The Economic Impacts of Generative AI on the Structure of Work

Mert Demirer
MIT

John J. Horton
MIT

Nicole Immorlica
Yale & Microsoft

Brendan Lucier
Microsoft

Peyman Shahidi*
MIT

Preliminary Draft

Click Here for the Most Recent Version

November 1, 2025

Abstract

We introduce a task-based model of production that responds to automation technologies such as AI. Production is a sequence of *steps* that firms aggregate into *tasks* and then into *jobs*, each performed by a single worker. A firm can reduce wage costs of jobs by assigning each worker a narrower set of tasks, thereby increasing specialization, but doing so introduces coordination frictions among a larger number of workers. The optimal organizational structure resulting from aggregations must balance these opposing forces. Production steps may be executed *manually* by humans, *augmented* with AI assistance under human oversight, or *automated* within an AI chain, where multiple consecutive steps are executed by AI and only the final output is verified by a human. A key implication of chaining is that it can overturn simple comparative-advantage logic: a step where a human has a comparative advantage over AI in isolation may nonetheless be optimally automated if it sits adjacent to strong AI-performing steps. Productivity gains therefore depend not only on AI quality but also on where AI-suitable steps are positioned within the workflow. We characterize how firms optimally allocate steps between humans and AI both in the short run, when job boundaries are fixed, and in the long run, when automation and job design are jointly chosen. This framework yields predictions about task- versus job-level exposure to AI, worker skill and specialization, and the non-linear productivity gains that arise when marginal improvements in AI quality trigger discrete reorganizations of work. Finally, we show how the production functions derived from firm-level cost minimization can be mapped into a CES production function at the aggregate level to study the macroeconomic implications of improving AI quality.

1 Introduction

Recent advances in Artificial Intelligence (AI) carry the potential to significantly impact work processes and economic productivity. Questions about the impact of automation technology on production and labor

*Emails: Demirer: mdemirer@mit.edu; Horton: jjhorton@mit.edu; Immorlica: nicimm@microsoft.com; Lucier: brlucier@microsoft.com; Shahidi: peymansh@mit.edu.

are not new, but the rise of generative AI is notable for the breadth of occupational tasks that it might influence, including traditionally white-color knowledge work Eloundou et al. (2023). As generative AI and language models see rapid adoption in the workplace, multiple effects—both local and global—can arise. In the short term, we might expect efficiency gains from individual task-level improvements and partial automation of certain production steps, increasing worker productivity. In the long term, however, it is tempting to imagine larger disruptions to labor as AI usage becomes more widespread. The proliferation of AI tools could potentially enable reorganization of tasks, job responsibilities, and worker skill requirements, with substantial implications for the labor market.

To explore potential long-run impacts of new AI technology, we treat both the definition of worker tasks and the partitioning of tasks into job responsibilities as endogenous market outcomes. A firm seeks to optimize its cost of production by choosing which steps of its production sequence to automate, which steps to assign to human workers, and how to split human labor into jobs with different skill requirements. For example, in the absence of effective automation technology, a firm might hire multiple high-skill workers in different job roles to accomplish distinct steps of its production process, incurring hand-off costs whenever work passes from one worker to another. As AI technology improves, the firm might instead choose to automate many of these steps—lumping them together into one logical task—and then hire a single lower-skill worker to oversee this automation and validate the production outcome. Whether this strategy is effective depends not only on the effectiveness of the AI technology at accomplishing individual production steps, but on the sequential relationship between steps, the relative ease of verification versus manual execution, and hand-off costs between workers.

One might naturally wonder what is special about AI technology in this scenario, relative to other forms of automation. An important feature of AI is its generality: its potential to influence many different tasks, albeit disproportionately. As such, an improvement in AI technology can simultaneously impact automation decisions across multiple points in a production chain, leading to large shifts in overall production strategy. By endogenizing the organization of worker tasks and job assignments, we seek to capture non-local effects that might depend on the sequencing of production steps and their relation to each other.

Table 1: Summary of AI's Impact in Short and Long Run

Time Horizon	**Job Design**	**Worker Skills and Wages**	**Automation Strategy**	**Comments**
Short-run	Fixed	Fixed	Optimized within jobs	Productivity gains by automating existing tasks
Long-run	Flexible	Skills and relative wages adjust to job design	Optimized across jobs	Joint optimization of automation and job design to minimize total labor costs

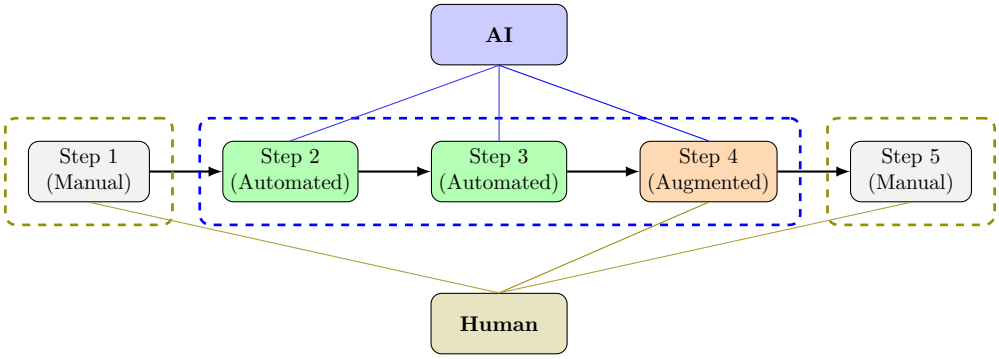
We use our model of automation and labor assignment to explore the effect of AI and the resulting automation of tasks. We investigate how AI advancements affect this division in the short-run—where human capital needs remain fixed but AI is used to increase worker productivity by reducing the time it takes to complete tasks—and in the long-run—where both human capital requirements and firm automation strategies can jointly adjust.

Automation versus Augmentation. Central to our model is the distinction between full automation of a production step versus worker augmentation. In our model, three modes of step completion are recognized: manual, augmented, and automated. *Manual* completion of a step means that a human carries out the work without any AI assistance. *Augmented* completion involves a human performing the step with the use of AI, which might (or might not) reduce labor costs. For example, a human describes the parameters of a step to an AI tool, the AI then executes the step, and its output is reviewed and approved by the human. In contrast, we say a step has been *automated* if it is completed by an AI end-to-end without any direct human intervention. We view automation as a way of putting certain production steps "under the hood" of a logical meta-task that a worker is attempting to complete. For example, as part of an AI-augmented request to complete one step (such as writing a report), an AI might also be assigned to complete one or more

preceding steps (such as running a statistical analysis) as part of that single request. Any such preceding step is said to be automated, and the set of all such automated steps, plus the final step that the worker is actively engaged with, can be thought of as a single worker task. By choosing which steps to automate and which to assign either manually or via augmentation, the firm implicitly designs the set of tasks exposed to its workers.

Although augmented and automated production steps both involve AI, they differ in one crucial respect: AI augmentation demands direct human oversight of the AI's output, whereas automation does not. Figure 1 illustrates these concepts using a sequence of five steps. Steps 1 and 5 are performed manually by a human. Steps 2, 3, and 4 form an AI chain and are done using AI. Steps 2 and 3 are automated because their outputs feed directly into subsequent AI tasks without human review. Steps 4 is augmented because its output is evaluated by a human before proceeding. The resulting worker tasks in this job thus become $1, 4$ and 5.

Figure 1: Illustrative Example for Division of Labor



Notes: This figure illustrates division of labor across five steps: Steps 1 and 5 are manual; Steps 2 and 3 are AI-automated (done without direct human intervention); Step 4 is AI-augmented (done with AI but requires human oversight). Lines from the AI box show which steps involve AI, and lines from the Human box indicate steps requiring human execution or oversight. Dashed boxes group steps according to their task assignments. Steps 1 and 5 are human tasks while Steps 2-4 form an AI chain task.

The decision to deploy AI on a given production step compares the cost of manual execution to the unified AI-based execution cost. At first glance, this might seem like another way of stating that steps get assigned to whichever factor has the comparative advantage in that step, but this is not the case: position within the productive process profoundly changes the optimal allocation of steps between humans and AI. When a single step is performed by an AI as an augmented task, there is always a human requesting the task and evaluating its output in an AI-augmented manner. But with two or more steps it is possible to reconfigure production so that an AI completing one step passes its output directly into the next step without human intervention (i.e., the first step is automated). The AI chain (Steps 2–4) in Figure 1 illustrates this case: Steps 2 and 3 are fully automated (their outputs flow from one step to the next without direct human intervention) whereas Step 4 is AI-augmented with human oversight at the end of the chain. This chaining is potentially a source of large productivity gains *but* it requires AIs that can perform each step with a high probability of success. The model's basic setup shares conceptual similarities with the O-ring model of production (Kremer, 1993), and the role humans play in the model as offering judgment on AI outputs is consistent with the emphasis in Agrawal et al. (2019).

A key implication of the model is that chaining steps together into a single task can reverse comparative advantage assignments for individual steps. It can be efficient to deploy an AI on a step even when humans hold a comparative advantage in that step in isolation. A surprising result of chaining is strong AI performance on one step can shift another step from human to AI execution through chaining, even if the AI's capability on the second step is inferior to that of a human. Step 3 in Figure 1 illustrates on such case: with appropriate parameter values, it may be that, *in isolation*, it is more efficient to complete Step 3 manually rather than with the assistance of AI, yet overall production costs are minimized when it is bundled with

Steps 2 and 4 in an AI-automated chain. This might occur if it is especially easy to verify the output of Step 4.

More generally, we should expect the gains from automation to be greatest when highly automatable steps occur together in the production process. Informally, we can think of a job as more or less "fragmented" in its exposure to AI depending on whether the steps on which AI is most effective are mutually coincident or interleaved with steps for which humans hold a strong comparative advantage. In Section 4.1 we define a metric of fragmentation for jobs within our model and show that it approximates the impact of optimal AI deployment.

In our model, with highly capable AIs, the human role shrinks to asking and judging. This perhaps sounds like an uninspiring role but it is not necessarily a low-paid or unimportant one. Consider the famed music producer Rick Rubin, who is not poor, discussing his market value in an interview with Anderson Cooper:

RICK RUBIN: I've no technical ability. And I know nothing about music.
ANDERSON COOPER: Well, you must know something.
RICK RUBIN: I know what I like and what I don't like. I'm decisive about what I like and what I don't like.
ANDERSON COOPER: So what are you being paid for?
RICK RUBIN: The confidence that I have in my taste, and my ability to express what I feel, has proven helpful for artists.

We might be paid for being hepful to AIs. And this help might be valuable, even if we have no ability to do the tasks ourselves. In other words, automation can enable new forms of labor-labor substitution by separating task execution from task evaluation, allowing workers to leverage judgment skills even in domains where they lack execution abilities, potentially upending the allocation of workers to occupations in hard-to-forsee ways (see Autor and Handel (2013) on this human capital-task-wage relationship).

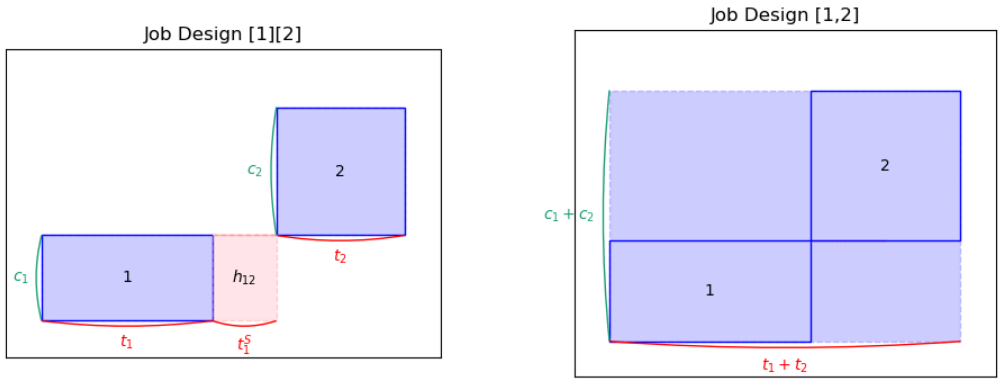
Jobs and Worker Skill. In the short-run, the primary impact of AI automation is to modify the amount of time needed for workers to complete their work. But over a longer time horizon, the optimal deployment of AI automation influences not only the time required to complete tasks, but also the level of required worker skill (i.e., the human capital investment required to train labor to be proficient at the task) and even the optimal division of tasks into jobs. To model these effects, we associate with each production step both a time requirement and a skill requirement. To complete a job, a worker must have the skills required of all tasks that make up the job description. The total labor cost (i.e., wage bill) of a job increases proportional to both the time needed to complete all steps and with the total skill level of the worker assigned to the job.

To capture the inefficiencies of fragmenting work into narrowly defined jobs, we introduce a *hand-off cost*, in time, that must be paid whenever output moves from one worker's job to another's. This creates a tension in job design: specialized jobs, containing fewer tasks, align worker skills more precisely with the job's requirements but incur higher hand-off costs. By contrast, generalist jobs, which bundle multiple tasks with different skill needs, reduce hand-off costs but require workers to acquire a broad set of skills, only a subset of which is deployed at any given time.

Figure 2 illustrates the trade-off firms face when deciding how to assign tasks to workers in a production process that aggregates steps into two tasks. In both panels, the shaded regions represent total wage bills. On the left, the firm employs two specialized workers, paying lower wage rates c_1 and c_2 but incurring coordination costs between them. On the right, the firm combines both tasks into one job, avoiding coordination costs altogether but paying a higher per-unit-time wage rate $c_1 + c_2$ to a more skilled worker.

In some production sequences, adjacent tasks differ sharply in their time and skill profiles—the two cost dimensions that are focal in our model. When a time-consuming, low-skill task is immediately preceded or followed by a short, high-skill task, having a single worker perform both tasks has the unfortunate effect that a necessarily highly skilled individual spends a significant amount of time doing low-skill work. We refer to such tasks as "tent-pole tasks." The sharp contrast between the skill and time costs of tent-pole tasks creates a friction that has historically driven the division of labor. For instance, rather than having a

Figure 2: Illustration of the Trade-off between Specialization and Coordination in Task Assignment



Note: Each task is represented by a rectangle whose height corresponds to its skill requirement (c) and width to its time requirement (t). The figure contrasts two organizational choices. In the left panel, each task is performed by a separate worker, reducing required skill per worker but introducing a hand-off cost (the pink rectangle) due to coordination and transfer frictions. In the right panel, both tasks are bundled into a single job performed by one worker, eliminating hand-off costs but requiring a worker skilled enough to perform both tasks and compensated with a higher per-unit-time wage.

high-skill worker complete both tasks, it may be more efficient to assign time-consuming, low-skill tasks to a different low-skill worker, even if this incurs extra hand-off costs.

As the deployment of AI automation changes the way worker tasks are defined, and hence changes the profile of time and skill requirements over the course of production, this can in turn influence the economic forces that determine the level of worker specialization. If certain tasks become automated, tent-pole scenarios like the one described above may shift or be avoided altogether, impacting the optimal assignment of production tasks across workers. This, in turn, can influence the relative balance of skilled labor as an input to production, with implications for wages and the distribution of productivity gains. We view this as a potential source of structural deskilling, whereby optimal AI deployment and automation can either increase or decrease the relative demand for skilled labor in a given production process.

Optimizing Automation and its Implications. Assigning production steps to factors of production is not a completely trivial optimization problem, even computationally. For a firm with m steps in its production process, the number of possible production arrangements grows exponentially in m. For the short-term optimization problem of optimizing the deployment of AI automation to a fixed set of job responsibilities and worker skill levels, we show how to analytically find the optimal automation strategy for an m-step job in time $O(m^2)$ using dynamic programming. Long-term optimization—the joint problem of optimizing labor (the division of steps into jobs) and the use of AI automation (including optimal chaining)—can also be solved in polynomial time, subject to an error term that can be made arbitrarily small. Although we do not do it in this paper, this algorithmic approach could be combined with micro-details of task content and composition (as in Frey and Osborne (2017); Felten et al. (2021); Eloundou et al. (2023)) to create rich estimates of how technological change in various tasks would impact workers.

The chaining feature of the model connects to—and potentially reconciles—competing views about automation in production. Despite the emphasis on task-level substitution between AI and labor in most of the AI and labor literature, Autor et al. (2003), Acemoglu and Restrepo (2019) and Bresnahan et al. (2002), among others, argue that that meaningful substitution happens primarily at the system level. One could interpret the model as showing how task-level decisions aggregate into system-level automation through chaining. In our model, there is no Ship-of-Theseus requirement that tasks are automated one by one;

rather, entire chains can be automated all at once—consistent with the Bresnahan et al. (2002) perspective.

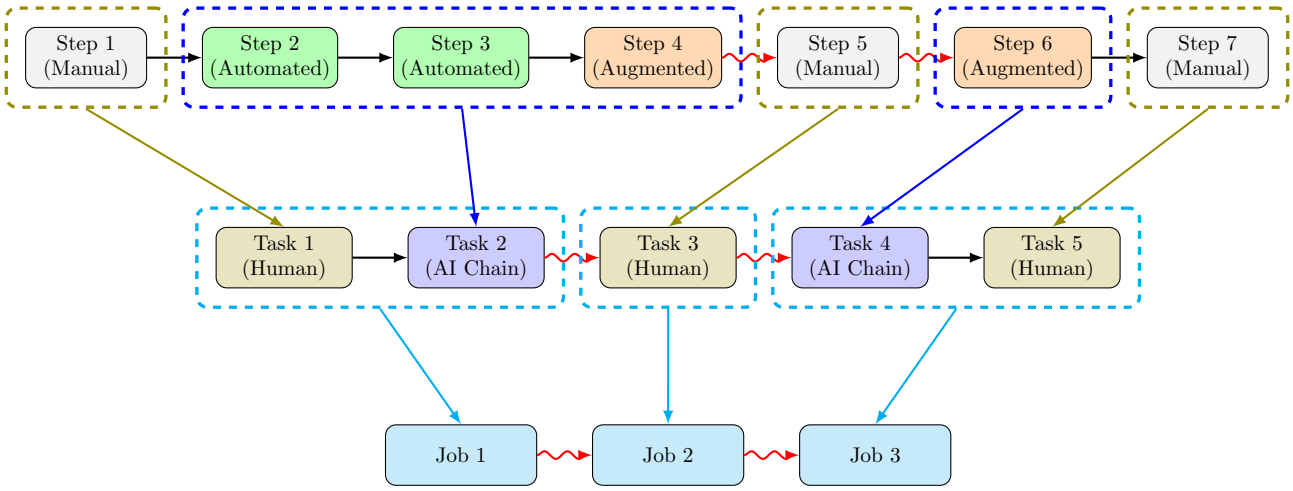
This ability for automation to be implemented in large leaps implies that improvements in AI quality can generate nonlinear effects on labor demand and wages. In the short run, higher AI performance reduces task time costs, potentially lowering labor demand. In the long run, however, worker retraining and unrestricted task reallocation can yield new wage schedules that reflect updated human capital and time cost parameters. These threshold effects imply that marginal AI advances may have little impact until they cross a short-run redesign or long-run restructuring tipping point, after which labor demand and job structures may shift abruptly. Our framework thus offers testable predictions on when and how AI-driven productivity gains trigger structural reorganization in labor markets.

Our model implies a form of algorithmic cost function for production, defined implicitly as the solution to an optimization procedure. However, as we show, the solution can also be interpreted as a standard CES production function that can be embedded within a general model of the economy. This allows us to distinguish manual and AI-assisted labor as separate inputs to production and model the aggregate demand for each.

2 Model

A firm's production process consist of a sequence of steps $\mathcal{S} = (s_1, \ldots, s_m)$. The firm can partition contiguous subsequences of steps into a sequence of tasks $\mathcal{T} = (T_1, \ldots, T_n)$ where $T_b = (s_i, \ldots, s_{i+\ell})$ for some i and ℓ. The firm can also partition contiguous subsequences of the resulting tasks into jobs $\mathcal{J} = (J_1, \ldots, J_p)$. Figure 3 provides an illustration for a production process with $m = 7$ steps, $n = 5$ tasks, and $p = 3$ jobs.

Figure 3: Illustrative Example of a Firm's Production



Notes: This figure illustrates a firm's production process in three layers. The top layer includes $m = 7$ steps categorized into Manual (gray), Automated (green), and Augmented (orange) modes of execution. Steps are grouped by dashed boxes colored according to their corresponding task boxes in the layer below. The second layer aggregates steps into $n = 5$ Tasks, labeled as Human-executed tasks (olive) or AI Chain tasks (blue) depending on the mode of execution. Dashed boxes in this layer are colored cyan to correspond with the jobs layer below. The third layer consolidates tasks into $p = 3$ jobs, represented in cyan-colored boxes. Vertical colored arrows between layers show aggregation from individual steps into tasks and thenx jobs. Within each layer, horizontal arrows (black and red) show the flow of production. The red curly arrows represent the hand-off costs between jobs, evident from the third layer, that can be traced back to tasks and steps in the upper layers.

The cost of any such partitioning depends on the time and skill needed for the component steps, the mode in which the step is completed, and the grouping of steps and tasks. We discuss these in turn.

2.1 Steps

Each step can be completed with or without the use of AI. If the firm wishes to hire a human worker to complete step i manually (without AI), it must hire a worker of skill level c_i^H and pay them for their time t_i^H spent completing step i. Alternatively, if the firm wishes to augment a human worker with AI to complete step i, it must hire a worker of (a potentially different) skill level c_i^M and pay them for their time t_i^M spent collaborating with AI[1] to complete step i. Such a collaboration succeeds independently with some (step-dependent) probability $q_i = \alpha^{d_i}$ where α is the quality of the general purpose AI technology and d_i represents how "difficult" step i is for AI. The collaboration must be repeated until it succeeds, incurring the time cost of t_i^M per iteration for a total expected time cost of t_i^M/q_i.

Definitions 1 and 2 formalize the two modes of performing a single step explicitly.

Definition 1 (Manual Step)**.** *A step i is said to be performed manually if it is executed entirely by a human worker without AI assistance. The associated skill and time costs for a manual step are denoted by (c_i^H, t_i^H).*

Definition 2 (Augmented Step)**.** *A step i is said to be augmented if it is executed by the AI, after which its output is reviewed and approved by a human worker. The associated skill and (expected) time costs of managing a single augmented step are denoted by $(c_i^M, \frac{t_i^M}{q_i})$, where $q_i \in (0, 1]$ is the AI's probability of successfully completing the step.*

2.2 Tasks

Tasks are the basic unit of work of a human worker. Any step performed in isolation during the production process, either manually or in collaboration with AI augmentation, is a task. The associated skill and time costs for the resulting task are precisely the skill and time costs for the associated step performed in the chosen mode. Steps can also be automated by chaining them together, creating a new aggregate task.

Definition 3 (Automated Step)**.** *A step is said to be automated if it is executed entirely by AI without direct human intervention, and its output is passed directly to a subsequent augmented or automated step. The direct human costs (in both skill and time dimension) associated with an automated step are zero.*

Definition 4 (AI Chain)**.** *An AI chain is a contiguous block of one or more sequential steps executed by AI, in which all steps but the final one are automated and the final step is augmented. An AI chain spanning steps $(s_\ell, \ldots, s_r)$ has steps $(s_\ell, \ldots, s_{r-1})$ automated and step s_r augmented. The skill and time costs of this AI chain are given by:*

$$\left(c_r^M, \frac{t_r^M}{\prod_{i=\ell}^r q_i}\right),$$

where q_i denotes the AI success probability for step s_i.

We can think of an AI chain as a single aggregate task that is being delegated to an AI. Successful completion requires that the AI complete all steps in $(s_\ell, \ldots, s_r)$. The human worker who manages the AI execution of this chain need only prompt for and evaluate the goal step, s_r; all other steps in the chain are assumed to be fully automated "under the hood" and hence beneath the awareness of the human worker. Managing an AI attempt at the chain therefore has the same costs as augmenting step s_r: namely, skill cost c_r^M and time cost (per attempt) t_r^M. Successful completion of the chain requires that the AI complete *every* constituent step successfully. Assuming independent failures, this occurs with probability $\prod_{i=\ell}^r q_i$, which we can view as the success probability for the aggregate chained task. Putting this all together yields skill and time costs $(c_r^M, t_r^M / \prod_{i=\ell}^r q_i)$ for the AI chain, as described in Definition 4.

Note also that since $q_i = \alpha^{d_i}$, where recall d_i is a measure of the difficulty of step s_i, the probability of successful completion of an AI chain spanning steps $(s_\ell, \ldots, s_r)$ can be written as $\alpha^{(\sum_{z=\ell}^r d_z)}$. We can therefore think of $\sum_{z=\ell}^r d_z$ as the total difficulty of the chain, which aggregates additively over its constituent steps.

[1] This could, for example, represent the time spent formulating a prompt for AI and checking the response.

To summarize, a task is either a manually-performed step, an AI-augmented step, or an AI chain. A sequence of tasks $\mathcal{T}$ can be viewed as a partition of step sequence $\mathcal{S}$ into contiguous subsequences, with each singleton being designated for either manual or augmented execution.

2.3 Jobs

A job is a subsequence of tasks that are assigned to a single human worker. Recall that each task T_b in task sequence $\mathcal{T} = \{T_1, \ldots, T_n\}$ has associated skill and time costs (c_b, t_b) that can depend on whether the task is being completed manually or with the aid of AI. A job J is a contiguous subsequence of tasks, say $J = (T_b, \ldots, T_{b+\ell})$ for some b and ℓ. We say that a partition $\mathcal{J} = (J_1, \ldots, J_p)$ of all tasks into jobs is a *job design*.

The firm hires one worker for each job in its job design $\mathcal{J}$. The worker for job J_j completes all tasks associated with their job. The total time needed to complete all tasks in job J_j is $\sum_{T_b \in J_j} t_b$.

A worker's wage is determined by the total skill required to complete their job. The total skill required to complete the tasks in job J_j is $\sum_{T_b \in J_j} c_b$. We then assume that a worker's wage is proportional to the skill level of their job.[2] That is,

$$\text{Wage}_j = \sum_{T_b \in J_j} c_b. \tag{1}$$

The firm must pay workers their wage per unit of time regardless of which tasks they are assigned to complete at any given moment.[3]

While we allow the firm to specialize its job design and partition its production process into distinct jobs, we acknowledge that there are inherent inefficiencies and frictions that are introduced when splitting work among multiple workers. Absent such frictions in our model, it would always be optimal for a firm to specialize its workers as much as possible, assigning a distinct worker to each task of its production process. We therefore introduce *hand-off costs* to a job design, as follows. In addition to the time required completing tasks, a worker may need to spend time handing off their work to another worker if J_j is not the final job in the job design $\mathcal{J}$. We will assume throughout that the hand-off time of a worker assigned to job J_j depends only on the final step of job J_j (see Figure 3). That is, conditional on where the hand-off occurs in the production process, its cost does not otherwise depend on previous hand-off events. Given step s_i, we write t_i^S to denote the additional hand-off time spent by a worker for whom the last step of their job is s_i. For notational convenience we define the hand-off time for the final step s_m to be zero: $t_m^S = 0$. Also for notational convenience, we will write $t^S(J_j)$ to denote the hand-off time of job J_j, which recall is equal to t_i^S if s_i is the final step in job J_j. Thus the total time needed to complete job J_j, including hand-off costs, is

$$t^S(J_j) + \sum_{T_b \in J_j} t_b.$$

Taking into account both time and skill requirements, the total wage bill paid to a worker assigned to job J_j, per unit of output, is

$$\text{WageBill}_j = \left(\sum_{T_b \in J_j} c_b \right) \left(t^S(J_j) + \sum_{T_b \in J_j} t_b \right). \tag{2}$$

[2] One motivation for this formulation is workers paying a human capital investment to acquire the skills necessary for a job. This investment must be offset by wages, and we express skill levels in units of their corresponding requisite wage. Rather than formalizing such a wage model here, we impose this as an assumption and provide a more detailed wage formulation in Section 5.

[3] This derivation implicitly assumes a common base wage rate; i.e., all wage differentiation is driven by the human capital cost of acquiring skills. More generally, different tasks T_b within job J_j could have different base wage rates w_b that could be influenced, for example, by the supply of and demand for labor of the corresponding type. One special case, explicitly motivated and discussed in Section 5, assumes manual tasks are executed by human labor at base wage rate w_H, and AI-assisted tasks are executed by AI management labor at base wage rate w_M. The formulation in Equation 1 implicitly normalizes these base wage rates to 1, incorporating them into skill costs c_b for notational simplicity. We maintain this normalization throughout subsequent sections until we explicitly distinguish human labor and AI management labor base wage rates in Section 5.

Equation (2) highlights two opposing forces that shape a job's boundaries. Adding more tasks to a worker's job increases the cumulative skill requirement and thus the wage rate for that worker. However, if tasks are kept separate, workers must incur additional hand-off costs at each job boundary. These two effects—higher wages from combining tasks versus higher hand-off costs from keeping them separate—jointly determine the optimal degree of task aggregation. We explore this trade-off in greater detail in Section 2.5.

2.4 Firm's Organizational Structure

Given the sequence of steps $\mathcal{S} = \{s_1, \ldots, s_m\}$, the firm can design both the partition $\mathcal{T}$ of steps into tasks and the partition $\mathcal{J}$ of tasks into jobs. The choice of $\mathcal{T}$ determines the time and skill requirements of each task. Given $\mathcal{T}$, the choice of $\mathcal{J}$ then determines worker wage rates and time needed per unit of output (including hand-off costs). The full optimization problem faced by the firm can be expressed as

$$\min_{\mathcal{T}} \min_{\mathcal{J}} \text{TotalCost}(\mathcal{J}; \mathcal{T}) = \sum_{J_j \in \mathcal{J}} \text{WageBill}_j = \sum_{J_j \in \mathcal{J}} \left[\left(\sum_{T_b \in J_j} c_b \right) \left(t^S(J_j) + \sum_{T_b \in J_j} t_b \right) \right] \tag{3}$$

where recall that, for each $T_b \in \mathcal{T}$, t_b and c_b are as described in Section 2.2 and can depend on the selected mode of operation for individual steps.

We refer to (3) as the firm's *long-term* optimization problem, as it anticipates the adjustment of worker wages to fit the skill requirements of each job and sets job responsibilities accordingly. We also define a *short-term* optimization exercise in which jobs and worker wages are fixed but the firm may still wish to use AI to maximize worker productivity. This is equivalent to minimizing the time needed for a worker to perform a unit of work in their assigned job. It therefore suffices to optimize for each job separately. Thus, in the *short-term* optimization problem, we can assume without loss of generality that the sequence of steps $\mathcal{S} = (s_1, \ldots, s_m)$ is to be completed by a single worker paid at a (normalized) unit wage. The resulting optimization problem faced by the firm can be expressed as

$$\min_{\mathcal{T}} \sum_{T_b \in \mathcal{T}} t_b \tag{4}$$

where recall that if $T_b = (s_\ell)$ is a manual task then $t_b = t_\ell^H$, and otherwise if $T_b = (s_\ell, \ldots, s_r)$ for some $\ell \leq r$ then $t_b = \frac{t_r^M}{\prod_{i=\ell}^r q_i}$ where q_i is the AI's probability of successfully completing step s_i.

2.5 Hand-off Costs and the Limits of Worker Specialization

Here we discuss how hand-off costs determine the optimal organization of production and limit full worker specialization in the firm's long-run problem (3). We leverage insights from a numerical example as well as a geometric illustrations, which together show how task aggregation balances higher wages against lower coordination costs.

Consider a production process where, for some fixed automation strategy $\mathcal{T}$, consists of $n = 3$ tasks with the following cost parameters:

Task	c_b	t_b	t_b^S
1	3	1	3
2	1	2	0.5
3	2	2	0

With three tasks, there are four ways the firm can design jobs.[4] Denoting jobs by square brackets, with three tasks the four possible job designs are $\{[1][2][3], \quad [1,2][3], \quad [1][2,3], \quad [1,2,3]\}$. The cost of these job designs for the example above is given in Table 2. In the absence of hand-off costs in Panel (a), the optimal job design corresponds to full specialization: each task forms a separate job assigned to a different worker.

[4]More generally, a production process with n tasks has $2^{(n-1)}$ unique job designs.

Table 2: Role of Hand-off Costs in Organizational Structure

Panel (a): Without Hand-off Costs

Job Design	**Job**	**Job Tasks**	$\sum c_b$	$\sum t_b$	**Job Cost**	**Total Cost**	**Optimal Design**
[1][2][3]	1	$\{1\}$	3	1	3	9	✓
	2	$\{2\}$	1	2	2		
	3	$\{3\}$	2	2	4		
[1,2][3]	1	$\{1,2\}$	4	3	12	16	
	2	$\{3\}$	2	2	4		
[1][2,3]	1	$\{1\}$	3	1	3	15	
	2	$\{2,3\}$	3	4	12		
[1,2,3]	1	$\{1,2,3\}$	6	5	30	30	

Panel (b): Including Hand-off Costs

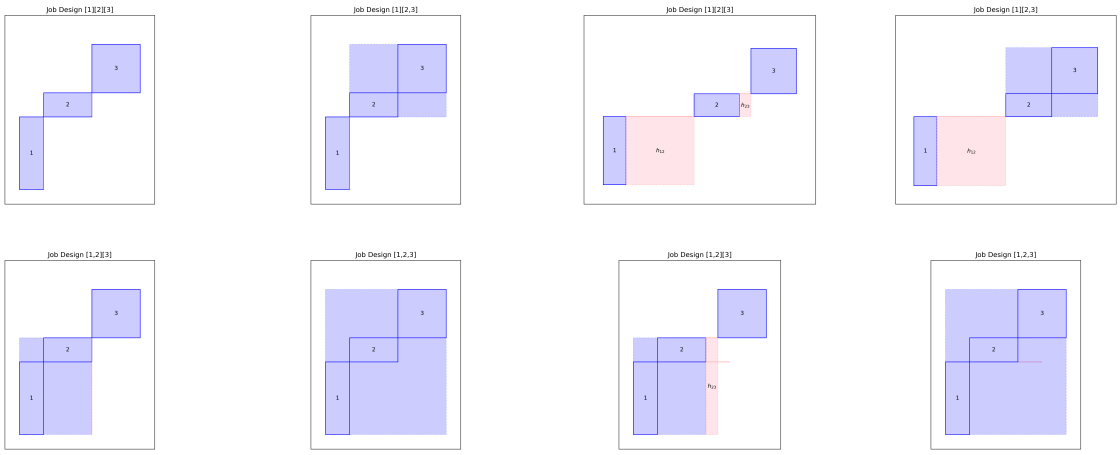
Job Design	**Job**	**Job Tasks**	$\sum c_b$	$t^S + \sum t_b$	**Job Cost**	**Total Cost**	**Optimal Design**
[1][2][3]	1	$\{1\}$	3	4	12	18.5	
	2	$\{2\}$	1	2.5	2.5		
	3	$\{3\}$	2	2	4		
[1,2][3]	1	$\{1,2\}$	4	3.5	14	18	✓
	2	$\{3\}$	2	2	4		
[1][2,3]	1	$\{1\}$	3	4	12	24	
	2	$\{2,3\}$	3	4	12		
[1,2,3]	1	$\{1,2,3\}$	6	5	30	30	

Notes: This table reports total production costs for each possible job design in an example production process with a fixed automation strategy and $n = 3$ tasks. The cost parameters are given by $(c_b, t_b, t_b^S) = \{(3,1,3),(1,2,0.5),(2,2,0)\}$. In the absence of hand-off frictions, full specialization, corresponding to job design [1][2][3], is cost-minimizing. Once hand-off costs are taken into account, however, the optimal organizational structure changes to job design [1,2][3], which avoids the large coordination cost between tasks 1 and 2 at the expense of employing a higher-cost (more skilled) worker to complete those tasks.

When hand-off costs are introduced in Panel (b), however, the optimal design changes to one in which tasks 1 and 2 are combined into a single job, while task 3 remains separate. This structure is optimal because it avoids the large hand-off cost that would otherwise arise between tasks 1 and 2 due to the large hand-off time value $t_1^S = 3$. By aggregating tasks 1 and 2, the firm pays a higher wage to a more skilled and more versatile worker but eliminates the costly coordination between the two tasks, lowering total production cost. Equation (2) captures this trade-off explicitly: adding more tasks to a worker's job raises the wage rate through the skill term $\sum_{T_b \in J_j} c_b$, while keeping tasks separate raises total cost through additional hand-offs $t^S(J_j)$. These two effects jointly determine the optimal degree of task aggregation.

The cost of different job designs admits a geometric interpretation as well. Figure 4 visualizes the production process of the example in Table 2 as stacked rectangles, where each rectangle's height corresponds to c_b and its width to t_b. Panel (a) depicts the job design when hand-off costs are zero, Panel (b) shows

Figure 4: Geometric Interpretation of Job Designs



(a) Job Design Without Hand-off Costs

(b) Job Design With Hand-off Costs

Notes: Each panel visualizes the production process as stacked rectangles, where the area of each job's bounding box represents its wage bill. Panel (a) sets hand-off costs to zero, in which case it is optimal to assign one task per job. Panel (b) introduces hand-off costs, which alters the cost structure and makes the [1,2][3] design cost-minimizing.

the same process with hand-off costs. Visually, optimal job design balances two opposing forces. Combining tasks into a single job eliminates the pink rectangles representing additional hand-off costs between tasks, but creates inefficiencies due to the higher skill level of the worker required to do the job. Conversely, splitting tasks into separate jobs eliminates the intra-job inefficiencies but incurs hand-off time costs at each job boundary. These boundary costs scale with the cumulative human capital of the switching worker, so the height of the pink hand-off rectangle in Panel (b) equals the sum of human capital heights for that job. Cost-effective job design must balance these effects.

3 Optimization

In this section we consider the optimization problems (both short-term and long-term) faced by a firm choosing how to integrate AI automation technology into their production process. We begin by studying the short-term problem of optimizing automation strategy keeping job design and worker wages fixed. We then move on to the joint optimization of automation and labor taking into account long-term wage impacts.

3.1 Short-Term Optimization: Automation Design

We begin with a "short-term" optimization problem: the job design and worker wages are assumed to be fixed, so the goal is to find the choice of automation strategy for a single job that minimizes the total time cost. This optimization problem analyzes short-run benefits of AI: where worker skills are fixed but AI automation can still be employed to speed up tasks through some combination of augmentation and automation.

Since the job design and worker wage are fixed, it suffices to optimize for the amount of time spent to complete a unit of work for each job separately. So, from this point onward, we assume that there is a single job J with m steps $\mathcal{S} = (s_1, \ldots, s_m)$, and our goal is to find the sequence of tasks $\mathcal{T}$ that minimizes total completion time. It turns out that the time-optimal production strategy can be calculated via dynamic programming in $O(m^2)$ time.

Proposition 1. *Given m steps organized into a single job, the time-optimal automation strategy can be calculated in time $O(m^2)$ via dynamic programming.*

Proof. We first note the following recursive formulation of the optimization problem. For all $k \leq m$, let $C[k]$ denote the minimum time needed to complete a hypothetical job that only includes steps 1 through k. Note that, because of the way we defined $C[k]$, task k cannot be automated in any minimum-time solution that determines $C[k]$; it can only be augmented or performed manually. Note also that $C[m]$ is the time cost of our desired optimal solution.

We now show how to calculate $C[k]$ recursively. As a base case we have $C[0] = 0$, as the empty set of steps requires no time to complete. For $k \geq 1$, $C[k]$ is the lesser of

$$C[k-1] + t_k^H$$

and

$$\min_{\ell<k}\left\{C[\ell] + \frac{t_k^M}{\prod_{i=\ell+1}^{k} q_i}\right\}.$$

In other words, we either complete step k manually (in which case we separately optimize over steps 1 through $k-1$) or we augment step k. In the latter case, we then optimize over the length of the AI chain that ends with k. This could be a singleton chain with no automation (corresponding to $\ell = k-1$), or a longer chain that automates one or more steps before step k (corresponding to a choice of $\ell < k-1$). For any such choice of ℓ, we then separately optimize the production strategy for tasks 1 through ℓ.

Using this formulation, given the values $(C[0], \ldots, C[k-1])$, we can calculate $C[k]$ in time $O(k)$ by considering each potential choice of ℓ. Doing so for each $k = 1, 2, \ldots, m$ yields the value of $C[m]$ (and the corresponding automation strategy) in total time $O(m^2)$. □

3.2 Warm-up to Long-Term Optimization: Job Design without AI

We now move to a broader optimization problem in which the firm can reorganize job requirements and the assignment of tasks to workers. As a warm-up to the full joint optimization of jobs and automation, we first show how to optimize the design of jobs for a fixed task structure. We can interpret this as a labor optimization problem without AI automation, where each step has only a single (i.e., manual) mode of completion and hence the task structure is fully determined.

Proposition 2. *Given a fixed sequence of tasks $\mathcal{T} = (T_1, \ldots, T_n)$ with skill and time costs $c = (c_1, \ldots, c_n)$ and $t = (t_1, \ldots, t_n)$, the cost-minimizing job design $\mathcal{J}$ can be computed in time $O(n^2)$ by dynamic programming.*

Proof. We first note the following recursive formulation of the optimization problem. For all $k \leq n$, let $C[k]$ denote the minimum cost of a job design for tasks 1 through k, *including hand-off costs for task k*. Note then that $C[n]$ is the cost of the optimal job design for all tasks, recalling that the hand-off cost for the final job (i.e., the job that includes task n) is always 0.

We now show how to calculate $C[k]$ recursively. As a base case we have $C[0] = 0$. For $k \geq 1$ we have

$$C[k] = \min_{s<k} \left\{ C[s] + \left[\left(\sum_{i=s+1}^{k} c_i \right) \left(t_k^S + \sum_{i=s+1}^{k} t_i \right) \right] \right\}.$$

In other words, we optimize over the choice of the final job $\{s+1, \ldots, k\}$, accounting recursively for the optimal job design for remaining jobs. Note that we make implicit use of the assumption that the handoff cost t_k^S depends on the final task T_k of this final job but is otherwise independent of the job design. Using this formulation, we can calculate each $C[k]$ in time $O(k)$ by considering each potential choice of s. Doing so for each $k = 1, 2, \ldots, n$ yields the value of $C[n]$ (and the corresponding optimal job design) in total time $O(n^2)$. □

3.3 Full Long-Term Optimization

We now consider full joint optimization of job design and automation strategy, accounting for both time and skill costs of workers assigned to the resulting jobs. In other words, the firm's goal is to choose both the set of AI chains to implement—yielding a vector of skill and time costs—along with a job design for the resulting tasks.

3.3.1 Recursive Formulation of Optimization Problem

Here we present a recursive formulation of the firm's cost minimization problem. In Section 3.3.2 we propose an approach to calculate an approximation of the optimal solution via dynamic programming.

Consider a production process with m steps $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$. For $0 \leq i \leq m$, let $V(i)$ denote the minimum cost required to complete steps 1 through i using one or more jobs, including any hand-off costs to a subsequent worker. Note then that $V(m)$ is the solution to the desired optimization problem, recalling that the hand-off time for the final job will always be 0. Note also that it is implicitly assumed in the definition of $V(i)$ that it optimizes over job designs for steps 1 through i in which the final job terminates after the completion of step i.

It will also be helpful to consider optimal designs for steps 1 through i in which the final job doesn't necessarily terminate after the completion of step i, but rather continues onward to subsequent steps. To that end, we introduce an auxiliary function $W(i,\ c,\ t)$. This function denotes the minimum cost of completing steps 1 through i, as well as some additional (but unspecified) set of tasks that have already been assigned as a single job to a single worker (referred to as the "active worker"). It is assumed that the tasks assigned to the active worker have total time cost t and total skill cost c. The time cost t is assumed to already include any hand-off cost for this active worker (which recall depends only on the final step of the worker's final task). Crucially, the minimum is taken over all automation strategies and job designs, including those that expand the number of tasks assigned to the active worker (e.g., by including step i in the active worker's job). In this sense, $W(i,\ c,\ t)$ captures the minimum cost across all feasible job designs and automation strategies that may potentially expand the active worker's task assignment.

Note that, for all $i \leq n$, we have

$$V(i) = W(i,\ 0,\ t_i^S).$$

This is because, in $V(i)$, all job designs must end with a job that completes at step i and hands off to the following worker. Consequently, the hand-off cost associated with step i must be incurred by the final worker. Equivalently, this can be thought of as assigning the active worker an additional task with zero skill requirement but a time cost equal to the hand-off cost of step i right after step i itself. This scenario is represented explicitly by $W(i,\ 0,\ t_i^S)$.

Also note that, for the base case $i = 0$, we have

$$W(0,\ c,\ t) = c\,t, \quad \text{for all } c, t.$$

This is because, if $i = 0$, then there are no further steps to add to the job of the active worker. The active worker's skill and time requirements are therefore determined entirely by the steps they have already been assigned.

We are now ready to describe a recursive formulation of the function $W(i,\ c,\ t)$ for $i \geq 1$. We emphasize that in the definition of $W(i,\ c,\ t)$, the active worker has not been assigned step i; we can think of step i as the highest-indexed step that has not yet been assigned to a worker.

Proposition 3 (Recursive Formulation of Job Design Problem)**.** *For each $i \geq 1$, the minimum cost function $W(i,\ c,\ t)$ satisfies the following recursive relation:*

$$W(i,\ c,\ t) = \min\Bigg\{ c\,t\ +\ V(i),$$
$$W(i-1,\ c + c_i^H,\ t + t_i^H),$$
$$\min_{r<i}\ W\left(r,\ c + c_i^M,\ t + \frac{t_i^M}{\prod_{s=r+1}^{i} q_s}\right)\Bigg\}.$$

The cost of the optimal joint automation and job design for a given sequence of m steps is then $V(m) = W(m,\ 0,\ 0)$.

The recursive formulation in Proposition 3 evaluates the minimum cost among three distinct alternatives at each step:

- **Option (1):** The term $c\,t\ +\ V(i)$ corresponds to not adding any further steps to the job of the active worker. All steps 1 through i will be completed by other workers. The active worker's total cost is then $c\,t$, and $V(i)$ is the total cost of completing steps 1 through i (including hand-off costs to the active worker).
- **Option (2):** The term $W(i-1,\ c+c_i^H,\ t+t_i^H)$ corresponds to designating step i for manual completion, and adding the corresponding (singleton) task. In this case, the manual execution costs (c_i^H, t_i^H) of step i are added to the accumulated costs of tasks already assigned to the active worker, extending their job to include step i as well.
- **Option (3):** The term $\min_{r<i}\ W\left(r,\ c + c_i^M,\ t + \frac{t_i^M}{\prod_{s=r+1}^{i} q_s}\right)$ corresponds to creating an AI chain task for steps $r+1$ through i, with step i being augmented and steps $r+1$ through $i-1$ automated, and assigning the resulting task to the job of the active worker. Step i is thus chained with zero or more preceding (automated) steps, and the associated AI management costs are added to the active worker's skill and time costs. The index r corresponds to the highest-index step that is not added to this AI chain, which can be any value between 0 and $i-1$.

Note that in evaluating option (3) of the recursive step for $W(i,\ c,\ t)$, step i can only be augmented (rather than automated) as each step of the recursion adds a full AI chain to an active worker's job. Automation strategies in which step i is automated are considered when performing task calculations $W(r, c, t)$ with $r > i$.

3.3.2 Calculating an Approximately Optimal Solution

Given our recursive formulation, we wish to use dynamic programming to solve for the jointly optimal job design and automation strategy by filling in the values of $W(i,\ c,\ t)$ for all i, c, and t. Since c and t take on continuous values, we will discretize all skill costs c and time costs t to appropriate powers of $(1+\epsilon)$, where $\epsilon > 0$ is an arbitrarily small error term. We obtain the following result.

Proposition 4. *Fix any sequence of m steps $\mathcal{S} = (s_1, \ldots, s_m)$ for which all skill costs, time costs, and hand-off costs lie in $[1/B, B]$ for some $B > 0$. Then for any $\epsilon > 0$, an approximately cost-minimizing pair*

of automation strategy $\mathcal{T}$ and job design $\mathcal{J}$ minimizing expression (3) *to within a factor of* $(1+\epsilon)$ *can be computed in time* $O(m^2\epsilon^{-2}\log^2(mB))$ *by dynamic programming.*

Proof. The first step of proving Proposition 4 is to determine the range of powers of $(1+\epsilon)$ that we must consider in our discretization. Suppose that all manual and augmented skill and time costs, as well as all hand-off costs, lie in $[1/B, B]$ for some $B > 0$. That is, these parameters require $O(\log B)$ bits to represent. In this case, note that any subsequence of steps could be completed at a total cost of at most $2mB^2$ by performing them all manually by different workers, incurring a time and handoff cost of at most B each for a total time of $2B$, and a skill cost of at most B for a total cost of $2B^2$ for each of the m tasks. We therefore need not consider any solution containing a job with total cost greater than $2mB^2$. In particular, since any non-zero skill cost for any job is at least $1/B$, we need not consider any job design in which a job has total time cost greater than $2mB^3$. Furthermore, each job has total skill cost lying between $1/B$ and mB (as skill costs combine additively) and time cost at least $1/B$ (as each step's manual cost is at least $1/B$, and any AI chain's time cost is at least the management cost of its final step which is also at least $1/B$).

We conclude that when filling table $W(i, c, t)$, it suffices to consider skill costs c lying in the range $[1/B, mB]$ and time costs t lying in the range $[1/B, 2mB^3]$. There are $O(\epsilon^{-1}\log(mB))$ powers of $(1+\epsilon)$ in each of these ranges, so our table will have $O(m\epsilon^{-2}\log^2(mB))$ total entries. Each entry can be filled in time $O(m)$ by considering the recursive formulation in Proposition 3, for a total runtime of $O(m^2\epsilon^{-2}\log^2(mB))$.

Note that while the table formally describes only the cost of a solution and not the solution itself, one can also read off the corresponding task and job design as well. Indeed, by recording which of the options described in Proposition 3 achieves the minimum for each table entry $W(i, c, t)$, one can trace out the corresponding task and job designs starting with $V(m) = W(m, 0, 0)$. The pair of automation strategy $\mathcal{T}$ and job design $\mathcal{J}$ can therefore be computed in time $O(m^2\epsilon^{-2}\log(mB))$, the same as the time required to fill the table.

It remains to bound the error introduced by our discretization. Consider the recursive formulation in Proposition 3 and suppose that c and t are rounded down to the nearest power of $(1+\epsilon)$. Since time and skill costs are multiplied together to calculate the total cost of a proposed job, this discretization introduces a multiplicative error of at most $(1+\epsilon)^2$ into our cost calculation for the first option when minimizing in the recursive definition of $W(i, c, t)$. For the other two options, note that we accumulate time and skill costs additively when chaining tasks together for a single worker. As c and t are rounded to a power of $(1+\epsilon)$, adding additional (accurate) skill and time costs and subsequently rounding maintains a multiplicative error of at most $(1+\epsilon)$ on the total time and skill costs.

We conclude that if we fill in our table for all $W(i,\ c,\ t)$ where $i \le m$ and c and t are discretized into powers of $(1+\epsilon)$, rounding down to nearest values of $(1+\epsilon)$ on recursive calls into the table, we obtain a $1+O(\epsilon)$ approximation to the optimal solution. An appropriate change of variables (scaling ϵ by a constant) yields a $(1+\epsilon)$ approximation factor in total runtime $O(m^2\epsilon^{-2}\log^2(mB))$, as claimed in Proposition 4. □

4 Discussion

4.1 Job-level AI Exposure and the Fragmentation Index

For a single step in isolation, the decision whether or not to deploy AI augmentation depends simply on whether the manual execution cost exceeds the AI management cost. That is, the optimal choice of whether to use AI assistance depends only on the AI exposure of the step in question. For two or more steps, however, the optimal deployment of AI depends on more than the exposure of each step in isolation. This can arise because of the benefits of automating multiple steps together in an AI chain. Even if one step can be completed more effectively through manual work, it may be preferable to automate it as part of a larger collection of steps that can be jointly delegated to AI. Whether this occurs in the optimal automation strategy depends on the relationship between other nearby steps in the production process.

Returning to the short-run optimization problem described in (4) and Section 3.1, we can interpret job-level AI exposure as the extent to which an optimal automation strategy employs AI automation and augmentation. Intuitively, a run of *consecutive* steps in the production process that an AI can perform

effectively is a natural candidate for an AI chain. A job that contains such runs of consecutive steps are therefore likely to benefit most from AI automation. On the other hand, a job for which AI-friendly steps are separated by intermediate steps that an AI is likely to fail is less exposed to large-scale automation, even though its task-level exposure to AI may appear high.

Example 1. *Consider a job consisting of m steps, where m is even. Each step takes time 5 to complete manually and has an AI management time of 1, but the steps vary in how difficult they are for an AI to complete: odd-numbered steps can be successfully completed by an AI with probability 1, whereas even-numbered steps will be successfully completed with probability only 0.1, as shown below:*

$$(Easy)_1 \longrightarrow (Hard)_2 \longrightarrow (Easy)_3 \longrightarrow \cdots\cdots \longrightarrow (Hard)_{m-2} \longrightarrow (Easy)_{m-1} \longrightarrow (Hard)_m$$

In this scenario, it is suboptimal to attempt to use AI on any of the even-numbered steps, even as part of a chain. The optimal automation strategy is to employ AI-augmentation on the odd-numbered steps (for a cost of 1 each) and perform even-numbered steps manually (for a cost of 5 each), resulting in an overall time cost of $3m$.

Example 2. *Next suppose that easy and hard steps are not interleaved, but rather the first $m/2$ steps can each be completed by AI with probability 1 and the last $m/2$ steps can each be completed with probability 0.1, as follows:*

$$(Easy)_1 \longrightarrow (Easy)_2 \longrightarrow \cdots\cdots \longrightarrow (Easy)_{\frac{m}{2}} \longrightarrow (Hard)_{\frac{m}{2}+1} \longrightarrow \cdots\cdots \longrightarrow (Hard)_{m-1} \longrightarrow (Hard)_m$$

In this case, the optimal strategy chains together the first $m/2$ steps into a single automated AI chain, for a combined cost of 1. The remaining $m/2$ steps are then performed manually. This results in an overall time cost of $1 + 5m/2$, which is strictly less than $3m$ as long as there are four or more steps.

To make this intuition more precise, let us return to the short-run optimization problem described in (4) and Section 3.1. Recall that in this short-run problem we are fixing a single job assigned to a worker with a fixed wage, so our focus is on optimizing the time cost of production. Consider the special case where AI management costs are normalized: $t_i^M = 1$ for all i. That is, each step of production requires the same amount of time to prompt and manage one attempt by an AI process. Under this assumption, we will define a measure of the dispersion of AI-exposed tasks in a production process, which we refer to as the *fragmentation index* of a job. We then show that the fragmentation index of a job approximates (up to constant factors) the time cost of an optimal automation strategy for that job. In other words, jobs for which the fragmentation index is high will tend to yield less benefit from AI automation in the short-term where worker wages and job boundaries are fixed. Notably, this intuition makes heavy use of the short-run assumption that worker wages are fixed and independent of task skill requirements; we discuss examples where this intuition fails when describing long-run effects in Section 4.2.

To define the fragmentation index, consider a random process in which each step s_i succeeds independently with probability q_i; any task that does not succeed is said to fail. Write F for the set of steps that fail, and $\mathcal{C} = \{C_1, \ldots, C_k\}$ for the random variable representing the collection of maximal connected components of non-failed steps. The weight of each $C_j \in \mathcal{C}$ is defined to be $\omega(C_j) = \min\{1, \sum_{i \in C_j} t_i^H\}$. That is, each C_j has weight 1 unless the sum of the manual time costs for each of its steps is less than 1 (due to our assumption that AI management costs are normalized to 1).

Given a realization of $\mathcal{C}$ and F, we define the realized fragmentation to be

$$\sum_{i \in F} \min\left\{t_i^H, \frac{t_i^M}{q_i}\right\} + \sum_{C_j \in \mathcal{C}} \omega(C_j). \tag{5}$$

The *fragmentation index* is defined to be the expected value of the realized fragmentation.

Intuitively, we expect the fragmentation index to be lower when highly automatable steps (that is, those for which the AI are likely to succeed) are clustered together, since a large cluster of highly automatable steps are more likely to realize as a single connected component when failures are realized.

We will show that the fragmentation index is within a constant factor of the cost of the optimal (short-term) automation strategy for a given fixed job's step sequence.

Proposition 5. *Fix a single job with a sequence of m steps $\mathcal{S} = \{s_1, \ldots, s_m\}$, each with $t_i^M = 1$. Let FI denote its fragmentation index and let OPT denote the minimum time cost over all automation strategies. Then $\frac{1}{8}OPT \leq FI \leq \frac{5}{4}OPT$. If we further assume $t_i^H \geq 1$ for all i, then $\frac{1}{4}OPT \leq FI \leq \frac{5}{4}OPT$.*

We prove Proposition 5 in Appendix A. The key take-away from Proposition 5 is that jobs with high fragmentation index—i.e., those for which the expected number of *consecutive* successful step executions by an AI is low—will tend to have higher time costs even under optimal use of AI chaining. A key driver of efficiency gains via AI automation is therefore not simply the expected number of steps that can be automated effectively, but the number of *adjacent* steps in the production process that have high exposure to AI.

4.2 Impact of AI Deployment on Worker Skill and Specialization

While the fragmentation index captures the short-run impact of AI deployment on task completion time, in the long run automation also alters the skill requirements of workers and the degree of specialization within the workforce. This can dampen or even reverse the short-run intuition that gains from AI derive primarily from time savings, as the following example shows.

Example 3. *Consider a job consisting of a single step. This step has manual time and skill requirements $(t^H, c^H) = (5, 5)$. Suppose that, under AI augmentation, this step has an AI management time of 1 (per attempt), an AI management skill requirement 1, and probability 1/8 of successful completion. Despite the task taking longer to complete with AI (due to the low likelihood of success on any individual attempt), the reduced skill requirement of managing the AI process means that it is firm-optimal to employ AI augmentation.*

The impact of AI on worker skill reflects two common narratives about the impact of AI on work: first, that AI can automate mundane or repetitive tasks and allow high-skill workers to concentrate more of their time on meaningful high-skill tasks; second, that AI could lead to deskilling as high-skilled labor for manual task completion is replaced with low-skill AI management. Our framework unifies these two perspectives by endogenizing (a) the nature (i.e., time and skill requirements) of work to be automated and (b) how the automation strategy impacts requisite worker skill. If replacing a given sequence of production steps in given job with an AI chain is beneficial, in the long term, then either the total time needed to complete the steps is shortened, the total skill needed to manage the AI automation is reduced relative to completing the steps manually, or both. For an example of the latter, a step with low skill requirement but high time requirement (such as Step 2 in Figure 4) may be optimally augmented by AI even when doing so increases the skill needed for AI management, if the time savings are substantial enough. In this case, the skill required to complete the same job might increase: AI augmentation can be viewed as complementing worker capabilities. Alternatively, it may also be beneficial to employ AI on production steps with high skill requirements (such as Step 1 in Figure 4) even if this results in an AI-assisted task that takes longer to complete, as long as it requires substantially less worker skill to manage the AI. This can ultimately lead to a deskilling of the labor force assigned to the job as AI capital substitutes for high-skilled human work.

This discussion has so far focused on impacts within a single job. Our framework likewise captures the impact of AI automation on job design, and specifically on worker specialization across tasks. Consider again the example from Figure 4, which illustrates how hand-off costs and the structure of tent-pole tasks[5] can influence the partition of production steps into specialized jobs. As AI automation influences the time and skill profile of tasks, the incentive to specialize workers likewise changes.

To predict the direction of this effect, one hypothesis is that AI automation will tend to have a normalizing effect on tasks, reverting both skill requirements and time requirements toward the mean and making tasks more "square-like" (in terms of our geometric interpretation of job design costs). If so, and if hand-off

[5]Recall that a tent-pole task refers to a high-skill but short-duration task that is adjacent to a low-skill but long-duration task.

time costs remain unaffected, then (roughly speaking) tent-pole inefficiencies due to grouping tasks together into the same job would be reduced. This suggests that the adoption of AI automation may reduce worker specialization.

Example 4. *Consider a two-step production process, with manual skill and time costs* $(c_1^H, t_1^H) = (2, 4)$ *and* $(c_2^H, t_2^H) = (4, 2)$ *and hand-off cost* $t_1^S = 5$*. That is, the first step is low-skill but time-intensive and the second step is high-skill but can be completed quickly. In this example, combining both steps into a single job (at a labor cost of* $(2 + 4)(4 + 2) = 36$*) is strictly worse than separating into two specialized jobs with a handoff (at a total cost of* $(2)(4 + 5) + (4)(2) = 26$*). However, if each step of production could be augmented separately via AI to yield effective skill and time costs of* $(2, 2)$ *for each, then the optimal design combines both augmented steps into a single job at a total cost of* $(2 + 2)(2 + 2) = 16$*, which is better than separating into two distinct jobs (incurring cost* $(2)(2 + 5) + (2)(2) = 18$*).*

It is also technically possible in our model for AI automation to increase the amount of specialization in an optimal job design. This could happen if AI-augmented steps have higher skill requirements than the corresponding manual steps, leading to an increased need for high-skill workers. Even if AI management is assumed to require no more skill than manual completion, an increased use of AI automation can lead to more responsibilities being combined into a single job, leading to a higher worker skill requirement, as the following example shows.

Example 5. *Consider a different two-step production process, with manual skill and time costs* $(c_1^H, t_1^H) = (c_2^H, t_2^H) = (3, 3)$ *and hand-off cost* $t_1^S = 5$*. The optimal job design separates these into two separate jobs at a total cost of* $(3)(3+5)+(3)(3) = 33$*, with each job requiring a worker of skill* 3*. Suppose AI augmentation can allow each step to be completed with an AI management skill of* 2*, a management time of* $1/4$*, and an AI success probability of* $1/8$*, for a total expected execution time of* $(1/4) \times (1/8)^{-1} = 2$*. In this case, the optimal design employs AI augmentation in each step and combines the two steps into a single job, resulting in a total cost of* $(2+2)(2+2) = 16$ *and requiring a worker of skill* 4*. Note that this is preferable to combining the two steps into a single AI chain, which would result in a total cost of* $(2)((1/4) \times (1/8)^{-1} \times (1/8)^{-1}) = 32$*.*

4.3 Non-Linear Impacts of AI Improvements

Improvements to AI technology naturally lead to improvements in the overall cost of production, but these effects need not be linear. The full economic impact of a disruptive general-purpose technology can take significant time and investment to materialize, with notable historical examples ranging from the steam engine to electricity to computers. In each case, a new technology enables modest short-term improvements to existing production processes, but the full impact is not felt until large-scale reorganization of production is enabled. This drives a potentially non-monotone marginal value from technology improvements, where major improvements are only unlocked after surpassing a threshold of capability at which they become feasible.

Our framework captures such non-monotonicities in the marginal value of technological improvements. This arises in our model via optimization over different automation and job design strategies. One can think of parameter $\alpha \in (0, 1]$ in our model as metric of the quality of a general purpose AI technology, taken to be the probability that a task of normalized difficulty 1 is completed successfully. We can then model general technology improvements as increasing the value of α. For a given *fixed* automation strategy (described by a task sequence $\mathcal{T}$) and job design (described by job sequence $\mathcal{J}$), the total cost can be expressed as a polynomial in $(1/\alpha)$. As a result, the cost of $\mathcal{T}$ and $\mathcal{J}$ shifts in a continuous and smooth manner as α increases, with monotone marginal gains to technology improvements. However, the optimal choice of design (expressed as optimization problem (3)) involves taking the minimum-cost solution over many possible automation and job designs. Automation strategies with a higher degree of AI chaining will naturally involve higher powers of $1/\alpha$ in their cost expressions, meaning that they are more sensitive to changes in α and become preferable only at higher values of α (i.e., as AI becomes more effective as a tool). This naturally leads to scenarios where the productivity improvements of AI integration are modest at low levels of quality but can grow sharply past a certain inflection point, as we describe in the following example illustrated in Figure 5 below.

Example 6. *Consider a production process with two steps. The first step is short but high-skill and difficult for AI tools to perform correctly. Specifically, its manual skill cost is* $c_1^H = 5$*, its manual time cost is* $t_1^H = 1$*, and its hand-off time is* 1*. Its AI difficulty score,* d_1*, is* 6*, meaning that an AI can successfully complete the step with probability* $q_1 = \alpha^6$*. The AI management skill and time requirements are the same as the manual execution requirements:* $(c_1^M, t_1^M) = (5, 1)$*.*

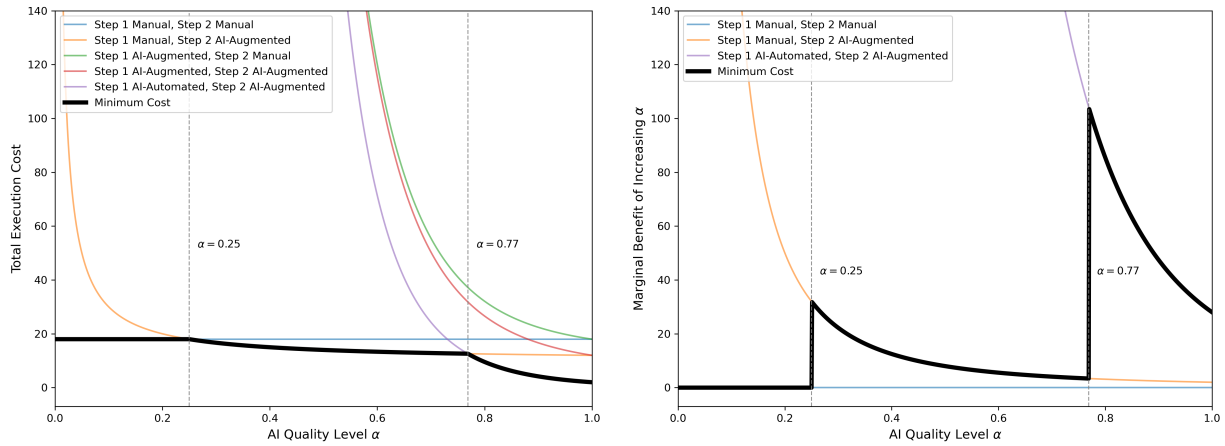
The second step is low-skill and timely to execute manually, but easy for AI. It has $(c_2^H, t_2^H) = (2, 4)$ *and AI difficulty score* 1*. The skill needed to manage the AI is also* 2*, but the time needed to manage the AI is reduced to* 1*:* $(c_2^M, t_2^M) = (2, 1)$*.*

Consider the optimal automation strategy and job design for this example as a function of α*, the general AI quality metric. The costs of different automation strategies are plotted in Panel (a) of Figure 5. For* $\alpha < 0.25$*, it is optimal to complete each step manually as a separate task, and to separate the two tasks into separate specialized jobs. This incurs a total cost of* $(5)(1+1)+(2)(4) = 18$*. As this strategy does not employ AI, the cost is not affected by improvements to* α *in this range, which is evident in Panel (b) of the figure.*

For $\alpha \in (0.25, 0.77)$*, the optimal automation strategy changes: it is better to use AI augmentation for the second step, reducing that step's total completion cost from* $(2)(4)$ *to* $(2)(1/\alpha)$*. As* α *increases, the total execution cost drops gradually from* 18 *(at* $\alpha = 0.25$*) to* $10 + 2/\alpha \approx 12.5$ *(at* $\alpha \approx 0.77$*). This gradual improvement is visible as a modest slope in Panel (b).*

However, once α *grows larger than* 0.77*, the optimal automation strategy and job design changes more dramatically. At this point, is is better to combine both steps into a single AI chain, automating step* 1*, and to assign the resulting aggregate task to a single low-skill worker. The completion cost of this strategy is* $(2)(1/\alpha^7)$*. This cost starts at approximately* 12.5 *at* $\alpha = 0.77$ *but drops sharply, eventually converging to a total cost of* 2 *at* $\alpha = 1$*. This sharp drop indicates substantial marginal benefits from incremental improvements in AI quality in this region, illustrated by the pronounced jump in the slope in Panel (b) at* $\alpha \approx 0.77$*.*

Figure 5: Cost Analysis of Automation Strategies in Example 6



(a) Total Cost by Automation Strategy

(b) Marginal Benefit of Improving AI Quality

Notes: Panel (a) shows total execution costs under various automation strategies as AI quality (α) improves. Panel (b) illustrates the marginal reduction in optimal costs associated with increased AI quality. Dashed vertical lines indicate thresholds where optimal automation strategies shift.

5 Macro-level Production Function

So far, the mode has focused on individual firm decision-making: how a firm allocates production steps among automation, augmentation, and manual execution, and how its micro-level choices of task assignment and job design determine internal productivity and costs. Ultimately, however, our interest extends beyond the firm to the broader economy. Improvements in AI quality not only alter how individual firms organize production but also affect the composition of inputs used across the economy and the aggregate relationship between labor, capital, and output. Yet the firm-level optimization framework developed above prevents direct analysis of these aggregate effects and also makes our approach distinct from the large literature in which production is modeled directly at the task level and then aggregated to the economy level (e.g., Acemoglu et al. (2022, 2024); Acemoglu (2025); Autor and Thompson (2025)). To bridge this gap, this section examines whether the micro-level cost-minimization problem of firms can be aggregated into well-behaved firm- and economy-level production functions. In particular, we show that the detailed, many-input Leontief representation of production at the task level can be reduced to a more compact formulation with only a small number of effective inputs, and that aggregating across heterogeneous firms(in terms of AI quality level) yields a macro-level production function with a CES form. This provides a tractable link between micro-level organization decisions and macroeconomic implications of improving AI quality.

We begin by showing how firm's micro-level decisions can be explicitly represented by a Leontief production function at the task level. Next, we demonstrate how, for the purposes of analyzing labor allocation within a firm, this detailed task-level production function can be simplified into a Leontief production function with just two aggregate inputs: skill-adjusted AI management labor and skill-adjusted human labor (we will describe what we mean by "skill-adjusted" below). Finally, we explore how the production functions of numerous individual firms, which differ in their AI quality levels, can be aggregated into a single, economy-wide CES production function. In this macro-level representation, the inputs are economy-wide aggregates of the corresponding micro-level inputs from individual firms, thereby facilitating clearer insights into the overall labor allocation in the economy.

Before the aggregation analysis, let us first describe the setup of production. A firm uses two production inputs: (1) skill-adjusted AI management labor (to complete AI-assisted tasks), and (2) skill-adjusted human labor (to perform manual tasks). These two types of labor demand different compensations: executing an AI-assisted task with skill and time cost of one demands w_M, while completing a manual task with similar skill and time requirements commands a compensation of w_H. We call w_M and w_H base wage rates for AI management and human labor, respectively.

Firms use these two inputs to complete tasks. Recall that the skill and time requirements of tasks are determined by the firm's automation strategy $\mathcal{T}$, while the aggregate skill requirements of jobs depend jointly on the automation strategy and job design $\mathcal{J}$. Specifically, the per unit of time compensation of a worker employed to do tasks of a job is determined by all tasks in that job. Consider Job 1 in Figure 3 for example. The worker assigned to perform Job 1 is required to obtain not only the skill required for manual Task 1, c_1^H, but must also possess the required skill for AI-assisted Task 2, c_2^M, as the worker's total compensation per unit of time is determined at the job level, and equals $c_1^H + c_2^M$. Therefore, the more tasks firm includes in a job, the higher required compensation for every task in that job.

In order to produce, the firm hires human labor to perform manual tasks and AI management labor for AI-assisted tasks. Therefore, the total compensation of a job will be a weighted average of skill costs of the job's tasks, with weights being the base wage rates of each type of labor. Specifically, the compensation for job J will be:

$$w_H \left(\sum_{T_b^H \in J} c_b^H \right) + w_M \left(\sum_{T_b^M \in J} c_b^M \right), \tag{6}$$

which is composed of the sum of contributions of skill costs from its manual tasks denoted by set T_b^H (each weighted by w_H) and its AI-assisted tasks denoted by set T_b^M (each weighted by w_M).[6]

[6]Notice that this formulation is essentially equivalent to the original definition of wage in (1), as pointed out in Footnote 3. If we multiply each skill cost in (6) by its respective base wage rate and redefine the task's skill cost as the resulting product,

Next, we define skill-adjusted labor, which is the unit of work in our analysis. Let $J(b)$ denote the job to which task b belongs, and $O(b)$ be the mode of execution of task b (i.e., $O(b) = H$ if b is a manual task, or $O(b) = M$ if b is an AI chain). Define the skill-adjusted time requirement of task b as:

$$\tau_b = \frac{w_H \left(\sum_{T_\ell^H \in J(b)} c_\ell^H \right) + w_M \left(\sum_{T_\ell^M \in J(b)} c_\ell^M \right)}{w_{O(b)}} t_b.$$

Note that the fraction appearing behind t_b represents an effective skill adjustment factor, as the numerator captures the total compensation of the job task b belongs to, while the denominator normalizes by the base wage rate for the specific mode of execution. The resulting fraction thus has units of skill intensity. This characterization becomes useful later as it allows expressing the (expected) wage bill paid for task b simply as $w_M \tau_b \alpha^{-d_b}$ if it is AI-assisted or as $w_H \tau_b$ if it is executed manually.

5.1 Within-Firm Aggregation: Leontief to Leontief

We now consider the firm's production. To produce output, each firm must complete requirements of all m production steps $s_1, \ldots, s_m$. This naturally results in a Leontief production structure with multiple inputs.[7] Although firms choose whether to execute each step manually or by the help of AI, we show that analyzing labor allocation can be simplified. Specifically, production function of the firm can be represented by a Leontief function with only two firm-level aggregate inputs: AI management labor and human labor.

Suppose the firm solves the long-run cost minimization problem (3) for a given AI quality level α, and obtains the optimal automation strategy $\mathcal{T}$ and job design $\mathcal{J}$. Let $|\mathcal{T}| = n$, implying the original m production steps are organized into n distinct tasks, and $|\mathcal{J}| = p$, indicating these n tasks are grouped into p jobs. Given the fixed automation strategy, the execution mode for each production step is predetermined. This lets us consider and work directly with tasks rather than individual steps. Furthermore, with the job design held fixed, each firm takes the job boundaries and the total skill requirements within jobs as given.

We assume hand-offs are performed manually by the worker who completes the final task of each job. This allows us to simplify the analysis in two ways: (1) Since job design is fixed, we know exactly which steps occur at the job boundaries and thus incur hand-off time costs. Therefore, hand-off of job J_j can be treated as a standalone, human-executed task with time requirement $t^S(J_j)$ and skill requirement equal to the total skill requirement of job J_j.[8] (2) The order of tasks—including the newly defined hand-off tasks—can be rearranged within the production sequence. Specifically, we relabel tasks such that tasks 1 to k are AI-assisted and thus require AI management labor, while tasks $k+1$ to n, along with the hand-off tasks $n+1$ to $n+p-1$, are executed manually and require human labor.[9]

The firm's task-level production can be expressed in the following Leontief form:

$$x = \min \left\{ \frac{l_1}{\tau_1^M \, \alpha^{-d_1}}, \cdots, \frac{l_k}{\tau_k^M \, \alpha^{-d_k}}, \frac{l_{k+1}}{\tau_{k+1}^H}, \cdots, \frac{l_n}{\tau_n^H}, \frac{l_{n+1}}{\tau^S(J_1)}, \cdots, \frac{l_{n+p-1}}{\tau^S(J_{p-1})} \right\}. \tag{7}$$

The x on the left-hand side represents the firm's output while l_b is the amount of labor assigned to task b. All other variables remain as previously defined.

Since the automation strategy and job design are fixed the required amount of skill-adjusted time to spend on each task in the denominators of (7) are fixed, and the firm takes them as given. In equilibrium, allocation of labor to tasks satisfies

$$x = \frac{l_1}{\tau_1^M \, \alpha^{-d_1}} = \cdots = \frac{l_k}{\tau_k^M \, \alpha^{-d_k}} = \frac{l_{k+1}}{\tau_{k+1}^H} = \cdots = \frac{l_n}{\tau_n^H} = \frac{l_{n+1}}{\tau^S(J_1)} = \cdots = \frac{l_{n+p-1}}{\tau^S(J_{p-1})}.$$

we arrive at the original formulation given in (1). The formulation in (6) allows tasks to contribute differently to the job's wage depending on their mode of execution and type of labor that need to perform them.

[7] Different automation strategies and job designs lead to different input proportions but share the same fundamental Leontief form.

[8] Denote the skill-adjusted time cost of hand-off task of job J_j with $\tau^S(J_j)$.

[9] Recall that the final job p requires no hand-off by definition. The last hand-off thus occurs at the end of job $p-1$.

Notice that the ratio of labor allocated to any two tasks a and b is independent of output level and wage rates (regardless of whether tasks a and b are executed manually or by the help of AI), and solely depends on their relative skill-adjusted time requirements, which are fixed given $\mathcal{T}$ and $\mathcal{J}$. The fixed ratio of labor inputs implies that the rate of substitution between any pair of tasks is independent of the labor allocated to other tasks:

$$\frac{\partial}{\partial l_z}\left(\frac{\frac{\Delta x}{\Delta l_a}}{\frac{\Delta x}{\Delta l_b}}\right) = 0, \qquad \forall z \neq a, b. \tag{8}$$

A necessary and sufficient condition to aggregate the firm's production function from a Leontief with one input per task into a Leontief with (firm-level) aggregate human labor and AI management labor is that the rate of substitution between any two tasks within the same aggregate input type is independent of all tasks in the other aggregate input type (Leontief (1947); Fisher (1965); Felipe and Fisher (2003)). In other words, the relative amount of labor allocated to any two human-executed (AI-managed) tasks should depend exclusively on tasks within the human-executed (AI-managed) group.

The condition expressed in (8) satisfies the aggregation requirement above.[10] Thus, the firm's production function can be represented as:

$$x = \min\left\{\frac{\bar{\alpha}\, l_M}{\tau_M}, \frac{l_H}{\tau_H}\right\}. \tag{9}$$

Here, l_M and l_H represent respectively the firm-level aggregate AI management labor and human labor. Moreover, the aggregate skill-adjusted AI and human time requirements, τ_M and τ_H, are defined as the sum of skill-adjusted time requirements of tasks in their respective groups:

$$\tau_M = \sum_{b=1}^{k} \tau_b^M, \tag{10}$$

$$\tau_H = \sum_{j=1}^{p-1} \tau^S(J_j) + \sum_{b=k+1}^{n} \tau_b^H. \tag{11}$$

Finally, $\bar{\alpha}$ is the firm's effective AI quality level defined as

$$\bar{\alpha} = \frac{\sum_{b=1}^{k} \tau_b^M}{\sum_{b=1}^{k} \tau_b^M \alpha^{-d_b}}, \tag{12}$$

so that the following relationship holds:

$$\frac{\tau_M}{\bar{\alpha}} = \sum_{b=1}^{k} \tau_b^M \alpha^{-d_b}.$$

In short, (9) allows us to analyze labor allocation using the simpler two-input Leontief production function for a given automation strategy $\mathcal{T}$ and job design $\mathcal{J}$. This firm-level aggregate Leontief function implies the equilibrium condition:

$$x = \frac{\bar{\alpha}\, l_M}{\tau_M} = \frac{l_H}{\tau_H}. \tag{13}$$

The representation in (9) helps us gain clearer insights into the allocation between human and AI management labor at the firm level by removing the need to deal with the complex, many-input production function (7) at the level of steps.

[10] In fact, this condition is stricter than necessary. It not only guarantees labor independence across aggregated input types, but also imposes a stronger condition: that the rate of substitution between tasks within the same aggregate input type is independent even of other tasks within the same aggregate type.

5.2 Cross-Firm Aggregation: Leontief to CES

Next, we analyze whether Leontief production functions of many firms can be collectively represented by a single economy-wide production function, where each input in the macro function aggregates the corresponding inputs from the micro functions. To do so, we draw on the extensive literature on aggregation theorems in production theory, which establishes conditions for the existence of an aggregate production function based on individual firms' production functions. The central idea is to introduce an appropriate form of heterogeneity across firms, allowing micro-level production functions to be smoothed into a well-behaved aggregate function (See Sato (1975), Chapters 2 and 4, for existence conditions.).[11] While most existence results in this literature do not yield a closed-form functional representation, we show here that under certain conditions on firm heterogeneity, firm-level Leontief production functions can be aggregated into an economy-wide CES production function.

Consider a unit mass of firms in the economy, each producing output with AI management labor, human labor, and capital. We assume capital is a fixed input, so firms make labor decisions conditional on their given capital stock. Moreover, we impose a Leontief structure between capital and labor inputs, normalizing the production process so that exactly one unit of capital is required per unit of output produced.[12] Firms do not know their individual effective AI quality levels before production occurs; instead, they share a common prior belief regarding the distribution from which these quality levels are drawn.

Production unfolds in two stages. In the first stage, firms solve the long-run cost minimization problem (3), committing to an automation strategy $\mathcal{T}$ and a job design $\mathcal{J}$ based on their *expected* AI quality levels. Since firms hold the same expectations about the distribution of effective AI quality, they choose identical automation strategies and job designs. In the second stage, firms learn their realized effective AI quality levels, hire the required labor and capital, and begin production.[13] The resulting Leontief production function, determined by the previously selected automation strategy, job design, and the firm's realized effective AI quality, takes a form similar to (9).

This two-stage structure mirrors realistic firm behavior. Firms initially decide how to structure operations and post job vacancies. Only after making these structural commitments do they hire workers and acquire machines to begin production. Once hiring occurs, firms discover the actual productivity levels of inputs, which necessitates operating under their previously chosen organizational structures.

In what follows, we demonstrate that the heterogeneity in effective AI quality level can be structured so that micro-level Leontief production functions (9) aggregate into a macro-level CES production function, in which the CES inputs are aggregates of the micro-level inputs (Levhari (1968)). Suppose specifically that the macro-level production function takes the form:

$$X = \left(\theta_M \, M^\rho + \theta_H \, H^\rho + (1 - \theta_M - \theta_H) \, K^\rho\right)^{\frac{1}{\rho}}, \tag{14}$$

where X represents (economy-wide) aggregate output, M denotes aggregate AI management labor, H denotes aggregate human labor, K is aggregate capital, and parameters θ_M and θ_H represent the weights of corresponding inputs. Since the measure of firms in the economy is normalized to 1, aggregate capital is also normalized to 1, making the third term in (14) effectively constant at $1 - \theta_M - \theta_H$.

To link the macro production function above to the micro production functions, we must relate aggregated firm-level inputs l_M, l_H from (9) to the economy-wide aggregates M, H in (14). This requires either determining macro parameters from the distribution of heterogeneity, or specifying the form of heterogeneity given a set of macro-level parameters. We adopt the latter approach and derive the output probability density function in terms of effective AI quality, denoted by $\phi(\bar{\alpha})$, as a function of θ_M, θ_H, ρ. Throughout, we assume that $\rho < 0$ (which implies elasticity of substitution $\sigma < 1$), indicating that macro-level production exhibits some degree of complementarity.

[11] For instance, Houthakker (1955) studies aggregation from micro Leontief functions to a macro Cobb-Douglas function; Levhari (1968) investigates aggregation from micro Leontief functions to a macro CES function; and Sato (1969) explores aggregation from micro CES to macro CES functions. See Baqaee and Farhi (2019) for a broader formulation of this problem.

[12] This normalization can be interpreted as assuming identical capital productivity across firms. Since we introduce firm-level heterogeneity through variations in AI quality, we abstract away from additional sources of heterogeneity across firms.

[13] As production depends on the realized effective AI quality level, $\bar{\alpha}$ also determines firms' sizes.

Normalize the output price to $p = 1$, so that w_M and w_H can be interpreted as real wage rates for a unit of skill-adjusted AI management and human labor, respectively. A firm produces only if it earns nonnegative profits:

$$w_M\, l_M + w_H\, l_H \leq x.$$

Substituting for l_H and x from (13) into the profitability condition yields:

$$w_M\, l_M + w_H\, \frac{\tau_H\, \bar{\alpha}\, l_M}{\tau_M} \leq \frac{\bar{\alpha}\, l_M}{\tau_M}.$$

Rearranging terms and canceling l_M gives the lower bound of firms' effective AI quality level:[14]

$$\bar{\alpha} \geq \frac{w_M\, \tau_M}{1 - w_H\, \tau_H}.$$

This lower bound implies that only firms with realized effective AI quality level above this threshold will produce in equilibrium and others exit the market. Moreover, observe from (12) that for $d_b \geq 0$ the upper bound of $\bar{\alpha}$ is 1 and is achieved when $\alpha \to 1^-$. Therefore:

$$\frac{w_M\, \tau_M}{1 - w_H\, \tau_H} \leq \bar{\alpha} \leq 1. \tag{15}$$

Let $\phi(\bar{\alpha})$ be the distribution of output across firms according to their effective AI quality level. A firm with effective AI quality $\bar{\alpha}$ thus produces output $x = \phi(\bar{\alpha})$ by definition. From equation (13), it directly follows that the AI management labor and human labor used by this firm are: $l_M = (\tau_M/\bar{\alpha})\phi(\bar{\alpha})$, and $l_H = \tau_H\, \phi(\bar{\alpha})$. Consequently, aggregate output X, aggregate AI management labor M, and aggregate human labor H can be expressed as:

$$X = \int_{\frac{w_M\, \tau_M}{1 - w_H\, \tau_H}}^{1} \phi(\bar{\alpha})\, d\bar{\alpha}, \tag{16}$$

$$M = \int_{\frac{w_M\, \tau_M}{1 - w_H\, \tau_H}}^{1} \tau_M\, \frac{\phi(\bar{\alpha})}{\bar{\alpha}}\, d\bar{\alpha}, \tag{17}$$

$$H = \int_{\frac{w_M\, \tau_M}{1 - w_H\, \tau_H}}^{1} \tau_H\, \phi(\bar{\alpha})\, d\bar{\alpha}. \tag{18}$$

Substituting the aggregated variables from the micro-level equations (16, 17, 18) into the aggregate production function (14), we obtain:

$$\int_{\frac{w_M\, \tau_M}{1 - w_H\, \tau_H}}^{1} \phi(\bar{\alpha})\, d\bar{\alpha} = \left(\theta_M \left(\tau_M \int_{\frac{w_M\, \tau_M}{1 - w_H\, \tau_H}}^{1} \frac{\phi(\bar{\alpha})}{\bar{\alpha}}\, d\bar{\alpha} \right)^{\rho} + \theta_H \left(\tau_H \int_{\frac{w_M\, \tau_M}{1 - w_H\, \tau_H}}^{1} \phi(\bar{\alpha})\, d\bar{\alpha} \right)^{\rho} + (1 - \theta_M - \theta_H) \right)^{\frac{1}{\rho}}. \tag{19}$$

Solving for $\phi(\bar{\alpha})$ in terms of θ_M, θ_H, ρ we obtain the following distribution for effective AI quality:

$$\phi(\bar{\alpha}) = \frac{(1 - \theta_M - \theta_H)^{\frac{1}{\rho}}\, (1 - \theta_H\, \tau_H^{\rho})^{\frac{\rho}{\rho - 1}}\, (\theta_M\, \tau_M^{\rho})^{\frac{1}{1 - \rho}}}{1 - \rho} (\bar{\alpha})^{\frac{1}{\rho - 1}} \left[1 - \theta_H\, \tau_H^{\rho} - (1 - \theta_H\, \tau_H^{\rho})^{\frac{\rho}{\rho - 1}}\, (\theta_M\, \tau_M^{\rho})^{\frac{1}{1 - \rho}}\, (\bar{\alpha})^{\frac{\rho}{\rho - 1}} \right]^{-\frac{1 + \rho}{\rho}}. \tag{20}$$

See Appendix B for derivation details.

This completes the production function aggregation procedure. The derived distribution (20) characterizes firm-level heterogeneity in the effective AI quality level required to support a macro CES production function of the form (14), with economy-wide aggregate human and AI management labor inputs and parameters θ_M, θ_H, ρ, obtained from micro-level Leontief production functions of the form (9).

[14] We assume the parameters w_M, w_H, τ_M, τ_H are such that $0 < (w_M\, \tau_M)/(1 - w_H\, \tau_H) < 1$.

References

Acemoglu, Daron, "The simple macroeconomics of AI," *Economic Policy*, 2025, *40* (121), 13–58.

_ **and Pascual Restrepo**, "Automation and new tasks: How technology displaces and reinstates labor," *Journal of Economic Perspectives*, 2019, *33* (2), 3–30.

_ **, David Autor, Jonathon Hazell, and Pascual Restrepo**, "Artificial intelligence and jobs: Evidence from online vacancies," *Journal of Labor Economics*, 2022, *40* (S1), S293–S340.

_ **, Fredric Kong, and Pascual Restrepo**, "Tasks At Work: Comparative Advantage, Technology and Labor Demand," Working Paper 32872, National Bureau of Economic Research August 2024.

Agrawal, Ajay, Joshua S Gans, and Avi Goldfarb, "Exploring the impact of artificial intelligence: Prediction versus judgment," *Information Economics and Policy*, 2019, *47*, 1–6.

Autor, David and Neil Thompson, "Expertise," *Journal of the European Economic Association*, 06 2025, *23* (4), 1203–1271.

Autor, David H and Michael J Handel, "Putting tasks to the test: Human capital, job tasks, and wages," *Journal of labor Economics*, 2013, *31* (S1), S59–S96.

_ **, Frank Levy, and Richard J Murnane**, "The skill content of recent technological change: An empirical exploration," *The Quarterly journal of economics*, 2003, *118* (4), 1279–1333.

Baqaee, David Rezza and Emmanuel Farhi, "JEEA-FBBVA Lecture 2018: The Microeconomic Foundations of Aggregate Production Functions," *Journal of the European Economic Association*, 10 2019, *17* (5), 1337–1392.

Bresnahan, Timothy F, Erik Brynjolfsson, and Lorin M Hitt, "Information technology, workplace organization, and the demand for skilled labor: Firm-level evidence," *The quarterly journal of economics*, 2002, *117* (1), 339–376.

Eloundou, Tyna, Sam Manning, Pamela Mishkin, and Daniel Rock, "Gpts are gpts: An early look at the labor market impact potential of large language models," *arXiv preprint arXiv:2303.10130*, 2023.

Felipe, Jesus and Franklin M Fisher, "Aggregation in production functions: what applied economists should know," *Metroeconomica*, 2003, *54* (2-3), 208–262.

Felten, Edward, Manav Raj, and Robert Seamans, "Occupational, industry, and geographic exposure to artificial intelligence: A novel dataset and its potential uses," *Strategic Management Journal*, 2021, *42* (12), 2195–2217.

Fisher, Franklin M, "Embodied technical change and the existence of an aggregate capital stock," *The Review of Economic Studies*, 1965, *32* (4), 263–288.

Frey, Carl Benedikt and Michael A Osborne, "The future of employment: How susceptible are jobs to computerisation?," *Technological forecasting and social change*, 2017, *114*, 254–280.

Houthakker, Hendrik S, "The Pareto Distribution and The Cobb-Douglas Production Function in Activity Analysis," *The Review of Economic Studies*, 1955, *23* (1), 27–31.

Kremer, Michael, "The O-Ring Theory of Economic Development*," 08 1993.

Leontief, Wassily, "Introduction to a theory of the internal structure of functional relationships," *Econometrica, Journal of the Econometric Society*, 1947, pp. 361–373.

Levhari, David, "A note on Houthakker's aggregate production function in a multifirm industry," *Econometrica: journal of the Econometric Society*, 1968, pp. 151–154.

Sato, Kazuo, "Micro and Macro Constant-Elasticity-of-Substitution Production Functions in a Multifirm Industry," *Journal of Economic Theory*, 1969, *1* (4), 438–453.

_ , *Production Functions and Aggregation*, Vol. 90, North-Holland Publishing Company, 1975.

A Technical Details - Fragmentation Index

In this appendix we prove Proposition 5, which relates the cost of the optimal short-term automation strategy for a fixed job to the fragmentation index of that job. Intuitively, we expect a production process in which automatable tasks are clustered together will benefit more from automation than one where steps with high and low levels of automation are interspersed. This is due to the potential benefits from AI chains.

First recall the definition of fragmentation index for a given step sequence $\mathcal{S} = (s_1, \ldots, s_m)$. As a reminder, we assume for Proposition 5 that $t_i^M = 1$ for all s_i. Consider a random process in which each step s_i succeeds independently with probability q_i; any step that does not succeed is said to fail. Write F for the set of steps that fail, and $\mathcal{C} = \{C_1, \ldots, C_k\}$ for the random variable representing the collection of maximal connected components of non-failed steps. The weight of each $C_j \in \mathcal{C}$ is defined to be $\omega(C_j) = \min\{1, \sum_{s_i \in C_j} t_i^H\}$. That is, each C_j has weight 1 unless the sum of the manual time costs for each of its steps is less than 1 (due to our assumption that AI management costs are normalized to 1).

We now introduce some notation: for each s_i, write $c_i = \min\left\{t_i^H, \frac{t_i^M}{q_i}\right\}$. That is, c_i is the minimum cost to implement step s_i individually (whether manually or through AI augmentation). Given a realization of $\mathcal{C}$ and F, we define the realized fragmentation to be

$$\sum_{s_i \in F} c_i + \sum_{C_j \in \mathcal{C}} \omega(C_j).$$

The *fragmentation index* is defined to be the expected value of the realized fragmentation.

We now fix the sequence $\mathcal{S} = (s_1, \ldots, s_m)$, write FI for the fragmentation index of $\mathcal{S}$, and let OPT be the cost of the optimal (i.e., time-minimizing) task structure for those steps.

We begin by showing that FI is not much larger than OPT.

Proposition 6. $FI \leq \frac{5}{4} OPT$.

Proof. Fix an arbitrary task sequence $\mathcal{T} = (T_1, \ldots, T_n)$. Partition this into two sets of tasks: $\mathcal{T}_H$ containing all (singleton) tasks that are completed manually, and $\mathcal{T}_M$ containing all other tasks (consisting of AI-augmented singletons and AI chains). We first claim that

$$FI \leq \sum_{T_j \in \mathcal{T}_M} \left(1 + \sum_{s_i \in T_j} (1 - \alpha^{d_i})(1 + c_i)\right) + \sum_{(s_i) \in \mathcal{T}_M} t_i^H.$$

To see why, first note that for any human-executed task $(s_i) \in \mathcal{T}_H$, either the task fails in the realization of F (in which case it contributes $c_i \leq t_i^H$ to FI), or it succeeds (in which case it appears in some connected component C_j, contributing at most t_i^H to the component's weight $\omega(C_j)$).

Next consider the machine-assisted tasks. For any $T_j \in \mathcal{T}_M$, consider the set of steps in T_j that fail in any given realization of F. If no steps in T_j fail, then T_j collectively contributes at most 1 to the realized fragmentation (since all $s_i \in T_j$ lie in the same connected component of non-failed tasks). Otherwise, each failed task $s_i \in T_j$ contributes c_i to the realized fragmentation (for itself, recalling that c_i is the minimal cost of executing s_i on its own) plus an additional value of at most 1 for the weight of a potential new connected component of non-failed tasks. As each task $s_i \in T_j$ fails independently with probability $(1 - \alpha^{d_i})$, we get that the contribution of tasks in T_j to the fragmentation index is at most $1 + \sum_{s_i \in T_j} (1 - \alpha^{d_i})(1 + c_i)$ as claimed.

On the other hand, recall that if we take $\mathcal{T}$ to be the cost-minimizing task sequence, then

$$OPT = \sum_{T_j \in \mathcal{T}_M} \alpha^{-d(T_j)} + \sum_{(s_i) \in \mathcal{T}_H} t_i^H$$

by definition, where we write $d(T_j) = \sum_{s_i \in T_j} d_i$ for the total difficulty of steps in task T_j.

Comparing our expression for OPT with our bound on FI, we see that each task in $\mathcal{T}_H$ contributes the same amount to each, whereas each $T_j \in \mathcal{T}_M$ contributes $\sum_{s_i \in T_j} (1 - \alpha^{d_i})(1 + c_i)$ to the former and $\alpha^{-d(T_j)}$

to the latter. We will show that, for each $T_j \in \mathcal{T}_M$, $1 + \sum_{s_i \in T_j}(1-\alpha^{d_i})(1+c_i) \leq \frac{5}{4}\alpha^{-d(T_j)}$, which will complete the proof.

First, since $c_i \leq \alpha^{-d_i}$ for each $s_i \in \mathcal{S}$, we have $1+\sum_{s_i \in T_j}(1-\alpha^{d_i})(1+c_i) \leq 1+\sum_{s_i \in T_j}(1-\alpha^{d_i})(1+\alpha^{-d_i})$.

Next note that for any $x, y \in [0,1]$, $(1-xy)(1+\frac{1}{xy}) \geq (1-x)(1+\frac{1}{x})+(1-y)(1+\frac{1}{y})$. (This can be checked by expanding and taking derivatives.) Repeatedly applying this fact, we get that $1+\sum_{s_i \in T_j}(1-\alpha^{d_i})(1+\alpha^{-d_i}) \leq 1+(1-\alpha^{d(T_j)})(1+\alpha^{-d(T_j)}) = 1+\alpha^{-d(T_j)}-\alpha^{d(T_j)}$ for any $T_j \in \mathcal{T}_M$. The ratio between this expression and $\alpha^{-d(T_j)}$ is maximized at $\alpha^{-d(T_j)} = 1/2$, achieving a value of $5/4$ as claimed. □

Note that this bound of $5/4$ is tight. Suppose we have 3 steps in $\mathcal{S}$: the first and last always succeed, and the second succeeds with probability $1/2$ and has a manual cost of 2. Then the optimal policy automates all three together for an expected cost of 2, whereas the fragmentation index is $(1/2)\times 1+(1/2)\times(1+2+1) = 5/2$.

We next argue that F is not much smaller than OPT. We begin with an analysis under the assumption that $t_i^H \geq 1$ for all $s_i \in \mathcal{S}$. That is, for each step s_i, completing the step manually is at least as costly as managing an AI tool to complete the step in a single attempt (with guaranteed success).

Proposition 7. *Suppose $t_i^H \geq 1$ for all $s_i \in \mathcal{S}$. Then $FI \geq \frac{1}{4}OPT$.*

Proof. Consider the following task sequence. Beginning with the first step s_1, consider the maximal contiguous sequence of steps T whose success probability $\alpha^{d(T)}$ is greater than or equal to $1/2$. If that set is empty (i.e., the first step has success probability strictly less than $1/2$) then the first step is set to be completed individually, either manually or automated, whichever is cheaper. In this case, we'll say the resulting singleton task is completed *individually*. Otherwise, the sequence $T = (s_1, \ldots, s_\ell)$ (which could still be a singleton task) is added to the task sequence as an AI chain; we call this a *non-individual* task. This process is then repeated beginning with the next step (which is s_2 in the former case, or $s_{\ell+1}$ in the latter case), until all steps have been added to a task. Call the resulting task sequence $\mathcal{T}$.

Let ALG denote the cost of the task sequence $\mathcal{T}$. Note that we must have $ALG \geq OPT$. We will argue that $FI \geq ALG/4$, which will prove the claim.

We first define some notation. Write $\mathcal{T}_I$ for the set of individual tasks in $\mathcal{T}$ and $\mathcal{T}_{NI}$ for the set of non-individual tasks. Note that $\alpha^{d_i} < 1/2$ for all $(s_i) \in \mathcal{T}_I$, by construction. Note also that $\mathcal{T}_H \subseteq \mathcal{T}_I$ (recalling that $\mathcal{T}_H$ is the set of all singleton tasks executed manually), and this containment may be strict if $\mathcal{T}_I$ contains singleton tasks that are augmented. We emphasize that $\mathcal{T}_{NI}$ may still include singleton tasks, but any singleton task $(s_i) \in \mathcal{T}_{NI}$ must have the property that $\alpha^{d_i} \geq 1/2$.

If the final task s_m from $\mathcal{S}$ is contained in some $T_j \in \mathcal{T}_{NI}$, we say that T_j is the *terminal* task. All other tasks in $\mathcal{T}_{NI}$ are non-terminal tasks. Note that if the final step is a singleton task in $\mathcal{T}_I$ then no task is terminal. For each non-terminal task $T_j \in \mathcal{T}_{NI}$, we'll write $\overline{T}_j$ to denote T_j plus the first step that follows T_j. For example, if $T_j = (s_i, \ldots, s_\ell)$, then $\overline{T}_j = (s_i, \ldots, s_\ell, s_{\ell+1})$. The set $\mathcal{T}_{NI}$ then has the property that for each task $T_j \in \mathcal{T}_{NI}$ we have $\alpha^{d(T_j)} \geq 1/2$, and for each non-terminal $T_j \in \mathcal{T}_{NI}$ we have $\alpha^{(d(\overline{T}_j))} < 1/2$.

We are now ready to relate FI and ALG. Let $k = |\mathcal{T}_{NI}|$ be the number of non-individual tasks. Note first that

$$ALG = \sum_{T_j \in \mathcal{T}_{NI}} \alpha^{-d(T_j)} + \sum_{(s_i) \in \mathcal{T}_I} c_i \leq 2k + \sum_{(s_i) \in \mathcal{T}_I} c_i.$$

This is because $\alpha^{d(T_j)} \geq 1/2$ for each $T_j \in \mathcal{T}_{NI}$, and hence $\alpha^{-d(T_j)} \leq 2$.

Next we bound FI. Note that the assumption $t_i^H \geq 1$ implies that, in any realization of the connected components of non-failed steps $\mathcal{C} = (C_1, \ldots, C_\ell)$, $\omega(C_j) = 1$ for all j. This implies that the realized fragmentation is equal to 1 plus, for each $s_i \in F$ (i.e., each step s_i that fails), a contribution of c_i plus an additional 1 in the event that $i > 1$ and the task immediately preceding s_i did not fail. (This additional cost of 1 accounts for creating a new connected component of non-failed steps ending with s_{i-1}.)

We claim that $FI \geq k/2 + \frac{1}{2}\sum_{(s_i) \in \mathcal{T}_I} c_i$. We will show this by charging the realized fragmentation to tasks in $\mathcal{T}_{NI}$ and $\mathcal{T}_I$, so that each $T_j \in \mathcal{T}_{NI}$ is charged at least 1 with probability at least $1/2$, and each $(s_i) \in \mathcal{T}_I$ is charged at least c_i with probability at least $1/2$.

To see this, let E_j denote the event that some step in $\overline{T}_j$ fails, for each non-terminal $T_j \in \mathcal{T}_{NI}$. Note that E_j occurs with probability at least $1/2$, since $\alpha^{d(\overline{T}_j)} < 1/2$. Also, if E_j occurs, then either a step $s_i \in T_j$ fails, or no step in T_j fails but the step immediately following T_j does. In the former case, we will charge the $c_i \geq 1$ contribution of s_i's failure to T_j. In the latter case, it must be that the step immediately preceding the failed task did not fail; so we will charge the additional contribution of 1 from s_i's failure (as described above) to T_j. Additionally, if there is a terminal T_j in $\mathcal{T}_{NI}$, then we charge the baseline 1 from the calculation of FI to that terminal T_j. Aggregating over all these cases, we have that at least 1 is charged to each set $T_j \in \mathcal{T}_{NI}$ with probability at least $1/2$. Finally, for each $(s_i) \in \mathcal{T}_I$ that fails (which occurs with probability at least $1/2$, by construction), we charge the c_i portion of its contribution to task (s_i). We conclude that $FI \geq k/2 + \frac{1}{2}\sum_{(s_i)\in\mathcal{T}_I} c_i$ as claimed.

But now since $FI \geq k/2 + \frac{1}{2}\sum_{(s_i)\in\mathcal{T}_I} c_i$ and $ALG \leq 2k + \sum_{(s_i)\in\mathcal{T}_I} c_i$, we conclude $FI \geq ALG/4$ as claimed. □

Can the approximation factor in Proposition 7 be improved? While we do not have a matching lower bound of 4, we do know that the approximation factor is at least $\frac{2}{3}(2+\sqrt{2}) \approx 2.276$. Here is an example. Suppose the problem instance is a sequence of m steps, each with success probability $1/\sqrt{2}$ and manual cost $\sqrt{2}$. The task sequence described in the proof of Proposition 7 then handles each task separately, for a total cost of $m\sqrt{2}$. The optimal task sequence must therefore perform at least this well. The fragmentation index is such that each task contributes $\sqrt{2}$ if it fails, plus 1 if the preceding task did not fail, for a total contribution of $(1-1/\sqrt{2})(\sqrt{2}+1(1/\sqrt{2}) = \frac{3}{2}(\sqrt{2}-1) \approx 0.6213$. As m grows large, the ratio between $m\sqrt{2}$ and $1 + m \times 0.6213$ approaches $\frac{2}{3}(2+\sqrt{2}) \approx 2.276$.

We can relax the assumption in Proposition 7 that $t_i^H \geq 1$ for all s_i, but at the cost of a worse approximation factor.

Proposition 8. *For general t_i^H (i.e., allowing $t_i^H < 1$ for some steps s_i), $FI \geq \frac{1}{8}OPT$.*

Proof. The argument follows the proof of Proposition 7, with the following changes.

First we make a slight change in the definition of the task sequence $\mathcal{T}$ that defines ALG. If a constructed task T_j has the property that $\sum_{s_i\in T_j} t_i^H < 1$ (which implies it is cheaper to run all steps of T_j manually than as an AI chain that is guaranteed to succeed), we switch to running the tasks of T_j manually in the task sequence. In our analysis, we still consider T_j to be a set in $\mathcal{T}_{NI}$; but its cost is taken to be $\sum_{s_i\in T_j} t_i^H$.

This modification changes our upper bound on the total cost of ALG to the following:

$$ALG \leq 2 \sum_{T_j\in\mathcal{T}_{NI}} \min\left\{1, \sum_{s_i\in T_j} t_i^H\right\} + \sum_{(s_i)\in\mathcal{T}_I} c_i.$$

Then, to bound FI, we claim that

$$FI \geq \frac{1}{4} \sum_{T_j\in\mathcal{T}_{NI}} \min\left\{1, \sum_{s_i\in T_j} t_i^H\right\} + \frac{1}{2} \sum_{(s_i)\in\mathcal{T}_I} c_i$$

which would prove the claim.

To see why this bound on FI holds, we employ a charging argument as before. Recall that for each non-terminal $T_j \in \mathcal{T}_{NI}$, the event E_j occurs with probability at least $1/2$. When it does, we charge to T_j the contribution c_i to FI from any $s_i \in F$ that lies in T_j. Furthermore, when E_j occurs, for each connected component C_ℓ that intersects T_j we additionally charge the contribution $\omega(C_\ell)$ from FI to T_j. Crucially, the contribution from each C_ℓ can be charged to at most two different tasks T_j in this way: once for the leftmost T_j that it intersects, and once for the rightmost T_j that it intersects. (The reason is that if some T_j is a subset of C_ℓ, then by definition no step of T_j fails and hence E_j did not occur. So this charging can only occur for a task T_j that intersects C_ℓ but is not a subset of C_ℓ, of which there are at most two.)

Moreover, this charging to T_j adds up to a total value of at least $\min\left\{1, \sum_{s_i \in T_j} t_i^H\right\}$. Since the charging occurs with probability 1/2 for each $T_j \in \mathcal{T}_{NI}$, and we are at most double-counting each $\omega(C_\ell)$, we conclude that the expected sum of all $\omega(C_\ell)$, plus c_i for all failed tasks s_i that lie in some $T_j \in \mathcal{T}_{NI}$, is at least 1/4 of $\sum_{T_j \in \mathcal{T}_{NI}} \min\left\{1, \sum_{s_i \in T_j} t_i^H\right\}$.

The charging for $(s_i) \in \mathcal{T}_I$ remains unchanged: the value c_i is charged in the event that step s_i fails, which occurs with probability at least 1/2.

Taken together, we obtain our desired 8 approximation. □

We suspect the factor of 8 in Proposition 8 is not tight. But, as we now show, the factor is strictly greater than the factor 4 from Proposition 7, so the assumption that $t_i^H \geq 1$ for all s_i is necessary for Proposition 7 to hold.

Consider a step sequence $\mathcal{S}$ that alternates between (a) steps of manual cost 0 and success probability $1/\sqrt{2} - \epsilon$ where ϵ is vanishingly small, and (b) steps of manual cost 1 and success probability 1. Suppose there are $K > 1$ such pairs. The task sequence described in the proof of Proposition 8 above groups the steps into pairs of two-step tasks, for a total cost of $K\sqrt{2}$ (ignoring ϵ). The fragmentation index is 1 plus 1 for each task that fails, which is $1 + K(1 - 1/\sqrt{2})$ (again ignoring the impact of ϵ). As K grows large, the ratio tends to $2(\sqrt{2} + 1)$ which is strictly greater than 4.

B Technical Details - Derivation of Effective AI Quality Heterogeneity Distribution

In this Appendix we describe how to obtain the analytical formula for distribution of heterogeneity $\phi(\bar{\alpha})$ expressed in (20) by extending the work of Levhari (1968).

Define $u = (w_M\,\tau_M)/(1 - w_H\,\tau_H)$ and rewrite equation (19) as:

$$\left(\int_u^1 \phi(\bar{\alpha})\,d\bar{\alpha}\right)^{\rho} = \theta_M \left(\tau_M \int_u^1 \frac{\phi(\bar{\alpha})}{\bar{\alpha}}\,d\bar{\alpha}\right)^{\rho} + \theta_H \left(\tau_H \int_u^1 \phi(\bar{\alpha})\,d\bar{\alpha}\right)^{\rho} + (1 - \theta_M - \theta_H)\,. \tag{21}$$

Our goal is to solve for $\phi(\bar{\alpha})$ in the equation above as a function of θ_M, θ_H, ρ. Define:

$$\Gamma(u) = \int_u^1 \phi(\bar{\alpha})\,d\bar{\alpha}, \tag{22}$$

and

$$\Psi(u) = \int_u^1 \frac{\phi(\bar{\alpha})}{\bar{\alpha}}\,d\bar{\alpha}. \tag{23}$$

Note that $\Gamma^{'}(u) = -\phi(u)$ and $\Psi^{'}(u) = -\phi(u)/u$. Rewrite equation (21) as:

$$\Gamma^{\rho}(u) = \theta_M\,\tau_M^{\rho}\,\Psi^{\rho}(u) + \theta_H\,\tau_H^{\rho}\,\Gamma^{\rho}(u) + (1 - \theta_M - \theta_H).$$

Rearranging terms, we obtain:

$$(1 - \theta_H\,\tau_H^{\rho})\,\Gamma^{\rho}(u) = \theta_M\,\tau_M^{\rho}\,\Psi^{\rho}(u) + (1 - \theta_M - \theta_H). \tag{24}$$

Differentiating equation (24) with respect to u yields:

$$(1 - \theta_H\,\tau_H^{\rho})\,\Gamma^{\rho-1}(u) = \theta_M\,\tau_M^{\rho}\,\Psi^{\rho-1}(u)\,u^{-1}.$$

Rearrange this to solve for $\Psi^{\rho-1}(u)$ as a function of u and $\Gamma^{\rho-1}(u)$:

$$\Psi^{\rho-1}(u) = \frac{(1 - \theta_H\,\tau_H^{\rho})}{\theta_M\,\tau_M^{\rho}}\Gamma^{\rho-1}(u)\,u.$$

Finally, express $\Psi(u)$ in terms of $\Gamma(u)$:

$$\Psi(u) = (1 - \theta_H\,\tau_H^{\rho})^{\frac{1}{\rho-1}}\,(\theta_M\,\tau_M^{\rho})^{\frac{1}{1-\rho}}\,u^{\frac{1}{\rho-1}}\,\Gamma(u). \tag{25}$$

Substituting $\Psi(u)$ from equation (25) into equation (24), we have:

$$(1 - \theta_H\,\tau_H^{\rho})\,\Gamma^{\rho}(u) = (1 - \theta_H\,\tau_H^{\rho})^{\frac{\rho}{\rho-1}}\,(\theta_M\,\tau_M^{\rho})^{\frac{1}{1-\rho}}\,u^{\frac{\rho}{\rho-1}}\,\Gamma^{\rho}(u) + (1 - \theta_M - \theta_H)\,.$$

Rearranging terms to solve for $\Gamma(u)$, we get:

$$\Gamma(u) = (1 - \theta_M - \theta_H)^{\frac{1}{\rho}} \left[1 - \theta_H\,\tau_H^{\rho} - (1 - \theta_H\,\tau_H^{\rho})^{\frac{\rho}{\rho-1}}\,(\theta_M\,\tau_M^{\rho})^{\frac{1}{1-\rho}}\,u^{\frac{\rho}{\rho-1}}\right]^{-\frac{1}{\rho}}.$$

Recall that $\Gamma'(u) = -\phi(u)$. Differentiating $\Gamma(u)$, we obtain:

$$\begin{aligned}\Gamma'(u) &= \frac{\partial}{\partial u}\left[(1 - \theta_M - \theta_H)^{\frac{1}{\rho}} \left[1 - \theta_H\,\tau_H^{\rho} - (1 - \theta_H\,\tau_H^{\rho})^{\frac{\rho}{\rho-1}}\,(\theta_M\,\tau_M^{\rho})^{\frac{1}{1-\rho}}\,u^{\frac{\rho}{\rho-1}}\right]^{-\frac{1}{\rho}}\right] \\ &= \frac{(1 - \theta_M - \theta_H)^{\frac{1}{\rho}}\,(1 - \theta_H\,\tau_H^{\rho})^{\frac{\rho}{\rho-1}}\,(\theta_M\,\tau_M^{\rho})^{\frac{1}{1-\rho}}}{\rho - 1} u^{\frac{1}{\rho-1}} \left[1 - \theta_H\,\tau_H^{\rho} - (1 - \theta_H\,\tau_H^{\rho})^{\frac{\rho}{\rho-1}}\,(\theta_M\,\tau_M^{\rho})^{\frac{1}{1-\rho}}\,u^{\frac{\rho}{\rho-1}}\right]^{-\frac{1+\rho}{\rho}}.\end{aligned}$$

Thus, we have:

$$\phi(\bar{\alpha}) = \frac{(1 - \theta_M - \theta_H)^{\frac{1}{\rho}}\,(1 - \theta_H\,\tau_H^{\rho})^{\frac{\rho}{\rho-1}}\,(\theta_M\,\tau_M^{\rho})^{\frac{1}{1-\rho}}}{1 - \rho}(\bar{\alpha})^{\frac{1}{\rho-1}} \left[1 - \theta_H\,\tau_H^{\rho} - (1 - \theta_H\,\tau_H^{\rho})^{\frac{\rho}{\rho-1}}\,(\theta_M\,\tau_M^{\rho})^{\frac{1}{1-\rho}}\,(\bar{\alpha})^{\frac{\rho}{\rho-1}}\right]^{-\frac{1+\rho}{\rho}}, \tag{26}$$

which is the formula provided in (20) in the main text.