APPENDIX

Lemma 7.1 ((4.1 in Section IV)): TBD-DEC is NP-Complete in the strong sense.

Proof: To prove this result, we use a reduction to the 3-Partition problem [18]. Indeed, let us consider an instance of 3-Partition consisting of $3m$ items a_i such that $\sum a_i = mB$ and $\forall i,\ \frac{B}{4} < a_i < \frac{B}{2}$ and let us set $\forall j,\quad d_j = 3,\quad b_j = B$, $n = 3m$, $\forall i,\ w_i = a_i$ and $K = mB$. Since the overall out degree of the servers is at most $3m$ and since all $3m$ clients must be used in order to reach throughput mB, each server must be connected to exactly 3 clients and no client should be connected to more than one server. Since the overall capacity of the servers is $m \times B$, each server must be connected to 3 clients whose aggregated capacity is exactly B, what achieves the NP-Completeness proof. ■

Lemma 7.2 (4.3 in Section IV)): If $\mathcal{LC} \preceq \mathcal{R}$, and $\mathcal{LC}'$ and $\mathcal{R}'$ are obtained from $\mathcal{LC}$ and $\mathcal{R}$ as described above, then $\mathcal{LC}' \preceq \mathcal{R}'$.

Proof: We begin by proving two lower bounds for $\mathcal{R}'(1,k)$. Since $\mathcal{R}'$ is obtained from $\mathcal{R}$ by a valid allocation, there exists a set $C \subseteq [1,n]$ of chosen clients, and assigned values v_i for $i \in C$ such that: Card$(C) \leq d$, $\forall i, v_i \leq \mathcal{R}_i$ (where $\mathcal{R}_i$ denotes the capacity of the i^{th} client in $\mathcal{R}$), and $\sum v_i \leq b$. There also exists a sorting permutation σ such that $\mathcal{R}'_{\sigma(i)} = \mathcal{R}_i$ if $i \notin C$, and $\mathcal{R}'_{\sigma(i)} = \mathcal{R}_i - v_i$ if $i \in C$. We can then write $\mathcal{R}'(1,k)$ in two different ways,

$$\begin{aligned}\mathcal{R}'(1,k) &= \sum_{i:i\notin C\wedge\sigma(i)\leq k} \mathcal{R}_i + \sum_{i:i\in C\wedge\sigma(i)\leq k} \mathcal{R}_i - v_i \\ &= \sum_{i:\sigma(i)\leq k} \mathcal{R}_i - \sum_{i:i\in C\wedge\sigma(i)\leq k} v_i\end{aligned}$$

For $k > d$, since there are at least $k-d$ indexes i such that $i \notin c \wedge \sigma(i) \leq k$, and since $\mathcal{R}(1,k-d)$ is the sum of the $k-d$ smallest $\mathcal{R}_i$ values, then $\sum_{i:i\notin c\wedge\sigma(i)\leq k} \mathcal{R}_i \geq \mathcal{R}(1,k-d)$. Together with $\mathcal{R}_i - v_i \geq 0$, we obtain the first upper bound

$$\mathcal{R}'(1,k) \geq \mathcal{R}(1,k-d) \qquad \forall k > d. \tag{4}$$

Similarly, since there are k indexes i such that $\sigma(i) \leq k$, then $\sum_{i:\sigma(i)\leq k} \mathcal{R}_i \geq \mathcal{R}(1,k)$. Together with $\sum_{i\in C} v_i \leq b$, we obtain the second upper bound

$$\mathcal{R}'(1,k) \geq \mathcal{R}(1,k) - b. \tag{5}$$

To complete the proof, we need to evaluate $\mathcal{LC}'(1,k)$. Since we identified three main situations when adding a server, we evaluate $\mathcal{LC}'(1,k)$ in each possible situation.

Case 1 $\exists l$ such that $\mathcal{LC}(l,l+d-1) < b$ and $\mathcal{LC}(l,l+d) \geq b$: In this case (see lines 4 to 8 in Algorithm 1) the algorithm allocates completely clients $\mathcal{C}_l, \mathcal{C}_{l+1}, \ldots, \mathcal{C}_{l+d-1}$ to $\mathcal{S}$ and only partially client $\mathcal{C}_{l+d}$, whose remaining capacity is w'_{l+d}. The first d clients of the list $\mathcal{LC}'$ will thus have zero capacity, and $\mathcal{C}'_{l+d}$ will be reinserted at the same position as pointed out earlier. Then, the updated list $\mathcal{LC}'$ is equal to

$$\begin{aligned}&\{\mathcal{C}'_l = 0, \ldots, \mathcal{C}'_{l+d-1} = 0, \mathcal{C}_1, \ldots, \mathcal{C}_{l-1},\\ &\mathcal{C}'_{l+d} = \mathcal{LC}(l,l+d) - b, \mathcal{C}_{l+d+1}, \ldots, \mathcal{C}_n\}.\end{aligned}$$

Then, for $k \leq d$, $\mathcal{LC}'(1,k)$ is a sum over the completely allocated reinserted clients, and thus $\mathcal{LC}'(1,k) = 0$. For the second interval $d < k \leq l-1+d$, $\mathcal{LC}'(1,k)$ is a sum of the first $k-d$ capacities in $\mathcal{LC}$, since they were shifted by d positions (due to the insertion of d clients at the beginning of the list), and so $\mathcal{LC}'(1,k) = \mathcal{LC}(1,k-d)$. If the interval includes one more client, *i.e.,* $d < k \leq l+d$, the sum is the same than in the previous interval, but the last element in the sum is replaced by the size of the split client that has been inserted, $\mathcal{LC}'(1,k) = \mathcal{LC}(1,k-d-1) + w'_{l+d}$. Finally when $l+d < k$, the sum is equal to the sum in the original list, decreased by the total capacity allocated to $\mathcal{S}$, $\mathcal{LC}'(1,k) = \mathcal{LC}(1,k) - b$.

Now, using Equations (4) and (5), and the fact that $\mathcal{LC} \preceq \mathcal{R}$, we have:

$$\begin{aligned}\mathcal{LC}'(1,k) &= 0 \leq \mathcal{R}'(1,k) \quad \text{for} \quad k \leq d\\ \mathcal{LC}'(1,k) &= \mathcal{LC}(1,k-d-1) + w'_{l+d}\\ &\leq \mathcal{LC}(1,k-d) \leq \mathcal{R}(1,k-d)\\ &\leq \mathcal{R}'(1,k) \quad \text{for} \quad d < k \leq l+d\\ \mathcal{LC}'(1,k) &= \mathcal{LC}(1,k) - b \leq \mathcal{R}(1,k) - b\\ &\leq \mathcal{R}'(1,k) \quad \text{for} \quad l+d < k.\end{aligned}$$

Case 2 $A(1,d) \geq b$: In this case (see lines 9 to 14 in Algorithm 1), since SEQ uses the first $l \leq d$ clients, there is no reordering of the list. The new list $\mathcal{LC}'$ can therefore be written as $\{\mathcal{C}'_1, \ldots, \mathcal{C}'_{l-1}, \mathcal{C}'_l, \mathcal{C}_{l+1}, \ldots, \mathcal{C}_n\}$, where $\mathcal{C}'_i$ has zero capacity for $i < l$. Moreover, since the overall allocated capacity is equal to b, then $\mathcal{LC}'(1,k) = 0$ when $k \leq l-1$ and $\mathcal{LC}'(1,k) = \mathcal{LC}(1,k) - b$ for $k > l-1$. Hence, Equation (5) combined to $\mathcal{LC} \preceq \mathcal{R}$ leads to $\mathcal{LC}'(1,k) \leq \mathcal{R}'(1,k)$.

Case 3 $A(n-d,n) < b$: In this case (see lines 15 to 18 in Algorithm 1), SEQ allocates completely the d last clients to $\mathcal{S}$, and therefore all reinserted clients $\mathcal{C}'_i$ will have zero capacity and will be reinserted at the beginning of the list. The new list $\mathcal{LC}'$ can therefore be written as $\{\mathcal{C}'_{n-d+1}, \ldots, \mathcal{C}'_n, \mathcal{C}_1, \ldots, \mathcal{C}_{n-d}\}$. Therefore, $\mathcal{LC}'(1,k) = 0$ when $k \leq d$ and $\mathcal{LC}'(1,k) = \mathcal{LC}(1, k-(d+1))$ for $k > d$. Once again, Equation (4) combined with $\mathcal{LC} \preceq \mathcal{R}$ leads to $\mathcal{LC}'(1,k) \leq \mathcal{R}'(1,k)$. ■

Lemma 7.3 ((5.4 in Section V)): If $\mathcal{R}$ is an augmented version of $\mathcal{LC}$, then $\mathcal{R}'$ is an augmented version of $\mathcal{LC}'$, and the allocations $\mathcal{A}$ and $\mathcal{B}$ differ by at most 4 changes.

Proof: Let $\mathcal{R}$ consist of $\{\mathcal{C}_1, \ldots, \mathcal{C}_{p-1}, \mathcal{X}, \mathcal{C}'_p, \mathcal{C}_{p+1}, \ldots, \mathcal{C}_n\}$, where $w'_p = w_p + y$ and the capacity of $\mathcal{X}$ ($w_\mathcal{X}$) is equal to x. The first step of the proof consists in computing the partial sums $\mathcal{R}(u, v)$ for any $u \leq v \leq n$. A quick case study shows that

$$\mathcal{R}(u,v) = \begin{cases} \mathcal{LC}(u-1, v-1) & \text{if } p < u-1, \\ \mathcal{LC}(u-1, v-1) + y & \text{if } p = u-1, \\ \mathcal{LC}(u, v-1) + x + y & \text{if } u \leq p < v, \\ \mathcal{LC}(u, v-1) + x & \text{if } p = v, \\ \mathcal{LC}(u, v) & \text{if } p > v. \end{cases}$$

In particular, since by hypothesis $x \leq w_p$ and $x + y \leq w_{p+1}$, then in all cases, $\mathcal{R}(u, v) \leq \mathcal{LC}(u, v)$. Furthermore, since $x \geq w_{p-1}$, $\mathcal{R}(u, v) \geq \mathcal{LC}(u-1, v-1)$ also holds in all cases.

Let us now consider the application of OSEQ(d, b) to $\mathcal{LC}$. Without loss of generality, we consider that a suitable interval $[l, l+d]$ has been found, *i.e.,* that $\mathcal{LC}(l, l+d-1) < b$ and $\mathcal{LC}(l+1, l+d) \geq b$. In that case, allocation $\mathcal{A}$ is $(\mathcal{C}_l, \ldots, \mathcal{C}_{l+d-1}, \mathcal{C}^{(a)}_{l+d})$, and the updated list $\mathcal{LC}'$ is $(\mathcal{C}_1, \ldots, \mathcal{C}_{l-1}, \mathcal{C}^{(b)}_{l+d}, \ldots, \mathcal{C}_n)$, where $\mathcal{C}^{(a)}_{l+d}$ and $\mathcal{C}^{(b)}_{l+d}$ are the two parts of the split client $\mathcal{C}_{l+d}$.

If the change from $\mathcal{LC}$ to $\mathcal{R}$ lies outside of the interval $[l, l+d]$ (*i.e.,* $p < l$ or $p > l+d$), then this change does not affect to the execution of Algorithm OSEQ, and allocations $\mathcal{A}$ and $\mathcal{B}$ are the same. In that case, thus, the result holds.

Otherwise, the resulting allocation $\mathcal{B}$ depends on the value of $\mathcal{R}(l+1, l+d)$. Indeed, our previous bounds for $\mathcal{R}(u, v)$ shows that $\mathcal{R}(l, l+d-1) < b$, and $\mathcal{R}(l+2, l+d+1) \geq b$. Thus, either $[l, l+d]$ or $[l+1, l+d+1]$ is the suitable interval for the application of OSEQ to $\mathcal{R}$.

Let us suppose that $\mathcal{R}(l+1, l+d) \geq b$*:* In this case, the resulting allocation $\mathcal{B}$ is $(\mathcal{C}_l, \ldots, \mathcal{X}, \mathcal{C}'_p, \ldots, \mathcal{C}^{(a)}_{l+d-1})$. It differs from $\mathcal{A}$ by 4 changes: the addition of $\mathcal{X}$, the removal of $\mathcal{C}^{(a)}_{l+d}$, and the modification of both $\mathcal{C}_p$ and $\mathcal{C}_{l+d-1}$. The updated list of clients $\mathcal{R}'$ is $(\mathcal{C}_1, \ldots, \mathcal{C}_{l-1}, \mathcal{C}^{(b)}_{l+d-1}, \mathcal{C}_{l+d}, \ldots, \mathcal{C}_n)$. It is thus an augmented version of $\mathcal{LC}'$, since $\mathcal{C}^{(b)}_{l+d-1}$ is inserted between $\mathcal{C}_{l-1}$ and the capacity of $\mathcal{C}^{(b)}_{l+d}$ is increased to w_{l+d}. Indeed, by definition of the splitting process, $w(\mathcal{C}^{(b)}_{l+d-1}) = \mathcal{R}(l, l+d) - b$ and $w(\mathcal{C}^{(b)}_{l+d}) = \mathcal{LC}(l, l+d) - b$, which implies $w(\mathcal{C}^{(b)}_{l+d-1}) \leq w(\mathcal{C}^{(b)}_{l+d})$.

If $p = l + d$, then $\mathcal{X}$ is the split client. This case actually results in only two changes in the allocations: the addition of (one part of) $\mathcal{X}$, and the removal of $\mathcal{C}^{(a)}_{l+d}$. The updated list $\mathcal{R}'$ is also an augmented version of $\mathcal{LC}'$, with the remaining part of $\mathcal{X}$ inserted and the capacity of $\mathcal{C}^{(b)}_{l+d}$ increased to $w_{l+d} + y$.

Let us now suppose that $\mathcal{R}(l+1, l+d) < b$*:* In this case, the suitable interval is $[l+1, l+d+1]$ and thus the resulting allocation $\mathcal{B}$ is $(\mathcal{C}_{l+1}, \ldots, \mathcal{X}, \mathcal{C}'_p, \ldots, \mathcal{C}^{(a')}_{l+d})$. Once again, it differs from $\mathcal{A}$ by 4 changes: the addition of $\mathcal{X}$, the removal of $\mathcal{C}_l$, and the modification of both $\mathcal{C}_p$ and $\mathcal{C}^{(a)}_{l+d}$. The updated list of clients $\mathcal{R}'$ is $(\mathcal{C}_1, \ldots, \mathcal{C}_l, \mathcal{C}^{(b')}_{l+d}, \ldots, \mathcal{C}_n)$. It is therefore an augmented version of $\mathcal{LC}'$, since $\mathcal{C}_l$ is inserted in right after $\mathcal{C}_{l-1}$ and the capacity of $\mathcal{C}^{(b)}_{l+d}$ is increased to $w(\mathcal{C}^{(b')}_{l+d})$. In this case, $w_l \leq w(\mathcal{C}^{(b)}_{l+d})$ comes from the ordering property of SEQ, see Section V-A.

If $p = l$, then $\mathcal{X}$ is actually not included in $\mathcal{B}$. Thus, this case results in two changes only: the modification of the capacity of $\mathcal{C}_p$, and the fact that $\mathcal{C}_{l+d}$ is split differently. The updated list $\mathcal{R}'$ is also an augmented version of $\mathcal{LC}'$, with $\mathcal{X}$ inserted and the capacity of $\mathcal{C}^{(b)}_{l+d}$ increased.

■

In the context of section VI, we analyze here a worst-case instance on which LCBC achieves a throughput significantly lower than SEQ, for the same resource augmentation on the degree. For fixed even m and B, let us consider an instance with m servers, each of capacity B and degree $d = \frac{B}{2}$, $\frac{mB}{2}$ *small* clients of size 1, with an additional *big* client of size $\frac{mB}{2}$.

On this instance, the solution of SEQ is to assign to each server d small clients and a part of size $\frac{B}{2}$ of the big client. This solution achieves a throughput of mB.

On the other hand, LCBC assigns first the big client to as few servers as possible ($\frac{m}{2}$ of them), with parts of size B. Once this is done, $\frac{m}{2}$ servers remain unused, and each can only accommodate $d+1$ small clients because of the degree constraint. The total throughput of this solution is thus $B\frac{m}{2} + (d+1)\frac{m}{2} = mB\left(\frac{1}{2} + \frac{1}{4} + \frac{1}{2B}\right)$.

Hence, when B grows, the ratio between the throughput of SEQ and LCBC tends to $\frac{3}{4}$.