Appendix A: Details for Section 2.4

IP formulation

By rearranging and taking the square on both sides of the submodular capacity constraint (5), we obtain:

$$V^2 y_i + \Big(\sum_{j=1}^{N} \mu_j x_{ij}\Big)^2 - 2V y_i \sum_{j=1}^{N} \mu_j x_{ij} \geq D(\alpha)^2 \cdot \sum_{j=1}^{N} b_j x_{ij}.$$

Note that since y_i is binary, we have $y_i^2 = y_i$. We next look at the term $y_i \sum_{j=1}^{N} \mu_j x_{ij}$. One can linearize this term by using one of the following two methods.

1. Since $y_i = 1$ if and only if at least one $x_{ij} = 1$, we have the constraint: $\sum_{j=1}^{N} x_{ij} \leq M y_i$, for a large positive number $\tilde{M}$ (actually, one can take $\tilde{M} = N$). Consequently, one can remove the y_i in the above term.
2. One can define a new variable $t_{ij} \triangleq y_i x_{ij}$ and add the four following constraints:

$$t_{ij} \leq y_i; \quad t_{ij} \leq x_{ij}; \quad t_{ij} \geq 0; \quad t_{ij} \geq x_{ij} + y_i - 1.$$

Next, we look at the term: $\big(\sum_{j=1}^{N} \mu_j x_{ij}\big)^2$. Since $x_{ij}^2 = x_{ij}$, we remain only with the terms $x_{ij} \cdot x_{ik}$ for $k > j$. One can now define a new variable for each such term, i.e., $z_{ijk} \triangleq x_{ij} \cdot x_{ik}$ with the four constraints as before:

$$z_{ijk} \leq x_{ij}; \quad z_{ijk} \leq x_{ik}; \quad z_{ijk} \geq 0; \quad z_{ijk} \geq x_{ij} + x_{ik} - 1.$$

The resulting formulation is a linear integer program. Note that the decision variables t_{ij} and z_{ijk} are continuous, while only x_{ij} and y_i are binary.

Appendix B: Proof of Theorem 1

Proof. In this proof, we make use of the submodular functions defined by Svitkina and Fleischer (2011) for load balancing problems. Denote the jobs by $1, 2, \cdots, N$, and for every subset of jobs $S \subseteq [N]$, let $f(S)$ be the cost of the set S (i.e., the capacity cost induced by the function f). We use two submodular functions f and f' (defined formally next) which are proved to be indistinguishable with a polynomial number of value oracle queries (see Lemma 5.1 of Svitkina and Fleischer 2011). Let denote $x = \ln(N)$. Note that Svitkina and Fleischer (2011) require x to be any parameter such that x^2 dominates $\ln(N)$ asymptotically and hence, this includes the special case we are considering here. Define $m_0 = \frac{5\sqrt{N}}{x}$, $\alpha_0 = \frac{N}{m_0}$, and $\beta_0 = \frac{x^2}{5}$. We choose N such that m_0 takes an integer value. Define $f(S)$ to be $\min\{|S|, \alpha_0\}$, and $f'(S)$ to be $\min\{\sum_i \min\{\beta_0, |S \cap V_i|\}, \alpha_0\}$ where $\{V_i\}_{i=1}^{m_0}$ is a random partitioning of $[N]$ into m_0 equal sized parts. Note that by definition, both set functions f and f' are monotone and submodular.

As mentioned, it is proved in Svitkina and Fleischer (2011) that the submodular functions f and f' cannot be distinguished from each other with a polynomial number of value oracle queries with high probability. We construct two instances of the bin packing problem with monotone submodular capacity constraints by using f and f' as follows. In both instances, the capacity of each machine is set to β_0.

In the first instance, a set S is feasible (i.e., we can schedule all its jobs in a machine) if and only if $f(S) \leq \beta_0$. By definition, $f(S)$ is greater than β_0 if $|S|$ is greater than β_0. Therefore, any feasible set S in this first instance consists of at most β_0 jobs. Consequently, in the first instance, any feasible assignment of jobs to machines requires at least $\frac{N}{\beta_0}$ machines. We define the second instance of the bin packing problem based on the submodular function f'. A set S is feasible in the second instance, if and only if $f'(S) \leq \beta_0$. Since $f'(V_j)$ is at most β_0 for each $1 \leq j \leq m_0$, each set V_j is a feasible set in this second instance. Therefore, we can assign each V_j to a separate machine to process all jobs, and consequently, m_0 machines suffice to do all the tasks in the second instance. We note that with our parameter setting, m_0 is much smaller that $\frac{N}{\beta_0}$. We then conclude that the optimum solutions of these two instances differ significantly.

We next prove the claim of Theorem 1 by using a contradiction argument. Assume that there exists a polynomial time algorithm ALG for the bin packing problem with monotone submodular capacity constraints with an approximation factor better than $\frac{\sqrt{N}}{\ln(N)}$. We next prove that by using ALG, we can distinguish between the two set functions f and f' with a polynomial number of value oracles, which contradicts the result of Svitkina and Fleischer (2011). The task of distinguishing between the functions f and f' can be formalized as follows. We have value oracle access to a set function g, and we know that g is either the same as f or the same as f'. The goal is to find out whether $g = f$ or $g = f'$ using a polynomial number of value oracle queries. We construct a bin packing instance with N jobs, capacity constraints g, and ask the algorithm ALG to solve this instance. If ALG uses less than $\frac{N}{\beta_0}$ machines to process all jobs, we can say that g is the same as f' (since with capacity constraints f, there does not exist a feasible assignment of all jobs to less than $\frac{N}{\beta_0}$ machines). On the other hand, if ALG uses at least $\frac{N}{\beta_0}$ machines, we can say that g is equal to f. This follows from the fact that if g was equal to f', the optimum number of machines would have been at most m_0. Since ALG has an approximation factor better than $\frac{\sqrt{N}}{\ln(N)}$, the number of machines used by ALG should have been less than $m_0 \times \frac{\sqrt{N}}{\ln(N)} = \frac{5\sqrt{N}}{x} \times \frac{\sqrt{N}}{\ln(N)} = \frac{N}{\beta_0}$. Therefore, using at least $\frac{N}{\beta_0}$ machines by ALG is a sufficient indicator of g being the same as f. This argument implies that an algorithm with an approximation factor better than $\frac{\sqrt{N}}{\ln(N)}$ for the bin packing problem with monotone submodular constraints yields a way of distinguishing between f and f' with a polynomial number of value oracle queries (since ALG is a polynomial time algorithm), which contradicts the result of Svitkina and Fleischer (2011). □

Appendix C: Details related to Observation 1

For ease of exposition, we first address the case with two job classes. Classes 1 and 2 have parameters (μ_1, b_1) and (μ_2, b_2) respectively. For example, an interesting special case is when one class of jobs is more predictable relative to the other (i.e., $\mu_1 = \mu_2 = \mu$, $b_2 = b$ and $b_1 = 0$). In practice, very often, one class of jobs has low variability (i.e., close to deterministic), whereas the other class is more volatile. For example, class 1 can represent loyal recurring customers, whereas class 2 corresponds to new customers.

We assume that we need to decide the number of machines to purchase, as well as how many jobs of types 1 and 2 to assign to each machine. Our goal is to find the right mix of jobs of classes 1 and 2 to assign to each machine (note that this proportion can be different for each machine). Consider a given machine i and denote by n_1 and n_2 the number of jobs of classes 1 and 2 that we assign to this machine. We would like to ensure that the chance constraint is satisfied in each machine with the given parameter α. Assuming that $V > n_1\mu_1 + n_2\mu_2$, we obtain:

$$\frac{[V - n_1\mu_1 - n_2\mu_2]^2}{n_1 b_1 + n_2 b_2} = D(\alpha)^2. \tag{8}$$

For a given α, one can find the value of n_1 as a function of n_2 that satisfies equation (8):

$$n_1(n_2) = \frac{V - n_2\mu_2}{\mu_1} + \frac{1}{2\mu_1^2}\left[b_1 D(\alpha)^2 - \sqrt{b_1^2 D(\alpha)^4 + 4b_1 D(\alpha)^2 (V - n_2\mu_2)\mu_1 + 4\mu_1^2 n_2 b_2 D(\alpha)^2}\right]. \qquad (9)$$

As we mentioned, an interesting special case is when both classes of jobs have the same expectation, i.e., $\mu_1 = \mu_2 = \mu$ but one type of jobs is much more predictable (i.e., smaller range or variance). In the extreme case, one can assume that class 1 jobs are deterministic, (i.e., $b_1 = 0$). In this case, equation (9) becomes:

$$n_1(n_2) = \frac{V - n_2\mu}{\mu} - \frac{\sqrt{n_2 b_2}}{\mu} D(\alpha). \qquad (10)$$

Alternatively, by directly looking at the modified capacity constraint (5) for this special case, we obtain:

$$V = (n_1 + n_2)\mu + D(\alpha)\sqrt{n_2 b_2}. \qquad (11)$$

Equation (11) can be interpreted as follows. Any additional job of type 1 takes μ from the capacity budget V, whereas any additional job of type 2 is more costly. The extra cost depends on both the uncertainty of the job (b_2) and the overcommitment policy ($D(\alpha)$). The higher one of these two factors is, the larger the capacity we should plan for jobs of type 2 (i.e., "safety buffer"). Note that the submodular nature of constraint (5) implies that this marginal extra cost decreases with the number of jobs n_2. In other words, when n_2 becomes large, each additional job of type 2 will converge to take a capacity of μ. More generally, by taking the derivative of the above expression, any additional job of type 2 will take $\mu + 0.5D(\alpha)\sqrt{b_2}/\sqrt{n_2}$ (where n_2 here represents how many jobs of type 2 are already assigned to this machine).

In Figure 5, we plot equation (10) for a specific instance with $V = 30$ and different values of α. As expected, if $n_2 = 0$, we can schedule $n_1 = 50$ jobs of class 1 to reach exactly the capacity V, no matter what is the value of α. On the other hand, for $\alpha = 0.99$, if $n_1 = 0$, we can schedule $n_2 = 38$ jobs of class 2. As the value of n_2 increases, the optimal value of n_1 decreases. For a given α, any point (n_1, n_2) on the curve (or below) guarantees the feasibility of the chance constraint. The interesting insight is to characterize the proportion of jobs of classes 1 and 2 per machine. For example, if we want to impose $n_1 = n_2$ in each machine, what is the optimal value for a given α? In our example, when $\alpha = 0.99$, we can schedule $n_1 = n_2 = 21$ jobs. If we compare to the case without overcommitment (i.e., $\alpha = 1$), we can schedule 18 jobs from each class. Therefore, we obtain an improvement of 16.67%. More generally, if the cost (or priority) of certain jobs is higher, we can design an optimal ratio per machine so that it still guarantees to satisfy the chance constraint. To summarize, for the case when $V = 30$, $\mu = 0.65$, $\overline{A} = 1$, $\underline{A} = 0.3$ and $\alpha = 0.99$, one can schedule either 50 jobs of class 1 or 38 jobs of class 2 or any combination of both classes according to equation (10).

We next consider solving the offline problem when our goal is to schedule N_1 jobs of class 1 and N_2 jobs of class 2. The numbers N_1 and N_2 are given as an input, and our goal is to minimize the number of machines denoted by M^*, such that each machine is assigned a pair (n_1, n_2) that satisfies equation (9). Since n_1 and n_2 should be integer numbers, one can compute for a given value of α, all the feasible pairs that lie on the curve

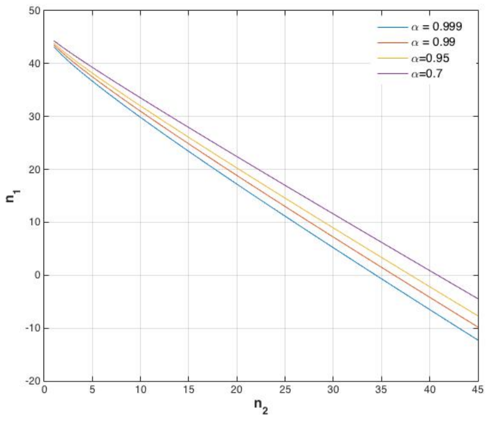


Figure 5 **Parameters:** $\overline{A}=1$, $\underline{A}=0.3$, $\mu=0.65$, $V=30$.

or just below (we have at most $K=\min_{i=1,2}\max n_i$ such pairs). In other words, for each $k=1,2,\ldots,K$, we compute a pair of coefficients denoted by (β_k,γ_k). The optimization becomes a cutting stock problem:

$$
\begin{aligned}
M^* = \min_{z_k} \quad & \sum_{k=1}^{K} z_k \\
\text{s.t.} \quad & \sum_{k=1}^{K} \beta_k z_k \geq N_1 \\
& \sum_{k=1}^{K} \gamma_k z_k \geq N_2 \\
& z_k \geq 0, \text{integer} \qquad \forall k.
\end{aligned}
\tag{12}
$$

The decision variable z_k represents the number of times we use the pair (β_k,γ_k), and M^* denotes the optimal number of machines. As a result, assuming that we only have two classes of jobs (with different μs and bs), one can solve the deterministic linear integer program in (12) and obtain a solution for problem (SMBP). Note that the above treatment can easily be extended to more than two classes of jobs. However, this becomes computationally prohibitive, as the number of variables is exponential in the number of classes.

Appendix D: Proof of Theorem 2

Proof. Let m be the number of machines that ALG purchases when serving jobs $\{1,2,\cdots,N\}$. For any machine $1\leq i\leq m$, we define S_i to be the set of jobs assigned to machine i. Without loss of generality, we assume that the m machines are purchased in the order of their indices. In other words, machines 1 and m are the first and last purchased ones respectively. For any pair of machines $1\leq i<i'\leq m$, we next prove that the set $S_i\cup S_{i'}$ is infeasible for the modified capacity constraint, i.e., $\sum_{j\in S_i\cup S_{i'}}\mu_j+\sqrt{\sum_{j\in S_i\cup S_{i'}}b_j}>1$. Let j be the first job assigned to machine i'. Since ALG is lazy, assigning j to machine i upon its arrival time was not feasible, i.e., the set $\{j\}\cup S_i$ is infeasible. Since we only assign more jobs to machines throughout the course of the algorithm, and do not remove any job, the set $S_{i'}\cup S_i$ is also infeasible.

In the next Lemma, we lower bound the sum of μ_j+b_j for the jobs in an infeasible set.

Lemma 1. *For any infeasible set T, we have $\sum_{j\in T}(\mu_j+b_j)>\frac{3}{4}$.*

Proof. For any infeasible set T, we have by definition $\sum_{j\in T}\mu_j + \sqrt{\sum_{j\in T} b_j} > 1$. We denote $x = \sum_{j\in T}\mu_j$, and $y = \sqrt{\sum_{j\in T} b_j}$. Then, $y > 1-x$. If x is greater than 1, the claim of the lemma holds trivially. Otherwise, we obtain: $x + y^2 > x + (1-x)^2 = x^2 - x + 1 = (x-\frac{1}{2})^2 + \frac{3}{4} \geq \frac{3}{4}$. We conclude that $\sum_{j\in T}(\mu_j + b_j) > \frac{3}{4}$. □

As discussed, for any pair of machines $i < i'$, the union of their sets of jobs $S_i \cup S_{i'}$ is an infeasible set that does not fit in one machine. We now apply the lower bound from Lemma 1 to the infeasible set $S_i \cup S_{i'}$, i.e., $\sum_{j\in S_i\cup S_{i'}}(\mu_j + b_j) > \frac{3}{4}$. We can sum up this inequality for all $\binom{m}{2}$ pairs of machines i and i' to obtain:

$$\sum_{1\leq i<i'\leq m}\ \sum_{j\in S_i\cup S_{i'}} (\mu_j + b_j) > \frac{3}{4}\binom{m}{2}. \tag{13}$$

We claim that the left hand side of (13) is equal to $(m-1)\sum_{j=1}^{N}(\mu_j + b_j)$. We note that for each job $j\in S_k$, the term $\mu_j + b_j$ appears $k-1$ times in the left hand side of (13) when i' is equal to k. In addition, this term also appears $m-k$ times when i is equal to k. Therefore, every $\mu_j + b_j$ appears $k-1+m-k = m-1$ times, which is independent of the index k of the machine that contains job j. As a result, we obtain:

$$\sum_{j=1}^{N}(\mu_j + b_j) > \frac{3}{4(m-1)}\binom{m}{2} = \frac{3m}{8}.$$

On the other hand, we use the optimal assignment to upper bound the sum $\sum_{j=1}^{N}(\mu_j + b_j)$, and relate m to $\mathbb{O}PT$. Let $T_1, T_2, \cdots, T_{\mathbb{O}PT}$ be the optimal assignment of all jobs to $\mathbb{O}PT$ machines. Since T_i is a feasible set, we have $\sum_{j\in T_i}\mu_j + \sqrt{\sum_{j\in T_i} b_j} \leq 1$, and consequently, we also have $\sum_{j\in T_i}(\mu_j + b_j) \leq 1$. Summing up for all the machines $1 \leq i \leq \mathbb{O}PT$, we obtain: $\sum_{j=1}^{N}(\mu_j + b_j) = \sum_{i=1}^{\mathbb{O}PT}\sum_{j\in T_i}(\mu_j + b_j) \leq \mathbb{O}PT$. We conclude that: $\mathbb{O}PT \geq \sum_{j=1}^{N}(\mu_j + b_j) > \frac{3m}{8}$. This completes the proof of $m < \frac{8\mathbb{O}PT}{3}$. □

Appendix E: Proof of Theorem 4

Proof. Let n_1 (resp. n_2) be the number of machines purchased by FIRST-FIT with only a single job (resp. at least two jobs), and S_1 (resp. S_2) be the set of n_1 (resp. n_2) jobs assigned to these machines. Our goal is to prove that $n_1 + n_2 \leq \frac{9}{4}\mathbb{O}PT + 1$. We know that any pair of jobs among the n_1 jobs in S_1 does not fit in a single machine (by the definition of FIRST-FIT). Therefore, any feasible allocation (including the optimal allocation) needs at least n_1 machines. In other words, we have $\mathbb{O}PT \geq n_1$. This observation also implies that the sum of $\mu_j + b_j$ for any pair of jobs in S_1 is greater than $\frac{3}{4}$ (using Lemma 1). If we sum up all these inequalities for the different pairs of jobs in S_1, we have: $(n_1 - 1)\sum_{j\in S_1}(\mu_j + b_j) > \binom{n_1}{2}\frac{3}{4}$. We note that the $n_1 - 1$ term on the left side appears because every job $j \in S_1$ is paired with $n_1 - 1$ other jobs in S_1, and the $\binom{n_1}{2}$ term on the right side represents the total number of pairs of jobs in S_1. By dividing both sides of this inequality by $n_1 - 1$, we obtain $\sum_{j\in S_1}(\mu_j + b_j) > \frac{3n_1}{8}$.

We also lower bound $\sum_{j\in S_2}(\mu_j + b_j)$ as a function of n_2 as follows. Let $m_1 < m_2 < \cdots < m_{n_2}$ be the machines that have at least two jobs, and the ordering shows in which order they were purchased (e.g., m_1 was purchased first). Define M_i to be the set of jobs in machine m_i. By the definition of FIRST-FIT, any job j in machine m_{i+1} could not be assigned to machine m_i because of the feasibility constraints for any $1 \leq i < n_2$. In other words, the set of jobs M_i with any job $j \in M_{i+1}$ form an infeasible set. Therefore, we have $\mu_j + b_j + \sum_{j'\in M_i}(\mu_{j'} + b_{j'}) > \frac{3}{4}$. For each $1 \leq i < n_2$, one can pick two distinct jobs j_1 and j_2 from M_{i+1}, and write the following two inequalities:

$$\mu_{j_1} + b_{j_1} + \sum_{j'\in M_i}(\mu_{j'} + b_{j'}) > \frac{3}{4} \qquad \text{and} \qquad \mu_{j_2} + b_{j_2} + \sum_{j'\in M_i}(\mu_{j'} + b_{j'}) > \frac{3}{4}.$$

Summing up these two inequalities implies that: $\mu_{j_1} + b_{j_1} + \mu_{j_2} + b_{j_2} + 2\sum_{j' \in M_i}(\mu_{j'} + b_{j'}) > \frac{3}{2}$. Since j_1 and j_2 are two distinct jobs in M_{i+1}, we have:

$$\sum_{j \in M_{i+1}} (\mu_j + b_j) + 2 \sum_{j' \in M_i} (\mu_{j'} + b_{j'}) > \frac{3}{2}.$$

Next, we sum up this inequality for different values of $i \in \{1, 2, \cdots, n_2 - 1\}$ to achieve that:

$$2 \sum_{j \in M_1} (\mu_j + b_j) + 3 \sum_{i=2}^{n_2 - 1} \sum_{j \in M_i} (\mu_j + b_j) + \sum_{j \in M_{n_2}} (\mu_j + b_j) > \frac{3}{2} \times (n_2 - 1),$$

and consequently, we have $3\sum_{i=1}^{n_2} \sum_{j \in M_i}(\mu_j + b_j) > \frac{3}{2} \times (n_2 - 1)$. This is equivalent to $\sum_{j \in S_2}(\mu_j + b_j) > \frac{1}{2} \times (n_2 - 1)$. Combining both inequalities, we obtain: $\sum_{j \in S_1 \cup S_2}(\mu_j + b_j) > \frac{3}{8}n_1 + \frac{1}{2}(n_2 - 1)$. On the other hand, $\mathbb{O}PT$ is at least $\sum_{j \in S_1 \cup S_2}(\mu_j + b_j)$. We then have the following two inequalities:

$$\mathbb{O}PT \geq n_1, \tag{14}$$

$$\mathbb{O}PT > \frac{3}{8}n_1 + \frac{1}{2}(n_2 - 1). \tag{15}$$

We can now multiply (14) by $\frac{1}{4}$ and (15) by 2, and sum them up. We conclude that $\frac{9}{4}\mathbb{O}PT + 1$ is greater than $n_1 + n_2$, which is the number of machines purchased by FIRST-FIT. □

Appendix F: Proof of Theorem 5

We first state the following Lemma that provides a lower bound on $\mathbb{O}PT$. For any $a, b \geq 0$, we define the function $f(a, b) = \frac{2a + b + \sqrt{b(4a + b)}}{2}$.

LEMMA 2. *For any feasible set of jobs S, the sum $\sum_{j \in S} f(\mu_j, b_j)$ is at most* 1.

Proof. Define $\bar{\mu} = \frac{\sum_{j \in S} \mu_j}{|S|}$ and $\bar{b} = \frac{\sum_{j \in S} b_j}{|S|}$. Since the function f is concave with respect to both a and b, using Jensen's inequality we have $\sum_{j \in S} f(\mu_j, b_j) \leq |S| f(\bar{\mu}, \bar{b})$. Since S is a feasible set, $\mathbb{C}ost(S) = |S|\bar{\mu} + \sqrt{|S|\bar{b}} \leq 1$. The latter is a quadratic inequality with variable $x = \sqrt{|S|}$, so that we we can derive an upper bound on $|S|$ in terms of $\bar{\mu}$ and $\bar{b}$. Solving the quadratic form $\bar{\mu}X + \sqrt{\bar{b}X} = 1$ yields:

$$X = \frac{-\sqrt{\bar{b}} \pm \sqrt{\bar{b} + 4\bar{\mu}}}{2\bar{\mu}}.$$

We then have $\sqrt{|S|} \leq \frac{\sqrt{4\bar{\mu} + \bar{b}} - \sqrt{\bar{b}}}{2\bar{\mu}} = \frac{2}{\sqrt{(4\bar{\mu} + \bar{b})} + \sqrt{\bar{b}}}$. Therefore, $|S| \leq \frac{4}{4\bar{\mu} + \bar{b} + \bar{b} + 2\sqrt{\bar{b}(4\bar{\mu} + \bar{b})}} = \frac{1}{f(\bar{\mu}, \bar{b})}$, where the equality follows by the definition of f. Equivalently, $|S| f(\bar{\mu}, \bar{b}) \leq 1$, and hence $\sum_{j \in S} f(\mu_j, b_j) \leq 1$. □

Proof of Theorem 5. For any arbitrary allocation of jobs, applying Lemma 2 to all the machines implies that the number of machines is at least $\sum_{j=1}^{N} f(\mu_j, b_j)$. So it suffices to upper bound the number of machines as a function of $\sum_{j=1}^{N} f(\mu_j, b_j)$. For any given machine, we prove that the sum of $f(\mu_j, b_j)$ for all jobs j assigned to this machine is at least $1 - O(\epsilon + \delta)$. Consider the set of jobs S assigned to a given machine. Similar to the proof of Lemma 2, we define $\bar{\mu} = \frac{\sum_{j \in S} \mu_j}{|S|}$ and $\bar{b} = \frac{\sum_{j \in S} b_j}{|S|}$. Define $r = \frac{\sum_{j \in S} b_j}{\sum_{j \in S} \mu_j} = \frac{\bar{b}}{\bar{\mu}}$. We start by lower bounding $f(\mu_j, b_j)$ as a function of $f(\mu_j, r\mu_j)$. Recall that each machine is δ-homogeneous, i.e., for all pairs of jobs in the same machine, the ratios $\frac{b_j}{\mu_j}$ are at most a multiplicative factor of $1 + \delta$ away from each other. Hence, any ratio $\frac{b_j}{\mu_j}$ is at least $\frac{r}{(1+\delta)}$. Consequently, we have $b_j \geq \frac{r\mu_j}{1+\delta}$ which implies that:

$$f(\mu_j, b_j) \geq f\big(\frac{\mu_j}{1 + \delta}, b_j\big) \geq f\big(\frac{\mu_j}{1 + \delta}, \frac{r\mu_j}{1 + \delta}\big) = \frac{1}{1 + \delta} f(\mu_j, r\mu_j) \geq (1 - \delta) f(\mu_j, r\mu_j).$$

The first and second inequalities follow from the monotonicity of the function f, the equality follows from the definition of f, and the last inequality holds since $\delta \geq 0$. It now suffices to lower bound $\sum_{j\in S} f(\mu_j, r\mu_j)$ so as to obtain a lower bound for $\sum_{j\in S} f(\mu_j, b_j)$. We note that for any $\eta \geq 0$, we have $f(\eta\mu, \eta b) = \eta f(\mu, b)$ by the definition of f. Applying $\eta = \mu_j$, $\mu = 1$ and $b = r$, we obtain $f(\mu_j, r\mu_j) = \mu_j f(1, r)$ which implies:

$$\sum_{j\in S} f(\mu_j, r\mu_j) = \sum_{j\in S} \mu_j f(1,r) = f(A,B),$$

where $A = \sum_{j\in S} \mu_j$, and $B = r\sum_{j\in S} \mu_j = \sum_{j\in S} b_j$. The last equality above follows from $f(\eta\mu, \eta b) = \eta f(\mu, b)$ with $\eta = A$. Recall that each machine is ϵ-full, i.e., $\mathbb{C}ost(S) \geq 1-\epsilon$ or equivalently, $A + \sqrt{B} \geq 1 - \epsilon$. Let $A' = \frac{A}{(1-\epsilon)^2}$ and $B' = \frac{B}{(1-\epsilon)^2}$. Then, we have:

$$A' + \sqrt{B'} = \frac{A}{(1-\epsilon)^2} + \frac{\sqrt{B}}{1-\epsilon} \geq \frac{A+\sqrt{B}}{1-\epsilon} \geq 1.$$

Since $B = rA$, we also have $B' = rA'$, and the lower bound can be rewritten as follows: $A' + \sqrt{rA'} \geq 1$, which is the same quadratic form as in the proof of Lemma 2. Similarly, we prove that $A' \geq \frac{4}{4+2r+2\sqrt{r(4+r)}}$ which is equal to $\frac{1}{f(1,r)}$. We also know that $f(A', B') = f(A', rA') = A' f(1,r)$. We already proved that $A \geq \frac{1}{f(1,r)}$, so we have $f(A', B') \geq 1$. By definition of A' and B', we know that$f(A,B) = (1-\epsilon)^2 f(A', B') \geq (1-\epsilon)^2$. We conclude that the sum $\sum_{j\in S} f(\mu_j, b_j) \geq (1-\delta) f(A,B) \geq (1-\delta)(1-\epsilon)^2$. As a result, for each machine the sum of $f(\mu_j, b_j)$ is at least $(1-\delta)(1-\epsilon)^2$. Let m be the number of purchased machines. Therefore, the sum $\sum_{j=1}^{N} f(\mu_j, b_j)$ is lower bounded by $(1-\delta)(1-\epsilon)^2 m$, and at the same time upper bounded by $\mathbb{O}PT$. Consequently, m does not exceed $\frac{\mathbb{O}PT}{(1-\delta)(1-\epsilon)^2}$ and this concludes the proof. □

Appendix G: Proof of Theorem 6

Proof. We first prove that the algorithm terminates in a finite number of iterations. Note that all the update operations (except the first one) reduce the number of purchased machines, and hence, there are no more than N of those. As a result, it suffices to upper bound the number of times we perform the first update operation. Since we assign jobs to lower id machines, there cannot be more than N^2 consecutive first update operations. Consequently, after at most $N \times N^2 = N^3$ operations, the algorithm has to terminate.

Next, we derive an upper bound on the number of purchased machines at the end of the algorithm. Note that all the machines belong to one of the following four categories: (1) Single job machines, i.e., the set A_1. (2) Medium machines with only one non-good job – denoted by the set B. (3) Medium machines with at least two non-good jobs – denoted by the set C. (4) Large machines, i.e., the set A_5.

Let a, b, c, and d be the number of machines in A_1, B, C, and A_5 respectively. Since no update operation is possible (as the algorithm already terminated), the b non-good jobs assigned to the machines in the set B do not fit in any of the single job machines, and no pair of them fit together in a new machine. Consider these b non-good jobs in addition to the a jobs in the machines of the set A_1. No pair of these $a+b$ jobs fit in one machine together and therefore, $\mathbb{O}PT \geq a + b$.

We also know that $\mathbb{O}PT \geq \sum_{j=1}^{N}(\mu_j + b_j)$. Next, we derive a more elaborate lower bound on $\mathbb{O}PT$ by writing the sum of $\mu_j + b_j$ as a linear combination of the sizes of the sets A_1, B, C, and A_5. For each machine i, let M_i be the set of jobs assigned to this machine. Let $i_1 < i_2 < \cdots < i_d$ be the indices of machines in the set A_5, where $d = |A_5|$. Since we cannot perform the first update operation anymore, we can say that no job in machine $i_{\ell+1}$ fits in machine i_ℓ for any $1 \le \ell < d$. Therefore, $\mu_j + b_j + \sum_{j' \in M_{i_\ell}} (\mu_{j'} + b_{j'}) > \frac{3}{4}$ for any $j \in M_{i_{\ell+1}}$ (using Lemma 1). We write this inequality for 5 different jobs (arbitrarily chosen) in $M_{i_{\ell+1}}$ (recall that there are at least 5 jobs in this machine), and for all the values of $1 \le \ell < d$. If we sum up all these $5(d-1)$ inequalities, then the right hand side would be $5(d-1) \times \frac{3}{4}$. On the other hand, the term $\mu_j + b_j$ for every job in these d machines appears on the left hand side at most $5+1=6$ times. Therefore, by summing up these inequalities, we obtain: $\sum_{i \in A_5} \sum_{j \in M_i} (\mu_j + b_j) > \frac{3}{4} \times \frac{5(d-1)}{6} = \frac{5(d-1)}{8}$.

Each machine in A_5 has at least 5 jobs. Therefore, the term $\frac{5}{6}$ appears in the lower bound. With a similar argument, each machine in either B or C has at least 2 jobs and hence, this term is now replaced by $\frac{2}{3}$. The inequalities for every pair of machines in B and C are then: $\sum_{i \in B} \sum_{j \in M_i} (\mu_j + b_j) > \frac{3}{4} \times \frac{2(b-1)}{3} = \frac{b-1}{2}$, and $\sum_{i \in C} \sum_{j \in M_i} (\mu_j + b_j) > \frac{3}{4} \times \frac{2(c-1)}{3} = \frac{c-1}{2}$.

Next, we lower bound $\sum_{i \in A_1 \cup C} \sum_{j \in M_i} (\mu_j + b_j)$ as a function of a and c in order to complete the proof. Recall that each machine in C has at least two non-good jobs. If we pick one of these non-good jobs j, and a random machine i from A_1, with probability at least $\frac{a-5}{a}$, job j does not fit in machine i. This follows from the fact that a non-good job fits in at most 5 machines in A_1 and hence, a random machine in A_1 would not be be able to fit job j with probability at least $\frac{a-5}{a}$. Therefore, $\mu_j + b_j + \sum_{j' \in M_i} (\mu_{j'} + b_{j'}) \ge \frac{3}{4}$ with probability at least $\frac{a-5}{a}$. We next consider two different cases depending on the value of c.

If c is at least $\frac{a}{2}$, we pick a random machine in C, and two of its non-good jobs j_1 and j_2 arbitrarily. We also pick a random machine in A_1. For each of these two jobs, the sum $\mu_j + b_j$ of the non-good job and the single job in the selected machine in A_1 is greater than $\frac{3}{4}$ with probability at least $\frac{a-5}{a}$. Summing up these two inequalities, we obtain:

$$\frac{2}{a}\Big[\sum_{i \in A_1} \sum_{j \in M_i} (\mu_j + b_j)\Big] + \frac{1}{c}\Big[\sum_{i \in C} \sum_{j \in M_i} (\mu_j + b_j)\Big] > \frac{a-5}{a} \times \frac{3}{2}. \tag{16}$$

The left hand side of the above equation is composed of two terms. The first term is obtained through picking a random machine in A_1 (i.e., with probability $\frac{1}{a}$), and once the machine is picked, we sum up both equations so we obtain $\frac{2}{a}$. For the second term, every machine in the set C is chosen with probability $\frac{1}{c}$. When the machine is selected, we sum up on all the jobs and hence get an upper bound. As we have shown, $\sum_{i \in C} \sum_{j \in M_i} (\mu_j + b_j) \ge \frac{c-1}{2}$. Combining these two inequalities leads to (using $c \ge \frac{a}{2}$):

$$\sum_{i \in A_1 \cup C} \sum_{j \in M_i} (\mu_j + b_j) > \frac{a}{2} \times \frac{3(a-5)}{2a} + (1 - \frac{a}{2c}) \times \frac{c-1}{2} \ge \frac{3a}{4} - \frac{15}{4} + \frac{c}{2} - \frac{a}{4} - \frac{1}{2} = \frac{a+c}{2} - 4.25.$$

By combining the three different bounds (on $A_1 \cup C$, B and A_5), we obtain $\sum_{j=1}^{N} (\mu_j + b_j) \ge \frac{a+b+c}{2} + \frac{5d}{8} - 5.875$. Since $\mathbb{O}PT \ge \sum_{j=1}^{N} (\mu_j + b_j)$, the number of purchased machines $a+b+c+d$ is no more than $2\mathbb{O}PT + 11$.

In the other case, we have $c < \frac{a}{2}$. Note that inequality (16) still holds. However, since the coefficient $1 - \frac{a}{2c}$ becomes negative, we cannot combine the two inequalities as before. Instead, we lower bound the sum $\mu_j + b_j$ of jobs in A_1. We know that there is no pair of a jobs in the A_1 machines that fit together in one machine. Therefore, $\sum_{i \in A_1} \sum_{j \in M_i} (\mu_j + b_j) \ge \frac{3a}{8}$. Next, we multiply inequality (16) by c, and combine it with this new lower bound on $\sum_{i \in A_1} \sum_{j \in M_i} (\mu_j + b_j)$, to obtain (using $c < \frac{a}{2}$):

$$\sum_{i \in A_1 \cup C} \sum_{j \in M_i} (\mu_j + b_j) > c \times \frac{3(a-5)}{2a} + (1 - \frac{2c}{a}) \times \frac{3a}{8} > \frac{3c}{2} - \frac{5}{4} + \frac{3a}{8} - \frac{3c}{4} = \frac{3a}{8} + \frac{3c}{4} - \frac{5}{4}.$$

Combining this inequality with similar ones on the sets B and A_5, we obtain $\mathbb{O}PT \geq \sum_{j=1}^{N}(\mu_j + b_j) > \frac{3a}{8} + \frac{b}{2} + \frac{3c}{4} + \frac{5d}{8} - \frac{19}{8}$. Finally, combining this with $\mathbb{O}PT \geq a+b$ leads to $a+b+c+d \leq \frac{8}{5}\mathbb{O}PT + \frac{2}{5}\mathbb{O}PT + \frac{19}{5} = 2\mathbb{O}PT + 3.75$, which concludes the proof. □

Appendix H: Alternative constraints

H.1. Alternative constraints

Consider the (SMBP) problem, with the following family of constraints, parametrized by $0.5 \leq p \leq 1$:

$$\sum_{j=1}^{N} \mu_j x_{ij} + D(\alpha)\Big(\sum_{j=1}^{N} b_j x_{ij}\Big)^p \leq V y_i, \tag{17}$$

Note that this equation is still monotone and submodular in the assignment vector $\mathbf{x}$, and captures some notion of risk pooling. In particular, the "safety buffer" reduces with the number of jobs already assigned to each machine. The motivation behind such a modified capacity constraint lies in the shape that one wishes to impose on the term that captures the uncertain part of the job. In one extreme ($p = 1$), we consider that the term that captures the uncertainty is linear and hence, as important as the expectation term. In the other extreme case ($p = 0.5$), we consider that the term that captures the uncertainty behaves as a square root term. For a large number of jobs per machine, this is known to be an efficient way of handling uncertainty. Note also that when $p = 0.5$, we are back to equation (5), and when $p = 1$ we have a commonly used benchmark (see more details in Section 7). One can extend our analysis and derive an approximation factor for the online problem as a function of p for any lazy algorithm.

Corollary 1. *Consider the bin packing problem with the modified capacity constraint* (17). *Then, any lazy algorithm* ALG *purchases at most* $\frac{2}{f(p)}\mathbb{O}PT$ *machines, where* $\mathbb{O}PT$ *is the optimum number of machines to serve all jobs and* $f(p)$ *is given by:*

$$f(p) = 1 - (1-p)p^{\frac{1}{p}-1}.$$

The proof is in a very similar spirit as in Theorem 2 and is not repeated for conciseness. Intuitively, we find parametric lower and upper bounds on $\big(\sum_{j=1}^{N} b_j x_{ij}\big)^p$ in terms of $\sum_{j=1}^{N} b_j x_{ij}$. Note that when $p = 0.5$, we recover the result of Theorem 2 (i.e., a 8/3 approximation) and as p increases, the approximation factor converges to 2. In Figure 6, we plot the approximation factor as a function of $0.5 \leq p \leq 1$.

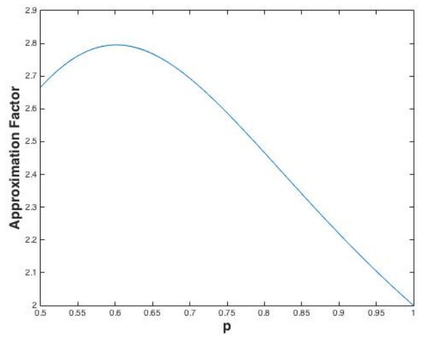
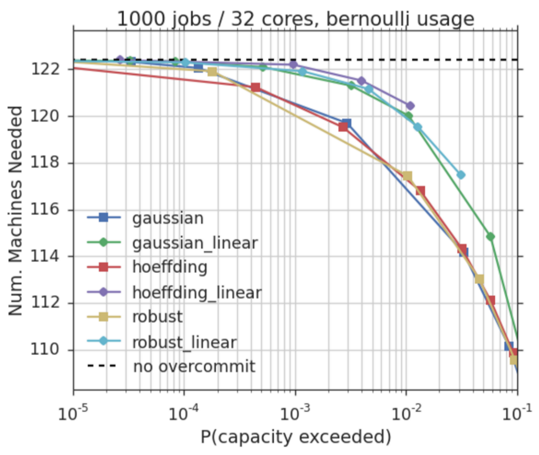
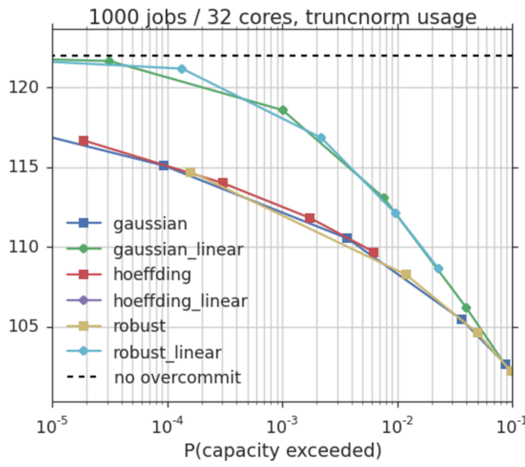


Figure 6 Approximation factor $\frac{2}{f(p)}$ as a function of p.

Appendix I: Additional Details Related to Section 7



(a) Bernoulli usage



(b) Truncated Gaussian usage

Figure 7 Results for 32 core machines.