Appendix (e-companion)

EC.1. Appendix to Section 1

EC.1.1. Example: Parallel Computing on Heterogeneous Architectures

A typical application area for malleable scheduling is parallel computing. With the advent of specialized processing units (optimized, e.g., for graphical or machine learning computations), the corresponding scheduling systems have to be able to handle the ensuing heterogeneity. For example, Bleuse et al. (2017) devise a scheduler for a platform in which computational tasks can be either assigned to a single GPU or processed in parallel across multiple multicore CPUs. In the following, we explain how their model, and similar heterogeneous parallel computing environments, can be readily modeled within the generalized malleable scheduling framework. We further show that under natural and widespread assumptions on the parallelization behavior of the underlying computational tasks, known as Brent's (1974) lemma—an assumption also made by Bleuse et al. (2017)—and Amdahl's (1967) law, respectively, the corresponding processing speed functions of jobs are subadditive or fractionally subadditive, respectively.

Consider an environment consisting of a set of malleable computational tasks J to be distributed on a set of different processing units M, with each unit having a specific type $k \in K$, where K is the set of all types and M_k represents the set of units of type k. Units of the same type have the same architecture (e.g., CPU or GPU) but may have differing clock speeds, denoted by s_i for $i \in M$. Each task can be distributed over multiple processing units, but only if they are all of the same type. Thus, the processing time of task $j \in J$ on processing units $S \subseteq M$ is given by $f_j(S) = \min_{k \in K} f_{jk}(S \cap M_k)$, where $f_{jk}(T)$ for $k \in K$ denotes the processing time of j when scheduled on processing units $T \subseteq M_k$ of type k. Note that this also implies that the corresponding processing speed of S for j is given by $g_j(S) = \frac{1}{f_j(S)} = \max_{k \in K} \frac{1}{f_{jk}(S \cap M_k)}$.

We will examine two regimes for the processing time functions f_{jk}, corresponding to widespread assumptions in parallel computing: (i) the monotone work assumption, justified by Brent's (1974) lemma and (ii) Amdahl's (1967) law. As we will see, in the former case the processing speeds are subadditive and in the latter case they are fractionally subadditive.

EC.1.1.1. Processing speeds under Brent's lemma. In the following, we will assume that the joint speed of processing units $T \subseteq M_k$ for completing task j is determined by the total clock speed $\sum_{i \in T} s_i$, i.e.,

$$f_{jk}(T) = \bar{f}_{jk}(\textstyle\sum_{i \in T} s_i) \tag{EC.1}$$

for some function $f_{jk} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. Following Bleuse et al.'s (2017) interpretation of Brent's (1974) lemma, the speed-up achieved by parallelization is at most linear, justifying the monotone work assumption, i.e.,

$$s \cdot \bar{f}_{jk}(s) \leq t \cdot \bar{f}_{jk}(t) \quad \text{for all } s, t \in \mathbb{R}_{\geq 0} \text{ with } s \leq t. \tag{EC.2}$$

The following lemma shows that in under these conditions, the processing speed function g_j of j is subadditve.

Lemma EC.1. *If for each $k \in K$, the processing time function f_{jk} is of the form* (EC.1) *and fulfils* (EC.2)*, then g_j is subadditive.*

Proof. We first observe that for $s, t \in \mathbb{R}_{\geq 0}$, the monotone work assumption implies

$$\frac{s}{\bar{f}_{jk}(s+t)} \leq \frac{s+t}{\bar{f}_{jk}(s)} \quad \text{and} \quad \frac{t}{\bar{f}_{jk}(s+t)} \leq \frac{s+t}{\bar{f}_{jk}(t)}$$

for all $k \in K$. Adding the two inequalities and dividing by $s+t$ yields

$$\frac{1}{\bar{f}_{jk}(s+t)} \leq \frac{1}{\bar{f}_{jk}(s)} + \frac{1}{\bar{f}_{jk}(t)}$$

for all $k \in K$. Note that this implies

$$\frac{1}{f_{jk}((S \cup T) \cap M_k)} \leq \frac{1}{f_{jk}(S \cap M_k)} + \frac{1}{f_{jk}(T \cap M_k)}$$

for all $S, T \subseteq M$ and all $k \in K$. As a consequence, g_j is the maximum of subadditive functions and hence subadditive itself. □

EC.1.1.2. Processing speeds under Amdahl's law. We now assume that the processing times are determined by Amdahl's (1967) law, another widespread assumption in parallel computing, which assumes linear speed-up by parallelization but considers the fact that some parts of a computational task might not be parallelizable. For each task $j \in J$, let $W_j \in \mathbb{R}_{\geq 0}$ indicate the amount of work, i.e., number of operations, necessary to complete j. Let $\rho_j \in [0,1]$ indicate the fraction of j that can be parallelized and distributed over multiple processors. The remaining $1-\rho_j$ fraction of j is non-parallelizable, it consists of the work that is necessary for initialization and has to be carried out on each participating processor individually. Amdahl's law determines the processing time for task j on processor set $S \subseteq M_k$ of type $k \in K$ as follows:

$$f_{jk}(S) = \frac{(1-\rho_j)|S| + \rho_j}{\sum_{i \in S} s_i} \cdot W_j. \tag{EC.3}$$

The following lemma shows that if processing times for each $k \in K$ follow Amdahl's law as described above, the corresponding processing speed function is fractionally subadditive.

LEMMA EC.2. *If for each $k \in K$, the processing time function f_{jk} is of the form* (EC.3)*, then g_j is fractionally subadditive.*

Proof. Let $T \subseteq M$ and let $\alpha : 2^T \to [0,1]$ with $\sum_{S \subseteq T : i \in S} \alpha(S) \geq 1$ for all $i \in T$. Let $k^* \in K$ such that $g_j(T) = \frac{1}{f_{jk^*}(T \cap M_{k^*})}$. Then

$$\begin{aligned}
\sum_{S \subseteq T} \alpha(S) g_j(S) &= \sum_{S \subseteq T} \alpha(S) \max_{k \in K} \frac{1}{f_{jk}(S \cap M_k)} \geq \sum_{S \subseteq T} \alpha(S) \frac{1}{f_{jk^*}(S \cap M_{k^*})} \\
&= \sum_{S \subseteq T} \alpha(S) \frac{\sum_{i \in S \cap M_{k^*}} s_i}{((1-\rho_j)|S \cap M_{k^*}| + \rho_j) W_j} \geq \sum_{S \subseteq T} \alpha(S) \frac{\sum_{i \in S \cap M_{k^*}} s_i}{((1-\rho_j)|T \cap M_{k^*}| + \rho_j) W_j} \\
&\geq \frac{\sum_{i \in T \cap M_{k^*}} s_i}{((1-\rho_j)|T \cap M_{k^*}| + \rho_j) W_j} = g_j(T),
\end{aligned}$$

where the second inequality follows from $S \subseteq T$ and the third inequality follow from the fact that $\sum_{S \subseteq T : i \in S} \alpha(S) \geq 1$ for all $i \in T$. We conclude that g_j is fractionally subadditive. □

Efficient demand oracle for processing speeds under Amdahl's law. We further remark that a demand oracle (i.e., an algorithm determining a maximizer of $g_j(S) - \sum_{i\in S} p_i$ for a given vector $p \in \mathbb{R}^M_{\geq 0}$) for processing speed functions g_j of the type described above can be implemented as follows: Let $j \in J$ and $p \in \mathbb{R}^M_{\geq 0}$. For each $k \in K$, each $\ell \in \{0, \dots, |M_k|\}$, and each $S \subseteq M_k$ define

$$h_{k\ell}(S) := \frac{\sum_{i\in S} s_i}{((1-\rho_j)\ell + \rho_j)W_j} - \sum_{i\in S} p_i$$

and let $S_{k\ell} \in \arg\max_{S\subseteq M_k, |S|=\ell} h_{k\ell}(S)$. Note that $h_{k\ell}$ is an additive function and thus such a maximizer of cardinality ℓ can be computed greedily in polynomial time. The demand oracle determines $k' \in K$ and $\ell' \in \{0, \dots, |M_{k'}|\}$ so as to maximize $h_{k'\ell'}(S_{k'\ell'})$ and returns $S_{k'\ell'}$. As there are only $\sum_{k\in K} |M_k|$ possible choices for k' and ℓ', the demand oracle can be implemented to run in polynomial time.

To see that $S_{k'\ell'}$ is indeed a maximizer to the demand query, consider any maximizer $S^* \in \arg\max_{S\subseteq M} g_j(S) - \sum_{i\in S} p_i$. Let $k^* \in K$ such that $g_j(S^*) = \frac{1}{f_{jk}(S^*\cap M_{k^*})}$ and let $\ell^* := |S^*|$. Note that

$$\begin{aligned} g_j(S^*) - \sum_{i\in S^*} p_i \;&=\; h_{k^*\ell^*}(S^*\cap M_{k^*}) - \sum_{i\in S^*\setminus M_{k^*}} p_i \;\leq\; h_{k^*\ell^*}(S_{k^*\ell^*}) \\ &\leq\; g_j(S_{k^*\ell^*}) - \sum_{i\in S_{k^*\ell^*}} p_i \;\leq\; g_j(S_{k'\ell'}) - \sum_{i\in S_{k'\ell'}} p_i, \end{aligned}$$

where the first inequality follows from choice of $S_{k^*\ell^*}$ and the fact that $p_i \geq 0$ for all $i \in M$, the second inequality follows from the facts that $h_{k\ell}(S) = \frac{1}{f_{jk}(S)} - \sum_{i\in S} p_i$ for $S \subseteq M_k$ with $|S| = \ell$ and that $g_j(S) = \max_{k\in K} \frac{1}{f_{jk}(S\cap M_k)}$, and the final inrequality follows from choice of k' and ℓ'.

EC.1.2. Lower Bounds on the Assignment-Schedule Gap

EC.1.2.1. An Example with Large Assignment-Schedule Gap for Arbitrary Processing Speed Functions Consider n jobs and $\frac{n\cdot(n-1)}{2}$ machines, where each machine contributes to the execution of exactly two distinct jobs, and each job requires a specific subset of $n-1$ machines in order to be executed with processing time 1 (otherwise the job cannot be executed, i.e., $f_j(S) = \infty$ for all jobs $j \in J$ and all machine sets $S \subseteq M$ such that do not contain all $n-1$

machines required for job j). In order to visualize the allocation of machines to jobs, one can think of a clique graph K_n with n nodes, where each node represents a distinct job and every edge corresponds to a machine that contributes to the execution of both its adjacent job nodes.

Note that there is only one assignment with finite load for the above instance. In this assignment, the load of each machine is exactly 2, as each machine processes exactly two jobs of processing time 1. On the other hand, it is not hard to verify that the makespan of any feasible schedule for the same assignment cannot be less than n. Indeed, by construction of our instance, every two jobs share exactly one machine (i.e., the one corresponding to their common edge in the clique) and, thus, their executions cannot be overlapping. Hence, since jobs have to be processed sequentially, each with processing time 1, we conclude that any feasible schedule has makespan at least n.

EC.1.2.2. A Lower Bound on the Assignment-Schedule Gap for Additive Processing Speed Functions In the following, we describe an example instance of generalized malleable scheduling with additive processing speeds in which the gap between the makespan of an optimal schedule and the load of an optimal assignment is $\frac{\phi+1}{2} \approx 1.309$, where ϕ is the golden ratio. The instance has three machines $M = \{a, b, c\}$ and three jobs $J = \{1, 2, 3\}$. For $S \subseteq M$ and $j \in J$, we define the processing speed $g_j(S) := \sum_{i \in S} s_{ij}$, where the values s_{ij} for $i \in M$ and $j \in J$ are given in Table EC.1, with $\delta := \phi - 1 = \frac{\sqrt{5}-1}{2}$.

	a	b	c
1	δ	0	$1-\delta$
2	0	δ	$1-\delta$
3	0.5	0.5	0

Table EC.1 The values s_{ij} for the additive processing speed functions in the example described in Appendix EC.1.2.2. Rows correspond to jobs, columns correspond to machines. The value of δ is $\phi - 1 = \frac{\sqrt{5}-1}{2}$.

Consider the assignment $\mathbf{S}$ with $S_1 = \{a, c\}$, $S_2 = \{b, c\}$, and $S_3 = \{a, b\}$. Note that $f_j(S_j) = 1/g_j(S_j) = 1$ for all $j \in J$ and that, furthermore, $|\{j \in J : i \in S_j\}| = 2$ for all $i \in M$. Hence $L(\mathbf{S}) = 2$.

Consider any schedule $(\mathbf{T}, \mathbf{t})$ and let $C_{\max} = \max_{j \in J} t_j + f_j(T_j)$ be its makespan. We show that $C_{\max} \geq 1 + \frac{1}{\delta} = \phi + 1 \approx 2.618$, establishing the claimed lower bonud on the assignment-schedule gap. By contradiction assume that $C_{\max} < 1 + \frac{1}{\delta}$. Note that $1 + \frac{1}{\delta} = \frac{1}{1-\delta}$ by construction, and hence $g_1(T_1) > 1 - \delta$ and $g_2(T_2) > 1 - \delta$. This implies $a \in T_1$ and $b \in T_2$. Furthermore, $T_3 \cap \{a, b\} \neq \emptyset$ because $g_3(T_3) > 0$.

We distinguish three cases:

- If $b \notin T_3$, then $a \in T_3$ and $f_3(T_3) = 2$. Because $f_1(T_1) \geq 1$, this imples

$$C_{\max} \geq \sum_{j \in J : a \in T_j} f_j(T_j) = f_1(T_1) + f_3(T_3) \geq 3.$$

- If $a \notin T_3$, then $b \in T_3$ and $f_3(T_3) = 2$. Because $f_2(T_2) \geq 1$, this implies

$$C_{\max} \geq \sum_{j \in J : b \in T_j} f_j(T_j) = f_2(T_2) + f_3(T_3) \geq 3.$$

- If $\{a, b\} \subseteq T_3$ then $1 + \frac{1}{\delta} > C_{\max} \geq f_1(T_1) + f_3(T_3) \geq f_1(T_1) + 1$ implies $g_1(T_1) > \delta$ and hence $\{a, c\} \subseteq T_1$. By a symmetric argument, $\{b, c\} \subseteq T_2$. Hence $f_j(T_j) = 1$ for all $j \in J$ and $T_j \cap T_{j'} \neq \emptyset$ for all $j, j' \in J$. This implies that for each $j, j' \in J$ with $j \neq j'$ either $t_j \geq t_{j'} + 1$ or $t_{j'} \geq t_j + 1$. Hence there must by a job j with $t_j \geq 2$ and thus $C_{\max} \geq 3$.

We conclude that $C_{\max} \geq 3 > 1 + \frac{1}{\delta}$ in all three cases, a contradiction.

EC.1.3. IP Formulation for Assignment

The following IP is an exact formulation for the ASSIGNMENT problem, with binary variables $x(S, j)$ for every $j \in J$ and every $S \subseteq M$, indicating that job j should be processed on machine set S.

$$\begin{aligned}
\textbf{minimize:} \quad & C & & & \text{(EC.4)} \\
\textbf{s.t.:} \quad & \sum_{S \subseteq M : S \neq \emptyset} x(S, j) \;=\; 1 & & \forall\, j \in J \\
& \sum_{j \in J} \sum_{S \subseteq M : i \in S} \frac{1}{g_j(S)} x(S, j) \;\leq\; C & & \forall\, i \in M \\
& x(S, j) \;\in \{0, 1\} & & \forall\, S \subseteq M,\, j \in J
\end{aligned}$$

EC.1.4. Lower Bounds on the Approximability

In this section, we discuss some lower bounds on the approximability of the generalized malleable scheduling problems Assignment and Scheduling. When processing speeds are submodular, both problems generalize the classic (non-malleable) scheduling problem of makespan minimization on a set of unrelated machines, denoted by $R||C_{\max}$ in the literature. In this problem, we are given a set of jobs J, a set of machines M, and processing times p_{ij} for job $j \in J$ on machine $i \in M$. The goal is to find an assignment $\sigma : J \to M$ that assigns each job to a machine so that the makespan $\max_{i \in M} \sum_{j \in J : \sigma(j) = i} p_{ij}$ is minimized.

This problem can be recast as a generalized malleable scheduling problem by defining $f_j(S) := \min_{i \in S} p_{ij}$ for $j \in J$ and $S \subseteq M$ (yielding submodular processing speeds $g_j(S) = \max_{i \in S} \frac{1}{p_{ij}}$). Indeed, note that because the processing time $f_j(S)$ is determined entirely by the fasted machine for job j in the set S, both for Assignment and for Scheduling there is an optimal solution where each job is only assigned to a single machine and thus for the constructed instance both problems are equivalent to the original $R||C_{\max}$ instance.

The problem $R||C_{\max}$ has been studied extensively in scheduling literature, with the best known approximation result for the problem in its general form still being the 2-approximation by Lenstra et al. (1990). Lenstra et al. (1990) also showed that, unless $P = NP$, there is no approximation algorithm with a guarantee better than 1.5 for $R||C_{\max}$, even when only considering the so-called *restricted assignment* setting, in which for each $j \in J$ there is a value p_j and a set M_j such that $p_{ij} = p_j$ if $i \in M_j$ and $p_j = \infty$ otherwise, i.e., the processing time of a job is independent of the machine, but the job can only be assigned to a machines from a certain subset.

Note that in the restricted assignment case, the processing speeds for the corresponding generalized malleable scheduling instance are of the form $g_j(S) = \frac{\min\{|S \cap M_j|, 1\}}{p_j}$ and hence they are scaled matroid rank functions (for the uniform rank-1 matroid on M_j). Therefore, the inapproximability bound of 1.5 applies to Assignment and Scheduling, even when all processing speeds are scaled matroid ranks (and thus also for all of the more general classes of processing speeds introduced in Section 1.2).

For the case of fractionally subadditive processing speed functions, we provide a slightly stronger inapproximability result below. We emphasize that the processing speed functions constructed in the corresponding reduction allow for a straightforward efficient demand oracle and thus the hardness even holds when assuming such an oracle (recall that Theorem 5 in Section 4 establishes a much stronger query-complexity based inapproximability when fractionally subadditive processing speeds are only given by a value oracle). The lower bound of 2 is particularly interesting, as it shows that generalized malleable scheduling with fractionally subadditive processing speeds is indeed strictly harder than the restricted-assignment version of $R||C_{\max}$, for which it is possible to estimate the optimal makespan within a factor of $11/6+\varepsilon$ in polynomial time for any $\varepsilon > 0$ (Jansen and Rohwedder 2017).

THEOREM EC.1. *The following decision problem is NP-hard: Given an instance of* ASSIGNMENT *with fractionally subadditive processing speeds given by a demand oracle and such that all processing times are in* $\{1, 2, \infty\}$*, decide whether there is an assignment* $\mathbf{S}$ *such that* $f_j(S_j) = 1$ *for all* $j \in J$ *and* $S_j \cap S_{j'} = \emptyset$ *for* $j \neq j'$*. In particular, unless* $P = NP$*, there is no* $(2-\varepsilon)$*-approximation algorithm for any* $\varepsilon > 0$ *for* ASSIGNMENT *or* SCHEDULING*, even when restricted to fractionally subadditive processing speeds given by a demand oracle.*

Proof. The inapproximability follows via a reduction from 3-DIMENSIONAL MATCHING. In this problem, we are given three disjoint sets V_1, V_2, V_3 of equal size, and a set of triplets $\mathcal{E} \subseteq 2^{V_1 \cup V_2 \cup V_3}$ with $|E \cap V_i| = 1$ for all $i \in \{1, 2, 3\}$. The task is to decide whether there exists a *3-dimensional matching*, i.e., a set $\mathcal{F} \subseteq \mathcal{E}$ such that for each $v \in V_1 \cup V_2 \cup V_3$ there is exactly one $F \in \mathcal{F}$ with $v \in F$. 3-DIMENSIONAL MATCHING is NP-complete (Garey and Johnson 1979).

Given an instance of 3-DIMENSIONAL MATCHING, we construct an instance of ASSIGNMENT which has an optimal value of 1 if there exists a 3-dimensional matching, and at least 2 otherwise. To this end, we let $J := V_1$ and $M := V_2 \cup V_3$. For $j \in J$ let $\mathcal{E}_j := \{E \in \mathcal{E} : j \in E\}$. For each $E \in \mathcal{E}_j$, define $\alpha_{jE} \in \mathbb{R}^M$ by

$$\alpha_{jE}(i) = \begin{cases} \frac{1}{2} & \text{if } i \in E, \\ 0 & \text{otherwise.} \end{cases}$$

For $j \in J$ and $S \subseteq M$ let $g_j(S) := \max_{E \in \mathcal{E}_j} \sum_{i \in S \cap E} \alpha_{jE}(i)$. Note that g_j is the maximum of non-negative additive functions and hence fractionally subadditive (Feige 2009). Note further that $f_j(S) = \frac{1}{g_j(S)} \in \{1, 2, \infty\}$ for all $j \in J$ and $S \subseteq M$, and that, in particular, $f_j(S) = 1$ if and only if there is an $E \in \mathcal{E}_j$ with $E \subseteq \{j\} \cup S$. Thus, there is a 3-dimensional matching if and of only if there is an assignment $\mathcal{S}$ with (i) $f_j(S_j) = 1$ for each $j \in J$ and (ii) $S_j \cap S_{j'} = \emptyset$ for $j \neq j'$. Hence, deciding the existence of such an assignment is *NP*-hard.

Note further that such an assignment has maximum machine load 1 and that $(\mathbf{S}, \mathbf{t})$ for the starting time vector $t_j = 0$ for all $j \in J$ is a feasible schedule with makespan 1. Moreover, any assignment not fulfilling both (i) and (ii) has a machine with load at least 2. Thus, a $(2 - \varepsilon)$-approximation algorithm for Assignment or Scheduling would find an assignment with makespan 1 for the constructed instance if such an assignment exists, and could thus decide whether a 3-dimensional matching exists in the original instance.

Finally, note that an efficient demand oracle for the functions g_j can be easily obtained by picking a maximizer of $g_j(S) - \sum_{i \in S} p_i$ among all sets $S \subseteq M$ with $|S| \leq 2$, because $|E \cap M| \leq 2$ for all $E \in \mathcal{E}_j$. □

EC.1.5. Approximation Algorithms for Scheduling

Relaxed decision procedures. Many approximation algorithms for makespan minimization rely on the construction of a *ρ-relaxed decision procedure* (Lenstra et al. 1990), namely, a polynomial-time routine which, given a value C, either correctly asserts that there exists no assignment (resp., schedule) with makespan less than C, or returns a feasible assignment (resp., schedule) with makespan at most ρC. It is well-known that access to such a routine immediately implies a polynomial-time ρ-approximation algorithm for the underlying problem of makespan minimization. Indeed, this can be achieved by binary search over the possible range of C, given that lower and upper bounds on the optimal makespan can be computed. Since all the algorithms presented in this work follow the above regime, we focus on constructing such procedures.

Rounding theorem for scheduling unrelated machines A recurrent theme in our analysis it the use of the following *rounding theorem* due to Lenstra et al. (1990). Let M be a set of jobs and J be a set of machines, where each job must be assigned to a single machine (that is, jobs are non-malleable). Let $p_{ij} \in \mathbb{R}_{\geq 0}$ be the *unrelated* processing time for assigning job j to machine i.

For any C, we consider the following feasibility LP:

$$\begin{aligned} \sum_{i \in M} x_{ij} &= 1 && \forall\, j \in J \\ \sum_{j \in J} p_{ij} x_{ij} &\leq C && \forall\, i \in M \\ x_{ij} &= 0 \quad \forall\, i \in M,\ j \in J \text{ with } p_{ij} > C \\ x &\geq 0 \end{aligned}$$

THEOREM EC.2 **(Rounding theorem (Lenstra et al. 1990))**. *Any feasible (fractional) extreme point solution to the above LP can be rounded into an integral solution of makespan at most* $2C$.

We now briefly discuss the main idea behind the above result. Given any feasible extreme point solution x to the above LP, we can construct the *support graph* G as follows: We consider a node u_j for each job j and a node v_i for each machine i. For each $x_{ij} > 0$, we define an edge $\{u_j, v_i\}$. It can be proved that G is a *pseudoforest*, in the sense that all its connected components are either trees, or contain at most one cycle. Hence, the edges of G (and of any pseudoforest) can be oriented in a way that each node has an in-degree of at most one. Given such an orientation, the rounding proceeds in two steps: First, all integral assignments are maintained and the involved edges and job-nodes are removed from G. Then, every fractionally assigned job is assigned to any of each children machines. Note that each fractionally assigned job always has at least one child machine by construction of the orientation and by the constraints of the LP. By feasibility of the LP, the load of each machine due to the integrally assigned jobs is at most C. Further, due to the orientation, each machine is assigned at most one job from the assignment graph and this job has processing time $p_{ij} \leq C$. Thus, the makespan of the produced schedule is at most $2C$.

EC.2. Appendix to Section 2

EC.2.1. Transformation for Subadditive Processing Speeds

As we already remark in Section 2, the following approximation result is well-known for monotone subadditive functions:

LEMMA EC.3 **(Bhawalkar and Roughgarden, 2011)**. *Let $g : 2^M \to \mathbb{R}_{\geq 0}$ be a monotone subadditive function. There exists a fractionally subadditive function $h : 2^M \to \mathbb{R}_{\geq 0}$, such that for any $S \subseteq M$ it holds*

$$g(S) \geq h(S) \geq \frac{1}{\ln(|M|)} g(S).$$

Note that Lemma EC.3 requires the function g to be monotone. Since we do not assume monotonicity of our processing speeds, we first observe that Lemma EC.3 can also be applied to non-monotone subadditive functions via the following intermediate transformation.

LEMMA EC.4. *Let $g : 2^M \to \mathbb{R}_{\geq 0}$ be a subadditive function. Then the function g' defined by $g'(S) := \max_{U \subseteq S} g(U)$ is monotone and subadditive.*

Proof. Note that g' is monotone by construction. To see that g' is subadditive, let $S, T \subseteq M$ and let $U \subseteq S \cup T$ be so that $g'(S \cup T) = g(U)$. Note that

$$g'(S \cup T) = g(U) \leq g(U \cap S) + g(U \cap T) \leq g'(S) + g'(T),$$

where the first inequality follows from subadditivity of g and the fact that $U \subseteq S \cup T$, and the second inequality follows from definition of g' and the fact that $U \cap S \subseteq S$ and $U \cap T \subseteq T$. □

Together, Lemmas EC.3 and EC.4 imply the following approximation guarantee for (not necessarily monotone) subadditive functions.

COROLLARY EC.1. *Let $g : 2^M \to \mathbb{R}_{\geq 0}$ be a subadditive function. There exists a fractionally subadditive function $h : 2^M \to \mathbb{R}_{\geq 0}$, such that for any $S \subseteq M$ it holds*

$$\max_{U \subseteq S} g(U) \geq h(S) \geq \frac{1}{\ln(|M|)} g(S).$$

Proof. Let $g'(S) := \max_{U \subseteq S} g(U)$. By Lemma EC.4 g' is monotone and subadditive. So we can apply Lemma EC.3 to g' and obtain h such that $\max_{U \subseteq S} g(U) = g'(S) \geq h(S) \geq \frac{1}{\ln(|M|)} g'(S) \geq \frac{1}{\ln(|M|)} g(S)$. □

We are now ready to prove the bound on the schedule/assignment gap for subadditive processing speeds.

COROLLARY 1. *Assume that all processing speed functions are subadditive. For any assignment* $\mathbf{S}$*, there exists a well-structured assignment* $\mathbf{S}'$ *with* $L(\mathbf{S}') \leq \mu^\star \ln(|M|) L(\mathbf{S})$*, where* $\mu^\star$ *is as defined in Theorem 1.*

Proof. Consider any assignment $\mathbf{S}$ with $L(\mathbf{S}) = C$. Consider a new instance of the SCHEDULING problem, in which the processing speed function g_j of each job j is replaced by its fractionally subadditive approximation h_j as given by Corollary EC.1. Note that for the assignment $\mathbf{S}$ the maximum machine load of any machine with respect to processing speeds h_j is at most $\ln(|M|) \cdot C$. For this new instance, moreover, Theorem 1 guarantees the existence of a well-structured assignment $\mathbf{T}$ of maximum machine load at most $\mu^* \cdot \ln(|M|) \cdot C$ with respect to h_j.

Finally, we construct a third assignment $\mathbf{U}$ by assigning each job $j \in J$ to a set $U_j \subseteq T_j$ with $g_j(U_j) = \max_{U \subseteq T_j} g_j(U)$. Note that $\mathbf{U}$ is well-structured because $\mathbf{T}$ is well-structured and $U_j \subseteq T_j$ for all $j \in J$. Moreover, because $g_j(U_j) = \max_{U \subseteq T_j} g(U) \geq h_j(T_j)$ for all $j \in J$, we conclude that $L(\mathbf{U}) \leq \mu^* \cdot \ln(|M|) \cdot C$ when measuring machine loads for $\mathbf{U}$ with respect to the original processing speeds g_j again. Thus $\mathbf{S}' = \mathbf{U}$ fulfils the statement of the corollary. □

EC.2.2. Existence of Δ (Proof of Lemma 1)

LEMMA 1. *There exists* $\Delta \in \mathbb{R}^{M \times J}$ *such that*

1. $\sum_{i \in S_j} \Delta_{ij} = g_j(S_j)$ *for all* $j \in J$,
2. $\sum_{i \in T} \Delta_{ij} \leq g_j(T)$ *for all* $j \in J$ *and* $T \subseteq S_j$.

As noted in Section 2, the lemma follows from the Bondareva-Shapley theorem (Bondareva 1963). For completeness, we give a direct proof using LP duality below.

Proof. Let $j \in J$. Consider the following pair of primal and dual LP.

$$\begin{aligned}
&\textbf{maximize:} && \sum_{i \in S_j} \Delta_{ij} && \textbf{minimize:} && \sum_{T \subseteq S_j} \alpha(T) \cdot g_j(T) \\
&\textbf{s.t.:} && \sum_{i \in T} \Delta_{ij} \;\le\; g_j(T) \quad \forall T \subseteq S_j && \textbf{s.t.:} && \sum_{T \subseteq S_j : i \in T} \alpha(T) \;\ge\; 1 \qquad \forall i \in S_j \\
&&& \Delta \;\ge\; 0 && && \alpha \;\ge\; 0
\end{aligned}$$

Note that both LPs have feasible solutions and let Δ and α be optimal solutions to the above LPs. By optimality and feasibility of α, it is not hard to verify that $0 \le \alpha(T) \le 1$ for each $T \subseteq S_j$. By strong duality and the fact that g_j is fractionally subadditive for each $j \in J$, we get

$$\sum_{i \in S_j} \Delta_{ij} = \sum_{T \subseteq S_j} \alpha(T) g_j(T) \ge g_j(S_j).$$

Therefore, the values Δ_{ij} obtained from optimal solutions to the primal LP above, for each $j \in J$, fulfill the conditions of the lemma. □

EC.2.3. Bounding the Maximum Load in the Reduced Assignment (Lemma 6)

For convenience, we recall the definitions of $J_i^1 := \{j \in J : i_j = i, x'_{ij} > \lambda\}$ as the set of jobs assigned to any machine $i \in M$ via a Type 1 assignment and $\ell_i := \sum_{j \in J_i^1} \frac{1}{\Delta_{ij}} x'_{ij}$ as the (fractional) load of i due to such Type 1 assignments in the extreme point solution x' to (LP1). We further recall that we defined $T_j(\theta) := \{i \in \bar{S}_j : \ell_i \le \theta\}$ for $j \in J$ and that θ_j^* was chosen as a minimizer of $\max_{i \in T_j(\theta)} \frac{1}{\lambda}\ell_i + f_j(T_j(\theta))$.

Lemma 6. *Let* $\mathbf{T}$ *be the assignment defined by* $T_j = T_j(\theta^*)$ *if* $S'_j = \bar{S}_j$ *and* $T_j = \{i_j\}$ *otherwise. Then* $\mathbf{T}$ *is well-structured and* $L(\mathbf{T}) \le \mu L(\mathbf{S})$*, where* $\mu := \frac{e^{(1-\lambda)/\lambda}}{\lambda(e^{(1-\lambda)/\lambda}-1)}$.

Proof. The fact that $\mathbf{T}$ is well-structured follows immediately from the fact that $\mathbf{S}'$ is well-structured and that $T_j \subseteq S'_j$ for all $j \in J$. In particular, for every machine $i \in M$, the set of jobs processed by i consists of the jobs $j \in J_i^1$ assigned to i via a Type 1 assignment and potentially a single job $j' \in J$ with $i \in T_{j'}(\theta^*_{j'})$ with $x'_{i_{j'}j'} \le \lambda$ assigned to i via a Type 2 assignment. Recall that in the proof of Lemma 4, we showed that

$$\sum_{j \in J_i^1} f_j(\{i\}) \le \frac{1}{\lambda}\ell_i \le \frac{1}{\lambda} C. \tag{EC.5}$$

Thus, if a machine $i \in M$ only processes jobs assigned to it via Type 1 assignments, then its load in $\mathbf{T}$ is at most $\frac{1}{\lambda}C \leq \mu C$ by definition of μ. It thus remains to bound the load of machines $i \in M$ for which there exist some Type 2 job $j \in J$ with $i \in T_j(\theta_j^*)$ and $x'_{ij} \leq \lambda$.

Now fix any such Type 2 job $j \in J$ and note that the load of any machine $i \in T_j = T_j(\theta_j^*)$ is $f_j(T_j(\theta_j^*)) + \sum_{j' \in J_i^1} f_{j'}(\{i\}) \leq f_j(T_j(\theta_j^*)) + \frac{1}{\lambda}\ell_i$ by (EC.5). For any $\theta > 0$, the maximum load of a machine in T_j is therefore bounded from above by

$$\max_{i \in T_j(\theta_j^*)} \frac{\ell_i}{\lambda} + f_j(T_j(\theta_j^*)) \leq \max_{i \in T_j(\theta)} \frac{\ell_i}{\lambda} + f_j(T_j(\theta)) \leq \frac{\theta}{\lambda} + f_j(T_j(\theta)),$$

where the first inequality follows from the definition of θ_j^* and the second inequality follows from the definition of $T_j(\theta)$.

It thus suffices to show that there exists some $\theta > 0$ such that $\theta/\lambda + f_j(T_j(\theta)) \leq \mu C$. To this end, define $h_j(\theta) := \sum_{i \in T_j(\theta)} \Delta_{ij}$ and notice that

$$\int_{\theta=0}^{C} h_j(\theta) d\theta = \sum_{i \in \bar{S}_j} \Delta_{ij} \left(C - \ell_i\right).$$

Note that $\ell_i + \frac{1}{\Delta_{ij}} x_{ij} \leq C$ for each $i \in \bar{S}_j$, by feasibility of the LP solution and the definition of ℓ_i. By summing over all $i \in \bar{S}_j$ and using the fact that $\sum_{i \in \bar{S}_j} x_{ij} \geq 1 - \lambda$, we obtain $1 - \lambda \leq \sum_{i \in \bar{S}_j} \Delta_{ij} \left(C - \ell_i\right)$ and, thus, $1 - \lambda \leq \int_{\theta=0}^{C} h_j(\theta) d\theta$.

Now assume by contradiction that $\theta/\lambda + f_j(T_j(\theta)) > \mu C$ for all $\theta > 0$. Note that this is equivalent to $g_j(T_j(\theta)) < \frac{1}{\mu C - \theta/\lambda}$ and hence we obtain

$$\frac{1}{\mu C - \theta/\lambda} > g_j(T_j(\theta)) \geq \sum_{i \in T_j(\theta)} \Delta_{ij} = h_j(\theta),$$

where the last inequality follows by Lemma 1 applied to $T = T_j(\theta)$. By combining the above bounds, we obtain

$$1 - \lambda \leq \int_{\theta=0}^{C} h_j(\theta) d\theta < \int_{\theta=0}^{C} \frac{1}{\mu C - \theta/\lambda} d\theta = \lambda \int_{\omega=\mu\lambda-1}^{\mu\lambda} \frac{1}{\omega} d\omega = \lambda \ln \frac{\mu\lambda}{\mu\lambda - 1}.$$

Using the definition of μ we obtain $\ln \frac{\mu\lambda}{\mu\lambda-1} = \ln e^{(1-\lambda)/\lambda} = \frac{1-\lambda}{\lambda}$, and hence the above inequality yields $1 - \lambda < 1 - \lambda$, a contradiction. This shows that $\theta_j^*/\lambda + f_j(T_j(\theta_j^*)) \leq \mu C$, thus completing the proof of the lemma. □

EC.3. Appendix to Section 4

EC.3.1. Hardness of Approximation for Fractionally Subadditive Processing Speeds (Theorem 5)

We prove the inapproximability result for ASSIGNMENT with fractionally subadditive processing speeds given by a value oracle. The proof follows from adapting a construction by Feige (2009) that shows a similar inapproximability for a welfare maximization problem.

THEOREM 5. *Any algorithm with approximation guarantee $o(n^{\frac{1}{3}})$ for fractionally subadditive processing speeds requires an exponential number of queries to a value oracle.*

Proof. We consider an instance of $n \geq 2$ jobs and $m = n^2$ machines, where the set of machines is partitioned into n sets $T_1, \dots, T_n$ of size n each. For each job j and set of machines S, let

$$f_j(S) := \frac{n}{\max\{|S| + n^{5/3}, n \cdot |S \cap T_j|\}}.$$

It is easy to verify that the processing speeds $g_j(S) := 1/f_j(S)$ are fractionally subadditive for every job j, and that the optimal makespan in the above instance is $\mathrm{OPT} = 1/n$ (i.e., when each job j is processed just on its corresponding set T_j).

The partition $T_1, \dots, T_n$ is unknown to the decision maker, who can only access the values of f_j via a value oracle. By the argumentation of Feige (2009, Section 1.2), however, the decision maker cannot find any set S with $|S \cap T_j| \geq \frac{|S|}{n} + n^{2/3}$ within a polynomial number of value queries, since the probability of choosing any such set S when selecting a subset of M uniformly at random is smaller than the inverse of any polynomial in n. Hence, we can assume that any algorithm using only a polynomial number of value queries will return an assignment $\mathbf{S}$ with $f_j(S_j) = \frac{n}{|S_j| + n^{5/3}}$ for all $j \in J$.

Now, if there is even a single job j with $|S_j| \leq n^{5/3}$, then $f_j(S_j) \geq \frac{1}{2} \cdot n^{-2/3} = \Omega(n^{1/3}) \cdot \mathrm{OPT}$. On the other hand, if $|S_j| \geq n^{5/3}$ for all $j \in J$, then by a volume argument there will be at least one machine running at least $n^{2/3}$ jobs, and each of these jobs takes at least time $\Omega(1/n)$. Thus our makespan is off from OPT by at least a factor of $\Omega(n^{2/3})$. $\square$

EC.3.2. Approximating Truncated Welfare for Subadditive Functions using a Demand Oracle

In this section, we explain how to obtain a polynomial-time $(6+\varepsilon)$-approximation for Truncated Welfare for any fixed $\varepsilon > 0$ when the functions g_j are subadditive and we are given access to a demand oracle that given $p \in \mathbb{R}^M_{\geq 0}$ returns a set $S \subseteq M$ maximizing $g_j(S) - \sum_{i \in S} p_i$. The approximation follows from a result by Feige (2009), who devised a 2-approximation for Welfare Maximization (the special case of the Truncated Welfare problem with $t = \infty$) for monotone subadditive functions given by a demand oracle. As an interesting artifact of our results, we show that the assumption of monotonicity is actually not necessary to obtain this 2-approximation; see Appendix EC.3.2.4.

We remark that, although truncating a function preserves subadditivity, Feige's (2009) result for Welfare Maximization does not immediately imply a 2-approximation for Truncated Welfare as the demand oracle for the original function does not yield a demand oracle for the truncated function. Instead, we show that those demand oracles can be used to obtain an *approximate dual separation routine* for the LP relaxation used by Feige (2009) that then allows us to apply his algorithm.

EC.3.2.1. LP Relaxation and Feige's Algorithm Feige (2009) establishes the following "oblivious rounding" result. Given a set of monotone subadditive functions $h_j : M \to \mathbb{R}_{\geq 0}$ for $j \in J$, let x^* be a (not necessarily optimal) solution to the following LP:

$$\begin{aligned}
\textbf{maximize:} \quad & \sum_{j \in J} \sum_{S \subseteq M} h_j(S) x_{Sj} && && (\mathrm{P}_{\mathrm{Welfare}}) \\
\textbf{s.t.:} \quad & \sum_{S \subseteq M} x_{Sj} \leq 1 && \forall j \in J \\
& \sum_{j \in J} \sum_{S \subseteq M : i \in S} x_{Sj} \leq 1 && \forall i \in M \\
& x \geq 0
\end{aligned}$$

Feige (2009, Section 3.2.2) shows that, given the non-zero entries of x^*, we can find in polynomial time (in $|J|$, $|M|$, and the encoding size of x^*) disjoint sets S_j for $j \in J$ such that

$$\sum_{j \in J} h_j(S_j) \geq \frac{1}{2} \sum_{j \in J} \sum_{S \subseteq M} h_j(S) x^*_{Sj}.$$

Importantly, Feige's rounding algorithm is oblivious in the sense that, when given access to the solution x^*, it does not require any further access to the functions h_j.

EC.3.2.2. The $(6+\varepsilon)$-Approximation for Truncated Welfare Now consider an instance of Truncated Welfare. In the following, we will assume that each g_j for $j \in J$ is monotone. We show in Appendix EC.3.2.4 that this assumption is without loss of generality. We construct an instance of Welfare Maximization with functions h_j for $j \in J$ by defining $h_j(S) := \min\{g_j(S), t\}$ for $j \in J$ and $S \subseteq M$. Note that because each function g_j is subadditive and monotone, so are the truncated functions h_j. Note further that with this choice of h_j, the optimal value Z^* of ($\mathrm{P}_{\mathrm{Welfare}}$) is an upper bound on the optimal value of the Truncated Welfare instance, as any solution to the latter induces a solution to the former of the same value.

In Appendix EC.3.2.3 below, we show that a demand oracle for the functions g_j can be used to compute in polynomial time a solution x^* to ($\mathrm{P}_{\mathrm{Welfare}}$) with $\sum_{j\in J}\sum_{S\subseteq M} h_j(S)x^*_{Sj} \geq \frac{1}{3+\varepsilon}Z^*$. We can thus apply Feige's algorithm to x^* (without any further calls to the demand oracles). This yields disjoint sets S_j for $j \in J$ with $\sum_{j\in J} h_j(S_j) \geq \frac{1}{6+2\varepsilon}Z^*$. Because, Z^* is an upper bound on the optimal value of the Truncated Welfare instance, we thus obtain a $(6+\varepsilon)$-approximation for Truncated Welfare (when using $\varepsilon/2$ instead of ε in the subroutine described below).

EC.3.2.3. A $(3+\varepsilon)$-Approximation to ($\mathrm{P}_{\mathrm{Welfare}}$) via Approximate Dual Separation In the following, we show how a demand oracle for the functions g_j for $j \in J$ can be used to obtain an efficient 3-approximation for ($\mathrm{P}_{\mathrm{Welfare}}$) via devising an approximate separation routine for the dual of ($\mathrm{P}_{\mathrm{Welfare}}$). To this end, consider the linear program ($\mathrm{D}(\mathcal{S},\lambda)$) for $\lambda > 0$ and $\mathcal{S} \subseteq 2^M$

$$\begin{aligned} \textbf{minimize:} \quad & \sum_{j\in J} y_j + \sum_{i\in M} z_i && (\mathrm{D}(\mathcal{S},\lambda)) \\ \textbf{s.t.:} \quad & y_j + \sum_{i\in S} z_i \;\geq\; \lambda h_j(S) \quad \forall j \in J,\, S \in \mathcal{S} \\ & y, z \;\geq\; 0 \end{aligned}$$

and note that ($\mathrm{D}(2^M, 1)$) is the dual to ($\mathrm{P}_{\mathrm{Welfare}}$).

THEOREM EC.3. *There is an algorithm that, given access to a demand oracle for g_j for all $j \in J$ and $\varepsilon > 0$, computes in polynomial time a feasible solution (y', z') to $(\mathrm{D}(2^M, 1))$ and a set $\mathcal{S} \subseteq 2^M$ such that the optimal value of $(\mathrm{D}(\mathcal{S}, 3))$ is bounded from below by $\frac{1}{1+\varepsilon} \cdot \left(\sum_{j \in J} y'_j + \sum_{i \in M} z'_i \right)$.*

Before we prove this theorem, we observe that it indeed implies that we can obtain a solution to $(\mathrm{P}_{\mathrm{Welfare}})$ whose value is within a factor of $3 + \varepsilon$ of the optimal LP value Z^*. To this end, let y', z' and $\mathcal{S}$ be the output of the algorithm described by the theorem. Now consider the dual to $(\mathrm{D}(\mathcal{S}, 3))$:

$$\begin{aligned}
\textbf{maximize:} \quad & \sum_{j \in J} \sum_{S \in \mathcal{S}} 3 h_j(S) x_{Sj} && && (\mathrm{P}'(\mathcal{S})) \\
\textbf{s.t.:} \quad & \sum_{S \in \mathcal{S}} x_{Sj} \;\leq\; 1 && \forall j \in J \\
& \sum_{j \in J} \sum_{S \in \mathcal{S} : i \in S} x_{Sj} \;\leq\; 1 && \forall i \in M \\
& x \;\geq\; 0
\end{aligned}$$

Let x^* be an optimal solution to $(\mathrm{P}'(\mathcal{S}))$, which we can compute efficiently as $\mathcal{S}$ is of polynomial size (because the algorithm in Theorem EC.3 runs in polynomial time). We conclude that

$$3 \cdot \sum_{j \in J} \sum_{S \in \mathcal{S}} h_j(S) x^*_{Sj} \geq \frac{1}{1+\varepsilon} \cdot \left(\sum_{j \in J} y'_j + \sum_{i \in M} z'_i \right) \geq \frac{Z^*}{1+\varepsilon},$$

where the first inequality follows from LP duality and the fact the optimal value of $(\mathrm{D}(\mathcal{S}, 3))$ is bounded from below by $\sum_{j \in J} y'_j + \sum_{i \in M} z'_i$, and the second inequality follows from the fact that y', z' is a feasible solution to $(\mathrm{D}(2^M, 1))$, which has the same optimal value as $(\mathrm{P}_{\mathrm{Welfare}})$.

Because x^*, extended by $x^*_{Sj} = 0$ for all $j \in J$ and $S \in 2^M \setminus \mathcal{S}$, is also a feasible solution to $(\mathrm{P}_{\mathrm{Welfare}})$, we obtained a $(3+\varepsilon)$-approximation (when replacing ε with $\frac{\varepsilon}{3}$). It thus remains to prove Theorem EC.3. This is based on the algorithm described in the proof of the following lemma.

LEMMA EC.5. *There is a polynomial time algorithm that, given access to a demand oracle for g_j for all $j \in J$ and vectors $y \in \mathbb{R}^J_{\geq 0}$ and $z \in \mathbb{R}^M_{\geq 0}$, either correctly asserts that (y, z) is a feasible solution to $(\mathrm{D}(2^M, 1))$ or it finds $j \in J$ and $S \subseteq M$ such that $y_j + \sum_{i \in S} z_i < 3 \cdot h_j(S)$.*

Proof. For each $j \in J$, the algorithm checks which of the four cases listed below holds and reports one of two possible outcomes: Either it asserts that all constraints of (D(2^M, 1)) pertaining j are fulfilled, i.e.,

$$\textbf{feasible}(j): \qquad y_j + \sum_{i \in S} z_i \geq h_j(S) \qquad \forall\, S \subseteq M$$

or it finds a violated constraint of (D(2^M, 3)) for j and some $S \subseteq M$, i.e.,

$$\textbf{violated}(S, j): \qquad y_j + \sum_{i \in S} z_i < 3h_j(S).$$

Case 1: It holds that $t \leq y_j$. In this case, assert **feasible**(j).

Case 2: Case 1 does not hold, but $y_j + z_i < h_j(\{i\})$ for some $i \in M' := \{i \in M : g_j(\{i\}) \geq t\}$. In this case, report **violated**$(\{i\}, j)$.

Case 3: Cases 1 and 2 do not hold, but $g_j(S^*) \leq y_j + \sum_{i \in S^*} z_i$, where $\bar{M} := \{i \in M : g_j(\{i\}) < t\}$ and $S^* \in \arg\max_{S \subseteq \bar{M}} g_j(S) - \sum_{i \in S} z_i$. In this case, assert **feasible**(j).

Case 4: Cases 1, 2, and 3 do not hold. Construct a partition $\mathcal{U}$ of S^* such that $g_j(U) \leq 2t$ for all $U \in \mathcal{U}$ and there is $U' \in \mathcal{U}$ such that $g_j(U) \geq t$ for all $U \in \mathcal{U} \setminus \{U'\}$. We show below that such a partition can be constructed and moreover, that there is at least one $\bar{U} \in \mathcal{U}$ with $y_j + \sum_{i \in U} z_i < 3h_j(\bar{U})$. Report **violated**$(\bar{U}, j)$.

If **feasible**(j) is reported for all $j \in J$, the algorithm asserts that y, z is feasible for (D(2^M, 1)); otherwise it returns some $j \in J$ and $S \subseteq M$ for which **violated**(S, j) was reported.

Next, we argue that the assertion made by the algorithm for each case is correct and that, moreover, the corresponding computation can be executed in polynomial time using a demand oracle for g_j.

In Case 1, it holds that $t \leq y_j$ and hence $y_j + \sum_{i \in S} z_i \geq t \geq h_j(S)$ for all $S \subseteq M$ because $z_i \geq 0$ for all $i \in M$. Hence the algorithm correctly asserts **feasible**(j) in this case. This case can be checked in constant time.

In Case 2, the condition $y_j + z_i < h_j(\{i\})$ immediately implies the correctness of the assertion **violated**$(\{i\}, j)$. Moreover, this case can be checked in linear time.

Now assume the algorithm asserts **feasible**(j) in Case 3. Let $S \subseteq M$. We distinguish two subcases.

- If $S \subseteq \bar{M}$ then $h_j(S) - \sum_{i \in S} z_i \leq g_j(S) - \sum_{i \in S} z_i \leq g_j(S^*) - \sum_{i \in S^*} z_i \leq y_j$ by choice of S^*.
- If $S \not\subseteq \bar{M}$ then there is $i' \in M'$ with $i' \in S$. This implies $y_j + \sum_{i \in S} z_i \geq y_j + z_{i'} \geq h_j(\{i'\}) = t$,

where the first inequality follows $z \geq 0$, the second inequality follows from the fact that Case 2 does not hold, and the final ineqaulity follows from $g_j(\{i'\}) \geq t$ by $i' \in M'$.

Thus, in both subcases, $y_j + \sum_{i \in S} z_i \geq h_j(S)$ and the algorithm correctly asserts **feasible**(j). Moreover, note that $\bar{M}$ can be constructed in linear time and that the set S^* can be obtained by a single call to the demand oracle, using $p_i := z_i$ for $i \in \bar{M}$ and $p_i := \infty$ for $i \in M'$. Hence Case 3 can be checked efficiently.

Finally, we show that in Case 4, there exists a partition $\mathcal{U}$ of S^* with the desired properties. The partition can be constructed greedily as follows:

1. Start with $\mathcal{U} := \emptyset$, $U' := S^*$, and $U := \emptyset$.
2. While $g_j(U') > 2t$, repeat the following steps:
   - Let $i \in U'$ be arbitray, set $U' := U' \setminus \{i\}$ and $U := U \cup \{i\}$.
   - If $g_j(U) > t$, then set $\mathcal{U} := \mathcal{U} \cup \{U\}$ and reset $U := \emptyset$.
3. Add U' to $\mathcal{U}$.

Note that whenever some set U is added to $\mathcal{U}$ in Step 2 above, then $g_j(U \setminus \{i\}) < t$ for the element i that is added to U in this iteration of the while loop, as otherwise U would have been reset to $\emptyset$ in the preceding iteration. Hence subadditivity implies $g_j(U) \leq g_j(U \setminus \{i\}) + g_j(\{i\}) \leq 2t$, because $i \in S^* \subseteq \bar{M}$ and hence $g_j(\{i\}) \leq t$. Therefore $g_j(U) \leq 2t$ for all $U \in \mathcal{U}$ (note that the constraint is trivially fulfilled for U' at the end of the procedure). Moreover, $g_j(U) \geq t$ for all $U \in \mathcal{U} \setminus \{U'\}$ by construction.

Now consider any such a partition $\mathcal{U}$ of S^*, i.e., $g_j(U) \leq 2t$ for all $U \in \mathcal{U}$, and there is $U' \in \mathcal{U}$ such that $g_j(U) \geq t$ for at all $U \in \mathcal{U} \setminus \{U'\}$. Assume by contradiction that $y_j + \sum_{i \in U} z_i \geq 3h_j(U)$ for all

$U \in \mathcal{U}$. Note that in particular $g_j(U') \leq 3h_j(U') \leq y_j + \sum_{i \in U'} z_i$, because $g_j(U') \leq 2t$. Furthermore, we obtain

$$g_j(U) \leq 2t \leq y_j - t + \sum_{i \in U} z_i \leq \sum_{i \in U} z_i$$

for $U \in \mathcal{U} \setminus \{U'\}$, where the first inequality follows from the construction of $\mathcal{U}$, the second inequality follows $3t = 3h_j(U) \leq y_j + \sum_{i \in U} z_i$ because $h_j(U) = \min\{g_j(U), t\} = t$ for $U \neq U'$, and the third inequality follows from the fact that $t > y_j$ because Case 1 does not hold. We thus obtain the contradiction

$$y_j + \sum_{i \in S^*} z_i < g_j(S^*) \leq g_j(U') + \sum_{U \in \mathcal{U} \setminus \{U'\}} g_j(U) \leq y_j + \sum_{i \in S^*} z_i,$$

where the first inequality follows because Case 3 does not hold, the second inequality follows from subadditivity, and the final inequality follows from $g_j(U) \leq \sum_{i \in U} z_i$ for $U \in \mathcal{U} \setminus \{U'\}$ and $g_j(U') \leq y_j + \sum_{i \in U'} z_i$ as derived above.

Thus, there is $\bar{U} \in \mathcal{U}$ with $y_j + \sum_{i \in \bar{U}} z_i < 3h_j(\bar{U})$ and the algorithm correctly reports **violated**$(\bar{U}, j)$. The partition $\mathcal{U}$ can be constructed in linear time using the greedy algorithm sketched above. Once the partition is computed, the set $\bar{U}$ can be found in linear time by enumeration. □

We are finally ready to prove Theorem EC.3.

Proof of Theorem EC.3. For $\mu \in \mathbb{R}$ define the two polyhedra

$$Q(\mu) := \{(y,z) \in \mathbb{R}^{J \times M} : (y,z) \text{ is a solution to } (\mathrm{D}(2^M, 3)) \text{ of value at most } \mu\} \text{ and}$$

$$\bar{Q}(\mu) := \{(y,z) \in \mathbb{R}^{J \times M} : (y,z) \text{ is a solution to } (\mathrm{D}(2^M, 1)) \text{ of value at most } \mu\}.$$

Note that Lemma EC.5 establishes that the *approximate separation problem* for $Q(\mu)$ and its approximation $\bar{Q}(\mu)$ can be solved in polynomial time: Given $y \in \mathbb{R}^J$ and $z \in \mathbb{R}^M$, we can either confirm $(y,z) \in \bar{Q}(\mu)$, or we can find a hyperplane separating (y,z) from $Q(\mu)$. As observed by Schulz and Uhan (2013, Theorem A.6), the ellipsoid method can be used to turn such an algorithm into a polynomial-time algorithm for solving the following *approximate non-emptiness problem*:

Find a vector $(y,z) \in \bar{Q}(\mu)$ or assert that $Q(\mu)$ is empty. In the latter case, we also obtain a collection $\mathcal{S} \subseteq 2^M$ of subsets of M corresponding to the constraints generated to prove that $Q(\mu)$ is empty, i.e., the polyhedron

$$Q_{\mathcal{S}}(\mu) := \{(y,z) \in \mathbb{R}^{J \times M} : (y,z) \text{ is a solution to } (\mathrm{D}(\mathcal{S},3)) \text{ of value at most } \mu\}$$

is empty.

We embed the algorithm for the approximate non-emptiness problem in a binary search. We can make the following two assumptions without loss of generality:

- For $\mu = t$, the algorithm asserts that $Q(t)$ is empty. Assume that instead it finds a solution $(y,z) \in \bar{Q}(t)$. If there is $j \in J$ with $g_j(M) \geq t$, then setting $\mathcal{S} = \{M\}$ ensures that $Q_{\mathcal{S}}(t)$ is empty. If there is no $j \in J$ with $g_j(M) \geq t$, then $g_j(S) = h_j(S)$ for all $S \subseteq M$. Thus, the demand oracle for the functions g_j is also a demand oracle for the functions h_j and we can compute an optimal solution (y',z') to $(\mathrm{D}(2^M,1))$ via exact separation.
- For $\mu = |J|t$, the algorithm returns $(y,z) \in \bar{Q}(|J|t)$. Indeed, $y_j = t$ for a $j \in J$ and $z_i = 0$ for $i \in M$ is a feasible solution to $(\mathrm{D}(2^M,1))$ of value $|J|t$.

Thus starting with the $\mu_L := t$ and the upper bound $\mu_U := |J|t$, we can embed the algorithm for the approximate non-emptiness problem in a binary search: As long as $\mu_L < \frac{\mu_U}{1+\varepsilon}$, iteratively query the algorithm for $\mu := \frac{\mu_L + \mu_U}{2}$; if $Q(\mu)$ is asserted to be empty, set $\mu_L := \mu$; if a solution to $\bar{Q}(\mu)$ is returned, set $\mu_U := \mu$. After $\log \frac{|J|}{1+\varepsilon}$ iterations this procedure terminates with $\mu_L \geq \frac{\mu_U}{1+\varepsilon}$ yielding $\mathcal{S} \subseteq 2^M$ and $(y',z') \in \mathbb{R}^{J \times M}$ such that $Q_{\mathcal{S}}(\mu_L)$ is empty and $(y',z') \in \bar{Q}(\mu_U)$. Note that by definition of $\bar{Q}$, the vector (y',z') is a feasible solution to $(\mathrm{D}(2^M,1))$ of value μ_U. Moreover, emptiness of $Q_{\mathcal{S}}(\mu_L)$ implies that the optimal value of $(\mathrm{D}(\mathcal{S},3))$ is bounded from below by $\mu_L \geq \frac{\mu_U}{1+\varepsilon} \geq \frac{\sum_{j \in J} y'_j + \sum_{i \in M} z'_i}{1+\varepsilon}$. Thus, y', z' and $\mathcal{S}$ fulfil the requirements of Theorem EC.3, completing the proof. □

EC.3.2.4. Assuming Monotonicty is Without Loss of Generality Finally, we argue that the assumption that g_j is monotone for each $j \in J$ is without loss of generality. Indeed, if this is not the case, we can replace g_j by $g'_j(S) := \max_{U \subseteq S} g_j(U)$. The functions $g'_j(S)$ are monotone and

subadditive by Lemma EC.4 in Appendix EC.2.1. Note further that the instances of Truncated Welfare I for the functions g_j and I' for the functions g'_j, respectively, are equivalent in the following sense: Any solution $\mathbf{S}$ to I is also a solution to I' with at least the same value (as $g'_j(S) \geq g_j(S)$ for all $j \in J$ and $S \subseteq M$). Moreover, for any solution $\mathbf{S}'$ to I' we can compute in polynomial time a solution $\mathbf{S}$ to I with the same value by using, for each $j \in J$, the demand oracle for g_j to obtain $S_j \in \arg\max_{U \subseteq S'_j} g_j(U)$ (this can be done by calling the oracle with $p_i := 0$ for $i \in S'_j$ and $p_i := \infty$ for $i \in M \setminus S'_j$; note further that $g_j(S_j) = g'_j(S'_j)$ for all $j \in J$ and that because the S'_j for $j \in J$ are disjoint, so are the S_j). Thus, in particular, the instances I and I' have the same optimal value and we can transform any solution to I' into a solution to I with the same value (and therefore also the same approximation guarantee).

Finally, the following lemma reveals that any demand oracle for g_j is also a demand oracle for g'_j. Thus, we can apply any approximation algorithm for Truncated Welfare with monotone subadditive functions given by a demand oracle to I' to obtain the same approximation guarantee for I.

Lemma EC.6. *Let $g : 2^M \to \mathbb{R}_{\geq 0}$ and define $g'(S) := \max_{U \subseteq S} g(U)$ for each $S \subseteq M$. Let $p \in \mathbb{R}^M_{\geq 0}$ and $T \in \arg\max_{S \subseteq M} g(S) - \sum_{i \in S} p_i$. Then $g(T) = g'(T)$ and $T \in \arg\max_{S \subseteq M} g'(S) - \sum_{i \in S} p_i$.*

Proof. Let $p \in \mathbb{R}^M_{\geq 0}$. Consider the respective maximizers $T \in \arg\max_{S \subseteq M} g(S) - \sum_{i \in S} p_i$ and $T' \in \arg\max_{S \subseteq M} g'(S) - \sum_{i \in S} p_i$. Note that $g(T) - \sum_{i \in T} p_i \leq g'(T') - \sum_{i \in T'} p_i$ because $g(S) \leq g'(S)$ for all $S \subseteq M$. Now let $U' \subseteq T'$ with $g(U') = \max_{S \subseteq T'} g(S) = g'(T')$ and observe that

$$g'(T') - \sum_{i \in T'} p_i \;\leq\; g(U') - \sum_{i \in U'} p_i \;\leq\; g(T) - \sum_{i \in T} p_i,$$

where the first inequality follows from $g'(T') = g(U')$ by choice of U', as well as $U' \subseteq T'$ and $p \geq 0$, and the second inequality follows from maximality of T. We conclude that $g(T) - \sum_{i \in T} p_i = g'(T') - \sum_{i \in T'} p_i$. □

Remark EC.1. We remark that the same argumentation also applies to Feige's (2009) original 2-approximation for Welfare Maximization with subadditive functions. That is, although

monotonicity is listed as an assumption by Feige (2009, Section 1), the 2-approximation can also be applied to non-monotone subadditive functions given by a demand oracle, using the transformation described above.

EC.3.3. Proof of Lemma 10

LEMMA 10. *Assume there exists an assignment* $\mathbf{S}$ *for some set of jobs* $J_1 \subseteq J$ *with maximum load* C' *such that* $|S_j| = 1$ *for all* $j \in J_1$*. Then we can construct in polynomial-time an assignment that assigns at least* $|J_1|$ *jobs (possibly different from* J_1*) with a maximum load at most* $2C'$*.*

Proof. Let $J_1 \subseteq J$ be the subset of $|J_1|$ jobs that are assigned without splitting in the given partial assignment and let $i_j \in M$ be the machine where each job $j \in J_1$ is assigned to. We consider the following LP:

$$\begin{aligned}
\textbf{maximize:} \quad & \sum_{i\in M}\sum_{j\in J} x_{ij} && && \text{(LP-H)}\\
\textbf{s.t.:} \quad & \sum_{i\in M} x_{ij} \;\le\; 1 && && \forall j \in J\\
& \sum_{j\in J} \frac{1}{g_j(\{i\})} x_{ij} \;\le\; C' && && \forall i \in M\\
& x_{ij} \;\equiv\; 0 && \forall i \in M, j \in J \text{ with } f_j(\{i\}) > C' &&\\
& x \;\ge\; 0 && &&
\end{aligned}$$

By setting $x_{ij} = 1$ for each $j \in J_1$ and $i = i_j$, and $x_{ij} = 0$, otherwise, we can easily verify that all the constraints of (LP-H) are trivially satisfied. Further, for the same assignment, the objective becomes $\sum_{i\in M}\sum_{j\in J} x_{ij} = |J_1|$. Hence, we conclude that (LP-H) is feasible and its optimal value is at least $|J_1|$. Let x be an optimal extreme point solution of (LP-H), which satisfies $\sum_{i\in M}\sum_{j\in J} x_{ij} \ge |J_1|$. Given that for each job j it holds $\sum_{i\in M} x_{ij} \le 1$, it follows that at least $|J_1|$ jobs appear in the support of x (that is, $\sum_{i\in M} x_{ij} > 0$ for at least $|J_1|$ jobs). Thus, it suffices to prove that we can round x into a feasible schedule in a way such that any job that appears in its support is integrally assigned to a machine.

By working along the lines of Lenstra et al. (1990) (see Appendix EC.1.5), we can construct an assignment graph based on the support of x. It can be verified that, if x is an extreme point solution of (LP-H), then the graph is a pseudoforest (Shmoys and Tardos 1993). Therefore, by using the rounding algorithm of Lenstra et al. (1990) we can schedule all the jobs in the support of x within makespan at most $2C'$ and, thus, the desired lemma follows. □