Strategyproof Scheduling with Predictions

Eric Balkanski
Columbia University, New York, NY, USA

Vasilis Gkatzelis
Drexel University, Philadelphia, PA, USA

Xizhi Tan
Drexel University, Philadelphia, PA, USA

Abstract

In their seminal paper that initiated the field of algorithmic mechanism design, Nisan and Ronen [27] studied the problem of designing strategyproof mechanisms for scheduling jobs on unrelated machines aiming to minimize the makespan. They provided a strategyproof mechanism that achieves an n-approximation and they made the bold conjecture that this is the best approximation achievable by any deterministic strategyproof scheduling mechanism. After more than two decades and several efforts, n remains the best known approximation and very recent work by Christodoulou et al. [11] has been able to prove an $\Omega(\sqrt{n})$ approximation lower bound for all deterministic strategyproof mechanisms. This strong negative result, however, heavily depends on the fact that the performance of these mechanisms is evaluated using worst-case analysis. To overcome such overly pessimistic, and often uninformative, worst-case bounds, a surge of recent work has focused on the "learning-augmented framework", whose goal is to leverage machine-learned predictions to obtain improved approximations when these predictions are accurate (consistency), while also achieving near-optimal worst-case approximations even when the predictions are arbitrarily wrong (robustness).

In this work, we study the classic strategic scheduling problem of Nisan and Ronen [27] using the learning-augmented framework and give a deterministic polynomial-time strategyproof mechanism that is 6-consistent and $2n$-robust. We thus achieve the "best of both worlds": an $O(1)$ consistency and an $O(n)$ robustness that asymptotically matches the best-known approximation. We then extend this result to provide more general worst-case approximation guarantees as a function of the prediction error. Finally, we complement our positive results by showing that any 1-consistent deterministic strategyproof mechanism has unbounded robustness.

2012 ACM Subject Classification Theory of computation → Algorithmic mechanism design

Keywords and phrases Mechanism Design with Predictions, Strategyproof Scheduling

Digital Object Identifier 10.4230/LIPIcs.ITCS.2023.11

Related Version *Full Version*: https://arxiv.org/abs/2209.04058

Funding All three authors were supported by NSF grant CCF-2210502
Eric Balkanski: Columbia Center of AI Technology (CAIT), in collaboration with Amazon, faculty research award, and NSF grant IIS-2147361
Vasilis Gkatzelis: NSF CCF-2047907, and NSF CCF-2008280
Xizhi Tan: NSF CCF-2047907

1 Introduction

In their seminal paper which initiated the field of algorithmic mechanism design, Nisan and Ronen [27] focused on a natural job scheduling problem involving strategic agents: a set of m jobs needs to be scheduled on a set of n machines aiming to minimize the makespan, and each machine is owned by an agent who requires a monetary compensation in exchange for processing the jobs assigned to them. What makes this problem particularly demanding is that the compensation each agent receives needs to be at least as high as the cost that they

suffer for processing the jobs assigned to them (i.e., the required processing time), yet the actual processing times are private information that only that agent knows. Therefore, an agent can choose to misreport (either overstate or understate) their processing times if doing so would change the outcome (i.e., their assignment and their compensation) to one that they prefer. To avoid this issue, the goal is to design *strategyproof* mechanisms that carefully choose the outcome to ensure that no agent has an incentive to misreport their costs, while also ensuring that the makespan of the chosen assignment is as small as possible.

Nisan and Ronen [27] proposed a strategyproof mechanism for this problem and showed that the allocation generated by this mechanism is an n-approximation of the optimal makespan. Even though this guarantee is much less appealing than the 2-approximation that one can achieve in polynomial time if the costs are publicly known [21], they went on to make the bold conjecture that this is the best possible worst-case approximation that any deterministic strategyproof mechanism can achieve (polynomial time or not). After more than two decades, and despite several efforts to tackle this, now classic, problem, no strategyproof mechanism with an approximation better than $O(n)$ is known (not even randomized). Instead, after a sequence of inapproximability results, proving gradually increasing constant approximation lower bounds for deterministic mechanisms (e.g., [27, 13, 19, 16, 14]), a very recent breakthrough by Christodoulou et al. [11] made major progress by proving a lower bound of $1 + \sqrt{n-1}$ for deterministic mechanisms. This result paints a very pessimistic picture, implying that even if a better strategyproof mechanism exists, its worst-case approximation will not be practical. However, this heavily depends on the, rather unrealistic assumption, that the mechanism has no information regarding the costs of the agents.

Facing analogous worst-case impossibility results that are overly pessimistic and rather uninformative, a recent surge of work has focused on the "learning-augmented framework", aiming to achieve more refined and informative bounds, while retaining the benefits of worst-case analysis (see [26] for a survey). This framework assumes that the designer is provided with some machine-learned predictions that can be used to design more practical algorithms that achieve near optimal performance when the predictions are correct (this is called the *consistency* guarantee). However, rather than assuming that the predictions are always accurate and forfeiting the benefits of worst-case analysis altogether, this framework seeks to also provide strong guarantees, even if the predictions are arbitrarily inaccurate (this is called the *robustness* guarantee). More formally, the consistency of an algorithm is the worst-case approximation guarantee that it achieves assuming that the prediction it was provided with is accurate, while its robustness is the worst-case approximation guarantee without any assumptions regarding the prediction accuracy (see Section 2 for more details). The quality of an algorithm is then evaluated based on both its consistency and its robustness. This framework was initially restricted to algorithms, but it was very recently extended to settings involving strategic agents [1, 17, 32], motivating the design of new mechanisms leveraging predictions to overcome the incentive issues that arise. In this paper, we revisit the classic scheduling problem of Nisan and Ronen [27] using the learning-augmented framework and design mechanisms enhanced with predictions regarding the machines' processing times.

Given predictions regarding the machines' processing times, a naive solution would be to assume the predictions are accurate and output a schedule with small makespan according to the predicted processing times alone (i.e., without eliciting the agents' true processing times). This is a strategyproof mechanism that would achieve a consistency of $O(1)$, but its robustness would be unbounded (e.g., even if a single job's processing time is mispredicted, it could end up being assigned to a machine where its actual processing time is arbitrarily high).

On the other hand, the mechanism of Nisan and Ronen [27] would achieve a robustness of $O(n)$, but its consistency would also be no better than $O(n)$, since its output disregards the predictions. In an attempt to provide a middle ground between these two extremes, very recent work by Xu and Lu [32] proposed a strategyproof mechanism that can guarantee a consistency of $O(1)$ with bounded robustness, but the robustness guarantee of $O(n^3)$ that it achieves is much worse than the $O(n)$ robustness achieved in [27]. Our main result in this paper is a strategyproof that combines the "best of both worlds": a consistency of $O(1)$ with the best-known robustness of $O(n)$.

1.1 Our results

In this paper we leverage the learning-augmented framework to develop a deeper understanding of this classic strategic scheduling problem, and design much more practical mechanisms enhanced with predictions. In particular, we assume that the mechanism is provided with predictions $\hat{p}(i,j)$ regarding the amount of time that each machine i would require in order to fully process each job j; crucially, these predictions may be highly inaccurate.

Our main result is a deterministic polynomial-time strategyproof mechanism that guarantees a consistency of $(4+2\gamma)$ and a robustness of $(1+\frac{1}{\gamma})n$ for any choice of $\gamma \in \left(0, \frac{n}{2}-1\right)$, which is a parameter that the designer can determine to receive their desired trade-off between consistency and robustness. For example, setting $\gamma = 1$ yields a mechanism with a small constant consistency of 6 (i.e., a 6-approximation of the optimal makespan when the predictions are accurate) while maintaining a robustness of $2n$ (i.e., a worst-case approximation guarantee that asymptotically matches the best-known approximation, irrespective of how inaccurate the predictions may be).

Our main mechanism, ScaledGreedy, uses the predicted processing times to pre-compute a schedule that (approximately)[1] minimizes the makespan, assuming these predictions are correct. The mechanism then uses this schedule as a guide and adds a "bias" in favor of scheduling jobs on the machines that they were assigned to in this schedule. After introducing this bias, the mechanism follows a simple greedy procedure, so the main technical novelty is the subtle way in which this bias towards the predictions is introduced, while ensuring that we asymptotically match the best known robustness when the predictions are arbitrarily inaccurate. Roughly speaking, the mechanism carefully chooses a subset of jobs and proceeds to assign each of them to the same machine as in the pre-computed schedule, unless the processing times reported by the machines for that job differ significantly from the predicted ones. Before presenting this mechanism and its analysis in Section 4, we first dedicate Section 3 to a simpler mechanism, SimpleScaledGreedy, which provides a warm-up toward our main result while combining a consistency of $O(1)$ with a robustness of $O(n^2)$.

In Section 5, we take one step further and propose a mechanism that provides worst-case approximation guarantees as a function of the prediction error. In other words, rather than focusing only on the two extremes of consistency (where the error is zero) and robustness (where the error is unbounded), this mechanism guarantees a good approximation as long as the prediction error is not too high. Given a prediction $\hat{\mathbf{p}}$ of the actual processing times $\mathbf{p}$, we let the prediction error η be the largest ratio between the predicted processing time and actual processing time for any machine i, job j pair, i.e., $\eta = \max_{i,j} \max\left\{\frac{\hat{p}(i,j)}{p(i,j)}, \frac{p(i,j)}{\hat{p}(i,j)}\right\}$. Our mechanism takes as input an error tolerance parameter $\bar{\eta} > 0$ and achieves an approximation of $O(\eta^2)$ as long as $\eta \leq \bar{\eta}$ and $O(\bar{\eta}^2 n)$ otherwise.

[1] Note that even if the processing times are known in advance, no known polynomial time algorithm can achieve an approximation factor better than 2, and it is NP-hard to achieve a factor better than 1.5 [21].

Finally, in Section 6 we complement our positive results with an impossibility result, showing that achieving a consistency of 1, i.e., returning an optimal schedule whenever the predictions are accurate, necessary leads to unbounded robustness (irrespective of any computational limitations).

1.2 Related work

Strategyproof scheduling. Apart from proposing their n-approximate mechanism and conjecturing that this is the best possible guarantee across all deterministic and strategyproof mechanisms, Nisan and Ronen [27, 28] also proved an approximation lower bound of 2. This lower bound was later improved to 2.41 by Christodoulou et al. [13], then 2.61 by Koutsoupias and Vidali [19]. More recently, Giannakopoulos et al. [16] further raised this lower bound to 2.755 and then Dobzinski and Shaulker [14] pushed it to 3. Whether a constant approximation is possible remained an open problem until very recently, when the breakthrough result of [11] eventually proved a lower bound of $1 + \sqrt{n-1}$. Earlier work by Ashlagi et al. [3] had also shown a lower bound of n for the special class of mechanisms that are anonymous. Note that although these lower bounds focus on deterministic mechanisms, even if we allow randomization, the best known approximation guarantee achievable by a randomized strategyproof mechanism remains $O(n)$ [10].

Learning-augmented framework. Motivated by the shortcomings of worst-case analysis, an exciting new literature has focused on the design and analysis of "learning-augmented algorithms", or "algorithms with predictions" (see [26] for a survey of some early work and [23] for a more up-to-date list of papers in this area). This literature assumes that the algorithm is provided with predictions regarding the instance at hand and its performance is evaluated using *consistency* and *robustness*, which are the two primary metrics introduced by Lykouris and Vassilvtiskii [24]. During the last five years, more than 100 papers have revisited classic algorithmic problems using this framework, including online paging [24], scheduling [29], and secretary problems [15, 2], optimization problems with covering [7] and knapsack constraints [18], as well as Nash social welfare maximization [9] and several graph problems [4]. This line of work also studied online scheduling problems (e.g., [30, 25, 20, 6, 22, 8, 5]), but it was restricted to non-strategic settings and the predictions were used to overcome information limitations regarding the future, rather than limitations regarding privately held information. Very recently, Agrawal et al. [1] and, independently, also Xu and Lu [32] extended the framework to settings involving strategic agents. Agrawal et al. [1] provided optimal learning-augmented mechanisms, focusing on the strategic facility location problem, while Xu and Lu [32] considered a variety of different problems.

One of the problems considered in [32] was the strategic scheduling problem that we study in this paper. The authors gave a strategyproof mechanism that is $O(\gamma)$-consistent and $O(n^3/\gamma^2)$-robust, for any $\gamma \in [1, n]$. The mechanism in this paper used greedy allocation with weights determined by the predictions. However, to achieve any interesting consistency bounds (i.e., $o(n)$), their mechanism cannot guarantee a robustness better than $\omega(n^2)$; in particular, to achieve an optimal consistency of $O(1)$, the mechanism's robustness must be $\Omega(n^3)$. While our high-level approach shares similarities with this work, our mechanism use the predictions in a more cautious way trying to avoid the unnecessary scaling. Our main result significantly improves upon theirs achieving the best of both worlds: an optimal consistency of $O(1)$ and the best-known robustness of $O(n)$.

2 Preliminaries

Makespan minimization on unrelated machines. In the classic problem of scheduling on unrelated machines there is a set N of n machines, a set M of m jobs, and processing times $p(i,j)$ for a job $j \in M$ to be processed on machine $i \in N$. We use $\mathbf{x} \in \{0,1\}^{n \times m}$ to denote the *allocation* of jobs to machines, where $x(i,j) = 1$ if and only if job j is allocated to machine i. Given an instance $\mathbf{p}$ of processing times and an allocation $\mathbf{x}$, the *makespan* is the maximum over all machines of the total processing time assigned to that machine, i.e., $\texttt{MS}(\mathbf{p},\mathbf{x}) = \max_i \sum_j p(i,j) \cdot x(i,j)$. The goal is to find an allocation $\mathbf{x}$ such that the makespan is minimized. We denote the optimal makespan for an instance $\mathbf{p}$ by $\texttt{OPT}(\mathbf{p}) = \min_{\mathbf{x}} \texttt{MS}(\mathbf{p},\mathbf{x})$ and we let $\mathbf{x}^\star(\mathbf{p})$ denote an optimal allocation for instance $\mathbf{p}$ (so $\texttt{MS}(\mathbf{p},\mathbf{x}^\star(\mathbf{p})) = \texttt{OPT}(\mathbf{p})$). We abuse notation with $\texttt{OPT}$ and $\mathbf{x}^\star$ when the instance $\mathbf{p}$ is clear from context.

Strategic makespan minimization. In the strategic version of this problem, each machine i is controlled by a distinct self-interested agent. Each such agent incurs a cost for processing jobs that is equal to its load, i.e., the sum of the processing times of the jobs assigned to i and needs to receive a payment in exchange. Specifically, given an allocation $\mathbf{x}$, if agent i receives a payment ρ_i, and their utility is equal to $\rho_i - \sum_{j:x(i,j)=1} p(i,j)$. The processing times $\mathbf{p}_i = (p(i,1),\ldots,p(i,m))$ are private information that is known only to the agent controlling machine i, and each agent can misreport this information aiming to increase her utility. A scheduling mechanism consists of an allocation rule $\mathbf{x}(\mathbf{p}) \in \{0,1\}^{n \times m}$ for assigning jobs to machines and a payment rule $\boldsymbol{\rho}(\mathbf{p}) \in \mathbb{R}^n$ for compensating the machines for their costs. A scheduling mechanism is *strategyproof* if truthfully reporting $\mathbf{p}_i$ is a dominant strategy for every machine i, i.e., it is always an optimal strategy for agent i to report the truth, regardless of what every other agent reports.

Characterization of strategyproof mechanisms. Seminal work on strategic scheduling has characterized the family of allocations rules $\mathbf{x}(\mathbf{p})$ that admit a payment rule $\boldsymbol{\rho}(\mathbf{p})$ such that $(\mathbf{x}(\mathbf{p}), \boldsymbol{\rho}(\mathbf{p}))$ is a strategyproof scheduling mechanism.

▶ **Definition 1** (Monotonicity Property). *An allocation algorithm is monotone if fixing the processing time of all other agents $\mathbf{p}_{-i}$, for every two reports of agent i, $\mathbf{p}_i$ and $\mathbf{p}'_i$, the associated allocations $\mathbf{x}$ and $\mathbf{x}'$ satisfy*

$$\sum_{j \in M} (x(i,j) - x'(i,j))(p(i,j) - p'(i,j)) \leq 0.$$

▶ **Lemma 2** ([28, 31]). *An allocation algorithm $\mathbf{x}(\mathbf{p})$ admits a payment rule $\boldsymbol{\rho}(\mathbf{p})$ s.t. $(\mathbf{x}(\mathbf{p}), \boldsymbol{\rho}(\mathbf{p}))$ is a strategyproof scheduling mechanism if and only if $\mathbf{x}(\mathbf{p})$ satisfies the monotonicity property.*

This fundamental result reduces the mechanism design problem of strategic scheduling to an algorithm design problem where the goal is to find a (near)-optimal monotone allocation. One simple but natural mechanism that satisfies the above property is the *greedy mechanism*: it assigns each job j to the machine with minimum processing time for job j, i.e., $x(i,j) = 1$ if $i = \arg\min_{i' \in N} p(i',j)$. Nisan and Ronen [28] analyzed this mechanism and showed that it achieves an n-approximation to the optimal makespan.

Learning-augmented mechanism design. In the learning-augmented mechanism design framework, before requesting the processing time vector $\mathbf{p}_i$ from each machine i, the designer is provided with a prediction $\hat{\mathbf{p}}$ of the entire $n \times m$ processing time values $p(i,j)$. The

designer can use this information to choose the rules of the mechanism but, as in the standard stretagyproof scheduling setting, the final mechanism, denoted $\mathbf{x}(\mathbf{p}, \hat{\mathbf{p}})$, needs to be strategyproof. We use *consistency* and *robustness* to measure the performance of the mechanism. where consistency is the worst-case approximation given a accurate prediction, i.e., $\mathbf{p} = \hat{\mathbf{p}}$ and the robustness is the worst-case approximation given any predictions. Given a instance $\mathbf{p}$, a mechanism is α-*consistent* if it achieves an α-approximation ratio when the prediction is correct ($\hat{\mathbf{p}} = \mathbf{p}$), i.e.,

$$\max_{\mathbf{p}} \left\{ \frac{\texttt{MS}(\mathbf{p}, \mathbf{x}(\mathbf{p}, \mathbf{p}))}{\texttt{MS}(\mathbf{p}, \mathbf{x}^\star(\mathbf{p}))} \right\} \leq \alpha.$$

A mechanism is β-*robust* if it achieves a β-approximation ratio even when the predictions are arbitrarily wrong, i.e.,

$$\max_{\mathbf{p}, \hat{\mathbf{p}}} \left\{ \frac{\texttt{MS}(\mathbf{p}, \mathbf{x}(\mathbf{p}, \hat{\mathbf{p}}))}{\texttt{MS}(\mathbf{p}, \mathbf{x}^\star(\mathbf{p}))} \right\} \leq \beta.$$

One can also evaluate the worst-case approximation ratio as a function of the prediction error $\eta \geq 0$. In this paper we define the error $\eta(\mathbf{p}, \hat{\mathbf{p}})$ as the largest ratio between the predicted processing time and actual processing time for any i j pair, formally, $\eta = \max_{i \in N, j \in M} \max \left\{ \frac{\hat{p}(i,j)}{p(i,j)}, \frac{p(i,j)}{\hat{p}(i,j)} \right\}$. Given a prediction error η, a mechanism achieves a $\gamma(\eta)-$approximation if

$$\max_{\mathbf{p}, \hat{\mathbf{p}}: \eta(\mathbf{p}, \hat{\mathbf{p}}) \leq \eta} \left\{ \frac{\texttt{MS}(\mathbf{p}, \mathbf{x}(\mathbf{p}, \hat{\mathbf{p}}))}{\texttt{MS}(\mathbf{p}, \mathbf{x}^\star(\mathbf{p}))} \right\} \leq \gamma(\eta).$$

Note that for $\eta = 1$ the bound corresponds to the consistency guarantee and for $\eta \to \infty$ it captures the robustness guarantee.

3 Warm-Up: a 4-Consistent and n^2-Robust Mechanism

As a warm-up, we first present a strategyproof mechanism that is 4-consistent and n^2-robust. In the next section, we build on this mechanism to obtain a mechanism with $O(n)$ robustness.

The mechanism. Nisan and Ronen [27] showed that the greedy mechanism that assigns each job j to the machine with minimum processing time for job j has a wost-case approximation factor of n, implying that it is n-consistent and n-robust. On the other hand, returning a schedule that is optimal for the predicted instance gives a 1-consistent mechanism with unbounded robustness. SimpleScaledGreedy, described formally in Mechanism 1, is a variant of the greedy mechanism where the processing times are scaled as a function of the predictions. More formally, instead of greedily assigning job j to the machine i with minimum processing time $p(i, j)$, SimpleScaledGreedy assigns j to the machine i with minimum scaled processing time $r(i, j) \cdot p(i, j)$, where $r(i, j)$ are scalars that are defined by the mechanism. We note that even when this scaling depends arbitrarily on the predicted instance $\hat{\mathbf{p}}$, this mechanism is monotone (which we formally show in Lemma 8), and thus strategyproof (these scalars are independent of the reported instance $\mathbf{p}$).

The central part of our mechanism thus consists of constructing scalars $r(i, j)$ as a function of the predictions. The mechanism first computes an assignment $\hat{\mathbf{x}}$ for the predicted instance by using a, not necessarily strategyproof, scheduling algorithm Makespan-Min that is given as input to the mechanism. We call $\hat{\mathbf{x}}$ the predicted assignment. Let $\hat{\imath}_j$ be the machine that j is assigned to according to the predicted assignment, which we call the predicted machine

of job j. Note that with scalars $r(i,j) = \frac{\hat{p}(\hat{\imath}_j,j)}{\hat{p}(i,j)}$, we have that $r(i,j) \cdot \hat{p}(i,j) = \hat{p}(\hat{\imath}_j,j)$ for all jobs j. Thus, with such scaling factors and by breaking ties in favor of $\hat{\imath}_j$, each job j would be assigned to its predicted machine $\hat{\imath}_j$. The issue with such scaling factors is that they can be either arbitrarily large or arbitrarily small, which causes the robustness to be unbounded since these factors can cause a job to be allocated on an undesirable machine when the predictions are inaccurate. To avoid this issue, we cap the scaling factors to ensure that they have values between 1 and n. We break ties in favor of $\hat{\imath}_j$ when $r(\hat{\imath}_j,j) \cdot p(\hat{\imath}_j,j) = \arg\min_i r(i,j) \cdot p(i,j)$, and otherwise we break them in favor of the machine with minimum index.

Note that when the prediction is correct, i.e., $\hat{\mathbf{p}} = \mathbf{p}$, the mechanism assigns a job j to its predicted machine $\hat{\imath}_j$ unless $\min_i \hat{p}(i,j) \leq \frac{\hat{p}(\hat{\imath}_j,j)}{n}$, i.e., unless there exists a machine i with much smaller predicted processing time for j than the predicted machine $\hat{\imath}_j$ of job j. If $\min_i \hat{p}(i,j) \leq \frac{\hat{p}(\hat{\imath}_j,j)}{n}$, and the prediction is correct, the mechanism assigns j to the machine $\arg\min_i p(i,j)$ with the smallest true (and predicted) processing time for j.

Mechanism 1 SimpleScaledGreedy.

Input: instance $\mathbf{p} \in \mathbb{R}^{n \times m}$, predicted instance $\hat{\mathbf{p}} \in \mathbb{R}^{n \times m}$, scheduling algorithm Makespan-Min
$\hat{\mathbf{x}} \leftarrow$ Makespan-Min$(\hat{\mathbf{p}})$
$\hat{\imath}_j \leftarrow$ the machine i that job j is assigned to according to $\hat{\mathbf{x}}$, i.e., $\hat{x}(\hat{\imath}_j,j) = 1$, for each $j \in M$
$r(i,j) \leftarrow \max\left(1, \min\left(\frac{\hat{p}(\hat{\imath}_j,j)}{\hat{p}(i,j)}, n\right)\right)$, for each $(i,j) \in N \times M$
$i_j \leftarrow \arg\min_i r(i,j) \cdot p(i,j)$, break ties in favor of $\hat{\imath}_j$, for each $j \in M$
if $i = i_j$ **then** $x(i,j) = 1$ **else** $x(i,j) = 0$, for each $(i,j) \in N \times M$
return $\mathbf{x}$

We first analyze the mechanism's consistency, then its robustness, and finally show that it is strategyproof.

The consistency. We first introduce some notation. We assume that Makespan-Min is an α-approximation algorithm for makespan minimization and let $M_i = \{j \in M : x(i,j) = 1\}$ and $\hat{M}_i = \{j \in M : \hat{x}(i,j) = 1\}$ be the sets of jobs j such that j is assigned to machine i by SimpleScaledGreedy and the predicted assignment, respectively. To bound the total processing time of the jobs M_i assigned to machine i, for each $i \in N$, we separately bound the total processing times of $M_i \cap \hat{M}_i$ and $M_i \setminus \hat{M}_i$. Bounding the total processing time of jobs in $M_i \cap \hat{M}_i$, i.e., jobs assigned to machine i according to both the mechanism's assignment $\mathbf{x}$ and the predicted assignment $\hat{\mathbf{x}}$, by $\alpha\texttt{OPT}$ is almost immediate when the predictions are accurate.

▶ **Lemma 3.** *Assume that $\hat{\mathbf{p}} = \mathbf{p}$, then, for all $i \in N$, $\sum_{j \in M_i \cap \hat{M}_i} p(i,j) \leq \alpha \mathit{OPT}(\mathbf{p})$.*

Proof. For any machine $i \in N$, observe that

$$\sum_{j \in M_i \cap \hat{M}_i} p(i,j) \leq \sum_{j \in \hat{M}_i} p(i,j) = \sum_{j \in \hat{M}_i} \hat{p}(i,j) \leq \texttt{MS}(\hat{\mathbf{p}}, \hat{\mathbf{x}}) \leq \alpha\texttt{OPT}(\hat{\mathbf{p}}) = \alpha\texttt{OPT}(\mathbf{p}),$$

where first equality is since $\mathbf{p} = \hat{\mathbf{p}}$, the second inequality is by definition of $\hat{M}_i$ and the makespan of allocation $\hat{\mathbf{x}}$, the last inequality is since Makespan-Min is an α-approximation algorithm, and the last equality is since $\hat{\mathbf{p}} = \mathbf{p}$. ◀

The main part of the analysis of the consistency bound is the total processing time of jobs that are assigned to i even though i is not their predicted assignment, i.e., the total processing time of $M_i \setminus \hat{M}_i$. We first show that such jobs must have a scalar $r(i,j) = n$.

▶ **Lemma 4.** *For any job $j \in M$ and machine $i \in N$, if $\hat{\mathbf{p}} = \mathbf{p}$ and $j \in M_i \setminus \hat{M}_i$, then $r(i,j) = n$.*

Proof. Consider a job $j \in M$ and machine $i \in N$ such that $\hat{\mathbf{p}} = \mathbf{p}$ and $j \in M_i \setminus \hat{M}_i$. Since $j \in M_i \setminus \hat{M}_i$, we have that $\hat{\imath}_j \neq i$ and $r(i,j) \cdot p(i,j) < r(\hat{\imath}_j, j) \cdot p(\hat{\imath}_j, j)$ by definition of SimpleScaledGreedy. We get that

$$r(i,j) < r(\hat{\imath}_j, j) \cdot \frac{p(\hat{\imath}_j, j)}{p(i,j)} = \max\left(1, \min\left(\frac{\hat{p}(\hat{\imath}_j, j)}{\hat{p}(\hat{\imath}_j, j)}, n\right)\right) \cdot \frac{p(\hat{\imath}_j, j)}{p(i,j)} = \frac{p(\hat{\imath}_j, j)}{p(i,j)} = \frac{\hat{p}(\hat{\imath}_j, j)}{\hat{p}(i,j)},$$

where the first equality is by definition of $r(i,j)$ and the last equality since $\hat{\mathbf{p}} = \mathbf{p}$. Since $r(i,j) < \frac{\hat{p}(\hat{\imath}_j, j)}{\hat{p}(i,j)}$ then again by the definition $r(i,j)$ we get that $r(i,j) = n$. ◀

We are now ready to bound the total processing time of $M_i \setminus \hat{M}_i$.

▶ **Lemma 5.** *Assume that $\hat{\mathbf{p}} = \mathbf{p}$, then, for all $i \in N$, $\sum_{j \in M_i \setminus \hat{M}_i} p(i,j) \leq \alpha \cdot \mathit{OPT}(\mathbf{p})$.*

Proof. Let i be an arbitrary machine and assume that $\hat{\mathbf{p}} = \mathbf{p}$. First, observe that

$$\sum_{j \in M_i \setminus \hat{M}_i} p(i,j) \leq \sum_{j \in M_i \setminus \hat{M}_i} \frac{r(\hat{\imath}_j, j)}{r(i,j)} p(\hat{\imath}_j, j) = \frac{1}{n} \sum_{j \in M_i \setminus \hat{M}_i} p(\hat{\imath}_j, j) \leq \frac{1}{n} \sum_{j \in M} p(\hat{\imath}_j, j) = \frac{1}{n} \sum_{j \in M} \hat{p}(\hat{\imath}_j, j)$$

where the first inequality is since j was assigned to machine i by SimpleScaledGreedy, the first equality since $r(\hat{\imath}_j, j) = 1$ and $r(i,j) = n$ for $j \in M_i \setminus \hat{M}_i$ by Lemma 4. We also have that

$$\frac{1}{n} \sum_{j \in M} \hat{p}(\hat{\imath}_j, j) = \frac{1}{n} \sum_{i \in N} \sum_{j \in \hat{M}_i} \hat{p}(i,j) \leq \max_{i \in N} \sum_{j \in \hat{M}_i} \hat{p}(i,j) = \mathtt{MS}(\hat{\mathbf{p}}, \hat{\mathbf{x}}) \leq \alpha \cdot \mathtt{OPT}(\hat{\mathbf{p}}) = \alpha \cdot \mathtt{OPT}(\mathbf{p})$$

where the last inequality is since Makespan-Min is an α-approximation algorithm. Combing the above two series of inequalities, we get $\sum_{j \in M_i \setminus \hat{M}_i} p(i,j) \leq \alpha \cdot \mathtt{OPT}(\mathbf{p})$. ◀

By combining Lemma 3 and Lemma 5, we get that the mechanism is 2α-consistent.

▶ **Lemma 6.** *SimpleScaledGreedy is a mechanism that is 2α-consistent.*

Proof. Assume that $\mathbf{p} = \hat{\mathbf{p}}$. For any machine i, we have

$$\sum_{j \in M_i} p(i,j) = \sum_{j \in M_i \cap \hat{M}_i} p(i,j) + \sum_{j \in M_i \setminus \hat{M}_i} p(i,j) \leq \alpha \mathtt{OPT}(\mathbf{p}) + \alpha \mathtt{OPT}(\mathbf{p}) = 2\alpha \mathtt{OPT}(\mathbf{p})$$

where the inequality is by Lemma 3 and Lemma 5. ◀

The robustness. The crucial part of the mechanism that allows obtaining a robustness of n^2 is that the scalars $r(i,j)$ are bounded between 1 and n.

▶ **Lemma 7.** *SimpleScaledGreedy is a mechanism that is $\Theta(n^2)$-robust.*

Proof. We first show that the robustness is at most n^2. Consider an instance $\mathbf{p}$, an optimal assignment $\mathbf{x}^\star$ for $\mathbf{p}$, a job j, and a machine i such that $j \in M_i$. Let $i_j^\star$ be the machine that j is assigned to according to $\mathbf{x}^\star$. By the mechanism, we have $1 \leq r(i', j) \leq n$ for all machines i'. We therefore have:

$$p(i,j) \leq r(i,j) \cdot p(i,j) \leq r(i_j^\star, j) \cdot p(i_j^\star, j) \leq n \cdot p(i_j^\star, j),$$

where the second inequality is by line 5 and the fact that j is assigned to i. We get that

$$\texttt{MS}(\mathbf{p}, \mathbf{x}) = \max_{i \in N} \sum_{j \in M_i} p(i,j) \leq \sum_{i \in N} \sum_{j \in M_i} p(i,j) \leq n \sum_{i \in N} \sum_{j \in M_i} p(i_j^\star, j) \leq n^2 \max_{i \in N} \sum_{j \in M_i} p(i_j^\star, j) = n^2 \texttt{OPT}.$$

In fact, the robustness guarantee is tight. Due to space limitation, we defer the proof to the appendix A. ◀

Strategyproofness. We show that the mechanism is strategyproof by proving that it is monotone and then using Lemma 2.

▶ **Lemma 8.** *SimpleScaledGreedy is a strategyproof mechanism.*

Proof. Consider a machine $i \in N$ and a predicted assignment $\hat{\mathbf{x}}$. Consider two instances $\mathbf{p}$ and $\mathbf{p}'$ that differ only on machine i and the associated allocations $\mathbf{x}$ and $\mathbf{x}'$ returned by SimpleScaledGreedy when the predicted assignment is $\hat{\mathbf{x}}$. Given a fixed predicted assignment $\hat{\mathbf{x}}$, the mechanism is deterministic, so $x(i,j) \in \{0,1\}$ for all $i \in N$ and $j \in M$. Since the predicted assignment is fixed, we also have that the scalars $r(i,j)$ are fixed.

Let $j \in M$ be an arbitrary job. Without loss of generality, we assume that $p(i,j) \leq p'(i,j)$. Consider the case where $x(i,j) = 0$, i.e., job j is not assigned to machine i for processing times $\mathbf{p}$. By the mechanism, this implies that $i \neq \arg\min_{i'} r(i',j) \cdot p(i',j)$. Since (1) $p(i',j) = p'(i',j)$ for $i' \neq i$, (2) $p(i,j) \leq p'(i,j)$, and (3) $i \neq \arg\min_{i'} r(i',j) \cdot p(i',j)$, we get that $i \neq \arg\min_{i'} r(i',j) \cdot p'(i',j)$, which implies $x'(i,j) = 0$. We thus obtain that $x(i,j) \geq x'(i,j)$, which implies that $(x(i,j) - x'(i,j))(p(i,j) - p'(i,j)) \leq 0$ since $p(i,j) \leq p'(i,j)$. SimpleScaledGreedy is thus a monotone mechanism, which, by Lemma 2, implies that it is strategyproof. ◀

The main result for SimpleScaledGreedy. Combining Lemmas 6, 7, and 8, we obtain the main result for SimpleScaledGreedy.

▶ **Theorem 9.** *SimpleScaledGreedy is a strategyproof, 2α-consistent, and $\Theta(n^2)$-robust mechanism.*

To obtain a polynomial-time mechanism, we can have $\alpha = 2$ and a consistency of 4 with Makespan-Min being the 2-approximation algorithm for makespan minimization on unrelated machines by Lenstra et al. [21]. By ignoring running time considerations, we can obtain $\alpha = 1$ and a consistency of 2 with Makespan-Min finding an optimal schedule.

4 The Scaled Greedy Mechanism

In this section, we present a deterministic polynomial-time mechanism that is $(4 + 2\gamma)$-consistent and $\left(1 + \frac{1}{\gamma}\right) n$-robust, where $\gamma \in \left(0, \frac{n}{2} - 1\right)$ is a parameter that controls the tradeoff between the consistency and robustness achieved by the mechanism. This mechanism builds on the mechanism from the previous section. In the next section, we extend this mechanism to obtain a mechanism that achieves an approximation as a function of the prediction error.

4.1 The mechanism

ScaledGreedy, which is described formally in Mechanism 2, defines scalars $r(i,j)$ and assigns a job j to the machine i with minimum scaled processing time $r(i,j) \cdot p(i,j)$. As in SimpleScaledGreedy, the mechanism first computes a predicted assignment $\hat{\mathbf{x}}$ for the predicted instance $\hat{\mathbf{p}}$. Recall that scalars $r(i,j) = \frac{\hat{p}(\hat{\imath}_j,j)}{\hat{p}(i,j)}$ are designed so that the mechanism assigns a job j to its predicted assignment $\hat{\imath}_j$ when the predictions are correct, even if j has a smaller predicted processing time on i than on $\hat{\imath}_j$. To improve the robustness from quadratic to linear, ScaledGreedy upper bounds the scalars $r(i,j)$ more aggressively, and more carefully, than SimpleScaledGreedy, which simply upper bounds them by n. Scalars that are close to 1 are intuitively better for the robustness of the mechanism since the greedy mechanism, which achieves an n-approximation, corresponds to the mechanism where the scalars are all equal to 1. The mechanism upper bounds the scalars by maintaining sets J_i, I and T.

Mechanism 2 ScaledGreedy.

Input: instance $\mathbf{p} \in \mathbb{R}^{n\times m}$, predicted instance $\hat{\mathbf{p}} \in \mathbb{R}^{n\times m}$, scheduling algorithm Makespan-Min, prediction confidence parameter $\gamma \in (0, \frac{n}{2} - 1)$
$\hat{\mathbf{x}} \leftarrow$ Makespan-Min$(\hat{\mathbf{p}})$
$\hat{\imath}_j \leftarrow$ the machine i that job j is assigned to according to $\hat{\mathbf{x}}$, i.e., $\hat{x}(\hat{\imath}_j, j) = 1$, for each $j \in M$
$r(i,j) \leftarrow \max\left(\frac{\hat{p}(\hat{\imath}_j,j)}{\hat{p}(i,j)}, 1\right)$ for each $(i,j) \in N \times M$
$J_i \leftarrow \emptyset$ for all $i \in N$
$I \leftarrow N$
$T \leftarrow \{(i,j) : j \in M, i = \arg\min_{i' \in I : \hat{p}(i',j) < \hat{p}(\hat{\imath}_j,j)} \hat{p}(i',j)\}$
while T *is not empty* **do**
    $(i^*, j^*) \leftarrow \arg\max_{(i,j)\in T} \frac{\hat{p}(\hat{\imath}_j,j)}{\hat{p}(i,j)}$
    $J_{i^*} \leftarrow J_{i^*} \cup \{j^*\}$
    for *all* i *s.t.* $\hat{p}(i^*, j^*) \le \hat{p}(i, j^*)$ **do**
        update $r(i, j^*) \leftarrow 1$
    $I \leftarrow \{i \in N : \sum_{j\in J_i} \hat{p}(i,j) < \gamma \texttt{MS}(\hat{\mathbf{p}}, \hat{\mathbf{x}})\}$
    $T \leftarrow \{(i,j) : j \in M \setminus (\cup_i J_i), i = \arg\min_{i' \in I : \hat{p}(i',j) < \hat{p}(\hat{\imath}_j,j)} \hat{p}(i',j)\}$
$i_j \leftarrow \arg\min_i r(i,j) \cdot p(i,j)$, break ties in favor of $\hat{\imath}_j$ first and i' if $j \in J_{i'}$ second, for each $j \in M$
if $i = i_j$ **then** $x(i,j) = 1$ **else** $x(i,j) = 0$, for each $(i,j) \in N \times M$
return $\mathbf{x}$

J_i is the set of jobs j for which ScaledGreedy assigns j to i when the predictions are correct even though their predicted assignment is not i. When ScaledGreedy adds a job j^* to J_{i^*}, it allows to set the scalars $r(i,j^*)$ to $r(i,j^*) = 1$, for all machines i such that $\hat{p}(i^*,j^*) \le \hat{p}(i,j^*)$, while guaranteeing that job j^* is assigned to i^* when the predictions are correct. The sets J_i are initially empty. To maintain a good consistency, we bound the total predicted processing time of the jobs in a set J_i by $\gamma\texttt{MS}(\hat{\mathbf{p}}, \hat{\mathbf{x}})$. I is the set of machines for which this bound has not yet been violated, i.e., the machines i that are such that we can still add jobs to J_i. I initially consists of all the machines. T is the set of pairs (i,j) such that job j is a candidate to be added to J_i. A pair (i,j) is in T if (1) j has not yet been assigned by ScaledGreedy to another machine, i.e., $j \notin \cup_i J_i$, (2) the upper bound on

the total predicted processing time of the jobs already in J_i has not yet been violated, i.e., $i \in I$, (3) j has a smaller processing time on i than on its predicted assignment $\hat{\imath}_j$, and (4) i is the machine with minimum predicted processing time $\hat{p}(i,j)$ among the machines in I. At each iteration, job j^* is added J_{i^*} where (i^*, j^*) is the pair (i,j) with maximum $\frac{\hat{p}(\hat{\imath}_j,j)}{\hat{p}(i,j)}$ in T. For each job j, if there are multiple machines with minimum scaled processing time $r(i,j) \cdot p(i,j)$, ties are broken in favor of, in order of priority, (1) machine $\hat{\imath}_j$ and, if there is one, (2) the machine i' such that $j \in J_{i'}$, and (3) the machine with minimum index.

4.2 The analysis

We first analyze the mechanism's consistency and robustness, and finally show it is strategyproof. The following simple observation is used for both the consistency and the robustness analyses.

▶ **Observation 10.** *For every machine $i \in N$ and job $j \in M$, at every iteration of* ScaledGreedy*, we have $r(i,j) \geq 1$ and, if $\hat{p}(i,j) \geq \hat{p}(\hat{\imath}_j, j), r(i,j) = 1$.*

The consistency. As in the analysis of SimpleScaledGreedy, we assume that Makespan-Min is an α-approximation algorithm for makespan minimization and let $M_i = \{j \in M : x(i,j) = 1\}$ and $\hat{M}_i = \{j \in M : \hat{x}(i,j) = 1\}$ be the sets of jobs j such that j is assigned to machine i by ScaledGreedy and the predicted assignment, respectively. To bound the total processing time of the jobs M_i assigned to a machine i, we again separately bound the total processing time of $M_i \cap \hat{M}_i$ and $M_i \setminus \hat{M}_i$. The next lemma, which bounds the total processing time of $M_i \cap \hat{M}_i$ when the predictions are correct, is identical as Lemma 3 for SimpleScaledGreedy, and its proof is also identical to the proof of Lemma 3.

▶ **Lemma 11.** *Assume that $\hat{\mathbf{p}} = \mathbf{p}$, then, for all $i \in N$, $\sum_{j \in M_i \cap \hat{M}_i} p(i,j) \leq \alpha \mathit{OPT}(\mathbf{p})$.*

Next, to bound the total processing time of $M_i \setminus \hat{M}_i$, we first characterize the jobs in $M_i \setminus \hat{M}_i$ as being exactly the jobs in J_i.

▶ **Lemma 12.** *Assume that $\hat{\mathbf{p}} = \mathbf{p}$, then, for every machine $i \in N$, we have that $J_i = M_i \setminus \hat{M}_i$.*

Proof. Consider a machine $i \in N$ and a job $j \in J_i$. We want to show that $j \in M_i \setminus \hat{M}_i$. Note that since $j \in J_i$, we had $(i,j) \in T$ at some iteration of the mechanism, which implies by Line 14 that $\hat{p}(i,j) < \hat{p}(\hat{\imath}_j, j)$. Since $\hat{p}(i,j) < \hat{p}(\hat{\imath}_j, j)$, we have that $i \neq \hat{\imath}_j$. Since $\hat{x}(\hat{\imath}_j, j) = 1$ by definition of $\hat{\imath}_j$, we have that $\hat{x}(i,j) = 0$, which implies $j \notin \hat{M}_i$.

Next, we show that $j \in M_i$, which we show by proving that $r(i,j)p(i,j) \leq r(i',j)p(i',j)$ for all $i' \notin \{i, \hat{\imath}_j\}$ and $r(i,j)p(i,j) < r(\hat{\imath}_j, j)p(\hat{\imath}_j, j)$. There are two cases of machines $i' \notin \{i, \hat{\imath}_j\}$. The first case is if i' is such that $\hat{p}(i',j) < \hat{p}(i,j)$, which is the main part of the proof. Since $\hat{p}(i,j) < \hat{p}(\hat{\imath}_j, j)$ (which was shown earlier in this proof), we have $\hat{p}(i',j) < \hat{p}(\hat{\imath}_j, j)$, so $r(i',j)$ was initialized to $r(i',j) = \frac{\hat{p}(\hat{\imath}_j,j)}{\hat{p}(i',j)}$ in line 4. For all $i'' \neq i$, we have that $j \notin J_{i''}$ since, by line 14, $(i'',j) \notin T$ if $j \in J_i$, which implies that j cannot be added to $J_{i''}$. Thus, the only iteration of the mechanism where $j^\star = j$ is when $i^\star = i$, and since $\hat{p}(i',j) < \hat{p}(i,j)$, this implies that $r(i',j)$ is never updated in line 12. Together, with the fact that it was initialized to $r(i',j) = \frac{\hat{p}(\hat{\imath}_j,j)}{\hat{p}(i',j)}$, we have that $r(i',j) = \frac{\hat{p}(\hat{\imath}_j,j)}{\hat{p}(i',j)}$ when the mechanism terminated. Thus, when $\mathbf{p} = \hat{\mathbf{p}}$, we have

$$r(i,j)p(i,j) = 1 \cdot \hat{p}(i',j) < \hat{p}(\hat{\imath}_j, j) = \frac{\hat{p}(\hat{\imath}_j,j)}{\hat{p}(i',j)} \cdot \hat{p}(i',j) = r(i',j)p(i',j). \tag{1}$$

where the first equality is since $r(i,j) = 1$ (because $j \in J_i$). The second case of $i' \notin \{i, \hat{\imath}_j\}$ is if i' such that $\hat{p}(i',j) \geq \hat{p}(i,j)$. In this case, we have that

$$p(i,j) \cdot r(i,j) = \hat{p}(i,j) \leq \hat{p}(i',j) \leq p(i',j) \cdot r(i',j), \tag{2}$$

where the last inequality is by Observation 10. Next, note that

$$p(i,j) \cdot r(i,j) = \hat{p}(i,j) < \hat{p}(\hat{\imath}_j,j) = p(\hat{\imath}_j,j) \cdot r(\hat{\imath}_j,j), \tag{3}$$

where the inequality is since, at some iteration of the mechanism, $(i,j) \in T$ and the last equality since $r(\hat{\imath}_j,j) = 1$. Since we have shown that $r(i,j)p(i,j) \leq r(i',j)p(i',j)$ for all $i' \notin \{i, \hat{\imath}_j\}$ and $r(i,j)p(i,j) < r(\hat{\imath}_j,j)p(\hat{\imath}_j,j)$, we have that, by line 15 of the mechanism, $x(i,j) = 1$, which implies that $j \in M_i$. Since we have shown that $j \notin \hat{M}_i$ and $j \in M_i$, we get $j \in M_i \setminus \hat{M}_i$.

It remains to show that if $j \in M_i \setminus \hat{M}_i$, then $j \in J_i$. Consider an arbitrary job $j \in M_i \setminus \hat{M}_i$. Since $j \notin \hat{M}_i$, $i \neq \hat{\imath}_j$. Since $j \in M_i$ and $i \neq \hat{\imath}_j$, we have that

$$r(i,j)p(i,j) < r(\hat{\imath}_j,j)p(\hat{\imath}_j,j) = p(\hat{\imath}_j,j)$$

where the inequality is by line 15 of the mechanism and the equality since $r(\hat{\imath}_j,j) = 1$. Thus, $r(i,j) < \frac{p(\hat{\imath}_j,j)}{p(i,j)}$, which implies $r(i,j) = 1$. Thus there is some iteration of the mechanism where $j^\star = j$ got added to $J_{i^\star}$ for some machine $i^\star$. Assume by contradiction that $i^* \neq i$, so $j \in J_{i^\star}$ for $i^* \neq i$. Then, by the first part of this proof, we have that $j \in M_{i^\star}$, which is a contradiction with $j \in M_i$. Thus, $i^* = i$ and $j \in J_i$. ◀

The next lemma bounds the total processing time of jobs in $M_i \setminus \hat{M}_i$ by using the fact that $J_i = M_i \setminus \hat{M}_i$ and by bounding the total processing time of jobs in J_i.

▶ **Lemma 13.** *For any input parameter $\gamma \geq 0$, then, for every machine $i \in N$, if $J_i = M_i \setminus \hat{M}_i$, we have that $\sum_{j \in M_i \setminus \hat{M}_i} \hat{p}(i,j) < (1+\gamma)\alpha \mathit{OPT}(\hat{\mathbf{p}})$.*

Proof. Consider an arbitrary machine i and let j' denote the last job added into J_i by the mechanism. First observer that

$$\sum_{j \in M_i \setminus \hat{M}_i} \hat{p}(i,j) = \sum_{j \in J_i} \hat{p}(i,j) = \sum_{j \in J_i \setminus \{j'\}} \hat{p}(i,j) + \hat{p}(i,j') < \gamma \mathtt{MS}(\hat{\mathbf{p}}, \hat{\mathbf{x}}) + \hat{p}(i,j') < \gamma \mathtt{MS}(\hat{\mathbf{p}}, \hat{\mathbf{x}}) + \hat{p}(\hat{\imath}_{j'}, j')$$

where the first inequality is by Line 13 and Line 14 of the mechanism, and the last inequality is since $j' \in J_i$. Next, we have

$$\gamma \mathtt{MS}(\hat{\mathbf{p}}, \hat{\mathbf{x}}) + \hat{p}(\hat{\imath}_{j'}, j') \leq (1+\gamma)\mathtt{MS}(\hat{\mathbf{p}}, \hat{\mathbf{x}}) \leq (1+\gamma)\alpha \mathtt{OPT}(\hat{\mathbf{p}})$$

where the first inequality is since the makespan of an assignment is at least the processing time of a job j on a machine i, for any j assigned to i, the second inequality since Makespan-Min is an α-approximation algorithm. ◀

By combining Lemma 11 to bound $\sum_{j \in M_i \cap \hat{M}_i} p(i,j)$ and Lemma 13 to bound $\sum_{j \in M_i \setminus \hat{M}_i} p(i,j)$, the next lemma bounds $\sum_{j \in M_i} p(i,j)$ by $(2+\gamma)\alpha$ for any machine i, which implies that ScaledGreedy is $(2+\gamma)\alpha$-consistent.

▶ **Lemma 14.** *For any input parameter $\gamma \geq 0$, the ScaledGreedy mechanism is $(2+\gamma)\alpha$-consistent.*

Proof. Assume that $\mathbf{p} = \hat{\mathbf{p}}$ and consider an arbitrary machine i, then we have $J_i = M_i \setminus \hat{M}_i$ by Lemma 12. Thus

$$\sum_{j \in M_i} p(i,j) = \sum_{j \in M_i \cap \hat{M}_i} p(i,j) + \sum_{j \in M_i \setminus \hat{M}_i} p(i,j) \leq \alpha \mathtt{OPT}(\mathbf{p}) + (1+\gamma)\alpha \mathtt{OPT}(\mathbf{p}) = (2+\gamma)\alpha \mathtt{OPT},$$

where the inequality is by Lemma 11, Lemma 13 and the assumption that $\mathbf{p} = \hat{\mathbf{p}}$. ◀

The robustness. Recall that the main reason that SimpleScaledGreedy achieved bounded robustness is thanks to the scalars being individually upper bounded by n. To show that ScaledGreedy achieves linear robustness, we give a stronger upper bound on the sum of the largest scalar on each machine. More precisely, in Lemma 17, we show that $\sum_{i:\max_j r(i,j)>1} \max_j r(i,j) \leq \frac{n}{\gamma}$, which is the main lemma to show the linear robustness. To prove Lemma 17, we first need two helper lemmas. The first is that for a fixed machine i, the jobs added to J_i, i.e. the jobs j with scalars $r(i,j)$ modified from $\frac{\hat{p}(\hat{\imath}_j,j)}{\hat{p}(i,j)}$ to 1 are the jobs with largest ratio $\frac{\hat{p}(\hat{\imath}_j,j)}{\hat{p}(i,j)}$.

▶ **Lemma 15.** *For any machine $i \in N$, job $j \in J_i$, and job $j' \notin J_i$, if $r(i,j') > 1$, then*

$$\frac{\hat{p}(\hat{\imath}_j,j)}{\hat{p}(i,j)} \geq \frac{\hat{p}(\hat{\imath}_{j'},j')}{\hat{p}(i,j')}.$$

Proof. Assume for contradiction that $\frac{\hat{p}(\hat{\imath}_{j'},j')}{\hat{p}(i,j')} > \frac{\hat{p}(\hat{\imath}_j,j)}{\hat{p}(i,j)}$ for some $j \in J_i$ and $j' \notin J_i$ such that $r(i,j') > 1$. Since $j \in J_i$, we have that $(i,j) \in T$ at some iteration of the mechanism. We let the iteration when j is added to J_i by the mechanism be t. First note that $\hat{p}(i,j') < \hat{p}(\hat{\imath}_{j'},j')$, otherwise $\frac{\hat{p}(\hat{\imath}_{j'},j')}{\hat{p}(i,j')} \leq 1 < \frac{\hat{p}(\hat{\imath}_j,j)}{\hat{p}(i,j)}$ (where the strict inequality is since $j \in J_i$), which is a contradiction.

Since j was added to J_i at iteration t, we have $i \in I$ at iteration t, which implies $i \in I$ at all iterations $t' \leq t$. There are two cases. If, at iteration t, $j' \in J_{i'}$ for some machine i', then j' was added to $J_{i'}$ at some iteration $t' < t$. Thus, $(i',j') \in T$ at iteration t'. By line 14, this implies that $\hat{p}(i',j') \leq \hat{p}(i,j')$ since $i \in I$ at iteration t'. By line 12, j' added to $J_{i'}$ and $\hat{p}(i',j') \leq \hat{p}(i,j')$ imply $r(i,j') = 1$, which is a contradiction. Thus, at iteration t, $j' \notin \cup_{i'} J_{i'}$.

Since $\hat{p}(i,j') < \hat{p}(\hat{\imath}_{j'},j')$ and $j' \notin \cup_{i'} J_{i'}$, there exists machine i' such that $(i',j') \in T$ at iteration t and such that $\hat{p}(\hat{\imath}_{j'},j') \leq \hat{p}(i,j')$. We get that

$$\frac{\hat{p}(\hat{\imath}_{j'},j')}{\hat{p}(i',j')} \geq \frac{\hat{p}(\hat{\imath}_{j'},j')}{\hat{p}(i,j')} > \frac{\hat{p}(\hat{\imath}_j,j)}{\hat{p}(i,j)}$$

where the inequality is by the assumption of the proof. Since $(i',j') \in T$ and $\frac{\hat{p}(\hat{\imath}_{j'},j')}{\hat{p}(i',j')} > \frac{\hat{p}(\hat{\imath}_j,j)}{\hat{p}(i,j)}$, we have that $(i^*,j^*) \neq (i,j)$ at iteration t, which is a contradiction with j that is added to J_i at iteration t. ◀

The second helper lemma shows that if there is a job j with scalar $r(i,j) > 1$ with respect to machine i, i.e. a scalar that the mechanism did not decrease to 1, then we have that $\sum_{j' \in J_i} \hat{p}(i,j') \geq \gamma \texttt{MS}(\hat{\mathbf{p}}, \hat{\mathbf{x}})$, i.e., the upper bound $\gamma \texttt{MS}(\hat{\mathbf{p}}, \hat{\mathbf{x}})$ on the total processing time of jobs in J_i has been violated. Informally, the scalar $r(i,j)$ couldn't be decreased to 1 because J_i was already full.

▶ **Lemma 16.** *For any $\gamma > 0$, when ScaledGreedy terminates, for any machine i, if there is a job j such that $r(i,j) > 1$, then $\sum_{j' \in J_i} \hat{p}(i,j') \geq \gamma MS(\hat{\mathbf{p}}, \hat{\mathbf{x}})$.*

Proof. Assume for contradiction, that there exist a machine i with $r(i,j') > 1$ for some job j' and $\sum_{j \in J_i} \hat{p}(i,j) < \gamma \texttt{MS}(\hat{\mathbf{p}}, \hat{\mathbf{x}})$. In other words, $i \in I$ throughout the execution of the mechanism. Since $r_{i,j'} > 1$, we have $\hat{p}(i,j') < \hat{p}(\hat{\imath}_{j'},j')$, otherwise $r_{i,j'} = 1$ by line 4. If, at the end of the execution of ScaledGreedy, $j' \in J_{i'}$ for some machine i', then j' was added to $J_{i'}$ at some iteration t'. Thus, $(i',j') \in T$ at iteration t'. By line 14, this implies that $\hat{p}(i',j') \leq \hat{p}(i,j')$ since $i \in I$ at iteration t'. By line 12, j' added to $J_{i'}$ and $\hat{p}(i',j') \leq \hat{p}(i,j')$ imply $r(i,j') = 1$, which is a contradiction. Thus, when ScaledGreedy terminates, $j' \notin \cup_{i'} J_{i'}$.

Since $\hat{p}(i,j') < \hat{p}(\hat{\imath}_{j'},j')$ and $j' \notin \cup_{i'} J_{i'}$, either there exists machine i' such that $(i',j') \in T$ at the end of the execution and such that $\hat{p}(\hat{\imath}_{j'},j') \leq \hat{p}(i,j')$ or $(i,j') \in T$. However, both cases can't happen since the mechanism only terminates when T is empty. ◀

The next lemma is the main lemma for the robustness analysis, it upper bounds the sum of the largest scalars on each machine. To prove it, we use the two previous helper lemmas.

▶ **Lemma 17.** *For any $\gamma > 0$, when ScaledGreedy terminates, we have that*

$$\sum_{i:\max_j r(i,j)>1} \max_j r(i,j) \leq \frac{n}{\gamma}.$$

Proof. First note that the sets J_i for $i \in N$ are disjoint, since once any j is added to J_i for some machine i, j is no longer consider for T by line 14. We therefore have:

$$\texttt{MS}(\hat{\mathbf{p}},\hat{\mathbf{x}}) = \max_i \sum_{j:\hat{\imath}_j=i} \hat{p}(\hat{\imath}_j,j) \geq \frac{\sum_j \hat{p}(\hat{\imath}_j,j)}{n} \geq \frac{\sum_{i:\max_j r(i,j)>1} \sum_{j\in J_i} \hat{p}(\hat{\imath}_j,j)}{n}, \tag{4}$$

Where the first equation is by the definition of predicted machine, the first inequality is because the maximum is weakly greater than the average, and the last inequality is due to the fact that J_i and $J_{i'}$ are disjoint for any two distinct machines $i, i' \in N$. By Lemma 15, for any $j \in J_i$, we have

$$\hat{p}(\hat{\imath}_j,j) = \hat{p}(i,j) \cdot \frac{\hat{p}(\hat{\imath}_j,j)}{\hat{p}(i,j)} \geq \hat{p}(i,j) \cdot \max_j r(i,j). \tag{5}$$

Combining Inequalities (4) and (5) we get

$$\texttt{MS}(\hat{\mathbf{p}},\hat{\mathbf{x}}) \geq \frac{\sum_{i:\max_j r(i,j)>1} \max_j r(i,j) \sum_{j\in J_i} \hat{p}(i,j)}{n}.$$

By Lemma 16 we know that for any machine i such that $\max_j r(i,j) > 1$, $\sum_{j\in J_i} \hat{p}(i,j) \geq \gamma \texttt{MS}(\hat{\mathbf{p}},\hat{\mathbf{x}})$. Substitute it in the equation above we get that:

$$\texttt{MS}(\hat{\mathbf{p}},\hat{\mathbf{x}}) \geq \frac{\sum_{i:\max_j r(i,j)>1} \max_j r(i,j) \cdot \gamma \texttt{MS}(\hat{\mathbf{p}},\hat{\mathbf{x}})}{n} \quad \Rightarrow \quad \sum_{i:\max_j r(i,j)>1} \max_j r(i,j) \leq \frac{n}{\gamma}. \blacktriangleleft$$

We are now ready to show the robustness of the algorithm.

▶ **Lemma 18.** *For any $\gamma > 0$, ScaledGreedy is a mechanism that is $(1+\frac{1}{\gamma})n$-robust.*

Proof. First note that since $\texttt{OPT}$ is the maximum finishing time of any machine in the optimal schedule, and since the maximum is greater than the average we have:

$$\texttt{OPT} \geq \frac{\sum_{j\in M} p(i^\star_j,j)}{n}. \tag{6}$$

We now partition the set of jobs M into two subsets, let S^* be the set of jobs j such that $r(i^\star_j,j) = 1$ and $M \setminus S^*$ be the rest of the jobs j with $r(i^\star_j,j) > 1$.

For any job $j \in S^*$, let i_j be the machine the mechanism assigns j on, i.e., $x(i_j,j) = 1$. we have

$$p(i_j,j) \leq r(i,j)p(i_j,j) \leq r(i^\star_j,j)p(i^\star_j,j) = p(i^\star_j,j), \tag{7}$$

where the first inequality is by Observation 10, the second inequality is by the fact that $x(i_j, j) = 1$ and the last equality is by definition of S^*. Combining Inequalities (6) and (7) we get

$$\sum_{j \in S^*} p(i_j, j) \leq \sum_{j \in S^*} p(i_j^\star, j) \leq \sum_{j \in M} p(i_j^\star, j) \leq n \cdot \texttt{OPT}. \tag{8}$$

Now consider any job $j \in M \setminus S^*$, first note that since $r(i_j^\star, j) > 1$, $i_j^\star \in \{i : \max_j r(i, j) > 1\}$. Similarly as the analysis of S^* we have

$$p(i_j, j) \leq r(i, j)p(i_j, j) \leq r(i_j^\star, j)p(i_j^\star, j) \tag{9}$$

Now consider the optimal makespan $\texttt{OPT}$ again, by the definition of optimal schedule, we have:

$$\texttt{OPT} = \max_i \sum_{j : i_j^\star = i} p(i_j^\star, j) \geq \max_{i : \max_j r(i,j) > 1} \sum_{j : i_j^\star = i} p(i_j^\star, j). \tag{10}$$

Combining Inequalities (9) and (10), we get:

$$\begin{aligned}
\sum_{j \in M \setminus S^*} p(i_j, j) &\leq \sum_{j \in M \setminus S^*} r(i_j^\star, j)p(i_j^\star, j) && \text{(by Inequality (9))} \\
&\leq \sum_{i : \max_j r(i,j) > 1} \sum_{j : i_j^\star = i} r(i_j^\star, j)p(i_j^\star, j) \\
&\leq \sum_{i : \max_j r(i,j) > 1} (\max_j r(i, j)) \cdot \sum_{j : i_j^\star = i} p(i_j^\star, j) \\
&\leq \sum_{i : \max_j r(i,j) > 1} (\max_j r(i, j))\texttt{OPT} && \text{(by Inequality (10))} \\
&\leq \frac{n}{\gamma} \cdot \texttt{OPT}, && (11)
\end{aligned}$$

where the last inequality is due to Lemma 17. Combining Inequalities (8) and (11) we get:

$$\max_i \sum_{j \in M_i} p(i, j) \leq \sum_{j \in M} p(i_j, j) \leq \sum_{j \in S^*} p(i_j, j) + \sum_{j \in M \setminus S^*} p(i_j, j) \leq \left(1 + \frac{1}{\gamma}\right) n \cdot \texttt{OPT}. \quad \blacktriangleleft$$

Strategyproofness.

▶ **Lemma 19.** *ScaledGreedy is a strategyproof mechanism.*

Proof. Consider a machine $i \in N$ and a predicted assignment $\hat{\mathbf{x}}$. Consider two instances $\mathbf{p}$ and $\mathbf{p}'$ that differ only on machine i and the associated allocations $\mathbf{x}$ and $\mathbf{x}'$ returned by ScaledGreedy when the predicted assignment is $\hat{\mathbf{x}}$. Given a fixed predicted assignment $\hat{\mathbf{x}}$, the mechanism is deterministic, so $x(i, j) \in \{0, 1\}$ for all $i \in N$ and $j \in M$. Note that line 4 to 14 don't use $\mathbf{p}$, thus the scalars are, as for SimpleScaledGreedy, independent of $\mathbf{p}$. The remaining of the proof follows identically as the analysis of SimpleScaledGreedy being strategyproof (Lemma 8) ◀

The main result for ScaledGreedy. Combining Lemmas 14, 18, and 19, we obtain the main result for ScaledGreedy, which is that it is a strategyproof mechanism that, with input parameter γ being any constant, achieves constant consistency and linear robustness.

▶ **Theorem 20.** *For any* $\gamma \in \left(0, \frac{n}{2} - 1\right)$, ScaledGreedy *is a strategyproof,* $(2+\gamma)\alpha$*-consistent, and* $\left(1 + \frac{1}{\gamma}\right)$ *n-robust algorithm.*

To obtain a polynomial-time algorithm, we can have $\alpha = 2$ and a consistency of $4 + 2\gamma$ with Makespan-Min being the 2-approximation algorithm for makespan minimization on unrelated machines by Lenstra et al [21]. By ignoring running time consideration, we can obtain $\alpha = 1$ and a consistency of $2 + \gamma$ with Makespan-Min finding an optimal schedule.

5 The Error Tolerant Scaled Greedy Mechanism

In this section, we give a mechanism that builds on the mechanism from the previous section and achieves a constant approximation not only when the predictions are accurate, but also when the predictions are approximately accurate. Recall from Section 2 that, given a prediction $\hat{\mathbf{p}}$ of the actual instance $\mathbf{p}$, the prediction error η is the largest ratio between the predicted processing time and actual processing time for any i, j pair, i.e., $\eta = \max_{i \in M, j \in N} \max\left\{\frac{\hat{p}_{ij}}{p_{ij}}, \frac{p_{ij}}{\hat{p}_{ij}}\right\}$. This mechanism takes as input an error tolerance parameter $\bar{\eta} > 0$ and achieves an approximation of $(2+\gamma)\alpha\eta^2$ if $\eta \leq \bar{\eta}$ and $(1 + \frac{1}{\gamma})\bar{\eta}^2 n$ otherwise, where, similarly as in the previous section, γ is the prediction confidence parameter and α is the approximation of the scheduling algorithm used in the mechanism.

The mechanism. ErrorTolerantScaledGreedy, formally described in Mechanism 3, is similar to ScaledGreedy, but takes as input an additional parameter $\bar{\eta} > 0$ that is the error tolerance of the mechanism. Recall from the previous section that if $j \in J_i$ for some machine i, then, when the predictions are correct, ScaledGreedy assigns j to machine i, otherwise it assigns j to its predicted assignment $\hat{\imath}_j$. ErrorTolerantScaledGreedy aims to achieve this assignment not only when the predictions are correct, but also when they are approximately correct, with one exception that is later discussed. In order to achieve this assignment when the predictions are approximately correct, the mechanism sets the scalars $r(i, j)$ of pairs i, j such that job j that should be assigned to i to $r(i, j) = \frac{1}{\bar{\eta}^2}$ in lines 12 and 16. The exception previously mentioned is that it is not only the scalar of $j^\star \in J_{i^\star}$ that is updated to $\frac{1}{\bar{\eta}^2}$ in line 12, but the scalars $r(i, j^\star)$ for all machines i such that $\hat{p}(i^*, j^*) \leq \hat{p}(i, j^*)$. The remainder of the mechanism is identical to ScaledGreedy.

The analysis. Note that mechanisms ScaledGreedy and ErrorTolerantScaled-Greedy are identical except lines 12, 15 and 16. It is easy to verify that Lemma 15, Lemma 16, and Lemma 17 for ScaledGreedy also hold for ErrorTolerantScaled-Greedy. The Observation 10 also holds true for ErrorTolerantScaledGreedy for any $r(i, j)$ given that $i \neq \hat{\imath}_j$ and $j \notin J_i$. We first show that the characterization of the jobs in J_i as $J_i = M_i \setminus \hat{M}_i$ for ScaledGreedy when the predictions are accurate now also holds for ErrorTolerantScaledGreedy when the error is bounded by the error tolerance, i.e., $\eta(\mathbf{p}, \hat{\mathbf{p}}) \leq \bar{\eta}$.

▶ **Lemma 21.** *Given any prediction* $\hat{\mathbf{p}}$ *and actual instance* $\mathbf{p}$*, if* $\eta(\mathbf{p}, \hat{\mathbf{p}}) \leq \bar{\eta}$*, then, for every machine* $i \in N$*, we have* $J_i = M_i \setminus \hat{M}_i$*.*

Proof. By Inequality (1), (2) and (3) from the proof of Lemma 12 we know that if $j \in J_i$ for some machine i', $\hat{p}(i', j) \leq \hat{p}(i, j) \cdot r(i, j)$ for any i, j pair. By line 12 of the mechanism and the fact that the mechanism did not modify $r(i, j)$ for any other machine $i \notin \{i', \hat{\imath}_j\}$, we have for any $i \notin \{i', \hat{\imath}_j\}$:

Mechanism 3 ErrorTolerantScaledGreedy.

1 **Input:** instance $\mathbf{p} \in \mathbb{R}^{n\times m}$, predicted instance $\hat{\mathbf{p}} \in \mathbb{R}^{n\times m}$, scheduling algorithm Makespan-Min, prediction confidence parameter $\gamma \in (0, \frac{n}{2} - 1)$, error tolerance parameter $\bar{\eta} > 0$
2 $\hat{\mathbf{x}} \leftarrow$ Makespan-Min$(\hat{\mathbf{p}})$
3 $\hat{\imath}_j \leftarrow$ the machine i that job j is assigned to according to $\hat{\mathbf{x}}$, i.e., $\hat{x}(\hat{\imath}_j, j) = 1$, for each $j \in M$
4 $r(i,j) \leftarrow \max\left(\frac{\hat{p}(\hat{\imath}_j,j)}{\hat{p}(i,j)}, 1\right)$ for each $(i,j) \in N \times M$
5 $J_i \leftarrow \emptyset$ for all $i \in N$
6 $I \leftarrow N$
7 $T \leftarrow \{(i,j) : j \in M, i = \arg\min_{i' \in I:\hat{p}(i',j)<\hat{p}(\hat{\imath}_j,j)} \hat{p}(i',j)\}$
8 **while** *T is not empty* **do**
9 $\quad (i^*, j^*) \leftarrow \arg\max_{(i,j)\in T} \frac{\hat{p}(\hat{\imath}_j,j)}{\hat{p}(i,j)}$
10 $\quad J_{i^*} \leftarrow J_{i^*} \cup \{j^*\}$
11 $\quad$ **for** *all i s.t. $\hat{p}(i^*, j^*) \le \hat{p}(i, j^*)$* **do**
12 $\quad\quad$ update $r(i, j^*) \leftarrow \frac{1}{\bar{\eta}^2}$
13 $\quad I \leftarrow \{i \in N : \sum_{j\in J_i} \hat{p}(i,j) < \gamma \texttt{MS}(\hat{\mathbf{p}}, \hat{\mathbf{x}})\}$
14 $\quad T \leftarrow \{(i,j) : j \in M \setminus (\cup_i J_i), i = \arg\min_{i' \in I:\hat{p}(i',j)<\hat{p}(\hat{\imath}_j,j)} \hat{p}(i',j)\}$
15 **for** *all jobs j s.t $j \notin \cup_{i\in N} J_i$* **do**
16 $\quad$ update $r(\hat{\imath}_j, j) \leftarrow \frac{1}{\bar{\eta}^2}$
17 $i_j \leftarrow \arg\min_i r(i,j) \cdot p(i,j)$, break ties in favor of $\hat{\imath}_j$ first and i' if $j \in J_{i'}$ second, for each $j \in M$
18 **if** $i = i_j$ **then** $x(i,j) = 1$ **else** $x(i,j) = 0$, for each $(i,j) \in N \times M$
19 **return x**

$$p(i',j) \cdot r(i',j) \le \eta \hat{p}(i',j) \cdot r(i',j) = \eta \hat{p}(i',j) \cdot 1/\bar{\eta}^2 = \hat{p}(i',j)\frac{\eta}{\bar{\eta}^2} \le \hat{p}(i',j) \cdot 1/\bar{\eta}$$
$$\le \hat{p}(i,j) \cdot r(i,j) \cdot 1/\bar{\eta} \le p(i,j) \cdot r(i,j) \cdot \frac{\eta}{\bar{\eta}} \le p(i,j) \cdot r(i,j),$$

Where the first inequality is by the definition of η, the second inequality is due to $\eta \le \bar{\eta}$, the third inequality is since $r(i,j) \ge 1$ by Observation 10, and the forth inequality is again by the definition of η. We also have:

$$p(i'j) \cdot r(i',j) = \hat{p}(i,j) \cdot \frac{1}{\bar{\eta}^2} < \hat{p}(\hat{\imath}_j,j) \cdot \frac{1}{\bar{\eta}^2} = p(\hat{\imath}_j,j) \cdot r(\hat{\imath}_j,j)$$

Since $p(i',j) \cdot r(i',j) \le p(i,j) \cdot r(i,j)$ and $p(i'j) \cdot r(i',j) < p(\hat{\imath}_j,j) \cdot r(\hat{\imath}_j,j)$, by line 17, we have that $x(i',j) = 1$ in ErrorTolerantScaledGreedy. Following a similar argument we get for any job j such that $j \notin \cup_{i\in N} J_i$, we have $x(\hat{\imath}_j, j) = 1$ in ErrorTolerantScaledGreedy. ◀

Given that $J_i = M_i \setminus \hat{M}_i$, we have Lemma 13 holds true. The approximation obtained by ErrorTolerantScaledGreedy when the error is within the error tolerance is $(2 + \gamma)\alpha\eta^2$. The analysis of this approximation is similar to the analysis of the consistency of ScaledGreedy.

▶ **Lemma 22.** *Given an error tolerance bound $\bar{\eta}$, and an actual error η, the approximation ratio of mechanism* ErrorTolerantScaledGreedy *is $(2+\gamma)\alpha\eta^2$ if $\eta \leq \bar{\eta}$.*

Proof. By Lemma 21, when $\eta(\mathbf{p}, \hat{\mathbf{p}}) \leq \bar{\eta}$, the assignment $\mathbf{x}$ returned by the mechanism over input $(\mathbf{p}, \hat{\mathbf{p}}, \text{Makespan-Min}, \gamma, \bar{\eta})$ is identical to the assignment returned by ScaledGreedy over input $(\mathbf{p}, \mathbf{p}, \text{Makespan-Min}, \gamma)$. By Lemma 13 we have for any machine i

$$\begin{aligned}
\sum_{j \in M_i} \hat{p}(i,j) \cdot x(i,j) &= \sum_{j \in M_i \cap \hat{M}_i} \hat{p}(i,j) + \sum_{j \in M_i \setminus \hat{M}_i} \hat{p}(i,j) \\
&= \sum_{j \in M_i \cap \hat{M}_i} \hat{p}(i,j) + \sum_{j \in J_i} \hat{p}(i,j) && \text{(by Lemma 21)} \\
&\leq \mathtt{MS}(\hat{\mathbf{p}}, \hat{\mathbf{x}}) + (1+\gamma)\mathtt{MS}(\hat{\mathbf{p}}, \hat{\mathbf{x}}) && \text{(by Lemma 13)} \\
&\leq (2+\gamma)\mathtt{MS}(\hat{\mathbf{p}}, \hat{\mathbf{x}}) \\
&\leq (2+\gamma)\alpha\mathtt{MS}(\hat{\mathbf{p}}, \hat{\mathbf{x}}^\star). && (12)
\end{aligned}$$

where $\hat{\mathbf{x}}^\star$ is optimal schedule for the predicted instance and the last inequality is by the fact that Makespan-Min is a α-approximation algorithm. Let $\mathbf{x}^\star$ be the optimal schedule for the the actual instance $\mathbf{p}$, we have:

$$\mathtt{MS}(\hat{\mathbf{p}}, \hat{\mathbf{x}}^\star) \leq \mathtt{MS}(\hat{\mathbf{p}}, \mathbf{x}^\star).$$

Now again given the definition of error η we have:

$$\mathtt{MS}(\hat{\mathbf{p}}, \hat{\mathbf{x}}^\star) \leq \mathtt{MS}(\hat{\mathbf{p}}, \mathbf{x}^\star) = \max_i \sum_{j \in M} \hat{p}(i,j) \cdot x^\star(i,j) \leq \max_i \sum_{j \in M} \eta \cdot p(i,j) \cdot x^\star(i,j) = \eta \cdot \mathtt{OPT}. \tag{13}$$

Combining Inequalities (12) and (13) we get:

$$\max_i \sum_{j \in M} p(i,j) \cdot x(i,j) \leq \max_i \sum_{j \in M} \eta \hat{p}(i,j) \cdot x(i,j) \leq (2+\gamma)\alpha\eta\mathtt{MS}(\hat{\mathbf{p}}, \hat{\mathbf{x}}^\star) \leq (2+\gamma)\alpha\eta^2\mathtt{OPT}. \blacktriangleleft$$

The approximation obtained by ErrorTolerantScaledGreedy when the error is not within the error tolerance is $(1+\frac{1}{\gamma})\bar{\eta}^2 n$. The analysis of this approximation is similar to the analysis of the robustness of ScaledGreedy.

▶ **Lemma 23.** *Given an error tolerance bound $\bar{\eta}$, the approximation ratio of mechanism* ErrorTolerantScaledGreedy *is $(1+\frac{1}{\gamma})\bar{\eta}^2 n$.*

Proof. We again partition the set of jobs M into two subset, let S^* be the set of jobs j such that $r(i_j^\star, j) \leq 1$ and $M \setminus S^*$ be the rest of the jobs j with $r(i_j^\star, j) > 1$. Since ErrorTolerantScaledGreedy only modify some scalar to $\frac{1}{\bar{\eta}^2}$, combining with Observation 10 we have: $r(i,j) \geq \frac{1}{\bar{\eta}^2}$ for any i, j. Next,

$$\frac{1}{\bar{\eta}^2} p(i_j, j) \leq r(i,j) p(i_j, j) \leq r(i_j^\star, j) p(i_j^\star, j) \leq p(i_j^\star, j), \tag{14}$$

where the second inequality is by the fact that $x(i_j, j) = 1$, and the last equality is by definition of S^*. We get that

$$\sum_{j \in S^*} p(i_j, j) \leq \bar{\eta}^2 \sum_{j \in S^*} p(i_j^\star, j) \leq \bar{\eta}^2 \sum_{j \in M} p(i_j^\star, j) \leq \bar{\eta}^2 n \cdot \mathtt{OPT}. \tag{15}$$

where the first inequality is by Equation (14) and the last inequality is by Inequality (6) from Lemma 18. Now consider any job $j \in M \setminus S^*$. Similarly as the analysis of S^* we have

$$\frac{1}{\bar{\eta}^2} p(i_j, j) \leq r(i, j) p(i_j, j) \leq r(i_j^\star, j) p(i_j^\star, j).$$

Next, we have:

$$\sum_{j \in M \setminus S^*} p(i_j, j) \leq \bar{\eta}^2 \sum_{j \in M \setminus S^*} r(i_j^\star, j) p(i_j^\star, j) \leq \bar{\eta}^2 \frac{n}{\gamma} \cdot \texttt{OPT}, \tag{16}$$

where the second inequality is by Inequality (11). Combining (15) and (16) we get:

$$\max_i \sum_{j \in M_i} p(i, j) \leq \sum_{j \in M} p(i_j, j) \leq \sum_{j \in S^*} p(i_j, j) + \sum_{j \in M \setminus S^*} p(i_j, j) \leq \left(1 + \frac{1}{\gamma}\right) \bar{\eta}^2 n \cdot \texttt{OPT}. \quad \blacktriangleleft$$

The proof for the strategyproofness ErrorTolerantScaledGreedy is identical as the proof for the strategyproofness of ScaledGreedy(Lemma 19). Together with Lemma 22 and Lemma 23, we get the main result for ErrorTolerantScaledGreedy.

▶ **Theorem 24.** *For any $\gamma \in \left(0, \frac{n}{2} - 1\right)$ and error tolerance bound $\bar{\eta} > 0$, ErrorTolerantScaledGreedy is a strategyproof mechanism that achieves an approximation of $(2 + \gamma)\alpha\eta^2$ if $\eta \leq \bar{\eta}$ and $\left(1 + \frac{1}{\gamma}\right)\bar{\eta}^2 n$ otherwise, where η is the prediction error.*

Thus, with any constants $\gamma, \bar{\eta}$, and with Makespan-Min being the 2-approximation scheduling algorithm from [21], we obtain a polynomial-time mechanism that achieves an $O(1)$-approximation when the prediction error η is such that $\eta \leq \bar{\eta}$, and an $O(n)$-approximation otherwise.

6 Perfect Consistency Implies Unbounded Robustness

Recall that every strategyproof scheduling mechanism must be monotone. Below is a very useful lemma that comes immediately from the monotonicity property and is used in most of the lower bound proofs.

▶ **Lemma 25** ([12]). *Consider instances $\mathbf{p}$ and $\mathbf{p}'$ and let $\mathbf{x}$ and $\mathbf{x}'$ be the allocation produced by a strategyproof scheduling mechanism for the given two instances, respectively. If $\mathbf{p}$ and $\hat{\mathbf{p}}$ only differs in the processing time of machine i in such a way that $p'(i, j) > p(i, j)$ when $x(i, j) = 0$, and $p'(i, j) < p(i, j)$ when $x(i, j) = 1$, then $\mathbf{x}' = \mathbf{x}$.*

▶ **Theorem 26.** *No deterministic strategyproof scheduling mechanism with 1-consistency can achieve any bounded robustness.*

Proof. Consider the the predicted instance $\hat{\mathbf{p}}$ instance with 2 machines and 2 jobs, and the processing time as in the table on the left of Figure 1. Consider an actual instance that is the same as the prediction, i.e., $\mathbf{p} = \hat{\mathbf{p}}$.

By 1-consistency, the mechanism needs to output $x_{11} = x_{22} = 1$ (the allocation is marked with $\star$). Now consider another instance $\mathbf{p}'$ represent by table on the right of Figure 1 and consider machine 1, since $p'(1, 1) < p(1, 1)$and $x(1, 1) = 1$ and $p'(1, 2) > p(1, 2)$ and $x(1, 2) = 1$, by Lemma 25, any truthful mechanism needs to output $x'(1, 1) = x(1, 1) = 1$ and $x'(1, 2) = x(1, 2) = 0$, resulting a makespan at lest K. Since the optimal makespan of $\mathbf{p}'$ is $1 + \epsilon$ by assigning both jobs to machine 1, it means the robustness of any strategyproof mechanism is at most K. Since K can be arbitrarily large, the robustness is therefore unbounded. $\blacktriangleleft$

m\j	1	2
1	$K^\star$	1
2	∞	$K^\star$

$\rightarrow$

m\j	1	2
1	$0^\star$	$1+\epsilon$
2	∞	$K^\star$

Figure 1 The predicted (on the left) and actual (on the right) instances showing any deterministic 1-consistent mechanism suffers unbounded robustness.

References

1 Priyank Agrawal, Eric Balkanski, Vasilis Gkatzelis, Tingting Ou, and Xizhi Tan. Learning-augmented mechanism design: Leveraging predictions for facility location. In David M. Pennock, Ilya Segal, and Sven Seuken, editors, *EC '22: The 23rd ACM Conference on Economics and Computation, Boulder, CO, USA, July 11 - 15, 2022*, pages 497–528. ACM, 2022.

2 Antonios Antoniadis, Themis Gouleakis, Pieter Kleer, and Pavel Kolev. Secretary and online matching problems with machine learned advice. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, pages 7933–7944, 2020.

3 Itai Ashlagi, Shahar Dobzinski, and Ron Lavi. Optimal lower bounds for anonymous scheduling mechanisms. *Math. Oper. Res.*, 37(2):244–258, 2012.

4 Yossi Azar, Debmalya Panigrahi, and Noam Touitou. Online graph algorithms with predictions. *Proceedings of the Thirty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, 2022.

5 Eric Balkanski, Tingting Ou, Clifford Stein, and Hao-Ting Wei. Scheduling with speed predictions. *CoRR*, abs/2205.01247, 2022. `arXiv:2205.01247`.

6 Étienne Bamas, Andreas Maggiori, Lars Rohwedder, and Ola Svensson. Learning augmented energy minimization via speed scaling. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

7 Etienne Bamas, Andreas Maggiori, and Ola Svensson. The primal-dual method for learning augmented algorithms. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, pages 20083–20094, 2020.

8 Evripidis Bampis, Konstantinos Dogeas, Alexander V. Kononov, Giorgio Lucarelli, and Fanny Pascual. Scheduling with untrusted predictions. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 4581–4587. ijcai.org, 2022.

9 Siddhartha Banerjee, Vasilis Gkatzelis, Artur Gorokh, and Billy Jin. Online nash social welfare maximization with predictions. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*. SIAM, 2022.

10 George Christodoulou, Elias Koutsoupias, and Annamária Kovács. Mechanism design for fractional scheduling on unrelated machines. *ACM Trans. Algorithms*, 6(2):38:1–38:18, 2010.

11 George Christodoulou, Elias Koutsoupias, and Annamária Kovács. On the nisan-ronen conjecture. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 839–850. IEEE, 2021.

12 George Christodoulou, Elias Koutsoupias, and Angelina Vidali. A lower bound for scheduling mechanisms. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, 2007.

13 George Christodoulou, Elias Koutsoupias, and Angelina Vidali. A lower bound for scheduling mechanisms. *Algorithmica*, 55(4):729–740, 2009.

14 Shahar Dobzinski and Ariel Shaulker. Improved lower bounds for truthful scheduling. *CoRR*, abs/2007.04362, 2020. `arXiv:2007.04362`.

15 Paul Dütting, Silvio Lattanzi, Renato Paes Leme, and Sergei Vassilvitskii. Secretaries with advice. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 409–429, 2021.

16 Yiannis Giannakopoulos, Alexander Hammerl, and Diogo Poças. A new lower bound for deterministic truthful scheduling. In Tobias Harks and Max Klimm, editors, *Algorithmic Game Theory - 13th International Symposium, SAGT 2020, Augsburg, Germany, September 16-18, 2020, Proceedings*, volume 12283 of *Lecture Notes in Computer Science*, pages 226–240. Springer, 2020.

17 Vasilis Gkatzelis, Kostas Kollias, Alkmini Sgouritsa, and Xizhi Tan. Improved price of anarchy via predictions. In David M. Pennock, Ilya Segal, and Sven Seuken, editors, *EC '22: The 23rd ACM Conference on Economics and Computation, Boulder, CO, USA, July 11 - 15, 2022*, pages 529–557. ACM, 2022.

18 Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. Online knapsack with frequency predictions. *Advances in Neural Information Processing Systems*, 34, 2021.

19 Elias Koutsoupias and Angelina Vidali. A lower bound of 1+*phi* for truthful scheduling mechanisms. In Ludek Kucera and Antonín Kucera, editors, *Mathematical Foundations of Computer Science 2007, 32nd International Symposium, MFCS 2007, Ceský Krumlov, Czech Republic, August 26-31, 2007, Proceedings*, volume 4708 of *Lecture Notes in Computer Science*, pages 454–464. Springer, 2007.

20 Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1859–1877. SIAM, 2020.

21 Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46:259–271, 1990.

22 Shi Li and Jiayi Xian. Online unrelated machine load balancing with predictions revisited. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 6523–6532. PMLR, 2021.

23 Alexander Lindermayr and Nicole Megow. Alps. URL: `https://algorithms-with-predictions.github.io/`.

24 Thodoris Lykouris and Sergei Vassilvtiskii. Competitive caching with machine learned advice. In *International Conference on Machine Learning*, pages 3296–3305. PMLR, 2018.

25 Michael Mitzenmacher. Scheduling with predictions and the price of misprediction. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 14:1–14:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

26 Michael Mitzenmacher and Sergei Vassilvitskii. *Algorithms with Predictions*, pages 646–662. Cambridge University Press, 2021. `doi:10.1017/9781108637435.037`.

27 Noam Nisan and Amir Ronen. Algorithmic mechanism design (extended abstract). In Jeffrey Scott Vitter, Lawrence L. Larmore, and Frank Thomson Leighton, editors, *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*, pages 129–140. ACM, 1999.

28 Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games Econ. Behav.*, 35(1-2):166–196, 2001.

29 Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ml predictions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018.

30 Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ML predictions. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9684–9693, 2018.

31 Michael E. Saks and Lan Yu. Weak monotonicity suffices for truthfulness on convex domains. In John Riedl, Michael J. Kearns, and Michael K. Reiter, editors, *Proceedings 6th ACM Conference on Electronic Commerce (EC-2005), Vancouver, BC, Canada, June 5-8, 2005*, pages 286–293. ACM, 2005.

32 Chenyang Xu and Pinyan Lu. Mechanism design with predictions. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 571–577. ijcai.org, 2022. doi:10.24963/ijcai.2022/81.

A Missing Analysis from Section 3

▶ **Lemma 7.** *SimpleScaledGreedy is a mechanism that is $\Theta(n^2)$-robust.*

Proof. We first show that the robustness is at most n^2. Consider an instance $\mathbf{p}$, an optimal assignment $\mathbf{x}^\star$ for $\mathbf{p}$, a job j, and a machine i such that $j \in M_i$. Let $i_j^\star$ be the machine that j is assigned to according to $\mathbf{x}^\star$. By the mechanism, we have $1 \le r(i', j) \le n$ for all machines i'. We therefore have:

$$p(i,j) \le r(i,j) \cdot p(i,j) \le r(i_j^\star, j) \cdot p(i_j^\star, j) \le n \cdot p(i_j^\star, j),$$

where the second inequality is by line 5 and the fact that j is assigned to i. We get that

$$\texttt{MS}(\mathbf{p}, \mathbf{x}) = \max_{i \in N} \sum_{j \in M_i} p(i,j) \le \sum_{i \in N} \sum_{j \in M_i} p(i,j) \le n \sum_{i \in N} \sum_{j \in M_i} p(i_j^\star, j) \le n^2 \max_{i \in N} \sum_{j \in M_i} p(i_j^\star, j) = n^2 \texttt{OPT}.$$

Consider the predicted instance in Figure 2 (a).

m\j	1	2	…	$n-1$	n	$n+1$	…	$2n-2$
1	1				$n(n-1)$			
2		1				$n(n-1)$		
…			1				$n(n-1)$	
$n-1$				1				$n(n-1)$
n	n	n	n	n				

(a)

m\j	1	2	…	$n-1$	n	$n+1$	…	$2n-2$
1	$1+\epsilon$				0	0	0	0
2		$1+\epsilon$			0	0	0	0
…			$1+\epsilon$		0	0	0	0
$n-1$				$1+\epsilon$	0	0	0	0
n	n	n	n	n				

(b)

Figure 2 (a) Predicted instance $\hat{\mathbf{p}}$, empty entry means $\hat{p}(i,j) = \infty$. (b) Actual instance $\mathbf{p}$, empty entry means $p(i,j) = \infty$.

Assume that Makespan-Min computes the optimal makespan for the predicted instance. We have $\hat{x}(n,i) = 1$ for $i \in [1, n-1]$. Then the mechanism would set $r(i,i) = \frac{1}{n}$ for any $i \in [1, n-1]$. Now consider the actual instance in Figure 2(b), note that for any $i \in [1, n-1]$, we have:

$$r(i,i) \cdot p(i,i) = n \cdot (1+\epsilon) > n = r(n,i) \cdot r(n,i).$$

Since machine n has a lower scaled processing time for job $i \in [1, n]$, the mechanism would assign all such jobs to machine n, leading to a makespan of $n(n-1)$, whereas the optimal makespan is $1+\epsilon$. ◀