10 Receive graphs

In this chapter, we will consider the problem of scheduling receive graphs in the LogP model. Note that this problem is equivalent to the problem of scheduling send graphs under an independent data semantics. Like in Chapter 9, the structure of minimum-length schedules will be used to construct good schedules for receive graphs.

In Section 10.1, it is shown that constructing minimum-length schedules for receive graphs on an unrestricted number of processors is a strongly NP-hard optimisation problem. This is proved using a polynomial reduction similar to the one presented in the proof of Lemma 9.1.1.

In Section 10.2, two polynomial-time approximation algorithms are presented. Both algorithms assume that $g$ does not exceed $o$. The first approximation algorithm constructs schedules for receive graphs on an unrestricted number of processors that are at most three times as long as a minimum-length schedule on an unrestricted number of processors. In Section 10.2.2, it is shown that a schedule on $P$ processors that is at most $3 + \frac{1}{k+1}$ times as long as a minimum-length schedule on $P$ processors can be constructed in polynomial time for all constant $k \in \mathbb{Z}^+$.

In Section 10.3, it is shown that if all task lengths are equal, then a minimum-length schedule for a receive graph on an unrestricted number of processors can be constructed in polynomial time. This is an improvement over the result of Kort and Trystram [55] who proved that a minimum-length schedule for a receive graph on an unrestricted number of processors can be constructed in polynomial time if $g$ does not exceed $o$ and all sources have the same execution length.

10.1 An NP-completeness result

In Chapter 9, it was proved that constructing minimum-length schedules for send graphs on an unrestricted number of processors is a strongly NP-hard optimisation problem. This was proved using the polynomial reduction from 3PARTITION presented in the proof of Lemma 9.1.1. Let $(G, \mu, L, o, g, \infty)$ be the instance constructed by this reduction for an instance of 3PARTITION. The send graph $G$ contains $m + 2$ large tasks that must be scheduled on different processors. These are the only tasks that are scheduled after the communication operations in a minimum-length schedule for $(G, \mu, L, o, g, \infty)$.

By reversing all arcs in send graph $G$, we obtain a receive graph $G'$. In a minimum-length schedule for $(G', \mu, L, o, g, \infty)$, the large tasks are the only ones that are scheduled before the communication operations. Hence the reversal of the minimum-length schedule for the send graph can be viewed as a minimum-length schedule for the receive graph. Thus a similar reduction as the one presented in the proof Lemma 9.1.1 can be used to prove that constructing minimum-length schedules for receive graphs on an unrestricted number of processors is a strongly NP-hard optimisation problem.

Theorem 10.1.1. Constructing minimum length schedules for instances $(G, \mu, L, o, g, \infty)$, such that $G$ is a receive graph, is a strongly NP-hard optimisation problem.

Theorem 10.1.1 shows that it is unlikely that a minimum-length schedule for an instance
(G,μ,c,L,o,g,∞), such that G is a receive graph and g > o, can be constructed in polynomial time. It is unknown whether minimum-length schedules on an unrestricted number of processors can be constructed in polynomial time if g does not exceed o. Kort and Trystram [55] proved that if g ≤ o and all tasks have the same length, then a minimum-length schedule for a receive graph can be constructed in polynomial time.

10.2 Two approximation algorithms

In this section, two polynomial-time approximation algorithms for scheduling receive graphs in the LogP model are presented. The first is presented in Section 10.2.1. It constructs schedules for receive graphs on an unrestricted number of processors. The length of these schedules are at most three times as long as a minimum-length schedule on an unrestricted number of processors. The algorithm presented in Section 10.2.2 constructs schedules for receive graphs on a restricted number of processors. It is shown that for each constant k ∈ Z^+, a schedule on P processors that is at most 3 + 1/k times as long as a minimum-length schedule on P processors can be constructed in polynomial time.

Both algorithms divide the set of sources of a receive graph into two sets. Let G be a receive graph. Consider an instance (G,μ,c,L,o,g,P). A source y of G is called communication intensive if μ(y) > c(y)o. Otherwise, it is called computation intensive. Hence a source y of G is communication intensive if the total duration of the send operations needed to send the result of y to another processor exceeds the execution length of y. The sets of communication-intensive and computation-intensive sources will be used to compute lower bounds on the length of minimum-length schedules for receive graphs.

10.2.1 An unrestricted number of processors

In this section, an approximation algorithm for scheduling receive graphs on an unrestricted number of processors is presented. For this algorithm, we will assume that g does not exceed o. The algorithm constructs schedules for receive graphs on an unrestricted number of processors that are at most three times as long as a minimum-length schedule on an unrestricted number of processors. The algorithm is similar to the 3-approximation algorithm of Hollerman et al. [46] for scheduling send and receive graphs in a model of parallel computation that resembles the LogP model.

We start by proving some properties of minimum-length schedules for receive graphs on an unrestricted number of processors. The next lemma shows that if a source of a receive graph G is not scheduled on the same processor as the sink of G, then the receive operations corresponding to this source may be scheduled after the sources of G that are scheduled on the same processor as the sink of G. This result is not true if g exceeds o. If g exceeds o, then some sources of G may have to be scheduled between the receive operations in a minimum-length schedule for G on an unrestricted number of processors.

**Lemma 10.2.1.** Let G be a receive graph with sink x and sources y_1,...,y_n. If g ≤ o, then there is a minimum-length schedule (σ,π) for (G,μ,c,L,o,g,∞), such that for all sources y_i and y_j of G, if π(y_i) = π(x) and π(y_j) ≠ π(x), then σ(y_i) < σ(r_j,π(x),k) for all k ≤ c(y_j).
Proof. Assume \( g \leq o \). Let \((\sigma, \pi)\) be a minimum-length schedule for \((G, \mu, c, L, o, g, \infty)\). We may assume that \( x \) is scheduled on processor 1. Let \( y_i \) and \( y_j \) be two sources of \( G \). Assume \( \pi(y_i) = 1 \) and \( \pi(y_j) \neq 1 \). Assume \( \sigma(y_i) > \sigma(r_{y_j,1,k}) \) for some \( k \leq c(y_j) \). We may assume that \( \sigma(y_i) = \sigma(r_{y_j,1,k}) + o \). Then \( y_i \) can be scheduled at time \( \sigma(r_{y_j,1,k}) \), \( r_{y_j,1,k} \) at time \( \sigma(r_{y_j,1,k}) + \mu(y_i) \) and \( s_{y_j,1,k} \) at time \( \sigma(r_{y_j,1,k}) + \mu(y_i) - o - L \) without violating the feasibility of \((\sigma, \pi)\) or increasing its length. By repeating this step, a minimum-length schedule \((\sigma, \pi)\) for \((G, \mu, c, L, o, g, \infty)\) is constructed in which no source of \( G \) is scheduled after a receive operation on processor \( \pi(x) \). \( \square \)

Lemma 10.2.2 proves that in a minimum-length schedule for a receive graph \( G \) on an unrestricted number of processors, all processors that do not execute the sink of \( G \) need to execute at most one task. Unlike Lemma 10.2.1, this result is true for scheduling with arbitrary \( o \) and \( g \).

**Lemma 10.2.2.** Let \( G \) be a receive graph with sink \( x \) and sources \( y_1, \ldots, y_n \). There is a minimum-length schedule \((\sigma, \pi)\) for \((G, \mu, c, L, o, g, \infty)\), such that for all processors \( p \neq \pi(x) \), at most one source of \( G \) is executed on processor \( p \).

**Proof.** Let \((\sigma, \pi)\) be a minimum-length schedule for \((G, \mu, c, L, o, g, \infty)\). We may assume that \( x \) is scheduled on processor 1. Assume two sources \( y_i \) and \( y_j \) of \( G \) are scheduled on processor \( p \neq 1 \). Let processor \( p' \) be a processor on which no task of \( G \) is executed. Then \( y_j \) can be scheduled on processor \( p' \) at time \( \sigma(y_j) \) and send operation \( s_{y_j,1,k} \) on the same processor at time \( \sigma(s_{y_j,1,k}) \) for all \( k \leq c(y_j) \). This does not violate the feasibility of \((\sigma, \pi)\) nor does it increase its length. By repeating this step, we obtain a minimum-length schedule \((\sigma, \pi)\) for \((G, \mu, c, L, o, g, \infty)\), such that at most one source of \( G \) is executed on processor \( p \) for all processors \( p \neq \pi(x) \). \( \square \)

The following lemma shows that there is a minimum-length schedule for a receive graph \( G \) on an unrestricted number of processors, in which the receive operations corresponding to the sources of \( G \) with a small execution length are scheduled before the receive operations corresponding to the sources of \( G \) with a large execution length.

**Lemma 10.2.3.** Let \( G \) be a receive graph with sink \( x \) and sources \( y_1, \ldots, y_n \). There is a minimum-length schedule \((\sigma, \pi)\) for \((G, \mu, c, L, o, g, \infty)\), such that for all sources \( y_i \) and \( y_j \) of \( G \), if \( \mu(y_i) < \mu(y_j) \) and \( \pi(y_i), \pi(y_j) \neq \pi(x) \), then \( \sigma(y_i, \pi(x), k_i) < \sigma(y_j, \pi(x), k_j) \) for all \( k_i \leq c(y_i) \) and \( k_j \leq c(y_j) \).

**Proof.** Let \((\sigma, \pi)\) be a minimum-length schedule for \((G, \mu, c, L, o, g, \infty)\). We may assume that \( x \) is scheduled on processor 1. From Lemma 10.2.2, we may assume that all processors \( p \neq 1 \) execute at most one task of \( G \). Let \( y_i \) and \( y_j \) be two sources of \( G \) that are not scheduled on processor 1. Assume \( \mu(y_i) < \mu(y_j) \) and \( \sigma(y_i) = \sigma(y_j) = 0 \). Receive operations \( r_{y_i,1,k} \) can start at time \( \mu(y_i) + L + o \) on processor 1, receive operations \( r_{y_j,1,k} \) at time \( \mu(y_j) + L + o \). Assume \( \sigma(r_{y_i,1,k_j}) < \sigma(r_{y_j,1,k_j}) \) for some \( k_i \leq c(y_i) \) and \( k_j \leq c(y_j) \). Then \( r_{y_i,1,k_j} \) can be scheduled at time \( \sigma(r_{y_i,1,k_j}) \) and \( r_{y_j,1,k_j} \) at time \( \sigma(r_{y_j,1,k_j}) \). In addition, send operations \( s_{y_i,1,k} \) and \( s_{y_j,1,k} \) can be scheduled \( L + o \) time units before receive operations \( r_{y_i,1,k} \) and \( r_{y_j,1,k} \), respectively. This does not violate the feasibility of \((\sigma, \pi)\) or increase its length, because all receive operations have length \( o \). By repeating this step, we obtain a minimum-length schedule \((\sigma, \pi)\) for \((G, \mu, c, L, o, g, P)\), such that for all sources \( y_i \) and \( y_j \) of \( G \), if \( \pi(y_i), \pi(y_j) \neq \pi(x) \) and \( \mu(y_i) < \mu(y_j) \), then receive operation \( r_{y_i,\pi(x), k_j} \) is scheduled before receive operation \( r_{y_j,\pi(x), k_j} \) for all \( k_i \leq c(y_i) \) and \( k_j \leq c(y_j) \). \( \square \)

125
Lemma 10.2.4 shows that in a minimum-length schedule for a receive graph $G$ on an unrestricted number of processors, all communication-intensive sources of $G$ may be scheduled on the same processor as the sink of $G$.

**Lemma 10.2.4.** Let $G$ be a receive graph with sink $x$ and sources $y_1, \ldots, y_n$. If $g \leq o$, then there is a minimum-length schedule $(\sigma, \pi)$ for $(G, \mu, c, L, o, g, \infty)$, such that for all sources $y_i$ of $G$, if $\mu(y_i) \leq c(y_i)\sigma$ then $\pi(y_i) = \pi(x)$.

**Proof.** Assume $g \leq o$. Let $(\sigma, \pi)$ be a minimum-length schedule for $(G, \mu, c, L, o, g, \infty)$. We may assume that $x$ is executed on processor 1. From Lemmas 10.2.1 and 10.2.3, we may assume that the sources on processor 1 are scheduled before the receive operations of the sources scheduled on another processor and that for each source $y_i$ of $G$, if $y_i$ is not scheduled on processor 1, then the receive operations $r_{y_i,1,j}$ are scheduled on processor 1 without interruption. Assume $y_i$ is a source of $G$, such that $\mu(y_i) \leq c(y_i)\sigma$ and $\pi(y_i) \neq 1$. We may assume that $\sigma(r_{y_i,1,1}) < \cdots < \sigma(r_{y_i,1,c(y_i)})$. Then $r_{y_i,1,c(y_i)}$ finishes at time $\sigma(r_{y_i,1,1}) + c(y_i)\sigma \geq \sigma(r_{y_i,1,1}) + \mu(y_i)$. Then $y_i$ can be scheduled at time $\sigma(r_{y_i,1,1})$ on processor 1 without increasing the length of $(\sigma, \pi)$ or violating its feasibility. By repeating this step, we obtain a minimum-length schedule $(\sigma, \pi)$ for $(G, \mu, c, L, o, g, \infty)$, such that for all sources $y_i$ of $G$, if $\mu(y_i) \leq c(y_i)\sigma$, then $y_i$ is scheduled on processor $\pi(x)$.

The next lemma proves that it can be determined in polynomial time whether the schedule for a receive graph $G$ in which all tasks of $G$ are scheduled on the same processor is a minimum-length schedule for $G$ on an unrestricted number of processors.

**Lemma 10.2.5.** Let $G$ be a receive graph with sink $x$ and sources $y_1, \ldots, y_n$. If $g \leq o$, then a schedule for $(G, \mu, c, L, o, g, \infty)$ of length $\mu(x) + \sum_{i=1}^{n} \mu(y_i)$ is a minimum-length schedule for $(G, \mu, c, L, o, g, \infty)$ if and only if for all sources $y_i$ of $G$, if $\mu(y_i) > c(y_i)\sigma$, then $\sum_{j=1}^{n} \mu(y_j) \leq (c(y_i) + 1)\sigma + L + \mu(y_i)$.

**Proof.** Assume $g \leq o$. We will prove that a minimum-length schedule for $(G, \mu, c, L, o, g, P)$ has length $\mu(x) + \sum_{i=1}^{n} \mu(y_i)$ if and only if for all computation-intensive sources $y_i$ of $G$, $\sum_{j=1}^{n} \mu(y_j) \leq (c(y_i) + 1)\sigma + L + \mu(y_i)$.

(⇒) Assume a minimum-length schedule for $(G, \mu, c, L, o, g, \infty)$ has length $\mu(x) + \sum_{i=1}^{n} \mu(y_i)$. Let $y_i$ be a source of $G$. Assume $\mu(y_i) > c(y_i)\sigma$. It will be proved by contradiction that $\sum_{j=1}^{n} \mu(y_j) \leq (c(y_i) + 1)\sigma + L + \mu(y_i)$. Suppose $\sum_{j=1}^{n} \mu(y_j) > (c(y_i) + 1)\sigma + L + \mu(y_i)$. Then construct a schedule $(\sigma, \pi)$ for $(G, \mu, c, L, o, g, \infty)$ as follows. Tasks $y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_n$ are scheduled without interruption on processor 1 from time 0 onward. $y_i$ is scheduled on processor 2 at time 0. For all $k \leq c(y_i)$, receive operation $r_{y_i,1,k}$ is scheduled on processor 1 at time $\max \{\sum_{j \neq i} \mu(y_j), \mu(y_i) + o + L\} + (k - 1)\sigma$. For all $k \leq c(y_i)$, send operation $s_{y_i,1,k}$ is scheduled on processor 2 at time $\sigma(r_{y_i,1,k}) - L - \sigma$. $x$ is scheduled immediately after $r_{y_i,1,c(y_i)}$ on processor 1. Then $(\sigma, \pi)$ is a feasible schedule for $(G, \mu, c, L, o, g, \infty)$ of length

$$
\mu(x) + \max \{\mu(y_i) + (c(y_i) + 1)\sigma + o + L, \sum_{j \neq i} \mu(y_j) + c(y_i)\sigma\} < \mu(x) + \sum_{j=1}^{n} \mu(y_j).
$$
Contradiction.

\((\Leftarrow)\) Assume for all sources \(y_i\) of \(G\), if \(\mu(y_i) > c(y_i)\), then \(\sum_{j=1}^{n} \mu(y_j) \leq (c(y_i) + 1) + L + \mu(y_i)\). Let \((\sigma, \pi)\) be a minimum-length schedule for \((G, \mu, c, L, o, g, \infty)\). Since there is a schedule for \((G, \mu, c, L, o, g, \infty)\) of length \(\mu(x) + \sum_{i=1}^{n} \mu(y_i)\), the length of \((\sigma, \pi)\) is at most \(\mu(x) + \sum_{i=1}^{n} \mu(y_i)\). It is proved by contradiction that \((\sigma, \pi)\) has length \(\mu(x) + \sum_{i=1}^{n} \mu(y_i)\). Suppose the length of \((\sigma, \pi)\) is less than \(\mu(x) + \sum_{i=1}^{n} \mu(y_i)\). Then at least one source \(y_i\) of \(G\) is not scheduled on the same processor as \(x\). From Lemma 10.2.4, we may assume that all communication-intensive sources \(y_i\) of \(G\) are scheduled on processor \(\pi(x)\). Hence we may assume that \(\mu(y_i) > c(y_i)\). So \((\sigma, \pi)\) has length at least

\[
\mu(y_i) + (c(y_i) + 1) + L + \mu(x) \geq \mu(x) + \sum_{i=1}^{n} \mu(y_i).
\]

Contradiction.

The properties of minimum-length schedules proved in the preceding lemmas will be used to compute upper bounds on the length of the schedules constructed by Algorithm UNRESTRICTED RECEIVE GRAPH SCHEDULING. Consider an instance \((G, \mu, c, L, o, g, \infty)\), such that \(G\) is a receive graph and \(g \leq o\). Assume \(G\) has sink \(x\) and sources \(y_1, \ldots, y_n\). Algorithm UNRESTRICTED RECEIVE GRAPH SCHEDULING constructs a schedule \((\sigma, \pi)\) for \((G, \mu, c, L, o, g, \infty)\) as follows. The communication-intensive sources of \(G\) and its sink \(x\) are scheduled on processor 1. All computation-intensive sources of \(G\) are scheduled on a separate processor. The receive operations are scheduled after the sources on processor 1, such that if \(\mu(y_i) < \mu(y_j)\) and \(y_i\) and \(y_j\) are not scheduled on processor 1, then receive operations \(r_{y_i, 1, k_i}\) are executed before receive operations \(r_{y_j, 1, k_j}\). Algorithm UNRESTRICTED RECEIVE GRAPH SCHEDULING is presented in Figure 10.1.

**Example 10.2.6.** Consider the instance \((G, \mu, c, 1, 2, 2, \infty)\) shown in Figure 10.2. Algorithm UNRESTRICTED RECEIVE GRAPH SCHEDULING constructs a schedule for \((G, \mu, c, 1, 2, 2, \infty)\) as follows. The set \(Y_1 = \{y_1, y_2, y_3\}\) contains the communication-intensive sources of \(G\). These tasks are scheduled on processor 1 from time 0 onward. The other tasks are scheduled on a separate processor. Since the execution length of \(y_4\) is smaller than that of \(y_5\), the communication operations of \(y_4\) are executed before those of \(y_5\). Sink \(x\) is scheduled on processor 1 after the last receive operation. So Algorithm UNRESTRICTED RECEIVE GRAPH SCHEDULING constructs the schedule for \((G, \mu, c, 1, 2, 2, \infty)\) shown in Figure 10.3.

Now we will prove that Algorithm UNRESTRICTED RECEIVE GRAPH SCHEDULING correctly constructs feasible schedules for receive graphs on an unrestricted number of processors.

**Lemma 10.2.7.** Let \(G\) be a receive graph. Let \((\sigma, \pi)\) be the schedule for \((G, \mu, c, L, o, g, \infty)\) constructed by Algorithm UNRESTRICTED RECEIVE GRAPH SCHEDULING. If \(g \leq o\), then \((\sigma, \pi)\) is a feasible schedule for \((G, \mu, c, L, o, g, \infty)\).
Algorithm \textsc{Unrestricted Receive Graph Scheduling}

\textbf{Input.} An instance \((G, \mu, c, L, o, g, \infty)\), such that \(g \leq o\) and \(G\) is a receive graph with sink \(x\) and sources \(y_1, \ldots, y_n\), such that \(\mu(y_1) \leq \cdots \leq \mu(y_n)\).

\textbf{Output.} A feasible schedule \((\sigma, \pi)\) for \((G, \mu, c, L, o, g, \infty)\).

\begin{algorithm}
1. \texttt{idle}(1) := 0
2. \(p := 1\)
3. for \(i := 1 \text{ to } n\)
   4. do if \(\mu(y_i) \leq c(y_i)\)
      5. then \(\sigma(y_i) := \texttt{idle}(1)\)
      6. \(\pi(y_i) := 1\)
      7. \(\texttt{idle}(1) := \texttt{idle}(1) + \mu(y_i)\)
   8. else \(p := p + 1\)
   9. \(\sigma(y_i) := 0\)
  10. \(\pi(y_i) := p\)
4. for \(i := 1 \text{ to } p\)
  5. do let \(y\) be the sink of \(G\) executed on processor \(i\)
  6. for \(j := 1 \text{ to } c(y)\)
  7. do \(\sigma(r_{y,1,j}) := \max\{\texttt{idle}(1), \mu(y) + L + jo\}\)
  8. \(\pi(r_{y,1,j}) := 1\)
  9. \(\sigma(s_{y,1,j}) := \sigma(r_{y,1,j}) - L - o\)
 10. \(\pi(r_{y,1,j}) := i\)
 11. \(\texttt{idle}(1) := \sigma(r_{y,1,j}) + o\)
12. \(\sigma(x) := \texttt{idle}(1)\)
13. \(\pi(x) := 1\)
\end{algorithm}

\textbf{Figure 10.1.} Algorithm \textsc{Unrestricted Receive Graph Scheduling}

\textbf{Proof.} Assume \(g \leq o\). Let \((\sigma, \pi)\) be the schedule for \((G, \mu, c, L, o, g, \infty)\) constructed by Algorithm \textsc{Unrestricted Receive Graph Scheduling}. Obviously, processor 1 does not execute two tasks or communication operations at the same time. For all sinks \(y\) of \(G\), such that \(\pi(y) \neq 1\), and all \(j \in \{1, \ldots, c(y)\}\), send operation \(s_{y,1,j}\) starts after the completion time of \(y\). Because all processors \(p \neq 1\) execute at most one task, no processor executes two tasks or communication operations at the same time. Since \(g \leq o\) and no two communication operations are executed on the same processor at the same time, there is a delay of at least \(g\) time units between two consecutive send or receive operations on the same processor. In addition, the receive operations are scheduled \(L + o\) time units after the corresponding send operations. So \((\sigma, \pi)\) is a feasible schedule for \((G, \mu, c, L, o, g, \infty)\).

The time complexity of Algorithm \textsc{Unrestricted Receive Graph Scheduling} can be determined as follows. Let \(G\) be a receive graph. Sorting the sources of \(G\) by non-decreasing execution length takes \(O(n\log n)\) time. Clearly, assigning a starting time and a processor to the tasks of \(G\) and the communication operations takes \(O(n)\) time. It is easy to see that the remaining operations take \(O(n)\) time.
Figure 10.2. An instance \((G, \mu, c, 1, 2, 2, \infty)\)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>(y_1)</td>
<td>(y_2)</td>
<td>(y_3)</td>
<td>(r_{y_4,1,1})</td>
<td>(r_{y_5,1,1})</td>
<td>(r_{y_5,1,2})</td>
<td>(x)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(y_4)</td>
<td>(s_{y_4,1,1})</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(y_5)</td>
<td>(s_{y_5,1,1})</td>
<td>(s_{y_5,1,2})</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 10.3. A feasible schedule for \((G, \mu, c, 1, 2, 2, \infty)\)

Lemma 10.2.8. For all instances \((G, \mu, c, L, o, g, \infty)\), such that \(G\) is a receive graph and \(g \leq o\), Algorithm UNRESTRICTED RECEIVE GRAPH SCHEDULING constructs a feasible schedule for \((G, \mu, c, L, o, g, \infty)\) in \(O(n \log n)\) time.

Now we will prove that Algorithm UNRESTRICTED RECEIVE GRAPH SCHEDULING is a 3-approximation algorithm. Let \(G\) be a receive graph with sink \(x\) and sources \(y_1, \ldots, y_n\), such that \(\mu(y_1) \leq \cdots \leq \mu(y_n)\). Assume \(g \leq o\). Let \((\sigma, \pi)\) be the schedule for \((G, \mu, c, L, o, g, \infty)\) constructed by Algorithm UNRESTRICTED RECEIVE GRAPH SCHEDULING. Let \(y_{i_1}, \ldots, y_{i_k}\) be the sources of \(G\) that are not scheduled on processor 1. Then \(\mu(y_{i_j}) > c(y_{i_j})o\) for all \(j \leq k\). We will assume that \(i_1 \leq \cdots \leq i_k\). Let \(y_{i_{k+1}}, \ldots, y_{i_n}\) be the sources of \(G\) scheduled on processor 1, such that \(i_{k+1} \leq \cdots \leq i_n\).

Then \(x\) is scheduled immediately after receive operation \(r_{y_{i_k},1,c(y_{i_k})}\). If processor 1 is not idle before time \(\sigma(x)\), then \((\sigma, \pi)\) has length

\[
\sum_{j=k+1}^{n} \mu(y_{i_j}) + \sum_{j=1}^{k} c(y_{i_j})o + \mu(x).
\]

Otherwise, there is a \(j \in \{1, \ldots, k\}\), such that receive operation \(r_{y_{i_j},1,1}\) starts at time \(\mu(y_{i_j}) + L + o\) and processor 1 executes receive operations \(r_{y_{i_l},1,i}\), such that \(l \geq j\) and \(i \leq c(y_{i_l})\), without interruption from time \(\mu(y_{i_l}) + L + o\) until time \(\sigma(x)\). In this case, \((\sigma, \pi)\) has length

\[
\mu(y_{i_j}) + \sum_{l=j}^{k} c(y_{i_l})o + L + o + \mu(x).
\]

129
Let $\ell$ the length of $(\sigma, \pi)$. Then

$$\ell \leq \mu(x) + \max \left\{ \sum_{j=1}^{k} c(y_{ij})o + \sum_{j=k+1}^{n} \mu(y_{ij}), \max_{1 \leq j \leq k} \left( \mu(y_{ij}) + \sum_{l=1}^{k} c(y_{ij})o + L + o \right) \right\}.$$  

Let $\ell^*$ be the length of a minimum-length schedule for $(G, \mu, c, L, o, g, \infty)$. Clearly, $\ell^* \geq \mu(x) + \mu(y)$ for all sources $y$ of $G$. In addition, for each source $y_i$ of $G$, either $y_i$ itself or $c(y_i)$ receive operations are scheduled on the same processor as $x$ in a feasible schedule for $(G, \mu, c, L, o, g, \infty)$. Hence

$$\ell^* \geq \mu(x) + \sum_{i=1}^{n} \min \{ \mu(y_i), c(y_i) \}.$$  

Consequently,

$$\ell \leq \mu(x) + \max \left\{ \sum_{j=1}^{k} c(y_{ij})o + \sum_{j=k+1}^{n} \mu(y_{ij}), \max_{1 \leq j \leq k} \left( \mu(y_{ij}) + \sum_{l=1}^{k} c(y_{ij})o + L + o \right) \right\} \leq \max \{ \ell^*, \ell^* + \ell^* + L + o \} = 2\ell^* + L + o.$$  

If the length of a minimum-length schedule for $(G, \mu, c, L, o, g, \infty)$ equals $\mu(x) + \sum_{j=1}^{n} \mu(y_j)$, then this can be checked in linear time using Lemma 10.2.5. In that case, we can construct a minimum-length schedule for $(G, \mu, c, L, o, g, \infty)$ by scheduling all tasks on one processor. Otherwise, in a minimum-length schedule for $(G, \mu, c, L, o, g, \infty)$, there is a sink that is scheduled on a different processor than $x$. Hence $\ell^* \geq \mu(x) + 2o + L$ and $\ell \leq 2\ell^* + L + o \leq 3\ell^*$. Hence we have proved the following result.

**Theorem 10.2.9.** There is an algorithm with an $O(n \log n)$ time complexity that constructs feasible schedules for instances $(G, \mu, c, L, o, g, \infty)$, such that $G$ is a receive graph and $g \leq o$, with length at most $3\ell^*$, where $\ell^*$ is the length of a minimum-length schedule for $(G, \mu, c, L, o, g, \infty)$.

Note that if $L$ and $o$ are bounded by a constant, then Algorithm UNRESTRICTED RECEIVE GRAPH SCHEDULING is an approximation algorithm with asymptotic approximation ratio two.

### 10.2.2 A restricted number of processors

In this section, an approximation algorithm is presented that constructs schedules for receive graphs on a restricted number of processors. Consider an instance $(G, \mu, c, L, o, g, P)$, such that $G$ is a receive graph, $g \leq o$ and $P \neq \infty$. Algorithm RESTRICTED RECEIVE GRAPH SCHEDULING constructs a schedule for $(G, \mu, c, L, o, g, P)$. Like Algorithm UNRESTRICTED RECEIVE GRAPH SCHEDULING, the communication-intensive sources of $G$ will be scheduled on the same processor as its sink, the other sources of $G$ can be scheduled on any processor. A schedule for $(G, \mu, c, L, o, g, P)$ is constructed by extending a feasible schedule for the subgraph of $G$ induced by the set of computation-intensive sources of $G$. Algorithm RESTRICTED RECEIVE GRAPH SCHEDULING is presented in Figure 10.4.
Algorithm \textsc{Restricted receive graph scheduling}

\textbf{Input}: An instance \((G, \mu, c, L, o, g, P)\), such that \(g \leq o\), \(P \neq \infty\) and \(G\) is a receive graph with sink \(x\) and sources \(y_1, \ldots, y_n\).

\textbf{Output}: A feasible schedule \((\sigma, \pi)\) for \((G, \mu, c, L, o, g, P)\).

1. \(Y_1 := \{y_i \mid \mu(y_i) \leq c(y_i) o\}\)
2. \(Y_2 := \{y_i \mid \mu(y_i) > c(y_i) o\}\)
3. let \((\sigma, \pi)\) be a feasible schedule for \((G|\bar{Y_1}|, \mu, c, L, o, g, P)\)
4. \(\text{for } p := 1 \text{ to } P\)
5. \(\text{do } \text{idle}(p) := \max\{\sigma(y) + \mu(y) \mid y \in Y_2 \land \pi(y) = p\}\)
6. \(Y_{2,p} := \{y \in Y_2 \mid \pi(y) = p\}\)
7. \(\text{assume } \text{idle}(1) \leq \cdots \leq \text{idle}(P)\)
8. \(\text{for } y \in Y_1\)
9. \(\text{do } \sigma(y) := \text{idle}(1)\)
10. \(\text{idle}(1) := \text{idle}(1) + \mu(y)\)
11. \(\text{for } p := 2 \text{ to } P\)
12. \(\text{do for } j := 1 \text{ to } c(y)\)
13. \(\text{do } \sigma(r_{y,1,j}) := \max\{\text{idle}(1), \text{idle}(p) + L + jo\}\)
14. \(\pi(r_{y,1,j}) := 1\)
15. \(\sigma(s_{y,1,j}) := \sigma(r_{y,1,j}) - L - o\)
16. \(\pi(s_{y,1,j}) := p\)
17. \(\text{idle}(1) := \sigma(r_{y,1,j}) + o\)
18. \(\text{idle}(p) := \sigma(s_{y,1,j}) + o\)
19. \(\sigma(x) := \text{idle}(1)\)
20. \(\pi(x) := 1\)

Figure 10.4. Algorithm \textsc{Restricted receive graph scheduling}

Example 10.2.10. Consider the instance \((G, \mu, c, 1, 2, 2, 2)\) shown in Figure 10.5. Apart from the number of processors, this instance equals the one shown in Figure 10.2. The set \(Y_1 = \{y_1, y_2, y_3\}\) contains the communication-intensive sources of \(G\). These tasks are scheduled on processor 1. Assume Algorithm \textsc{Restricted receive graph scheduling} starts with a schedule in which \(y_4\) starts at time 0 on processor 1 and \(y_5\) at time 0 on processor 2. Then \(y_1, y_2\) and \(y_3\) are scheduled on the same processor as \(y_4\), because the execution length of \(y_4\) is smaller than that of \(y_5\). Receive operations \(r_{y,1,i}\) are scheduled after \(y_3\) on processor 2. \(x\) is executed after the last receive operation on processor 1. So Algorithm \textsc{Restricted receive graph scheduling} constructs the schedule for \((G, \mu, c, 1, 2, 2, 2)\) shown in Figure 10.6.

Now we will prove that Algorithm \textsc{Restricted receive graph scheduling} correctly constructs feasible schedules for receive graphs on a restricted number of processors.

Lemma 10.2.11. Let \(G\) be a receive graph. Let \((\sigma, \pi)\) be the schedule for \((G, \mu, c, L, o, g, P)\) constructed by Algorithm \textsc{Restricted receive graph scheduling}. If \(g \leq o\), then \((\sigma, \pi)\) is a feasible schedule for \((G, \mu, c, L, o, g, P)\).
Determined as follows. Let $(G, \mu, c, 1, 2, 2)$.

Figure 10.5. An instance $(G, \mu, c, 1, 2, 2)$

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_4$</td>
<td>$y_1$</td>
<td>$y_2$</td>
<td>$y_3$</td>
<td>$r_{y_3,1,1}$</td>
<td>$r_{y_3,1,2}$</td>
<td>$x$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$y_5$</td>
<td>$s_{y_5,1,1}$</td>
<td>$s_{y_5,1,2}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 10.6. A feasible schedule for $(G, \mu, c, 1, 2, 2)$

**Proof.** Assume $g \leq o$ and $G$ has sink $x$ and sources $y_1, \ldots, y_n$. Define $Y_1 = \{y_i | \mu(y_i) \leq c(y_i)o\}$ and $Y_2 = \{y_i | \mu(y_i) > c(y_i)o\}$. Let $(\sigma_0, \pi_0)$ be a feasible schedule for $(G, Y_2, \mu, c, L, o, g, P)$. Algorithm **RESTRICTED RECEIVE GRAPH SCHEDULING** extends $(\sigma_0, \pi_0)$ to a schedule $(\sigma, \pi)$ for $(G, \mu, c, L, o, g, P)$. It is easy to see that there is a delay of exactly $L$ time units between the completion time of a send operation and the starting time of the corresponding receive operation. Because $g \leq o$ and all receive operations are scheduled on processor 1, there is a delay of at least $g$ time units between a pair of consecutive send and receive operations on the same processor. So $(\sigma, \pi)$ is a feasible schedule for $(G, \mu, c, L, o, g, P)$.

The time complexity of Algorithm **RESTRICTED RECEIVE GRAPH SCHEDULING** can be determined as follows. Let $G$ be a receive graph with sink $x$ and sources $y_1, \ldots, y_n$. Let $Y_1 = \{y_i | \mu(y_i) \leq c(y_i)o\}$ and $Y_2 = \{y_i | \mu(y_i) > c(y_i)o\}$. $Y_1$ and $Y_2$ can be computed in $O(n)$ time. Let $(\sigma_0, \pi_0)$ be a feasible schedule for $(G, Y_2, \mu, c, L, o, g, P)$. Sorting the processors by non-decreasing maximum completion time takes $O(P\log P)$ time. Assigning a starting time and a processor to every task of $Y_1$ takes $O(n)$ time. It is easy to see that the starting times and processors for the communication operations can be assigned in linear time as well. So Algorithm **UNRESTRICTED RECEIVE GRAPH SCHEDULING** uses $O(n\log n)$ time apart from the time needed to construct $(\sigma_0, \pi_0)$.

**Lemma 10.2.12.** For all instances $(G, \mu, c, L, o, g, P)$, such that $G$ is a receive graph and $g \leq o$, if a feasible schedule for $n$ incomparable tasks can be constructed in $O(T(n))$ time, then Algorithm **RESTRICTED RECEIVE GRAPH SCHEDULING** constructs a feasible schedule for $(G, \mu, c, L, o, g, P)$ in $O(T(n) + n\log n)$ time.

Consider an instance $(G, \mu, c, L, o, g, P)$, such that $g \leq o$ and $G$ is a receive graph with sink $x$ and sources $y_1, \ldots, y_n$. Define $Y_1 = \{y_i | \mu(y_i) \leq c(y_i)o\}$ and $Y_2 = \{y_i | \mu(y_i) > c(y_i)o\}$. Let
Let \( \ell^* \) be the length of a minimum-length schedule for \((G, \mu, c, L, o, g, P)\) and \(\ell\) the length of \((\sigma, \pi)\). Because any schedule on a restricted number of processors can be viewed as a schedule on an unrestricted number of processors,

\[
\ell^* \geq \mu(x) + \sum_{i=1}^{n} \min\{\mu(y_i), c(y_i)\} = \mu(x) + \sum_{y \in Y_1} \mu(y) + \sum_{y \in Y_2} c(y) o.
\]

In addition, \(\ell^* \geq \mu(x) + \frac{1}{p} \sum_{i=1}^{n} \mu(y_i)\). If the schedule in which all tasks are scheduled on one processor is not of minimum length, then \(\ell^* \geq \mu(x) + L + 2o\).

Let \(y^*\) be a source of \(Y_2\) with a maximum completion time. Then its completion time equals the length of \((\sigma_0, \pi_0)\). It is possible that every task in \(Y_1\) is scheduled after \(y^*\). Hence

\[
\ell \leq \sigma(y^*) + \mu(y^*) + \sum_{y \in Y_1} \mu(y) + \sum_{y \in Y_2, \pi(y) \neq 1} c(y) o + L + o + \mu(x).
\]

Assume \(\ell_0\) is the length of \((\sigma_0, \pi_0)\) and \(\ell_0^*\) is the length of a minimum-length schedule for \((G[\gamma_2], \mu, c, L, o, g, P)\). Clearly, \(\ell_0 < \ell^*\). Assume \(\ell_0 \leq \rho \ell_0^*\). Then

\[
\ell \leq \sigma(y^*) + \mu(y^*) + \sum_{y \in Y_1} \mu(y) + \sum_{y \in Y_2, \pi(y) \neq 1} c(y) o + L + o + \mu(x)
\]

\[
\leq \rho \ell_0^* + \ell^* + L + o
\]

So if \(\ell^* > \mu(x) + \sum_{i=1}^{n} \mu(y_i)\), then \(\ell \leq (\rho + 1) \ell^*\). If the schedule in which all tasks are executed on one processor is of minimum length, then its length is at most \(\ell\). If \((\sigma, \pi)\) is longer than \(\mu(x) + \sum_{i=1}^{n} \mu(y_i)\), then replace \((\sigma, \pi)\) by the schedule in which all tasks are executed by the same processor. Then this schedule is at most \(\rho + 2\) times as long as a minimum-length schedule for \((G, \mu, c, L, o, g, P)\).

Note that if \(L\) and \(o\) are bounded by a constant, then Algorithm RESTRICTED RECEIVE GRAPH SCHEDULING is an approximation algorithm with asymptotic approximation ratio \(\rho + 1\).

There are many algorithms for scheduling incomparable tasks on \(P\) identical processors. Using Graham’s List scheduling algorithm [38, 39], we obtain an algorithm that constructs schedules on \(P\) processors that are at most \(4 - \frac{2}{p}\) times as long as a minimum-length schedule on \(P\) processors [92].

By using different algorithms, we obtain better approximation bounds. Coffman et al. [14] presented Algorithm MULTIFIT. \(k\) iterations of this algorithm construct schedules on \(P\) processors that are at most \(\frac{13}{14} + 2^{-k}\) time as long as a minimum-length schedule on \(P\) processors [94]. \(k\) iterations of Algorithm MULTIFIT take \(O(n \log n + kn \log P)\) time. Hence we have proved the following result.

**Theorem 10.2.13.** For all constant \(k \in \mathbb{Z}^+\), there is an algorithm with an \(O(n \log n)\) time complexity that constructs feasible schedules for instances \((G, \mu, c, L, o, g, P)\), such that \(G\) is a receive graph and \(g \leq o\), with length at most \((\frac{13}{14} + 2^{-k}) \ell^*\), where \(\ell^*\) is the length of a minimum-length schedule for \((G, \mu, c, L, o, g, P)\).
Hochbaum and Shmoys [45] presented a polynomial approximation scheme for scheduling incomparable tasks on identical processors. For each \( k \in \mathbb{Z}^+ \), a schedule on \( P \) processors that is at most \( 1 + \frac{1}{k+1} \) times as long as the length of a minimum-length schedule on \( P \) processors can be constructed in \( O((k + 1)n(k+1)\log(k+1)) \) time using this approximation scheme [62]. Hence we have proved the following result.

**Theorem 10.2.14.** For all constant \( k \in \mathbb{Z}^+ \), there is an algorithm with an \( O((k + 1)n(k+1)\log(k+1)) \) time complexity that constructs feasible schedules for instances \( (G, \mu, c, L, o, g, P) \), such that \( G \) is a receive graph and \( g \leq o \), with length at most \( (3 + \frac{1}{k+1}) \ell^* \), where \( \ell^* \) is the length of a minimum-length schedule for \( (G, \mu, c, L, o, g, P) \).

### 10.3 A polynomial special case

In Section 10.2, two approximation algorithms for scheduling receive graphs were presented. Constructing minimum-length schedules for receive graphs on an unrestricted number of processors is a strongly NP-hard optimisation problem. Kort and Trystram showed that if \( g \) does not exceed \( o \) and all sources of a receive graph have the same execution length, then a minimum-length schedule for this receive graph on an unrestricted number of processors can be constructed in polynomial time. In this section, this result is improved: it is proved that if all sources have the same execution length, then a minimum-length schedule on an unrestricted number of processors can be constructed in polynomial time even if \( g \) exceeds \( o \).

Consider an instance \( (G, \mu, c, L, o, g, \infty) \), such that \( G \) is a receive graph with sink \( x \) and sources \( y_1, \ldots, y_n \). Assume \( \mu(y_1) = \cdots = \mu(y_n) = \mu \). There is a minimum-length schedule for \( (G, \mu, c, L, o, g, \infty) \) in which the tasks and the communication operations are scheduled on at most \( n \) processors. From Lemma 10.2.2, we may assume that all processors, expect that one that executes \( x \), execute at most one source of \( G \). To obtain a minimum-length schedule for \( (G, \mu, c, L, o, g, \infty) \), the sources \( y \) with minimum \( c(y) \) should be scheduled on another processor than \( x \). Assume \( c(y_1) \leq \cdots \leq c(y_n) \). In a minimum-length \( m \)-processor schedule for \( (G, \mu, c, L, o, g, \infty) \), \( x \) is scheduled on processor 1, \( y_i \) on processor \( i+1 \) for all \( 1 \leq i \leq m-1 \) and the remaining sources of \( G \) on processor 1. Sources \( y_1, \ldots, y_{m-1} \) are completed at time \( \mu \). Then \( C_m = \sum_{i=1}^{m-1} c(y_i) \) receive operations have to be scheduled on processor 1.

The sinks \( y_1, \ldots, y_n \) have to be scheduled before the first receive operation or between the receive operations on processor 1. There is a delay of at least \( \max\{o, g\} - o \) time units between two consecutive receive operations on processor 1. Let \( \alpha(o, g) = \frac{\max\{o, g\} - o}{\mu} \). Because there is a delay of at least \( \max\{o, g\} - o \) time units between a pair of consecutive receive operations, at least \( \lceil \alpha(o, g) \rceil \) sources can be scheduled between a pair of consecutive receive operations. If at least \( \lceil \alpha(o, g) \rceil \) sources are scheduled between two consecutive receive operations, then we may assume that processor 1 is not idle between these receive operations. We may assume that at most \( \lceil \alpha(o, g) \rceil \) sources are scheduled between two consecutive receive operations: if more than \( \lceil \alpha(o, g) \rceil \) sources are scheduled between two consecutive receive operations, then the first of these receive operations can be scheduled at a later time without increasing the schedule length.

The length of an \( m \)-processor schedule depends on the number of sources executed between the receive operations. Let \( k \) be this number. We may assume that \( k \leq (C_m - 1) \lceil \alpha(o, g) \rceil \) and \( k \leq
Let $\ell_{m,k}$ be the minimum length of an $m$-processor schedule for $(G,\mu,c,L,o,g,P)$ in which $k$ sources are scheduled between the receive operations. In such an $m$-processor schedule, the first receive operation can start at time

$$\max\{ (n-k-(m-1))\mu, \mu+L+o \}.$$ 

If $[\alpha(o,g)]$ sources are scheduled between two consecutive receive operations, then the starting times of these receive operations differ $[\alpha(o,g)]\mu+o$. This is more than when the receive operations are scheduled with as little delay as possible. So each time $[\alpha(o,g)]$ sources are scheduled between two consecutive receive operations, the starting time of $x$ increases by $inc(o,g)$.

Hence $\ell_{m,k}$ equals

$$\max\{ (n-k-(m-1))\mu, \mu+L+o \} + (C_m-1)\max\{o,g\} + o + inc_{m,k}(o,g) + \mu(x),$$

where $inc_{m,k}(o,g) = \max\{0,(n-(C_m-1)[\alpha(o,g)])\}inc(o,g)$.

Let $\ell^n_m = \min_k \ell_{m,k}$. Then $\ell^n_m$ is the length of a minimum-length $m$-processor schedule for $(G,\mu,c,L,o,g,P)$. Since $c(y_i)$ is bounded by a constant for all sources $y_i$ of $G$, $\ell^n_m$ can be computed in $O(n)$ time. The length $\ell^*$ of a minimum-length schedule for $(G,\mu,c,L,o,g,P)$ equals $\min_{1 \leq m \leq n} \ell^n_m$. This can be computed in $O(n^2)$ time. If $\ell^* = \ell_{m,k}$, then $m$ and $k$ can be used to construct a schedule of length $\ell^*$ in linear time. Hence we have proved the following result.

**Theorem 10.3.1.** There is an algorithm with an $O(n^2)$ time complexity that constructs minimum-length schedules for instances $(G,\mu,c,L,o,g,\infty)$, such that $G$ is a receive graph and there is a positive integer $\mu$, such that $\mu(y) = \mu$ for all sources $y$ of $G$.

If $\max\{o,g\} - o$ is divisible by $\mu$, then a minimum-length schedule for $(G,\mu,c,L,o,g,\infty)$ can be constructed more efficiently. Let $G$ be a receive graph with sink $x$ and sources $y_1,\ldots,y_n$, such that $c(y_1) \leq \cdots \leq c(y_n)$. Assume $\max\{o,g\} - o$ is divisible by $\mu$. Then we may assume that in a minimum-length $m$-processor schedule for $(G,\mu,c,L,o,g,\infty)$, exactly $k_m = \min\{n-m+1,(C_m-1)[\alpha(o,g)]\}$ sources of $G$ are scheduled between the receive operations on processor 1 and that the remaining sources are scheduled before the first receive operation. Because $inc_{m,k_m}(o,g)$ equals zero, the length of such a schedule equals

$$\max\{ (n-k_m-(m-1))\mu, \mu+L+o \} + (C_m-1)\max\{o,g\} + o + \mu(x).$$

The values $\ell_{m,k_m}$ can be computed in linear time, because we assumed that $c(y_i)$ is bounded by a constant for all sources $y_i$ of $G$. Let $\ell^* = \min_{1 \leq m \leq n} \ell_{m,k_m}$. Assume $\ell^* = \ell_{m,k_m}$. Using $m$, a schedule for $(G,\mu,c,L,o,g,\infty)$ of length $\ell^*$ can be constructed in $O(n)$ time. Because $c(y_i)$ is bounded by a constant for all sources $y_i$ of $G$, sorting the sources of $G$ by non-decreasing message lengths tasks $O(n)$ time. Hence we have proved the following result.
Theorem 10.3.2. There is an algorithm with an $O(n)$ time complexity that constructs minimum-length schedules for instances $(G, \mu, c, L, o, g, \infty)$, such that $G$ is a receive graph and there is a positive integer $\mu$, such that $\mu(y) = \mu$ for all sources $y$ of $G$ and $\max\{o, g\} - o$ is divisible by $\mu$.

Both Theorem 10.3.1 and 10.3.2 improve a result of Kort and Trystram [55], who presented an algorithm that constructs minimum-length schedules for receive graphs with sources of equal length in $O(n^2)$ time if $g$ does not exceed $o$.

10.4 Concluding remarks

The problem of scheduling send and receive graphs in the LogP model was studied in Chapters 9 and 10, respectively. Although send and receive graphs can be transformed into each other by reversing the arcs, scheduling send graphs is less complicated than scheduling receive graphs. This is due to the fact that we consider a common data semantics. For receive graphs, there is no difference between a common data semantics and an independent data semantics. For send graphs, there is a difference. Scheduling send graphs under an independent semantics is the same as scheduling receive graphs: messages have to be sent for all sinks that are not scheduled on the same processor as the source. Scheduling send graphs under a common data semantics is less complicated, because at most one set of messages has to be sent to any processor.

Like for scheduling send graphs, there are a lot of possible generalisations. If $g \geq o$, then we can prove properties of minimum-length schedules similar to those proved in Section 10.2.1. However, these results do not allow us to prove that Algorithms UNRESTRICTED RECEIVE GRAPH SCHEDULING and RESTRICTED RECEIVE GRAPH SCHEDULING are approximation algorithms with a constant approximation ratio for scheduling with arbitrary $o$ and $g$. This is due to the fact that the number of communication operations that must be scheduled in an $m$-processor schedule for a receive graph depends on the processor assignment. Because the number of communication operations in an $m$-processor schedule for a send graph is independent of the processor assignment, we were able to present a 2-approximation algorithm for scheduling send graphs with arbitrary $o$ and $g$.

It is unknown whether minimum-length schedules on a restricted number of processors can be constructed in polynomial time if all sources have the same execution length. Kort and Trystram proved that if all sources have the same execution length and this length exceeds $\max\{g, 2o + L\}$, then a minimum-length schedule on two processors can be constructed in polynomial time. They also proved that if $c(y)$ is the same for all sources $y$ of a receive graph, then a minimum-length schedule for this receive graph on an unrestricted number of processors can be constructed in polynomial time.

Like for send graphs, the structure of minimum-length schedules for more general inforests is far more complicated than that of minimum-length schedules for receive graphs. Hence it is difficult to construct approximation algorithms with a constant approximation ratio for more general inforests. In Chapter 11, two algorithms are presented for scheduling general inforests in the LogP model.