm{V}_{z(\mathrm{B}_i)} : \big|l(v) - l(w)\big| \leq c_{-\pi_{\max}}$$

Beweis:
Nach Definition von `Prozessiere_Block` gilt:

$$\forall w \in \Gamma^+(v) \cap \mathrm{V}_{z(\mathrm{B}_i)} : \; l(w) \leq l(v) + c_{-\pi}\big((v,w)\big)$$

Wegen $l(v) = \bar{l}(v)$ nach Satz 2.2.5 und der in Abschnitt 2.2.1 gezeigten Korrektheit der Prozedur `Suche_Pfad` gilt:

$$\forall w \in \Gamma^+(v) \cap \mathrm{V}_{z(\mathrm{B}_i)} : \; l(v) \leq l(w) + c_{-\pi}\big((w,v)\big)$$

□

2.3 Prozedur Label_Intervall

Es wird gezeigt, wie die Prozedur Label_Intervall schnell realisiert werden kann. Ein Aufruf von Label_Intervall(Δ, w, J) bewirkt diese Operationen:

$l(w) \leftarrow \min\{l(w), \Delta\}$
für alle $\bar{w} \in \Gamma^+(w) \cap \mathrm{J}$ **mit** $l(w) + c_{\text{-}\pi}\big((w, \bar{w})\big) < l(\bar{w})$ **tue**
  Label_Intervall$\big(l(w) + c_{\text{-}\pi}\big((w, \bar{w})\big), \bar{w}, \mathrm{J}\big)$

2.3.1 Äquivalente Formulierung

Es wird zunächst eine äquivalente Formulierung der oben genannten Operationen angegeben, die ohne Rekursivität auskommt.

Satz 2.3.1:
Betrachte einen Aufruf der Prozedur Label_Intervall(Δ, w, J) *für ein Intervall* $\mathrm{J} \in \mathcal{I}$ *mit* $\mathrm{J} = \{w_1, \ldots, w_n\}$, *sodass* $(w_k, w_{k+1}) \in \mathrm{E}(\mathcal{G}'[\mathrm{J}])$ *für alle* $k \in \{1, \ldots, n-1\}$ *und einen Knoten* $w \in \mathrm{J}$. *Angenommen, es gilt* $w_i = w$. *Dann werden die folgenden Operationen ausgeführt:*

$l(w_i) \leftarrow \min\{l(w_i), \Delta\}$
$j \leftarrow i$
solange $j > 1$ **und** $l(w_j) + c_{\text{-}\pi}\big((w_j, w_{j-1})\big) < l(w_{j-1})$ **tue**
  $l(w_{j-1}) \leftarrow l(w_j) + c_{\text{-}\pi}\big((w_j, w_{j-1})\big)$
  $j \leftarrow j - 1$
$j \leftarrow i$
solange $j < n$ **und** $l(w_j) + c_{\text{-}\pi}\big((w_j, w_{j+1})\big) < l(w_{j+1})$ **tue**
  $l(w_{j+1}) \leftarrow l(w_j) + c_{\text{-}\pi}\big((w_j, w_{j+1})\big)$
  $j \leftarrow j + 1$

Beweis:
Wird für eine Kante $(w, \bar{w}) \in \mathrm{E}(\mathcal{G}'[\mathrm{J}])$ die Operation

$$l(\bar{w}) \leftarrow l(w) + c_{\text{-}\pi}\big((w, \bar{w})\big)$$

ausgeführt, dann gilt anschließend, da die Funktion $c_{\text{-}\pi}$ nicht negativ ist:

$$l(\bar{w}) + c_{\text{-}\pi}\big((\bar{w}, w)\big) \;=\; l(w) + c_{\text{-}\pi}\big((w, \bar{w})\big) + c_{\text{-}\pi}\big((\bar{w}, w)\big) \;\geq\; l(w)$$

□

Definition 2.3.2:
Für ein Intervall $\mathrm{I} \in \mathcal{I}$ *wird die Funktion* $d_{\mathrm{I}}(v,w)$ *für zwei Knoten* $v, w \in \mathrm{I}$ *definiert. Mit* $\mathrm{I} = \{v_1, \ldots, v_n\}$, *sodass* $(v_k, v_{k+1}) \in \mathrm{E}(\mathcal{G}'[\mathrm{I}])$ *für alle* $k \in \{1, \ldots, n-1\}$ *und* $v_i = v$ *und* $v_j = w$ *ist* $d_{\mathrm{I}}(v,w)$ *definiert durch:*

$$d_{\mathrm{I}}(v,w) := \begin{cases} \sum_{k=i}^{j-1} c_{\text{-}\pi}\left((v_k, v_{k+1})\right) & \text{falls } j > i \\ 0 & \text{falls } j = i \\ \sum_{k=j}^{i-1} c_{\text{-}\pi}\left((v_k, v_{k+1})\right) & \text{falls } j < i \end{cases}$$

Für die von der Prozedur `Label_Intervall` ausgeführten Operationen ergibt sich mit dieser Definition und Satz 2.3.1 eine weitere Formulierung, die es schließlich ermöglicht, eine schnelle Realisierung der Prozedur `Label_Intervall` anzugeben.

Satz 2.3.3:
Ein Aufruf von `Label_Intervall`(Δ, w, J) *für ein Intervall* $\mathrm{J} \in \mathcal{I}$ *und einen Knoten* $w \in \mathrm{J}$ *ist äquivalent zum Ausführen der folgenden Operationen:*

für alle $\bar{w} \in \mathrm{J}$ **tue**
  $l(\bar{w}) \leftarrow \min\left\{l(\bar{w}), \Delta + d_{\mathrm{J}}(w, \bar{w})\right\}$

Beweis:
Es wird ein Aufruf von `Label_Intervall`(Δ, w, J) betrachtet.
Für $\bar{w} = w$ ist die Aussage klar.
Für $\bar{w} \neq w$ bezeichne die Funktion l zu Beginn mit l_0 und nach Terminierung mit l_1. Es wird folgendes angenommen:

$$\forall\, (v,w) \in \mathrm{E}(\mathcal{G}'[\mathrm{J}]) :\ l_0(w) \leq l_0(v) + c_{\text{-}\pi}\left((v,w)\right) \tag{2.3.1}$$

Sei $\mathrm{J} = \{w_1, \ldots, w_n\}$ entsprechend Satz 2.3.1 mit $w_i = w$. Dann gibt es einen Index $i_r \geq i$, sodass für alle $j \in \{i, \ldots, i_r\}$ mit $j > i$ die Operation

$$l(w_j) \leftarrow l(w_i) + \sum_{k=i}^{j-1} c_{\text{-}\pi}\left((w_k, w_{k+1})\right) \tag{2.3.2}$$

ausgeführt wird.
Dann gilt für $l_1(w_{i_r})$:

$$l_1(w_{i_r}) + c_{\text{-}\pi}\left((w_{i_r}, w_{i_r+1})\right) \geq l_0(w_{i_r+1}) = l_1(w_{i_r+1}) \tag{2.3.3}$$

Sei $\bar{w} = w_j$.
Falls $j > i_r$, gilt wegen $\Delta \geq l_1(w_i)$ und $l_0(w_j) = l_1(w_j)$:

$$\begin{aligned}
& \Delta \quad + d_J(w_i, w_j) \\
\geq \quad & l_1(w_i) + \sum_{k \in \{i,\ldots,i_r-1\}} c_{\text{-}\pi}\big((w_k, w_{k+1})\big) + \sum_{k \in \{i_r,\ldots,j-1\}} c_{\text{-}\pi}\big((w_k, w_{k+1})\big) \\
\underset{\text{(2.3.1) u. (2.3.3)}}{\geq} \quad & l_1(w_i) + \sum_{k \in \{i,\ldots,i_r-1\}} \big(l_1(w_{k+1}) - l_1(w_k)\big) + \sum_{k \in \{i_r,\ldots,j-1\}} \big(l_1(w_{k+1}) - l_1(w_k)\big) \\
= \quad & l_1(w_j) = l_0(w_j)
\end{aligned}$$

Deshalb modifiziert die Operation

$$l(\bar{w}) \leftarrow \min\big\{l(\bar{w}), \Delta + d_J(w, \bar{w})\big\}$$

den Wert $l(\bar{w})$ nicht.
Falls $i < j \leq i_r$, folgt $l_0(w_i) > \Delta$ mit Voraussetzung (2.3.1). In diesem Fall folgt nach Satz 2.3.1:

$$l_0(w_j) > l_1(w_i) + d_J(w_i, w_j) = \Delta + d_J(w_i, w_j)$$

Deshalb kann die Operation (2.3.2) durch

$$l(w_j) \leftarrow \min\big\{l(w_j), \Delta + d_J(w_i, w_j)\big\}$$

ersetzt werden.
Analog dazu gibt es einen Index $i_l \leq i$ und es gelten die analogen Aussagen für $j < i_l$ und $i_l \leq j < i$.
Unter Voraussetzung (2.3.1) sind die in Satz 2.3.3 angegebenen Operationen daher äquivalent zu einem Aufruf von `Label_Intervall`(Δ, w, J).

Es bleibt zu zeigen, dass die Annahme (2.3.1) gültig ist. Dazu definiere $M \subseteq J$ als die Menge der Knoten $v \in J$, mit $l_0(v) \neq l_1(v)$. Mit einer Kante $(v, w) \in E(\mathcal{G}'[J])$ gilt

$$l_1(w) \leq l_1(v) + c_{\text{-}\pi}\big((v, w)\big)$$

- für $v, w \in J \setminus M$ nach Annahme (2.3.1)
- für $v \in J \setminus M, w \in M$ nach Annahme (2.3.1), da $l_1(w) \leq l_0(w)$ und $l_1(v) = l_0(v)$
- für $v, w \in M$ nach Definition der Prozedur `Label_Intervall` und Satz 2.3.1
- für $v \in M, w \in J \setminus M$ nach Definition der Prozedur `Label_Intervall` und Satz 2.3.1.

Da nur die Prozedur `Label_Intervall` die Funktion l innerhalb von `Suche_Pfad` modifiziert, ist die Annahme (2.3.1) begründet. □

2.3.2 Funktion $\Delta_{\mathcal{L}(\mathrm{I})}$

Um die Operationen der Prozedur `Label_Intervall` schnell auszuführen, wird für jedes Intervall $\mathrm{I} \in \mathcal{I}$ eine Funktion $\Delta_{\mathcal{L}(\mathrm{I})}$ definiert. Die Prozedur `Label_Intervall`(Δ, w, J) wird so realisiert, dass sie nur die Funktion $\Delta_{\mathcal{L}(\mathrm{J})}$ modifiziert. Der Zusammenhang mit der Funktion l ist für jede Ebene $z \in \mathrm{Z}$ gegeben durch:

$$\forall \mathrm{I} \in \mathcal{I}(z), v \in \mathrm{I}: \; l(v) = \begin{cases} \Delta_{\mathcal{L}(\mathrm{I})}(v_1) & \text{falls } \mathrm{VR}(z) = \text{horizontal } [\leftrightarrows] \\ \Delta_{\mathcal{L}(\mathrm{I})}(v_2) & \text{falls } \mathrm{VR}(z) = \text{vertikal } [\updownarrow] \end{cases}$$

Im folgenden Teil dieses Abschnitts 2.3 wird o.B.d.A. eine Ebene $z \in \mathrm{Z}$ mit $\mathrm{VR}(z) =$ horizontal $[\leftrightarrows]$ betrachtet.

Definition 2.3.4:
Für ein Intervall $\mathrm{I} \in \mathcal{I}(z)$ *gilt:*

$$\mathcal{L}(\mathrm{I}) \subseteq \mathbb{Z}_{\geq 0} \times \mathrm{X}(\mathrm{I})$$

Mit dieser Menge ist die Funktion $\Delta_{\mathcal{L}(\mathrm{I})}$ *definiert durch:*

$$\begin{aligned} \Delta_{\mathcal{L}(\mathrm{I})} : \left[\min \mathrm{X}(\mathrm{I}), \max \mathrm{X}(\mathrm{I})\right] &\to \mathbb{R} \\ x &\mapsto \inf\left\{\Delta' + \beta_{\mathrm{I}}(x', x) \;\middle|\; (\Delta', x') \in \mathcal{L}(\mathrm{I})\right\} \end{aligned}$$

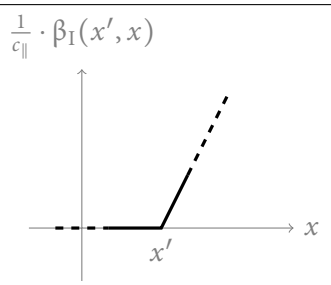
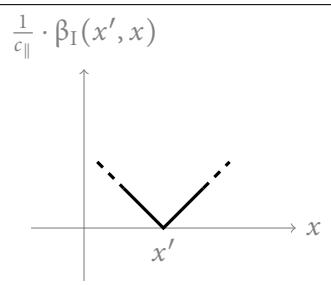
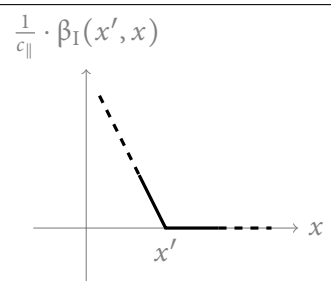
Mit $x', x \in \mathbb{R}$ *ist hierbei der Wert für* $\beta_{\mathrm{I}}(x', x) \in \mathbb{R}$ *gegeben durch:*

$$\beta_{\mathrm{I}}(x', x) := \alpha_{\mathrm{I}} \cdot (x - x') + c_{\parallel} \cdot |x - x'|$$

Die Funktion β_{I} lässt sich auch folgendermaßen angeben:

$$\beta_{\mathrm{I}}(x', x) := \begin{cases} 2c_{\parallel} \cdot |x - x'| & \text{falls } (\alpha_{\mathrm{I}} = c_{\parallel} \wedge x > x') \text{ oder } (\alpha_{\mathrm{I}} = -c_{\parallel} \wedge x < x') \\ c_{\parallel} \cdot |x - x'| & \text{falls } \alpha_{\mathrm{I}} = 0 \\ 0 & \text{sonst} \end{cases}$$

In der folgenden Tabelle ist die Funktion $\beta_{\mathrm{I}}(x', \cdot)$ für jeden Wert von α_{I} visualisiert:

		
$\alpha_{\mathrm{I}} = c_{\parallel}$	$\alpha_{\mathrm{I}} = 0$	$\alpha_{\mathrm{I}} = -c_{\parallel}$

Als Beispiel einer Funktion $\Delta_{\mathcal{L}(I)}$ für ein Intervall $I \in \mathcal{I}(z)$ mit $\alpha_I = c_\parallel$ sei

$$\mathcal{L}(I) = \{(\Delta_1, x_1), (\Delta_2, x_2), (\Delta_3, x_3), (\Delta_4, x_4)\},$$

sodass sich die in der folgenden Grafik schwarz eingezeichnete Funktion $\Delta_{\mathcal{L}(I)}$ ergibt:

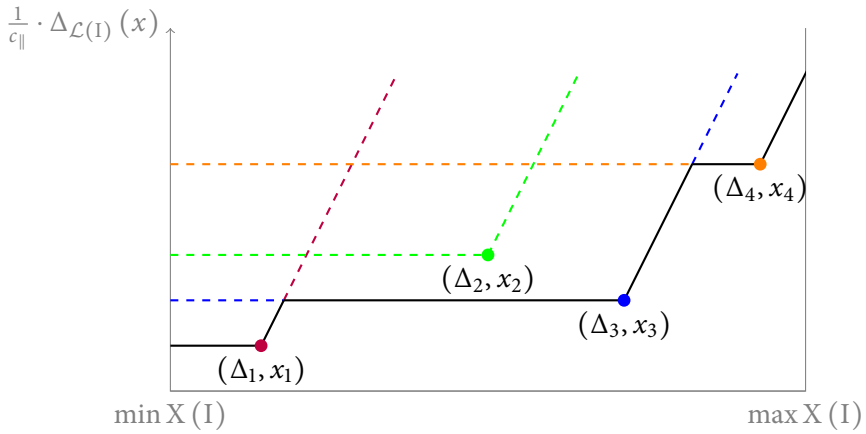


Abbildung 2.3.1: $\Delta_{\mathcal{L}(I)}$ (schwarz) für ein Intervall $I \in \mathcal{I}(z)$ mit $\alpha_I = c_\parallel$ und $\left|\mathcal{L}(I)\right| = 4$

Bemerkung 2.3.5:
Für ein Intervall $I \in \mathcal{I}$ *mit* $\alpha_I \neq 0$ *ist die Funktion* $\Delta_{\mathcal{L}(I)}$ *monoton.*

Die Definition der Funktion $\Delta_{\mathcal{L}(I)}$ für ein Intervall $I \in \mathcal{I}$ begründet sich durch den nachfolgenden Satz.

Satz 2.3.6:
Für ein Intervall $I \in \mathcal{I}(z)$ *mit* $y := y(I)$ *und* $x, x' \in X(I)$ *gilt:*

$$\beta_I(x', x) = d_I\big((x', y, z), (x, y, z)\big)$$

Beweis:
Mit $(x', y, z) = v$ und $(x, y, z) = w$ sei P der v-w-Pfad in $\mathcal{G}'[I]$. Für diesen Pfad P gilt:

$$\begin{aligned}
c_{-\pi}(P) &= \sum_{e=(v',w')\in E(P)} c_{-\pi}(e) \\
&= \sum_{e=(v',w')\in E(P)} \Big(c(e) - \big(\pi(v') - \pi(w')\big)\Big) \\
&= \sum_{e=(v',w')\in E(P)} c(e) - \big(\pi(v) - \pi(w)\big) \\
&= \sum_{(v',w')\in E(P)} c_\parallel \cdot |v_1' - w_1'| - \alpha_I \cdot (v_1 - w_1) = c_\parallel \cdot \underbrace{|v_1 - w_1|}_{|x'-x|} - \alpha_I \cdot \underbrace{(v_1 - w_1)}_{(x'-x)}
\end{aligned}$$

□

Satz 2.3.7:
Für ein Intervall $\mathrm{I} \in \mathcal{I}(z)$ *mit* $y := y(\mathrm{I})$ *gilt:*

$$\begin{aligned}\Delta_{\mathcal{L}(\mathrm{I})} : \left[\min \mathrm{X}(\mathrm{I}), \max \mathrm{X}(\mathrm{I})\right] &\to \mathbb{R} \\ x &\mapsto \inf\left\{\Delta' + d_{\mathrm{I}}\left((x', y, z), (x, y, z)\right) \,\middle|\, (\Delta', x') \in \mathcal{L}(\mathrm{I})\right\}\end{aligned}$$

Beweis: Folgt aus Satz 2.3.6. □

Bemerkung 2.3.8:
Nach Satz 2.3.3 und Satz 2.3.7 kann die Prozedur `Label_Intervall`(Δ, w, J) *durch die folgende Operation realisiert werden:*

$$\mathcal{L}(\mathrm{J}) \leftarrow \mathcal{L}(\mathrm{J}) \cup \left\{(\Delta, w_1)\right\}$$

Für ein Intervall $\mathrm{I} \in \mathcal{I}$ kann dies jedoch implizieren, dass die Menge $\mathcal{L}(\mathrm{I})$ Elemente enthält, die zur Angabe von $\Delta_{\mathcal{L}(\mathrm{I})}$ nicht notwendig sind. Ein Beispiel dafür ist das Paar (Δ_2, x_2) in Abbildung 2.3.2 auf Seite 37. Es gilt der Satz:

Satz 2.3.9:
Für ein Intervall $\mathrm{I} \in \mathcal{I}(z)$ *und ein Paar* (Δ', x') *mit* $x' \in \mathrm{X}(\mathrm{I})$ *gilt:*

$$\Delta' \geq \Delta_{\mathcal{L}(\mathrm{I})}(x') \quad \Rightarrow \quad \forall x \in \mathrm{X}(\mathrm{I}) : \ \Delta_{\mathcal{L}(\mathrm{I})}(x) \leq \Delta' + \beta_{\mathrm{I}}(x', x)$$

Beweis:
Zunächst gilt:

$$\forall x \in \mathrm{X}(\mathrm{I}) : \ \Delta_{\mathcal{L}(\mathrm{I})}(x) \leq \Delta_{\mathcal{L}(\mathrm{I})}(x') + \beta_{\mathrm{I}}(x', x)$$

- Klar für $\Delta_{\mathcal{L}(\mathrm{I})}(x') = \infty$.
- Für $\Delta_{\mathcal{L}(\mathrm{I})}(x') < \infty$ sei $(\Delta'', x'') \in \mathcal{L}(\mathrm{I})$, mit $\Delta_{\mathcal{L}(\mathrm{I})}(x') = \Delta'' + \beta_{\mathrm{I}}(x'', x')$.
  Dann gilt für alle $x \in \mathrm{X}(\mathrm{I})$:

$$\begin{aligned}\Delta_{\mathcal{L}(\mathrm{I})}(x') + \beta_{\mathrm{I}}(x', x) &= \Delta'' + \beta_{\mathrm{I}}(x'', x') + \beta_{\mathrm{I}}(x', x) \\ &\geq \Delta'' + \beta_{\mathrm{I}}(x'', x) \geq \Delta_{\mathcal{L}(\mathrm{I})}(x)\end{aligned}$$

Damit folgt für $x \in \mathrm{X}(\mathrm{I})$:

$$\Delta' \geq \Delta_{\mathcal{L}(\mathrm{I})}(x') \quad \Rightarrow \quad \Delta' + \beta_{\mathrm{I}}(x', x) \geq \Delta_{\mathcal{L}(\mathrm{I})}(x') + \beta_{\mathrm{I}}(x', x) \geq \Delta_{\mathrm{I}}(x)$$

□

Definition 2.3.10:
Für ein Intervall $\mathrm{I} \in \mathcal{I}(z)$ *heißt das Paar* (Δ', x') *mit* $x' \in \mathrm{X}(\mathrm{I})$ *„redundant bzgl.* $\mathcal{L}(\mathrm{I})$*“, wenn gilt:*

$$\Delta' \geq \Delta_{\mathcal{L}(\mathrm{I})}(x')$$

2.3.3 Realisierung von `Label_Intervall`

Um die Prozedur `Label_Intervall`(Δ, w, J) zu realisieren, muss entsprechend Bemerkung 2.3.8 und Satz 2.3.9 das Paar (Δ, w_1) zu $\mathcal{L}(\mathrm{J})$ hinzugefügt werden, falls dieses nicht redundant ist bzgl. $\mathcal{L}(\mathrm{J})$. Anschließend können, wiederum entsprechend Satz 2.3.9, sämtliche Paare $(\Delta', x') \in \mathcal{L}(\mathrm{J})$ aus $\mathcal{L}(\mathrm{J})$ entfernt werden, für die gilt:

$$(\Delta', x') \text{ ist redundant bzgl. } \mathcal{L}(\mathrm{I}) \setminus \{(\Delta', x')\}$$

Deshalb wird die Prozedur `Label_Intervall` wie folgt angegeben:

Prozedur `Label_Intervall`(Δ, w, J)

---

① ***Redundanz überprüfen***
  wenn (Δ, w_1) **redundant bzgl.** $\mathcal{L}(\mathrm{J})$ **dann**
    Stop

② ***Redundante Paare löschen***
  $\mathcal{L}(\mathrm{J}) \leftarrow \mathcal{L}(\mathrm{J}) \cup \{(\Delta, w_1)\}$
  für alle $(\Delta', x') \in \mathcal{L}(\mathrm{J}) : (\Delta', x')$ **redundant bzgl.** $\mathcal{L}(\mathrm{J}) \setminus \{(\Delta', x')\}$ **tue**
    $\mathcal{L}(\mathrm{J}) \leftarrow \mathcal{L}(\mathrm{J}) \setminus \{(\Delta', x')\}$

Bemerkung 2.3.11:
Wird für jedes Intervall $\mathrm{J} \in \mathcal{I}(z)$ *die Menge* $\mathcal{L}(\mathrm{J})$ *nur von der Prozedur* `Label_Intervall` *modifiziert, gilt stets vor Beginn und nach Terminierung von* `Label_Intervall`(Δ, w, J)*:*

$$\forall (\Delta', x') \in \mathcal{L}(\mathrm{J}) : \ (\Delta', x') \textit{ ist nicht redundant bzgl. } \mathcal{L}(\mathrm{J}) \setminus \{(\Delta', x')\}$$

Als Beispiel für die Operationen eines Aufrufes von `Label_Intervall`$(\Delta_6, w, \mathrm{J})$ für ein Intervall J sei $\alpha_{\mathrm{J}} = c_{\|}$, $\mathrm{J} = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$, $w = v_6$ und

$$\mathcal{L}(\mathrm{J}) = \{(\Delta_2, x_2), (\Delta_4, x_4), (\Delta_5, x_5), (\Delta_7, x_7)\},$$

sodass die Funktion $\Delta_{\mathcal{L}(\mathrm{J})}$ die in Abbildung 2.3.2 auf der nächsten Seite dargestellte Gestalt hat. Mit dem zusätzlichen Paar (Δ_6, x_6) sind die Paare (Δ_4, x_4) und (Δ_5, x_5) redundant bzgl.

$$\mathcal{L}(\mathrm{J}) \cup \{(\Delta_6, x_6)\} \setminus \{(\Delta_4, x_4), (\Delta_5, x_5)\},$$

wie in Abbildung 2.3.3 zu sehen. Nach dem Aufruf der Prozedur `Label_Intervall`(Δ, w, J) gilt deshalb:

$$\mathcal{L}(\mathrm{J}) = \{(\Delta_2, x_2), (\Delta_6, x_6), (\Delta_7, x_7)\}$$

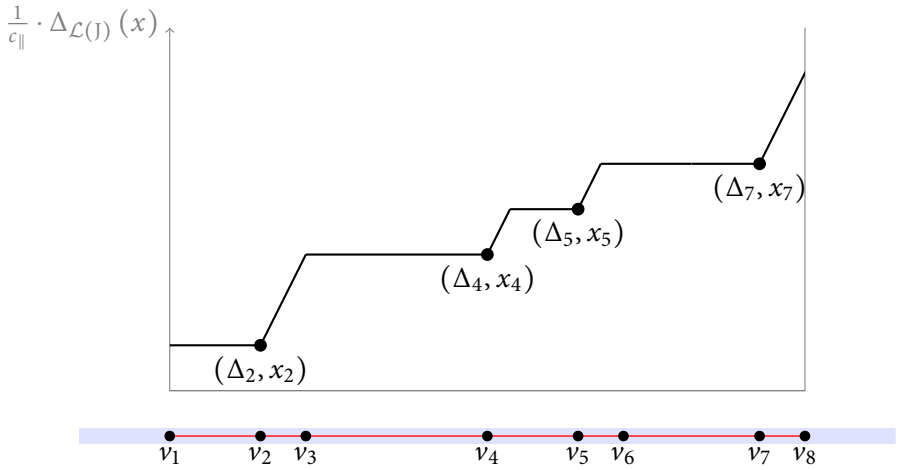


Abbildung 2.3.2: Labelfunktion $\Delta_{\mathcal{L}(J)}$ für $\left|\mathcal{L}(J)\right| = 4$ und $\alpha_J = c_\parallel$

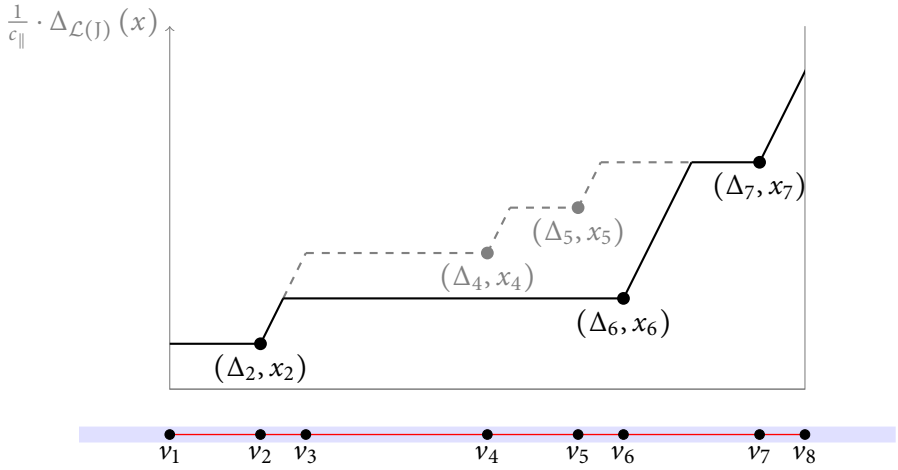


Abbildung 2.3.3: (Δ_4, x_4), (Δ_5, x_5) sind redundant bzgl. $\{(\Delta_2, x_2), (\Delta_6, x_6), (\Delta_7, x_7)\}$

2.3.4 Laufzeit von Label_Intervall(Δ, w, J)

Satz 2.3.12:
Ein Aufruf von Label_Intervall(Δ, w, J) *erfolgt so, dass zu Beginn gilt:*

$$\left|\mathcal{L}(\mathrm{J})\right| \leq |\mathcal{I}|$$

Beweis: Folgt aus Satz 2.7.1 in Abschnitt 2.7 und Bemerkung 2.3.11. □

Für jedes Intervall $\mathrm{I} \in \mathcal{I}$ ist die Menge $\mathcal{L}(\mathrm{I})$ definiert, um die Prozedur Label_Intervall schnell zu realisieren. Deshalb werden diese Mengen nur von dieser Prozedur modifiziert. Daher gilt nach Bemerkung 2.3.11 zu Beginn des Aufrufes von Label_Intervall(Δ, w, J):

$$\forall (\Delta', x') \in \mathcal{L}(\mathrm{J}): \ (\Delta', x') \text{ ist nicht redundant bzgl. } \mathcal{L}(\mathrm{J}) \setminus \left\{(\Delta', x')\right\} \tag{2.3.4}$$

Daraus folgt:

$$\forall (\Delta', x'), (\Delta'', x'') \in \mathcal{L}(\mathrm{J}): \quad (\Delta', x') \neq (\Delta'', x'') \ \Rightarrow \ x' \neq x''$$

Deshalb wird angenommen, dass zu Beginn des Aufrufes von Label_Intervall(Δ, w, J) gilt:

$$\mathcal{L}(\mathrm{J}) = \left\{(\Delta_1, x_1), \ldots, (\Delta_n, x_n)\right\} \quad \text{mit} \quad x_1 < \ldots < x_n$$

Dann ist die Teilmenge von $\mathcal{L}(\mathrm{J})$ gesucht, die ausreicht, um $\Delta_{\mathcal{L}(\mathrm{J})}(x)$ zu bestimmen und somit Redundanz zu verifizieren. Um diese anzugeben, wird der folgende Satz benötigt:

Satz 2.3.13:
Mit $f_i(x) := \Delta_i + \beta_{\mathrm{J}}(x_i, x)$ *gilt für alle* $i, j \in \{1, \ldots, n\}$*:*

$$\begin{aligned} x_i < x_j \leq x \ &\Rightarrow \ f_i(x) > f_j(x) \\ x \leq x_i < x_j \ &\Rightarrow \ f_i(x) < f_j(x) \end{aligned}$$

Beweis:
Mit Voraussetzung (2.3.4) gilt:

$$\begin{aligned} \left(x', x'' \geq \max\left\{x_i, x_j\right\}\right) \quad \vee \quad \left(x', x'' \leq \min\left\{x_i, x_j\right\}\right) \\ \Rightarrow \ f_i(x') - f_i(x'') = f_j(x') - f_j(x'') \end{aligned} \tag{2.3.5}$$

Damit folgt:

- $x_i < x_j \leq x \quad \Rightarrow \quad f_i(x_j) > f_j(x_j) = \Delta_j \quad \Rightarrow \quad f_i(x) > f_j(x)$
- $x \leq x_i < x_j \quad \Rightarrow \quad \Delta_i = f_i(x_i) < f_j(x_i) \quad \Rightarrow \quad f_i(x) < f_j(x)$

Die erste Implikation gilt nach (2.3.4), die zweite ergibt sich durch Addition von (2.3.5) mit den Ungleichungen, wobei $x' = x_j$ bzw. $x' = x_i$ gewählt wird. □

Definition 2.3.14:
Die Menge $\mathrm{N_J}(x) \subseteq \mathcal{L}(\mathrm{J})$*, die als „Nachbarn von x in* $\mathcal{L}(\mathrm{J})$*“ zu interpretieren ist, ist definiert durch:*

$$\mathrm{N_J}(x) := \begin{cases} \{(\Delta_1, x_1)\} & \text{falls } x < x_1 \\ \{(\Delta_n, x_n)\} & \text{falls } x > x_n \\ \{(\Delta_i, x_i)\} & \text{falls } x = x_i \\ \{(\Delta_i, x_i), (\Delta_{i+1}, x_{i+1})\} & \text{falls } x_i < x < x_{i+1} \\ \emptyset & \text{falls } \mathcal{L}(\mathrm{J}) = \emptyset \end{cases}$$

Mit den Elementen der Menge $\mathrm{N_J}(x)$ kann der Wert $\Delta_{\mathcal{L}(\mathrm{J})}(x)$ angegeben werden, wie der nachfolgende Satz beweist:

Satz 2.3.15:
Es gilt:

$$\Delta_{\mathcal{L}(\mathrm{J})}(x) = \min\left\{\Delta' + \beta_{\mathrm{J}}(x', x) \;\middle|\; (\Delta', x') \in \mathrm{N_J}(x)\right\}$$

Beweis:
Definiere:

$$\mathrm{L} := \left\{i \in \{1, \ldots, n\} \;\middle|\; x_i \leq x\right\} \qquad \mathrm{R} := \left\{i \in \{1, \ldots, n\} \;\middle|\; x_i > x\right\}$$

Mit $f_i(x) := \Delta_i + \beta_{\mathrm{J}}(x_i, x)$ gilt nach Satz 2.3.13:

$$\begin{aligned} \forall\, i, j \in \mathrm{L}: &\quad \left(x_i < x_j \;\Rightarrow\; f_i(x) > f_j(x)\right) \\ \forall\, i, j \in \mathrm{R}: &\quad \left(x_i < x_j \;\Rightarrow\; f_i(x) < f_j(x)\right) \end{aligned}$$

□

Die Menge $\mathcal{L}(\mathrm{J})$ soll durch einen geeigneten Suchbaum realisiert sein. Für jedes Element $(\Delta_i, x_i) \in \mathcal{L}(\mathrm{J})$ kann z. B. x_i als aufsteigend sortierter Schlüssel gewählt werden. Dann kann für $x \in \mathrm{X}(\mathrm{J})$ die Menge $\mathrm{N_J}(x)$ der „Nachbarn von x in $\mathcal{L}(\mathrm{J})$“ wegen $|\mathcal{L}(\mathrm{J})| \leq |\mathcal{I}|$ nach Satz 2.3.12 bestimmt werden in Zeit:

$$\mathcal{O}\left(\log |\mathcal{I}|\right)$$

In konstanter Zeit kann anschließend mit den Elementen aus $\mathrm{N_J}(x)$ nach Satz 2.3.15 verifiziert werden, ob (Δ, x) redundant ist bzgl. $\mathcal{L}(\mathrm{J})$. Falls nicht, also $\Delta < \Delta_{\mathcal{L}(\mathrm{J})}(x)$, wird die Operation

$$\mathcal{L}(\mathrm{J}) \leftarrow \mathcal{L}(\mathrm{J}) \cup \left\{(\Delta, x)\right\}$$

ausgeführt, die dem Einsortieren von (Δ, x) in $\mathcal{L}(\mathrm{J})$ entspricht. Diese Operation ist in Zeit $\mathcal{O}(\log |\mathcal{I}|)$ realisierbar.

Sei anschließend

$$\mathcal{L}(\mathrm{J}) = \left\{(\Delta_1, x_1), \ldots, (\Delta_{s-1}, x_{s-1}), (\Delta, x), (\Delta_{s+1}, x_{s+1}), \ldots, (\Delta_m, x_m)\right\}$$

mit $x_1 < \ldots < x_{s-1} \leq x < x_{s+1} < x_{m+1}$ und $(\Delta_s, x_s) := (\Delta, x)$. Dann existieren

$$\mathrm{L} \subseteq \{1, \ldots, s-1\} \quad \text{und} \quad \mathrm{R} \subseteq \{s+1, \ldots, m+1\},$$

sodass (Δ_i, x_i) redundant ist bzgl. $\mathcal{L}(\mathrm{J}) \setminus \{(\Delta_i, x_i)\}$, also $\Delta_i \geq \Delta_{\mathcal{L}(\mathrm{J})}(x_i)$ für alle $i \in \mathrm{L} \cup \mathrm{R}$. Diese Mengen L und R sind mit

$$f_i(x) := \Delta_i + \beta_{\mathrm{J}}(x_i, x)$$

bestimmt durch:

$$\mathrm{L} = \{i \in \{1, \ldots, s-1\} \mid \Delta_i = f_i(x_i) \geq f_s(x_i)\}$$
$$\mathrm{R} = \{i \in \{s+1, \ldots, m\} \mid \Delta_i = f_i(x_i) \geq f_s(x_i)\}$$

Denn wird für jeden Index $i \in \mathrm{L} \cup \mathrm{R}$ die Operation

$$\mathcal{L}(\mathrm{J}) \leftarrow \mathcal{L}(\mathrm{J}) \setminus \{(\Delta_i, x_i)\}$$

ausgeführt, gilt anschließend mit

$$\mathcal{L}(\mathrm{J}) = \{(\Delta_1, x_1), \ldots, (\Delta_k, x_k)\} \quad \text{und} \quad x_1 < \ldots < x_k.$$

für alle $i, j \in \{1, \ldots, k\}$ mit $i \neq j$:

$$\Delta_i = f_i(x_i) < f_j(x_i)$$

Das ist klar für den Index j mit $x_j = s$ und gilt sonst nach Voraussetzung (2.3.4). Weiterhin gilt:

$$\mathrm{L} \neq \emptyset \quad \Rightarrow \quad \exists l \in \{1, \ldots, s-1\} : \ \mathrm{L} = \{l, l+1, \ldots, s-1\}$$
$$\mathrm{R} \neq \emptyset \quad \Rightarrow \quad \exists r \in \{s+1, \ldots, m\} : \mathrm{R} = \{s+1, s+2, \ldots, r\}$$

Denn für $j \in \{1, \ldots, s-1\} \setminus \mathrm{L}$ gilt:

$$f_j(x_j) < f_s(x_j)$$

Damit folgt für alle $i \in \{1, \ldots, s-1\} \setminus \mathrm{L}$ mit $i < j$ wegen $x_i < x_j < x_s$:

$$\Delta_i = \underbrace{f_i(x_i) < \overbrace{f_j(x_i)}^{\text{nach Satz 2.3.13}}}_{\text{nach Vor. (2.3.4)}} < f_s(x_i)$$

Eine analoge Begründung ergibt sich für R. Die Laufzeit um die Mengen L und R zu bestimmen, ist demnach beschränkt durch

$$\mathcal{O}\left(\log|\mathcal{I}| + |\mathrm{L} \cup \mathrm{R}| \cdot \log|\mathcal{I}|\right)$$

Wird die Menge $\mathcal{L}(\mathrm{I})$ zudem als geordnete Liste dargestellt, so ergibt sich folgende Laufzeit um L und R zu bestimmen:

$$\mathcal{O}\left(\log|\mathcal{I}| + |\mathrm{L} \cup \mathrm{R}|\right)$$

Das Ausführen der Operation

$$\mathcal{L}(\mathrm{J}) \leftarrow \mathcal{L}(\mathrm{J}) \setminus \left\{(\Delta_i, x_i)\right\}$$

für jeden Index $i \in \mathrm{L} \cup \mathrm{R}$ reduziert die Anzahl der Elemente, die in dem Suchbaum $\mathcal{L}(\mathrm{J})$ gespeichert sind und ist in $\mathcal{O}(\log|\mathcal{I}|)$ realisierbar.

Zusammengefasst ergibt sich für einen Aufruf von `Label_Intervall` folgende Laufzeitschranke:

$$\mathcal{O}\left(\log|\mathcal{I}|\right) + \underbrace{|\mathrm{L} \cup \mathrm{R}| \cdot \mathcal{O}\left(\log|\mathcal{I}|\right)}_{\text{Entfernen redundanter Paare}}$$

Amortisiert entfällt höchstens folgende Laufzeit auf ein Paar $(\Delta, x) \in \mathcal{L}(\mathrm{J})$:

- $\mathcal{O}\left(\log|\mathcal{I}|\right)$ um (Δ, x) zur Menge $\mathcal{L}(\mathrm{J})$ hinzuzufügen bzw. Redundanz festzustellen.
- $\mathcal{O}\left(\log|\mathcal{I}|\right)$ um (Δ, x) zu einem späteren Zeitpunkt aus $\mathcal{L}(\mathrm{J})$ zu entfernen.

Mit den obigen Ausführungen ist dieser Satz bewiesen:

Satz 2.3.16:
Die Prozedur `Label_Intervall`(Δ, w, J) *ist in folgender amortisierter Zeit realisierbar:*

$$\mathcal{O}\left(\log|\mathcal{I}|\right)$$

2.4 Prozedur `Label_Nachbarn`

Um die Prozedur `Prozessiere_Block` schnell realisieren zu können, wird die in diesem Abschnitt vorgestellte Prozedur `Label_Nachbarn`$(\delta, \mathrm{I}, \Pi)$ genutzt. Diese Prozedur führt für ein Intervall $\mathrm{I} \in \mathcal{I}$ mit $z := z(\mathrm{I})$, eine Menge $\Pi \subseteq \mathbb{R}$ mit $\Pi = [a, b] : a, b \in \mathrm{X}(\mathrm{I})$, einen Wert $\delta \in \mathbb{Z}_{\geq 0}$ und

$$\mathrm{I}_\Pi := \begin{cases} \{v \in \mathrm{I} \mid v_1 \in \Pi\} & \text{falls } \mathrm{VR}(z) = \text{ horizontal } [\leftrightarrows] \\ \{v \in \mathrm{I} \mid v_2 \in \Pi\} & \text{falls } \mathrm{VR}(z) = \text{ vertikal } [\downuparrows] \end{cases}$$

die folgenden Operationen aus:

für alle $v \in \mathrm{I}_\Pi$ **tue**
  für alle $w \in \Gamma^+(v) \cap \mathrm{V}_z$ **tue**
    Sei $\mathrm{J} \in \mathcal{I}$ das Intervall mit $w \in \mathrm{J}$.
    `Label_Intervall`$\left(\delta + c_{\text{-}\pi}\left((v, w)\right), w, \mathrm{J}\right)$

In diesem Abschnitt wird ein Aufruf von `Label_Nachbarn`$(\delta, \mathrm{I}, \Pi)$ betrachtet und für die Ebene $z := z(\mathrm{I})$ gilt o.B.d.A. $\mathrm{VR}(z) =$ horizontal $[\leftrightarrows]$.

2.4.1 Reduktion von Labeloperationen

Nach Satz 2.2.4 können die Aufrufe von `Label_Intervall`$\left(\delta + c_{\text{-}\pi}\left((v, w)\right), w, \mathrm{J}\right)$ innerhalb der Prozedur `Label_Nachbarn` für die Kanten

$$(v, w) \in \delta^+(\mathrm{I}_\Pi) \cap \mathrm{E}(\mathcal{G}'[\mathrm{I}])$$

entfallen. Daher kann die Prozedur `Label_Nachbarn`$(\delta, \mathrm{I}, \Pi)$ mit der folgenden Definition durch die nachfolgenden Operationen realisiert werden:

Definition 2.4.1:
Definiere die Menge $\mathcal{N}(\mathrm{I}, \Pi)$ *der benachbarten Intervalle von* I *bzgl.* Π *durch:*

$$\mathcal{N}(\mathrm{I}, \Pi) := \left\{\mathrm{J} \in \mathcal{I}(z) \setminus \{\mathrm{I}\} \;\middle|\; \mathrm{J} \cap \Gamma^+(\mathrm{I}_\Pi) \neq \emptyset\right\}$$

für alle $\mathrm{J} \in \mathcal{N}(\mathrm{I}, \Pi)$ **tue**
  für alle $(v, w) \in \mathrm{E}(\mathrm{I}_\Pi : \mathrm{J})$ **tue**
    `Label_Intervall`$\left(\delta + c_{\text{-}\pi}\left((v, w)\right), w, \mathrm{J}\right)$

Um die Anzahl der Aufrufe von `Label_Intervall` zu reduzieren, ist für ein benachbartes Intervall $\mathrm{J} \in \mathcal{N}(\mathrm{I}, \Pi)$ eine Kante $(v', w') \in \mathrm{E}(\mathrm{I}_\Pi : \mathrm{J})$ mit der folgenden Eigenschaft gesucht:

$$\forall\, (v, w) \in \mathrm{E}(\mathrm{I}_\Pi : \mathrm{J}) : \; c_{\text{-}\pi}\left((v, w)\right) = c_{\text{-}\pi}\left((v', w')\right) + d_{\mathrm{I}}\left(w', w\right)$$

Dann gilt für alle Kanten $(v, w) \in \mathrm{E}\left(\mathrm{I}_\Pi : \mathrm{J}\right)$ und alle Knoten $\bar{w} \in \mathrm{J}$:

$$\begin{aligned} & \delta + c_{\text{-}\pi}\left((v, w)\right) \quad + d_{\mathrm{I}}\left(w, \bar{w}\right) \\ = \; & \delta + c_{\text{-}\pi}\left((v', w')\right) + d_{\mathrm{I}}\left(w', w\right) + d_{\mathrm{I}}\left(w, \bar{w}\right) \\ \geq \; & \delta + c_{\text{-}\pi}\left((v', w')\right) + d_{\mathrm{I}}\left(w', \bar{w}\right) \end{aligned}$$

Nach Satz 2.3.3 kann dann das Ausführen der Operationen

für alle $(v, w) \in \mathrm{E}\left(\mathrm{I}_\Pi : \mathrm{J}\right)$ **tue**
  `Label_Intervall`$\left(\delta + c_{\text{-}\pi}\left((v, w)\right), w, \mathrm{J}\right)$

ersetzt werden durch den Aufruf von

`Label_Intervall`$\left(\delta + c_{\text{-}\pi}\left((v', w')\right), w', \mathrm{J}\right)$.

Für alle Kanten $(v, w) \in \mathrm{E}(\mathrm{I}_\Pi : \mathrm{J})$ soll also gelten:

$$\begin{aligned} & d_{\mathrm{I}}\left(w', w\right) = c_{\text{-}\pi}\left((v, w)\right) - c_{\text{-}\pi}\left((v', w')\right) \\ \Leftrightarrow \; & d_{\mathrm{I}}\left(w', w\right) = c\left((v, w)\right) + \pi\left(w\right) - \pi\left(v\right) \\ & \qquad\qquad - \left(c\left((v', w')\right) + \pi(w') - \pi(v')\right) \end{aligned} \tag{2.4.1}$$

Mit der Menge

$$\mathrm{J}_\Pi := \mathrm{J} \cap \Gamma^+\left(\mathrm{I}_\Pi\right)$$

ist leicht zu sehen, dass eine solche Kante (v', w') nur gesucht ist, falls $|\mathrm{J}_\Pi| \geq 2$. In diesem Fall muss $\mathrm{J}_\Pi \subseteq \mathrm{N}_\uparrow$ oder $\mathrm{J}_\Pi \subseteq \mathrm{N}_\downarrow$ gelten, wobei $\mathrm{N}_\uparrow$ und $\mathrm{N}_\downarrow$ als die potenziellen Nachbarknoten nördlich und südlich von I definiert sind:

$$\begin{aligned} \mathrm{N}_\uparrow &:= \left\{ \min \Pi, \ldots, \max \Pi \right\} \times \left\{ \mathcal{Y}_z^{-1}\left(\mathcal{Y}_z\left(y\left(\mathrm{I}\right)\right) + 1\right) \right\} \times \left\{ z\left(\mathrm{I}\right) \right\} \\ \mathrm{N}_\downarrow &:= \left\{ \min \Pi, \ldots, \max \Pi \right\} \times \left\{ \mathcal{Y}_z^{-1}\left(\mathcal{Y}_z\left(y\left(\mathrm{I}\right)\right) - 1\right) \right\} \times \left\{ z\left(\mathrm{I}\right) \right\} \end{aligned}$$

In diesem Fall enthält $\mathrm{E}(\mathrm{I}_\Pi : \mathrm{J})$ also nur Kanten entgegen der Vorzugsrichtung. Diese haben nach Definition der Kantenkostenfunktion c in Abschnitt 1.1.9 alle die gleichen Kantenkosten, sodass also in Fortsetzung von (2.4.1) eine Kante $(v', w') \in \mathrm{E}(\mathrm{I}_\Pi : \mathrm{J})$ gesucht ist, sodass für alle Kanten $(v, w) \in \mathrm{E}(\mathrm{I}_\Pi : \mathrm{J})$ gilt:

$$\begin{aligned} & d_{\mathrm{I}}\left(w', w\right) = \pi\left(w\right) - \pi(v) - \left(\pi(w') - \pi(v')\right) \\ \Leftrightarrow \; & d_{\mathrm{I}}\left(w', w\right) = \pi\left(w\right) - \pi(w') - \left(\pi(v) - \pi(v')\right) \end{aligned}$$

Mit Satz 2.3.6 ist dies äquivalent zu:

$$\forall (v, w) \in \mathrm{E}(\mathrm{I}_\Pi : \mathrm{J}) : \; \beta_{\mathrm{J}}(w_1', w_1) = \alpha_{\mathrm{J}}(w_1 - w_1') - \alpha_{\mathrm{I}}(v_1 - v_1')$$

Wegen $v_1 = w_1$ und $v_1' = w_1'$ ist mit

$$X_J := X\left(I_\Pi \cap \Gamma^-\left(J_\Pi\right)\right)$$

ein Wert $x' \in X_J$ gesucht, für den gilt:

$$\forall x \in X_J : \ \beta_J(x', x) = (\alpha_J - \alpha_I) \cdot (x - x') \tag{2.4.2}$$

Mit Definition 2.3.4 ist dies äquivalent zu:

$$\begin{aligned} &\forall x \in X_J : \ \alpha_J \cdot (x - x') + c_\parallel \cdot |x - x'| = (\alpha_J - \alpha_I) \cdot (x - x') \\ \Leftrightarrow\ &\forall x \in X_J : \ \alpha_I \cdot (x' - x) = c_\parallel \cdot |x' - x| \end{aligned}$$

Demnach muss $x \le x'$ oder $x \ge x'$ für alle $x \in X_J$ gelten. Die folgende Tabelle gibt in Abhängigkeit der Werte α_J und α_I an, wie x beschränkt ist.

$\alpha_I = c_\parallel \ \Rightarrow \ x \le x'$	$\alpha_I = 0 \ \Rightarrow \ x = x'$	$\alpha_I = -c_\parallel \ \Rightarrow \ x \ge x'$

Für $\alpha_I \neq 0$ gilt:

$$\exists! x' \in X_J : \ f_{cost}(x', I) = \max\left\{f_{cost}(x, I) \;\middle|\; x \in X_J\right\}$$

Für diesen Wert x' folgt dann:

$$\forall x \in X_J : \begin{cases} x \le x' & \text{falls } \alpha_I = \ \ c_\parallel \\ x \ge x' & \text{falls } \alpha_I = -c_\parallel \end{cases}$$

Mit diesem Wert x' gilt also (2.4.2) für $\alpha_I \neq 0$.
Mit den obigen Ausführungen ist der folgende Satz bewiesen:

Satz 2.4.2:
Für $\alpha_I \neq 0$ *existiert eine einzige Kante* $(v', w') \in E(I_\Pi : J)$ *mit*

$$v' \in \arg\max\left\{\pi(v) \;\middle|\; (v, w) \in E(I_\Pi : J)\right\}.$$

Für diese Kante gilt:

$$c_{-\pi}\left((v, w)\right) = c_{-\pi}\left((v', w')\right) + d_I\left(w', w\right)$$

Für $\alpha_I = 0$ existiert eine solche Kante (v', w') nicht. In diesem Fall gilt aber:

$$\forall v, w \in I : \ \pi(v) = \pi(w)$$

Zusammengefasst kann die Prozedur `Label_Nachbarn`(δ, I, Π) folgendermaßen angegeben werden:

Prozedur `Label_Nachbarn`(δ, I, Π)

für alle $J \in \mathcal{N}(I, \Pi)$ **tue**
  für alle $(v', w') \in E(I_\Pi : J) : v' \in \arg\max\left\{\pi(v) \;\middle|\; v \in I_\Pi\right\}$ **tue**
    `Label_Intervall`$\left(\delta + c_{-\pi}\left((v', w')\right), w', J\right)$

Als Beispiel für den Aufruf von `Label_Nachbarn`(δ, I, Π) ist in der folgenden Abbildung gezeigt, für welche Knoten die Prozedur `Label_Intervall` aufgerufen wird. Die Knotenmenge I_Π ist grün eingezeichnet und es ist $\alpha_I = -c_{\|}$ angenommen. Dann wird die Prozedur `Label_Intervall`(Δ, w, J) für alle blau eingezeichneten Knoten $w \in \Gamma^+(I_\Pi)$ aufgerufen.

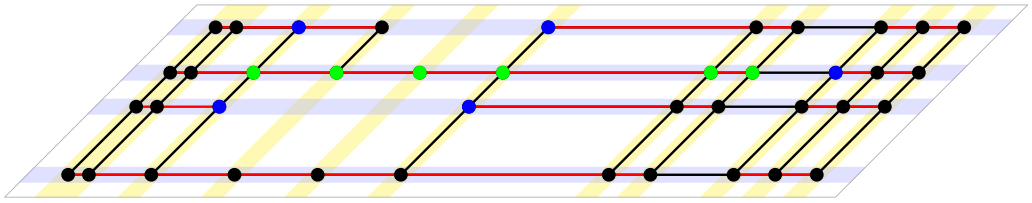


Abbildung 2.4.1: Beispiel für einen Aufruf von `Label_Nachbarn`

2.4.2 Laufzeit von `Label_Nachbarn`(δ, I, Π)

Da $\mathcal{I}_z(t)$ für alle Track-Koordinaten $t \in T_z$ und alle Ebenen $z \in Z$ als Suchbaum organisiert ist, kann die Menge $\mathcal{N}(I, \Pi)$ bestimmt werden in Zeit

$$\left|\mathcal{N}(I, \Pi)\right| \cdot \mathcal{O}\left(\log|\mathcal{I}|\right).$$

Werden diese Suchbäume zudem als geordnete Liste gespeichert, kann die Menge $\mathcal{N}(I, \Pi)$ bestimmt werden in Zeit

$$\mathcal{O}\left(\left|\mathcal{N}(I, \Pi)\right| + \log|\mathcal{I}|\right).$$

Diese Laufzeit liefert im Folgenden allerdings keinen Vorteil. Gilt

$$\exists! v' \in I_\Pi : \pi(v') = \max\left\{\pi(v) \;\middle|\; v \in I_\Pi\right\},$$

so ist die Anzahl der Aufrufe von `Label_Intervall` durch $|\mathcal{N}(I, \Pi)|$ beschränkt. Die Gesamtlaufzeit der Aufrufe von `Label_Intervall` ist dann mit Satz 2.3.16 beschränkt durch:

$$\left|\mathcal{N}(I, \Pi)\right| \cdot \mathcal{O}\left(\log|\mathcal{I}|\right)$$

Berücksichtigung des Sonderfalls $\mathcal{N}(I, \Pi) = \emptyset$ ergibt die Gesamtlaufzeit:

Satz 2.4.3:
Für ein Intervall $I \in \mathcal{I}(z)$ *ist die Prozedur* `Label_Nachbarn`(δ, I, Π) *unter der Voraussetzung*

$$\left|\arg\max\left\{\pi(v) \;\middle|\; v \in I : v_1 \in \Pi\right\}\right| = 1$$

in folgender Zeit realisierbar:

$$\left(1 + \left|\mathcal{N}(I, \Pi)\right|\right) \cdot \mathcal{O}\left(\log|\mathcal{I}|\right)$$

Ein Aufruf von `Label_Nachbarn` innerhalb von `Prozessiere_Block` wird so erfolgen, dass die Voraussetzung des Satzes erfüllt ist.

2.5 Prozedur `Prozessiere_Block`

In diesem Abschnitt wird gezeigt, wie die Prozedur `Prozessiere_Block` schnell realisiert werden kann. Ein Aufruf von `Prozessiere_Block`(B_i, δ) bewirkt diese Operationen:

für alle $v \in \mathrm{B}_i : l(v) = \delta$ **tue**
  für alle $w \in \Gamma^+(v)$ **tue**
    wenn $w_3 = v_3$ **dann**
      Sei $\mathrm{J} \in \mathcal{I}$ das Intervall mit $w \in \mathrm{J}$.
      `Label_Intervall`$\left(\delta + c_{\_\pi}\left((v,w)\right), w, \mathrm{J}\right)$
    sonst
      Sei $\mathrm{B} \in \mathcal{B}$ der Block mit $w \in \mathrm{B}$.
      $\mathcal{R}\left(\mathrm{B}, \delta + c_{\mathcal{B}}(\mathrm{B}_i, \mathrm{B})\right) \leftarrow \mathcal{R}\left(\mathrm{B}, \delta + c_{\mathcal{B}}(\mathrm{B}_i, \mathrm{B})\right) \cup \left\{\{w\}\right\}$

In diesem Abschnitt wird ein Aufruf von `Prozessiere_Block`(B_i, δ) betrachtet und für die Ebene $z := z(\mathrm{B}_i)$ gilt o.B.d.A. $\mathrm{VR}(z) = \text{horizontal } [\leftrightarrows]$.

2.5.1 Auswahl von Intervallen

Im Folgenden wird diese Definition benötigt:

Definition 2.5.1:
Für ein Intervall $\mathrm{I} \in \mathcal{I}(z)$ *ist die Menge* $\mathcal{D}(\mathrm{I}, \delta, l)$ *wie folgt definiert:*

- *Falls* $\alpha_\mathrm{I} = 0$*:*
$$\mathcal{D}(\mathrm{I}, \delta, l) := \left\{[v_1, v_1] \;\middle|\; v \in \mathrm{I} : l(v) = \delta\right\}$$
- *Falls* $\alpha_\mathrm{I} \neq 0$ *ist die Funktion* $\Delta_{\mathcal{L}(\mathrm{I})}$ *nach Bemerkung 2.3.5 monoton und daher* $\mathcal{G}'\left[\left\{v \in \mathrm{I} \;\middle|\; l(v) = \delta\right\}\right]$ *zusammenhängend. Deshalb:*
$$\mathcal{D}(\mathrm{I}, \delta, l) := \left\{\left[\min\left\{v_1 \;\middle|\; v \in \mathrm{I} : l(v) = \delta\right\}, \max\left\{v_1 \;\middle|\; v \in \mathrm{I} : l(v) = \delta\right\}\right]\right\}$$

Mit dieser Definition folgt:

Bemerkung 2.5.2:
Es gilt für ein Intervall $\mathrm{I} \in \mathcal{I}(z)$*:*
$$\left\{v_1 \;\middle|\; v \in \mathrm{I} : l(v) = \delta\right\} = \dot{\bigcup_{\Pi \in \mathcal{D}(\mathrm{I}, \delta, l)}} \Pi \cap \mathrm{X}(\mathrm{I})$$

Ein Beispiel für die Definition von $\mathcal{D}(\mathrm{I}, \delta, l)$ für ein Intervall I mit $\alpha_\mathrm{I} = c_{\|}$ bzw. mit $\alpha_\mathrm{I} = 0$ ist in den beiden folgenden Abbildungen angegeben.

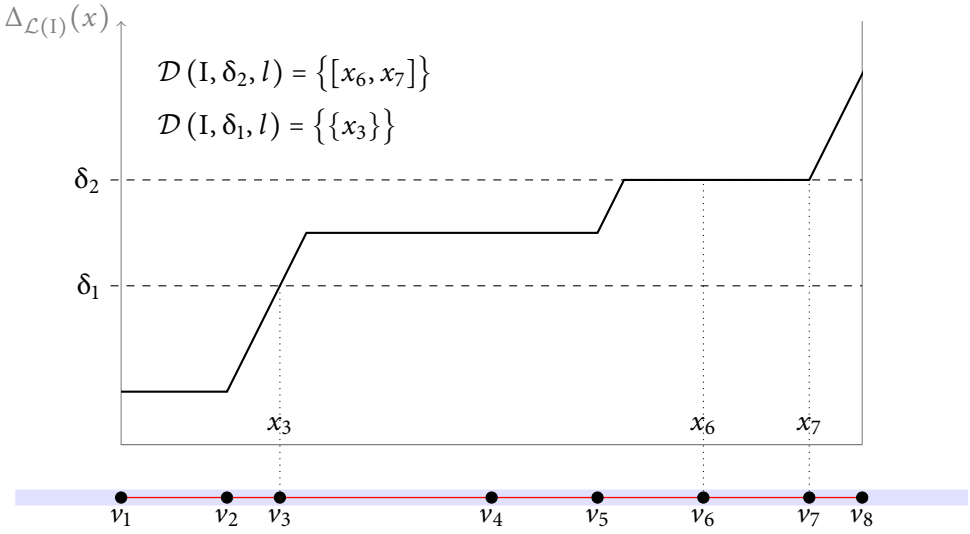


Abbildung 2.5.1: Beispiel für $\mathcal{D}(\mathrm{I}, \delta, l)$ für ein Intervall I mit $\alpha_\mathrm{I} = c_\parallel$

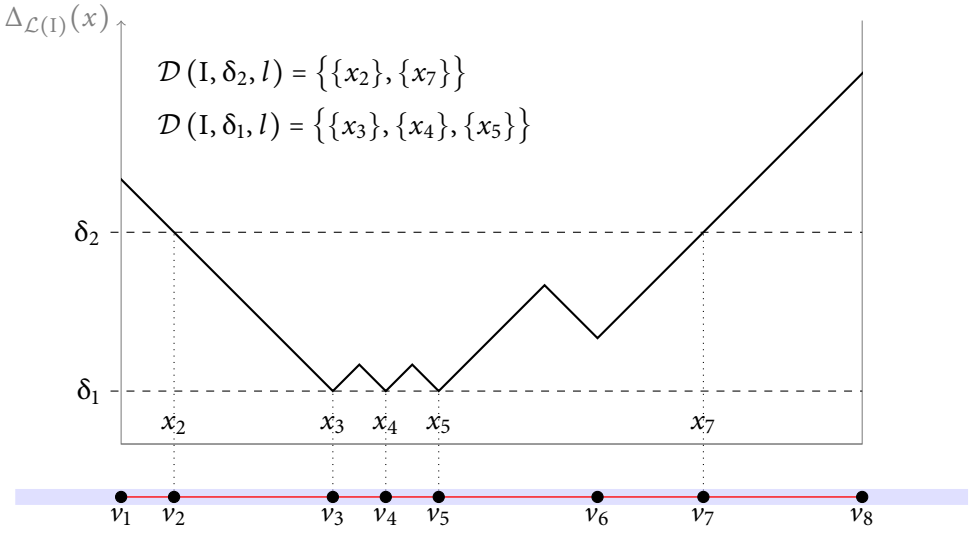


Abbildung 2.5.2: Beispiel für $\mathcal{D}(\mathrm{I}, \delta, l)$ für ein Intervall I mit $\alpha_\mathrm{I} = 0$

Die von der Prozedur `Prozessiere_Block` ausgeführten Operationen werden nun so formuliert, dass Intervalle ausgewählt werden:

$\mathrm{N} \leftarrow \emptyset$
für alle $\mathrm{I} \in \mathcal{I}(\mathrm{B}_i)$ **mit** $\exists v \in \mathrm{I} \setminus \mathrm{N} : l(v) = \delta$ **tue**
  für alle $\Pi \in \mathcal{D}(\mathrm{I}, \delta, l)$ **tue**
    für alle $(v, w) \in \delta^+\left(\left(\Pi \cap \mathrm{X}(\mathrm{I})\right) \times \mathrm{Y}(\mathrm{I}) \times \{z(\mathrm{I})\}\right)$ **tue**
      wenn $v_3 = w_3$ **dann**
        Sei $\mathrm{J} \in \mathcal{I}$ das Intervall mit $w \in \mathrm{J}$.
        `Label_Intervall`$\left(\delta + c_{-\pi}\left((v, w)\right), w, \mathrm{J}\right)$
      sonst
        Sei $\mathrm{B} \in \mathcal{B}$ der Block mit $w \in \mathrm{B}$.
        $\mathcal{R}\left(\mathrm{B}, \delta + c_{\mathcal{B}}(\mathrm{B}_i, \mathrm{B})\right) \leftarrow \mathcal{R}\left(\mathrm{B}, \delta + c_{\mathcal{B}}(\mathrm{B}_i, \mathrm{B})\right) \cup \left\{\{w\}\right\}$
    $\mathrm{N} \leftarrow \mathrm{N} \cup \left(\left(\Pi \cap \mathrm{X}(\mathrm{I})\right) \times \mathrm{Y}(\mathrm{I}) \times \{z(\mathrm{I})\}\right)$

In der äußersten Schleife soll das Intervall $\mathrm{I} \in \mathcal{I}(\mathrm{B}_i)$ ausgewählt werden, für das gilt:

$$\exists v \in \mathrm{I} : \; v \in \arg\max\left\{\pi(v') \;\middle|\; v' \in \mathrm{B}_i \setminus \mathrm{N} : l(v') = \delta\right\}$$

Dann wird jedes Intervall $\mathrm{I} \in \mathcal{I}(\mathrm{B}_i)$ maximal einmal ausgewählt:

Satz 2.5.3:
Bezeichnet die Folge $(\mathrm{I}_k)_{k \in \{1,\ldots,m\}}$ *das in Iteration* k *der äußersten Schleife ausgewählte Intervall, so gilt:*

$$\forall k, k' \in \{1, \ldots, m\} : \; \left(k \neq k' \;\Rightarrow\; \mathrm{I}_k \neq \mathrm{I}_{k'}\right)$$

Beweis:
Die Aussage wird durch Widerspruch gezeigt.
Angenommen, es existieren $k, k' \in \{1, \ldots, m\}$ mit $k \neq k'$ und $\mathrm{I}_k = \mathrm{I}_{k'} =: \mathrm{I}$. In Iteration k sei der Knoten $v \in \mathrm{I}$ so gewählt, dass gilt:

$$v \in \arg\max\left\{\pi(v') \;\middle|\; v' \in \mathrm{I} : l(v') = \delta\right\}$$

Zum späteren Zeitpunkt in Iteration k' existiert ein Knoten $w \in \mathrm{I}$ mit $l(w) > \delta$ in Iteration k und $l(w) = \delta$ in Iteration k'. Nun gilt für eine Kante $e = (v', w') \in \mathrm{E}(\mathcal{G}')$:

$$c_{-\pi}(e) = 0 \quad \Rightarrow \quad c(e) + \left(-\pi(v')\right) - \left(-\pi(w')\right) = 0$$

$$\underset{c(e)>0}{\Rightarrow} \quad \pi(v') > \pi(w')$$

Deshalb gilt für die *future cost*: $\pi(v) > \pi(w)$. Daraus folgt $\alpha_{\mathrm{I}} \neq 0$. Falls $\alpha_{\mathrm{I}} = c_{\|}$, so gilt $v_1 < w_1$ und falls $\alpha_{\mathrm{I}} = -c_{\|}$, so gilt $v_1 > w_1$. Dies ist ein Widerspruch, da in beiden Fällen wegen der Monotonie der Funktion $\Delta_{\mathcal{L}(\mathrm{I})}$ entsprechend Bemerkung 2.3.5 in Iteration k gilt:

$$l(w) \leq l(v)$$

□

Dieser Satz impliziert, dass die Angabe der Menge N entfallen kann. Die in Abschnitt 2.3 angegebene Prozedur `Label_Nachbarn` wird genutzt, um eine schnelle Realisierung der Prozedur `Prozessiere_Block` anzugeben:

Prozedur `Aktualisiere_Block`(B_i, δ)

für alle $\mathrm{I} \in \mathcal{I}(\mathrm{B}_i)$ **mit** $\exists v \in \mathrm{I} : l(v) = \delta$ **tue**

① ***Setzen der Labels***

für alle $\Pi \in \mathcal{D}(\mathrm{I}, \delta, l)$ **tue**

`Label_Nachbarn`$(\delta, \mathrm{I}, \Pi)$

für alle $\mathrm{B} \in \mathcal{B} \setminus \{\mathrm{B}_i\} : \mathrm{E}(\mathrm{B}_i, \mathrm{B}) \neq \emptyset$ **tue**

$\mathcal{R}\left(\mathrm{B}, \delta + c_{\mathcal{B}}(\mathrm{B}_i, \mathrm{B})\right) \leftarrow \mathcal{R}\left(\mathrm{B}, \delta + c_{\mathcal{B}}(\mathrm{B}_i, \mathrm{B})\right) \cup \left\{\Pi \times \mathrm{Y}(\mathrm{I}) \times \{z(\mathrm{B})\}\right\}$

Die Modifikation der Mengen $\mathcal{R}(\mathrm{B}, \Delta)$ mit $\mathrm{B} \in \mathcal{B}, \Delta \in \mathbb{Z}_{\geq 0}$ entspricht nicht der zu Beginn dieses Abschnitts 2.5 angegebenen Operation. In Abschnitt 2.6 wird jedoch eine schnelle Realisierung der Prozedur `Prozessiere_Registrierte` durch diesen Unterschied ermöglicht.

Im Folgenden bezeichnet $\hat{l}$ die Funktion l nach Terminierung von `Prozessiere_Block`. Da jedes Intervall in der äußeren Schleife nur einmal betrachtet wird, gilt in Iteration i der äußersten Schleife für das ausgewählte Intervall I:

$$\mathcal{D}\left(\mathrm{I}, \delta, l\right) = \mathcal{D}\left(\mathrm{I}, \delta, \hat{l}\right)$$

Für die nachfolgenden Abschnitte wird die Bemerkung formuliert:

Bemerkung 2.5.4:
Für einen Block $\mathrm{B} \in \mathcal{B}$ *wird während eines Aufrufes von* `Prozessiere_Block`(B_i, δ) *eine Teilmenge von*

$$\dot{\bigcup_{\mathrm{I} \in \mathcal{I}(\mathrm{B}_i)}} \mathcal{D}\left(\mathrm{I}, \delta, \hat{l}\right) \times \mathrm{Y}\left(\mathrm{I}\right) \times \left\{z(\mathrm{B})\right\}$$

zu $\mathcal{R}\left(\mathrm{B}, \delta + c_{\mathcal{B}}(\mathrm{B}_i, \mathrm{B})\right)$ *hinzugefügt. Bezeichnet* $\bar{\mathcal{R}}$ *die Menge der hinzugefügten Elemente, dann gilt, da benachbarte Ebenen unterschiedliche Vorzugsrichtungen haben:*

$$\Gamma^+\left(\left\{v \in \mathrm{B}_i \;\middle|\; l(v) = \delta\right\}\right) \cap \mathrm{B} \;=\; \bigcup_{\mathrm{R} \in \bar{\mathcal{R}}} \left(\mathrm{R} \cap \left(\mathrm{T}_{z(\mathrm{B})} \times \mathbb{R} \times \mathbb{R}\right)\right)$$

2.5.2 Laufzeit von `Prozessiere_Block`(B_i, δ)

Laufzeit von Abschnitt ①

Angenommen, das Intervall I ist ausgewählt und die Menge $\mathcal{D}(\mathrm{I}, \delta, l)$ berechnet. Dann wird für alle $\Pi \in \mathcal{D}(\mathrm{I}, \delta, l)$ die Prozedur `Label_Nachbarn` aufgerufen. Es gilt nach Definition 2.5.1 für jedes $\Pi \in \mathcal{D}(\mathrm{I}, \delta, l)$:

$$\left|\arg\max\left\{\pi(v) \;\middle|\; v \in \mathrm{I} : v_1 \in \Pi\right\}\right| = 1$$

Deshalb lässt sich die Laufzeit von `Label_Nachbarn` entsprechend Satz 2.4.3 beschränken. Es ergibt sich folgende maximale Laufzeit der Aufrufe von `Label_Nachbarn`:

$$\sum_{\Pi \in \mathcal{D}(\mathrm{I},\delta,l)} \left(1 + \left|\mathcal{N}(\mathrm{I},\Pi)\right|\right) \cdot \mathcal{O}\left(\log|\mathcal{I}|\right)$$

Da die Menge $\mathcal{R}(\mathrm{B},\Delta)$ nur in der Prozedur `Prozessiere_Block` modifiziert wird, gilt:

$$\forall \mathrm{B} \in \mathcal{B}, \delta + c_{-\pi_{\max}} < \Delta \in \mathbb{Z}_{\geq 0} : \ \mathcal{R}(\mathrm{B},\Delta) = \emptyset$$

Da die Menge $\mathcal{R}(\mathrm{B},\Delta)$ bei einem Aufruf von `Prozessiere_Block`(B,Δ) nicht mehr benötigt wird, können alle Mengen $\mathcal{R}(\mathrm{B},\Delta)$ mit $\mathrm{B} \in \mathcal{B}, \Delta \in \mathbb{Z}_{\geq 0}$ durch ein Array der Größe $|\mathcal{B}| \cdot c_{-\pi_{\max}}$ realisiert werden.

Definition 2.5.5:
Für einen Block $\mathrm{B}_i \in \mathcal{B}$ *definiere:*

$$\left|\Gamma^+(\mathrm{B}_i)\right| := \left|\left\{\mathrm{B} \in \mathcal{B} \;\middle|\; \mathrm{E}(\mathrm{B}_i : \mathrm{B}) \neq \emptyset\right\}\right|$$

Mit dieser Definition wird $\left|\mathrm{E}(\mathcal{B})\right|$ *folgendermaßen angegeben:*

$$\left|\mathrm{E}(\mathcal{B})\right| := \sum_{\mathrm{B}_i \in \mathcal{B}} \left|\Gamma^+(\mathrm{B}_i)\right|$$

Mit dieser Definition und weil nach Voraussetzung die Menge $\left\{\mathrm{B} \in \mathcal{B} \;\middle|\; \mathrm{E}(\mathrm{B}_i,\mathrm{B}) \neq \emptyset\right\}$ in Zeit $\mathcal{O}(\log|\mathcal{I}|)$ und die Kosten $c_{\mathcal{B}}(\mathrm{B}_i,\mathrm{B})$ für jeden Block B dieser Menge ebenfalls in Zeit $\mathcal{O}(\log|\mathcal{I}|)$ angegeben werden kann, ist Abschnitt ① in der folgenden Zeit realisierbar:

$$\sum_{\Pi \in \mathcal{D}(\mathrm{I},\delta,l)} \left(1 + \left|\mathcal{N}(\mathrm{I},\Pi)\right| + \left|\Gamma^+(\mathrm{B}_i)\right|\right) \cdot \mathcal{O}\left(\log|\mathcal{I}|\right) \tag{2.5.1}$$

Laufzeit der Bestimmung von $\mathcal{D}(\mathrm{I},\delta,l)$

Es wird gezeigt, dass die Menge $\mathcal{D}(\mathrm{I},\delta,l)$ in folgender Zeit bestimmt werden kann:

$$\left|\mathcal{D}(\mathrm{I},\delta,l)\right| \cdot \mathcal{O}\left(\log|\mathcal{I}|\right)$$

Dazu wird für jedes Intervall $\mathrm{J} \in \mathcal{I}(z)$ eine Menge $\mathcal{L}'(\mathrm{J})$ definiert. Jedes Paar $(\Delta,x) \in \mathcal{L}(\mathrm{J})$ korrespondiert zu genau einem Element $(\lambda,\Delta,x) \in \mathcal{L}'(\mathrm{J})$ mit $\lambda \in \mathbb{Z}_{\geq 0}$. Die Menge $\mathcal{L}'(\mathrm{J})$ soll als Suchbaum implementiert werden, der den Wert λ als Schlüssel für ein Element $(\lambda,\Delta,x) \in \mathcal{L}'(\mathrm{J})$ verwendet. Für ein Element $(\lambda,\Delta,x) \in \mathcal{L}'(\mathrm{J})$ und den Knoten $w \in \mathrm{J}$ mit $w_1 = x$ ist der Wert λ folgendermaßen definiert:

$$\lambda := \min\left\{ \Delta + d_{\mathrm{I}}(w,\bar{w}) \;\middle|\; \bar{w} \in \mathrm{I} : \begin{array}{l} 1.\ \bar{w} \text{ wurde noch nicht betrachtet innerhalb der Aufrufe von } \texttt{Prozessiere\_Block} \\ 2.\ l(w) = \Delta + d_{\mathrm{I}}(w,\bar{w}) \end{array} \right\} \tag{2.5.2}$$

Zu beachten ist, dass $\lambda = \infty$ ebenfalls möglich ist.

Um diese Menge $\mathcal{L}'(J)$ zu erhalten, müssen in den Prozeduren `Label_Intervall` und `Prozessiere_Block` zusätzliche Operationen ausgeführt werden.
Die in `Label_Intervall` ausgeführte Operation

$$\mathcal{L}(J) \leftarrow \mathcal{L}(J) \cup \{(\Delta, w_1)\}$$

muss durch die Operation

$$\mathcal{L}'(J) \leftarrow \mathcal{L}'(J) \cup \{(\Delta, \Delta, w_1)\}$$

und die Operation

$$\mathcal{L}(J) \leftarrow \mathcal{L}(J) \setminus \{(\Delta', x')\}$$

durch die Operation

$$\mathcal{L}'(J) \leftarrow \mathcal{L}'(J) \setminus \{(\Delta', \Delta', x')\}$$

ergänzt werden. Außerdem müssen nach Terminierung der Prozedur `Label_Intervall` die Werte λ_i für jedes Element $(\lambda_i, \Delta_i, x_i) \in \mathcal{L}'(J)$ neu berechnet werden, falls ein Paar (Δ, w_1) zu $\mathcal{L}(J)$ hinzugefügt wurde. Da jedes Element $(\Delta_i, x_i) \in \mathcal{L}(J)$ entsprechend Bemerkung 2.3.4 nicht redundant ist bzgl. $\mathcal{L}(J) \setminus \{(\Delta', x')\}$, werden bei dieser Neuberechnung nur die Werte λ_i modifziert, für die $(\Delta_i, x_i) \in N_J(w_1)$ gilt (vgl. Definition 2.3.14). Entsprechend dieser Definition gilt $|N_J(x')| \leq 2$. Da die Menge $\mathcal{L}'(J)$ als Suchbaum implementiert ist, ändert sich deshalb die Laufzeit der Prozedur `Label_Intervall` nicht.

Als Beispiel für die zusätzlichen Operationen, die in der Prozedur `Label_Intervall` ausgeführt werden müssen, sind die beiden nachfolgenden Abbildungen angegeben. Es wird ein Intervall $I = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$ mit $X(I) = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$ und ein Aufruf der Prozedur von `Label_Intervall`(Δ_6, x_6, I) betrachtet. Es wird angenommen, dass die Knoten $v \in I$ mit $l(v) = \Delta_2$ bereits innerhalb eines Aufrufes der Prozedur `Prozessiere_Block` betrachtet wurden. Ein Aufruf von `Label_Nachbarn`$(\Delta_2, I, [x_1, x_2])$ ist also bereits erfolgt. Dann gilt vor dem Aufruf von `Label_Intervall`:

$$\mathcal{L}'(I) = \{(\Delta_2 + c_{\|} \cdot (x_3 - x_2), \Delta_2, x_2), (\Delta_4, \Delta_4, x_4), (\Delta_5, \Delta_5, x_5), (\Delta_7, \Delta_7, x_7)\}$$

Danach sind $(\Delta_4, \Delta_4, x_4)$ und $(\Delta_5, \Delta_5, x_5)$ redundant bzgl. $\mathcal{L}(I)$:

$$\mathcal{L}'(I) = \{(\infty, \Delta_2, x_2), (\Delta_6, \Delta_6, x_6), (\Delta_7, \Delta_7, x_7)\}$$

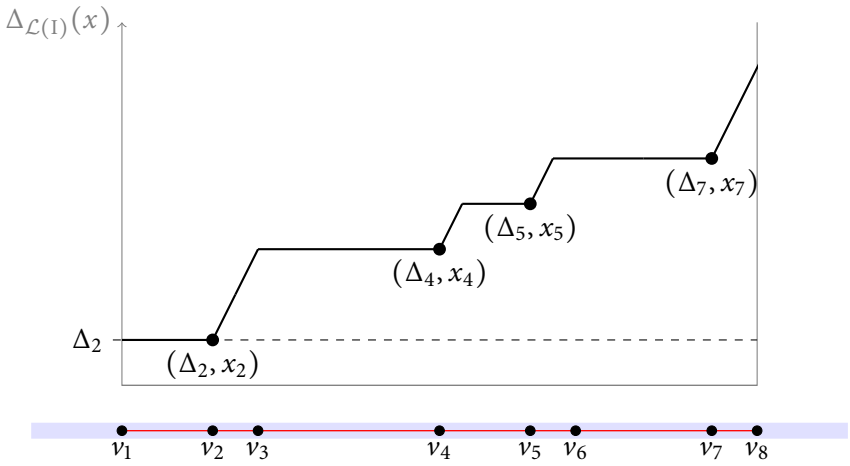


Abbildung 2.5.3: $l\big|_{\mathrm{I}}$ vor einem Aufruf von `Label_Intervall`

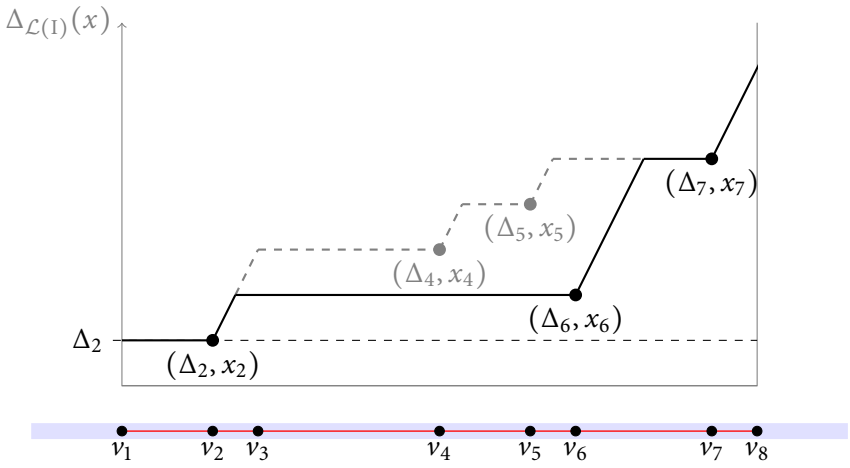


Abbildung 2.5.4: $l\big|_{\mathrm{I}}$ nach einem Aufruf von `Label_Intervall`

Nach (2.5.2) kann die Menge $\mathcal{D}(\mathrm{I}, \delta, l)$ mit den Elementen der Menge

$$\mathrm{M} := \left\{ \left(\lambda, \Delta', x'\right) \in \mathcal{L}'(\mathrm{I}) \;\middle|\; \lambda = \delta \right\}$$

bestimmt werden in Zeit

$$2 \cdot \left|\mathcal{D}(\mathrm{I}, \delta, l)\right| \cdot \mathcal{O}\left(\log|\mathcal{I}|\right).$$

Im Anschluss an diese Bestimmung muss der Werte λ_i für jedes Element $(\lambda_i, \Delta_i, x_i) \in \mathcal{L}'(\mathrm{I})$ mit $(\lambda_i, \Delta_i, x_i) \in \mathrm{M}$ neu berechnet werden. Wie zuvor beschrieben kann dies für jedes Element $(\lambda_i, \Delta_i, x_i) \in \mathcal{L}'(\mathrm{I})$ mit der Menge $\mathrm{N}_\mathrm{I}(x_i)$ geschehen. Damit ist die Laufzeit für diese Neuberechnung beschränkt durch

$$\left|\mathcal{D}(\mathrm{I}, \delta, l)\right| \cdot \mathcal{O}\left(\log|\mathcal{I}|\right).$$

Also ist das Bestimmen der Menge $\mathcal{D}(\mathrm{I}, \delta, l)$ mit Hilfe der Menge $\mathcal{L}'(\mathrm{I})$ in folgender Zeit möglich:

$$\left|\mathcal{D}(\mathrm{I}, \delta, l)\right| \cdot \mathcal{O}\left(\log|\mathcal{I}|\right) \tag{2.5.3}$$

Heap zur Verwaltung der Intervalle

Um die Intervallmenge $\mathcal{I}(\mathrm{B}_i)$ entsprechend zu verwalten, kann ein Heap $\mathcal{H}(\mathrm{B}_i)$ genutzt, werden, der für jedes Intervall $\mathrm{I} \in \mathcal{I}(\mathrm{B}_i)$ ein assoziiertes Element enthält, für das folgender Wert als Schlüssel definiert ist:

$$\min\left\{ l(v) \;\middle|\; v \in \mathrm{I} : \begin{array}{l} v \text{ wurde noch nicht betrachtet innerhalb} \\ \text{der Aufrufe von } \texttt{Prozessiere\_Block} \end{array} \right\}$$

Nach obiger Definition der Menge $\mathcal{L}'(\mathrm{I})$ ist dieses Minimum durch

$$\min\left\{\lambda \;\middle|\; \exists \Delta, x : (\lambda, \Delta, x) \in \mathcal{L}'(\mathrm{I})\right\}$$

gegeben.

Für jedes Intervall, für das `Label_Intervall` innerhalb der Prozedur `Label_Nachbarn` aufgerufen wird, muss also eine entsprechende Heap-Operation durchgeführt werden. Da $|\mathcal{H}(\mathrm{B}_i)|$ durch $|\mathcal{I}|$ begrenzt ist, kann diese in Zeit $\mathcal{O}(\log|\mathcal{I}|)$ realisiert werden. Damit ändert sich die Laufzeit von `Label_Intervall` nicht.

Weiterhin muss für jedes Intervall $\mathrm{I} \in \mathcal{I}(\mathrm{B}_i)$ mit $\mathcal{D}(\mathrm{I}, \delta, l) \neq \emptyset$ das assoziierte Heap-Element aktualisiert werden. Wie in Satz 2.5.3 gezeigt, muss jedes Intervall in der äußeren Schleife der Prozedur `Prozessiere_Block` nur einmal betrachtet werden. Deshalb ist die Anzahl dieser Operationen beschränkt durch:

$$\sum_{\mathrm{I} \in \mathcal{I}(\mathrm{B}_i)} \left|\mathcal{D}(\mathrm{I}, \delta, \hat{l})\right|$$

Damit ergibt sich folgende Gesamtlaufzeit für die Heap-Operationen:

$$\sum_{\mathrm{I} \in \mathcal{I}(\mathrm{B}_i)} \left|\mathcal{D}(\mathrm{I}, \delta, \hat{l})\right| \cdot \mathcal{O}\left(\log|\mathcal{I}|\right) \tag{2.5.4}$$

Alternativ können statt eines Heaps sogenannte „Buckets“ verwendet werden, sodass der Term $\mathcal{O}(\log|\mathcal{I}|)$ in (2.5.4) entfallen würde. Diese verbesserte Laufzeit ergibt allerdings keinen Vorteil für die theoretische Laufzeit von `Prozessiere_Block`.

Zusammenfassung der Laufzeiten

Definition 2.5.6:
Für einen Block $\mathrm{B} \in \mathcal{B}$ *und* $\Delta \in \mathbb{Z}_{\geq 0}$ *ist die Menge* $\mathcal{A}(\mathrm{B}, \Delta, l)$ *definiert durch:*

$$\mathcal{A}(\mathrm{B}, \Delta, l) := \bigcup_{\mathrm{I} \in \mathcal{I}(\mathrm{B})}^{\cdot} \{\mathrm{I}\} \times \mathcal{D}(\mathrm{I}, \Delta, l)$$

Weiterhin ist $\mathcal{A}(\Delta, l)$ *für* $\Delta \in \mathbb{Z}_{\geq 0}$ *definiert durch:*

$$\mathcal{A}(\Delta, l) := \bigcup_{\mathrm{B} \in \mathcal{B}}^{\cdot} \mathcal{A}(\mathrm{B}, \Delta, l)$$

Jedes Intervall in der äußeren Schleife der Prozedur `Prozessiere_Block` muss, wie in Satz 2.5.3 gezeigt, nur einmal betrachtet werden. Deshalb lautet eine obere Schranke der Laufzeit eines Aufrufes von `Prozessiere_Block`(B_i, δ) entsprechend (2.5.1), (2.5.3) und (2.5.4):

$$\left(\left|\Gamma^+(\mathrm{B}_i)\right| \cdot \left|\mathcal{A}(\mathrm{B}_i, \delta, \hat{l})\right| + \sum_{(\mathrm{I},\Pi) \in \mathcal{A}\left(\mathrm{B}_i, \delta, \hat{l}\right)} \left|\mathcal{N}(\mathrm{I}, \Pi)\right| \right) \cdot \mathcal{O}\left(\log |\mathcal{I}|\right) \tag{2.5.5}$$

Um diese Summe abzuschätzen, werden die maximalen reduzierten Kantenkosten definiert:

Definition 2.5.7:
Das Maximum aller reduzierten Kantenkosten ist definiert als

$$c_{-\pi_{\max}} := \max\left\{ c_{\text{-}\pi}(e) \;\middle|\; e \in \mathrm{E}(\mathcal{G}') \right\}.$$

Im Folgenden wird die Abschätzung

$$\sum_{(\mathrm{I},\Pi) \in \mathcal{A}\left(\mathrm{B}_i, \delta, \hat{l}\right)} \left|\mathcal{N}(\mathrm{I}, \Pi)\right| \leq \sum_{\delta' \in \left\{\delta - c_{-\pi_{\max}}, \ldots, \delta + c_{-\pi_{\max}}\right\}} 8 \cdot \left|\mathcal{A}(\delta', \hat{l})\right|$$

gezeigt. Dazu wird folgende in Satz 2.2.6 bewiesene Aussage über die Funktion $\hat{l}$ genutzt:

$$\forall v \in \mathrm{B}_i : \left(\hat{l}(v) = \delta \;\Rightarrow\; \forall w \in \Gamma^+(v) \cap \mathrm{V}_{z(\mathrm{B}_i)} : \left|\hat{l}(v) - \hat{l}(w)\right| \leq c_{-\pi_{\max}} \right)$$

Mit dieser Annahme folgt:

$$\bigcup_{(\mathrm{I},\Pi) \in \mathcal{A}\left(\mathrm{B}_i, \delta, \hat{l}\right)}^{\cdot} \mathcal{N}(\mathrm{I}, \Pi) \subseteq \mathrm{L} := \left\{ \mathrm{I} \in \mathcal{I}\left(z(\mathrm{B}_i)\right) \;\middle|\; \exists v \in \mathrm{I} : \hat{l}(v) \in \left\{\delta - c_{-\pi_{\max}}, \ldots, \delta + c_{-\pi_{\max}}\right\} \right\}$$

Nach Definition 2.5.6 gilt:

$$|\mathrm{L}| \leq \sum_{\delta' \in \left\{\delta - c_{-\pi_{\max}}, \ldots, \delta + c_{-\pi_{\max}}\right\}} \left|\mathcal{A}(\delta', \hat{l})\right| \tag{2.5.6}$$

Eine disjunkte Zerlegung von $\mathcal{N}(I,\Pi) = \mathcal{N}_0(I,\Pi) \cup \mathcal{N}_1(I,\Pi)$ ist für $(I,\Pi) \in \mathcal{A}(B,\delta,\hat{l})$ definiert durch:

$$\mathcal{N}_0(I,\Pi) := \left\{ J \in \mathcal{N}(I,\Pi) \;\middle|\; \Pi \setminus X(J) = \emptyset \right\}$$
$$\mathcal{N}_1(I,\Pi) := \left\{ J \in \mathcal{N}(I,\Pi) \;\middle|\; \Pi \setminus X(J) \neq \emptyset \right\}$$

Es gilt nach Definition von $\mathcal{G}'$:

$$\left|\mathcal{N}_0(I,\Pi)\right| \leq 2$$

Damit folgt:

$$\sum_{(I,\Pi)\in\mathcal{A}\left(B_i,\delta,\hat{l}\right)} \left|\mathcal{N}_0(I,\Pi)\right| \leq 2 \cdot \left|\mathcal{A}(B,\delta,\hat{l})\right|$$

Wegen

$$\forall J \in L : \left| \left\{ (I,\Pi) \in \mathcal{A}(B_i,\delta,\hat{l}) \;\middle|\; J \in \mathcal{N}_1(I,\Pi) \right\} \right| \leq 6$$

nach Definition von $\mathcal{G}'$ und

$$\dot{\bigcup_{(I,\Pi)\in\mathcal{A}\left(B_i,\delta,\hat{l}\right)}} \mathcal{N}_1(I,\Pi) \subseteq L$$

folgt:

$$\sum_{(I,\Pi)\in\mathcal{A}\left(B_i,\delta,\hat{l}\right)} \left|\mathcal{N}_1(I,\Pi)\right| \leq 6 \cdot |L|$$

Mit (2.5.6) gilt:

$$\sum_{(I,\Pi)\in\mathcal{A}\left(B_i,\delta,\hat{l}\right)} \left|\mathcal{N}_1(I,\Pi)\right| \leq \sum_{\delta'\in\left\{\delta-c_{-\pi_{\max}},\ldots,\delta+c_{-\pi_{\max}}\right\}} 6 \cdot \left|\mathcal{A}(\delta',\hat{l})\right|$$

Daraus ergibt sich:

$$\begin{aligned}\sum_{(I,\Pi)\in\mathcal{A}\left(B_i,\delta,\hat{l}\right)} \left|\mathcal{N}(I,\Pi)\right| &= \sum_{(I,\Pi)\in\mathcal{A}\left(B_i,\delta,\hat{l}\right)} \left|\mathcal{N}_0(I,\Pi)\right| + \left|\mathcal{N}_1(I,\Pi)\right| \\ &\leq \sum_{\delta'\in\left\{\delta-c_{-\pi_{\max}},\ldots,\delta+c_{-\pi_{\max}}\right\}} 8 \cdot \left|\mathcal{A}(\delta',\hat{l})\right|\end{aligned}$$

Mit den obigen Ausführungen folgt für die Laufzeit:

Satz 2.5.8:
Bezeichnet $\hat{l}$ die Funktion l nach Terminierung von `Prozessiere_Block`(B_i,δ) *dann kann diese Prozedur in folgender Zeit realisiert werden:*

$$\sum_{\delta'\in\left\{\delta-c_{-\pi_{\max}},\ldots,\delta+c_{-\pi_{\max}}\right\}} \left|\Gamma^+(B_i)\right| \cdot \left|\mathcal{A}(\delta',\hat{l})\right| \cdot \mathcal{O}\left(\log|\mathcal{I}|\right)$$

2.6 Prozedur `Prozessiere_Registrierte`

In diesem Abschnitt wird gezeigt, wie die Prozedur `Prozessiere_Registrierte` schnell realisiert werden kann. Ein Aufruf von `Prozessiere_Registrierte`$(\mathcal{R}(\mathrm{B}_i, \delta), \mathrm{B}_i, \delta)$ bewirkt die Operationen:

für alle $\mathrm{J} \in \mathcal{I}(\mathrm{B}_i)$ **tue**
  für alle $w \in \left\{ w \in \mathrm{J} \cap \mathrm{R} \;\middle|\; \mathrm{R} \in \mathcal{R}(\mathrm{B}_i, \delta) \right\}$ **tue**
    `Label_Intervall`(δ, w, J)

Im Folgenden wird ein Aufruf von `Prozessiere_Registrierte`$(\mathcal{R}(\mathrm{B}_i, \delta), \mathrm{B}_i, \delta)$ betrachtet und für die Ebene $z := z(\mathrm{B}_i)$ gilt o.B.d.A. $\mathrm{VR}(z) =$ vertikal $[\downdownarrows]$.

2.6.1 Reduzierte Kantenkosten

Es wird gezeigt, dass die Anzahl der Aufrufe der Prozedur `Label_Intervall` reduziert werden kann. Dazu betrachte ein Intervall $\mathrm{J} \in \mathcal{I}(\mathrm{B}_i)$ mit

$$\mathcal{R}_\mathrm{J} := \bigcup_{\mathrm{R} \in \mathcal{R}(\mathrm{B}_i, \delta)} \mathrm{R} \cap \mathrm{J} \neq \emptyset.$$

Falls $\alpha_\mathrm{J} \neq 0$, existiert genau ein Knoten $w \in \mathcal{R}_\mathrm{J}$, der die Funktion $\pi\big|_\mathrm{J}$ maximiert:

$$\{w\} = \arg\max \left\{ \pi(v) \;\middle|\; v \in \mathcal{R}_\mathrm{J} \right\}$$

Mit diesem Knoten w folgt nach Definition der reduzierten Kantenkosten (vgl. Tabelle 2.1.3) für einen beliebigen Knoten $w' \in \mathcal{R}_\mathrm{J}$:

$$\begin{aligned} \forall \bar{w} \in \mathrm{J}: \; \delta + d_\mathrm{I}(w, \bar{w}) &\leq \delta + \underbrace{d_\mathrm{I}(w, w')}_{=0} + d_\mathrm{I}(w', \bar{w}) \\ &= \delta + d_\mathrm{I}(w', \bar{w}) \end{aligned}$$

Falls $\alpha_\mathrm{J} = 0$, gilt $\pi(v) = \pi(w)$ für alle Knoten $v, w \in \mathrm{J}$. Also kann nach Satz 2.3.3 die Anzahl der Aufrufe von `Label_Intervall` reduziert werden, sodass die von der Prozedur `Prozessiere_Registrierte` ausgeführten Operationen äquivalent durch folgende Operationen realisiert werden:

für alle $\mathrm{J} \in \mathcal{I}(\mathrm{B}_i)$ **tue**
  für alle $w \in \arg\max \left\{ \pi(w) \;\middle|\; w \in \left\{ w \in \mathrm{J} \cap \mathrm{R} \;\middle|\; \mathrm{R} \in \mathcal{R}(\mathrm{B}_i, \delta) \right\} \right\}$ **tue**
    `Label_Intervall`(δ, w, J)

Als Beispiel für den Aufruf von `Prozessiere_Registrierte`$(\mathcal{R}(\mathrm{B}_i, \delta), \mathrm{B}_i, \delta)$ ist in der folgenden Abbildung gezeigt, für welche Knoten die Prozedur `Label_Intervall` aufgerufen wird. Die Intervalle, die mehr als einen Knoten enthalten, sind rot eingezeichnet. Für jedes $\mathrm{R} \in \mathcal{R}(\mathrm{B}_i, \delta)$ sei R' definiert durch

$$\mathrm{R}' := \left\{ \left(x, y, z(\mathrm{B}_{i-1})\right) \;\middle|\; \left(x, y, z(\mathrm{B}_i)\right) \in \mathrm{R} \right\}.$$

Es wird angenommen, dass die Knotenmenge $\left\{w \in R' \;\middle|\; R \in \mathcal{R}(B_i, \delta)\right\}$ durch die grün eingezeichneten Knoten gegeben ist. Es gibt nur einen Zielknoten $v \in T$. Dieser befindet sich in der Tiefe. Die Prozedur `Label_Intervall`(δ, w, J) wird für alle blau eingezeichneten Knoten $w \in B_i$ aufgerufen.

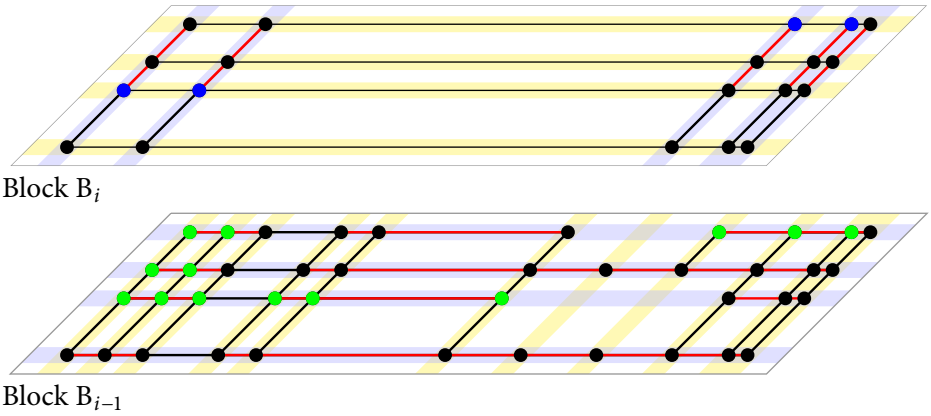


Abbildung 2.6.1: Beispiel für die Aufrufe der Prozedur `Label_Intervall` innerhalb von `Prozessiere_Registrierte`

2.6.2 „Sweepline"-Verfahren

Die Bestimmung der Mengen

$$\left\{w \in J \cap R \;\middle|\; R \in \mathcal{R}(B_i, \delta)\right\}$$

für die Intervalle $J \in \mathcal{I}(B_i)$ kann mit einem „Sweepline"-Verfahren durchgeführt werden. Da die Prozedur `Prozessiere_Block` genutzt wird, um die Menge $\mathcal{R}(B_i, \delta)$ anzugeben, gilt entsprechend Bemerkung 2.5.4:

$$\bigcup_{R \in \mathcal{R}(B_i,\delta)} R \cap V_z = \bigcup_{R \in \mathcal{R}(B_i,\delta)} \left(R \cap (T_z \times \mathbb{R} \times \mathbb{R})\right)$$

Für die Menge T_{B_i} der Track-Koordinaten des Blocks B_i gelte

$$T_{B_i} = \{t_1, \ldots, t_n\} \quad \text{mit} \quad t_1 < \ldots < t_n.$$

Definiere außerdem $t_0 := -\infty$ und $t_{n+1} := \infty$. Mit diesen Koordinaten sind folgende Ereignismengen für alle $t_k \in T_{B_i}$ definiert durch:

$$\mathcal{E}_{Start}\left(t_k, \mathcal{R}(B_i, \delta)\right) := \left\{y(R) \;\middle|\; R \in \mathcal{R}(B_i, \delta) : t_{k-1} < \min X(R) \leq t_k \leq \max X(R)\right\}$$
$$\mathcal{E}_{Stop}\left(t_k, \mathcal{R}(B_i, \delta)\right) := \left\{y(R) \;\middle|\; R \in \mathcal{R}(B_i, \delta) : \min X(R) \leq t_k \leq \max X(R) < t_{k+1}\right\}$$

Nach Definition von T_{B_i} gilt:

$$\mathcal{I}(B_i) = \dot{\bigcup_{t \in T_{B_i}}} \mathcal{I}_{B_i}(t)$$

Deshalb kann die Prozedur `Prozessiere_Registrierte` wie folgt realisiert werden:

Prozedur `Prozessiere_Registrierte`$(\mathcal{R}(\mathrm{B}_i, \delta), \mathrm{B}_i, \delta)$

---

① ***Initialisierung***

wenn $\mathcal{R}(\mathrm{B}_i, \delta) = \emptyset$ **dann**
  Stop
sonst
  $\mathcal{S} \leftarrow \emptyset$

② ***Setzen der VIA-Labels***

für alle $t_k \in \mathrm{T}_{\mathrm{B}_i}$ **tue**
  $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{E}_{Start}\left(t_k, \mathcal{R}(\mathrm{B}_i, \delta)\right)$
  für alle $\mathrm{J} \in \mathcal{I}_{\mathrm{B}_i}(t_k)$ **tue**
    für alle $w \in \arg\max\left\{\pi(v) \mid v \in \mathrm{J} : v_2 \in \mathcal{S}\right\}$ **tue**
      `Label_Intervall`(δ, w, J)
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \mathcal{E}_{Stop}\left(t_k, \mathcal{R}(\mathrm{B}_i, \delta)\right)$

Ein Beispiel für das „Sweepline“-Verfahren zeigt nachfolgende Abbildung. Die Intervalle, die mehr als einen Knoten enthalten, sind entsprechend Abbildung 2.6.1 rot eingezeichnet. Für jedes Element $\mathrm{R} \in \mathcal{R}(\mathrm{B}_i, \delta)$ ist die Menge R' definiert durch

$$\mathrm{R}' := \left\{\left(x, y, z(\mathrm{B}_{i-1})\right) \;\middle|\; \left(x, y, z(\mathrm{B}_i)\right) \in \mathrm{R}\right\}$$

grün eingezeichnet (vgl. Abbildung 2.6.1). Die Menge der Track-Koordinaten $\mathrm{T}_{\mathrm{B}_i}$ ist durch $\mathrm{T}_{\mathrm{B}_i} = \{t_1, t_2, t_3, t_4, t_5\}$ gegeben. „Die Sweepline läuft von links nach rechts.“ Für jeden blau eingezeichneten Knoten $w \in \mathrm{B}_i$ erfolgt ein Aufruf von `Label_Intervall`(δ, w, J).

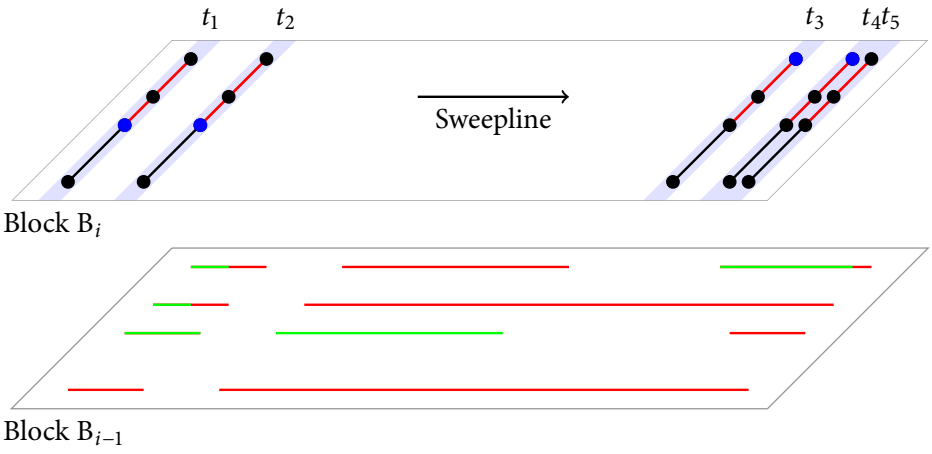


Abbildung 2.6.2: Beispiel für das „Sweepline“-Verfahren

Bemerkung 2.6.1:
Die Größe der „Sweepline“ $\mathcal{S}$ ist zu jedem Zeitpunkt beschränkt:

$$\mathcal{S} \subseteq \left\{ y(\mathrm{R}) \,\middle|\, \mathrm{R} \in \mathcal{R}(\mathrm{B}_i, \delta) \right\} \quad \Rightarrow \quad |\mathcal{S}| \leq |\mathcal{I}|$$

2.6.3 Laufzeit von `Prozessiere_Registrierte`$(\mathcal{R}(\mathrm{B}_i, \delta), \mathrm{B}_i, \delta)$

Laufzeit der Bestimmung der Ereignismengen

Die Ereignismengen $\mathcal{E}_{Start}\left(t_k, \mathcal{R}(\mathrm{B}_i, \delta)\right)$ und $\mathcal{E}_{Stop}\left(t_k, \mathcal{R}(\mathrm{B}_i, \delta)\right)$ können für alle Track-Koordinaten $t_k \in \mathrm{T}_{\mathrm{B}_i}$ mit Hilfe eines Arrays insgesamt bestimmt werden in Zeit:

$$\mathcal{O}\left(\left|\mathcal{R}(\mathrm{B}_i, \delta)\right|\right) \tag{2.6.1}$$

Laufzeit der Aufrufe von `Label_Intervall`

Wie in Abschnitt 2.6.1 beschrieben, wird die Prozedur `Label_Intervall` für ein Intervall $\mathrm{J} \in \mathcal{I}(\mathrm{B}_i)$ genau einmal aufgerufen, falls $\alpha_{\mathrm{J}} \neq 0$. Falls $\alpha_{\mathrm{J}} = 0$ ist die Anzahl der Aufrufe von `Label_Intervall` mit

$$\mathcal{R}_{\mathrm{J}} := \bigcup_{\mathrm{R} \in \mathcal{R}(\mathrm{B}_i, \delta)} \mathrm{R} \cap \mathrm{J}$$

durch $\left|\mathcal{R}_{\mathrm{J}}\right|$ gegeben. Deshalb gilt:

Bemerkung 2.6.2:
Die Prozedur `Label_Intervall`(δ, w, J) *wird für ein Intervall* $\mathrm{J} \in \mathcal{I}(\mathrm{B}_i)$ *mit* $\mathcal{R}_{\mathrm{J}} \neq \emptyset$ *für folgende Knoten w aufgerufen:*

- $\alpha_{\mathrm{J}} = 0$*: für alle Knoten* $w \in \mathrm{J}$ *mit* $w_2 \in \mathcal{S}$
- $\alpha_{\mathrm{J}} \neq 0$*: für den Knoten* $w \in \mathrm{J}$ *mit* $w_2 = \min\left(\mathrm{Y}(\mathrm{J}) \cap \mathcal{S}\right)$ *oder* $w_2 = \max\left(\mathrm{Y}(\mathrm{J}) \cap \mathcal{S}\right)$

Nach Terminierung der Prozedur `Prozessiere_Registrierte` wird die Funktion l im Folgenden mit $\hat{l}$ bezeichnet. Angenommen, es gilt:

$$\forall w \in \left(\bigcup_{\mathrm{R} \in \mathcal{R}(\mathrm{B}_i, \delta)} \mathrm{R} \cap \mathrm{B}_i \right) : \hat{l}(w) \in \left\{ \delta - 2 \cdot c_{-\pi_{\max}}, \ldots, \delta \right\}$$

Dann ist die Anzahl der Aufrufe von `Label_Intervall` entsprechend Definition 2.5.6 und Bemerkung 2.6.2 beschränkt durch:

$$\sum_{\delta' \in \left\{\delta - 2 \cdot c_{-\pi_{\max}}, \ldots, \delta\right\}} \left|\mathcal{A}(\mathrm{B}_i, \delta', \hat{l})\right|$$

Die Annahme ist gültig, wie im nächsten Abschnitt 2.7 gezeigt wird. Nach Satz 2.3.16 ist die Gesamtlaufzeit für die Aufrufe von `Label_Intervall` beschränkt durch:

$$\sum_{\delta' \in \left\{\delta - 2 \cdot c_{-\pi_{\max}}, \ldots, \delta\right\}} \left|\mathcal{A}(\mathrm{B}_i, \delta', \hat{l})\right| \cdot \mathcal{O}\left(\log |\mathcal{I}|\right) \tag{2.6.2}$$

Laufzeit der „Sweepline“-Operationen

Ist die Menge $\mathcal{S}$ als Suchbaum realisiert, so können alle Operationen

$$\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{E}_{Start}\left(t_k, \mathcal{R}\left(\mathrm{B}_i, \delta\right)\right) \quad \text{und} \quad \mathcal{S} \leftarrow \mathcal{S} \setminus \mathcal{E}_{Stop}\left(t_k, \mathcal{R}\left(\mathrm{B}_i, \delta\right)\right)$$

wegen $|\mathcal{S}| \leq |\mathcal{I}|$ nach Bemerkung 2.6.1 ausgeführt werden in Zeit:

$$\sum_{t_k \in \mathrm{T}_{\mathrm{B}_i}} \left(\left|\mathcal{E}_{Start}\left(t_k\right)\right| + \left|\mathcal{E}_{Stop}\left(t_k\right)\right|\right) \cdot \mathcal{O}\left(\log|\mathcal{I}|\right) \quad \leq \quad \left|\mathcal{R}\left(\mathrm{B}_i, \delta\right)\right| \cdot \mathcal{O}\left(\log|\mathcal{I}|\right) \tag{2.6.3}$$

Um für ein Intervall $\mathrm{J} \in \mathcal{I}\left(\mathrm{B}_i\right)$ die Menge $\mathrm{Y}\left(\mathrm{J}\right) \cap \mathcal{S}$ zu bestimmen, reicht es nach Bemerkung 2.6.2, Minimum und Maximum des Schnitts zu berechnen. Da $|\mathcal{S}| \leq |\mathcal{R}\left(\mathrm{B}_i, \delta\right)|$, ist dies in Zeit $\mathcal{O}(\log|\mathcal{R}(\mathrm{B}_i, \delta)|)$ für jedes $\mathrm{J} \in \mathcal{I}(\mathrm{B}_i)$ realisierbar. Aufsummiert ist demnach folgender Zeitaufwand nötig:

$$\left|\mathcal{I}\left(\mathrm{B}_i\right)\right| \cdot \mathcal{O}\left(\log\left|\mathcal{R}\left(\mathrm{B}_i, \delta\right)\right|\right) \tag{2.6.4}$$

Zusammenfassung der Laufzeiten

Aufsummieren von (2.6.1), (2.6.2), (2.6.3) und (2.6.4) impliziert den Satz:

Satz 2.6.3:
Nach Terminierung von `Prozessiere_Registrierte`$(\mathcal{R}(\mathrm{B}_i, \delta), \mathrm{B}_i, \delta)$ *bezeichne* $\hat{l}$ *die Funktion* l*. Angenommen, es gelten folgende Bedingungen:*

- *Es gilt:*
$$\forall w \in \left(\bigcup_{\mathrm{R} \in \mathcal{R}\left(\mathrm{B}_i, \delta\right)} \mathrm{R} \cap \mathrm{B}_i\right) : \ \hat{l}\left(w\right) \in \left\{\delta - 2 \cdot c_{-\pi_{\max}}, \ldots, \delta\right\}$$
- *Die Kardinalität der Menge* $\mathcal{R}\left(\mathrm{B}_i, \delta\right)$ *ist wie folgt beschränkt, wobei* $\bar{l}$ *die Labelfunktion* l *nach Terminierung von* `Suche_Pfad` *bezeichnet:*
$$\left|\mathcal{R}\left(\mathrm{B}_i, \delta\right)\right| \leq \sum_{\delta' \in \left\{\delta - c_{-\pi_{\max}}, \ldots, \delta\right\}} \left|\mathcal{A}(\delta', \bar{l})\right|$$

Dann ist die Prozedur `Prozessiere_Registrierte`$(\mathcal{R}(\mathrm{B}_i, \delta), \mathrm{B}_i, \delta)$ *realisierbar in Zeit:*

$$\sum_{\substack{\delta' \in \left\{\delta - 2 \cdot c_{-\pi_{\max}}, \ldots, \delta\right\}: \\ \mathcal{A}\left(\delta', \hat{l}\right) \neq \emptyset}} \left|\mathcal{A}(\delta', \hat{l})\right| \cdot \mathcal{O}\left(\log|\mathcal{I}|\right)$$

$$\sum_{\substack{\delta' \in \left\{\delta - c_{-\pi_{\max}}, \ldots, \delta\right\}: \\ \mathcal{A}\left(\delta', \bar{l}\right) \neq \emptyset}} \left(|\mathcal{I}| \cdot \mathcal{O}\left(\log\left|\mathcal{A}(\delta', \bar{l})\right|\right) + \left|\mathcal{A}(\delta', \bar{l})\right| \cdot \mathcal{O}\left(\log|\mathcal{I}|\right)\right)$$

Beweis:
Es gilt die Abschätzung:

$$\begin{aligned}
\log\left|\mathcal{R}\left(B_i, \delta\right)\right| &\leq \log\left(\sum_{\delta' \in \left\{\delta - c_{-\pi_{\max}}, \ldots, \delta\right\}} \left|\mathcal{A}(\delta', \bar{l})\right|\right) \\
&\leq \log\left(\prod_{\delta' \in \left\{\delta - c_{-\pi_{\max}}, \ldots, \delta\right\}} \left(\left|\mathcal{A}(\delta', \bar{l})\right| + 1\right)\right) \\
&= \sum_{\delta' \in \left\{\delta - c_{-\pi_{\max}}, \ldots, \delta\right\}} \log\left(\left|\mathcal{A}(\delta', \bar{l})\right| + 1\right) \\
&= \sum_{\substack{\delta' \in \left\{\delta - c_{-\pi_{\max}}, \ldots, \delta\right\}: \\ \mathcal{A}(\delta', \bar{l}) \neq \emptyset}} \mathcal{O}\left(\log\left|\mathcal{A}(\delta', \bar{l})\right|\right)
\end{aligned}$$

□

Im nächsten Abschnitt wird gezeigt, dass die Voraussetzungen dieses Satzes erfüllt sind. Daher erfolgt keine andere Laufzeitabschätzung der Prozedur `Prozessiere_Registrierte`. Um die Laufzeit der Prozedur `Suche_Pfad` im nächsten Abschnitt mit der von Hetzel [1998] angegebenen zu vergleichen, wird folgende Bemerkung benötigt:

Bemerkung 2.6.4:
Mit der zweiten Voraussetzung von Satz 2.6.3 kann (2.6.4) auch folgendermaßen lauten:

$$\left|\left\{v \in V(\mathcal{G}') \;\middle|\; \hat{l}(v) = \delta\right\}\right| \cdot \sum_{\substack{\delta' \in \left\{\delta - 2 \cdot c_{-\pi_{\max}}, \ldots, \delta\right\}: \\ \mathcal{A}(\delta', \bar{l}) \neq \emptyset}} \mathcal{O}\Big(\log \underbrace{\left|\mathcal{A}(\delta', \bar{l})\right|}_{\leq |V(\mathcal{G}')|}\Big) \tag{2.6.5}$$

2.7 Laufzeit einer Pfadsuche

In diesem Abschnitt wird die Laufzeit der Prozedur `Suche_Pfad`(S, T) angegeben. Dazu wird vorausgesetzt, dass für jede Verdrahtungsebene $z \in \mathrm{Z}$ und jede Track-Koordinate $t \in \mathrm{T}_z$ gilt:

$$\sum_{\mathrm{I} \in \mathcal{I}_z(t):\alpha_\mathrm{I}=0} |\mathrm{I}| \leq |\mathcal{I}| \tag{2.7.1}$$

Werden die in Abschnitt 1.3 vorgestellten Funktionen als *future cost* genutzt, ist diese Voraussetzung erfüllt.

2.7.1 Prozedur `Suche_Pfad`

Die schnelle Realisierung der Prozedur wird nochmals angegeben:

Prozedur `Suche_Pfad(S, T)`

---

① ***Initialisierung***

für alle $\mathrm{B} \in \mathcal{B}, \Delta \in \mathbb{Z}_{\geq 0}$ **tue**
  $\mathcal{R}(\mathrm{B}, \Delta) \leftarrow \emptyset$

für alle $w \in \mathrm{V}(\mathcal{G}')$ **tue**
  Sei $\mathrm{J} \in \mathcal{I}$ das Intervall mit $w \in \mathrm{J}$.
  wenn $w \in \mathrm{S}$ **dann**
    `Label_Intervall`$(\pi(w), w, \mathrm{J})$
  sonst
    `Label_Intervall`(∞, w, J)

$\delta \leftarrow \min\left\{l(w) \;\middle|\; w \in \mathrm{V}(\mathcal{G}')\right\}$

② ***Permanentlabeln der Knoten mit potenziellen Kosten*** δ

solange $\delta < \infty$ **und** $\nexists v \in \mathrm{T} : l(v) < \delta$ **tue**
  für alle $\mathrm{B}_i \in \mathcal{B}$ **tue**
    `Prozessiere_Registrierte`$(\mathcal{R}(\mathrm{B}_i, \delta), \mathrm{B}_i, \delta)$
    `Prozessiere_Block`(B_i, δ)
  $\delta \leftarrow \inf\left\{\left\{l(w) \;\middle|\; w \in \mathrm{V}(\mathcal{G}') : l(w) > \delta\right\} \cup \left\{\Delta > \delta \;\middle|\; \exists \mathrm{B}_i \in \mathcal{B} : \mathcal{R}(\mathrm{B}_i, \Delta) \neq \emptyset\right\}\right\}$

Um die Laufzeit der Prozedur `Suche_Pfad` anzugeben, wird zunächst der in Abschnitt 2.3 formulierte Satz 2.3.12 bewiesen. Wie in Abschnitt 2.3 beschrieben, ist dieser Satz Voraussetzung für die Laufzeitangabe der Prozedur `Label_Intervall`. Der Beweis von Satz 2.3.12 folgt aus folgendem Satz:

Satz 2.7.1:
Die Anzahl der Knoten $w \in \mathrm{J}$*, für die* `Label_Intervall`(Δ, w, J) *innerhalb der Prozedur* `Suche_Pfad` *aufgerufen wird, ist für ein Intervall* $\mathrm{J} \in \mathcal{I}$ *beschränkt durch* $|\mathcal{I}|$.

Beweis:
Definiere $z := z(\mathrm{J})$. Es gilt o.B.d.A. $\mathrm{VR}(z) = \text{horizontal } [\leftrightarrows]$.
Falls

$$\mathrm{X}(\mathrm{J}) \subseteq \bigcup_{z' \in \{z-1, z+1\} \cap \mathrm{Z}} \bigcup_{\mathrm{I} \in \mathcal{I}(z')} \left\{ x(\mathrm{I}) \right\},$$

so gilt $|\mathrm{J}| = |\mathrm{X}(\mathrm{J})| \leq |\mathcal{I}|$. Deshalb ist in diesem Fall nichts zu zeigen.
Angenommen, es existiert ein Knoten $w \in \mathrm{J}$ für den es kein $z' \in \{z-1, z+1\} \cap \mathrm{Z}$ gibt, sodass ein Intervall $\mathrm{I} \in \mathcal{I}(z')$ mit $x(\mathrm{I}) = w_1$ existiert. Definiere $x := w_1$. Dann kann ein Aufruf von `Label_Intervall`(Δ, w, J) nur in der Prozedur `Prozessiere_Block` erfolgen. Dieser Aufruf kann nur erfolgen, falls eine der folgenden Aussagen gilt:

- Es gibt einen Startknoten $v \in \mathrm{S}$ mit $v_1 = x$.
- Es gibt ein Intervall $\mathrm{I} \in \mathcal{I}(z)$ mit $\min \mathrm{X}(\mathrm{I}) = x$ oder $\max \mathrm{X}(\mathrm{I}) = x$.
- Es gibt ein Intervall $\mathrm{I} \in \mathcal{I}(\mathrm{B})$ mit $\min \mathrm{X}(\mathrm{I}) < x$ und $\max \mathrm{X}(\mathrm{I}) > x$ und $\alpha_{\mathrm{I}} = 0$.

Die Voraussetzung (2.7.1) wird für den dritten Fall benötigt. Da für jeden Knoten $v \in \mathrm{S}$ nach (2.1.1c) der Definition von $\mathcal{I}$ das Intervall $\{v\}$ in $\mathcal{I}$ enthalten ist, ist der Satz bewiesen. □

2.7.2 Laufzeit von `Suche_Pfad`(S, T)

Im Folgenden werden die Laufzeitanalysen der vorherigen Abschnitte genutzt, um die Laufzeit eines Aufrufes der Prozedur `Suche_Pfad`(S, T) anzugeben. Die Labelfunktion l nach Terminierung von `Suche_Pfad`(S, T) wird mit $\bar{l}$ bezeichnet. Außerdem bezeichnet δ_k den Wert δ in Iteration k der äußeren Schleife in ②. Insgesamt gibt es n Iterationen.

Laufzeit der Aktualisierung von δ

Wie in Abschnitt 2.5.2 beschrieben, wird in der Prozedur `Prozessiere_Block`(B_i, δ) zur Bestimmung der Intervalle $\mathrm{I} \in \mathcal{I}(\mathrm{B})$ mit $\exists v \in \mathrm{I} : l(v) = \delta$ ein Heap $\mathcal{H}$ benötigt. Diese Heaps können zusammengefasst werden, sodass der Wert

$$\inf \left\{ l(v) \;\middle|\; v \in \mathrm{V}(\mathcal{G}') : l(v) > \delta \right\}$$

in Zeit $\mathcal{O}(\log |\mathcal{I}|)$ bestimmt werden kann. Die Ausführungen in Abschnitt 2.5.2 implizieren außerdem, dass der Wert

$$\inf \left\{ \Delta > \delta \;\middle|\; \exists \mathrm{B}_i \in \mathcal{B} : \mathcal{R}(\mathrm{B}_i, \Delta) \neq \emptyset \right\}$$

in Zeit $c_{-\pi_{\max}}$ bestimmt werden kann. Deshalb ist das Bestimmen von δ_k für jeden Index $k \in \{1, \ldots, n\}$ in Zeit

$$\mathcal{O}\left(\log |\mathcal{I}|\right) + c_{-\pi_{\max}} \tag{2.7.2}$$

möglich.

Aufruf von `Prozessiere_Registrierte`

Im Folgenden bezeichnet $\hat{l}$ die Labelfunktion l nach Terminierung eines Aufrufes der Prozedur `Prozessiere_Registrierte`. Um die Laufzeitabschätzung nach Satz 2.6.3 nutzen zu können, wird die Aussage von Satz 2.2.5 genutzt. Mit diesem Satz folgt entsprechend Bemerkung 2.5.4, dass eine Teilmenge von

$$\bigcup_{\mathrm{I}\in\mathcal{I}(\mathrm{B}_j)}^{\cdot} \mathcal{D}\big(\mathrm{I},\delta,\hat{l}\big)\times\mathrm{Y}(\mathrm{I})\times\big\{z(\mathrm{B})\big\} \subseteq \bigcup_{\mathrm{I}\in\mathcal{I}(\mathrm{B}_j)}^{\cdot} \mathcal{D}\big(\mathrm{I},\delta,\bar{l}\big)\times\mathrm{Y}(\mathrm{I})\times\big\{z(\mathrm{B})\big\}$$

während eines Aufrufes von `Prozessiere_Block`(B_j,δ) zu $\mathcal{R}(\mathrm{B},\delta+c_{\mathcal{B}}(\mathrm{B}_j,\mathrm{B}))$ für einen Block $\mathrm{B}\in\mathcal{B}$ hinzugefügt wird. Damit folgt, dass zu Beginn eines Aufrufes der Prozedur `Prozessiere_Registrierte`$(\mathcal{R}(\mathrm{B}_i,\delta),\mathrm{B}_i,\delta)$ gilt:

$$\mathcal{R}(\mathrm{B}_i,\delta)\subseteq\bigcup_{\mathrm{B}\in\mathcal{B}}\left(\bigcup_{\mathrm{I}\in\mathcal{I}(\mathrm{B})}^{\cdot}\mathcal{D}\big(\mathrm{I},\delta-c_{\mathcal{B}}(\mathrm{B},\mathrm{B}_i),\bar{l}\big)\times\mathrm{Y}(\mathrm{I})\times\big\{z(\mathrm{B}_i)\big\}\right)$$

Also gilt:

$$\big|\mathcal{R}(\mathrm{B}_i,\delta)\big|\leq\sum_{\delta'\in\{\delta-c_{-\pi_{\max}},\ldots,\delta\}}\sum_{\mathrm{I}\in\mathcal{I}}\big|\mathcal{D}(\mathrm{I},\delta',\bar{l})\big|$$

Mit Definition 2.5.6 folgt:

$$\big|\mathcal{R}(\mathrm{B}_i,\delta)\big|\leq\sum_{\delta'\in\{\delta-c_{-\pi_{\max}},\ldots,\delta\}}\big|\mathcal{A}(\delta',\bar{l})\big|$$

Eine Voraussetzung, um die Laufzeit der Prozedur `Prozessiere_Registrierte` entsprechend Satz 2.6.3 anzugeben, ist damit erfüllt. Die zweite ist ebenfalls erfüllt, da nach Terminierung der Prozedur `Prozessiere_Registrierte`$(\mathcal{R}(\mathrm{B}_i,\delta),\mathrm{B}_i,\delta)$ gilt:

$$\forall w\in\left(\bigcup_{\mathrm{R}\in\mathcal{R}(\mathrm{B}_i,\delta)}\mathrm{R}\cap\mathrm{B}_i\right):\ \delta-2\cdot c_{-\pi_{\max}}\leq\hat{l}(w)\leq\delta$$

Die zweite Ungleichung gilt nach Definition der Prozedur `Prozessiere_Registrierte`. Um die erste zu zeigen, nutze die Existenz eines Knotens $v\in\Gamma^{-}(w)$ mit $v\in\mathrm{B}$, sodass nach Aufruf von `Prozessiere_Block`$\big(\mathrm{B},\delta-c_{-\pi}\big((v,w)\big)\big)$ gilt:

$$\exists\mathrm{R}\in\mathcal{R}(\mathrm{B}_i,\delta):\ w\in\mathrm{R}$$

Für diesen Knoten v gilt nach Satz 2.2.5:

$$\bar{l}(v)+c_{-\pi}\big((v,w)\big)=\delta.$$

Aus $\hat{l}(w)<\delta-2\cdot c_{-\pi_{\max}}$ folgt dann:

$$\hat{l}(w)+c_{-\pi}\big((w,v)\big)\leq\hat{l}(w)+2\cdot c_{-\pi_{\max}}-c_{-\pi}\big((w,v)\big)<\delta-c_{-\pi}\big((w,v)\big)=\bar{l}(v)$$

Dies ist ein Widerspruch zur Korrektheit der Prozedur `Suche_Pfad`, die in Abschnitt 2.2.1 bewiesen wurde.
Wegen

$$\left|\mathcal{A}(\delta,\hat{l})\right| \leq \left|\mathcal{A}(\delta,\bar{l})\right|$$

ist die Laufzeit von `Prozessiere_Registrierte`$(\mathcal{R}(\mathrm{B}_i,\delta),\mathrm{B}_i,\delta)$ nach Satz 2.6.3 beschränkt durch:

$$\sum_{\substack{\delta' \in \{\delta-2c_{-\pi_{\max}},\ldots,\delta\}:\\ \mathcal{A}(\delta',\bar{l})\neq\emptyset}} \left(\left|\mathcal{I}\right| \cdot \mathcal{O}\left(\log\left|\mathcal{A}(\delta',\bar{l})\right|\right) + \left|\mathcal{A}(\delta',\bar{l})\right| \cdot \mathcal{O}\left(\log\left|\mathcal{I}\right|\right)\right) \tag{2.7.3}$$

Aufruf von `Prozessiere_Block`

Entsprechend Satz 2.5.8 ist die Laufzeit von `Prozessiere_Block`(B_i,δ) mit Satz 2.2.5 beschränkt durch:

$$\sum_{\delta' \in \{\delta-c_{-\pi_{\max}},\ldots,\delta+c_{-\pi_{\max}}\}} \left|\Gamma^+(\mathrm{B}_i)\right| \cdot \left|\mathcal{A}(\delta',\bar{l})\right| \cdot \mathcal{O}\left(\log\left|\mathcal{I}\right|\right) \tag{2.7.4}$$

Gesamtlaufzeit Abschnitt ②

Aufsummieren von (2.7.2), (2.7.3) und (2.7.4) ergibt:

$$\sum_{\substack{\delta' \in \{\delta_i-2\cdot c_{-\pi_{\max}},\ldots,\delta_i+c_{-\pi_{\max}}\}:\\ \mathcal{A}(\delta',\bar{l})\neq\emptyset}} \left|\Gamma^+(\mathrm{B}_i)\right| \cdot \mathcal{O}\left(\left|\mathcal{I}\right| \cdot \log\left|\mathcal{A}(\delta',\bar{l})\right| + \left|\mathcal{A}(\delta',\bar{l})\right| \cdot \log\left|\mathcal{I}\right|\right)$$

Alle n Iterationen von Abschnitt ② sind deshalb realisierbar in Zeit:

$$\sum_{\substack{\delta' \in \{\delta_1,\ldots,\delta_n+c_{-\pi_{\max}}\}:\\ \mathcal{A}(\delta',\bar{l})\neq\emptyset}} c_{-\pi_{\max}} \cdot \left|\mathrm{E}(\mathcal{B})\right| \cdot \mathcal{O}\left(\left|\mathcal{I}\right| \cdot \log\left|\mathcal{A}(\delta',\bar{l})\right| + \left|\mathcal{A}(\delta',\bar{l})\right| \cdot \log\left|\mathcal{I}\right|\right) \tag{2.7.5}$$

Gesamtlaufzeit Abschnitt ①

Wie in Abschnitt 2.5.2 begründet, wird von den Mengen $\mathcal{R}(\mathrm{B},\delta)$ mit $\mathrm{B} \in \mathcal{B}, \Delta \in \mathbb{Z}_{\geq 0}$ nur eine Teilmenge benötigt, deren Größe durch $c_{-\pi_{\max}} \cdot |\mathcal{B}|$ beschränkt ist. Das Aufrufen von `Label_Intervall`(∞,w,J) kann für ein Intervall $\mathrm{J} \in \mathcal{I}$ und einen Knoten $w \in \mathrm{J}$ entfallen. Die Laufzeit von Abschnitt ① ist deshalb beschränkt durch:

$$c_{-\pi_{\max}} \cdot |\mathcal{B}| + |\mathrm{S}| \cdot \mathcal{O}\left(\log\left|\mathcal{I}\right|\right) \tag{2.7.6}$$

Zusammenfassung der Laufzeiten

Aufsummieren von (2.7.5) und (2.7.6) ergibt mit $|S| \leq |\mathcal{I}|$ nach Definition der Intervallmenge $\mathcal{I}$ den Satz:

Satz 2.7.2:
Die Prozedur `Suche_Pfad`(S, T) *kann in folgender Zeit realisiert werden:*

$$c_{-\pi_{\max}} \cdot |\mathcal{B}| + \sum_{\substack{\delta' \in \{\delta_1, \ldots, \delta_n + c_{-\pi_{\max}}\}: \\ \mathcal{A}(\delta', \bar{l}) \neq \emptyset}} c_{-\pi_{\max}} \cdot |E(\mathcal{B})| \cdot \mathcal{O}\left(|\mathcal{I}| \cdot \log|\mathcal{A}(\delta', \bar{l})| + |\mathcal{A}(\delta', \bar{l})| \cdot \log|\mathcal{I}|\right)$$

2.7.3 Laufzeitvergleich mit Hetzel [1995]

Die in Satz 2.7.2 genannte Laufzeitschranke für die Prozedur `Suche_Pfad` wird nun mit der in Hetzel [1995] angegebenen verglichen. In Hetzel [1995] erfolgt die Laufzeitanalyse mit Benutzung der in Abschnitt 1.3.1 vorgestellten Funktion π_H als *future cost*. Somit ist die Anzahl der Blöcke entsprechend den Ausführungen in Abschnitt 1.4.3 durch die Anzahl der Ebenen des Chips gegeben und kann deshalb als Konstante betrachtet werden. Außerdem beschränkt sich die Laufzeitanalyse in Hetzel [1995] auf $|T| = 1$ bzw. eine Intervallmenge $\mathcal{I}$, sodass $\alpha_I \neq 0$ für alle $I \in \mathcal{I}$ gilt. Also gilt $|\mathcal{A}(\delta, \bar{l})| \leq |\mathcal{I}|$ für alle $\delta \in \{\delta_1, \ldots, \delta_n + c_{-\pi_{\max}}\}$. Da $c_{-\pi_{\max}} \leq 2 \cdot \max\left\{c(e) \mid e \in E(\mathcal{G}')\right\}$ und dieses Maximum ebenfalls als Konstante für einen Chip betrachtet werden kann, ist die Laufzeit der Prozedur `Suche_Pfad` beschränkt durch:

$$\sum_{\delta' \in \{\delta_1, \ldots, \delta_n\}} \mathcal{O}\left(|\mathcal{I}| \cdot \log|\mathcal{I}|\right)$$

Außerdem kann mit Bemerkung 2.6.4 eine andere Laufzeitschranke angegeben werden. Nach dieser Bemerkung und Satz 2.2.5 kann der Term

$$|\mathcal{I}| \cdot \log|\mathcal{A}(\delta', \bar{l})|$$

der Summe in Satz 2.7.2 anders angegeben werden. Mit den Abschätzungen

$$|\mathcal{I}| \leq |V(\mathcal{G}')| \quad \text{und} \quad \sum_{\substack{\delta' \in \{\delta_1, \ldots, \delta_n + c_{-\pi_{\max}}\}: \\ \mathcal{A}(\delta', \bar{l}) \neq \emptyset}} |\mathcal{A}(\delta', \bar{l})| \leq |V(\mathcal{G}')|$$

ergibt sich neben Satz 2.7.2 die folgende Laufzeitschranke:

$$|V(\mathcal{G}')| \cdot \mathcal{O}\left(\log|V(\mathcal{G}')|\right)$$

In Hetzel [1995] wird die Laufzeitschranke

$$\min\left\{\mathcal{O}\left(U \cdot |\mathcal{I}| \cdot \log|\mathcal{I}|\right), |V(\mathcal{G}')| \cdot \mathcal{O}\left(\log|V(\mathcal{G}')|\right)\right\}$$

angegeben, wobei $U := \delta_n - \delta_1 + 1$. Die in Satz 2.7.2 genannte Laufzeitschranke der Prozedur `Suche_Pfad` ist also eine Erweiterung im Vergleich zu Hetzel [1995], da die Einschränkung $|T| = 1$ bzw. $\alpha_I \neq 0$ für alle $I \in \mathcal{I}$ nicht vorausgesetzt wird und die Funktion π nicht auf die in Abschnitt 1.3.1 vorgestellte Funktion π_H beschränkt ist.

2.8 Visualisierung einer Pfadsuche

Die nachfolgenden Abbildungen zeigen einen Ausschnitt einer Pfadsuche, die mit der Prozedur `Suche_Pfad` auf einem anwendungsspezifischen Chip der 45 nm Technologie durchgeführt wurde. Für eine Verdrahtungsebene $z \in Z$ ist die Menge der Track-Koordinaten so definiert, dass für die Kosten einer Kanten $e \in E(\mathcal{G}'[V_z])$ entgegen der Vorzugsrichtung gilt:

$z = 0:$	$c(e) = 140$
$z = 1:$	$c(e) = 140$
$z = 2:$	$c(e) = 190$

Die Abbildungen zeigen drei Verdrahtungsebenen. Die Menge der Startknoten S ist blau und die Menge der Zielknoten T ist rot hinterlegt. Jedes Intervall $I \in \mathcal{I}$ ist durch ein beige farbenes Rechteck dargestellt. Ein Intervall I wird erst initialisiert, wenn ein Knoten $v \in I$ gelabelt wird. Daher sind ebenfalls uninitialisierte Intervalle abgebildet, die ggfs. bei Initialisierung unterteilt werden. Der kürzester S-T-Pfad, der bestimmt wurde, ist gelb unterlegt. Die Intervallmenge $\mathcal{I}$ genügt neben den Bedingungen in Definition 2.1.1 weiteren Eigenschaften, die innerhalb von BonnRoute® definiert sind. Daher kann es insbesondere mehrere Intervalle geben, die nur einen Knoten enthalten.

Als *future cost* wurden die Abschnitt 1.3.1 vorgestellten Funktionen π_H bzw. π_P verwendet. Für das Beispiel gilt:

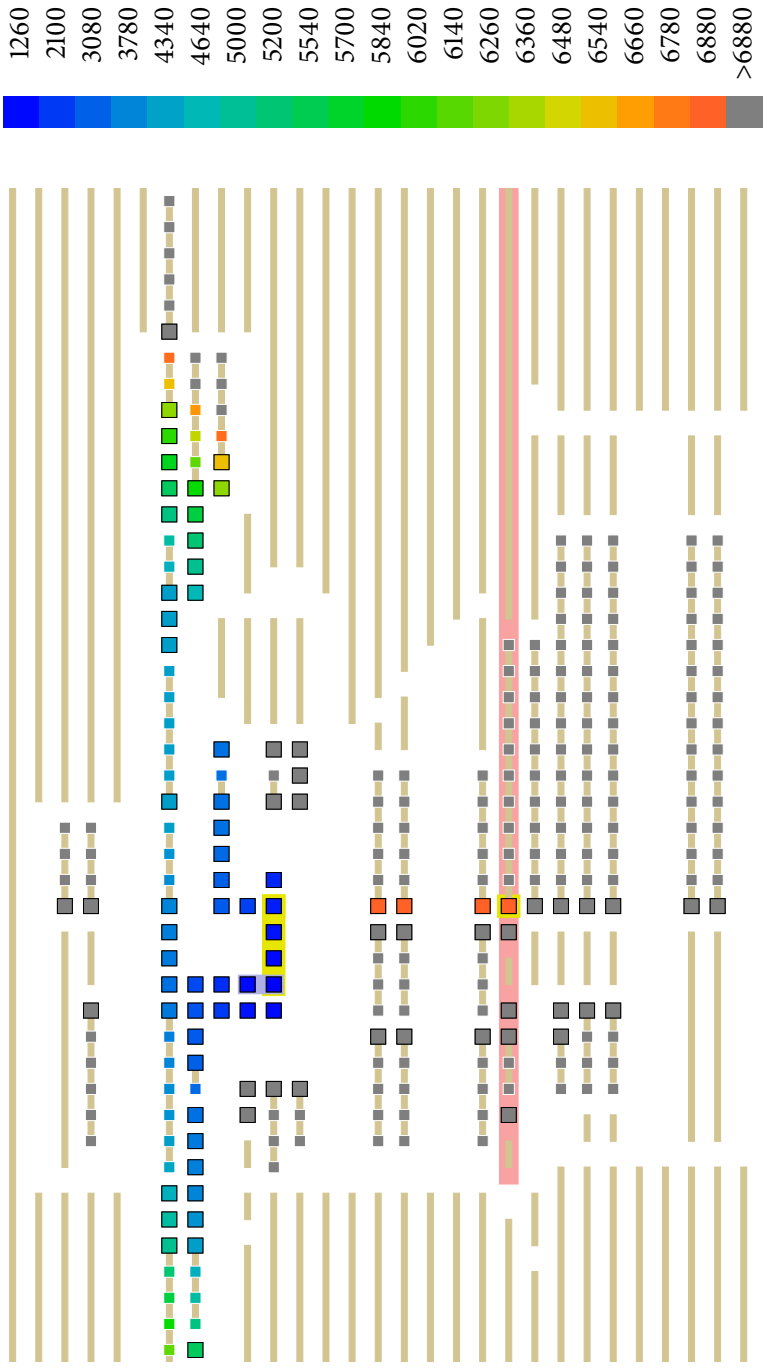
$$\min\left\{\pi_H(v) \;\middle|\; v \in S\right\} = 1260 \qquad \min\left\{\pi_P(v) \;\middle|\; v \in S\right\} = 2100$$

Die Länge eines kürzesten S-T-Pfades bzgl. c beträgt 6880 Einheiten.

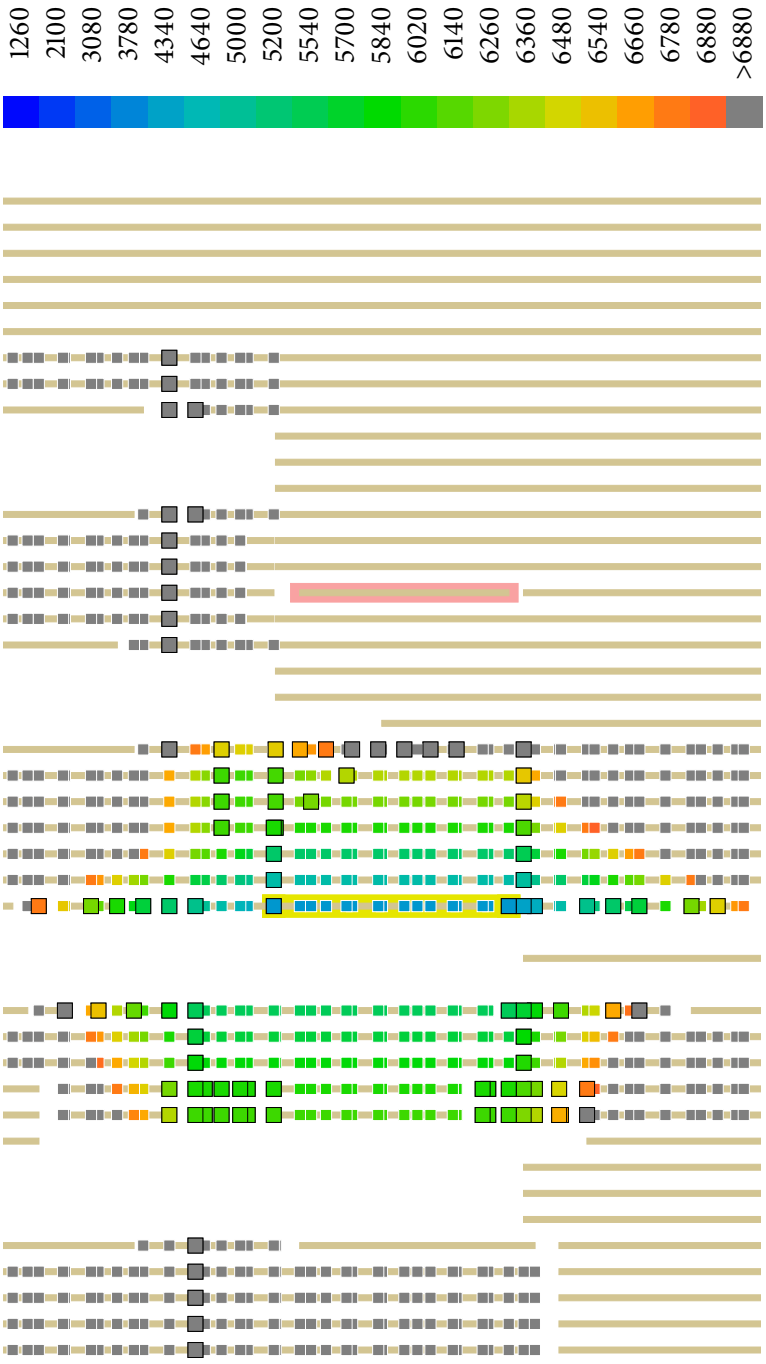
Um die Darstellung des Labelwertes $l(v)$ für einen Knoten $v \in V(\mathcal{G}')$ zu erklären, betrachte eine Verdrahtungsebene $z \in Z$ mit horizontaler [⇆] Vorzugsrichtung. Für einen Knoten $v \in V_z$ ist der Labelwert $l(v)$ durch ein Quadrat dargestellt, dessen Farbe entsprechend der Legende gewählt ist. Falls $\left(l(v), v_1\right) \in \Delta_{\mathcal{L}(I)}$ für das Intervall $I \in \mathcal{I}(z)$ mit $v \in I$, ist das Quadrat groß gewählt, andernfalls ist es klein gewählt. Innerhalb von BonnRoute® werden also nur die großen Quadrate abgespeichert.

Zu beachten ist, dass die gezeigte Instanz recht einfach gewählt ist. Allerdings ist eine derart große Menge an Zielknoten nicht unüblich, da oftmals an bereits gelegte Drähte angeschlossen werden muss. Außerdem wird deutlich, dass die Verwendung der *future cost* π_P weniger Labeloperationen impliziert.

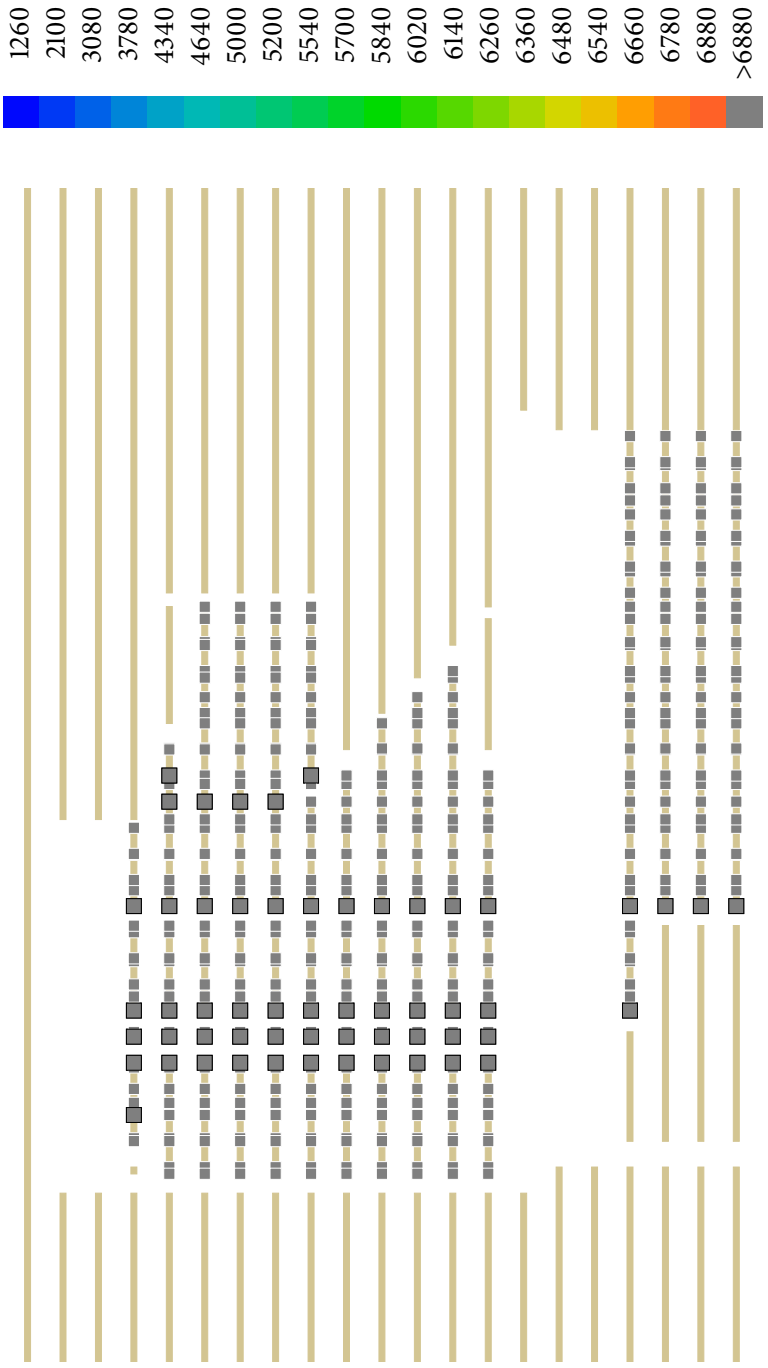
Die Laufzeit zur Verdrahtung des ganzen Chips kann nicht angegeben werden, da Anpassungen anderer Programmteile von BonnRoute® an die 45nm Technologie noch nicht abgeschlossen sind.



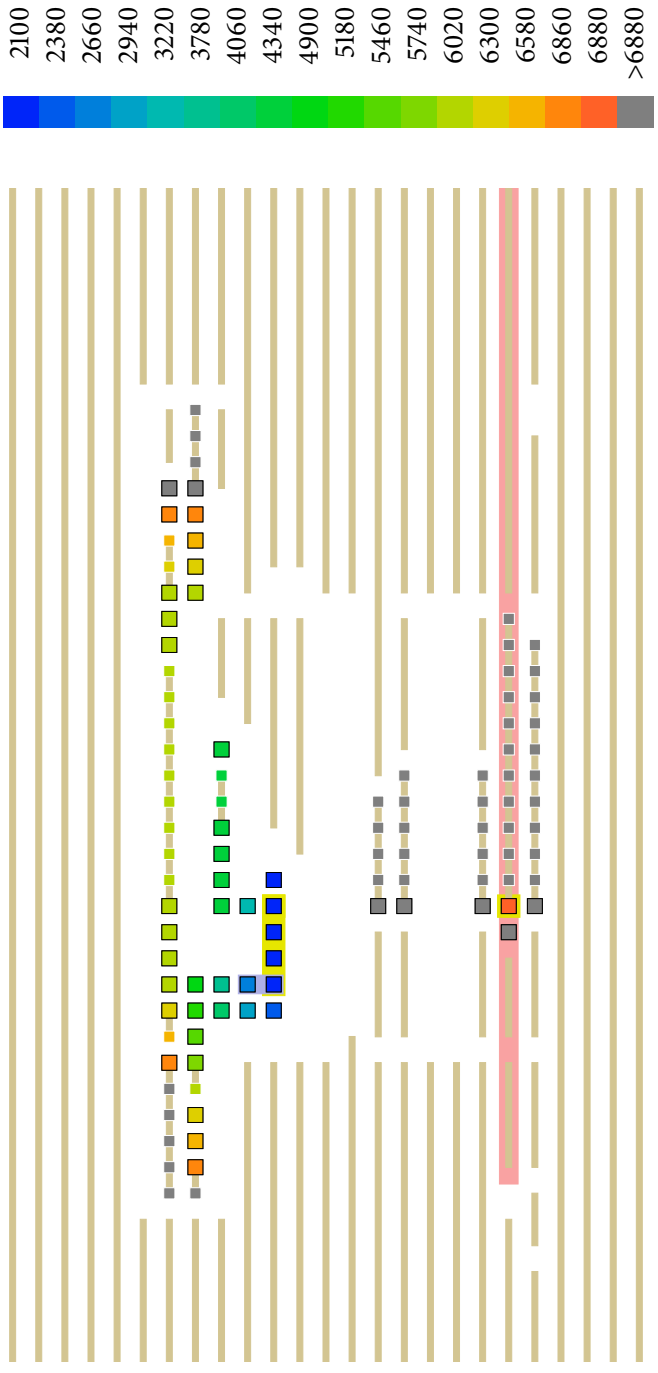
Ausschnitt Ebene 0 mit *future cost* $\pi = \pi_H$



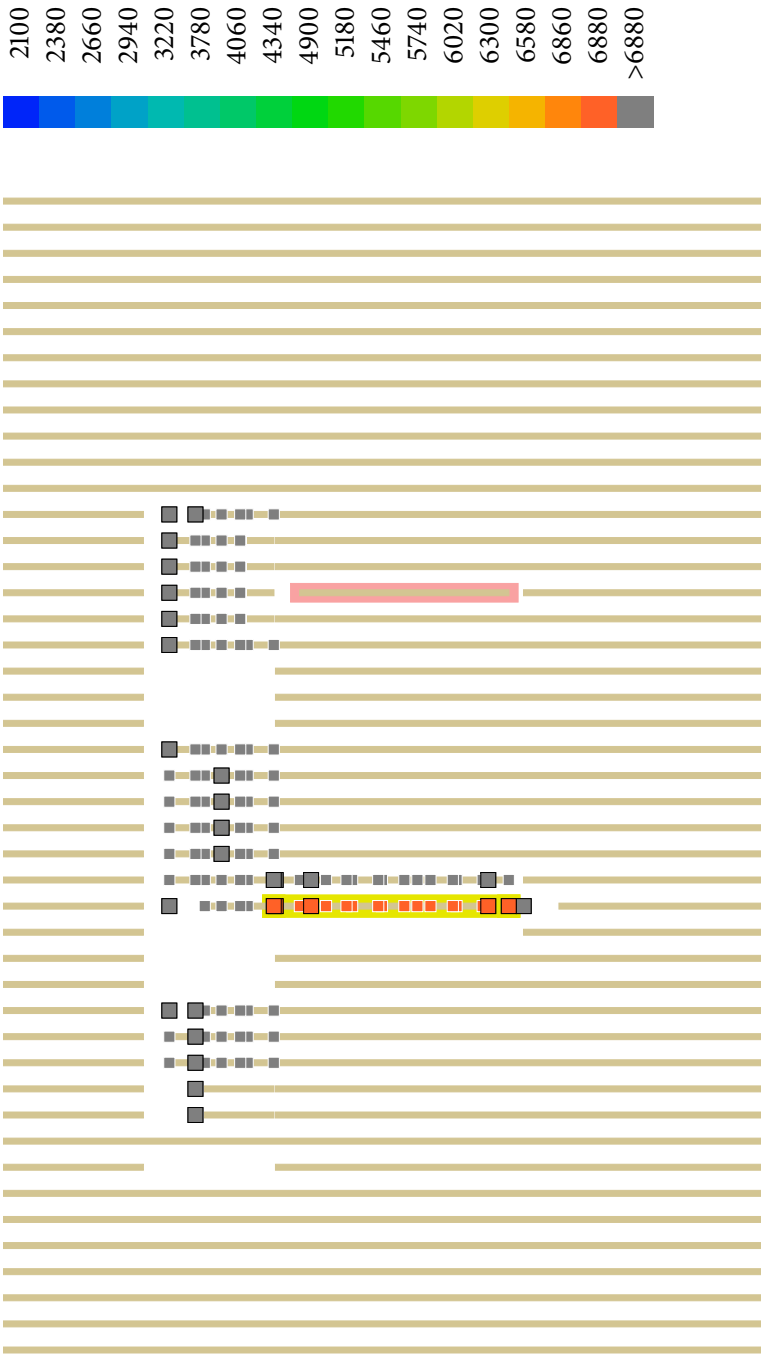
Ausschnitt Ebene 1 mit *future cost* $\pi = \pi_H$



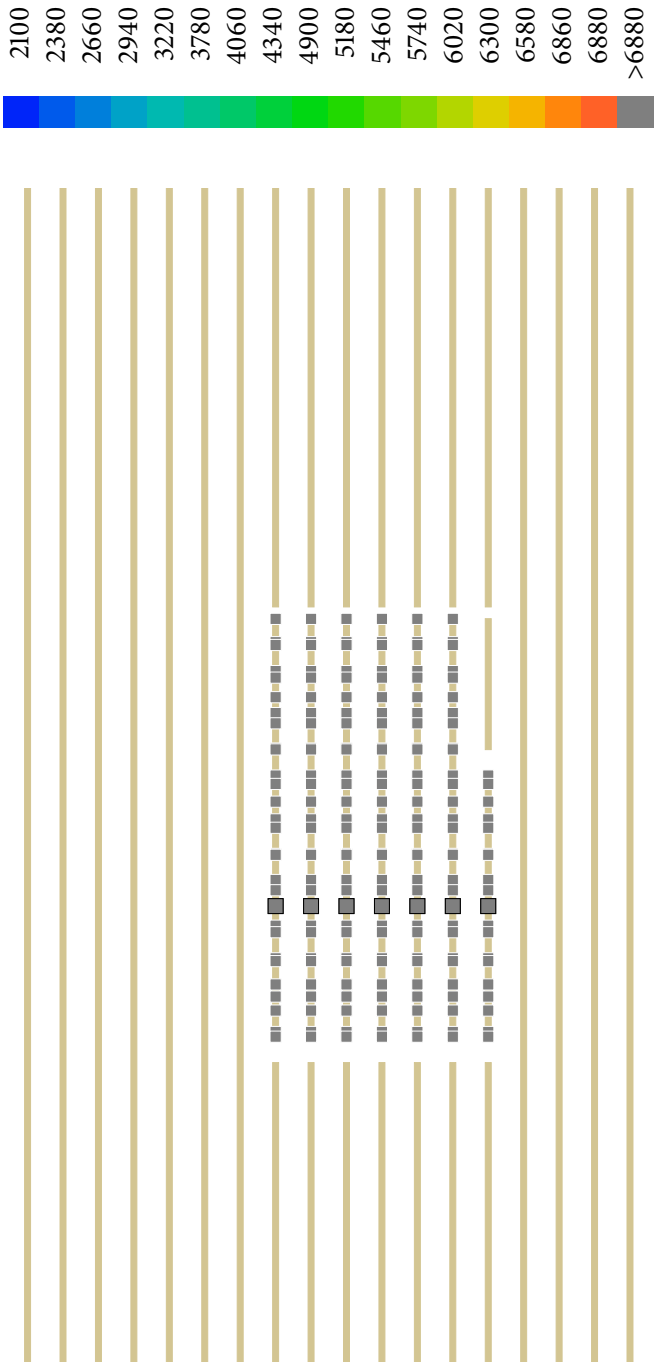
Ausschnitt Ebene 2 mit *future cost* $\pi = \pi_H$



Ausschnitt Ebene 0 mit *future cost* $\pi = \pi_P$



Ausschnitt Ebene 1 mit *future cost* $\pi = \pi_P$



Ausschnitt Ebene 2 mit *future cost* $\pi = \pi_P$

Literaturverzeichnis

Albrecht, C. [2001]: Zwei kombinatorische Optimierungsprobleme im VLSI-Design: Optimierung der Zykluszeit und der Slackverteilung und globale Verdrahtung. Dissertation, Forschungsinstitut für Diskrete Mathematik, Universität Bonn, 2001.

Dijkstra, E. W. [1959]: A note on two problems in connexion with graphs. *Numerische Mathematik 1*, Seiten 269–271, 1959.

Goldberg, A. V. und Harrelson, C. [2005]: Computing the Shortest Path: A* Search Meets Graph Theory. *Proceedings of 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'05)*, Seiten 156–165, 2005.

Hart, P. E., Nilsson, N. J. und Raphael, B. [1968]: A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, SSC 4, Seiten 100–107, 1968.

Hetzel, A. [1995]: Verdrahtung im VLSI-Design: Spezielle Teilprobleme und ein sequentielles Lösungsverfahren. Dissertation, Forschungsinstitut für Diskrete Mathematik, Universität Bonn, 1995.

Hetzel, A. [1998]: A Sequential Detailed Router for Huge Grid Graphs. *Proceedings of Design, Automation and Test in Europe (DATE 1998)*, Seiten 332–339, 1998.

Korte, B., Rautenbach, D. und Vygen, J. [2007]: BonnTools: Mathematical innovation for layout and timing closure of systems on a chip. *Proceedings of the IEEE 95*, Seiten 555–572, 2007.

Korte, B. und Vygen, J. [2008]: *Combinatorial Optimization (4th. Edition)*. Springer, 2008.

Müller, D. [2002]: Bestimmung der Verdrahtungskapazitäten im Global Routing von VLSI-Chips. Diplomarbeit, Forschungsinstitut für Diskrete Mathematik, Universität Bonn, 2002.

Müller, D. und Vygen, J. [2008]: Faster min-max resource sharing and applications. Technischer Report Nr. 08987, Forschungsinstitut für Diskrete Mathematik, Universität Bonn, 2008.

Peyer, S. [2007]: Shortest Path and Steiner Trees in VLSI Routing. Dissertation, Forschungsinstitut für Diskrete Mathematik, Universität Bonn, 2007.

Peyer, S., Rautenbach, D. und Vygen, J. [2006]: A Generalization of Dijkstra's Shortest Path Algorithm with Applications to VLSI Routing. Erscheint in *Journal of Discrete Algorithms*.

Rohe, A. [2001]: Sequential and Parallel Algorithms for Local Routing. Dissertation, Forschungsinstitut für Diskrete Mathematik, Universität Bonn, 2001.

Vygen, J. [2001]: Theory of VLSI Layout. Habilitationsschrift, Forschungsinstitut für Diskrete Mathematik, Universität Bonn, 2001.

Xing, Z., und Kao, R. [2002]: Shortest Path Search Using Tiles and Piecewise Linear Cost Propagation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21, Seiten 145–158, 2002.