Evolutionäre Optimierung von Sortiernetzwerken

Florian Forster

24. November 2009

Zusammenfassung

Sortiernetzwerke werden eingeführt und einige bekannte Konstruktionen werden vorgestellt (Off-Even-Transposition, Bitonic-Merge, Odd-Even-Merge, Pairwise). Transformationsmöglichkeiten für Sortiernetzwerke werden besprochen. Evolutionäre Algorithmen werden beschrieben und ein evolutionärer Algorithmus für die Optimierung von Sortiernetzwerken wird angegeben. Die mindestens von diesem Algorithmus erreichte Güte wird angegeben und die Transformation zu einer Markov-Kette wird gezeigt. *Natürlich: So fern ich das hinbekomme bzw. Recht behalte.*

Inhaltsverzeichnis

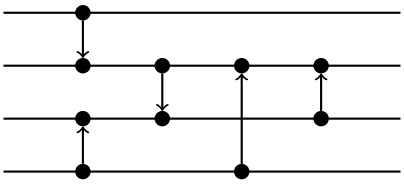


Abbildung 1: Einfaches Komparatornetzwerk mit vier Ein- bzw. Ausgängen, bestehend aus 5 Komparatoren.

1 Motivation und Einleitung

1.1 Motivation

- Sortiernetzwerke sind toll, weil ...
- Sortiernetzwerke sind einfach erklärt, aber trotzdem kompliziert.
- Bisher noch kein evolutionärer Algorithmus zur automatischen Optimierung von Sortiernetzwerken bekannt. *(Glaube ich zumindest.)*

1.2 Einleitung

1.2.1 Sortiernetzwerke

Komparatoren sind die Bausteine, die *Sortiernetzwerken* zugrunde liegen. Sie haben zwei Eingänge über die sie zwei Zahlen erhalten können. Ausserdem besitzt ein *Komparator* zwei Ausgänge, die im Gegensatz zu den Eingängen unterscheidbar sind: Die größere der beiden Zahlen wird immer auf dem einen, die kleinere der beiden Zahlen immer auf dem anderen Ausgang ausgegeben.

Wenn man nun mehrere *Komparatoren* miteinander kombiniert, also die Ausgänge von *Komparatoren* mit dem Eingängen anderer *Komparatoren* verbindet, erhält man ein *Komparatornetzwerk*.

Abbildung 1 zeigt ein einfaches Komparatornetzwerk aus fünf Komparatoren in der üblichen Darstellungsweise: Die horizontalen Linien stellen Leitungen von den Eingängen auf der linken Seite zu den Ausgängen auf er rechten Seite dar. Die vertikalen Pfeile symbolisieren die Komparatoren, die die Werte „auf den Leitungen“ vergleichen und ggf. vertauschen. Nach einem Komparator befindet sich die kleinere Zahl immer auf der Leitung, auf die der Pfeil zeigt, die größere Zahl befindet sich auf der Leitung auf der der Pfeil seinen Ursprung hat.

Komparatornetzwerke, die für jede beliebige Eingabepermutation eine Ausgabe erzeugen, die der Sortierung der Eingabe entspricht, heißen *Sortiernetzwerke*. Das in Abbildung 1 gezeigte Komparatornetzwerk ist kein Sotiernetzwerk: Die Eingabefolge $(1, 2, 3, 4)$ würde zur Ausgabe $(2, 1, 3, 4)$ führen – die bestehenden Sortierung wird also sogar zerstört.

Zu beweisen, dass ein gegebenes Komparatornetzwerk die Sortiereigenschaft *nicht* hat, ist mit einem gegebenen Gegenbeispiel also einfach möglich. Dieses Gegenbeispiel zu finden ist allerdings aufwendig.

TODO: Wie findet man die Gegenbeispiele? Die *Entscheidung*, ob ein Netzwerk sortiert, ist doch NP-vollständig, also müsste doch das Finden eines Gegenbeispiels im Allgemeinen auch exponentialle Laufzeit haben..? **TODO:** Wenn die *Entscheidung*, ob ein Netzwerk sortiert, NP-vollständig ist, müsse man dann nicht einen Zeugen für die Sortiereigenschaft angeben können?

TODO: $0-1$-Prinzip

Um zu überprüfen, ob ein gegebenes Komparatornetzwerk die Sortiereigenschaft besetzt, müssen nicht alle $n!$ Permutationen von n unterschiedlichen Zahlen ausprobieren. Stattdessen reicht es zu überprüfen, dass das Netzwerk alle 2^n $0-1$-Folgen sortiert.

Sortiernetzwerke:

- Ein Komparator-Netzwerk ist ...
- Ein Komparator-Netzwerk ist ein Sortiernetzwerk, wenn ...
- Die Frage nach der Sortiereigenschaft ist NP-vollständig.

1.2.2 Evolutionäre Algorithmen

Viele *kombinatorische Optimierungsprobleme* sind schwer zu lösen – die entsprechenden Entscheidungsprobleme liegen oft in der Komplexitätsklasse NP, sind also mit bekannten Verfahren nicht effizient exakt lösbar. Sollte sich herausstellen, dass diese Probleme nicht in der Komplexitätsklasse P liegen, wäre eine Konsequenz, dass es effiziente exakte Algorithmen für diese Probleme nicht geben kann. Falls sich hingegen herausstellt, dass diese Probleme in der Komplexitätsklasse P liegen, wird es mit großer Wahrscheinlichkeit noch einige Zeit dauern bis auch Algorithmen mit praktikablen Zeitkonstanten gefunden werden.

Aus diesem Grund besteht die Notwendigkeit einen Kompromiss einzugehen: Statt die bzw. eine der *optimalen* Lösungen als einzige Ausgabe des Algorithmus zuzulassen, wird eine „möglichst gute“ Lösung ausgegeben. Viele dieser Optimierungsalgorithmen orientieren sich an Vorgängen in der Natur, beispielsweise immitieren die „Ameisenalgorithmen“ das Verhalten von Ameisen auf der Futtersuche um kurze Rundreisen auf Graphen zu berechnen.

Bei *Evolutionären Algorithmen* stand die Evolution pate. Die Grundidee ist es, bestehende Lösungen zu neuen, unter Umständen besseren Lösungen zu kombinieren. Dabei bedient man sich der in der Evolutionstheorie etablierten Nomenklatur, beispielsweise werden konkrete Lösungen für ein Problem häufig als *Individuum* bezeichnet.

Die Vorgehensweise lässt sich abstrakt wie folgt beschreiben. Aus einer bestehenden Lösungsmenge, der *Population* werden zufällig Lösungen ausgesucht *(Selektion)* und zu einer neuen Lösung kombiniert (*Rekombination*). Unter Umständen wird die neue Lösung noch zufällig verändert *(Mutation)*, bevor sie in die bestehende Lösungsmenge integriert wird. Die Wahrscheinlichkeiten, beispielsweise bei der *Selektion*, sind dabei nicht zwangsläufig gleichverteilt – üblicherweise werden bessere Lösungen bevorzugt. Zur Bewertung die die sogenannte *Gütefunktion*.

Nicht alle Probleme eignen sich für diese Strategie: Zum einen muss es möglich sein, eine initiale Population zur Verfügung zu stellen, da diese als Basis aller weiteren Operationen dient. Das ist häufig keine große Einschränkung, da es oft einfach ist *irgendeine* Lösung anzugeben. Zum anderen muss eine Methode für die Rekombination existieren. Das insbesondere dann problematisch wenn *Nebenbedingungen* eingehalten werden müssen.

- Unter einem „Evolutionären Algorithmus“ versteht man ...
- Da die Sortiereigenschaft zu überprüfen NP-schwer ist, ist die Mutation *(vermutlich)* nicht (effizient) möglich.

2 Bekannte konstruktive Sortiernetzwerke

Übersicht über bekannte konstruktive Sortiernetzwerke.

2.1 Odd-Even-Transpositionsort

Das Sortiernetzwerk *Odd-Even-Transpositionsort* (OET) ist eines der einfachsten Sortiernetzwerke. Es besteht aus n *Schichten*, die jede „Leitung“ abwechselnd mit den benachbarten Leitungen verbindet. Abbildung 2 zeigt das OET-Netzwerk für $n = 8$ Leitungen.

2.2 Batcher's Mergesort

Ein Netzwerk von K. E. Batcher. Siehe: K.E. Batcher: Sorting Networks and their Applications. Proc. AFIPS Spring Joint Comput. Conf., Vol. 32, 307-314 (1968) **TODO:** Bibtex!

2.2.1 Der bitone Mischer

Das Netzwerk basiert auf dem *bitonen Mischer*, einem Komparator-Netzwerk, das eine beliebige bitone Folge in eine sortierte Listen umordnen kann. Eine *bitone Folge* ist eine monoton steigende Folge gefolgt von einer monoton fallenden Folge, oder ein zyklischer Shift davon. Abbildung 3 zeigt die vier prinipiellen Möglichkeiten die durch zyklische Shifts entstehen können. Die wichtigsten Varianten für Batcher's Mergesort-Netzwerk zeigen die Abbildungen 3(a) und 3(b). Sie erhält man, wenn man eine aufsteigend und eine absteigend sortierte Liste aneinanderhängt. Bei den anderen beiden Formen ist wichtig zu beachten, dass das letzte Element nicht größer (Abbildung 3(c)) bzw. kleiner (Abbildung 3(d)) als das erste Element der Folge sein darf.

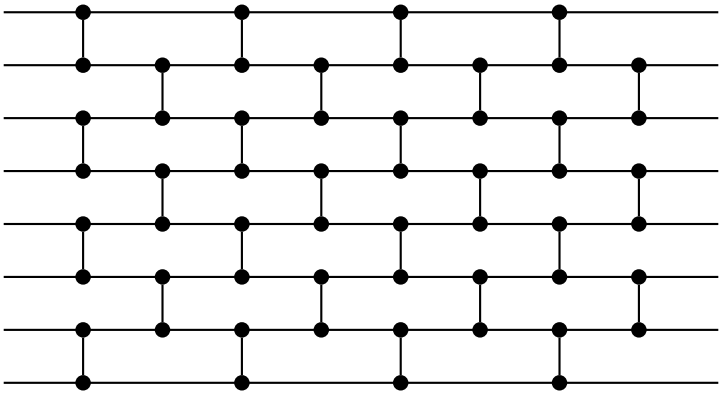
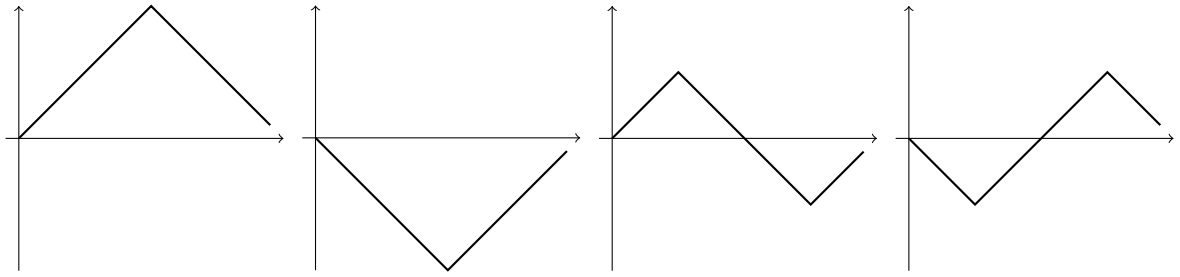
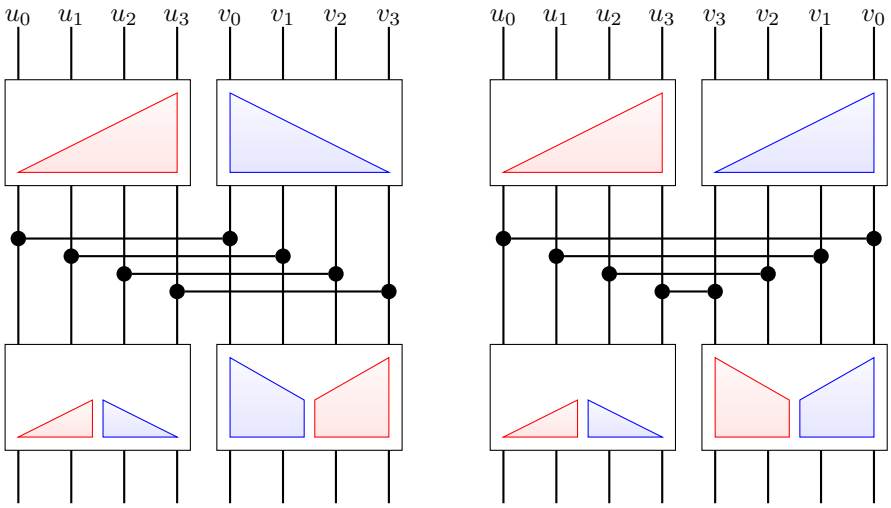


Abbildung 2: Das *Odd-Even-Transpositionsort* Netzwerk für acht Eingänge.



(a) aufsteigend, absteigend (b) absteigend, aufsteigend (c) aufsteigend, absteigend, aufsteigend (d) absteigend, aufsteigend, absteigend

Abbildung 3: Beispiele bitoner Folgen.



(a) normal (b) trichter

Abbildung 4: Schematischer Aufbau des bitonen Mischers: Jedes Element der aufsteigenden Folge $u_0, u_1, \ldots$ wird mit dem entsprechenden Element der absteigend sortierten Folge $v_0, v_1, \ldots$ verglichen. Die beiden resultierenden Teilfolgen sind wiederum biton.

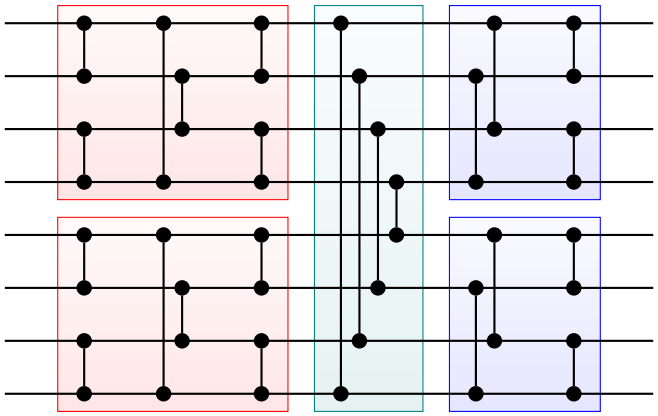


Abbildung 5: $S(8)$, Batcher's *bitone Mergesort-Netzwerk* für acht Eingänge. Markiert sind die beiden Instanzen von $S(4)$ (rot), die beiden bitonen Mischer $M(4)$ (blau) und die Komparatoren, die im letzten rekursiven Schritt hinzugefügt wurden (grün).

Der Mischer funktioniert folgendermaßen: Gegeben sind zwei Folgen mit je $m = \frac{n}{2}$ Elementen, $U = (u_0, u_1, \ldots, u_{m-1})$ und $V = (v_0, v_1, \ldots, v_{m-1})$. Die Folge U sei aufsteigend sortiert, die Folge V sei absteigend sortiert:

$$\begin{aligned} u_0 \leqq u_1 \leqq &\quad \ldots \quad \leqq u_{m-1} \\ v_0 \geqq v_1 \geqq &\quad \ldots \quad \geqq v_{m-1} \end{aligned}$$

Im ersten Schritt werden nun jeweils die Elemente an den gleichen relativen Positionen verglichen und ggf. vertauscht:

$$u_i \longleftrightarrow v_i, \quad 0 \leqq i < m$$

Sei $j \in \{0 \ldots m\}$ der Index der ersten Elemente u_j und v_j, die durch den gemeinsamen Komparator vertauscht werden. Unter der Annahme, dass Elemente nur vertauscht werden wenn, sie ungleich sind, muss $u_j > v_j$ gelten. Mit $u_j \leqq u_{j+1}$ und $v_j \geqq v_{j+1}$ folgt daraus $u_{j+1} > v_{j+1}$. Es werden also alle Elemente u_k und v_k mit $k \geqq j$ vertauscht. $j = m$ bezeichnet den Fall, in dem das größte Element der „linken" Folge, u_{m-1}, kleiner ist als das kleinste Element der „rechten" Folge, v_{m-1}. Daraus folgt, dass die entstehende Folge aus zwei bitonen Folgen besteht, die rekursiv zusammengeführt werden können. Abbildung 4(a) zeigt die Situationen vor und nach diesem Schritt des Mischers.

Mit dem bitonen Mischer auch zwei aufsteigend sortierte Folgen sortiert werden. Dazu ist lediglich das „Umbenennen" der Leitungen notwendig. Abbildung 4(b) zeigt das Schema des bitonen Mischers für zwei aufsteigend sortierte Foglen. Durch das Umbenennen verändert sich das Muster der Komparatoren ein wenig: Statt an eine Treppe erinnert das Muster nun an einen Trichter.

2.2.2 Batcher's Bitonic-Mergesort-Netzwerk

Das Sortiernetzwerk $S(n)$ mit n Eingängen besteht aus zwei Instanzen von $S(\frac{n}{2})$, dem Netzwerk mit $\frac{n}{2}$ Eingängen und dem bitonen Mischer $M(n)$. Die Rekursion bricht bei $n = 1$ ab – eine einelementige Liste ist immer sortiert. Das konkrete Netzwerk $S(8)$ ist in Abbildung 5 zu sehen. Eingezeichnet sind ebenfalls die beiden Instanzen des Netzwerks $S(4)$ (rot) sowie der bitone Mischer $M(8)$ (blau).

2.3 Odd-Even-Mergesort

Obwohl der Name ähnlich klingt, haben *Odd-Even-Mergesort* (OEM) und *Odd-Even-Transpositionsort* (OET, siehe Abschnitt 2.1) wenig gemein. Auch dieses Netzwerk ist von K. Batcher gefunden worden und wird rekursiv durch einen „Mischer" definiert.

2.3.1 Der Odd-Even-Mischer

Der *Odd-Even-Mischer* ist ein Komperatornetzwerk, dass zwei sortierte Folgen zu einer sortierten Ausgabe zusammenfügen kann. Dabei kommt es mit weniger Vergleichen aus als der *bitone Mischer*, der im Abschnitt 2.2.1 vorgestellt wurde.

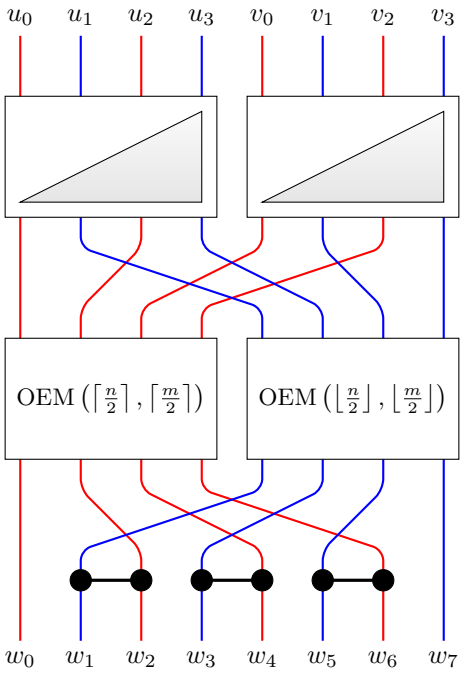


Abbildung 6: Schematischer Aufbau des *Odd-Even* Mischers. Im Vergleich zum bitonen Mischer für Acht kommt dieses Schema mit einem Komparator weniger aus. Der Effekt wird duch den rekursiven Aufbau noch verstärkt.

Der *Odd-Even-Mischer* selbst ist ebenfalls rekursiv aufgebaut: Die Eingabe für den Mischer mit $N = n + m$ Leitungen besteht aus den beiden sortierten Folgen $U = (u_0, u_1, \ldots, u_{n-1})$ und $V = (v_0, v_1, \ldots, v_{m-1})$. Die gesamte Eingabe sei $W = (w_0, w_1, \ldots, w_{N-1})$ mit:

$$w_i = \begin{cases} u_i, & i < n \\ v_{i-n}, & i \geqq n \end{cases}, \quad 0 \leqq i < N$$

Diese werden jetzt in insgesamt vier sortierte Folgen aufgeteilt, je eine Liste der geraden Indizes und je eine Liste der ungeraden Indizes.

$$\begin{aligned} U_{\text{gerade}} &= (u_0, u_2, u_4, \ldots) \\ U_{\text{ungerade}} &= (u_1, u_3, u_5, \ldots) \\ V_{\text{gerade}} &= (v_0, v_2, u_4, \ldots) \\ V_{\text{ungerade}} &= (v_1, v_3, u_5, \ldots) \end{aligned}$$

Die geraden Folgen U_{gerade} und V_{gerade} bzw. die ungeraden Folgen U_{ungerade} und V_{ungerade} werden rekursiv von kleineren *Odd-Even-Mischern* zusammengefügt, so dass sich am Ausgang der Mischer die Folgen

$$\begin{aligned} W_{\text{gerade}} &= (w_0, w_2, w_4, \ldots) \\ W_{\text{ungerade}} &= (w_1, w_3, w_5, \ldots) \end{aligned}$$

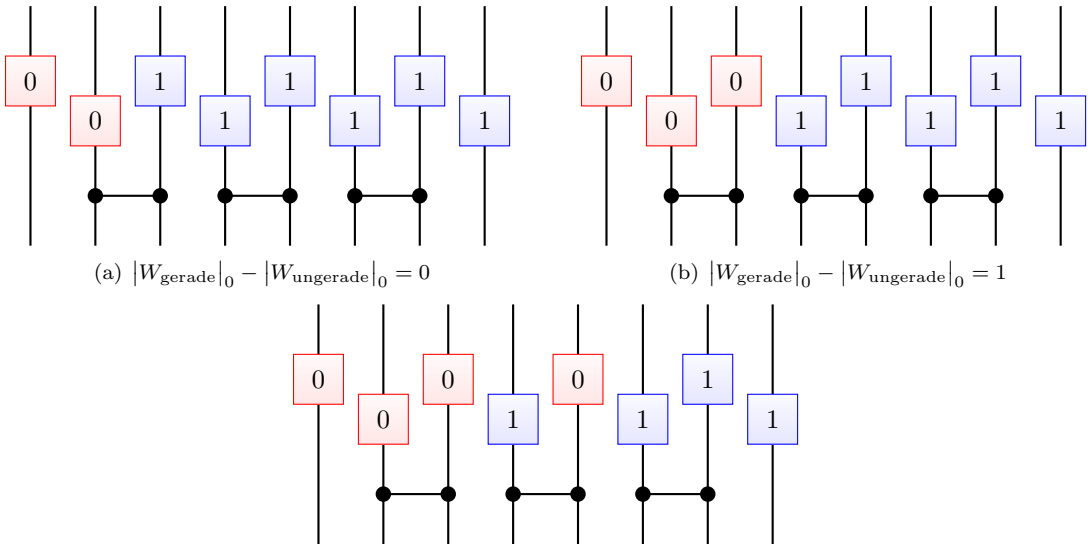
ergeben.

Anschließend werden die Komparatoren zwischen benachbarten Leitungen hinzugefügt,

$$w_{2i-1} \longleftrightarrow w_{2i}, \quad 1 \leqq i < \frac{N}{2}$$

die die Folge W sortieren. Den schematischen Aufbau des *Odd-Even-Mischers* zeigt Abbildung 6.

Leider bricht die Rekursion nicht so schön ab, wie das beim *bitonen Mischer* der Fall gewesen ist. Insbesondere für $n = m = 1$ würde – entsprechend der Konstruktionsvorschrift – ein leeres Netzwerk entstehen, was offensichtlich nicht korrekt wäre. Die Abbruchbedingungen für den rekursiven Aufbau lauten:



(a) $|W_{\text{gerade}}|_0 - |W_{\text{ungerade}}|_0 = 0$ (b) $|W_{\text{gerade}}|_0 - |W_{\text{ungerade}}|_0 = 1$

(c) $|W_{\text{gerade}}|_0 - |W_{\text{ungerade}}|_0 = 2$

Abbildung 7: Die drei Situationen, die nach dem Verzahnen der Ausgaben der kleineren *Odd-Even-Mischer* entstehen können. Ist die Differenz der Anzahl der Nullen gleich 0 oder 1, ist die Folge bereits sortiert. Im letzten Fall stellt einer der Komparatoren sicher, dass das Ergebnis sortiert ist.

- Falls $n = 0$ oder $m = 0$: Das Netzwerk ist leer.
- Falls $n = 1$ und $m = 1$: Das Netzwerk besteht aus einem einzelnen Komparator.

Dass die resultierende Folge sortiert ist, lässt sich mit dem *0-1-Prinzip* leicht zeigen: Da U und V sortiert sind, ist die Anzahl der Nullen in den geraden Teilfolgen, U_{gerade} bzw. V_{gerade}, größer oder gleich der Anzahl der Nullen in den ungeraden Teilfolgen U_{ungerade} bzw. V_{ungerade} – die Einsen verhalten sich entsprechend umgekehrt. Das trifft demnach auch auf die Folgen W_{gerade} und W_{ungerade} entsprechend zu:

$$\begin{aligned} |W_{\text{gerade}}|_0 &= |U_{\text{gerade}}|_0 + |V_{\text{gerade}}|_0 = \left\lceil \frac{1}{2} |U|_0 \right\rceil + \left\lceil \frac{1}{2} |V|_0 \right\rceil \\ |W_{\text{ungerade}}|_0 &= |U_{\text{ungerade}}|_0 + |V_{\text{ungerade}}|_0 = \left\lfloor \frac{1}{2} |U|_0 \right\rfloor + \left\lfloor \frac{1}{2} |V|_0 \right\rfloor \end{aligned}$$

Daraus folgt, dass W_{gerade} 0, 1 oder 2 Nullen mehr enthält als W_{ungerade}. In den ersten beiden Fällen ist die „verzahnte“ Ausgabe der beiden kleineren Mischer bereits sortiert. Nur im letzten Fall, wenn W_{gerade} 2 Nullen mehr enthählt als W_{ungerade}, muss eine Vertauschung stattfinden, um die Ausgabe zu sortieren. Die jeweiligen Situationen sind in Abbildung 7 dargestellt.

2.3.2 Das Odd-Even-Mergesort-Netzwerk

Auch beim *Odd-Even-Mergesort-Netzwerk* – wie beim *bitonen Mergesort-Netzwerk* – entsteht das Sortiernetzwerk aus dem *Odd-Even-Mischer* durch rekursives Anwenden auf einen Teil der Eingabe (üblicherweise die Hälfte der Leitungen) und anschließendes zusammenfügen. Abbildung 8 zeigt das Netzwerk für 8 Eingänge.

- Odd-Even-Transpositionsort
- Bitonic-Mergesort
- Odd-Even-Mergesort
- Pairwise sorting-network

3 Transformation von Sortiernetzwerken

- Komprimieren (Alle Komparatoren so früh wie möglich anwenden).
- Normalisieren (Transformation zu Standard-Sortiernetzwerken).

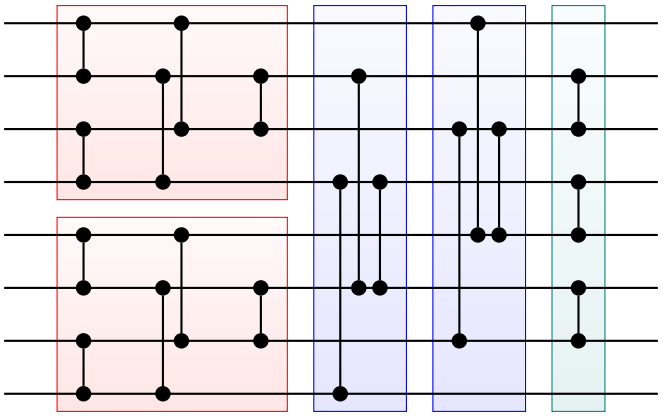


Abbildung 8: Das *Odd-Even-Mergesort-Netzwerk* für acht Eingänge.

3.1 Zwei Netzwerke kombinieren

- Mit dem Bitonic-Merge
- Mit dem Odd-Even-Merge
- Nach dem Pairwise sorting-network Schema.

3.2 Leitungen entfernen

Man kann ein gegebenes Sortiernetzwerk mit n Eingängen auf ein Sortiernetzwerk mit $(n-1)$ Leitungen verkleinern, indem man eine Leitung entfernt. Zunächst wird angenommen, dass das Minimum oder das Maximum an einem der Eingänge anliegt. Der Weg durch das Netzwerk zum entsprechenden Ausgang ist dadurch fest vorgegeben, insbesondere welche Komparatoren dafür sorgen, dass die Leitung gewechselt wird und welche nicht. Abbildung 9(a) zeigt den Weg eines Maximums durch das *Odd-Even-Transpositionsort-Netzwerk*.

Im nächsten Schritt werden alle beteiligten Komparatoren gelöscht bzw. ersetzt: Komparatoren, die *nicht* zu einem Wechsel der Leitung geführt haben, werden ersatzlos gelöscht. Diese Komparatoren sind in Abbildung 9(a) grün markiert. Die Komparatoren, die zum Wechsel der Leitung geführt haben, werden durch sich kreuzende Leitungen ersetzt. Das Resultat zeigt Abbildung 9(b). Wenn man die Maximum-Leitung entfernt (Abbildung 9(c)), erhält man ein Sortiernetzwerk für $(n-1)$ Leitungen.

Die letzte Abbildung, 9(d), zeigt das Sortiernetzwerk wieder mit den üblichen geraden Leitungen und die rot markierten Komparatoren wurden verschoben, so dass sich eine kompaktere Darstellung ergibt. Ausserdem ist das *Odd-Even-Transpositionsort-Netzwerk* für sieben Werte markiert. Der zusätzliche Komparator vor dem OET(7) hat keinen Einfluss auf die Ausgabe und kann entfernt werden.

- Min-Richtung
- Max-Richtung

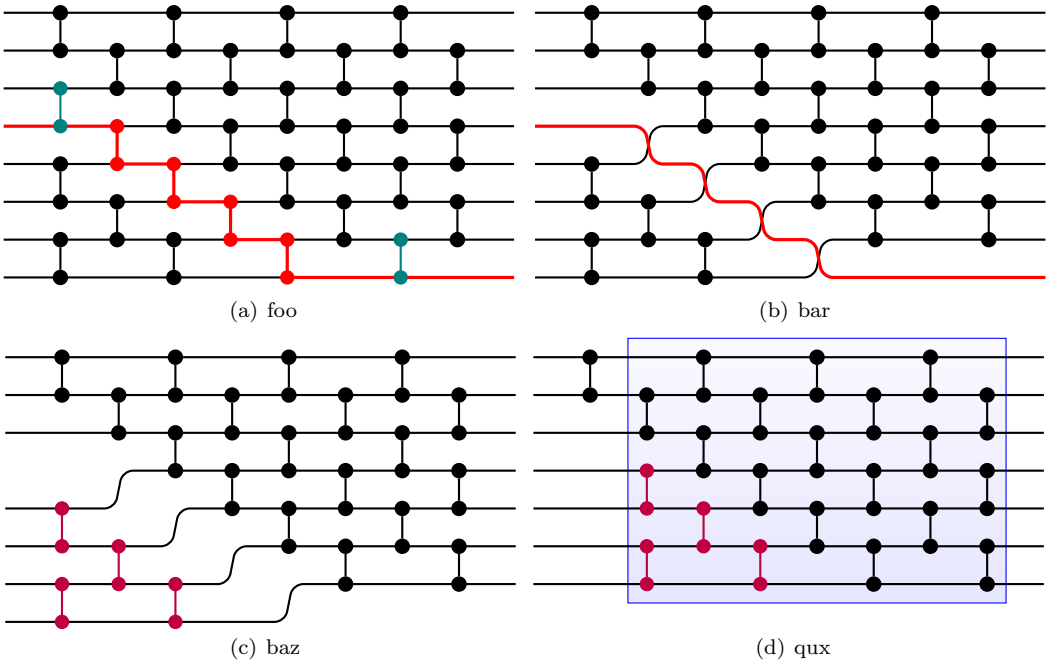
4 Der evolutionäre Ansatz

Um einen evolutionären Algorithmus für Sortiernetzwerke zu entwickeln, werden die vorgestellten Methoden kombiniert.

4.1 Bewertungsfunktion

Um Sortiernetzwerke überhaupt optimieren zu können, muss zunächst die *Güte* eines Netzwerkes definiert werden. Prinzipiell gibt es zwei Ziele, die interessant sind:

- Möglichst wenige Komparatoren („klein“)
- Möglichst wenige Schichten („schnell“)



(a) foo (b) bar

(c) baz (d) qux

Abbildung 9: Eine Leitung wird aus dem *Odd-Even-Transpositionsort* Netzwerk OET(8) entfernt: Auf der rot markierten Leitung wird ∞ angelegt. Da der Wert bei jedem Komparator am unteren Ende herauskommt, ist der Pfad fest vorgegeben. Da die restlichen Werte trotzdem noch richtig sortiert werden müssen, kann dieser Pfad herausgetrennt werden. In der letzten Abbildung ist OET(7) markiert.

Diese Ziele führen im Allgemeinen zu unterschiedlichen Netzwerken. Das kleinste bekannte Sortiernetzwerk für 16 Eingänge besteht aus 60 Komparatoren in 10 Schichten. Das schnellste Netzwerk besteht aus 61 Komparatoren in nur 9 Schichten.

Eine Gütefunktion, die die beiden Ziele „klein“ und „schnell“ berücksichtigen kann, hat die folgende allgemeine Form:

$$Guete(S) = w_{\text{Basis}} + w_{\text{Komparatoren}} \cdot |S|_{\text{Komparatoren}} + w_{\text{Schichten}} \cdot |S|_{\text{Schichten}}$$

Die Parameter $w_{\text{Komparatoren}}$ und $w_{\text{Schichten}}$ dienen dabei der Festlegung des Optimierungsziels. Wenn einer der beiden Parameter gleich Null ist, wird nur das jeweils andere Ziel verfolgt. Sind beide Parameter gleich Null, werden alle Netzwerke mit der gleich Güte bewertet – jegliche Ergebnisse sind dann rein zufälliger Natur.

Mit dem Parameter w_{Basis} kann auf die Selektion Einfluss genommen werden. Ist er groß, wird der relative Unterschied der Güten verschiedener Netzwerke kleiner, was die *Exploration*, das Absuchen des gesamten Lösungsraums, begünstigt. Wählt man w_{Basis} hingegen klein, in Abhängigkeit von den anderen beiden Parametern sind auch negative Werte möglich, werden die relativen Unterschiede groß. Dadurch wird die *Exploitation*, das Finden lokaler Optima, bevorzugt.

4.2 Selektion

...

4.3 Rekombination

Bei der Rekombination werden zwei Individuen – hier Sortiernetzwerke – zu einer neuen Lösung kombiniert. Dazu verwenden wir einen Mischer, zum Beispiel den *bitonen Mischer* (Abschnitt 2.2.1) oder den *Odd-Even-Mischer* (Abschnitt 2.3.1), um die beiden Netzwerke zu einem Netzwerk mit $2n$ Leitungen zusammenzufügen. Anschließend entfernen wir zufällig n Leitungen wie in Abschnitt 3.2 beschrieben.

Dieses Verfahren hat den großen Vorteil, dass es die Sortiereigenschaft erhält.

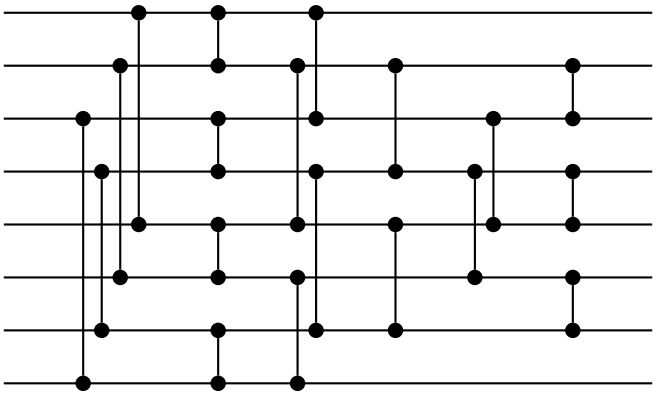


Abbildung 10: Ein mit dem evolutionären Algorithmus erzeugtes Sortiernetzwerk mit acht Eingängen. Es besteht aus 19 Komparatoren in 6 Schichten.

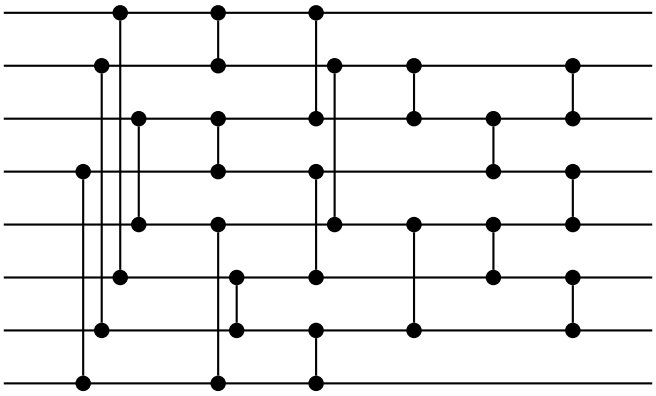


Abbildung 11: `images/08-e2-1237993371.tex`: 19 Komparatoren in 6 Schichten

4.4 Mutation

Zu einem vollständigen evolutionären Algorithmus gehört außerdem eine Mutation – eine zufällige Veränderung einer Lösung. Leider ist es nicht möglich ein Sortiernetzwerk zufällig zu verändern aber trotzdem die Sortiereigenschaft zu erhalten. Selbst das *Hinzufügen* eines zufälligen Komparators kann diese Eigenschaft zerstören.

Nach einer Mutation müsste man überprüfen, ob das neue Komparatornetzwerk die Sortiereigenschaft noch besitzt. Nach heutigem Wissenstand ist diese Überprüfung nur mit exponentiellem Aufwand möglich, etwa durch das Ausprobieren aller 2^n Bitmuster.

Um das Potenzial einer Mutation abzuschätzen habe ich in den evolutionären Algorithmus eine Überprüfung eingebaut. Unmittelbar vor dem Einfügen in die Population überprüft das Programm die Notwendigkeit jedes einzelnen Komparators. Dazu wurde nacheinander jeder Komparator entfernt und überprüft, ob das verbleibende Netzwerk die Sortiereigenschaft noch besitzt.

- Güte von Sortiernetzwerken (Anzahl der Komparatoren, Anzahl der Schichten, kobiniert)
- Rekombination: Merge Anhängen und Leitungen entfernen.

Ein Beispielnetzwerk, das von dem Algorithmus gefunden wird, zeigt Abbildung 10.

5 Shmoo-Äquivalenz

Die folgenden 16-Eingang-Sortiernetzwerke wurden alle mit dem *Algorithmus 1* gefunden. Sie haben alle 63 Komparatoren in 10 Schichten, jeweils die selbe Anzahl wie Odd-Even-Mergesort.

Um wiederkehrende Muster in den hinteren Schichten der erzeugten Sortiernetzwerke besser untersuchen zu können, wurden die erzeugten Netzwerke in Gruppen aufgeteilt. Zwei Netzwerke befinden sich

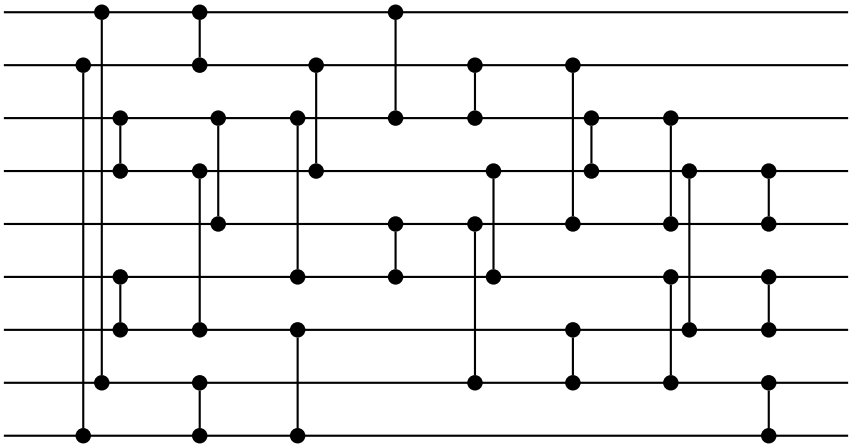


Abbildung 12: `images/09-e2-1237997073.tex`: 25 Komparatoren in 8 Schichten

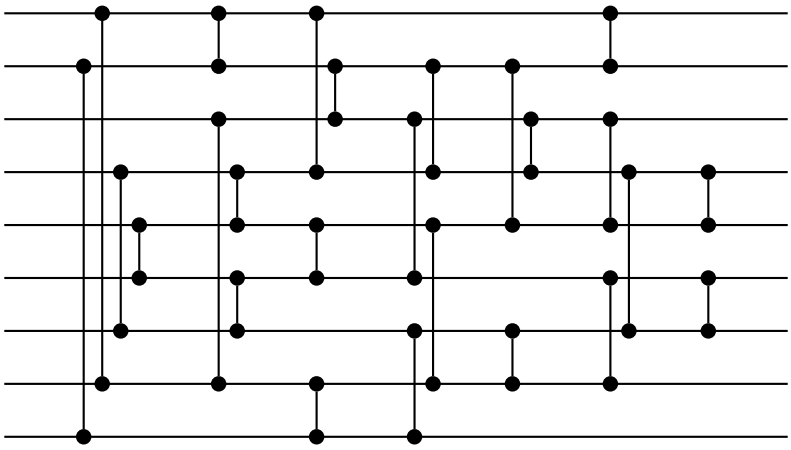


Abbildung 13: `images/09-e2-1237999719.tex`: 25 Komparatoren in 7 Schichten

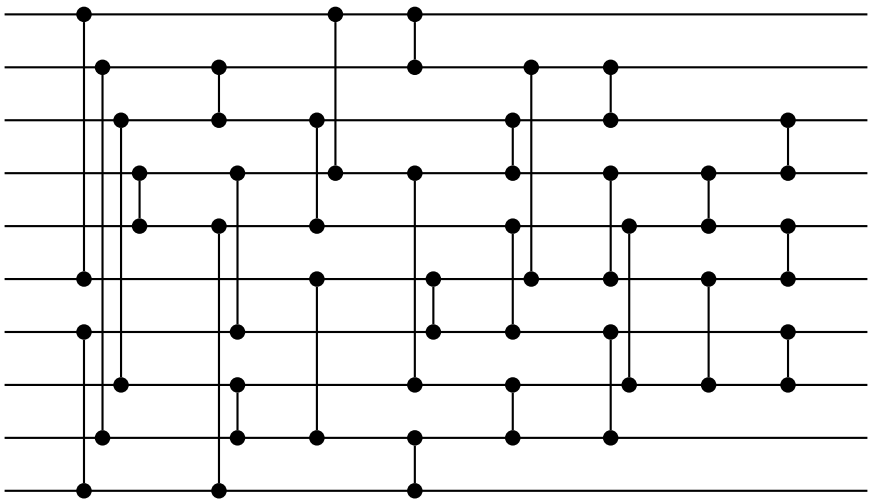


Abbildung 14: `images/10-e2-1239014566.tex`: 29 Komparatoren in 8 Schichten

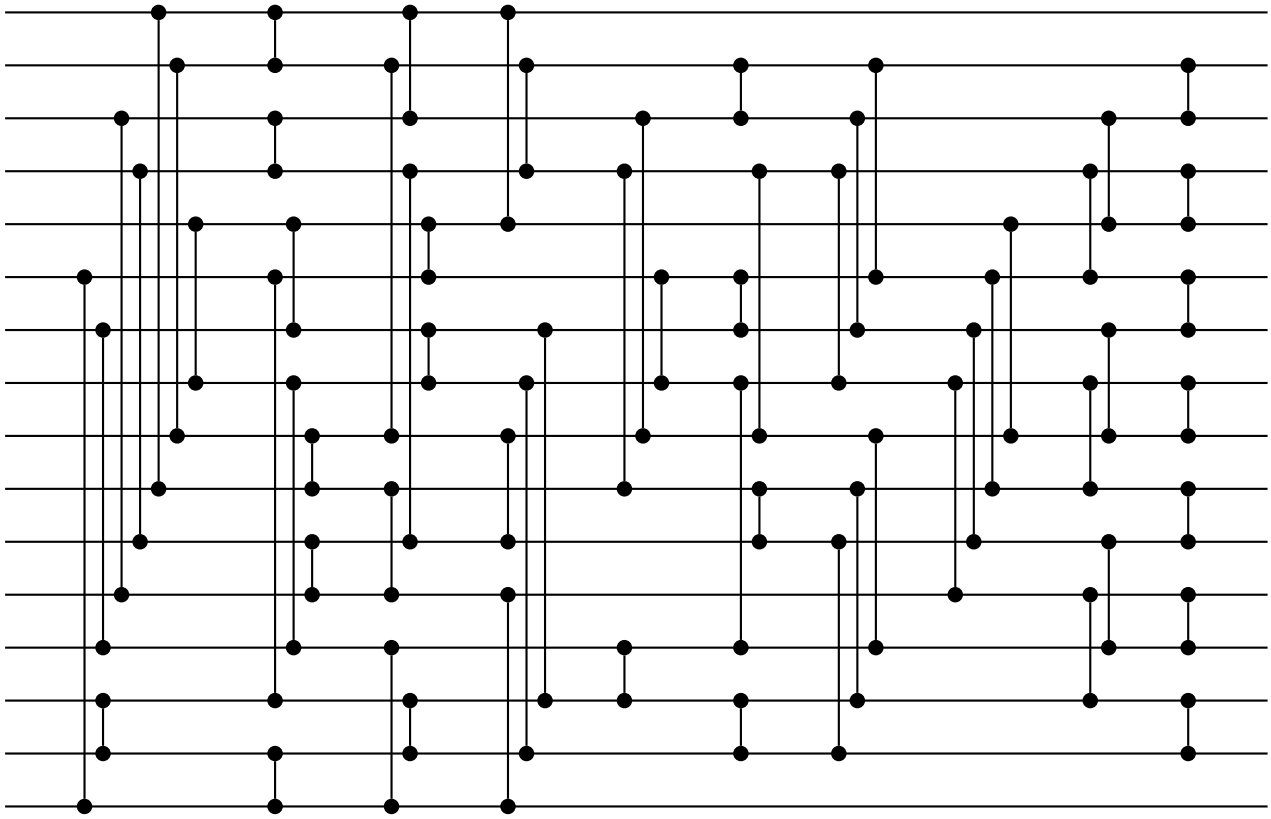


Abbildung 15: `images/16-e1/group0/16-e1-1258009316.tex`: 63 Komparatoren in 10 Schichten.

dann in der selben Gruppen, wenn die Nullen bzw. Einsen, die auf einer Leitung vorkommen können, nach der 5. Schicht (Schicht 4, da bei Null mit dem Zählen begonnen wird) nicht mehr ändert. Das heißt, dass die Schichten 0–4 unterschiedlich aufgebaut sind, aber den selben Effekt erziehlen. Die Schichten 5–9 sind hingegen innerhalb einer Gruppe austauschbar und oft (immer?) identisch.

Die Anzahl der Netzwerke in den jeweiligen Gruppen ist unterschiedlich. Zur Zeit sind in den Gruppen so viele Netzwerke:

Gruppe 0	21	50,0%
Gruppe 1	10	23,8%
Gruppe 2	6	14,3%
Gruppe 3	3	7,1%
Gruppe 4	2	4,8%

Die hinteren Schichten zwischen den Gruppen 1 und 3 schauen so aus, als wären sie nur gespiegelt. Warum kommt Gruppe 1 aber viel häufiger vor? Ggf. eine Konsequenz aus dem Normieren?

Dito für die Gruppen 2 und 4. Warum ist die eine häufiger?

Ist Gruppe 0 symmetrisch bzgl. der Leitungen?

5.1 Güte

- So gut kann man mindestens werden *(→ Bitonic-Mergesort, vermute ich).*
- Wie gut die Netzwerke werden, hängt stark vom verwendeten *Mischer ab.*

5.2 Vom evolutionären Algorithmus zu einer Markov-Kette

- Kombiniere immer das aktuelle Netzwerk mit sich selbst.
- Kann die Mindestgüte immernoch erreicht werden? (*Ich denke schon.*)
- Anzahl der erreichbaren Sortiernetzwerke.

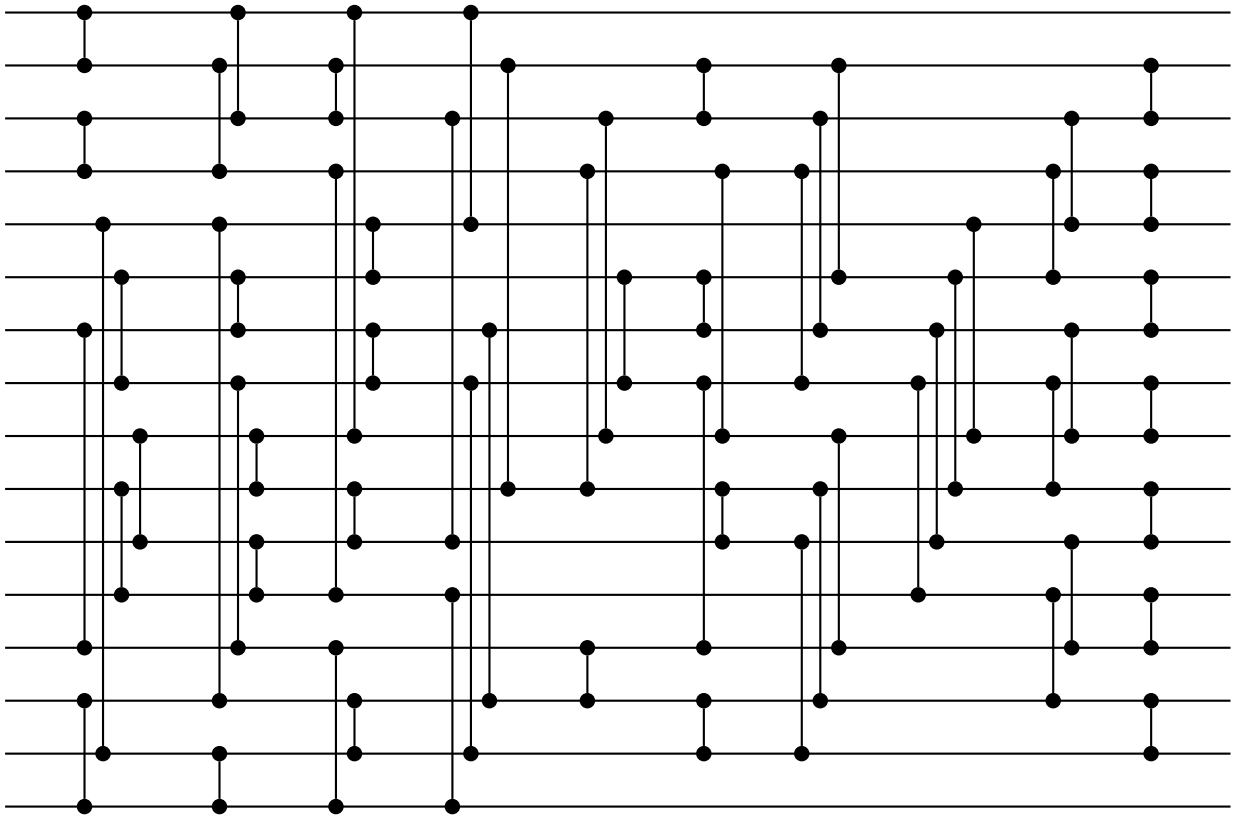


Abbildung 16: `images/16-e1/group0/16-e1-1258010866.tex`: 63 Komparatoren in 10 Schichten.

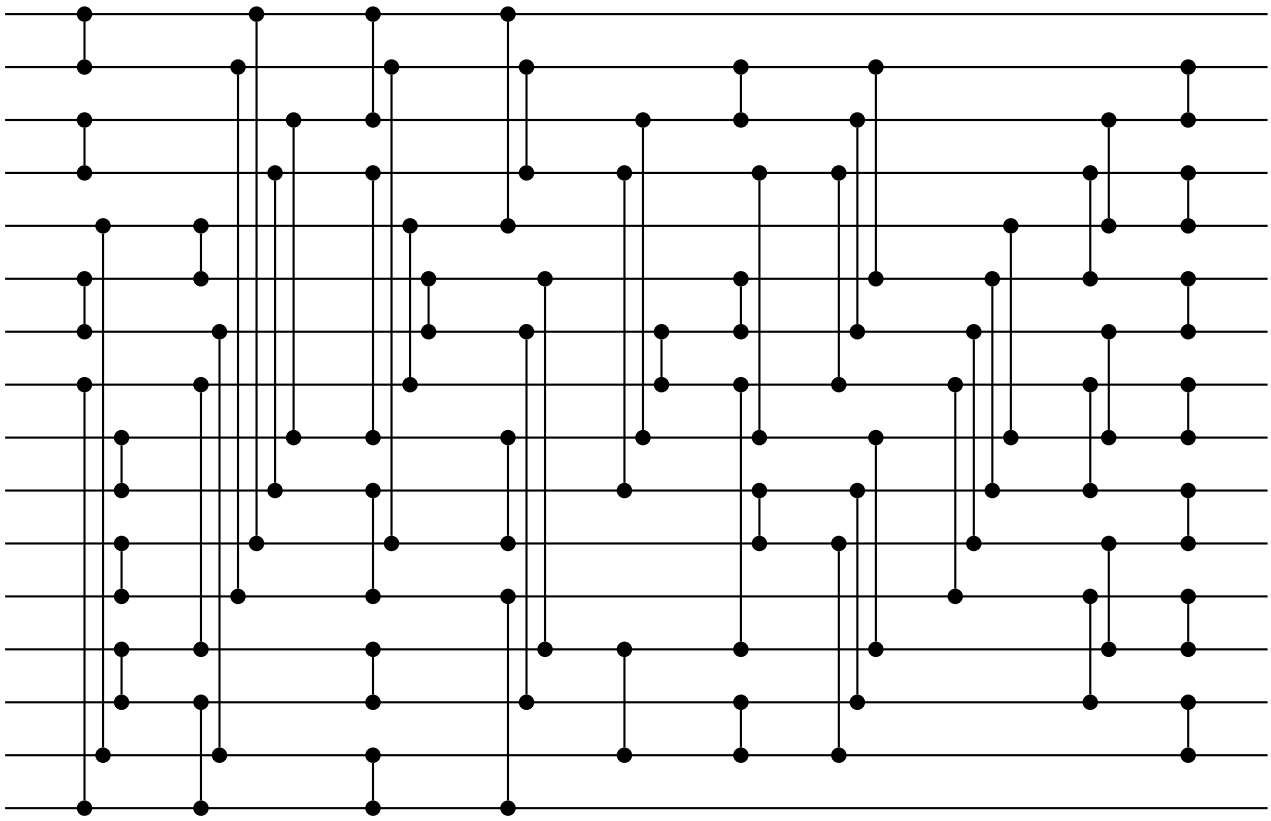


Abbildung 17: `images/16-e1/group0/16-e1-1258011861.tex`: 63 Komparatoren in 10 Schichten.

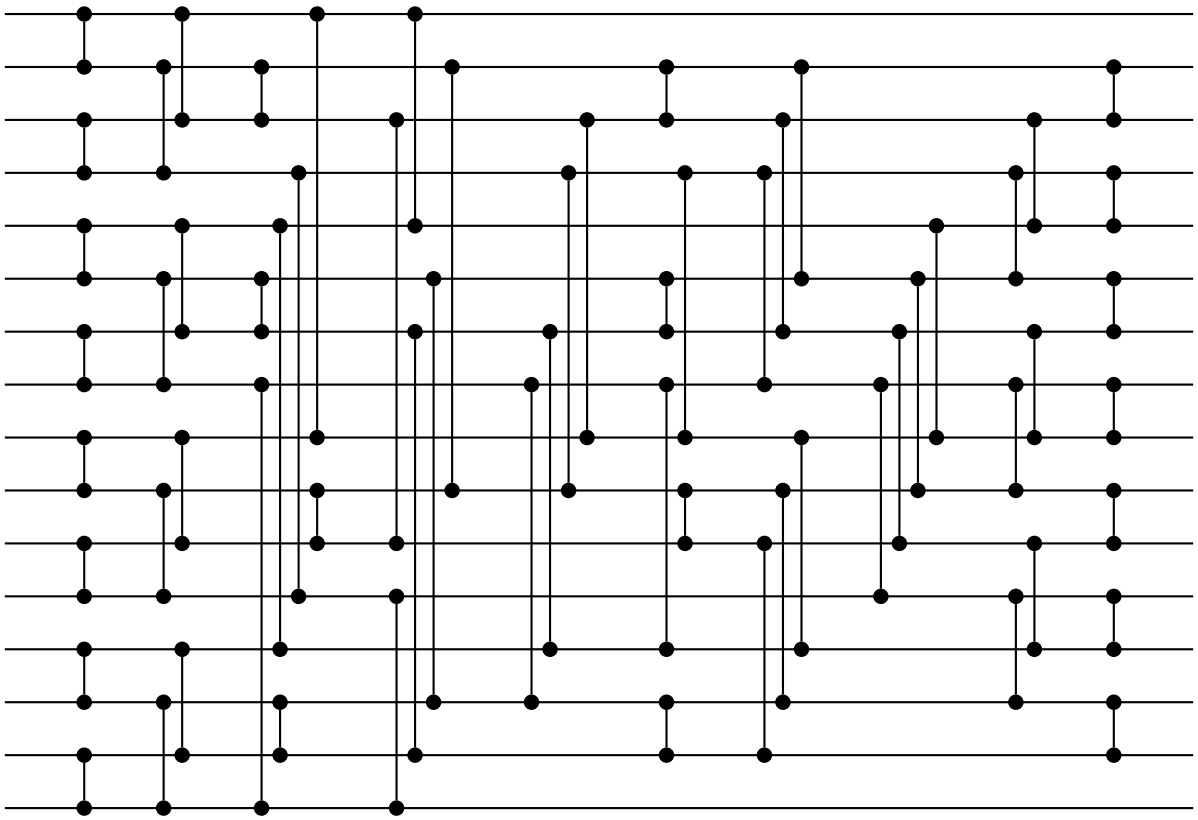


Abbildung 18: `images/16-e1/group0/16-e1-1259060992.tex`: 63 Komparatoren in 10 Schichten.

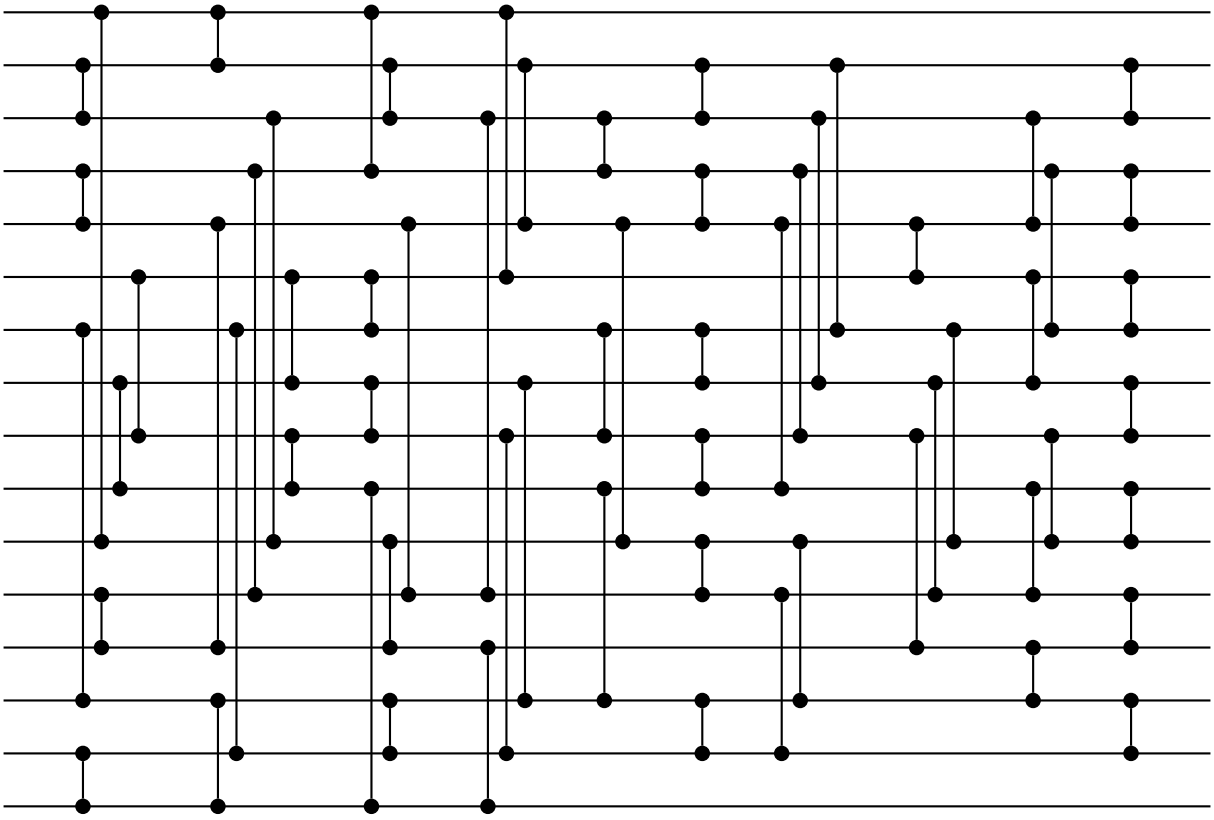


Abbildung 19: `images/16-e1/group1/16-e1-1258009982.tex`: 63 Komparatoren in 10 Schichten. Schichten 4–9 identisch zu 16-e1-1258030047 (Gruppe 1).

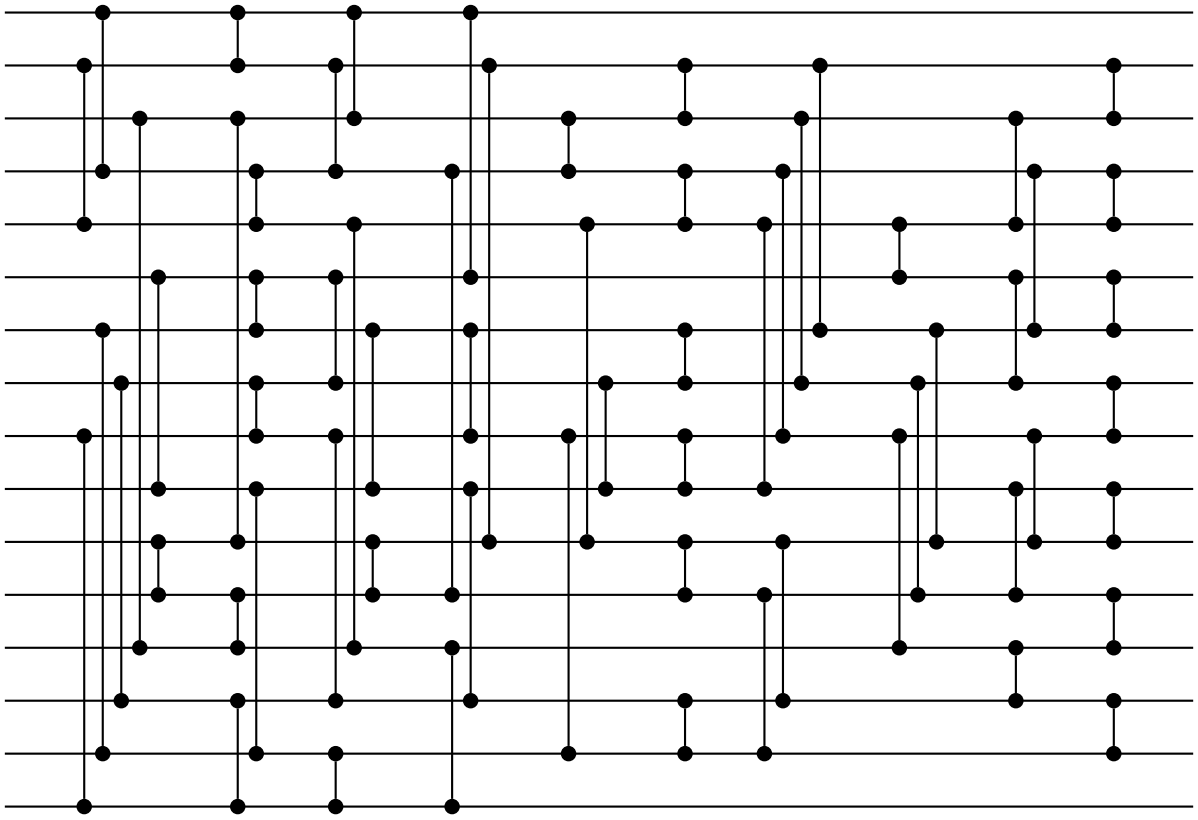


Abbildung 20: `images/16-e1/group1/16-e1-1258010023.tex`: 63 Komparatoren in 10 Schichten.

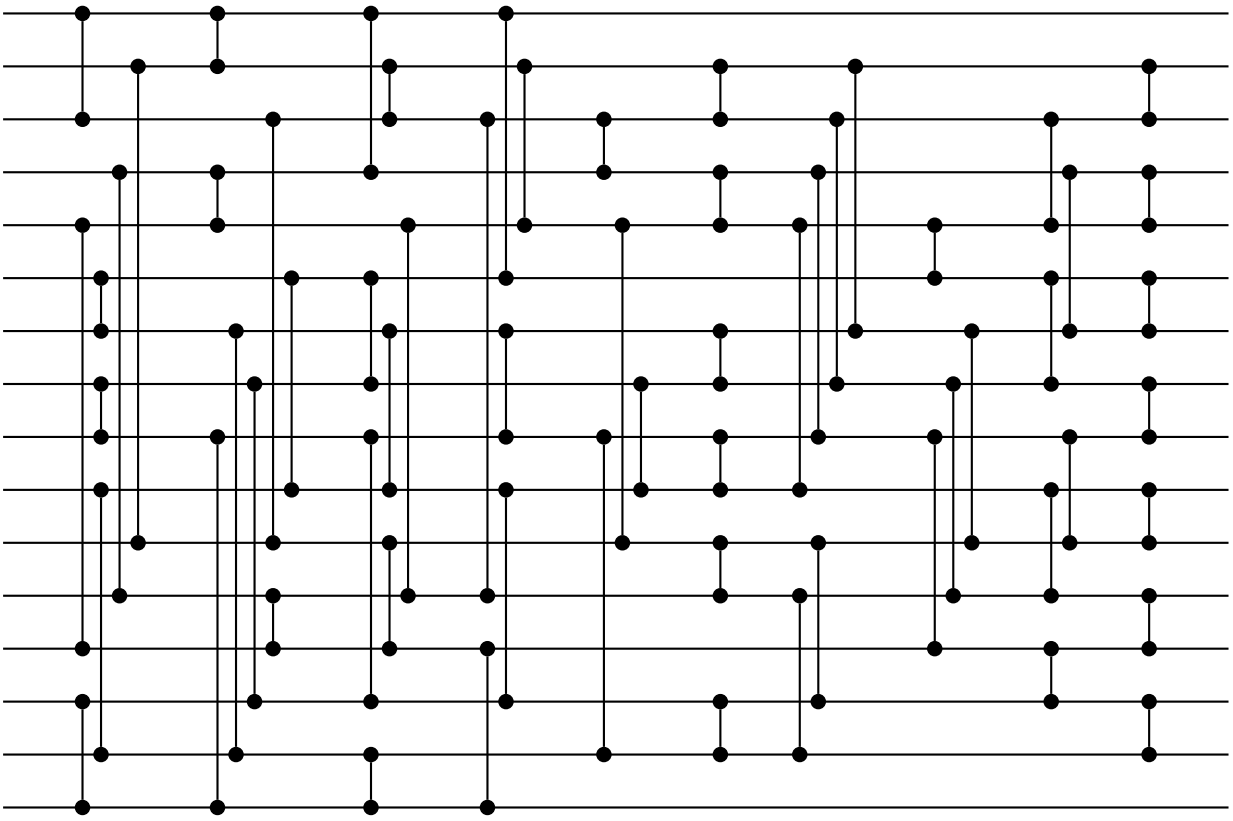


Abbildung 21: `images/16-e1/group1/16-e1-1258029734.tex`: 63 Komparatoren in 10 Schichten.

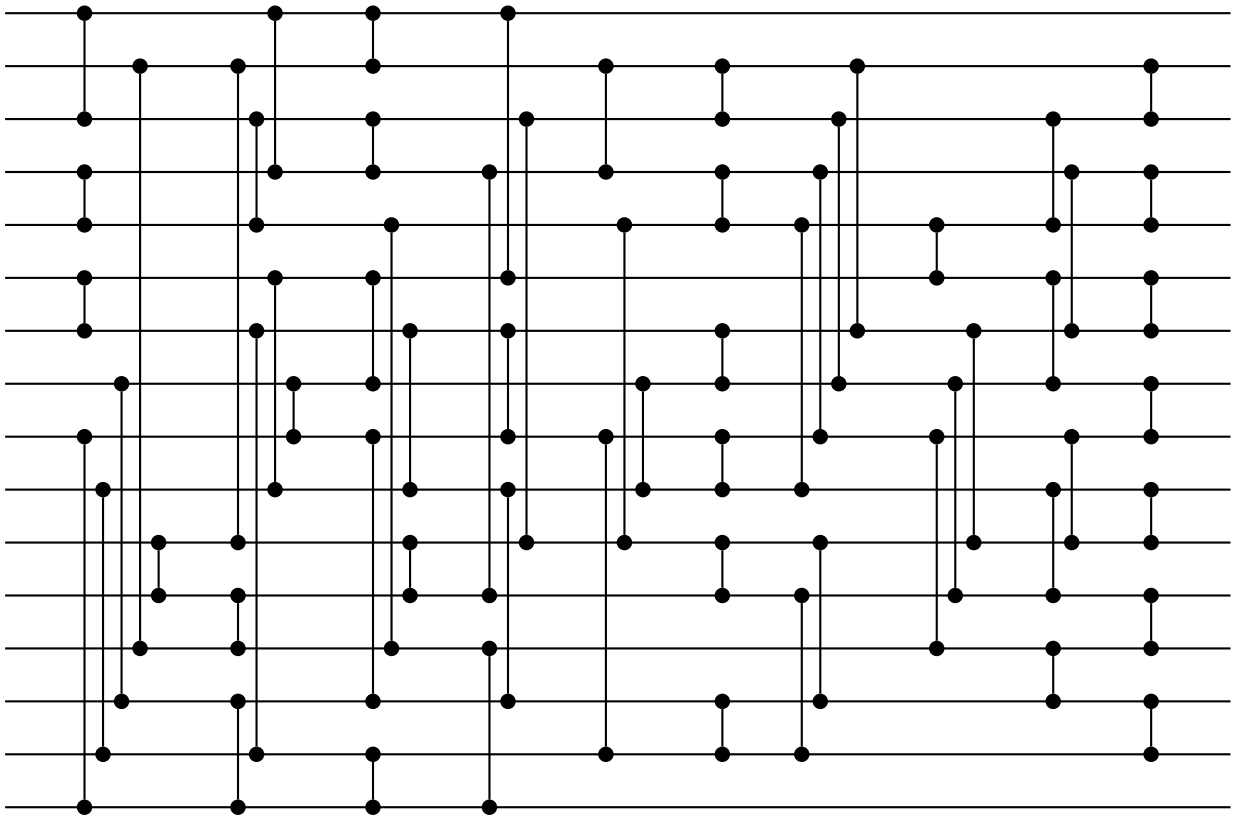


Abbildung 22: `images/16-e1/group1/16-e1-1258030047.tex`: 63 Komparatoren in 10 Schichten.

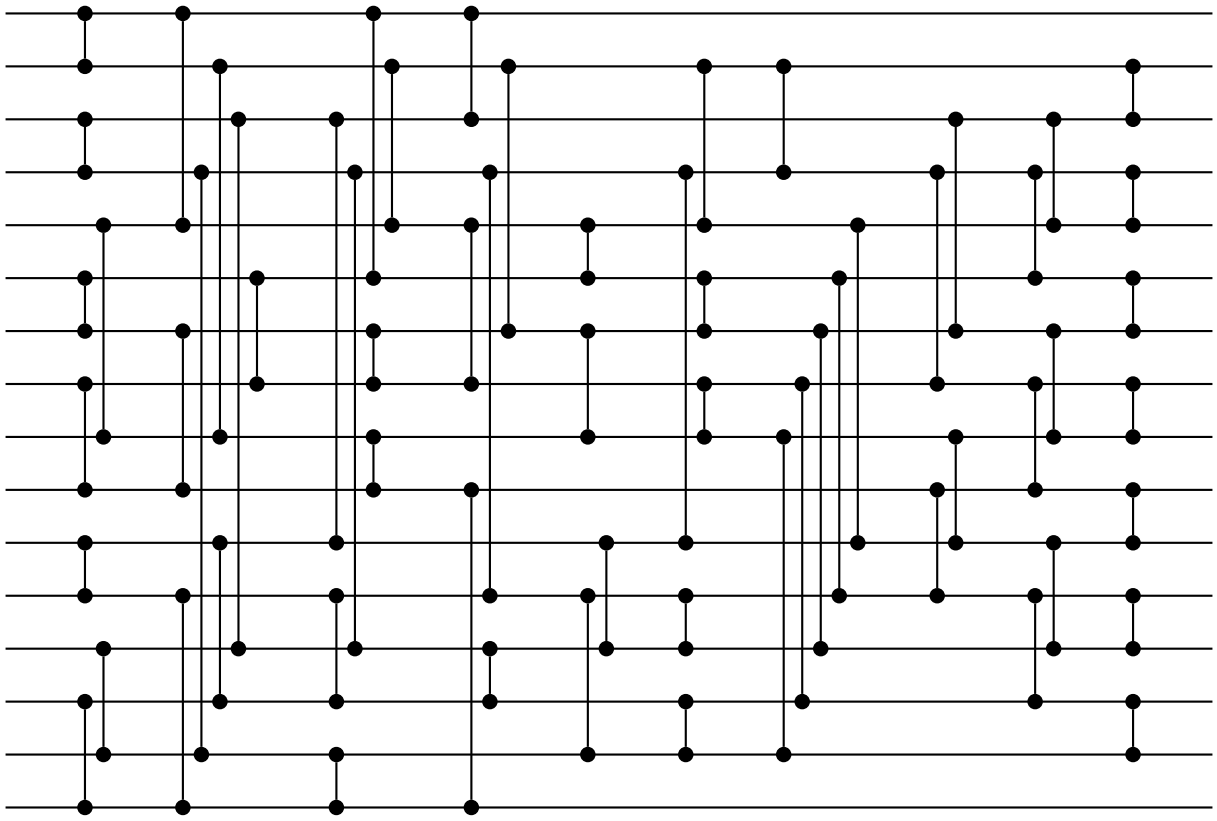


Abbildung 23: `images/16-e1/group2/16-e1-1258029063.tex`: 63 Komparatoren in 10 Schichten.

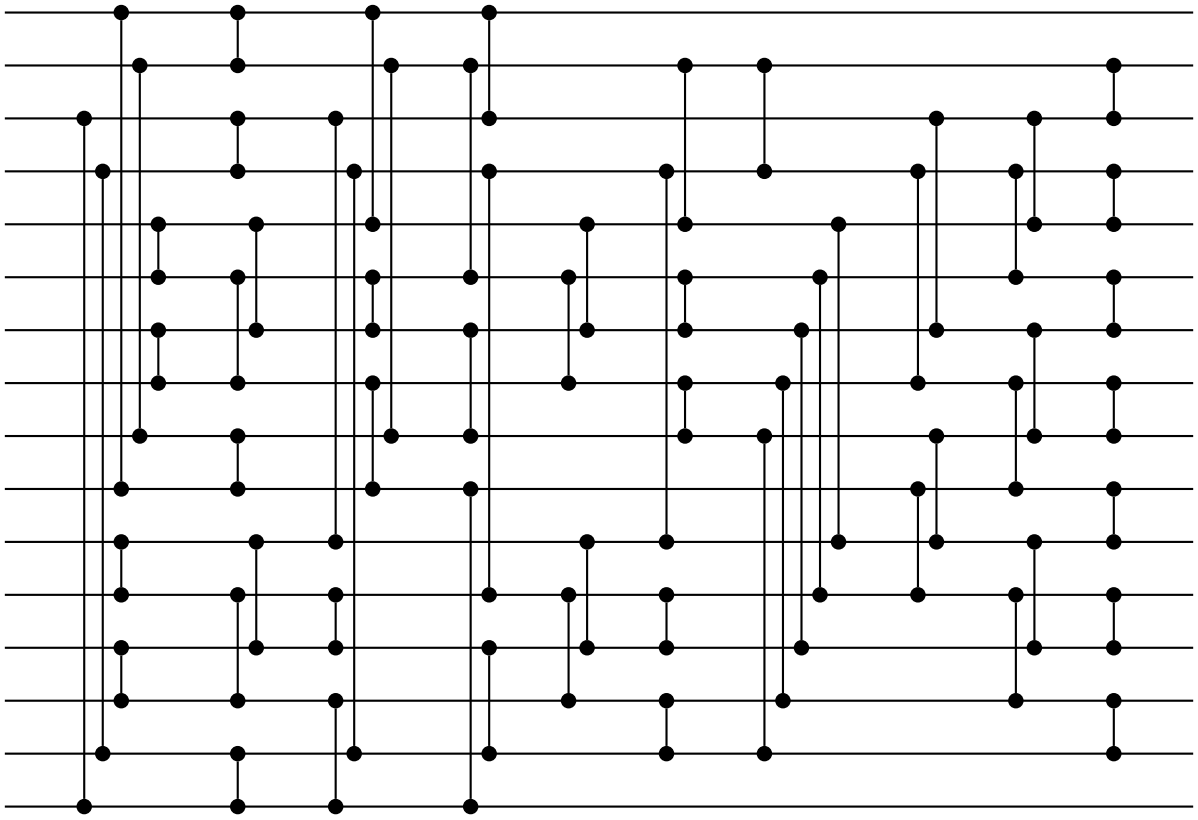


Abbildung 24: `images/16-e1/group2/16-e1-1258034821.tex`: 63 Komparatoren in 10 Schichten.

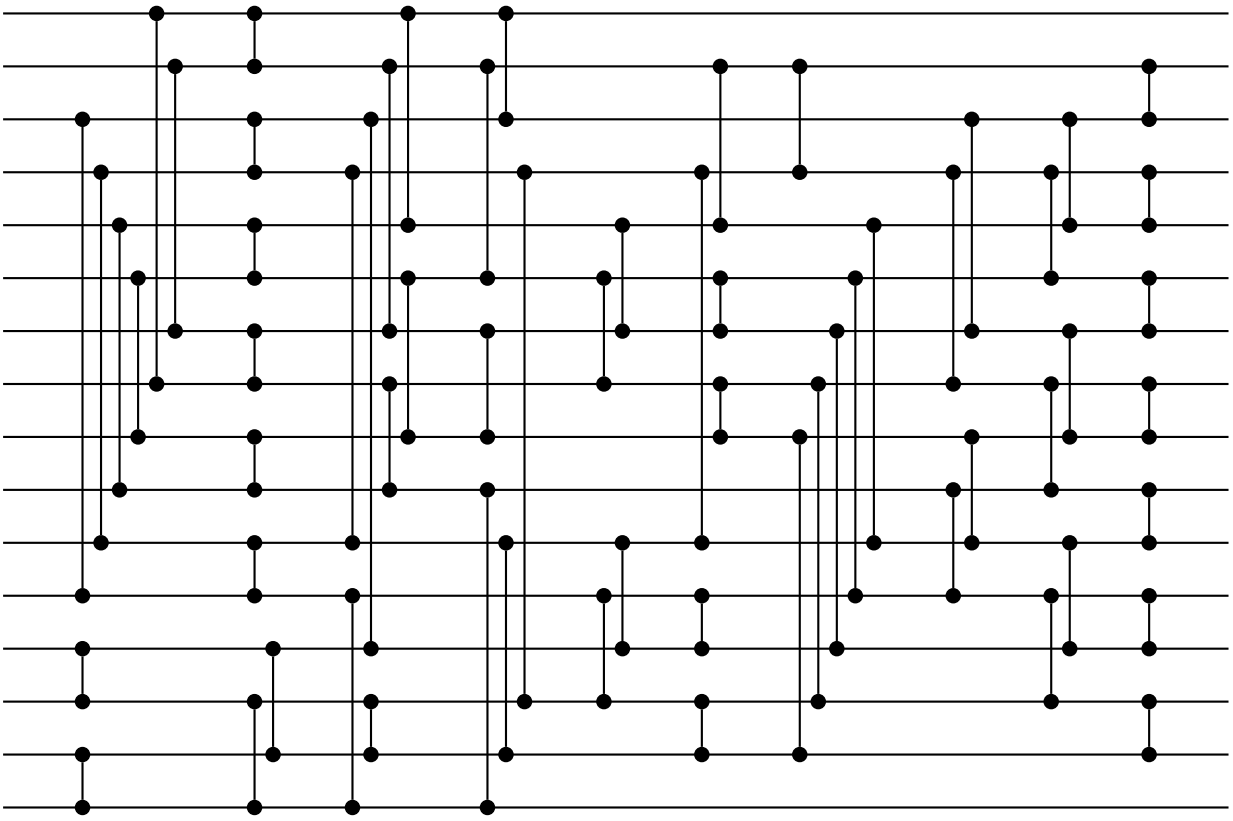


Abbildung 25: `images/16-e1/group2/16-e1-1259054993.tex`: 63 Komparatoren in 10 Schichten.

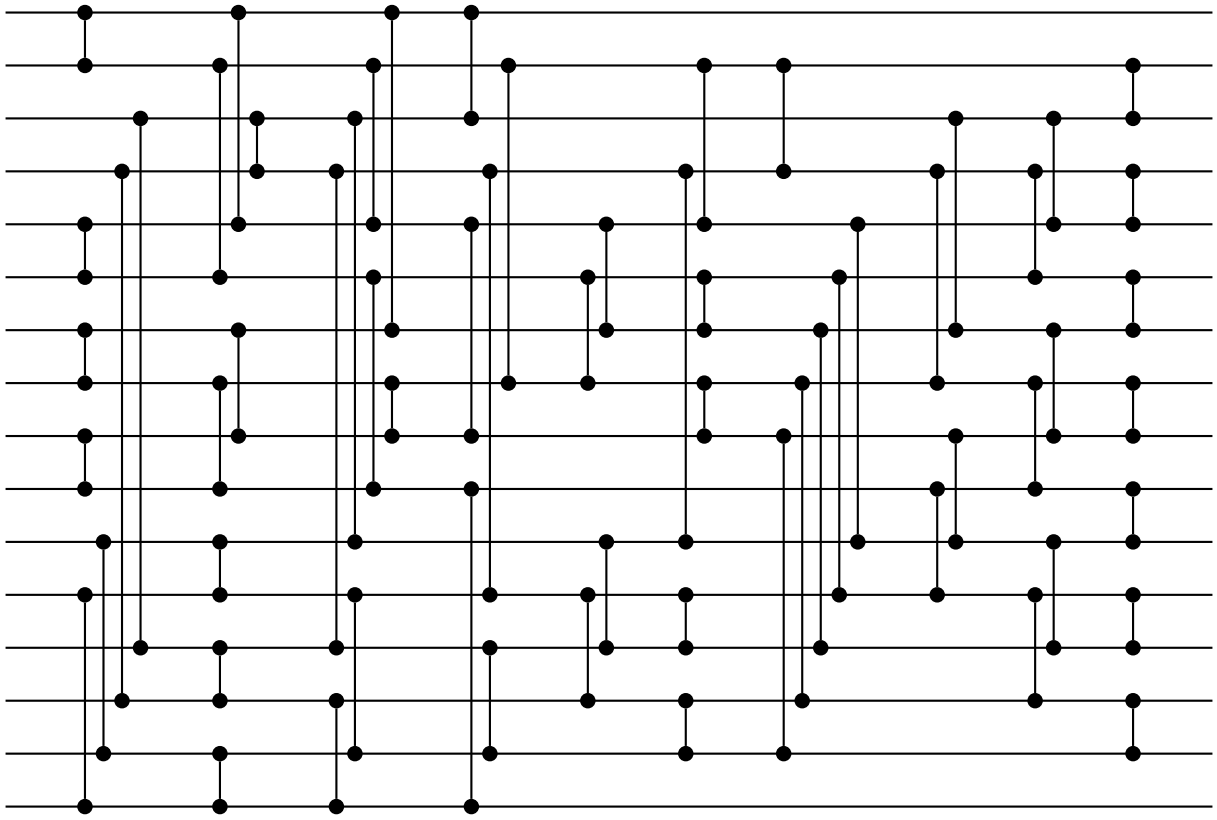


Abbildung 26: `images/16-e1/group2/16-e1-1259058588.tex`: 63 Komparatoren in 10 Schichten.

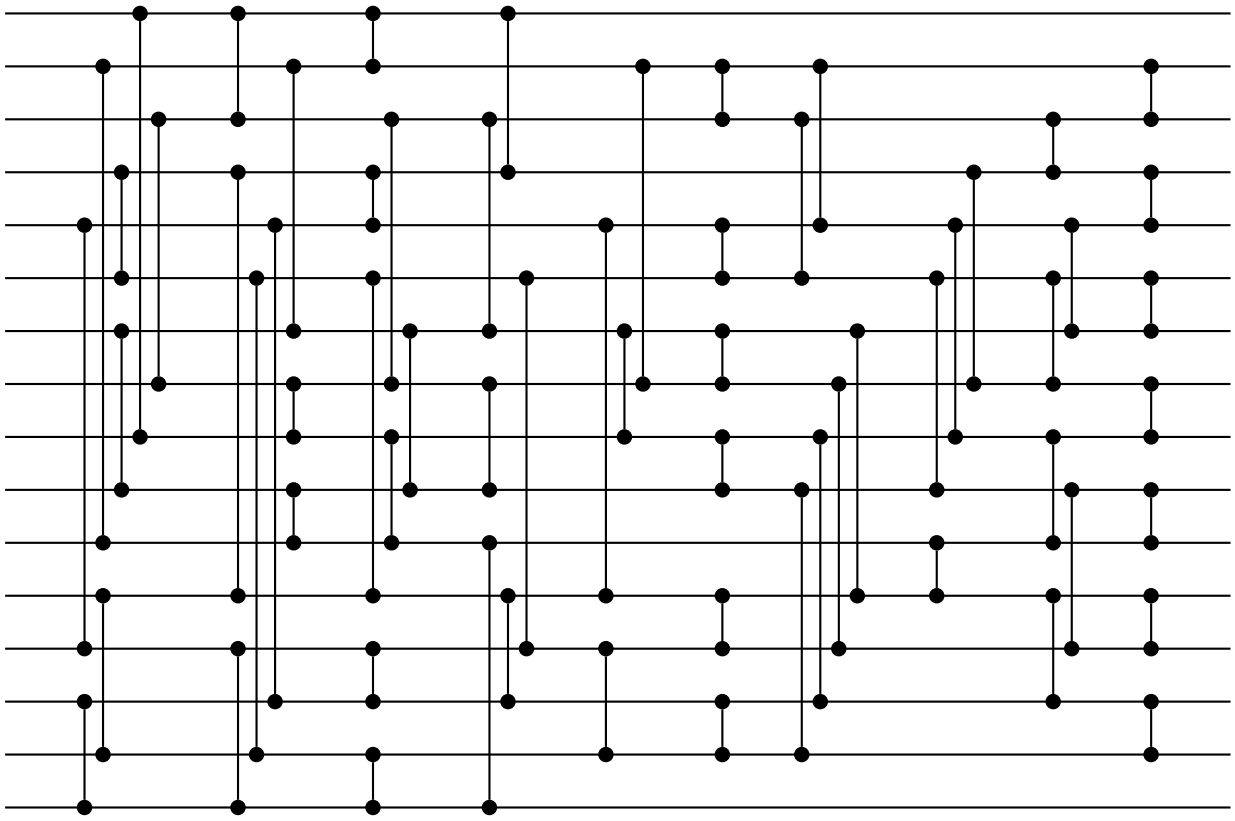


Abbildung 27: `images/16-e1/group3/16-e1-1258012027.tex`: 63 Komparatoren in 10 Schichten.

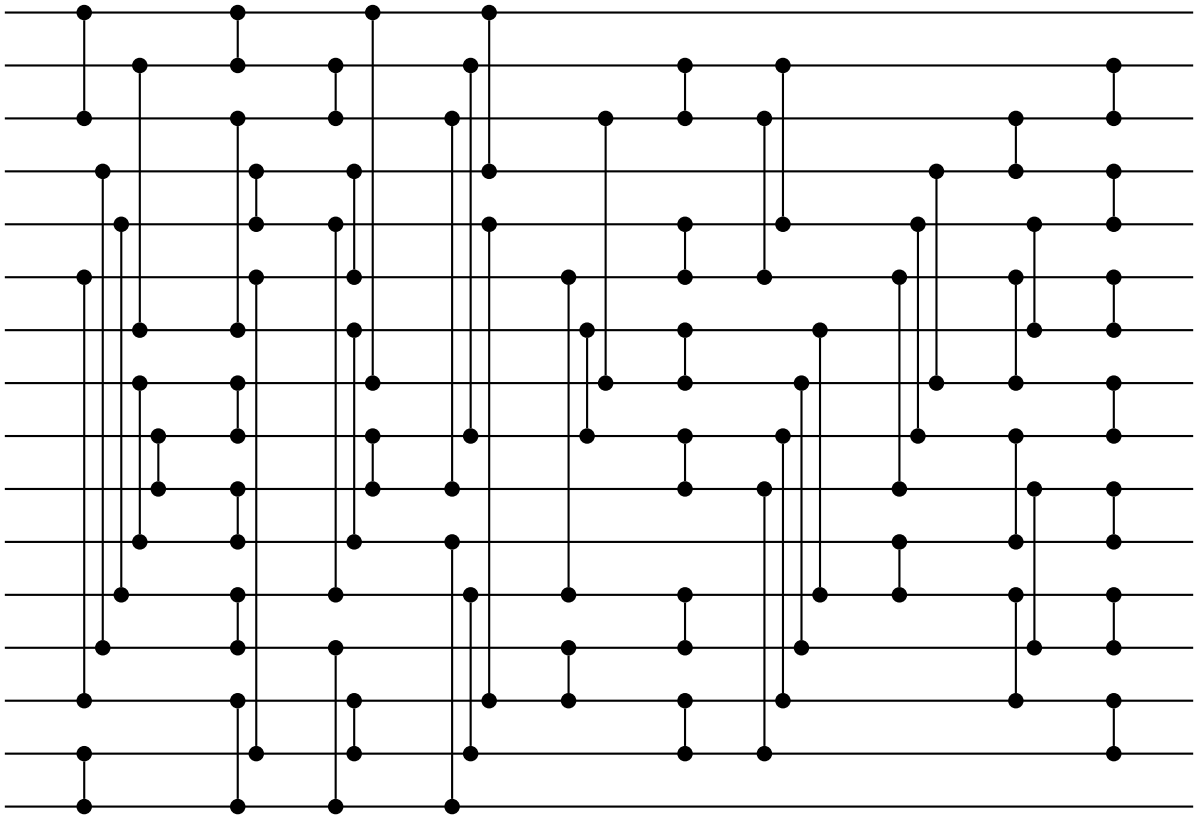


Abbildung 28: `images/16-e1/group3/16-e1-1258037039.tex`: 63 Komparatoren in 10 Schichten.

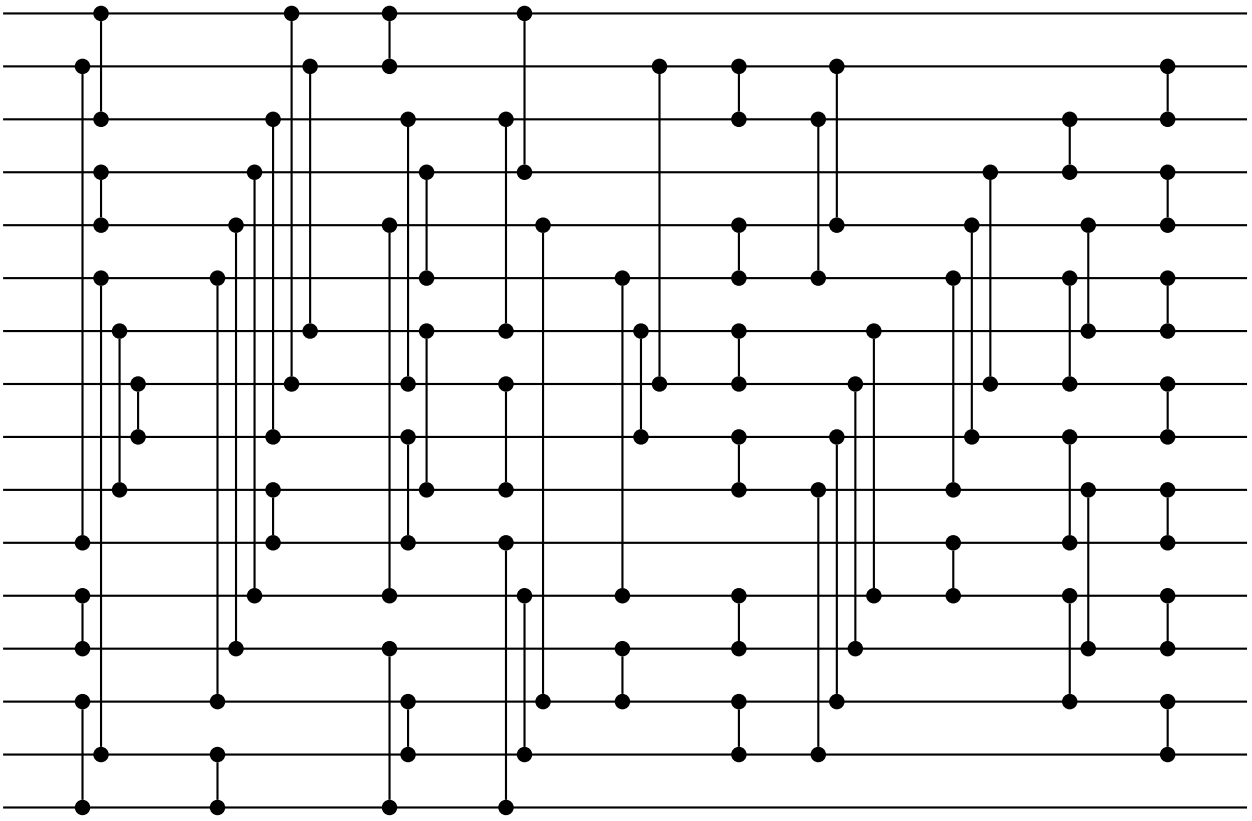


Abbildung 29: `images/16-e1/group3/16-e1-1259065042.tex`: 63 Komparatoren in 10 Schichten.

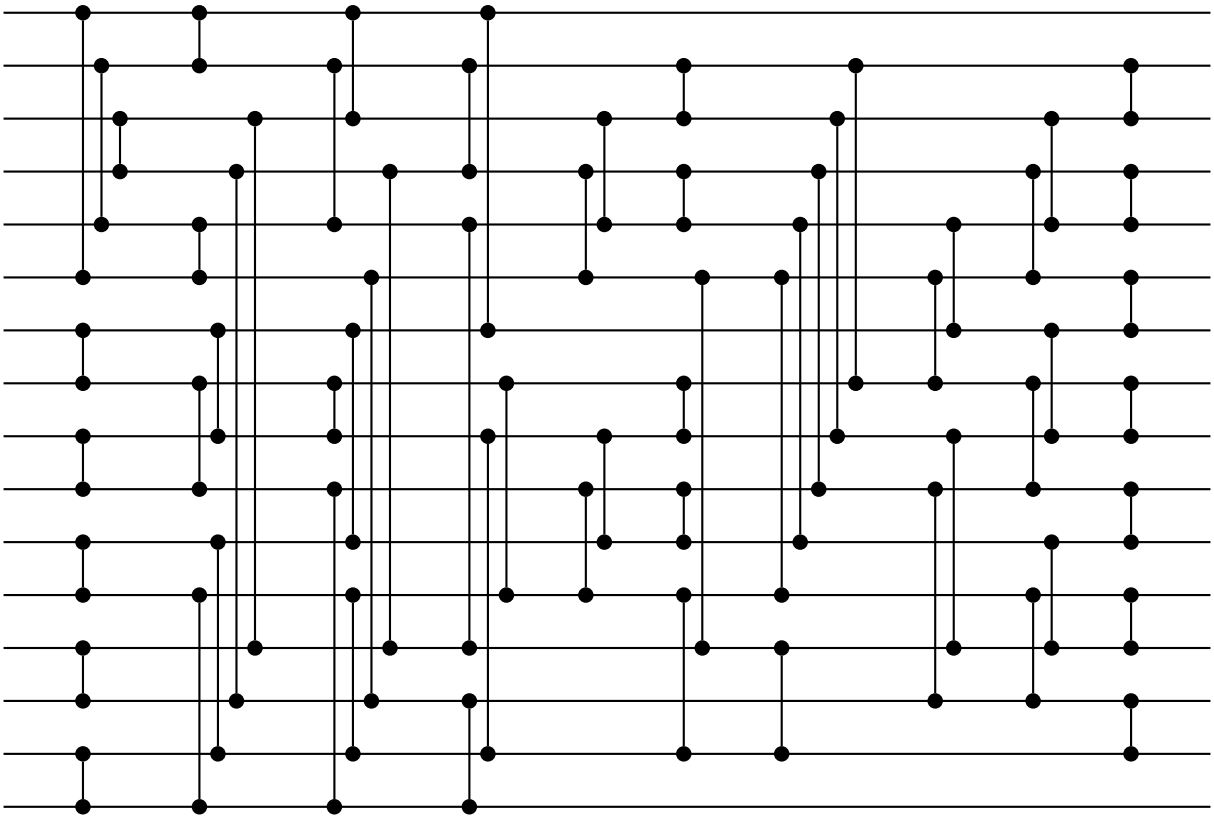


Abbildung 30: `images/16-e1/group4/16-e1-1259060520.tex`: 63 Komparatoren in 10 Schichten. (Gruppe 4).

6 Empirische Beobachtungen

- So schnell konvergiert der Algorithmus.
- ...

7 Ausblick

Das würde mir noch einfallen...

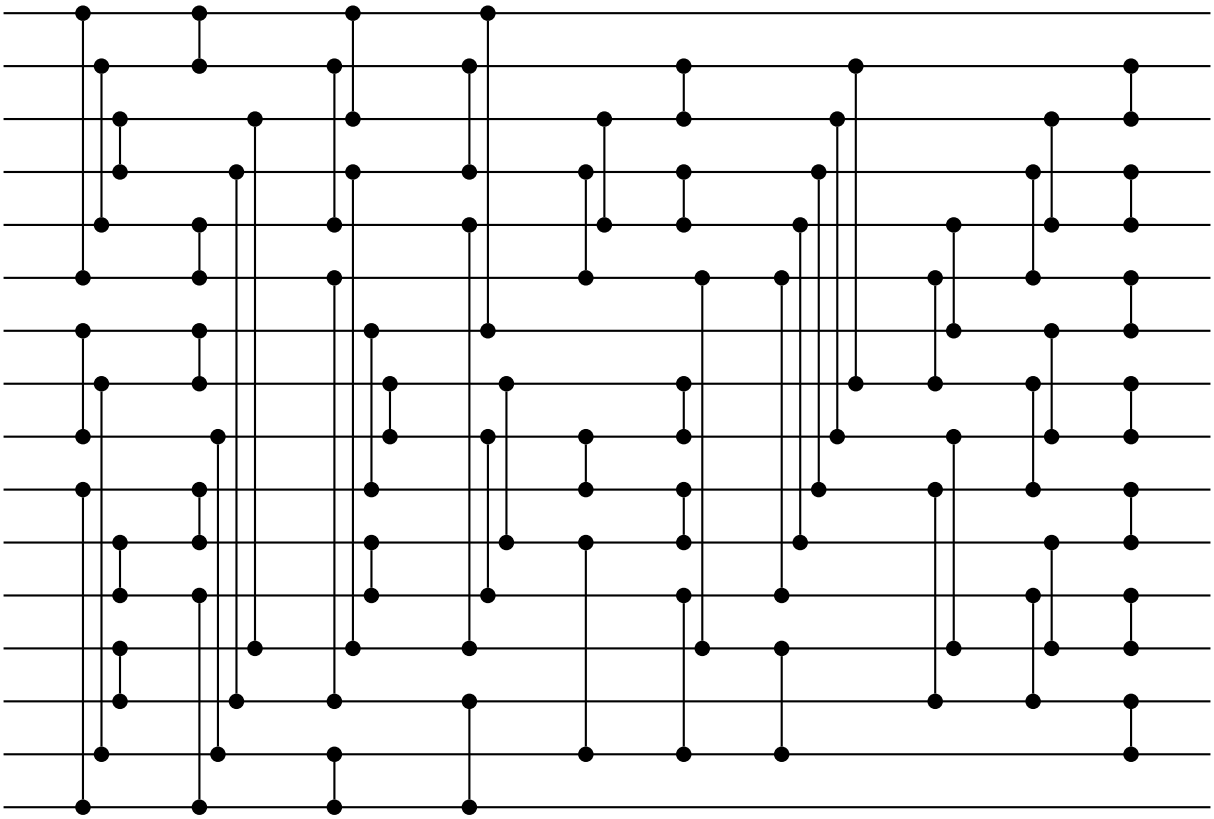


Abbildung 31: `images/16-e1/group4/16-e1-1259067171.tex`: 63 Komparatoren in 10 Schichten. (Gruppe 4).