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 $2\,\mathrm{OPT} + P$. Our algorithm has a running time of $O(Fn \log d)$, where d is the largest deadline and F the time required by a maximum-flow calculation for checking the feasibility of an instance.

Contents

Define problem, notation

Replace *active, active intervals* with *busy* everywere?

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**
            $t \leftarrow$ KEEPIDLE(k, t)
            $t \leftarrow$ KEEPACTIVE(k, t)
    function 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 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'
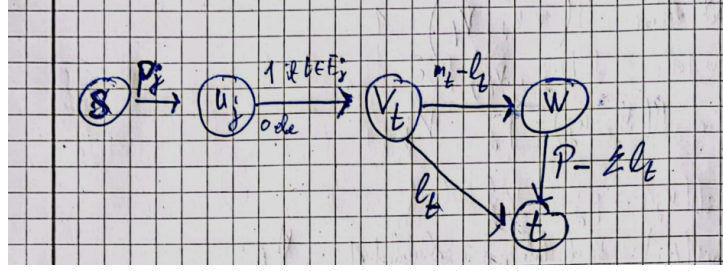
Figure 1.1: The Flow-Network for checking feasibility of a scheduling instance with lower and upper bounds l_t and m_t for the number of active processors at t.

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 feasbile 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 $f_{out}(v_t) = f_{in}(v_t) - l_t + l_t = f_{in}(v_t)$.

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

Use $T = T$ here?

Definition 2. *For schedule S, we define the* volume $\mathrm{v}_S(j, Q)$ *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 $\mathrm{fv}(j, Q)$ *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.*

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

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

$$\mathrm{uv}_S(j, Q) = \mathrm{v}_S(j, Q) - \mathrm{fv}(j, Q).$$

Definition 5. *We define the* possible volume $\mathrm{pv}(j, Q)$ *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.*

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

Definition 6. *We define the* space $\mathrm{space}_S(j, Q)$ *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.*

$$\mathrm{space}_S(j, Q) = \mathrm{pv}(j, Q) - \mathrm{v}_S(j, Q).$$

Since the corresponding schedule S will always be clear from context, we drop the subscript for $\mathrm{v, uv}$ and space. We extend our volume definitions to single timeslots $t \in T$ and to sets $J' \subseteq J$ of jobs by summing over all $j \in J'$, i.e.

$$\mathrm{v}(J', Q) \coloneqq \sum_{j \in J'} \mathrm{v}(j, Q),$$

$$\mathrm{v}(t) \coloneqq \mathrm{v}(J, \{t\}).$$

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

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. $\phi(Q) = \mathrm{fv}(J, Q)/|Q|$. 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 $\mathrm{def}(Q)$ *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. $\mathrm{def}(Q) = \mathrm{fv}(Q) - \sum_{t \in Q} m_t$.*

Definition 9. *We define the* excess $\mathrm{exc}(Q)$ *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. $\mathrm{exc}(Q) = \sum_{t \in Q} l_t - \mathrm{pv}(Q)$.*

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: $c(S) \ge P - \mathrm{def}(Q(S))$ or $c(S) \ge P - \mathrm{exc}(Q(\bar{S}))$, where $Q(S) \coloneqq \{t \mid v_t \in S\}$.*

Proof. Let $(S, \bar{S})$ be a s-t cut, let $J(S) \coloneqq \{j \mid u_j \in S\}$. 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: $|\{v_t \in \bar{S} \mid t \in E_j\}| = |E_j \setminus Q(S)| \geq p_j - \mathrm{fv}(j, Q(S))$

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

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

$$c(S) \geq \sum_{j \in J(\bar{S})} p_j + \sum_{j \in J(S)} p_j - \mathrm{fv}(j, Q(S)) + \sum_{t \in Q(S)} m_t$$

$$= P - \mathrm{fv}(J(S), Q(S)) + \sum_{t \in Q(S)} m_t$$

$$\geq P - \mathrm{def}(Q(S)).$$

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

- Node s: $\sum_{j \in J(\bar{S})} p_j \geq \mathrm{pv}(J(\bar{S}), Q(\bar{S}))$.

- Node u_j: $|E_j \setminus Q(S)| = |E_j \cap Q(\bar{S})| \geq \mathrm{pv}(j, Q(\bar{S}))$

- Node v_t: l_t

- Node w: $P - \sum_t l_t$

In total, we can lower bound the capacity with

$$c(S) \geq P - \sum_{t \in Q(\bar{S})} l_t + \mathrm{pv}(Q(\bar{S}))$$

$$= P - \mathrm{exc}(Q(\bar{S}))$$

$\square$

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

Proof. If $\mathrm{def}(Q) > 0$ for some Q, then some upper bounds m_t cannot be met. If $\mathrm{exc}(Q) > 0$ 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 $\mathrm{def}(Q(S)) > 0$ or $\mathrm{exc}(Q(\bar{S})) > 0$. $\square$

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

$$\mathrm{fv}(Q) = v(Q),$$
$$\mathrm{v}(t') \geq k - 1 \qquad \qquad \text{for } t' \in Q \text{ and}$$
$$\mathrm{v}(t') \geq k \qquad \qquad \text{for } t' \in Q \text{ with } t' \geq t.$$

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

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 $m_Q > \mathrm{v}(Q) \geq \mathrm{fv}(Q)$ which implies $m_Q \geq \mathrm{fv}(Q)$ 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 $m_Q \geq \mathrm{fv}(Q)$ 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. $\square$

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 ϕ, i.e.*

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

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 $\precsim$ 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 $C_t \precsim C_{t'}$ if and only $k < k'$ or $k = k'$ and $t \geq t'$. In other words, $\precsim$ is the same order in which* pltr *calculates the activations: from Top-Left to Bottom-Right.*

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

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

Definition 17. *Let* crit *$: \{C_t\} \to [m]$ 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 $t \in T$.*

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

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

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

$$\operatorname{rank}(D) \coloneqq \max\{\operatorname{rank}(t) \mid t \in D\}$$
$$\operatorname{crit}(D) \coloneqq \max\{\operatorname{rank}(t) \mid t \in D\}$$

Definition 19. *Let C be a critical set. A nonempty interval $V \subseteq T$ is a* valley *of rank(C) if $C \sim V$ and V is inclusion maximal. Let $C_1, \ldots, C_l$ 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_{pltr} in every $t \in V$.*

Lemma 22. *For every critical set C with $c \coloneqq \operatorname{crit}(C)$, 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 $|J_V| \coloneqq 0$ if V does not exist.*

Proof. By choice of C as critical set with $c = \operatorname{crit}(C)$ we have $\operatorname{v}(C \cap D) \geq (c-1) \cdot |C \cap D|$. If this inequality is fulfilled strictly, i.e. if $\operatorname{v}(C \cap D) > (c-1) \cdot |C \cap D|$, then with $\operatorname{fv}(C \cap D)/|C \cap D| \leq c - \delta$ we directly get $\operatorname{uv}(C \cap D)/|C \cap D| > \delta - 1$. This implies that there are at least δ jobs j scheduled in $C \cap D$ with $\operatorname{uv}(j, C \cap D) > 0$. 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}.

If on the other hand we have equality, i.e. $\mathrm{v}(C \cap D) = (c-1) \cdot |C \cap D|$, then let t be the activation of processor c for which C is critical set for. Since $\mathrm{v}(t) > c-1$, we must have $t \notin C \cap D$. By the same argument as before, we have that if $\mathrm{fv}(C \cap D)/|C \cap D| \leq c-\delta$, then $\mathrm{uv}(C \cap D)/|C \cap D| \geq \delta+1$. Now suppose that there is no job j scheduled in C s.t. $\mathrm{space}(j, C \cap D) > 0$. Then $\mathrm{fv}(C \setminus D) = \mathrm{v}(C \setminus D) > (c-1) \cdot |C \cap D|$. Hence $\mathrm{fv}(C \setminus D) = \mathrm{v}(C \setminus D) > (c-1) \cdot |C \cap D|$. We then get $\phi(C \setminus D) = \mathrm{v}(C \setminus D) > (c-1) \cdot (C \cap D)$ 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 $\mathrm{space}(j, C \cap D) > 0$ and hence

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

which again implies that there must be at least δ 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 δ jobs are contained in J_{V_l} or J_{V_r}. $\qquad\square$

3   Modification of our Schedule

We modify the schedule S_{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 $crit(C)$ processors active. Recall that for the single processor ltr algorithm, the crucial property for the approximation guarantee was that every idle interval of S_{OPT} can intersect at most 2 distinct idle intervals of S_{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_{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_{pltr} into S_{aug} by adding for every t with $k := crit(t) \geq 2$ and $\mathrm{v}(t) = k - 1$ an auxiliary active slot on processor k. This auxiliary active slot does not count towards the volume.

Lemma 23. *In S_{aug}, in every $t \in T$ with $crit(t) \geq 2$ processors $1, \ldots, crit(t)$ 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_{aug}. $\qquad\qquad\square$

3.2   Realignment

Algorithm 2 Realignment of S_{aug}

$\text{Res}(V) \leftarrow 2|J_V|$ for every valley V
for $k \leftarrow m$ to 1 **do**
    fill(T)
    $\text{Res}(V) \leftarrow \text{Res}(V) - 1$ 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 $\text{Res}(V_l) > 0$ **then**
            close(k, V_l)
        else if V_r exists and $\text{Res}(V_r) > 0$ **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 $1, \ldots, k-1$ 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 $1, \ldots, k-1$ to processor k at t

---

3.3   Invariants for Realignment

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

- *processor k_V is not fully filled yet, i.e. 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_{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 $\mathrm{Res}(V_l) + \mathrm{Res}(V_r) \geq 2\delta$.

2. For every $t \in C \cap V$, processors $1, \ldots, k_V$ 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_{real}. Then we show that invariant 2 implies invariant 3. For the induction base, consider S_{aug} and let V be an arbitrary valley in S_{aug} and C the critical set with $C \sim V, \mathrm{crit}(V) \coloneqq c$.

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_{aug} would have $\mathrm{crit}(t) = k_V > c$. This is a direct contradiction to $\mathrm{crit}(V) = \max_{t \in V} \mathrm{crit}(t) = c$.

The second invariant now follows since by construction of S_{aug} and our choice of C we have for every $t \in C$ that processors $1, \ldots, k_V, \ldots, c$ 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 $\mathrm{Res}(V)$ we get the desired lower bound of $\mathrm{Res}(V_l) + \mathrm{Res}(V_r) \geq 2\delta$.

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 $\mathrm{crit}(V) \geq 2$ 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 $1, \ldots, k$. 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 $k_V, \ldots, k$ 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 $t \coloneqq \min V - 1$ 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 $1, \ldots, k_V, \ldots, k_W$ 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 $A' \subseteq A$.
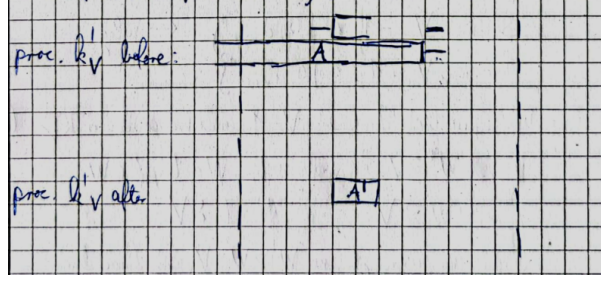
Figure 3.1: The situation for case 1 in Lemma 24.

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 $\mathrm{Res}(V_l) + \mathrm{Res}(V_r) \leq 2(\delta + 1)$ both before and after the close. Our induction hypothesis also implies that for every $t \in C \cap V$ processors $1, \ldots, k_V$ 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 $1, \ldots, k_V - 1 \geq k'_V$ 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 $\mathrm{fill}(k, W)$. 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 $\mathrm{fill}(k, W)$, 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 $k_V = k \geq k'_V$.

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

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 $\mathrm{Res}(V_l) + \mathrm{Res}(V_r) \geq 2\delta$.

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 $\mathrm{fill}(k, T)$ returns and $\mathrm{Res}(V')$ is decreased by 1 for every V' such that some valley intersecting V' has been closed during $\mathrm{fill}(k, T)$. 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 $\mathrm{fill}(k, T)$, no valley V' intersecting V was closed on k, then $\mathrm{Res}(V)$ does not change and Invariant 1 follows from the induction hypothesis and $k'_V \leq k_V$. If on the other hand some valley V' intersecting V was closed on k during $\mathrm{fill}(k, T)$, then $\mathrm{Res}(V)$ is decresed by 1 to $\mathrm{Res}'(V) := \mathrm{Res}(V) - 1$. As seen in cases 1 to 3, k_V decreases monotonically during $\mathrm{fill}(k, T)$. 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 $\mathrm{fill}(k, T)$ 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 $\phi(C \cap D) \leq k^0_V - (\delta + 1)$ and hence the left valley V_l or the right valley V_r of C, D exists and before the close $\mathrm{Res}(V_l) + \mathrm{Res}(V_r) \geq 2(\delta + 1)$. Since Res is decreased by at most 1, we have after the close $\mathrm{Res}'(V_l) + \mathrm{Res}'(V_r) \geq 2\delta$.

We conclude by showing that Invariant 2 implies Invariant 3. Let V be an arbitrary valley during the realignemnt of S_{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. $\qquad\square$

Lemma 25. *The result S_{real} of the realignment of S_{aug} is defined.*

Proof. The Invariants in Lemma 24 imply that whenever we are inside the while-loop of 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 suffcient Res to close some valley.

Since in the while-loop of $\mathrm{fill}(k, V)$ 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 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. $\qquad\square$

Lemma 26. *For every processor $k \in [m]$, every active interval A on processor k in S_{real} with $\mathrm{crit}(A) \geq 2$ we have $\hat{\phi}(A) > k - 1$.*

Proof. We show that $\mathrm{fill}(k, T)$ establishes the property on k. The claim then follows since $\mathrm{fill}(k, T)$ does not change the schedules of processors above k. Since in $\mathrm{fill}(k, T)$, 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_{real} with $A \sim V$ and $\mathrm{crit}(A) \geq 2$. No valley $W \supseteq V$ can have been closed on k since otherwise there would be no $A \subseteq V$ in S_{real}. Consider then the point in $\mathrm{fill}(k, V)$ 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 $\S_{real}$ with $A \sim V$. In particcular, one such A' must have $A' \subseteq A$, which directly implies $\hat{\phi}(A) \geq \hat{\phi}(A') > k - 1$. $\qquad\square$

Lemma 27. *The realignment of S_{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_{real}, then t is also an activation of some processor k' in S_{aug}.*

Proof. Consider the first step in the realignment of S_{aug} in which some $t \in T$ becomes an activation of some processor k' where t was no activation of any processor before this step. This step must be the closing of some valley V on some processor $k > k'$: <u>On processor k, we have seen that closing of some valley can only merge active intervals.</u> On processors above k, the schedule does not change.

> Explicitly make this into a separate lemma?

Active slots on processors $k'' < k$ are only removed (definition close), therefore $t - 1$ must have been active on processor k' and idle on $k' + 1, \ldots, k$ before the close. Refer to Figure 3.3 for a visual sketch of the situation.
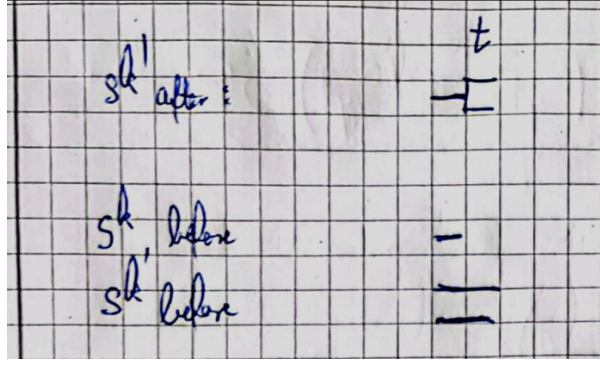
Figure 3.2:

If $t \in V$, then processor $k' + 1$ (or k) must have been active before at t. Hence t was already an activation before the close, contradicting our initial choice of t.

If $t \notin V$, then $t \succ V$. Let W be the valley s.t. V is closed during $\text{fill}(k, W)$, hence $W \supset V$. If $t \in W$, then $t \sim C_W$ and $t \in C_W$. By Invariant 2, processors $1, \ldots, k_W = k$ are active at t before the close. Again, this implies that t was an activation before the close already, contradicting our choice of t. If $t \notin W$, then let W' be the valley with $t \sim W$ and $t \in W'$. We have $W \prec t \sim W'$ and $W' \supset W$ and $t \in C_{W'}$. Therefore $k_{W'} \geq k_W = k$ and Invariant 2 implies that processors $1, \ldots, k$ are active at t before the close. Hence, t was activation before the close already, again contradicting our initial choice of t. $\square$

Lemma 28. *Let I an idle interval in S_{real} on some processor k and A_l, A_r the active intervals to the left and right of I with $\text{crit}(A_l) \leq 1$ and $\text{crit}(A_r) \leq 1$. Allow A_l to be empty, i.e. we might have $\min I = 0$, but A_r must be nonempty, i.e. $\max I < d^*$. Then $k = 1$ and $\hat{\phi}(A_l \cup I \cup A_r) > 0$.*

Proof. By Lemma 27 and $\text{crit}(A_r) \leq 1$, we know that $\min A_r$ is an activation of processor 1 in S_{aug}. Hence $\max I$ is idle in S_{aug} on processor 1 and hence on all processors (stair-property in S_{aug}). Since no jobs are scheduled at $\max I$, we know that $\text{crit}(\max I) \leq 1$ and $J_V = \emptyset$ for all valleys V containing the slot $\max I$, and hence also $\text{Res}(V) = 0$ at all times during the realignment. Therefore, no V intersecting $[\max I, \max A_r]$ was closed during realignment on any processor, since this V would contain $\max I$. Since A_r is an active interval with $\text{crit}(A_r) \leq 1$ (i.e. not containing activations of processors above 1 in S_{aug}), we must then have $k = 1$.

For I to be idle on processor $k = 1$ in S_{real} and $\text{crit}(I) \geq 2$, some $V \succeq I$ with $V \cap I \neq \emptyset$ and hence $V \supseteq \max I$ would have to been closed, which contradicts what we have just shown. Therefore $\text{crit}(I) \leq q$ and no valley V with $V \cap [\min A_l - 1, \max A_r + 1] \neq \emptyset$ can have been closed. Therefore, the constellation occurs exactly in the same way in S_{aug} on processor 1, see Figure 3.3.
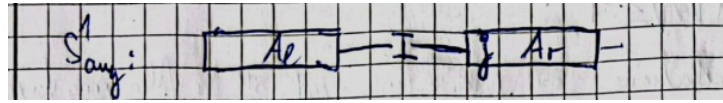

Figure 3.3:

Let j be the single job scheduled at $\min A_r$. We conclude by showing that $E_j \subseteq I \cup A_r$ and therefore $\hat{\phi}(I \cup A_r) > 0$. Otherwise, j could be scheduled at $\min I$ or $\max A_r + 1$. In the first case, pltr would have extended A_l by scheduling j at time $\min I$ instead of at $\min A_r$. In the second case, pltr would have extended the idle interval I by scheduling j at $\max A_r + 1$ intead of at $\min A_r$. $\square$

Lemma 29. *For every processor k, every idle interval on processor k in S_{OPT} intersects at most two distinct idle intervals of processor k in S_{real}.*

Proof. Let I_{OPT} be an idle interval in S_{OPT} on processor k intersecting three distinct idle intervals of processor k in S_{real}. Let I be the middle of these idle intervals. Lemma 28 and Lemma 26 imply that k active processors are required during I and its neighboring active intervals. This makes it impossible for S_{OPT} to be idle on processor k during the whole interval I_{OPT}. $\square$

12

3.4   Approximation Guarantee

Lemma 30. $\mathrm{costs}(S_{real}) \leq 2\,\mathrm{OPT} + P$

Proof. We first show that $\mathrm{idle}(S^k_{\mathrm{real}}) \leq 2\,\mathrm{off}(S^k_{\mathrm{OPT}}) + \mathrm{on}(S^k_{\mathrm{OPT}})$ for every processor $k \in [m]$. Let $mathcalI_1$ be the set of idle intervals on S^k_{real} intersecting some off-interval of S^k_{OPT}. Lemma 29 implies that $\mathcal{I}_1$ contains as most twice as many intervals as there are off-intervals in S^k_{OPT}. Therefore, the costs of these idle intervals is bounded by $2\,\mathrm{off}(S^k_{\mathrm{OPT}})$.

Let $\mathcal{I}_2$ be the set of idle intervals on S^k_{real} not intersecting any off-interval in S^k_{OPT}. The total length of these intervals is bounded by $\mathrm{on}(S^k_{\mathrm{OPT}})$.

We continue by showing that $\mathrm{active}(S_{real}) \leq 2P$. By construction of S_{aug} and the definition of Res and close, we introduce at most as many auxiliary active slots at every slot $t \in T$ as there are jobs scheduled at t in S_{pltr}: For S_{aug}, an auxiliary active slot is only added for t with $\mathrm{crit}(t) \geq 2$ and hence $\mathrm{v}(t) \geq 1$. Furthermore, initially $\mathrm{Res}(V) = 2|J_V|$ for every valley V and $\mathrm{Res}(V)$ is decremented if some V' intersecting V is closed during $\mathrm{fill}(k, T)$. During $\mathrm{fill}(k, T)$ every $t \in T$ is at most closed once (or rather one V' containing t is closed). Finally, auxiliary active slots introduced by S_{aug} are used in close. In conclusion, we have $\mathrm{costs}(S_{\mathrm{real}}) \leq 2\,\mathrm{off}(S_{\mathrm{OPT}}) + \mathrm{on}(S_{\mathrm{OPT}}) + 2P \leq 2\,\mathrm{OPT} + P$. □

Theorem 31. $\mathrm{costs}(S_{\mathrm{pltr}}) \leq 2\,\mathrm{OPT} + P$

Proof. We argue that $\mathrm{costs}(S_{\mathrm{pltr}}) \leq \mathrm{costs}(S_{\mathrm{real}})$. The theorem then follows from Lemma 30. We do this by transforming S_{real} back into S_{pltr} without increasing the costs of the schedule. Removing the auxiliary active slots clearly cannot increase the costs. Since the realignment of S_{aug} only moves active slots between processors, but not between different time slots, we can easily restore S_{pltr} (up to permutations of the jobs scheduled on the active processors at the same time slot) by moving all active slots back down to the lower numbered processors. By the same argument as in Lemma **??**, this does not increase the total costs of the schedule. □

> Define Lemma stair property somewhere

4   Running Time

Theorem 32. *Algorithm* pltr *has a running time of $O(Fn \log d)$, where F is the time needed for the flow calculation in Figure 1.*

Proof. First observe that every active interval is created by a pair of calls to keepIdle and keepActive. Every call to keepIdle searches for the maximal slot $t \in T$ (after the current time slot) such that the maximum-flow calculation in Figure 1 returns a flow of value P. Using binary search this can be done in $O(F \cdot \log(d^*))$ with F denoting the time required for finding the maximum flow.

We bound the number of active intervals across all processors in S_{pltr} by n. Note that if keepIdle returns d^*, then we do not have to calculate keepActive from d^* on. Therefore, the total number of calls to keepIdle and keepActive is bounded by $n+m$. If $m > n$ we can restrict our algorithm to use the first n processors only, as there cannot be more than n processors scheduling jobs at the same time. We derive the upper bound of n for the number of active intervals across all processors by constructing an injective mapping f from the set of active intervals to the jobs J. For this construction of f we consider the active intervals in the same order as the algorithm, i.e. from Top-Left to Bottom-Right. We construct f such that $f(A) = j$ for some active interval A only if $d_j \in A$.

Suppose we have constructed such a partial mapping for active intervals on processors $m, \ldots, k$ up to some active interval A on k. Let A' be the last plateau with $A' \subseteq A$ and let $k' \geq k$ be the processor of this plateau A'. By construction of f and the choice of A', there are at most $k' - (k+1)$ distinct jobs j with $d_j \in [\min A', \max A]$ already mapped to by f. This is since at most $k' - (k+1)$ active intervals on processors $k+1, \ldots, m$ intersect the interval $[\min A', \max A]$. Refer to Figure 4 for a visaul sketch.
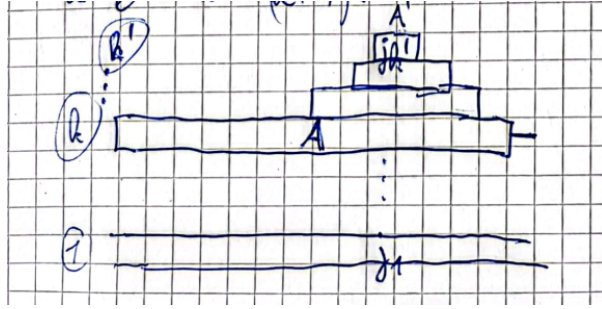
Figure 4.1:

Let C_t be the critical set over activation $t := \min A'$ of processor k'. Let $J' := \{j_1, \ldots, j_{k'}\}$ be the k' distinct jobs scheduled at t. We now that $\max A + 1 \notin C_t$ since $\mathrm{v}(\max A + 1) < k \leq k'$ and $\max A + 1 > t$. Therefore, every job $j \in J'$ with $d_j \geq \max A + 1$ is scheduled at slot $\max A + 1$. Hence there are at least $k' - (k-1)$ distinct jobs $j \in J'$ with $d_j \in [\min A', \max A]$ and there must be at least one such job j^* which is not mapped to by f so far and which we therefore can use for $f(A)$. □

References