

---

### Abstract

We study a particular scheduling setting in which a set of  $n$  jobs with individual release times and deadlines has to be scheduled across  $m$  homogeneous processors while minimizing the consumed energy. Idle processors can be turned off so as to save energy, while turning them on requires a fixed amount of energy. For the special case of a single processor, the greedy algorithm Left-to-Right guarantees an approximation factor of 2. We generalize this simple greedy policy to the case of multiple processors and show that the energy costs are still bounded by  $2\text{OPT} + P$ . Our algorithm has a running time of  $(n+m) \log(d^*)F$  where  $d^*$  is the largest deadline and  $F$  the costs of a maximum flow calculation for checking feasibility of an instance.

# Contents

<b>1</b>	<b>Algorithm</b>	<b>3</b>
<b>2</b>	<b>Structure of the PLTR-Schedule</b>	<b>5</b>
2.1	Preliminary Definitions . . . . .	5
2.2	Critical set of time slots . . . . .	5
2.3	Definitions based on critical sets . . . . .	7
<b>3</b>	<b>Modification of our Schedule</b>	<b>9</b>
3.1	Augmentation . . . . .	9
3.2	Realignment . . . . .	9
3.3	Invariants for Realignment . . . . .	9

# 1 Algorithm

---

**Algorithm 1** Parallel Left-to-Right
 

---

```

 $m_t \leftarrow m$ 
 $l_t \leftarrow 0$ 
for  $k \leftarrow m$  to 1 do
   $t \leftarrow 0$ 
  while  $t < d^*$  do
     $t \leftarrow \text{KEEPIDLE}(k, t)$ 
     $t \leftarrow \text{KEEPACTIVE}(k, t)$ 
function  $\text{KEEPIDLE}(k, t)$ 
  search for maximal  $t' \geq t$  s.t. exists feasible schedule with  $m_{t''} = k - 1 \forall t'' \in [t, t')$ 
   $m_{t'} \leftarrow k - 1 \forall t'' \in [t, t')$ 
  return  $t'$ 
function  $\text{KEEPACTIVE}(k, t)$ 
  search for maximal  $t' \geq t$  s.t. exists feasible schedule with  $l'_{t''} = \max\{k, l_{t''}\} k - 1 \forall t'' \in [t, t')$ 
   $m_{t'} \leftarrow k - 1 \forall t'' \in [t, t')$ 
  return  $t'$ 

```

---

Formal Problem definition, notation

Formulate and format keep-active, keepidle nicer.

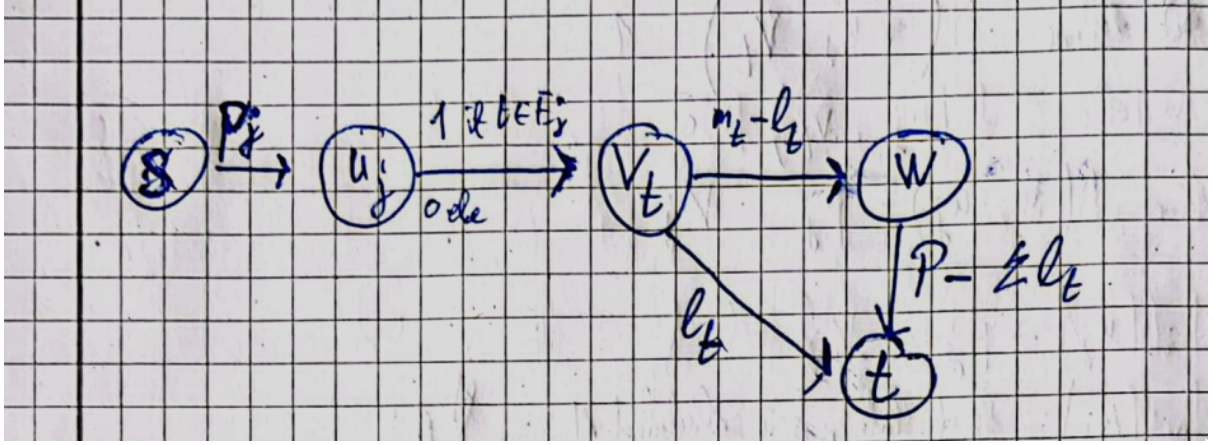
Properly define scheduling problem with lower and upper bounds  $l_t, m_t$ .Precisely relate  $l_t, m_t$  returned by Algorithm to assignment of jobs to processors, time slots.

Figure 1.1: The Flow-Network for checking feasibility of a scheduling instance with lower and upper bounds  $l_t$  and  $m_t$  for the number of active processors at  $t$ .

**Lemma 1.** *There exists a feasible solution to a scheduling instance with lower and upper bounds  $l_t, m_t$  if and only if the maximum  $s$ - $t$  flow in the corresponding flow network depicted in Figure 1 has value  $P$ .*

*Proof.* Let  $f$  be a  $s$ - $t$  flow of value  $|f| = P$ . We construct a feasible schedule from  $f$  respecting the lower and upper bounds given by  $l_t$  and  $m_t$ . For every  $j \in J$  and  $t \in T$ , if  $f(u_j, v_t) = 1$ , then schedule  $j$  at slot  $t$ . Since  $|f| = P$  and the capacity  $c(\{s\}, V \setminus \{s\}) = P$ , we have  $f_{in}(u_j) = p_j$  for every  $j \in J$ . Hence  $f_{out}(u_j) = \sum_{t \in E_j} f_{in}(v_t) = p_j$ . Hence every job  $j$  is scheduled in  $p_j$  distinct time slots.

The schedule respects the upper bounds  $m_t$ , since  $c(v_t, w) + c(v_t, t) \leq m_t - l_t + l_t$  and for every  $t$  at most  $m_t$  jobs are scheduled at  $t$ .

The schedule respects the lower bounds  $l_t$ , since  $c(V \setminus \{t\}, \{t\}) = P$  and hence  $f(v_t, t) = l_t$  for every slot  $t$ . By flow conservation we then have  $f_{in}(v_t) \geq l_t$  which implies that at least  $l_t$  jobs are scheduled at every slot  $t$ .

For the other direction consider a feasible schedule respecting the lower and upper bounds  $l_t, m_t$ . We construct a flow  $f$  of value  $P$  and show that it is maximal.

If  $j$  is scheduled at slot  $t$  and hence  $t \in E_j$ , define  $f(u_j, v_t) = 1$ , otherwise  $f(u_j, v_t) = 0$ . Define  $f(s, u_j) = p_j$  for every  $j \in J$ . Hence we have  $f_{in}(u_j) = p_j$  and  $f_{out}(u_j)$  must be  $p_j$  since this corresponds to the number of distinct time slots in which  $j$  is scheduled. Define  $f(v_t, t) = l_t$  for every slot  $t$ . Define  $f(v_t, w) = f_{in}(v_t) - l_t$ . We have  $f(v_t, w) \leq m_t - l_t$  since  $f_{in}(v_t)$  corresponds to the number of jobs scheduled at  $t$ , which is at most  $m_t$ . We also have  $f_{out}(v_t) = f_{in}(v_t) - l_t + l_t = f_{in}(v_t)$ .

Ambiguity of  $t$  used for time slots and sink in flow-network: Use  $\alpha$ - $\omega$  flowBriefly define  $c$  for edge capacities and cuts.Introduce notation for number of jobs scheduled at  $t$ ?

Define  $f(w, t) = P - \sum_t l_t$ . Then  $f_{in}(w) = \sum_t f_{in}(v_t) - l_t = \sum_t |\{j \in J \mid j \text{ scheduled at } t\}| - \sum_t l_t$ . Since the schedule is feasible, this corresponds to  $f_{in}(w) = P - \sum_t l_t = f_{out}(w)$ .  $\square$

## 2 Structure of the PLTR-Schedule

### 2.1 Preliminary Definitions

**Definition 2.** For schedule  $S$ , we define the volume  $v_S(j, Q)$  of job  $j \in J$  in a set  $Q \subseteq T$  of time slots as the number of time slots of  $Q$  for which  $j$  is scheduled at by  $S$ .

**Definition 3.** We define the forced volume  $fv(j, Q)$  of job  $j \in J$  in a set  $Q \subseteq T$  of time slots as the number of time slots of  $Q$  for which  $j$  has to be scheduled in every feasible schedule, i.e.

$$fv(j, Q) = \max\{0; p_j - |E_j \setminus Q|\}.$$

**Definition 4.** We define the unnecessary volume  $uv_S(j, Q)$  of job  $j \in J$  in a set  $Q \subseteq T$  of time slots as the amount of volume which does not have to be scheduled during  $Q$ , i.e.

$$uv_S(j, Q) = v_S(j, Q) - fv(j, Q).$$

**Definition 5.** We define the possible volume  $pv(j, Q)$  of job  $j \in J$  in a set  $Q \subseteq T$  of time slots as the maximum amount of volume which  $j$  can be feasibly scheduled in  $Q$ , i.e.

$$pv(j, Q) = \min\{p_j, |E_j \cap Q|\}.$$

**Definition 6.** We define the space  $space_S(j, Q)$  of job  $j \in J$  in a set  $Q \subseteq T$  for schedule  $S$  as the number of additional time slots, which  $j$  can be scheduled in  $Q$ , i.e.

$$space_S(j, Q) = pv(j, Q) - v_S(j, Q).$$

Since the corresponding schedule  $S$  will always be clear from context, we drop the subscript for  $v, uv$  and  $space$ . We extend our volume definitions to sets  $J' \subseteq J$  of jobs by summing over all  $j \in J'$ , i.e.

$$v(J', Q) = \sum_{j \in J'} v(j, Q).$$

If the first parameter is dropped, we refer to the whole set  $J$ , i.e.  $v(Q) = v(J, Q)$ . Clearly we have for every feasible schedule, every  $Q, j$  that  $fv(j, Q) \leq v(j, Q) \leq pv(j, Q)$ .

**Definition 7.** We define the density  $\phi(Q)$  for a set  $Q \subseteq T$  as the average amount of processing volume which has to be completed in every slot of  $Q$ , i.e.  $\phi(Q) = fv(J, Q)/|Q|$ . We also define  $\hat{\phi}(Q) = \max_{Q' \subseteq Q} \phi(Q')$ .

If  $\hat{\phi}(Q) > k - 1$ , then clearly at least  $k$  processors are required in some time slot  $t \in Q$  for every feasible schedule.

**Definition 8.** We define the deficiency  $def(Q)$  of a set  $Q \subseteq T$  of time slots as the difference between the amount of volume which has to be completed in  $Q$  and the processing capacity available in  $Q$ , i.e.  $def(Q) = fv(Q) - \sum_{t \in Q} m_t$ .

**Definition 9.** We define the excess  $exc(Q)$  of a set  $Q \subseteq T$  of time slots as the difference between the processor utilization required in  $Q$  and the amount of processing volume available in  $Q$ , i.e.  $exc(Q) = \sum_{t \in Q} l_t - pv(Q)$ .

### 2.2 Critical set of time slots

**Lemma 10.** For every  $s$ - $t$  cut  $(S, \bar{S})$  we have at least one of the following two lower bounds for the capacity  $c(S)$  of the cut:  $c(S) \geq P - def(Q(S))$  or  $c(S) \geq P - exc(Q(\bar{S}))$ , where  $Q(S) := \{t \mid v_t \in S\}$ .

*Proof.* Let  $(S, \bar{S})$  be a  $s$ - $t$  cut, let  $J(S) := \{j \mid u_j \in S\}$ . If  $w \notin S$ , consider the contribution of every node of  $S$  to the capacity of the cut.

- Node  $s$ :  $\sum_{j \in J(\bar{S})} p_j$ .
- Node  $u_j$ :  $|\{v_t \in \bar{S} \mid t \in E_j\}| = |E_j \setminus Q(S)| \geq p_j - fv(j, Q(S))$

- Node  $v_t$ :  $l_t + m_t - l_t = m_t$

The inequality for node  $u_j$  follows since  $\text{fv}(j, Q(S)) = \max\{0, p_j - |E_j \setminus Q(S)|\}$ . In total, we can lower bound the capacity with

$$\begin{aligned} c(S) &\geq \sum_{j \in J(\bar{S})} p_j + \sum_{j \in J(S)} p_j - \text{fv}(j, Q(S)) + \sum_{t \in Q(S)} m_t \\ &= P - \text{fv}(J(S), Q(S)) + \sum_{t \in Q(S)} m_t \\ &\geq P - \text{def}(Q(S)). \end{aligned}$$

If  $w \in S$ , again consider the contribution of every node of  $S$  to the capacity of the cut.

- Node  $s$ :  $\sum_{j \in J(\bar{S})} p_j \geq \text{pv}(J(\bar{S}), Q(\bar{S}))$ .
- Node  $u_j$ :  $|E_j \setminus Q(S)| = |E_j \cap Q(\bar{S})| \geq \text{pv}(j, Q(\bar{S}))$
- Node  $v_t$ :  $l_t$
- Node  $w$ :  $P - \sum_t l_t$

In total, we can lower bound the capacity with

$$\begin{aligned} c(S) &\geq P - \sum_{t \in Q(\bar{S})} l_t + \text{pv}(Q(\bar{S})) \\ &= P - \text{exc}(Q(\bar{S})) \end{aligned}$$

□

**Lemma 11.** *A scheduling instance with lower and upper bounds  $l_t$  and  $m_t$  is feasible if and only if  $\text{def}(Q) \leq 0$  and  $\text{exc}(Q) \leq 0$  for every  $Q \subseteq T$ .*

*Proof.* If  $\text{def}(Q) > 0$  for some  $Q$ , then some upper bounds  $m_t$  cannot be met. If  $\text{exc}(Q) > 0$  for some  $Q$ , then some lower bound  $l_t$  cannot be met. For the direction from right to left, consider an infeasible scheduling instance with lower and upper bounds. By Lemma 1 we have that the maximum flow  $f$  for this instance has value  $|f| < P$ . Hence, there must be a  $s$ - $t$  cut  $(S, \bar{S})$  of capacity  $c(S) < P$ . Lemma 10 now implies that  $\text{def}(Q(S)) > 0$  or  $\text{exc}(Q(\bar{S})) > 0$ . □

**Lemma 12.** *For every time slot  $t \in T$  for which some processor  $k \in [m]$  is activated in  $S_{\text{pltr}}$ , there exists a set  $Q \subseteq T$  of time slots with  $t \in Q$ ,*

$$\begin{aligned} \text{fv}(Q) &= v(Q), \\ v(t') &\geq k - 1 && \text{for } t' \in Q \text{ and} \\ v(t') &\geq k && \text{for } t' \in Q \text{ with } t' \geq t. \end{aligned}$$

*Proof.* Suppose for contradiction there is some activation  $t \in T$  of processor  $k \in [m]$  and no such  $Q$  exists for  $t$ . We show that pltr would have extended the idle interval on processor  $k$  which ends at  $t$ . Consider the step in pltr when  $t$  was the result of keepIdle on processor  $k$  and the corresponding lower and upper bounds  $m_{t'}, l_{t'}$  for  $t' \in T$  right after the calculation of  $t$  and the corresponding update of the bounds by keepIdle. We modify the bounds by decreasing  $m_t$  by 1. Note that at this point  $m_{t'} \geq k$  for every  $t' > t$  and  $m_{t'} \geq k - 1$  for every  $t'$ .

Consider  $Q \subseteq T$  s.t.  $t \in Q$  and  $\text{fv}(Q) < v(Q)$ . Before our modification we had  $m_Q := \sum_{t' \in Q} m_{t'} \geq v(Q) > \text{fv}(Q)$ . The inequality  $m_Q \geq v(Q)$  here follows since the upper bounds  $m_{t'}$  are monotonically decreasing during pltr. After our modification we still have  $m_Q \geq \text{fv}(Q)$ .

Consider  $Q \subseteq T$  s.t.  $t \in Q$  and  $v(t') < k - 1$  for some  $t'$ . At the step in pltr considered by us, we hence have  $m_{t'} \geq k - 1 > v(t')$  and therefore before our decrement of  $m_t$  we had  $m_Q > v(Q) \geq \text{fv}(Q)$  which implies  $m_Q \geq \text{fv}(Q)$  after the decrement of  $m_t$ .

Finally, consider  $Q \subseteq T$  s.t.  $t \in Q$  and  $v(t') < k$  for some  $t' > t$ . Again at the step in pltr considered by us, we have  $m_{t'} \geq k > v(t')$  which implies  $m_Q \geq \text{fv}(Q)$  after our decrement of  $m_t$ .

If for  $t$  no  $Q$  exists as characterized in the proposition,  $t$  cannot have been the result of keepIdle at this step in pltr, which is a contradiction. □

**Definition 13.** We call such  $Q$  for activations  $t$  of processor  $k$  characterized by Lemma 12 tight set  $Q_t$  over activation  $t$  of processor  $k$ .

**Definition 14.** A critical set  $C_t \subseteq T$  over an activation  $t$  is the maximum of the set of tight sets  $Q_t$  over activation  $t$  in regard to the density  $\phi$ , i.e.

$$C_t := \operatorname{argmax}\{\phi(Q) \mid Q \subseteq T \text{ is tight set over } t\}.$$

As the set of these critical sets  $C_t$  for fixed  $t$  is closed under union, for the sake of uniqueness, we take  $C_t$  to be the inclusion-maximal critical set over activation  $t$ .

### 2.3 Definitions based on critical sets

**Definition 15.** We define a total order  $\preceq$  on the set of critical sets  $C_t$  over all activations  $t$ . For activations  $t, t' \in T$  of processors  $k$  and  $k'$  respectively, we define  $C_t \preceq C_{t'}$  if and only if  $k < k'$  or  $k = k'$  and  $t \geq t'$ . In other words,  $\preceq$  is the same order in which pltr calculates the activations: from Top-Left to Bottom-Right.

Use  $\preceq$  instead of  $\preceq$  here

**Definition 16.** Let  $\operatorname{rank} : \{C_t\} \rightarrow \mathbb{N}$  be a mapping to the natural numbers corresponding to  $\preceq$ , i.e.

$$\operatorname{rank}(C_t) \leq \operatorname{rank}(C_{t'}) \Leftrightarrow C_t \preceq C_{t'}$$

**Definition 17.** Let  $\operatorname{crit} : \{C_t\} \rightarrow [m]$  be a mapping to the processors s.t.

$$\operatorname{crit}(C_t) = c \Leftrightarrow c \text{ is the highest processor activated at } t$$

**Definition 18.** We extend these definitions to general time slots  $t \in T$ .

$$\begin{aligned} \operatorname{rank}(t) &:= \begin{cases} \max\{\operatorname{rank}(C) \mid t \in C\} & \text{if } t \in C \text{ for some critical set } C \\ 0 & \text{otherwise} \end{cases} \\ \operatorname{crit}(t) &:= \begin{cases} \max\{\operatorname{crit}(C) \mid t \in C\} & \text{if } t \in C \text{ for some critical set } C \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

We also extend the definitions to intervals  $D \subseteq T$ .

$$\begin{aligned} \operatorname{rank}(D) &:= \max\{\operatorname{rank}(t) \mid t \in D\} \\ \operatorname{crit}(D) &:= \max\{\operatorname{crit}(t) \mid t \in D\} \end{aligned}$$

**Definition 19.** Let  $C$  be a critical set. A nonempty interval  $V \subseteq T$  is a valley of  $\operatorname{rank}(C)$  if  $C \sim V$  and  $V$  is inclusion maximal. Let  $C_1, \dots, C_l$  be the (maximal) intervals of  $C$ . A nonempty interval  $V$  is a valley of  $C$  if  $V$  is exactly the interval between  $C_a$  and  $C_{a+1}$  for some  $a < l$ , i.e.  $V = [\max C_a + 1, \min C_{a+1} - 1]$ .

**Definition 20.** For a critical set  $C$ , an interval  $D$  spans  $C$  if  $D \cap C$  contains only full subintervals of  $C$  and at least one subinterval of  $C$ . The left valley  $V_l$  of  $C$  and an interval  $D$  spanning  $C$  is the valley of  $C$  ending at  $\min(C \cap D) - 1$  (if such a valley of  $C$  exists). The right valley  $V_r$  of  $C$  and an interval  $D$  spanning  $C$  is the valley of  $C$  starting at  $\max(C \cap D) + 1$  (if such a valley of  $C$  exists).

**Definition 21.** For a valley  $V$ , we define the jobs  $J(V) \subseteq J$  as all jobs which are scheduled by  $S_{\text{pltr}}$  in every  $t \in V$ .

**Lemma 22.** For every critical set  $C$  with  $c := \operatorname{crit}(C)$ , every interval  $D$  spanning  $C$ : if  $\phi(C \cap D) \leq c - \delta$  for some  $\delta \in \mathbb{N}$ , then  $V_l$  or  $V_r$  is defined and  $|J_{V_l}| + |J_{V_r}| \geq \delta$ , where we take  $|J_V| := 0$  if  $V$  does not exist.

*Proof.* By choice of  $C$  as critical set with  $c = \operatorname{crit}(C)$  we have  $v(C \cap D) \geq (c - 1) \cdot |C \cap D|$ . If this inequality is fulfilled strictly, i.e. if  $v(C \cap D) > (c - 1) \cdot |C \cap D|$ , then with  $\operatorname{fv}(C \cap D)/|C \cap D| \leq c - \delta$  we directly get  $uv(C \cap D)/|C \cap D| > \delta - 1$ . This implies that there are at least  $\delta$  jobs  $j$  scheduled in  $C \cap D$  with  $uv(j, C \cap D) > 0$ . Such jobs must have  $E_j \cap (C \setminus D) \neq \emptyset$  and hence at least one of  $V_l$  and  $V_r$  for  $C$  and  $D$  must exist and the jobs must be contained in  $J_{V_l}$  or  $J_{V_r}$ .

Provide a rough visual sketch here

Make math more readable in this whole proof, e.g. by using fractions and display math or by replacing division by multiplication

If on the other hand we have equality, i.e.  $v(C \cap D) = (c - 1) \cdot |C \cap D|$ , then let  $t$  be the activation of processor  $c$  for which  $C$  is critical set for. Since  $v(t) > c - 1$ , we must have  $t \notin C \cap D$ . By the same argument as before, we have that if  $fv(C \cap D)/|C \cap D| \leq c - \delta$ , then  $uv(C \cap D)/|C \cap D| \geq \delta + 1$ . Now suppose that there is no job  $j$  scheduled in  $C$  s.t.  $space(j, C \cap D) > 0$ . Then  $fv(C \setminus D) = v(C \setminus D) > (c - 1) \cdot |C \cap D|$ . Hence  $fv(C \setminus D) = v(C \setminus D) > (c - 1) \cdot |C \cap D|$ . We then get  $\phi(C \setminus D) = v(C \setminus D) > (c - 1) \cdot (C \cap D)$  since by case assumption  $t \in C \setminus D$ . In conclusion,  $C \setminus D$  is still a tight set over  $t$  but has higher density than  $C$ , contradicting the choice of  $C$ . Therefore, there must exist a job  $j$  scheduled in  $C$  with  $space(j, C \cap D) > 0$  and hence

$$\frac{uv(C \cap D) + space(j, C \cap D)}{|C \cap D|} > \delta - 1,$$

which again implies that there must be at least  $\delta$  jobs scheduled in  $C$  with an execution interval intersecting both  $C \setminus D$  and  $C \cap D$ . This implies that the left valley  $V_l$  or the right valley  $V_r$  of  $C$  and  $D$  exist and that at least  $\delta$  jobs are contained in  $J_{V_l}$  or  $J_{V_r}$ .  $\square$



### 3 Modification of our Schedule

We modify the schedule  $S_{\text{pltr}}$  returned by our algorithm in two steps. The first step augments specific processors with auxiliary active slots, s.t. in every critical set  $C$ , there are at least the first  $\text{crit}(C)$  processors active. Recall that for the single processor ltr algorithm, the crucial property for the approximation guarantee was that every idle interval of  $S_{\text{OPT}}$  can intersect at most 2 distinct idle intervals of  $S_{\text{ltr}}$ . The second modification step is more involved and establishes this crucial property on every processor  $k \in [m]$  by making use of Lemma 22. It is important to note that these modification steps are only done for the sake of the analysis. By making sure that the costs can only be increased by this modification, we get an upper bound for the costs of  $S_{\text{pltr}}$ .

Give some high level explanation that we realign the jobs of  $J_{V_l}, J_{V_r}$  to higher processors where necessary.

#### 3.1 Augmentation

We transform  $S_{\text{pltr}}$  into  $S_{\text{aug}}$  by adding for every  $t$  with  $k := \text{crit}(t) \geq 2$  and  $v(t) = k - 1$  an auxiliary active slot on processor  $k$ . This auxiliary active slot does not count towards the volume.

**Lemma 23.** *In  $S_{\text{aug}}$ , in every  $t \in T$  with  $\text{crit}(t) \geq 2$  processors  $1, \dots, \text{crit}(t)$  are active.*

*Proof.* The property directly follows from our choice of the critical sets, the definition of  $\text{crit}(t)$  and the construction of  $S_{\text{aug}}$ .  $\square$

#### 3.2 Realignment

---

##### Algorithm 2 Realignment of $S_{\text{aug}}$

---

```

Res(V)  $\leftarrow 2|J_V|$  for every valley  $V$ 
for  $k \leftarrow m$  to 1 do
    fill( $T$ )
    Res(V)  $\leftarrow$  Res(V)  $- 1$  for every  $V$  s.t. some  $V'$  with  $V' \cap V \neq \emptyset$  was closed on processor  $k$ 
function fill( $k, V$ )
    if  $\text{crit}(V) \leq 1$  then
        return
    let  $C$  be critical set s.t.  $C \sim V$ 
    while exists active interval  $A \subseteq V$  on processor  $k$  with  $A \sim V$  and  $\hat{\phi}(A) \leq k - 1$  do
        let  $V_l, V_r$  be the left and right valley for  $C$  and interval  $A$  (if  $A$  spans  $C$ )
        if  $V_l$  exists and Res( $V_l$ )  $> 0$  then
            close( $k, V_l$ )
        else if  $V_r$  exists and Res( $V_r$ )  $> 0$  then
            close( $k, V_r$ )
    for every valley  $V' \subseteq V$  of  $C$  which has not been closed on  $k$  do
        fill( $k, V'$ )
function close( $k, V$ )
    for every  $t \in V$  which is idle on processor  $k$  do
        if processors  $1, \dots, k - 1$  idle at  $t$  then
            introduce new auxiliary active slot on processor  $k$  at time  $t$ 
        else
            move active slot at time  $t$  of highest processor among  $1, \dots, k - 1$  to processor  $k$  at  $t$ 

```

---

#### 3.3 Invariants for Realignment

**Lemma 24.** *For an arbitrary step during the realignment of  $S_{\text{aug}}$  let  $k_V$  be the highest processor s.t.*

- *processor  $k_V$  is not fully filled yet, i.e.  $\text{fill}(k_V, T)$  has not yet returned,*
- *no  $V' \supseteq V$  has been closed on  $k_V$  so far and*
- *there is a (full) active interval  $A \subseteq V$  on processor  $k_V$ .*

*We take  $k_V := 0$  if no such processor exists. At every step in realignment of  $S_{\text{aug}}$  the following invariants hold.*

1. If  $\phi(C \cap D) \leq k_V - \delta$  for some  $\delta \in \mathbb{N}$  and some interval  $D \subseteq T$  spanning  $C$ , then the left valley  $V_l$  or the right valley  $V_r$  of  $C, D$  exists and  $\text{Res}(V_l) + \text{Res}(V_r) \geq 2\delta$ .
2. For every  $t \in C \cap V$ , processors  $1, \dots, k_V$  are active at  $t$ .
3. Every active interval  $A \subseteq V$  on processor  $k_V$  with  $A \sim V$  spans  $C$ .

*Proof.* We show properties 1 and 2 via structural induction on the realigned schedule  $S_{\text{real}}$ . Then we show that invariant 2 implies invariant 3. For the induction base, consider  $S_{\text{aug}}$  and let  $V$  be an arbitrary valley in  $S_{\text{aug}}$  and  $C$  the critical set with  $C \sim V$ ,  $\text{crit}(V) := c$ .

We have  $k_V \leq c$ , otherwise  $V$  contains a full active interval on processor  $k_V > c$  and hence also an activation  $t \in V$  of processor  $k_V$ , which by construction of  $S_{\text{aug}}$  would have  $\text{crit}(t) = k_V > c$ . This is a direct contradiction to  $\text{crit}(V) = \max_{t \in V} \text{crit}(t) = c$ .

The second invariant now follows since by construction of  $S_{\text{aug}}$  and our choice of  $C$  we have for every  $t \in C$  that processors  $1, \dots, k_V, \dots, c$  are active at  $t$ .

For the first invariant, let  $D$  be an interval spanning  $C$  with  $\phi(C \cap D) \leq k_V - \delta$  for some  $\delta \in \mathbb{N}$ . With  $k_V \leq c$  we get  $\phi(C \cap D) \leq c - \delta$  and hence by Lemma 22, we have that the left valley  $V_l$  or the right valley  $V_r$  of  $C$  and  $D$  exists and  $|J_{V_l}| + |J_{V_r}| \geq \delta$ . With the initial definition of  $\text{Res}(V)$  we get the desired lower bound of  $\text{Res}(V_l) + \text{Res}(V_r) \geq 2\delta$ .

Now suppose that invariants 1 and 2 hold at all steps of the realignment up to a specific next step. Let  $V$  again be an arbitrary valley of  $\text{crit}(V) \geq 2$  and  $k$  the processor currently being filled. Let furthermore  $k_V, k'_V$  be the critical processor for  $V$  before and after, respectively, the next step in the realignment. We consider four cases for the next step in the realignment.

**Case 1:** Some  $V' \supseteq V$  is closed on processor  $k$ . Then no valley  $W$  intersecting  $V$  has been closed so far on  $k$ . Also, since close only moves the active slot highest active processor below  $k$ , we know that the stair property holds within  $V$  on processors  $1, \dots, k$ . We show that the closing of  $V'$  on  $k$  reduces the critical processor of  $V$  by at least 1, i.e.  $k'_V \leq k_V - 1$ . If  $k_V = k$ , then  $V' \supseteq V$  is closed on processor  $k_V$  and hence by definition we have  $k'_V \leq k_V - 1$ . If  $k_V < k$ , suppose for contradiction that  $k_V \leq k'_V \leq k$ , where  $k'_V \leq k$  again by definition since  $V' \subseteq V$  is closed on processor  $k$ .

Let  $A \subseteq V$  be a full active interval on  $k_V$  before the close of  $V'$ . We show that  $A \subset V$ , i.e. that there must be some  $t \in V$  idle on  $k_V$  before the close and hence by the stair-property processors  $k_V, \dots, k$  idle at  $t$  before the close. If  $V' \supseteq V$  is closed, clearly  $V \subset T$  by the choice of  $V_l$  and  $V_r$  as valleys of some critical set in the realignment definition. Hence we know that  $\min V - 1 \in T$  and  $\max V + 1 \in T$ . We show that  $t := \min V - 1$  must be active on  $k_V$  before the close. Let  $W \supseteq V$  be the valley with  $W \sim t$  and  $t \in W$ . We know that  $W \supseteq V$  since  $W \sim t \succ V$  since  $V$  is a valley. By our case assumption, no  $W' \subseteq W$  can have been closed on processor  $k$  so far. With  $W \supseteq V$  and the definition of  $k_W$  we get  $k_W \geq k_V$ , where  $k_W$  is the critical processor of  $W$  before the close. Our induction hypothesis now implies that processors  $1, \dots, k_V, \dots, k_W$  are active at  $t$  before the close. For  $A \subseteq V$  to be a (full) active interval on  $k_V$  before the close, we hence must have  $\min V \notin A$ . We know by definition of the realignment and function close, that for every  $k'$  with  $k_V \leq k' < k$  and every  $t \in V$ :

- If  $t$  was idle on  $k'$  before the close, then  $t$  is still idle on  $k'$  after the close (definition of close,  $k' < k$ ).
- If  $t$  was idle on  $k_V$  before the close, then  $t$  was idle on  $k'$  before (stair-property) and hence  $t$  is still idle on  $k'$  after the close.
- If  $t$  was part of full active interval  $A \subset V$  on  $k_V$  before the close, then  $t$  was idle on  $k_V + 1$  before the close (choice of  $k_V$ ). Hence  $t$  was idle on  $k$  before (stair-property) and hence  $t$  is idle on  $k_V$  after the close.

Taken together, for  $t \in V$  to be active on  $k'$  after the close,  $t$  must have been active on  $k'$  before the close (definition close,  $k' < k$ ) and  $t$  cannot have been part of a full active interval  $A \subseteq V$ . Hence  $t \in A$  for some *partial* active interval  $A \subseteq V$  on  $k'$  before the close. For  $A' \subseteq V$  to be a full active interval on  $k'_V$  after the close (with  $k_V \leq k'_V < k$ ), we must have  $A' \subseteq A$ .

define stair property

Double check if this is true (or if we only have one of the two guaranteed): Since  $V$  must be valley of some  $C$ , we should have both  $0 \notin V$  and  $d^* \notin V$ .

properly define partial active intervals

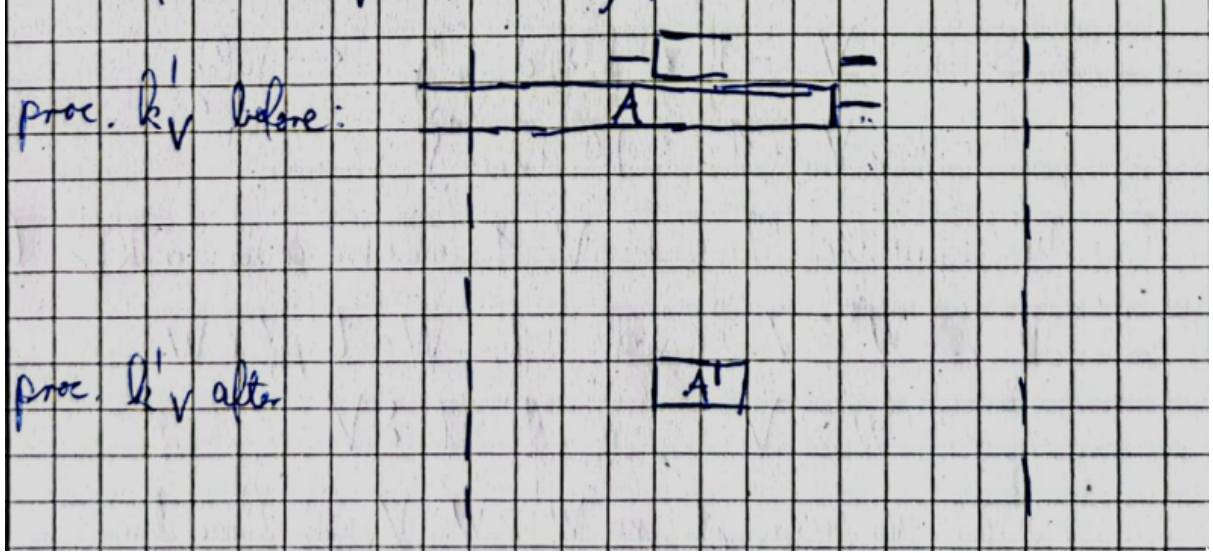


Figure 3.1: The situation for case 1 in Lemma 24.

Hence there must have been an active interval  $A'' \subseteq [\min A', \max A]$  on processor  $k'_V + 1 > k_V$  before the close, which contradicts the definition of  $k_V < k$ . In conclusion, we have  $k'_V \leq k_V - 1$  which allows us to prove our invariants 1 and 2. If  $\phi(C \cap D) \leq k'_V - \delta$  for some  $\delta \in \mathbb{N}$  and some interval  $D$  spanning  $C$ , then  $\phi(C \cap D) \leq k_V - (\delta + 1)$  and hence by induction hypothesis the left valley  $V_l$  or the right valley  $V_r$  for  $C, D$  exists and  $\text{Res}(V_l) + \text{Res}(V_r) \leq 2(\delta + 1)$  both before and after the close. Our induction hypothesis also implies that for every  $t \in C \cap V$  processors  $1, \dots, k_V$  are active before the close. Since at most the uppermost active slot is moved by close, after the close of  $V'$  we still have processors  $1, \dots, k_V - 1 \geq k'_V$  active.

**Case 2:** Some  $V' \subset V$  is closed on processor  $k$ . Again, no  $V'' \supseteq V$  can have been close on processor  $k$  so far. We show that  $k_V = k \geq k'_V$ , i.e. that the critical processor of  $V$  before the close of  $V'$  is the processor currently being filled. Let  $W$  be the valley for which  $V'$  is closed, i.e.  $V'$  is closed during  $\text{fill}(k, W)$ . We must have  $W \supset V'$  and therefore no  $W' \supseteq W$  has been closed on  $k$  so far. Also, for  $V'$  to be closed in  $\text{fill}(k, W)$ , there must be some active interval  $A \subseteq W$  on  $k$  before the close, hence  $k_W = k$ . Since  $V' \subset V$  and  $V' \subset W$ ,  $V$  and  $W$  intersect ( $V' \neq \emptyset$  by definition of  $V'$  as valley). If  $V \prec W$ , then by choice of  $V'$  as valley of  $C_W$  we must have  $V \subseteq V'$ , which contradicts our case assumption. Therefore  $V \succeq W$  and  $V \supseteq W$  implying  $k_V \leq k_W = k$  and since processor  $k + 1$  is already filled before the close we have  $k_V = k \geq k'_V$ .

For invariant 1, again let  $\phi(C \cap D) \leq k'_V - \delta \leq k_V - \delta$  for some  $\delta \in \mathbb{N}$  and some interval  $D$  spanning  $C$ . Our induction hypothesis implies that the left valley  $V_l$  or the right valley  $V_r$  of  $C, D$  exists and that both before and after the close we have  $\text{Res}(V_l) + \text{Res}(V_r) \geq 2\delta$ .

For invariant 2, observe that  $V' \cap C = \emptyset$  since  $V' \prec C$  by our case assumption. Our invariant 2 now directly follows from the induction hypothesis and  $k'_V \leq k_V$ .

**Case 3:** Some  $V'$  with  $V' \cap V = \emptyset$  is closed on processor  $k$ . We first show that  $\min V - 1 \notin V'$  and symmetrically  $\max V + 1 \notin V'$ . Consider  $t := \min V - 1$ . We know  $t \in T$  since  $V'$  is valley of some critical set. By choice of  $V$  and  $t$  we must have  $t \succ V$ . If  $t \in V'$ , we would have  $V' \succ V$  and hence  $V' \supseteq V$ , which contradicts our case assumption. Symmetrically, we know that  $\max V + 1 \notin V'$ . Therefore the close of  $V'$  does not modify the schedule within  $[\min V - 1, \max V + 1]$ , implying that no partial active interval in  $V$  before the close can become a full active interval. In summary we have  $k_V = k'_V$  and invariants 1 and 2 follow as in case 2.

**Case 4:** The call to  $\text{fill}(k, T)$  returns and  $\text{Res}(V')$  is decreased by 1 for every  $V'$  such that some valley intersecting  $V'$  has been closed during  $\text{fill}(k, T)$ . First observe that the schedule itself does not change by this step but processor  $k$  is now fully filled, which implies  $k'_V \leq k_V$ . Invariant 2 then follows directly from the induction hypothesis.

We consider two subcases. If during  $\text{fill}(k, T)$ , no valley  $V'$  intersecting  $V$  was closed on  $k$ , then  $\text{Res}(V)$  does not change and Invariant 1 follows from the induction hypothesis and  $k'_V \leq k_V$ . If

introduce this notation for critical set  $C_V$  of valley  $V$ , i.e.  $C_V \sim V$ .

on the other hand some valley  $V'$  intersecting  $V$  was closed on  $k$  during  $\text{fill}(k, T)$ , then  $\text{Res}(V)$  is decreased by 1 to  $\text{Res}'(V) := \text{Res}(V) - 1$ . As seen in cases 1 to 3,  $k_V$  decreases monotonically during  $\text{fill}(k, T)$ . Consider the schedule right before the first valley  $V'$  intersecting  $V$  is closed on  $k$ . Let  $k_V^0$  be the critical processor for  $V$  at this point of the realignment and  $k_V^1$  the critical processor right after  $V'$  is closed.

We have  $k_V' \leq k_V^0 - 1$ : If  $V' \supseteq V$ , then as argued in case 1  $k_V^1 \leq k_V^0 - 1$  and hence  $k_V' \leq k_V \leq k_V^1 \leq k_V^0 - 1$ . If  $V' \subset V$ , then as argued in case 2 we have  $k_V^0 = k$ . Since  $\text{fill}(k, T)$  returns in the next step of our case assumption, we have  $k_V' \leq k - 1$  and hence  $k_V' \leq k_V^0 - 1$ . Invariant 2 now follows by our (strong) induction hypothesis. If  $\phi(C \cap D) \leq k_V' - \delta$  then  $\phi(C \cap D) \leq k_V^0 - (\delta + 1)$  and hence the left valley  $V_l$  or the right valley  $V_r$  of  $C, D$  exists and before the close  $\text{Res}(V_l) + \text{Res}(V_r) \geq 2(\delta + 1)$ . Since  $\text{Res}$  is decreased by at most 1, we have after the close  $\text{Res}'(V_l) + \text{Res}'(V_r) \geq 2\delta$ .

We conclude by showing that Invariant 2 implies Invariant 3. Let  $V$  be an arbitrary valley during the realignment of  $S_{\text{aug}}$  and  $A \subseteq V$  an active interval on processor  $k_V$  with  $A \sim V$ . Let  $C$  be the critical set with  $C \sim V$ . Note that  $A \sim V$  implies that  $A$  intersects  $C$ . Assume for contradiction that  $A$  does not span  $C$ . The min  $A$  is within subinterval of  $C$  or symmetrically max  $A$  within subinterval of  $C$ . We assume the first case, i.e.  $t := \min A - 1 \in C$  and  $\min A \in C$ . The second case follows by symmetry. If  $t \in V$ , then time slot  $t$  is active on processor  $k_V$  by Invariant 2. Therefore,  $A$  cannot be a (full) active interval on processor  $k_V$ , contradicting the choice of  $A$ . If  $t = \min V - 1$ , then consider valley  $W$  with  $t \in W$  and  $t \sim W$ . We must have  $W \supset V$ ,  $W \succ V$  and  $t \in C_W$ . Therefore  $k_W \geq k_V$  and Invariant 2 implies that  $t = \min A - 1$  is active on processor  $k_V$ , again contradicting the choice of  $A$  as full active interval on processor  $k_V$ .  $\square$

**Lemma 25.** *The result  $S_{\text{real}}$  of the realignment of  $S_{\text{aug}}$*

*Proof.* The Invariants in Lemma 24 imply that whenever we are inside the while-loop of  $\text{fill}$ , the corresponding active interval  $A$  indeed spans the corresponding critical set  $C$  with  $C \sim A$ , hence  $V_r$  or  $V_l$  exists and there is sufficient  $\text{Res}$  to close some valley.

Since in the while-loop of  $\text{fill}(k, V)$  the active interval  $A \subseteq V$  on  $k_V$  always spans  $C$  if  $V \sim C$  (Invariant 3), the left valley  $V_l$  and the right valley  $V_r$  of the critical set  $C$  and interval  $A$  are properly defined. Also since  $\phi(A) \leq k - 1$ , Invariant 1 implies that  $V_l$  or  $V_r$  exists and that there is sufficient  $\text{Res}$  such that one of the two valleys of  $C$  is closed in this iteration. This reduces the number of idle intervals on processor  $k$  by at least 1, since Invariant 2 implies that  $V_l$  or  $V_r$  cannot end strictly within an idle interval on  $k$ . Hence all terms in the realignment are well defined and the realignment terminates.  $\square$

**Lemma 26.** *For every processor  $k \in [m]$ , every active interval  $A$  on processor  $k$  in  $S_{\text{real}}$ ,  $\hat{\phi}(A) > k - 1$ .*

*Proof.* We show that  $\text{fill}(k, T)$  establishes the property on  $k$ . The claim then follows since  $\text{fill}(k, T)$  does not change the schedules of processors above  $k$ . Since in  $\text{fill}(k, T)$ , we always close valleys for active intervals  $A$  on  $k$  spanning a corresponding critical set  $C$ , we know that on processor  $k$ , active intervals are only extended. Let  $A \subseteq V$  be an active interval on processor  $k$  in  $S_{\text{real}}$  with  $A \sim V$  and  $\text{crit}(A) \geq 2$ . No valley  $W \supseteq V$  can have been closed on  $k$  since otherwise there would be no  $A \subseteq V$  in  $S_{\text{real}}$ . Consider then the point in  $\text{fill}(k, V)$  when the while-loop terminates. Clearly at this point all  $A' \subseteq V$  with  $A' \sim V$  on processor  $k$  have  $\hat{\phi}(A') > k - 1$ . There must also be at least one such  $A'$  at this point for  $A \subseteq V$  to be an active interval on  $k$  in  $S_{\text{real}}$  with  $A \sim V$ . In particular, one such  $A'$  must have  $A' \subseteq A$ , which directly implies  $\hat{\phi}(A) \geq \hat{\phi}(A') > k - 1$ .  $\square$

**Lemma 27.** *The realignment of  $S_{\text{aug}}$  does not create new activation times but may only change the corresponding processor being activated, i.e. if  $t \in T$  is an activation of some processor  $k$  in  $S_{\text{real}}$ , then  $t$  is also an activation of some processor  $k'$  in  $S_{\text{aug}}$ .*

*Proof.* TODO  $\square$

**Lemma 28.**

*Proof.*  $\square$

## References