Chaining Tasks, Redefining Work: A Theory of AI Automation*

Mert Demirer
MIT & NBER

John J. Horton
MIT & NBER

Nicole Immorlica
Yale & Microsoft

Brendan Lucier
Microsoft

Peyman Shahidi
MIT

September 27, 2026

Abstract

We study the automation and productivity effects of AI when production is a sequence of interdependent steps with a Leontief structure. Each step can be executed manually by labor, augmented with AI, or automated within contiguous AI-executed steps called "chains." By characterizing the firm's cost-minimizing assignment of steps to humans and AI, we show that chaining can overturn comparative advantage logic, since human-advantaged steps may be optimally assigned to AI as part of a chain. Chaining also makes gains from automation depend on how AI-exposed steps cluster in the workflow beyond just how many there are, and can make the returns to improving AI quality non-monotone in a pattern resembling the productivity J-curve. Empirically, we show that existing automation patterns are consistent with the model's predictions: a step is more likely to be AI-executed when its neighbors are, and AI execution is more common where AI-exposed steps cluster in the workflow.

Keywords: Artificial Intelligence, Automation, O-Ring Production, Organization of Production
JEL codes: D24, L23, J24, O33

*Demirer: mdemirer@mit.edu; Horton: jjhorton@mit.edu; Immorlica: nicimm@microsoft.com; Lucier: brlucier@microsoft.com; Shahidi: peymansh@mit.edu. We thank Martin Fleming, Seyed Mahdi Hosseini Maasoum, Guy Lichtinger, Anand Shah, Philip Trammell, and Michael Zhao for helpful discussions. We also thank seminar and conference participants at MIT's Economics of AI Reading Group, FutureTech Lab, Social Analytics Lab, and IDE BIG.AI conference, as well as at Wharton, Columbia Business School, ACM EC'26, and ESIF 2026 at Cornell.

1 Introduction

A growing body of evidence shows that artificial intelligence (AI) tools raise productivity at the task level (Noy and Zhang, 2023; Brynjolfsson et al., 2025; Demirer et al., 2026). Although these gains may be substantial, the larger and likely longer run effects will come from reorganizing production, including shifts in the skill mix of tasks, human capital requirements, and the definition and design of jobs (Brynjolfsson et al., 2021). The workhorse framework in economics for studying the impacts of automation technologies is the task-based model, which represents production as the completion of a set of independent tasks (e.g., Autor et al., 2003; Acemoglu and Autor, 2011; Acemoglu and Restrepo, 2018, 2019, 2022). In these models, what matters is each task's amenability to substitution by, or complementarity with, automation technologies (e.g., computers, robots, or AI). This approach, however, abstracts from the order in which tasks are performed and from their grouping into jobs. We argue that this sequencing is economically consequential, because it changes equilibrium predictions about how jobs are reorganized and which units of work are automated when technologies such as AI enter production.

We model production as a Leontief technology over an ordered sequence of exogenously specified *steps*, and treat the definition of tasks and their partitioning across jobs as endogenous outcomes. A *step* is the primitive unit of work in our framework, corresponding to what classic models call a "task." We instead reserve the term *task* for a contiguous block of steps that the firm endogenously designates for joint execution by a human worker, potentially with the aid of AI. In the absence of AI, tasks optimally collapse to single-step blocks so that steps and tasks coincide. With AI, some steps are chained together and automated, creating a meta-task whose output a human worker validates.

The firm seeks to minimize production costs by choosing which steps of the sequence to delegate to AI and which to assign to human workers. As the AI technology improves, the firm might choose to automate many of these steps, hiring a single, potentially lower-skill worker to verify the output of that automation. Whether this strategy is optimal depends not only on the effectiveness of the AI technology at accomplishing individual production steps, but also on the sequential relationship between steps, the relative ease of verification versus manual execution, wages, and coordination costs between workers. These determinants are what set AI apart from earlier waves of automation. Its capabilities are broad but jagged, varying sharply even across adjacent steps in a workflow (Dell'Acqua et al., 2026), which makes the sequencing of work central to the deployment decision. As a result, improvements in AI can shift deployment decisions at multiple points across the production chain at once, triggering discrete changes in overall production strategy.

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

Time Horizon	AI Deployment	Job Design	Worker Wages	Comments
Short run	Optimized	Fixed	Fixed	Productivity gains by automating existing workflows
Long run	Optimized	Optimized	Adjust to AI deployment and job design	Joint optimization of AI deployment and job design to minimize total labor costs

Notes: This table summarizes how the firm's problem differs across the two horizons studied in the paper. In the short run the firm takes job boundaries and wages as given and chooses only its AI deployment; in the long run job boundaries are chosen jointly with AI deployment, and wages adjust to the skills each job then demands.

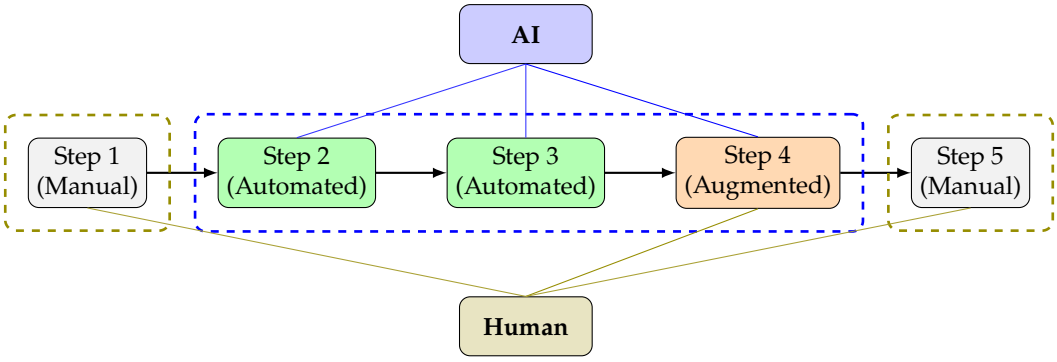
We distinguish two horizons in our framework by how much of its organization the firm is free to adjust, and study how AI affects production in each, as summarized in Table 1. In the short run, the firm takes its workers and their job boundaries as given, along with what those jobs pay, and deploys AI to minimize the time cost of production. In the long run, in tandem with AI deployment, the firm can also redraw the boundaries of jobs, which changes the skills each job demands and hence the wage it commands.

Automation versus Augmentation. Central to our framework 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. For example, a human describes the requirements of a step to an AI tool, the AI then executes the step, and its output is verified by the human. In contrast, we say a step has been *automated* if it is completed by AI end-to-end without any direct human intervention. We view AI automation as a way of putting certain production steps "under the hood" of a logical meta-task that a worker is attempting to complete.

To fix ideas, think of the workflow of a data scientist whose job requires completing five steps: defining the business question (Step 1), finding and fetching data needed for answering the question (Step 2), building an analysis pipeline (Step 3), drafting a report (Step 4), and presenting the findings (Step 5). As part of an AI-augmented request to complete one of these steps (such as Step 4: drafting a report), the AI might also be assigned to complete one or more preceding steps (such as Step 3: running the analysis, or Steps 2 and 3: finding data and running the 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 which we term an *AI chain*. By choosing which steps to automate, augment, or perform manually, the firm implicitly designs the set of tasks exposed to its workers.

Although augmented and automated production steps both involve AI, they differ in

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 requiring a human to verify its output). Lines from the AI box indicate steps involving AI, and lines from the Human box indicate steps requiring human execution or verification. Dashed boxes group steps according to their task assignments: Steps 1 and 5 form separate human tasks, while Steps 2–4 form an AI chain task.

whether a human verifies the AI's output, which augmentation requires and automation does not. Figure 1 illustrates these concepts using the five-step data scientist workflow described earlier. 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 steps without human review. Step 4 is augmented because its output is verified by a human before proceeding. The worker's job therefore consists of three tasks: the manual step 1, the AI chain spanning steps 2–4, whose only human input is verifying the output of step 4, and the manual step 5.

Automation is therefore available only inside a chain. A lone AI step must have its output verified, so chaining is what lets AI work unsupervised. This is potentially a source of large productivity gains, *but* it requires AIs that can perform each step with a high probability of success. In this sense, the model's basic setup is conceptually close to the O-ring model of production (Kremer, 1993; Gans and Goldfarb, 2026).

Optimizing Automation and Fragmentation. The decision to deploy AI on a given production step compares the cost of manual execution to the unified AI-based execution cost, which is not the same as assigning each step to whichever factor of production has comparative advantage in it. Chaining overturns the comparative advantage logic because a human verifies only the output of the chain's last step, so verification is a fixed cost of the chain rather than a marginal cost that scales with its length. Appending a neighboring step to an existing chain therefore costs only a lower probability of end-to-end success, whereas assigning that step to a human terminates the chain and creates an additional checkpoint, entailing another verification or the cost of manual execution. When AI is sufficiently reliable on the marginal step, avoiding this checkpoint can dominate, pulling

the step into AI execution even if manual execution is preferred for it in isolation. In the data scientist workflow, for instance, verifying a report produced after AI executes Steps 3–4 takes the same effort as verifying one produced after AI executes Step 4 alone; the AI is just less likely to succeed at the former. What tips a step into automation is therefore not just its own characteristics but also those of its neighbors.

More generally, we should expect the gains from AI automation to be greater when AI-able steps co-occur in the workflow. A job is more or less "fragmented" in its AI exposure depending on whether the steps where AI is most effective are clustered together or dispersed throughout the workflow. We formalize this idea in a *fragmentation index*, the expected cost of a simple execution plan endowed with foresight into AI's successes and failures, and show analytically that it approximately tracks the cost of optimal AI deployment. We then provide empirical evidence that, in practitioner-documented business processes, workflows with higher fragmentation see a weaker translation from AI exposure to AI execution.

Because chaining makes the cost of automating a step depend on its neighbors, the firm cannot settle each step's execution mode in isolation, and its problem ranges over all possible configurations of production. As the number of feasible arrangements grows exponentially in the number of steps, whether the firm can calculate its optimum is not obvious. We show the firm can indeed solve this optimization, exactly in the short run and to any desired accuracy once wages and job boundaries are also chosen in the long run.

Non-Monotone Effects of AI Improvements. The AI chaining feature of the model implies that improvements in AI quality can generate non-monotone productivity effects. For any fixed production arrangement, better AI reduces costs smoothly by raising AI success probabilities and lowering expected execution times. However, firms can optimize AI deployment over a discrete set of options, and the cost-minimizing configuration can switch at particular AI quality thresholds. As a result, marginal improvements in AI may have little effect when they do not change the optimal production arrangement, which is especially likely in the early days of adoption when AI quality is low. Once AI quality crosses a threshold, longer AI chains become viable and the optimal deployment shifts abruptly, reallocating steps between humans and AI. Our framework thus provides a micro-foundation for the productivity J-curve phenomenon (Brynjolfsson et al., 2021) and yields testable predictions on when and how AI-driven productivity gains trigger structural reorganization in labor markets.

Long-Run Job Design. As the firm can freely reorganize its production around what AI can do over longer horizons, AI changes what workers do and not just how quickly they do it, and through that, wages and labor demand. How broadly to define each job involves a trade-off: adding more tasks to a worker's job raises the wage that job must pay, since the worker has to acquire the skills all of those tasks require, while dividing them across more jobs adds coordination cost, since every job boundary forces a *hand-off* of intermediate output among workers.

Which way improvements in AI move job boundaries is ambiguous, because it presses on both sides of this trade-off. Absorbing steps into chains strips skill out of the tasks a worker retains, making broad jobs cheaper to sustain, while the same reduction, together with any direct saving in coordination costs, makes each boundary cheaper to draw. Our framework thus supplies not a prediction about job breadth but an account of what determines it, and of what would have to be measured to settle the question empirically.

Aggregate Production. Although we develop these results at the level of a single firm, we show that, under additional assumptions, firm-level production aggregates across the economy to a constant elasticity of substitution (CES) representation. Our framework can thus serve as a micro-foundation for economy-wide production functions and be used to study the aggregate effects of AI on labor markets and productivity.

Empirical Evidence. We complement our theoretical analysis with new empirical evidence that speaks directly to the core mechanisms emphasized by the model. Taking the model to data requires three things about every step: AI's ability to perform it, the extent to which AI currently executes it, and where it sits in the production sequence. The first two we draw from existing sources, linking O*NET tasks to human assessments of AI exposure (Eloundou et al., 2024) and to realized AI execution outcomes from Anthropic's Economic Index (Handa et al., 2025). The third, the sequence in which tasks are performed, is not recorded in O*NET, so we take two approaches. First, we use the Process Classification Framework (PCF) of the American Productivity & Quality Center, whose member working groups document the activities of each business process in the order practitioners perform them. Because the PCF covers only operational business processes rather than the full range of occupations, we also generate the ordering of each O*NET occupation's tasks ourselves, and we report our results using both.

We test, and find evidence consistent with, three key predictions of the model. First, the chaining mechanism implies that AI-executed steps should appear in contiguous blocks rather than scattered across the workflow. This is what we find in both datasets, where AI execution operates over consecutive steps far more often than chance would produce.

Second, chaining also creates strong local complementarities in AI automation decisions within a workflow, so that a step positioned next to AI-executed neighbors may be pulled into AI execution by the gains from automating it as part of a chain, even if human execution would be preferred for that step in isolation. We find that when conceptually similar steps appear across different workflows, a given step is more likely to be executed by AI in the workflow where its neighboring steps are also AI-executed. Third, the model emphasizes that fragmentation of AI-exposed steps plays a central role in determining the returns to AI automation. Controlling for the share of steps exposed to AI, workflows whose AI-exposed steps are more dispersed across the production sequence execute a lower share of their steps with AI.

Because our O*NET results rest on a task ordering that we generate with a language model, we show that they are robust to how that sequence is constructed. First, we show that the estimates we obtain are not artifacts of the particular prompt we use to elicit workflow sequences from an LLM (Appendix SA.D). Second, they are not artifacts of the particular way O*NET lists occupation tasks (Appendix SA.E). Third, our sequencing procedure recovers to a substantial degree the orderings of two corpora that no language model ordered, the PCF's practitioner-documented processes and process-mining event logs recorded as the work was carried out (Appendix SA.F).

Taken together, these results show that the sequencing logic at the heart of the model is already visible in how firms use AI at this early stage of adoption. What AI executes depends not only on comparative advantage and step-level characteristics, but also on where a step sits relative to its neighbors. This is why the share of steps exposed to AI can be a misleading predictor of occupational impacts when production steps are technologically interdependent. As adoption widens and AI becomes reliable enough to sustain longer chains, we expect these patterns to grow more pronounced.

2 Related Literature

Task Interdependence in Task-based Models of Automation. Our framework relates to the literature on task-based models of technical change, which generally view tasks as predetermined and technologically independent objects combined through a CES production function (Acemoglu and Autor, 2011; Acemoglu and Restrepo, 2018, 2019; Acemoglu et al., 2024). We depart from this literature by modeling production as a sequence of technologically interdependent steps, where AI chaining creates complementarities across adjacent steps that can overturn comparative advantage as the driver of who does what.[1] Chaining

[1]Two contemporary works by Trammell (2026) and Farboodi et al. (2026) study a different form of linkage across tasks, where performing a task raises productivity on others through learning, by workers in the

also lets step-level decisions aggregate into the system-level substitution that Autor et al. (2003), Acemoglu and Restrepo (2019), and Bresnahan et al. (2002) emphasize, since entire chains can be automated in a single leap rather than step by step.

Complementarities, O-Ring Dynamics, and the Productivity J-Curve. Our work contributes to the literature on friction-laden technology adoption and organizational complementarities, in which information technologies raise productivity mainly alongside complementary organizational change (Bresnahan et al., 2002; Bloom et al., 2016).[2] We micro-found these complementarities through the O-ring theory of production (Kremer, 1993). Chaining makes the payoff to automating a step depend on whether its neighbors can join the same AI block, so the returns to improving any one step turn on performance elsewhere and arrive only once AI quality crosses a threshold that makes longer chains worthwhile. Our results align with Gans and Goldfarb (2026), who likewise emphasizes task interdependence in O-ring production and obtains lumpy automation through time reallocation rather than workflow adjacency. Because these returns arrive only once AI quality passes such a threshold, the marginal benefit of improving AI is non-monotone, and in this sense our framework provides a micro-foundation for the "productivity J-curve" pattern of technology adoption (Brynjolfsson et al., 2021; McElheran et al., 2025; Furman and Seamans, 2019; Tambe and Hitt, 2012; McElheran et al., 2024).

Division of Labor and the Boundary of the Job. Our work also connects to the literature on the division of labor within a firm. The Coase–Williamson logic, in which economic boundaries are shaped by coordination, transaction, and governance costs (Coase, 1937; Williamson, 1971, 1979), applies to the "boundary of the job" as well, in that the firm chooses how many and which tasks to bundle into a worker's role, making the degree of specialization endogenous (Murphy, 1986; Becker and Murphy, 1992; Dessein and Santos, 2006). Our hand-off costs are the intra-firm analogue of the transaction costs that AI cannot fully subsume, capturing the tacit or social knowledge aspect of work (Deming, 2017), and playing the role that the cost of unbundling tasks plays in Garicano et al. (2026). AI nonetheless redraws job boundaries by stripping skill out of the steps it absorbs, effectively changing the skill requirements of jobs (Autor et al., 2024; Autor and Thompson, 2025; Autor et al., 2026).[3]

former and by machines in the latter. Our emphasis is instead on the ordering of tasks in production.

[2] Kim et al. (2026) provide recent experimental evidence of this pattern for AI, holding tools and training fixed across startups and varying only what founders learn about reorganizing production around them.

[3] Ide and Talamàs (2025) provides a distinct but related view, embedding AI as an autonomous problem solver in a knowledge hierarchy (Garicano, 2000; Garicano and Rossi-Hansberg, 2006) and studying how it reshapes spans of control within the firm.

Measuring AI Exposure and Realized Execution. A rapidly growing literature has sought to quantify the impact of new automation technologies on the labor market. Early work studies the susceptibility of occupations to computerization (Frey and Osborne, 2017), and more recent work develops measures of occupational exposure to AI (Brynjolfsson et al., 2018; Webb, 2020; Felten et al., 2021; Eloundou et al., 2024; Hampole et al., 2025; Tomlinson et al., 2025), typically by mapping technologies to specific tasks and then aggregating these task-level mappings to construct occupation-level indices. We build on these measures by showing how, in addition to the share of tasks exposed to AI, the arrangement of those tasks in a workflow also governs the gains from automation.

3 AI Chaining and Production in the Short Run

In this section and the next we consider the firm's production in the short run. A firm's production consists of a sequence of steps $\mathcal{S} = (s_1, \ldots, s_m)$, every one of which must be completed to produce a unit of output.[4] The firm can assign each step either to a human worker or to AI. A step assigned to a human takes worker time and demands skill. A step assigned to AI succeeds only with some probability, and its output may need to be reviewed by a human worker before it is accepted.

In the short run, job boundaries are already drawn, so each worker's job covers a fixed block of steps; workers have acquired the skills their roles require; and the firm has hired them at wages agreed in advance. The firm's only margin of adjustment is AI deployment, that is, which steps it assigns to AI. That choice sets how much time each worker spends completing steps, and hence the wage bill the firm pays. Given these short-run assumptions, we assume without loss of generality that there is a single worker type whose wage is normalized to 1. Section 5 relaxes these, letting the firm divide the steps among workers so that the skill a job demands, and hence the wage it commands, becomes endogenous. Table A.1 in Appendix A summarizes our notation.

3.1 Steps

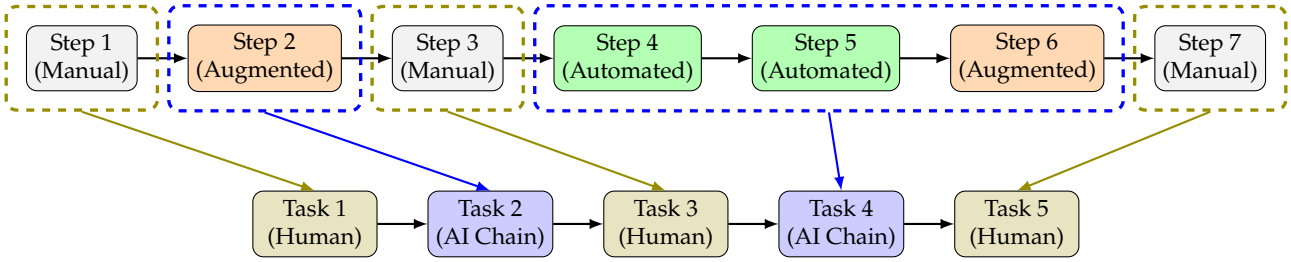
Steps are the smallest unit of work in our framework, and each is executed in one of three modes: *manually*, *augmented*, or *automated*. If the firm assigns step i to manual execution (denoted by M), a human worker spends time t_i^M completing the step without AI. Alternatively, the firm may assign the step to AI (denoted by A), in which case a worker spends time t_i^A prompting and verifying the output of one AI attempt at it.[5,6] Such an

[4] We hold the set of production steps fixed and abstract from the creation or disappearance of steps.

[5] This could, for example, represent the time spent specifying what the step requires and then checking whether what the AI returns meets it.

[6] With highly capable AIs, the human role shrinks to verifying AI's outputs, as in Agrawal et al. (2019).

Figure 2: Bundling Steps into Tasks via AI Strategy



Notes: The top layer shows seven production steps, each completed in manual (gray), automated (green), or augmented (orange) mode; dashed boxes group contiguous steps into tasks. The bottom layer shows the resulting tasks: a run of automated steps ending in an augmented step forms an AI chain (blue; here steps 4–6 become Task 4), and a lone augmented step is an AI chain of length one (here step 2 becomes Task 2), while a step performed manually on its own is a human task (olive). Horizontal arrows show the production flow within each layer, and each colored vertical arrow shows which dashed block of steps becomes which task.

attempt 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, and it must be repeated until it succeeds. Since the worker repeats prompting and verifying AI's output until success, the expected time cost of verification across trials is t_i^A/q_i. Definitions 1 and 2 formalize these two modes of execution.

Definition 1 (Manual Step)**.** *A step i is performed manually if it is executed entirely by a human worker without AI assistance. The time cost of a manual step is denoted by t_i^M.*

Definition 2 (Augmented Step)**.** *A step i is augmented if it is executed by AI, after which a human worker verifies its output. The expected time cost of verifying a single augmented step is t_i^A/q_i, where $q_i = \alpha^{d_i} \in (0, 1]$ is the AI's success probability on the step, increasing in the quality α of the general-purpose AI technology and decreasing in the step's difficulty d_i for AI.*

In the top layer of Figure 2, Step 1, for example, is a manual step, whereas Step 2 is an augmented step performed by the AI and verified by a human.

Manual and augmented execution specify the mode of a step in isolation. There is a third mode, which we call *automation*, that the firm obtains by assigning multiple consecutive steps to AI. In this case, the AI attempts to complete the entire run of steps without a human reviewing the intermediate outputs, and the worker verifies only the output of the last step. Every step before the last is then automated, since no human is involved in it, while the last step is augmented. Chaining steps in this way lowers the probability that an AI attempt succeeds, since the AI must complete every step in the run successfully, but it economizes on human time, since the output is verified only once. We call such a contiguous run of AI-executed steps an *AI chain*.

More precisely, an AI chain spans steps $(s_\ell, \ldots, s_r)$, where the worker verifies only the output of the final step s_r. One attempt at the chain therefore costs the same as verifying

the last step s_r, namely time t_r^A.[7] Assuming independent failures across steps, an AI attempt succeeds with probability $\prod_{i=\ell}^{r} q_i$, and the chain is retried until it succeeds, for a total expected time of $t_r^A / \prod_{i=\ell}^{r} q_i$. Since $q_i = \alpha^{d_i}$, the chain succeeds with probability $\alpha^{\sum_{i=\ell}^{r} d_i}$, so $\sum_{i=\ell}^{r} d_i$ is the chain's total difficulty, aggregating additively over its steps.

Note that a chain may consist of a single step, with $\ell = r$, in which case the chain is simply an augmented step, so an augmented step is equivalent to an AI chain of length one, a convention we adopt throughout the paper. We now give the formal definitions of automated steps and AI chains.

Definition 3 (Automated Step)**.** *A step is automated if it is executed by AI end-to-end, with no human verifying its output, so that its direct human time cost is zero.*[8]

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, with expected time cost $t_r^A / \prod_{i=\ell}^{r} q_i$.*

In Figure 2, Steps 4–6 form an AI chain. The human prompts the AI for the entire chain, but verifies the output of only Step 6, the augmented step. Steps 4 and 5 are hence performed without any direct human intervention and are automated. Step 2 forms an AI chain (of length one) on its own as well, containing only a single augmented step.

3.2 Tasks

If steps are the smallest unit of work, tasks are the smallest unit of *human* work. The distinction follows from the execution modes of steps above. A worker is involved in a step only when they perform it manually or verify what the AI returns at the end of a chain, whereas an automated step runs inside the chain entirely beneath their awareness. Every point at which a worker executes or verifies work therefore opens a task for them, and that task absorbs whatever the AI has run automatically since the previous such point. The steps of a workflow map into tasks accordingly. A manual step is a task on its own, and an entire AI chain is a single *composite* task, delegated to the AI at its start and received back at its augmented endpoint.

[7] One could instead let the cost of verifying a chain scale with its length, for example by charging a fixed overhead for every step the chain spans on top of verifying its endpoint. The channel we are describing would nonetheless remain the same, as extending a chain would still economize on execution cost, provided that this overhead is smaller than what the appended step would cost on its own. We do not model this overhead verification cost to avoid complicating the notation.

[8] That execution of automated steps does not incur any cost to the firm is consistent with the AI subscription business model, where the firm pays an upfront fixed cost to an AI provider and uses AI tools at zero marginal cost afterward. The forces described by the model remain unaffected even if we assumed that each time the firm deploys AI on a step it has to incur a cost.

The firm's choice of where to place chains is therefore what creates the tasks its workers hold, and we record that choice as an *AI deployment strategy*.

Definition 5 (AI Deployment Strategy)**.** *An AI deployment strategy (or AI strategy for short) is a partition of the step sequence $\mathcal{S} = (s_1, \ldots, s_m)$ into contiguous blocks, each of which is either a single manual step or an AI chain (automated or augmented). The blocks of the partition are the firm's tasks, forming the task sequence $\mathcal{T} = (T_1, \ldots, T_n)$, and each task T_b carries an expected time cost t_b: t_i^M for a manual step s_i, and $t_r^A / \prod_{i=\ell}^{r} q_i$ for a chain spanning $(s_\ell, \ldots, s_r)$.*

The AI deployment strategy in Figure 2 maps the seven steps in the top row into five tasks in the bottom row: two AI chains (a length-three chain of Steps 4–6 and a length-one chain of Step 2) and three manual tasks.

3.3 Short-Run Production

The firm employs human labor to perform tasks and produce the output, and labor is the only input used in production.[9] The worker carries out the entire step sequence $\mathcal{S} = (s_1, \ldots, s_m)$, which the firm's AI strategy organizes into the task sequence $\mathcal{T} = (T_1, \ldots, T_n)$, and is paid at the normalized rate of $w = 1$ per unit of time for the whole time production takes. The cost of production is therefore the total expected time requirement of the workflow's tasks, and the firm chooses the AI deployment strategy that minimizes

$$\min_{\mathcal{T}} \sum_{T_b \in \mathcal{T}} t_b. \tag{1}$$

Problem (1) defines the firm's short-run cost minimization problem. In the short run, AI is purely productivity-improving, compressing the time a fixed set of steps takes by letting the firm minimize cost through a combination of augmented and automated steps.

AI chaining makes solving this problem non-trivial. Because a chain links several steps into a single unit of execution, the firm cannot solve the problem in (1) separately for each step; the choice instead ranges over all feasible arrangements of the workflow. This linkage gives rise to rich dynamics in how AI is deployed across production, and hence in how automation affects it. We discuss these implications next.

4 Implications of Chaining

In this section we draw out three implications of AI chaining in the model. First, we show that chaining can overturn comparative advantage logic, in that a step that is cheaper to

[9] Viewed as a production function, our sequential technology is Leontief at the firm level, because producing a unit of output requires completing every step in fixed proportions. The Leontief form is natural for modeling a specific production process at a disaggregated level (De Loecker and Syverson, 2021).

perform manually in isolation may nonetheless be executed by AI in the optimal configuration of production (Section 4.1). Second, we introduce a *fragmentation index* that captures how dispersed a workflow's AI-able steps are, and show that it approximates the cost of the firm's optimal AI strategy, so that the gains from AI depend on the arrangement of AI-able steps (Section 4.2). Third, we show that the marginal gains from improving AI quality are non-monotone, abruptly rising whenever improvements in AI quality make redesigning workflows worthwhile (Section 4.3). All proofs are collected in Appendix B.

4.1 Overturning of Comparative Advantage

For a single step, the decision whether to deploy AI depends simply on whether the manual execution cost exceeds the AI augmentation cost. With more than one step, however, the optimal deployment of AI depends on more than each step's own characteristics, because of the benefits of automating multiple steps together in an AI chain. As a result, the factor of production holding the comparative advantage on a step in isolation need not be the one that performs it in the optimal production configuration. That is, a step that is cheaper to perform manually may still be executed by AI as part of a chain.

Consider a step k on which the human holds the comparative advantage, so that $t_k^M < t_k^A/q_k$, together with its immediate neighbors: a predecessor $k-1$, and a successor $k+1$ executed as a standalone augmented step.[10] Ask what it costs to admit $k-1$ and k into the chain that ends at $k+1$, holding the rest of the workflow fixed. The marginal cost of admitting the two is the chain's cost beyond verifying step $k+1$ alone,

$$\underbrace{\frac{t_{k+1}^A}{q_{k-1}\,q_k\,q_{k+1}}}_{\text{cost of chain over } \{k-1,k,k+1\}} - \underbrace{\frac{t_{k+1}^A}{q_{k+1}}}_{\text{cost of augmenting } k+1 \text{ alone}} = \underbrace{\frac{t_{k+1}^A\,(1-q_{k-1}q_k)}{q_{k-1}\,q_k\,q_{k+1}}}_{\text{marginal cost of folding the pair } \{k-1,k\} \text{ in}}.$$

Writing the joint failure risk as $1-q_{k-1}q_k = (1-q_{k-1}) + q_{k-1}(1-q_k)$ splits this marginal cost into a share for each admitted step separately,

$$\underbrace{\frac{t_{k+1}^A\,(1-q_{k-1}q_k)}{q_{k-1}\,q_k\,q_{k+1}}}_{\text{marginal cost of folding the pair } \{k-1,k\} \text{ in}} = \underbrace{\frac{t_{k+1}^A\,(1-q_{k-1})}{q_{k-1}\,q_k\,q_{k+1}}}_{\text{step } k-1\text{'s share}} + \underbrace{\frac{t_{k+1}^A\,(1-q_k)}{q_k\,q_{k+1}}}_{\text{step } k\text{'s share}}. \tag{2}$$

There are two channels through which the human-advantaged step k can end up automated. The first is that step k's share of the marginal cost, the second term in (2),

[10] The augmented step $k+1$ can be thought of as a longer AI chain terminating augmented at some step $r > k+1$. Such a chain can be treated as a single augmented step with composite cost parameters, namely verification cost t_r^A and success probability $\prod_{i=k+1}^{r} q_i$, so what follows applies unchanged.

falls below what executing it alone would cost, which by assumption is manual execution t_k^M. This condition indicates whether chaining steps k and $k+1$ alone costs less than the two being executed optimally on their own. Note that this does not depend on the predecessor, and is easiest to see when we set $q_{k-1} = 1$, so that step $k-1$ becomes "free" and its share in (2) vanishes. Importantly, step k's own verification cost t_k^A never enters this comparison, since an automated step's output is never checked on its own.

Even when the comparison above does not favor folding the step into the chain, step k may still end up automated. The second channel runs through the predecessor. Suppose the predecessor step $k-1$ is expensive to execute alone, whether manually or augmented, but reliable enough for AI that its contribution to the chain's failure risk is negligible (i.e., as $q_{k-1} \to 1$ the first term in (2) approaches zero). Admitting the pair $\{k-1, k\}$ is then worthwhile as long as the time saved by folding in the predecessor more than covers the shortfall step k leaves when it chains with its successor alone. Step k can therefore be automated to bridge the expensive but AI-reliable predecessor to the chain's endpoint, even though its own share of the marginal cost exceeds what executing it alone would cost and, without the predecessor, it would have stayed manual. Proposition 1 formalizes the overturning of comparative advantage and makes a more general claim.

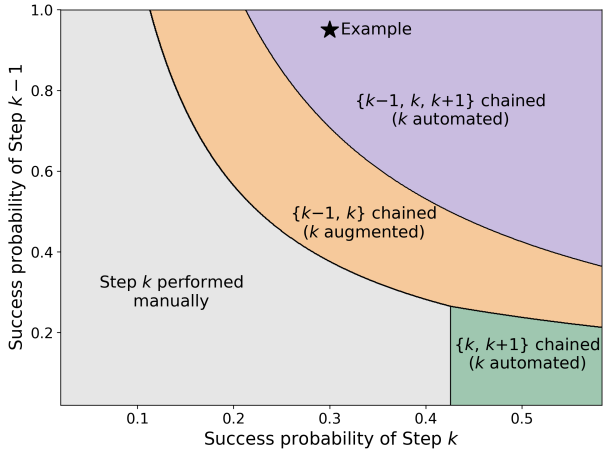
Proposition 1 (Overturning of Comparative Advantage)**.** *Hold the execution of the workflow outside the block $\{k-1, k, k+1\}$ fixed, and let no chain cross the block's boundary. Let the human hold the comparative advantage on step k, so that $t_k^M < t_k^A/q_k$. Then:*

(i) Whatever step k's own parameters, there is a nonempty open set of neighbor parameters at which the cost-minimizing arrangement has step k executed by AI (automated or augmented) no matter how wide the human's advantage on it is.

(ii) Fix the cost parameters of the three steps and let their success probabilities vary. If AI executes step k at (q_{k-1}, q_k, q_{k+1}), then it does so at every $(q'_{k-1}, q'_k, q'_{k+1})$ with $q'_i \geq q_i$ for each i.

Intuitively, part (i) of Proposition 1 says that whether a step is executed by AI depends not just on its own cost parameters, as comparative advantage logic would predict, but also on those of its neighbors. A step with identical characteristics can thus be automated in one workflow where it sits beside an AI-able run, yet remain manual in another where AI completes the neighboring steps only with low probability. Where part (i) says that being human-advantaged carries no guarantee of being executed one way or the other, part (ii) says that whatever the answer at a given set of parameters, making a step and/or its neighbors friendlier to AI can only push it toward AI execution. A step that AI executes thus never reverts to manual as its neighbors become more reliable, and the same holds when the step's own reliability improves alongside theirs.

Figure 3: When Is a Human-Advantaged Step Executed by AI?



Notes: The figure shows the four arrangements available for a human-advantaged Step k and its immediate neighbors, and shades which of them the firm chooses as the success probabilities of Steps $k-1$ and k vary. Every other parameter is held at the value given in the text. The human holds the comparative advantage on Step k throughout the plotted range. Each shaded region is labelled by the arrangement the firm chooses in it, and the star marks the parameters of that example.

To put this result into perspective, consider a workflow containing the following three steps: an expensive step that AI completes very reliably at position $k-1$, an *AI-hard* step at k, and an *AI-easy* step at $k+1$,[11]

$$\left(t^M_{k-1},\, t^A_{k-1},\, q_{k-1}\right) = (25,\, 25,\, 0.95), \quad \left(t^M_{k},\, t^A_{k},\, q_{k}\right) = (6,\, 3.5,\, 0.3), \quad \left(t^M_{k+1},\, t^A_{k+1},\, q_{k+1}\right) = (8,\, 4,\, 0.9).$$

In isolation, comparative advantage assigns steps $k-1$ and k to the human and step $k+1$ to AI. Yet the cost-minimizing strategy chains all three, augmented at step $k+1$, at a cost of 15.6 against the 35.4 the comparative advantage arrangement would cost.

Figure 3 shows the implications of Proposition 1 in this setup. It holds all parameters fixed except the success probabilities of Steps $k-1$ and k, and shades how the cost-minimizing arrangement executes Step k as those two vary, with the example itself marked by a star. The human keeps the comparative advantage on Step k throughout, since the horizontal axis stops where augmenting the step in isolation would break even against its manual cost of 6. Even so, Step k is performed manually in the lower left where AI is unreliable on both steps k and $k-1$. Moving to the right, the chain $\{k,\, k+1\}$ becomes more efficient at $q_k \approx 0.43$, where the failure risk Step k adds to it falls below the manual work it displaces; moving up instead leaves Step k untouched and reaches the three-step chain by way of the band in which the chain stops at Step k.[12] Varying a step's own

[11] One can think of an AI-easy step as one that is "exposed" to AI, and an AI-hard step as "unexposed."

[12] The orange band, where Step k is augmented, is not a general feature. When verifying AI's output at Step k is expensive enough relative to executing Step $k+1$ on its own, stopping the chain at Step k never pays, and

success probability and that of a single neighbor is thus enough to cover all four feasible ways of executing a human-advantaged step, minimal as that adjustment is. Part (ii) of Proposition 1 can be read off the same figure as well. Starting anywhere that AI executes Step k and moving up, to the right, or both, one never crosses back into the grey region, so improvements in AI reliability only ever add steps to those AI executes.

We must note that Proposition 1 is a *local* statement. It compares rearrangements of a step and its neighbors while holding the AI strategy on the rest of the workflow fixed, and so it falls short of determining whether the overturning takes place in the overall cost-minimizing arrangement of the entire production sequence. Nonetheless, the forces determining the execution mode of a single step in the global optimum are the ones outlined here.

4.2 Workflow-level AI Exposure and the Fragmentation Index

The previous section examined interdependence locally, asking how a step's immediate neighbors shape its execution mode. The same forces also operate at the level of the workflow as a whole. Whether, and how much, a workflow benefits from AI depends not only on how many of its steps AI can reliably perform, but on how those steps are arranged across the entire sequence. The logic is again the complementarity that chaining creates. The more fragmented the AI-able steps are across the production sequence, the fewer of them can share a verification. This section formalizes this logic and provides an approach to measurement. We first start with an example that makes the point numerically.

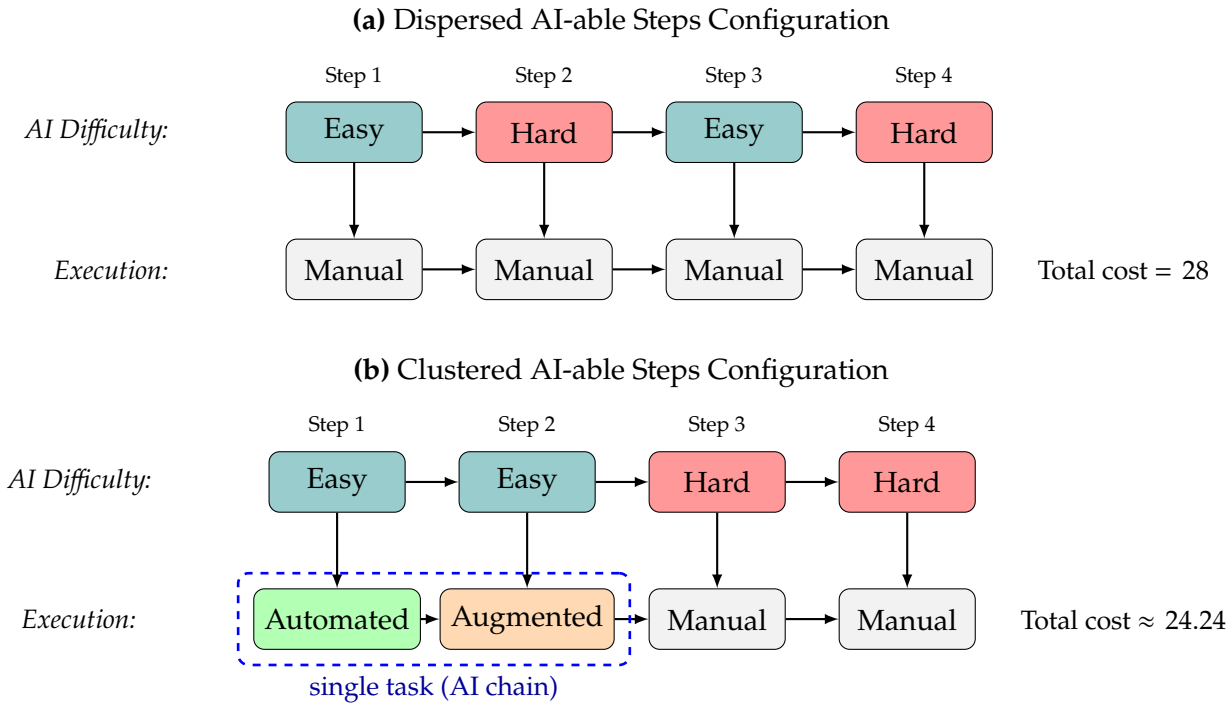
Example 1. Consider a workflow containing four steps built from the AI-hard step of Section 4.1 and a version of its AI-easy step that is costlier to verify and that AI completes less reliably, with two steps of each kind. Figure 4 shows the same four steps arranged in two different orders.

Panel (a) interleaves the two kinds, alternating AI-easy and AI-hard steps. Because every easy step is flanked by hard ones, no two AI-easy steps are ever adjacent, and pairing an easy step with a hard one in a chain costs more than running both alone, so no step is automated. The firm is left to treat each step on its own, and on its own neither kind of step is worth giving to AI. The workflow therefore uses no AI at all, even though half of its steps are ones AI performs comparatively well.

Panel (b) clusters the AI-easy steps instead, placing them side by side. Chaining then becomes a cost-effective option, because AI automates the first easy step at zero direct cost and verifies only once at the second. A single verification now covers two steps instead of none, which is enough to make AI worth deploying where it was not before.

the firm crosses directly from manual execution to automation.

Figure 4: Role of Clustering of AI-able Steps in Facilitating Automation



Notes: In each panel the top layer gives each step's difficulty for AI and the bottom layer its execution in the cost-minimizing strategy, colored as in Figure 2. The AI-hard steps have parameters $(t_i^M, t_i^A, q_i) = (6, 3.5, 0.3)$ and the AI-easy steps (8, 6, 0.7). The two panels contain exactly the same four steps and differ only in the order in which they are performed. In panel (a) every AI-easy step is flanked by AI-hard ones, so no two AI-able steps are adjacent, no chain can form, and every step is performed manually. In panel (b) the AI-easy steps form a contiguous run, which the firm executes as a single AI chain: Step 1 is automated and Step 2 is augmented and verified. The total cost at the right of each execution row makes the two arrangements directly comparable.

Note that the two arrangements contain exactly the same set of steps and differ only in the order in which those steps are performed, yet one of them uses no AI while the other hands half of its steps to AI. Dispersion of AI-able steps in the workflow can therefore stop AI exposure from converting into AI execution altogether, rather than merely making that execution dearer. This is precisely the margin that we explore empirically later. Example 1 isolates the arrangement of steps in the case that matters for what follows, where the steps AI performs poorly are ones the firm would not fold into a chain under any arrangement.[13]

As gains from AI automation appear to depend on how clustered or dispersed a workflow's AI-able steps are, having a measure of that arrangement would be useful, one that stands in for them without solving the firm's optimization problem outright. To build

[13] In the normalized case introduced below ($t_i^A = 1$), this holds whenever $t_h^M \leq 1/q_h - 2$ for each AI-hard step s_h. Extending a chain over s_h multiplies its expected cost by $1/q_h$ while saving only the cost of executing s_h on its own, which requires $1/q_h \geq 1 + \min\{t_h^M, 1/q_h\}$. The binding deviation is the one that instead merges the chains on either side of s_h, since it saves one of the two verifications those chains pay separately. It requires $1/q_h \geq 2 + \min\{t_h^M, 1/q_h\}$, and the minimum must then be t_h^M, which gives the condition above.

such a measure, return to the firm's AI deployment problem in (1) and consider the special case in which AI verification costs are normalized, $t_i^A = 1$ for all i, so that every step takes the same time to prompt and verify.[14] Suppose each step s_i succeeds independently with probability q_i, and say that any step that does not succeed *fails*. Write F for the set of steps that fail and $\mathcal{C}$ for the random collection of maximal connected components of non-failed steps, and assign each $C \in \mathcal{C}$ the weight $\omega(C) = \min\{1, \sum_{s_i \in C} t_i^M\}$, which is 1 unless the manual costs of its steps sum to less than the normalized cost of verifying one AI attempt. The realized fragmentation of the workflow is then

$$\sum_{s_i \in F} \min\left\{t_i^M, \frac{1}{q_i}\right\} + \sum_{C \in \mathcal{C}} \omega(C), \tag{3}$$

and the *fragmentation index* of the workflow is its expected value. Proposition 2 shows how the fragmentation index and optimal AI deployment cost are related.

Proposition 2. *Fix a sequence of m steps $\mathcal{S} = (s_1, \ldots, s_m)$, each with normalized AI verification cost $t_i^A = 1$. Let FI denote its fragmentation index and let OPT denote the minimum cost over all AI strategies. Then $\frac{1}{8}OPT \leq FI \leq \frac{5}{4}OPT$. If we further assume $t_i^M \geq 1$ for all i, then $\frac{1}{4}OPT \leq FI \leq \frac{5}{4}OPT$.*

Proposition 2 says that a single closed-form quantity tracks, up to constant factors, what the firm's whole optimization is worth, so a workflow with a high fragmentation index is costly to produce however cleverly the firm deploys AI. What the quantity in (3) measures is easiest to see through a prophet who can look ahead and tell which steps AI would complete on its first attempt. Runs of steps that will all succeed are natural candidates for a chain, so the prophet chains each such run, unless running that whole run manually is cheaper, and executes every remaining step on its own in whichever of the two modes is cheaper for it; the fragmentation index is the expected cost of that strategy. This is not the optimal strategy even with foresight, since a long chain with a good chance of success may be worth attempting even when the prophet knows it will fail on the first try. What Proposition 2 establishes is that its cost nonetheless approximates the cost of the optimal strategy, which enjoys no foresight at all.

Read this way, the index is lower when the steps AI performs reliably are clustered, since a long run of such steps is more likely to survive as a single connected component once failures are realized, and a single component is charged only once. When every step takes at least as long manually as a single AI verification (i.e., $t_i^M \geq 1$) every component

[14] The normalization puts every step on the same footing as far as the cost of using AI goes, leaving where the AI-able steps sit as the only thing that separates one workflow from another.

is charged $\omega(C) = 1$ and the fragmentation index reduces to[15]

$$FI = \underbrace{\sum_{i=1}^{m}(1 - q_i)\min\left\{t_i^M, \frac{1}{q_i}\right\} + \sum_{i=1}^{m} q_i}_{\text{invariant to the order of the steps}} - \sum_{i=2}^{m} q_{i-1}\, q_i. \tag{4}$$

The first two terms depend on the workflow's steps but not on the order in which they are performed. Across re-orderings of a given workflow the index is therefore decreasing in $\sum_{i=2}^{m} q_{i-1}q_i$, the total reliability of adjacent pairs of steps, and is lowest precisely when the steps AI performs reliably sit beside one another.[16] A further implication of the proof is an approximately optimal greedy algorithm for constructing AI strategies, where the firm groups steps into a chain as long as the probability of success stays sufficiently high, terminating the chain and starting a new one when it falls too low, and running a chain manually whenever its steps' total manual time is below the cost of one verification.

4.3 Non-Monotonicity of Gains from AI Improvements

So far we have asked how AI's impact on a workflow depends on the arrangement of its steps, holding the capability of the AI technology fixed. We now turn to what happens as that capability rises. We show that the gains from marginal improvements in AI quality accrue non-monotonically, because improvements that are at first absorbed within the existing arrangement eventually trigger a discrete reorganization of work.

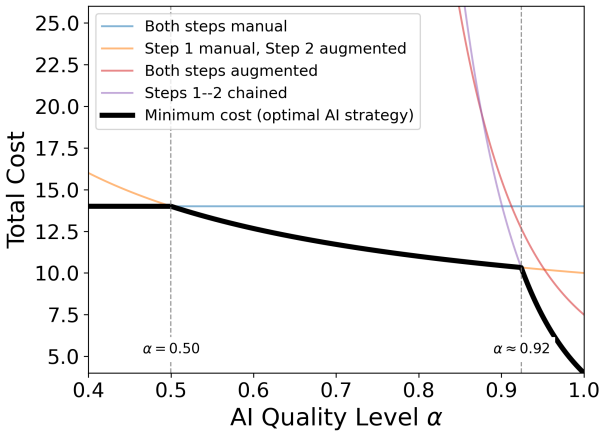
Recall that α measures the quality of the general-purpose AI technology, so that advances in that technology are represented by increases in its value. When AI quality is so low that the firm deploys AI nowhere, marginal improvements offer no return at all, since every step is performed manually and production costs do not depend on α. Once α passes the level at which deploying AI somewhere in the workflow becomes worthwhile, further improvements do pay off, but at a diminishing rate, since a more reliable AI is being applied to a fixed set of AI-executed steps. And when α reaches a level at which the firm re-optimizes its deployment strategy, extending a chain to absorb steps that were

[15] To see how Equation 4 is obtained, note that $t_i^M \geq 1$ implies $\sum_{s_i \in C} t_i^M \geq 1$ for every component C, so $\omega(C) = 1$ and the realized fragmentation in (3) becomes $\sum_{s_i \in F} \min\{t_i^M, 1/q_i\} + |\mathcal{C}|$. Taking expectations, step s_i fails with probability $1 - q_i$, and a component starts at s_i exactly when s_i succeeds and either $i = 1$ or s_{i-1} fails, so $\mathbb{E}|\mathcal{C}| = q_1 + \sum_{i=2}^{m} q_i(1 - q_{i-1}) = \sum_{i=1}^{m} q_i - \sum_{i=2}^{m} q_{i-1}q_i$.

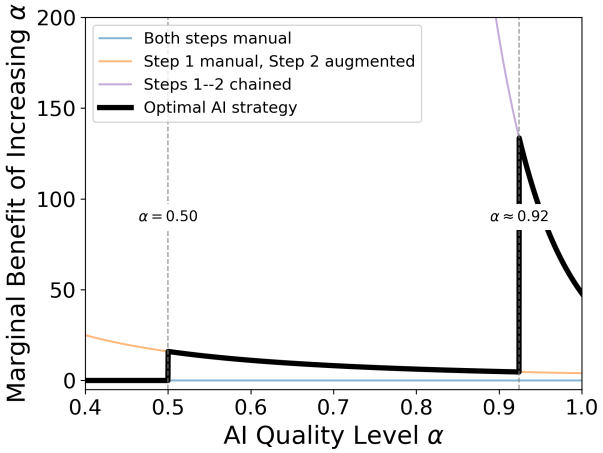
[16] That only adjacent pairs enter is a consequence of the assumption rather than an approximation. Once every component is charged the same 1 whatever its length, the arrangement matters only through how many components there are, and a component begins wherever a step that succeeds follows one that failed, which is an event involving two steps. Absent the assumption, $\omega(C)$ depends on the whole of C and no such expression in adjacent pairs exists. Without this assumption the charge-once saving can disappear altogether. If the manual costs of a merged run sum to less than the cost of one verification, ω is additive over that run's steps and the index does not respond to the arrangement at all.

Figure 5: Non-monotonicity of Gains from Improving AI Quality

(a) Total cost by AI strategy



(b) Marginal benefit of improving α



Notes: This figure plots the two-step workflow of Example 2 as AI quality α rises. Panel (a) shows the cost of each AI strategy, with the optimal strategy given by their lower envelope (thick black line). Table A.3 in Appendix A gives each strategy's cost and the range of α over which it is optimal. Panel (b) shows the marginal benefit of improving AI quality, $-\mathrm{d}C/\mathrm{d}\alpha$. The marginal benefit jumps at each α where the optimal strategy switches, so gains from AI improvements are non-monotone in α.

previously manual, the marginal return jumps up discontinuously, because improvements in quality now compound over a longer stretch of the workflow. We call the values of α at which the optimal strategy changes reorganization thresholds. Between them the returns to AI progress diminish, and at each of them those returns jump upward as longer AI chains become worth deploying.[17] Example 2 traces them in a two-step workflow.

Example 2. Take the AI-hard and AI-easy steps of Section 4.1 once more, relabelled steps 1 and 2 and keeping their time costs, but now let their difficulties be $d_1 = 11$ and $d_2 = 1$, so that AI succeeds at them with probabilities $q_1 = \alpha^{11}$ and $q_2 = \alpha$. At an AI quality of about $\alpha = 0.9$ these deliver the success probabilities used there, namely $q_1 \approx 0.3$ and $q_2 = 0.9$. Figure 5 traces the optimal cost and its marginal benefit as AI quality α rises.

With $q_1 = \alpha^{11}$ and $q_2 = \alpha$, the firm chooses among five configurations, ranging from both steps manual to both chained. Table A.3 in Appendix A collects them, together with the cost expression of each, the marginal benefit of improving AI quality, and the range of α over which the configuration is cost-minimizing. Below $\alpha = 0.5$ the firm uses no AI; between 0.5 and roughly 0.92 it augments the easy step alone; above that it chains the two.

Panel (b) shows that the marginal return to improving AI quality jumps up at each threshold where the optimal strategy switches, since a different strategy governs cost on

[17] Stated precisely, the returns never fall at a threshold, and they jump upward at every threshold where the cost curves of the two adjacent strategies cross non-tangentially. That is the generic case, and it holds at both thresholds of Example 2; Lemma B.4 in Appendix B.3 gives the condition and shows it cannot be dropped.

either side of the threshold. It is also worth noting that the jump at $\alpha \approx 0.92$ is larger than the one at $\alpha = 0.50$. Forming the chain pulls a second step under AI, so cost falls more steeply. This ordering is specific to this example, however, since a later threshold does not necessarily introduce a longer chain than an earlier one.

We formalize the non-monotonicity in marginal returns to improving AI quality in Appendix B.3. There we show that the cost of any fixed AI strategy varies smoothly with α, so that the firm's optimal cost is the lower envelope of finitely many smooth cost curves, and prove in Lemma B.4 that the marginal gains from improving AI diminish monotonically within each regime where a single strategy remains optimal, never fall at a reorganization threshold, and jump upward at every threshold at which the two adjacent cost curves cross non-tangentially, which is the generic case.

The non-monotone productivity gains described here are a familiar pattern for general-purpose technologies. The steam engine, electricity, and the computer each delivered only modest improvements in the early days of their adoption, and their full impact arrived once production had been reorganized around them, the shape Brynjolfsson et al. (2021) describe as the productivity J-curve. What the model shows is that the same shape arises within a single workflow, because the returns to better AI wait on discrete reorganizations of it. The reorganizations available here are nonetheless the narrowest kind, since the firm changes only which steps AI executes while job boundaries stay fixed. Over the horizon on which AI quality improves, firms can redefine jobs, changing the skills each demands and the wages it commands, and it is that larger reorganization we study next.

5 Job Design and Production in the Long Run

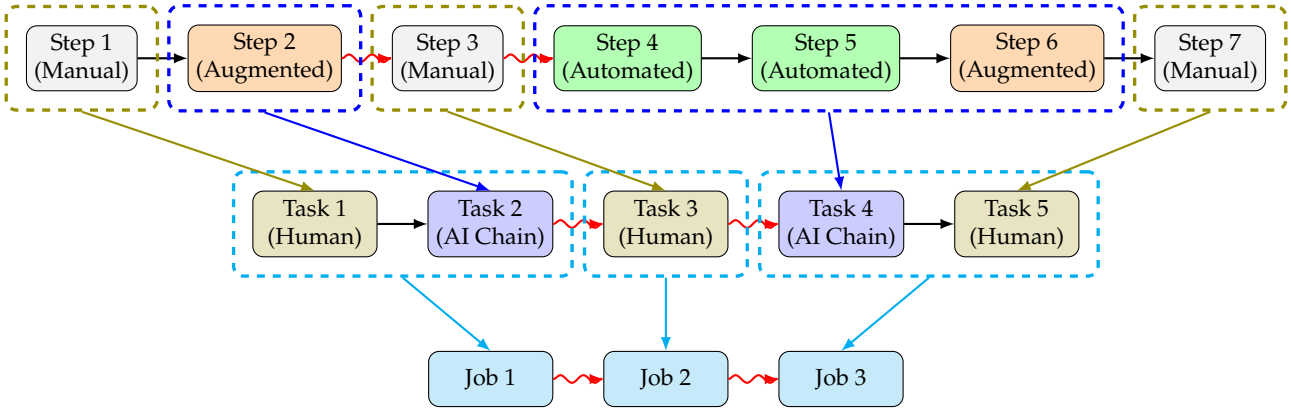
Throughout Sections 3 and 4 we held the set of steps assigned to the worker fixed, so that AI could improve only the time those steps took to execute. Here we relax this assumption and allow the firm to restructure production fully around the capabilities of AI.

Over longer horizons the firm can readjust its organizational structure, redrawing the boundaries between jobs and with them the allocation of steps to workers, while optimizing its use of AI. Moreover, adjustments in the labor market let the firm pay workers according to the actual skill requirements of the jobs it designs, effectively endogenizing wages. Below we develop the firm's long-run production problem.

5.1 Jobs and Wages

Just as an AI strategy partitions the steps into tasks, the firm's long-run choice partitions those tasks into jobs. Definition 6 formalizes that second layer of bundling, which Figure 6 adds to the example of Figure 2.

Figure 6: Illustrative Example of a Firm's Production in the Long Run



Notes: The top two layers reproduce Figure 2, with $m = 7$ steps grouped by dashed boxes into the $n = 5$ tasks below them. This figure adds the third layer, in which cyan dashed boxes group tasks into $p = 3$ jobs, each assigned to a separate worker. Red curly arrows mark the hand-offs incurred at job boundaries, and can be traced from the bottom layer up to the tasks and steps they separate.

Definition 6 (Job Design). *A* job *is a contiguous block of tasks* $J = (T_b, \ldots, T_{b'})$*, with* $b \leq b'$*, assigned to a single worker. A* job design *is a partition* $\mathcal{J} = (J_1, \ldots, J_p)$ *of the task sequence* $\mathcal{T}$ *into* p *such jobs, one for each worker the firm hires.*

In the bottom two layers of Figure 6, Tasks 1 and 2 form Job 1, Task 3 forms Job 2, and Tasks 4 and 5 form Job 3. Note that a worker's job can potentially contain both manual and AI chain tasks.

Completing a step demands not only time but also skill, by which we mean the body of knowledge and the specific know-how a worker must acquire in order to perform it. The short run had no need of this notion, as we assumed workers had acquired the skills necessary to perform all activities in their job. In the long run, however, the firm chooses which steps to assign to which worker, effectively changing the skill requirement of their jobs. We thus introduce skills to capture this other work dimension.

Accordingly, we extend Definitions 1–4 to incorporate the skill requirement of steps as well, so that the skill and time requirement of performing step i manually is (c_i^M, t_i^M), while verifying the output of one AI attempt at it costs (c_i^A, t_i^A). The two values need not coincide, since doing the work oneself calls for knowing how to do it, whereas verifying what an AI returns may call for knowledge of a different kind. As with time, a task inherits the skill of whichever mode executes it, so each task T_b in the task sequence $\mathcal{T} = (T_1, \ldots, T_n)$ carries a pair of skill and time requirements (c_b, t_b), which is (c_i^M, t_i^M) if T_b is the manual step s_i, and $\left(c_r^A, t_r^A / \prod_{i=\ell}^{r} q_i\right)$ if T_b is an AI chain spanning $(s_\ell, \ldots, s_r)$. Both cost components of a chain's pair (for a single verification attempt) are therefore inherited from its augmented endpoint. In particular, the chain demands only the skill required to

verify the augmented step's output, and an automated step inside the chain demands no skill of anyone, since no human ever handles it directly.

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 $\sum_{T_b \in J_j} t_b$. The total skill required to complete the tasks in job J_j is $\sum_{T_b \in J_j} c_b$.[18] We then assume that a worker's wage is proportional to the skill level of their job.[19] That is,

$$\text{Wage}_j = \sum_{T_b \in J_j} c_b. \tag{5}$$

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

$$\text{WageBill}_j = \left(\sum_{T_b \in J_j} c_b\right)\left(\sum_{T_b \in J_j} t_b\right). \tag{6}$$

Note what Equation (6) implies for a job. The worker is compensated for every skill the job demands over its whole duration, even though at any given moment they are performing a single task and drawing on only the portion of their knowledge that task requires. The remainder of their human capital sits idle at that moment and is paid for regardless.

A university professor is a concrete example. The job of a professor bundles research and teaching together with the small chores that come with them, and the wage it commands reflects the skill the bundle as a whole demands. Wiping the board demands almost none of that skill, yet the minutes spent on it are paid at the professor's rate rather than at what the chore would command on its own. For those minutes the expertise that makes the professor worth hiring sits unused, and the university pays for it all the same.

[18] We assume that the skills different steps require are non-overlapping, and thus add up linearly in jobs. Human capital requirements of steps do overlap in practice, but modeling that overlap would burden the results this section derives with complexities that obscure the main channels we aim to describe. That said, the results rest on a considerably weaker version of this assumption, namely that each step demand some increment of skill no other step supplies, so that a worker who takes on more steps in their job must acquire more skill. Any strictly increasing functional form delivers this; linear addition, which we adopt here, is simply the most tractable.

[19] 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 Appendix C.

5.2 Coordination and Hand-Off Cost

Absent any frictions, the firm would minimize production costs by specializing as far as it can. Equation (6) implies dividing a job in two splits its skill requirement between two workers while leaving the total time to be worked unchanged, so it strictly lowers the wage bill. Carried to its conclusion, this logic would have the firm fully specialize work and assign every task to its own worker, each of whom acquires exactly the knowledge their task requires and none of whom is ever paid for skill they are not using.

In reality, work is not organized this way, because dividing production among many workers is costly. Each division creates a point at which work must pass from one person to the next, and coordinating that intermediate output takes time, because whoever receives it has to be told what was done, why, and what state the output is now in, and whoever hands it over has to put it in a form that can be received.[20] We model this coordination friction with an additional *hand-off* cost, in time, that a worker spends passing their work on to the next worker, incurred whenever J_j is not the final job in the job design $\mathcal{J}$.

Because production in our model exhibits an order, we can be more specific about what a hand-off costs. A hand-off happens at a particular point in production, and what it costs depends on which point. Some places are natural seams, where the work has reached a self-contained intermediate output, a finished design or a signed-off specification, that can be described briefly and passed on. Others fall in the middle of something, where whoever receives the work has to reconstruct the context the previous worker had built up. We therefore attach the hand-off cost to the step at which the break occurs rather than to the worker or to the number of jobs, so that *where* the firm divides the workflow matters and not merely how often it divides. The red curly arrows in Figure 6 illustrate that hand-offs occur at job boundaries and that their cost reflects the characteristics of the last step in a worker's job.

Formally, given step s_i, we write t_i^H for the additional hand-off time spent by a worker for whom the last step of their job is s_i, and $t^H(J_j)$ for the hand-off time of job J_j, which is equal to t_i^H if s_i is the final step in J_j. The final job hands off to no one, so $t_m^H = 0$. Conditional on where the hand-off occurs in the production process, its cost does not otherwise depend on previous hand-off events. Thus the total time needed to complete job J_j, including hand-off costs, is

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

[20] Lee and Makridis (2023) formalize such interdependencies and construct an occupation-level index of coordination intensity from the O*NET dataset, which they relate to wages and to exposure to AI.

Because the worker is paid for as long as their job takes, this time enters the wage bill exactly as execution time does, so Equation (6) becomes

$$\text{WageBill}_j = \left(\sum_{T_b \in J_j} c_b\right)\left(t^H(J_j) + \sum_{T_b \in J_j} t_b\right). \tag{7}$$

Equation (7) highlights the 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 worker's wage. However, if tasks are kept separate, workers must incur hand-off costs at each job boundary. These two effects jointly determine the optimal degree of task aggregation.

Going back to the university professor example, this is perhaps why the chores fall to the professor in the first place. On wages alone the university should hand the cleaning of the board to a cheaper worker. What stops it is that the work would have to change hands mid-lecture, which means the professor must find someone, the lecture waits, and the worker must be told what is wanted. That costs more than the professorial minutes it saves, so the professor's job absorbs the chore.

We explore this trade-off in greater detail in Section 5.4, after introducing the problem the firm solves in choosing its jobs.

5.3 Long-Run Production

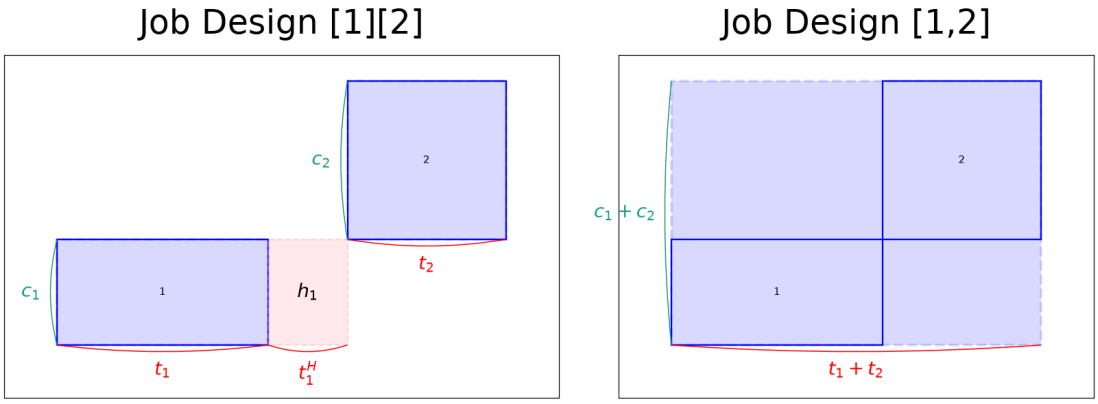
Given the sequence of steps $\mathcal{S} = (s_1, \ldots, s_m)$, the firm can design both the partition $\mathcal{T}$ of steps into tasks and the partition $\mathcal{J}$ of tasks into jobs. The choice of $\mathcal{T}$ determines the time and skill requirements of each task. Given $\mathcal{T}$, the choice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

$$\min_{\mathcal{T} \in \mathcal{P}(\mathcal{S})} \min_{\mathcal{J} \in \mathcal{P}(\mathcal{T})} \text{TotalCost}(\mathcal{J}; \mathcal{T}) = \sum_{J_j \in \mathcal{J}} \text{WageBill}_j = \sum_{J_j \in \mathcal{J}} \left[\left(\sum_{T_b \in J_j} c_b\right)\left(t^H(J_j) + \sum_{T_b \in J_j} t_b\right)\right] \tag{8}$$

where recall that, for each $T_b \in \mathcal{T}$, t_b is as described in Section 3.2 and c_b as above, both depending on the selected mode of operation for individual steps.

We refer to (8) as the firm's long-run optimization problem, as it anticipates the adjustment of worker wages to fit the skill requirements of each job and sets job responsibilities accordingly. It nests the short-run problem in (1) of Section 3.3 as the case of a single job spanning the whole production, with wage being fixed rather than chosen.

Figure 7: The Trade-off between Worker Specialization and Coordination



Notes: The blue bounded rectangles are tasks, with height the task's skill requirement (c) and width its time requirement (t). The shaded areas are the wage bills of jobs. In the left panel the two tasks are assigned to two specialized workers, and the pink rectangle is the hand-off cost incurred between them. In the right panel they are bundled into a single job, which eliminates the hand-off but requires a worker skilled enough to perform both tasks and paid at the corresponding rate throughout.

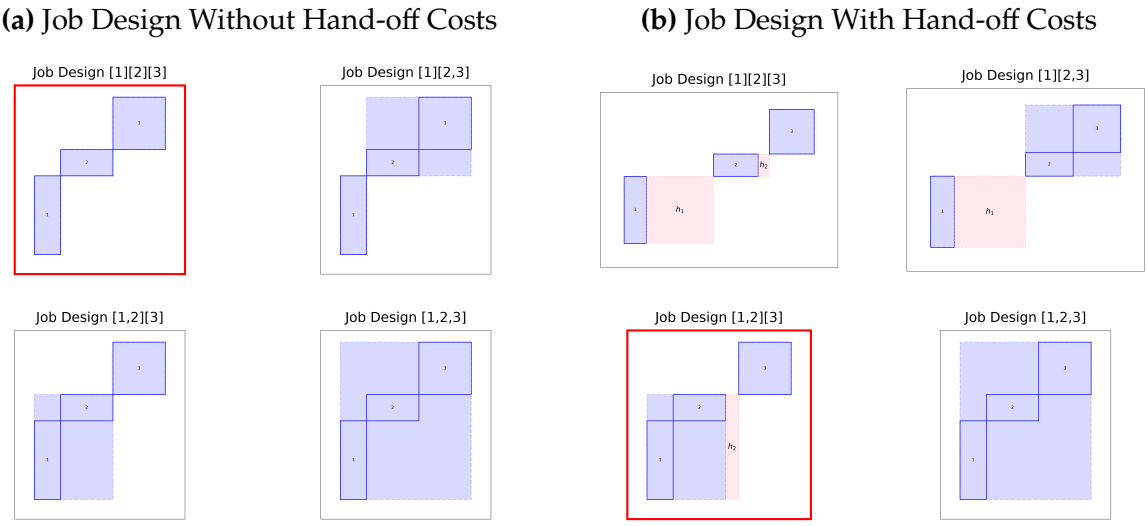
Next we discuss the implications of this optimization problem.

5.4 AI and the Limits of Worker Specialization

Here we discuss how AI affects the boundaries of jobs. Because the firm chooses those boundaries when it solves Problem (8), the answer runs through the trade-off that fixes them, between the skill a broad job demands and the coordination a narrow one requires. We set that trade-off out first, and then ask how AI moves it.

The wage bill in (7) is a product of a skill and a time, so it reads naturally as an area, which is what makes the trade-off visible. Draw each task as a rectangle whose height is its skill cost c_b and whose width its time cost t_b; a job's wage bill is then the area of the rectangle that spans the tasks it contains, as tall as the total skill it demands and as wide as the total time it takes. Figure 7 draws the two-task case, the smallest in which the firm has a choice to make. In the left panel the tasks go to two specialized workers, each of whom acquires only the skill their own task needs, so no one is paid for knowledge they are not using; but the work must pass between them, and the pink rectangle is what that hand-off costs. Notably, the hand-off shares the skill requirements of the worker doing it for an extra amount of time added to the last task in their job. In the right panel one worker performs both tasks, which removes the hand-off but obliges that worker to hold both skills throughout, the dead area sitting beside whichever task they are not currently performing. Bundling therefore buys the removal of a hand-off at the price of idle skill, and where those two prices balance determines the firm's degree of specialization.

Figure 8: Geometric Interpretation of Optimal Job Design



Notes: Each panel draws all four job designs of the three-task workflow in the convention of Figure 7, with the area of each job's bounding box its wage bill. The cost-minimizing design is outlined in red. Panel (a) sets hand-off costs to zero, in which case full specialization is optimal. Panel (b) introduces them, and the optimum moves to $[1, 2][3]$. The skill, time, and hand-off costs of the three tasks are given in the text, and Table A.2 in Appendix A reports the cost of each design.

With more than two tasks the optimum need not sit at either extreme. Consider a workflow that, under a fixed AI deployment strategy $\mathcal{T}$, has $n = 3$ tasks with the cost parameters

$$\left(c_b,\, t_b,\, t_b^H\right)_{b=1,2,3} = (3,\, 1,\, 3), \quad (1,\, 2,\, 0.5), \quad (2,\, 2,\, 0).$$

With three tasks there are four ways the firm can design jobs,[21] and Figure 8 draws all four in the same terms, once with hand-off costs switched off and once with them on; Table A.2 in Appendix A reports the cost of each job design. Panel (a) sets hand-off costs to zero and recovers the frictionless benchmark. The cost-minimizing design, outlined in red, is full specialization, with each task its own separate job. Panel (b) reinstates hand-offs and the optimum moves to $[1, 2][3]$, in which tasks 1 and 2 share a job and task 3 stands alone. The reason is the large hand-off time $t_1^H = 3$ that a boundary between tasks 1 and 2 would trigger, since bundling them avoids that hand-off at the price of a worker who must hold both skills and be paid for both throughout.

This is the machinery AI acts on, and it does so through two channels. The first is skill. An AI chain leaves the worker holding only the skill needed to verify its endpoint, so the tasks AI absorbs become short rectangles in the picture above. That cuts the idle

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

skill a bundled job pays for, which makes broader jobs cheaper to sustain and pushes the firm toward consolidation. But the same reduction works in the opposite direction as well, because what a hand-off costs is also a skill multiplied by a time. A worker whose job AI has made less skill-intensive is cheaper to leave idle at a boundary, so the hand-off the firm was bundling to avoid becomes less worth avoiding. In the workflow above, for instance, tasks 1 and 2 share a job because a skill-3 worker would otherwise sit through a hand-off of length 3; were AI to take over task 1 and leave only verification skill behind, that reason would weaken along with the wage rate it applies to.

The second channel is time. For many steps, checking what AI produces takes less time (in expectation) than performing them, so the tasks AI absorbs become narrow rectangles as well as short ones. Narrower tasks cut the idle skill a bundled job pays for, since what bundling wastes is one task's skill sitting through another task's time, and that pushes toward consolidation as the skill reduction did. The hand-off answers to the same force. We have treated t_i^H as a primitive to capture the tacit and social knowledge of transferring work, but AI plausibly lowers it as well, by absorbing the peripheral coordination a transfer requires, such as summarizing what was done or drafting the documentation that accompanies it. Cheaper hand-offs make every boundary cheaper to draw, which pushes toward narrower jobs and more specialization. Panel (a) of Figure 8 is the limiting case, with hand-offs free and full specialization optimal. Time therefore divides as skill does, in that a shorter task makes bundling cheaper, a shorter hand-off makes splitting cheaper, and the two work against each other.

What this framework delivers is therefore not a prediction about job breadth but an account of what determines it. AI presses on both sides of the specialization trade-off at once, and which way jobs move depends on the relative size of the skill and time it strips out and the coordination it leaves behind. That is an empirical question, and one the framework poses in a form that says what would have to be measured to answer it.

6 Extensions and Other Results

This section briefly provides two additional results that the framework delivers, but defers their full derivations and proofs to the appendix. Section 6.1 shows that, under additional assumptions, the firms' individual Leontief production functions can be aggregated into a single CES production function at the economy level. This connects our firm-level, sequential model of production to models that study the implications of automation from a macro perspective. Section 6.2 discusses how the cost minimization problems we have posed for the firm's short- and long-run optimization can actually be solved.

6.1 CES Representation

So far we have focused on a single firm's production: which steps it chains together, where it draws the boundaries of jobs, and what those jobs pay workers. These adjustments do not stay inside the firm. Repeated across firms, they are what determines how much labor the economy employs, how output responds to capital and to labor, and what becomes of both as AI quality improves. These are the terms the automation literature works in, writing production functions directly at the level of the aggregate economy (Acemoglu and Restrepo, 2018, 2022; Acemoglu, 2025).

Our framework can be expressed in the same terms. We distinguish two types of labor: *manual labor*, which performs the manual portion of work, and *AI management labor*, by which we mean the labor that verifies the AI's output at the end of a chain.[22] Appendix C shows that the firm-level technology developed earlier, in which the content of tasks and of jobs is chosen rather than given, can be aggregated to

$$Y = \left(\theta_A A^\rho + \theta_M M^\rho + (1 - \theta_A - \theta_M) K^\rho\right)^{\frac{1}{\rho}}, \tag{9}$$

over economy-wide AI management labor A, manual labor M, and capital K, with Y denoting aggregate output and $\rho < 0$ assumed, so that the elasticity of substitution $\sigma = 1/(1-\rho)$ lies below one. The aggregation supports Equation (9) for any such ρ rather than pinning a particular one. This is the familiar constant elasticity of substitution (CES) form (De Loecker and Syverson, 2021), with which the macroeconomic implications of automation are usually studied.

Arriving at Equation (9) is not straightforward, and it is worth stating what makes it work. No individual firm in our model substitutes AI management labor for manual labor, since producing a unit of output requires completing smaller units of production in fixed proportions, so each firm's technology is Leontief. Aggregate substitution comes instead from composition, in the manner of Houthakker (1955) and Levhari (1968). Firms have access to the same AI technology but differ in how effectively they are able to deploy it, and when AI management labor becomes relatively cheaper the economy's input mix moves because the set of firms for which using AI is worthwhile has moved, not because any firm has changed its own mix. Appendix C obtains that dispersion from the order in which a firm has to commit, settling on its AI strategy and job design before it learns

[22] The distinction between these two types of labor is deliberate rather than necessary. One could carry the aggregation through with a single type of labor, pooling the two requirements into one input. We keep them apart because the model already distinguishes the skill a step demands when a worker performs it, c_i^M, from the skill it demands when a worker verifies the output of an AI attempt at it, c_i^A, so two types of labor are the natural counterpart of a distinction we have already drawn.

how effectively it can put AI to use, and derives in closed form the distribution of output across realized AI effectiveness under which Equation (9) holds.[23]

The micro and macro representations of production are therefore not in competition. Our account of how chaining redraws tasks and jobs is not an alternative to the aggregate view of automation but a foundation for it, and the implications people draw from improving AI quality in a CES economy carry over along the margin the representation covers, the substitution between AI management labor and the rest of production. Importantly, the organization of work does not disappear in aggregation; it is absorbed into the parameters of the CES function. However, because that absorption is not invertible, aggregate data cannot reveal workflow structure, so objects such as the fragmentation index must be measured where the arrangement of work is still visible, which is what our empirical exercise in Section 7 does.

6.2 Computation of Firm's Short- and Long-run Optimums

Throughout the paper we have assumed the firm can obtain its cost-minimizing arrangement without discussing how it is found. This is not computationally trivial, as chaining prevents the firm from deciding each step's mode in isolation. Rather, the choice ranges over all feasible arrangements, of which there are exponentially many.

That exponential count overstates the difficulty, however, because many of the configurations overlap heavily. Any two arrangements that treat a fixed stretch of the workflow alike cost the same over that stretch, so the firm needs to calculate its cost just once. Both the short- and long-run problems can therefore be formulated recursively and solved by dynamic programming, building the workflow up from the beginning one step at a time and reusing the work spent on an early stretch for every arrangement of what follows. The two optimization problems differ only in how much a settled stretch has to remember about itself. In the short run the configuration's time cost is all that matters, and the optimum can be calculated exactly. In the long run the configuration's cost carries not just the time but also the skill of the worker whose job is still not terminated, and exactness gives way to an approximation the firm can make as tight as it likes. We take the cost minimization of the two horizons in turn, collecting the proofs in Appendix B.

The Short Run Optimization. With the job and its wage fixed, the firm minimizes total expected time in Problem (1). For $k \leq m$, let $C[k]$ denote the minimum time needed to

[23]The aggregate representation asks more of the economy than the firm-level argument does: capital productivity is common across firms, every firm organizes identically, realized effective AI quality is the single dimension along which firms differ, and the CES weights are tied to the skill-adjusted time requirements implied by the firm's chosen AI strategy and job design. See Appendix C for the details.

complete a hypothetical workflow containing only steps 1 through k, so that $C[0] = 0$ and $C[m]$ is the optimum. Step k is either performed manually, or augmented as the endpoint of an AI chain reaching back to some earlier step ℓ; in either case the steps before that point are optimized separately, which gives

$$C[k] = \min\left\{\underbrace{C[k-1] + t_k^M}_{\text{step } k \text{ manual}},\ \underbrace{\min_{\ell<k}\left(C[\ell] + \frac{t_k^A}{\prod_{i=\ell+1}^{k} q_i}\right)}_{\text{step } k \text{ ends a chain begun at } \ell+1}\right\}. \tag{10}$$

Because time costs enter Problem (1) additively, a stretch's time cost is all that the recursion in (10) needs from it. Each of the m values costs $O(m)$ to evaluate, since the chain ending at step k may begin anywhere before it. This makes the computational complexity of the algorithm $O(m^2)$. Proposition 3 formalizes this.

Proposition 3 (Short-run AI Deployment)**.** *Given a workflow with m steps, the AI strategy that solves Problem (1) can be calculated in time $O(m^2)$ via dynamic programming.*

The Long Run Optimization. The firm now chooses job boundaries as well, and the objective in (8) multiplies each job's total skill by its total time. A task's contribution to cost therefore depends on how much skill and how much time its job already carries, so the position alone no longer summarizes a stretch and we carry two further coordinates. Let $R(i, c, t)$ be the minimum cost of completing steps 1 through i when the worker currently accumulating tasks already holds total skill c and total time t (including their hand-off) and let $V(i)$ be the corresponding cost when that worker's job closes at step i, so that $V(i) = R(i, 0, t_i^H)$ with the job closure option removed. The base case is $R(0, c, t) = c\,t$, and the firm faces three options at each step rather than two,

$$R(i, c, t) = \min\left\{\underbrace{c\,t + V(i)}_{\text{close the job}},\ \underbrace{R\big(i-1,\ c + c_i^M,\ t + t_i^M\big)}_{\text{step } i \text{ manual}},\ \underbrace{\min_{0\le r<i} R\left(r,\ c + c_i^A,\ t + \frac{t_i^A}{\prod_{i'=r+1}^{i} q_{i'}}\right)}_{\text{step } i \text{ ends a chain begun at } r+1}\right\}, \tag{11}$$

with the optimum at $V(m)$, recalling that the final job hands off at no cost.[24] Closing a job is where the accumulated skill and time are finally multiplied into a wage bill, which is why the recursion has to carry both alongside the position i. That is also what makes the long run more complicated to solve.

[24] The first option is available only when the active worker already holds a task, $c > 0$; the states with $c = 0$ are the job-opening states, and taking their value to be $V(i)$ is what keeps the recursion well founded.

Skill and time costs are continuous, so the recursion must distinguish between infinitely many situations rather than the m of the short run, and it cannot be carried out to completion. Proposition 4 states that the optimal cost can nonetheless be approximated up to an error term that can be made arbitrarily small in polynomial time.

Proposition 4 (Long-run AI Deployment and Job Design)**.** *Fix a workflow with m steps whose skill and time costs, and whose non-zero hand-off costs, lie in $[1/B, B]$ for some $B > 0$. For any $\epsilon > 0$, a pair of AI strategy $\mathcal{T}$ and job design $\mathcal{J}$ whose long-run cost in (8) is within a factor $(1+\epsilon)$ of the minimum can be computed in time $O(m^4 \epsilon^{-2} \log^2(mB))$ via dynamic programming.*

Together these results also suggest a use for the framework beyond what we do here. Paired with detailed measurement of the time and skill content of individual tasks, the algorithms above could be run on real workflows to ask how a given advance in AI capability would redraw them.

7 Empirical Evaluation

In this section we empirically test the implications of our model. The predictions we take to the data are those of the short-run problem of Section 3, in which the boundaries of jobs are fixed and AI acts only on how the steps within them are executed. This is a deliberate restriction rather than a limitation of the theory. The long-run margins of Section 5 require firms to reorganize and workers to retrain, and not enough time has passed since AI tools became widely available for these adjustments to appear in the data we observe. What we can observe is where AI is being deployed inside workflows as they are currently arranged, which is precisely what the short-run problem speaks to. The three predictions we test are: (1) AI-executed steps tend to appear next to each other in the production sequence, forming AI chains; (2) a step is more likely to be executed by AI when its neighbors are AI-executed; and (3) workflows in which AI-exposed steps are more dispersed have fewer of those steps actually executed by AI.

Our analysis requires three attributes of each step: its AI exposure status (exposed or unexposed to AI), which is how we measure whether a step is AI-able in the sense of Section 4, its realized mode of execution (manual or AI-executed), and its position in the production sequence. We draw on five sources:

1. The O*NET 27.3 Database (National Center for O*NET Development, 2023),
2. Human-generated AI exposure labels for O*NET tasks from Eloundou et al. (2024),
3. Realized AI execution labels for O*NET tasks from the Anthropic Economic Index (Handa et al., 2025),
4. A GPT-generated task ordering for each O*NET occupation,

5. The APQC Process Classification Framework (PCF) (APQC, 2026).

The first four build our O*NET sample, in which a workflow is an occupation, and the fifth builds our PCF sample, in which a workflow is a process group. All sources are publicly available datasets apart from the O*NET task ordering, which we generate ourselves. Below we briefly discuss both samples before testing the predictions. Appendices SA.A and SA.F respectively give the construction details of the O*NET and PCF samples.

The O*NET database is the standard decomposition of work in the United States. It breaks each of more than 850 occupations into a list of constituent tasks, roughly 18,000 in total, and has served as the backbone of a large empirical literature on the task content of jobs and their exposure to new technologies (e.g., Autor and Handel, 2013; Deming, 2017; Brynjolfsson et al., 2018; Webb, 2020; Felten et al., 2021). Eloundou et al. (2024) map these O*NET tasks to AI exposure by developing a rubric asking whether access to an LLM would reduce the time required to perform a task by at least half while preserving output quality, and applying it to every task using both human annotators and GPT-4. We use their human-generated labels and treat tasks in their E1 and E2 categories as exposed to AI and all other tasks as unexposed. Whereas these labels capture what AI could in principle do, the Anthropic Economic Index (Handa et al., 2025) captures what AI actually does. It maps millions of conversations between users and Claude, Anthropic's AI chatbot, to O*NET tasks and classifies each conversation by whether the interaction augments or automates the task, providing a task-level measure of realized AI usage.[25]

The PCF catalogues business processes such as procurement, order management, after-sales service, inventory, and regulatory compliance, with APQC's member working groups recording the activities that compose each process in the order they are performed. Pooling APQC's Cross-Industry framework with its industry-specific ones provides 13,482 steps in 525 unique process groups. Since AI exposure and execution are defined on O*NET tasks, we carry those labels onto PCF steps by matching each PCF step to its nearest O*NET task using natural language embeddings, preserving unmatched steps in the workflows and coding them as neither AI-exposed nor AI-executed. The lossy label transfer leaves 12% (4%) of PCF steps AI-exposed (AI-executed), against 44% (13%) in O*NET.

The two corpora should be viewed as complements in our analysis. The PCF's work-

[25]One feature of the data shapes how we use it throughout. The model's automated versus augmented distinction is defined by production sequencing, whereas the Anthropic labels are assigned on task-level feasibility and are agnostic to workflow position, so the data occasionally shows an automated task followed by a manual one, which the model's definition of a chain does not allow. We therefore do not distinguish the two labels in what follows and treat both as indicating AI execution. Appendix SA.A documents the discrepancy and provides examples.

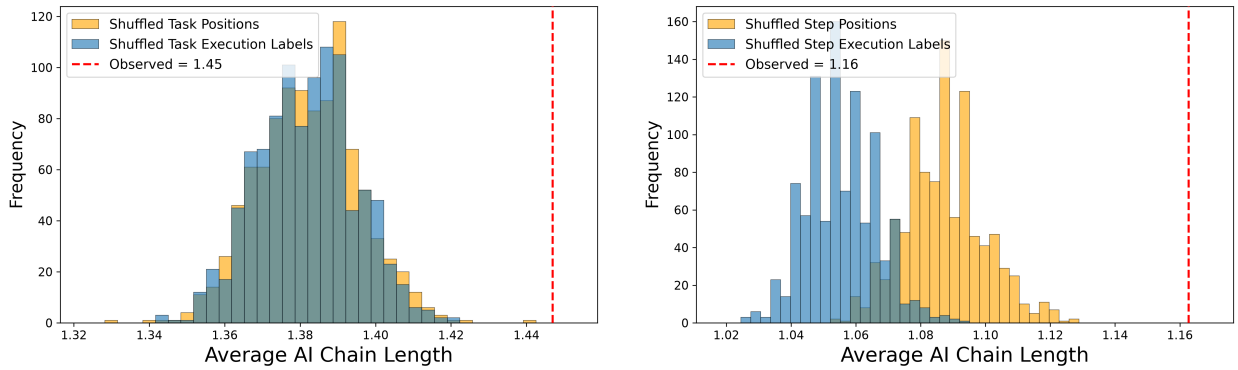
flow orderings are recorded by practitioners and are thus native to the dataset. However, the PCF covers only operational business processes, and its AI exposure and execution content is thin because of the lossy PCF–O*NET match. The O*NET corpus, on the other hand, has broader coverage of economic activity but lists an occupation's tasks without providing the order in which they are performed. To make the O*NET dataset suitable for our analysis, we construct workflow orderings for it by prompting GPT-5-mini (through Expected Parrot; Horton and Horton, 2024) with each occupation's complete set of tasks and asking for the most reasonable order in which they would be performed in practice.

Because our O*NET results rest on a task ordering that we generate with a language model we probe their validity using three robustness tests. First, we show that the task orderings, and by extension our conclusions, remain unchanged across ten deliberately varied alternative prompts (Appendix SA.D). Second, we test our task sequencing procedure against the PCF corpus as well as eight process-mining event logs from the 4TU.ResearchData database (de Leoni and Mannhardt, 2015; Mannhardt, 2016; van Dongen, 2017, 2020) whose workflow orderings are known and show that our approach restores the correct workflow order to a substantial degree (Appendix SA.F). Third, we restrict the O*NET sample to progressively more frequently performed tasks by dropping rarely performed "filler" steps that can separate steps in an actual workflow and find that our conclusions remain unchanged (Appendix SA.E). Together, findings of these tests imply that our O*NET results are unlikely to be artifacts of the prompt, the task sequencing procedure, or the way O*NET catalogues tasks.

Finally, the structure of our theoretical framework closely aligns with the structure of both datasets. Each production step in the model corresponds to a PCF step or an O*NET task, and each workflow corresponds to the model's step sequence. In the absence of AI, a step (that is, a PCF step or an O*NET task) is executed manually and maps directly into a task under our model's definition. Thus, without AI, a PCF step or an O*NET task can be viewed as both a step and a task, which matches how both datasets list their units of work. Under AI execution, however, multiple steps (corresponding to a subset of PCF steps or O*NET tasks) may be chained together and executed jointly, forming a composite task as defined in Section 3. In the rest of this section, we use the terms step, PCF step, O*NET task, and task interchangeably and expect the reader to keep these distinctions in mind. We similarly use the term workflow for both an O*NET occupation and a PCF process group without restating the mapping each time.

Figure 9: Average Length of AI Chains: Actual versus Placebo Datasets

(a) O*NET Occupations **(b)** PCF Process Groups



Notes: Red dashed lines show the observed average AI chain length in the actual dataset. Panel (a) uses the O*NET sample, in which a workflow is an occupation and the ordering is the GPT-imputed task sequence; Panel (b) uses the PCF sample, in which a workflow is a process group and the ordering is the one documented by practitioners, with exposure and execution labels transferred from each step's nearest O*NET task. Each panel overlays both placebo distributions, each over 1,000 draws. The orange histogram randomizes the positions of steps within each workflow, and the blue histogram reassigns steps, each carrying its execution label, across workflows within the same group, preserving workflow sizes; the grouping is the major SOC group in Panel (a) and the PCF Category in Panel (b). The horizontal scale differs between the panels, since only 4% of PCF steps carry an AI-execution label after the transfer against 13% in O*NET, which lowers the attainable chain length throughout the PCF sample.

7.1 Prediction #1: Tendency to Have Runs of Consecutive AI-executed Tasks

The first prediction of the model that we test is that AI-executed steps appear in contiguous blocks rather than being randomly scattered in the workflow. This prediction follows directly from Definition 4 on how AI chains are formed in the model. If firms deploy AI, chaining can lower costs because the worker verifies the output once only at the end of the chain rather than after every step.

We begin with the O*NET sample. We compute the observed average AI chain length and benchmark it against two randomization placebos. First, we permute task positions within each occupation's workflow. Second, we reassign tasks, each carrying its AI-execution label, randomly across occupations within the same major SOC occupation group. The two placebos isolate different margins: the first holds each occupation's task composition fixed and randomizes the ordering in its workflow, while the second holds each major occupation group's AI intensity and each occupation's size fixed and randomizes which tasks, and hence which execution labels, an occupation contains.

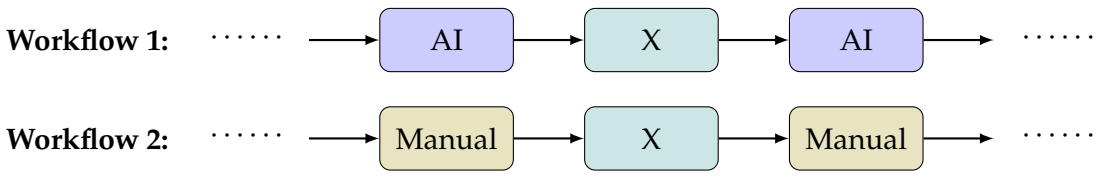
Panel (a) of Figure 9 overlays the two placebo distributions of average AI chain length, one across 1,000 task position permutations and one across 1,000 task AI-execution label reassignment permutations, and compares both with the observed mean AI chain length, marked by the dashed vertical line. We observe that the average AI chain length in the data is 1.45 and is noticeably larger than in both placebo distributions. This indicates

that AI-executed tasks tend to cluster within occupations, forming longer AI chains than would arise under either type of randomization. At the same time, the modest magnitude of 1.45 indicates that long AI chains remain rare, perhaps reflecting the early stage of AI adoption; our model predicts that chains will lengthen as AI quality improves.

Next we repeat the exercise on the PCF sample, substituting its own units throughout, with a process group standing in for an occupation, the activity sequence practitioners documented for the GPT-imputed task sequence, and the PCF Category for the major SOC occupation group within which the second placebo reassigns. Nothing else about the test changes, and Panel (b) of Figure 9 reports the same verdict. The average AI chain there spans 1.16 steps, against 1.45 in O*NET. The smaller magnitude is mechanical, since AI execution in the PCF is thinner than in O*NET, which leaves far less opportunity for long runs to form and puts the ceiling on the statistic well below the O*NET sample's. What the PCF sample can still ask is whether the runs that do form are longer than the same labels arranged at random along the same sequences, and they are. Because no language model placed these steps in order, the contiguity here is a property of the documented workflow and not of our sequencing procedure.

7.2 Prediction #2: AI Execution Is More Likely Next to AI-executed Steps

The second prediction of the model that we test is that when the same step appears in two workflows, in one sitting between strong AI performers and in the other located between manual ones, it is more likely to be executed by AI in the first. This is the empirical content of Proposition 1. Intuitively, when a step is surrounded by AI-executed steps, there are local gains from grouping it with its neighbors as part of a long AI chain that are absent when the step is situated next to manual steps. For example, step X is more likely to be executed by AI in the first worklfow below than in the second:



Testing this prediction requires identifying steps that appear across different workflows. Because O*NET tasks are occupation-specific, no single task is shared across occupations in the data. To work around this, we rely on O*NET's broader categories of Detailed Work Activities (DWAs), which group conceptually similar tasks across occupations. In our baseline O*NET sample, we drop tasks that are mapped to multiple DWAs. We further restrict attention to DWAs that contain tasks appearing in more than one occupation. Together, these restrictions reduce the sample to 10,708 tasks spread

across 1,748 DWAs. We treat the remaining tasks as instances of the same step appearing across different occupations and estimate the following regression[26]:

$$\Pr(\text{is_ai}_k = 1 \mid X_k) = \Lambda\Big(\beta_0 + \beta_1 \,\text{prev2_is_ai}_k + \beta_2 \,\text{prev_is_ai}_k + \beta_3 \,\text{next_is_ai}_k + \beta_4 \,\text{next2_is_ai}_k\Big), \tag{12}$$

where Λ is the logistic CDF and is_ai_k indicates whether step k is executed by AI. The variables $prev2_is_ai_k$ and $prev_is_ai_k$ equal 1 if the step two positions before or immediately before step k is AI-executed, respectively, and $next_is_ai_k$ and $next2_is_ai_k$ are defined analogously for the steps immediately after and two positions after step k. In implementations we also control for step k's AI exposure status and the number of steps in its workflow.

Table 2 reports average marginal effects from estimating Equation (12) for a series of specifications. The baseline in column (1) shows that AI execution anywhere in the focal task's local context is associated with a higher probability that the task is AI-executed, at one and at two positions away alike, with the effects of the adjacent steps roughly twice the size of those two positions out. The remaining columns ask how much of this survives more demanding comparisons. Columns (2)–(3) absorb occupation family differences with SOC major and minor group fixed effects, and column (4) restricts the variation to within detailed work activities with DWA fixed effects, addressing differences in AI-ability across types of task. Columns (5)–(6) repeat the exercise controlling for the number of same-DWA tasks in the focal task's occupation, which rules out mechanical proximity inflating the estimates when several same-DWA tasks sit in one workflow; columns (4)–(6) are all estimated on the same restricted sample, so they are comparable with one another rather than with column (1). Across the specifications that absorb occupation-family or DWA heterogeneity, columns (2)–(4) and (6), the pattern is the same. The immediate neighbor effect attenuates to between 0.04 and 0.06, relative to the baseline for columns (2)–(3) and to the restricted-sample estimate of column (5) for columns (4) and (6), but stays statistically and economically meaningful, while the effects two positions away fall to between −0.01 and 0.01 and lose significance. Column (5) carries no fixed effects and, estimated on the restricted sample of column (4), reproduces the baseline pattern rather than the attenuated one. That the baseline estimates attenuate tells us they partly reflect systematic differences across occupation families; that the immediate neighbor effect persists tells us the local pattern is not merely such a difference.

Because we have no external benchmark for what constitutes a large effect here, the

[26] In a robustness exercise we refine our definition of "similar" tasks. The resulting estimates closely mirror the patterns observed in the full sample in both sign and magnitude. See Appendix SA.B for details.

Table 2: Neighboring Tasks' AI Execution Status and a Task's AI Execution

	Probability that Focal Task (k) is AI-executed					
	(1)	(2)	(3)	(4)	(5)	(6)
Task ($k-2$) is AI-executed	0.05***	0.01	0.00	-0.01	0.06*	-0.01
	(0.02)	(0.01)	(0.01)	(0.03)	(0.03)	(0.03)
Task ($k-1$) is AI-executed	0.11***	0.06***	0.05***	0.05***	0.13***	0.05***
	(0.02)	(0.01)	(0.01)	(0.02)	(0.03)	(0.02)
Task ($k+1$) is AI-executed	0.11***	0.05***	0.05***	0.04**	0.10***	0.04**
	(0.02)	(0.01)	(0.01)	(0.02)	(0.02)	(0.02)
Task ($k+2$) is AI-executed	0.04**	-0.00	-0.01	0.00	0.04	0.00
	(0.02)	(0.01)	(0.01)	(0.02)	(0.03)	(0.02)
Pseudo R^2	0.130	0.183	0.180	0.197	0.030	0.197
Observations	10,708	10,708	9,861	4,096	4,096	4,096
SOC Group FE		Major	Minor			
DWA FE				✓		✓
NumTasks in DWA-Occupation Control					✓	✓

Notes: Clustered standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1. This table reports average marginal effects from estimating Equation (12). The dependent variable in all specifications is an indicator for whether task k is AI-executed (is_ai$_k$). The sample is restricted to similar tasks across occupations, identified as tasks belonging to the same DWA in the O*NET dataset. All specifications control for the AI exposure status of task k and for the total number of tasks in task k's occupation. Standard errors are bootstrapped using $B = 200$ replications and clustered at the DWA level. The estimating sample varies across columns. Minor-group fixed effects in column (3) drop groups with no within-group variation in the outcome, removing 847 tasks. The DWA fixed effects in columns (4) and (6) are identified only from DWAs containing both an AI-executed and a non-AI-executed task, a condition met by 534 of the 1,748 DWAs in the sample, leaving 4,096 tasks; column (5) carries no DWA fixed effects but is estimated on that same restricted sample so that it is directly comparable with column (4). The number of DWA clusters is 1,748 in columns (1)–(2), 1,705 in column (3), and 534 in columns (4)–(6).

magnitudes are hard to read on their own. We therefore re-estimate Equation (12) on 1,000 placebo datasets that randomize task positions within occupations, for the four specifications in columns (1)–(4) of Table 2, and report the result in Figure A.1 of Appendix A. Randomizing positions strips the ordering of any information, so the placebo distributions show what these coefficients would look like when the position of a task in the workflow does not matter. In each case, the actual orderings deliver stronger immediate-neighbor effects and weaker distant-neighbor effects than the placebos. This implies that local work context and proximity do work that the reshuffled orderings cannot reproduce by chance.

Overall, the results indicate that AI execution of immediate neighbors of the focal task is associated with a higher likelihood that the task is executed by AI, whereas the execution status of more distant neighbors matters much less, especially after controlling for a range of relevant factors. This aligns with the relatively short average AI chain length observed in the data (mean length 1.45 reported in Subsection 7.1), implying that tasks two positions away rarely fall in the same chain as the focal task. It is possible that with

improvements in AI quality, chains become longer and execution status of more distant neighbors becomes more relevant for the execution mode of the focal task.

For this prediction, we do not attempt a counterpart on the PCF sample, where too few similar steps are observed with both execution outcomes to support the comparison. See Appendix SA.F for details.

7.3 Prediction #3: AI Execution Is Lower Where AI-able Steps Are Dispersed

The two tests so far were local, in that they operated over a small window around a step. The last prediction of the model that we test is how the positioning of steps over the whole workflow matters for AI execution. Specifically, we test whether realized AI execution in a production sequence varies with how clustered or dispersed its AI-exposed steps are, and not merely with how many of its steps are exposed. This is the empirical content of Proposition 2.

To measure the fragmentation of AI-exposed steps in the data, we adapt the fragmentation index of Section 4.2 to our empirical setting. The *empirical fragmentation index* (EFI) is the ratio of the number of *potential tasks*, as defined by our theoretical model, to the total number of steps, which is fixed and is measured by the raw count of steps in the workflow. Consider a workflow with five steps. If none of the five steps are exposed to AI, then the number of tasks in the theoretical sense is five, and the EFI equals $5/5 = 1$. If instead two consecutive steps are exposed to AI (and thus can form an AI chain of length two when all AI-exposed steps turn into AI-executed steps), the number of tasks in the model falls to four and the EFI becomes $4/5 = 0.8$. Thus, the empirical fragmentation index decreases as more AI-exposed steps can potentially be consolidated into longer AI chains.[27]

Two components govern the empirical fragmentation index, and only one is the subject of Prediction #3. Write m_w for the number of steps in a workflow, k_w for how many of them are AI-exposed, and r_w for the number of maximal blocks of consecutive AI-exposed steps they form, each block being one potential AI chain. EFI decomposes exactly as

$$\text{EFI}_w \;=\; \frac{m_w - (k_w - r_w)}{m_w} \;=\; 1 - \underbrace{\frac{k_w}{m_w}}_{\text{exposure level}} + \underbrace{\frac{r_w}{m_w}}_{\text{arrangement}}, \tag{13}$$

where the last term is the object of interest. Holding k_w fixed, r_w runs from 1 when the

[27] The EFI is, after normalizing by the number of steps, the special case of the realized fragmentation expressed in (3) in which every step takes one unit of time in either mode, $t_i^M = t_i^A = 1$, and AI either always or never succeeds at a step, $q_i \in \{0, 1\}$. It therefore captures the arrangement of a workflow's AI-able steps, which is what we observe in the data, and not the differences in time cost and reliability that the theoretical index also prices. Since $t_i^M = 1$ satisfies the hypothesis of (4), the EFI is exactly one minus the share of adjacent AI-able pairs in the workflow, so it inherits the ordering that expression establishes.

Table 3: AI-exposed Step Fragmentation and AI Execution Outcomes

	Share of AI-executed Steps					
	O*NET Occupations			PCF Process Groups		
	(1)	(2)	(3)	(4)	(5)	(6)
Share of AI-exposed Steps	0.49***	0.48***	0.39***	0.28***	0.42***	0.28***
	(0.10)	(0.09)	(0.09)	(0.10)	(0.10)	(0.10)
Empirical Fragmentation Index	-0.01	-0.09	-0.04	-0.35***	-0.26**	-0.34***
	(0.10)	(0.09)	(0.09)	(0.10)	(0.11)	(0.11)
Fixed Effect		SOC Major	SOC Minor		PCF Category	Framework
Number of AI-able Steps Control	✓	✓	✓	✓	✓	✓
R-squared	0.35	0.64	0.72	0.37	0.49	0.38
Adj. R-squared	0.34	0.63	0.67	0.36	0.47	0.36
Observations	872	872	872	525	525	525

Notes: Standardized coefficients. Heteroskedasticity-robust standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1. This table reports results from estimating Equation (14). The dependent variable in all specifications is the share of steps executed by AI in the workflow. "Share of AI-exposed Steps" is the share of steps carrying an E1 or E2 label, the same label set from which the Empirical Fragmentation Index is constructed; the index itself captures how dispersed AI-exposed steps are across the workflow. All specifications control for the number of AI-able steps in the workflow. Columns (1)–(3) use the O*NET sample, where a workflow is an occupation and the ordering is the GPT-imputed task sequence; columns (4)–(6) use the PCF sample, where a workflow is a process group and the ordering is the one documented by practitioners. In the PCF sample, exposure and execution labels are transferred from each step's nearest O*NET task. All variables are standardized to unit standard deviation within their own sample, so the reported coefficients are in standard-deviation units and are not directly comparable in levels across the two samples.

AI-exposed steps are perfectly clustered to k_w when no two of them are adjacent. To read the index as a measure of dispersion, we must control for the share of AI-exposed steps, which nets out the level term and leaves the arrangement component. We therefore estimate the following regression:

$$\begin{aligned} \text{ai_execution}_w = \beta_0 + \beta_1 \, \text{ai_exposure}_w + \beta_2 \, \text{empirical_fragmentation_index}_w \\ + \beta_3 \, \text{num_ai_able_steps}_w + \varepsilon_w, \end{aligned} \tag{14}$$

where $ai_execution_w$ is the share of steps executed by AI in workflow w, $ai_exposure_w$ is the share of AI-exposed steps, $empirical_fragmentation_index_w$ is the index defined above, and $num_ai_able_steps_w$ is the number of AI-able steps in workflow w.[28] The model predicts that workflows with higher AI exposure should have a higher share of steps executed by AI ($\beta_1 > 0$), and that, conditional on exposure, AI execution should be lower where AI-able steps are more dispersed ($\beta_2 < 0$).

Table 3 reports the results. In each sample we estimate three specifications: no fixed effects, and then two groupings of workflows (SOC major and minor occupation groups in O*NET, PCF Category and Framework in the PCF). The fixed effects ensure the results

[28] Controlling for workflow length m_w instead, or dropping the count control altogether, moves the fragmentation coefficient by at most 0.04 standard deviations and changes no verdict.

are not driven by a narrow subset of occupation or process families. The exposure coefficient behaves as the model predicts in every column of both samples. A one-standard-deviation increase in the share of AI-exposed steps is associated with a 0.39 to 0.49 standard-deviation increase in the AI-executed share in O*NET and a 0.28 to 0.42 standard-deviation increase in the PCF, significant at the 1% level throughout.

The fragmentation coefficient is where the two samples part ways. In the O*NET sample it carries the predicted negative sign in all three specifications, but the point estimates are small and none is statistically distinguishable from zero. On O*NET occupations therefore we do not detect the workflow-level channel, but the estimates are not precise enough to rule it out either. A modest fragmentation effect would thus go undetected in this sample. On the PCF's documented sequences, however, the coefficient is negative and statistically significant in all three columns, at -0.35, -0.26 and -0.34, so that process groups whose AI-exposed steps are more scattered convert less of their exposure into execution.

That the channel is detected in the PCF and not in O*NET is what a test built on arrangement would lead one to expect. The fragmentation index reads the ordering of a workflow as a whole, so it stands its best chance of detection where that ordering is recorded rather than inferred, which is the PCF's advantage over the imputed sequences in O*NET. The PCF process groups cover a smaller part of economic activity than O*NET's occupational structure, and they inherit their labels through a lossy match, so the channel is documented on a narrower set of production workflows rather than across the economy. We read the results together as saying that the arrangement channel is present where the arrangement is measured well enough to see it.

We must emphasize that what we document in these exercises are associations that speak to the mechanism the model describes, and not causal effects. Measuring the latter would require exogenous variation in how much of a workflow is handed to AI and/or in how that workflow is arranged, which is not what we work with.

8 Conclusion

This paper develops a task-based framework for understanding how advances in AI reshape automation and production. By modeling production as a sequence of steps that the firm aggregates into tasks, we show how firms optimally allocate work between humans and AI, whether in augmented or fully automated form. A central feature of the model is the possibility of AI chains, in which multiple steps are executed by AI and only the final output is verified by a human. We show that chaining can overturn comparative advantage logic in factor assignment by amplifying the importance of local work context.

That is, whether a unit of work is assigned to a human or to AI depends not only on the characteristics of the step in question but also on those of its neighbors in production. By the same logic, how much a workflow stands to gain from automation depends on how it is arranged and not only on how much of it is exposed to AI. When AI-able steps are clustered, one verification at the end of a chain covers them all; when they are scattered across the workflow, each must be verified separately, and the gains from automation shrink. We capture this in a fragmentation index, a measure of how dispersed a workflow's AI-able steps are, which approximates the cost of the firm's optimal AI deployment.

We characterize the firm's optimal AI deployment decision across two horizons: a short run, in which the boundaries of jobs and what they pay are both fixed, so AI only improves the time it takes to produce a unit of output; and a long run, in which the firm redraws those boundaries jointly with its AI deployment, trading the lower wages of narrower jobs against the coordination costs of maintaining more of them.

We also show that improvements in AI quality can induce discrete reorganizations of work. A key implication of chaining is that marginal increases in AI quality may generate little or no cost savings until a threshold is crossed, after which the optimal production configuration changes discontinuously, so that the marginal returns to improving AI are non-monotone. Our framework thus helps explain why firms invest so heavily in AI capabilities even when short-run returns appear limited, and it is consistent with the J-curve pattern of technology adoption (Brynjolfsson et al., 2021).

We complement our theoretical analysis with empirical evidence consistent with the core mechanisms implied by the model. Using a task-level dataset of AI exposure and AI execution outcomes, we find that AI-executed steps tend to appear in contiguous AI chains, that when comparable steps appear across occupations those sitting in a more AI-executed local context are significantly more likely to be executed by AI themselves, and that workflows whose AI-exposed steps are more fragmented exhibit lower realized AI execution conditional on exposure. These patterns align with the central economic forces highlighted by the model: the importance of positional relationships among steps, the local complementarities created by AI chains, and the role of AI-exposed step dispersion in shaping execution outcomes.

References

Acemoglu, D. (2025). The Simple Macroeconomics of AI. *Economic Policy 40*(121), 13–58.

Acemoglu, D. and D. Autor (2011). Skills, Tasks and Technologies: Implications for Employment and Earnings. In *Handbook of Labor Economics*, Volume 4, pp. 1043–1171. Elsevier.

Acemoglu, D., F. Kong, and P. Restrepo (2024). Tasks At Work: Comparative Advantage, Technology and Labor Demand. *NBER Working Paper, No. 32872*.

Acemoglu, D. and P. Restrepo (2018). The Race between Man and Machine: Implications of Technology for Growth, Factor Shares, and Employment. *American Economic Review 108*(6), 1488–1542.

Acemoglu, D. and P. Restrepo (2019). Automation and New Tasks: How Technology Displaces and Reinstates Labor. *Journal of Economic Perspectives 33*(2), 3–30.

Acemoglu, D. and P. Restrepo (2022). Tasks, Automation, and the Rise in U.S. Wage Inequality. *Econometrica 90*(5), 1973–2016.

Agrawal, A., J. S. Gans, and A. Goldfarb (2019). Exploring the Impact of Artificial Intelligence: Prediction versus Judgment. *Information Economics and Policy 47*, 1–6.

APQC (2026). Process Classification Framework (PCF) – Cross Industry, Version 8.0. https://www.apqc.org/process-frameworks. American Productivity and Quality Center. The industry-specific frameworks are drawn from the same collection.

Autor, D., C. Chin, A. Salomons, and B. Seegmiller (2024). New Frontiers: The Origins and Content of New Work, 1940–2018. *The Quarterly Journal of Economics 139*(3), 1399–1465.

Autor, D., C. Chin, A. Salomons, and B. Seegmiller (2026). What makes new work different from more work? *Annual Review of Economics 18*.

Autor, D. and N. Thompson (2025). Expertise. *Journal of the European Economic Association 23*(4), 1203–1271.

Autor, D. H. and M. J. Handel (2013). Putting Tasks to the Test: Human Capital, Job Tasks, and Wages. *Journal of Labor Economics 31*(S1), S59–S96.

Autor, D. H., F. Levy, and R. J. Murnane (2003). The Skill Content of Recent Technological Change: An Empirical Exploration. *The Quarterly Journal of Economics 118*(4), 1279–1333.

Becker, G. S. and K. M. Murphy (1992). The Division of Labor, Coordination Costs, and Knowledge. *The Quarterly Journal of Economics 107*(4), 1137–1160.

Bloom, N., R. Sadun, and J. Van Reenen (2016). Management as a Technology? *The Quarterly Journal of Economics 132*(3), 1–60.

Bresnahan, T. F., E. Brynjolfsson, and L. M. Hitt (2002). Information Technology, Workplace Organization, and the Demand for Skilled Labor: Firm-level Evidence. *The Quarterly Journal of Economics 117*(1), 339–376.

Brynjolfsson, E., D. Li, and L. Raymond (2025). Generative AI at Work. *The Quarterly Journal of Economics 140*(2), 889–942.

Brynjolfsson, E., T. Mitchell, and D. Rock (2018). What Can Machines Learn, and What Does It Mean for Occupations and the Economy? In *AEA Papers and Proceedings*, Volume 108, pp. 43–47.

Brynjolfsson, E., D. Rock, and C. Syverson (2021). The Productivity J-Curve: How Intangibles Complement General Purpose Technologies. *American Economic Journal: Macroeconomics 13*(1), 333–72.

Coase, R. H. (1937). The Nature of the Firm. *Economica 4*(16), 386–405.

de Leoni, M. and F. Mannhardt (2015). Road Traffic Fine Management Process. https://doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5. 4TU.ResearchData.

De Loecker, J. and C. Syverson (2021). An Industrial Organization Perspective on Productivity. In *Handbook of Industrial Organization*, Volume 4, pp. 141–223. Elsevier.

Dell'Acqua, F., E. McFowland III, E. R. Mollick, H. Lifshitz-Assaf, K. C. Kellogg, S. Rajendran, L. Krayer, F. Candelon, and K. R. Lakhani (2026). Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of Artificial Intelligence on Knowledge Worker Productivity and Quality. *Organization Science*. Articles in Advance.

Deming, D. J. (2017). The Growing Importance of Social Skills in the Labor Market. *The Quarterly Journal of Economics 132*(4), 1593–1640.

Demirer, M., L. Musolff, and L. Yang (2026). Writing Code vs. Shipping Code: Productivity Effects Across Generations of AI Coding Tools. *Working Paper*.

Dessein, W. and T. Santos (2006). Adaptive Organizations. *Journal of Political Economy 114*(5), 956–995.

Eloundou, T., S. Manning, P. Mishkin, and D. Rock (2024). GPTs are GPTs: Labor Market Impact Potential of LLMs. *Science 384*(6702), 1306–1308.

Farboodi, M., A. J. Koh, and A. Xia (2026). Data-Driven Automation. *NBER Working Paper, No. 35320*.

Felten, E., M. Raj, and R. Seamans (2021). Occupational, Industry, and Geographic Exposure to Artificial Intelligence: A Novel Dataset and its Potential Uses. *Strategic Management Journal 42*(12), 2195–2217.

Frey, C. B. and M. A. Osborne (2017). The Future of Employment: How Susceptible are Jobs to Computerisation? *Technological Forecasting and Social Change 114*, 254–280.

Furman, J. and R. Seamans (2019). AI and the Economy. *Innovation Policy and the Economy 19*, 161–191.

Gans, J. S. and A. Goldfarb (2026). O-Ring Automation. *NBER Working Paper, No. 34639*.

Garicano, L. (2000). Hierarchies and the Organization of Knowledge in Production. *Journal of Political Economy 108*(5), 874–904.

Garicano, L., J. Li, and Y. Wu (2026). Weak bundle, strong bundle: How AI redraws job boundaries. Technical report, Centre for Economic Policy Research.

Garicano, L. and E. Rossi-Hansberg (2006). The Organization of Work in the Knowledge Economy. *Econometrica 74*(4), 1199–1227.

Hampole, M., D. Papanikolaou, L. D. W. Schmidt, and B. Seegmiller (2025). Artificial Intelligence and the Labor Market. *NBER Working Paper, No. 33509*.

Handa, K., A. Tamkin, M. McCain, S. Huang, E. Durmus, S. Heck, J. Mueller, J. Hong, S. Ritchie, T. Belonax, K. K. Troy, D. Amodei, J. Kaplan, J. Clark, and D. Ganguli (2025). Which Economic Tasks are Performed with AI? Evidence from Millions of Claude Conversations. *arXiv preprint arXiv:2503.04761*.

Horton, J. and R. Horton (2024). EDSL: Expected Parrot Domain Specific Language for AI Powered Social Science. *Expected Parrot Working Paper*.

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

Ide, E. and E. Talamàs (2025). Artificial Intelligence in the Knowledge Economy. *Journal*

of Political Economy 133(12), 3762–3800.
Kim, H., D. Kim, and R. Koning (2026). Mapping AI into Production: A Field Experiment on Firm Performance. *INSEAD Working Paper, No. 2026/20/STR*.
Kremer, M. (1993). The O-ring Theory of Economic Development. *The Quarterly Journal of Economics 108*(3), 551–575.
Lee, D. T. and C. A. Makridis (2023). A Task-Interdependency Model of Complex Collaboration Towards Human-Centered Crowd Work. *arXiv preprint arXiv:2309.00160*.
Levhari, D. (1968). A Note on Houthakker's Aggregate Production Function in a Multifirm Industry. *Econometrica*, 151–154.
Mannhardt, F. (2016). Sepsis Cases – Event Log. https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460. 4TU.ResearchData.
McElheran, K., J. F. Li, E. Brynjolfsson, Z. Kroff, E. Dinlersoz, L. Foster, and N. Zolas (2024). AI Adoption in America: Who, What, and Where. *Journal of Economics & Management Strategy 33*(2), 375–415.
McElheran, K., M.-J. Yang, Z. Kroff, and E. Brynjolfsson (2025). The Rise of Industrial AI in America: Microfoundations of the Productivity J-Curve(s). *Working Paper*.
Murphy, K. M. (1986). *Specialization and Human Capital*. Ph. D. thesis, The University of Chicago.
National Center for O*NET Development (2023). O*NET 27.3 Database. https://www.onetcenter.org/database.html. U.S. Department of Labor, Employment and Training Administration.
Noy, S. and W. Zhang (2023). Experimental Evidence on the Productivity Effects of Generative Artificial Intelligence. *Science 381*(6654), 187–192.
Tambe, P. and L. M. Hitt (2012). The Productivity of Information Technology Investments: New Evidence from IT Labor Data. *Information Systems Research 23*(3), 599–617.
Tomlinson, K., S. Jaffe, W. Wang, S. Counts, and S. Suri (2025). Working with AI: Measuring the Applicability of Generative AI to Occupations. *arXiv preprint arXiv:2507.07935*.
Trammell, P. (2026). Workflows and Automation. *Digital Economy Lab, Stanford University Working Paper*.
van Dongen, B. (2017). BPI Challenge 2017. https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b. 4TU.ResearchData. Loan application handling at a Dutch financial institution.
van Dongen, B. (2020). BPI Challenge 2020. https://doi.org/10.4121/uuid:52fb97d4-4588-43c9-9d04-3604d4613b51. 4TU.ResearchData. Collection of five university travel and reimbursement event logs.
Webb, M. (2020). The Impact of Artificial Intelligence on the Labor Market. *Available at SSRN 3482150*.
Williamson, O. E. (1971). The Vertical Integration of Production: Market Failure Considerations. *American Economic Review 61*(2), 112–123.
Williamson, O. E. (1979). Transaction-Cost Economics: The Governance of Contractual Relations. *The Journal of Law and Economics 22*(2), 233–261.

Chaining Tasks, Redefining Work: A Theory of AI Automation

Mert Demirer, John J. Horton, Nicole Immorlica, Brendan Lucier, and Peyman Shahidi

Online Appendix

A Additional Tables and Figures

Table A.1: Notation Used in the Theory Sections

Panel A: Steps, Tasks, and AI Deployment (Section 3: Short Run)	
Symbol	**Description**
$\alpha \in (0, 1]$	Quality of the general-purpose AI technology
s_i	Production step i
$\mathcal{S} = (s_1, \ldots, s_m)$	Sequence of the firm's m production steps
t_i^M	Time cost of completing step i manually
t_i^A	Time cost of verifying the output of one AI attempt at step i
d_i	Difficulty of step i for AI
$q_i = \alpha^{d_i}$	Probability that AI completes step i successfully
T_b	Task b: a manual step or an AI chain (with a standalone augmented step considered a length-one AI chain)
$\mathcal{T} = (T_1, \ldots, T_n)$	AI deployment strategy; a partition of the steps into n tasks
t_b	Expected time cost of task b: t_i^M if a manual step, $t_r^A / \prod_{i=\ell}^{r} q_i$ if an AI chain over steps $\ell, \ldots, r$
Panel B: Skills, Jobs, Wages, Hand-offs, and Job Design (Section 5: Long Run)	
Symbol	**Description**
c_i^M	Skill required to complete step i manually
c_i^A	Skill required to verify the output of one AI attempt at step i
c_b	Skill required by task b: c_i^M if a manual step, c_r^A if an AI chain augmented at step r; an automated step requires none
J_j	Job j: a contiguous block of tasks assigned to a single worker
$\mathcal{J} = (J_1, \ldots, J_p)$	Job design; a partition of the tasks into p jobs
t_i^H	Hand-off time incurred when a worker's final step is s_i
$t^H(J_j)$	Hand-off time of job J_j, equal to t_i^H for its final step s_i
Wage_j	Wage rate of job J_j's worker, equal to the total skill it requires, $\sum_{T_b \in J_j} c_b$
WageBill_j	Per-unit-output labor cost of job J_j: $\left(\sum_{T_b \in J_j} c_b\right)\left(t^H(J_j) + \sum_{T_b \in J_j} t_b\right)$

Notes: Steps are indexed by i, tasks by b, and jobs by j; superscripts M and A denote the manual and AI modes of executing a step, and H a hand-off between consecutive workers. The long-run problem uses both panels.

Table A.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	$\sum c_b$	$t^H + \sum t_b$	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 workflow with a fixed automation strategy and $n = 3$ tasks. The four job designs are drawn in Figure 8, and the cost parameters of the tasks are given in Section 5.4. A design's total is the sum over its jobs of the job's total skill times the total time it takes, hand-off included, which is Equation (7) applied job by job. In Panel (a), 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, in Panel (b) 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.

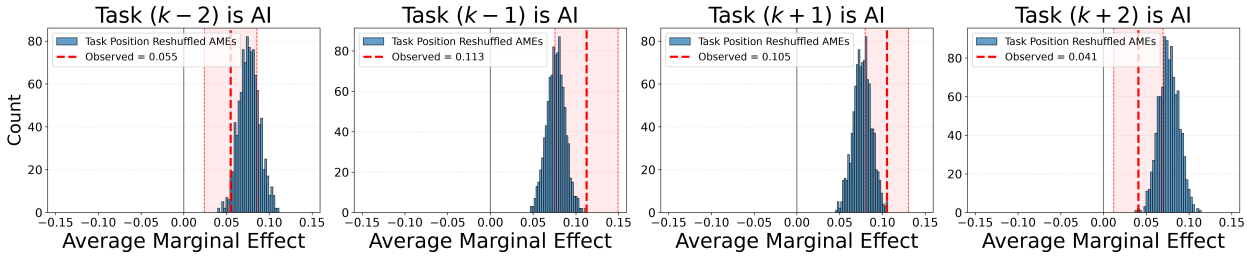
Table A.3: Configurations, Costs, and Marginal Gains in Example 2

Configuration	Time cost (C)	$-\mathrm{d}C/\mathrm{d}\alpha$	Optimal for
Both steps manual	$t_1^M + t_2^M = 14$	0	$\alpha \leq 0.50$
Step 1 augmented	$t_1^A/q_1 + t_2^M = 3.5\alpha^{-11} + 8$	$38.5/\alpha^{12}$	never
Step 2 augmented	$t_1^M + t_2^A/q_2 = 6 + 4/\alpha$	$4/\alpha^2$	$0.50 \leq \alpha \lesssim 0.92$
Both steps augmented	$t_1^A/q_1 + t_2^A/q_2 = 3.5\alpha^{-11} + 4/\alpha$	$38.5/\alpha^{12} + 4/\alpha^2$	never
Steps 1–2 chained	$t_2^A/(q_1 q_2) = 4\alpha^{-12}$	$48/\alpha^{13}$	$\alpha \gtrsim 0.92$

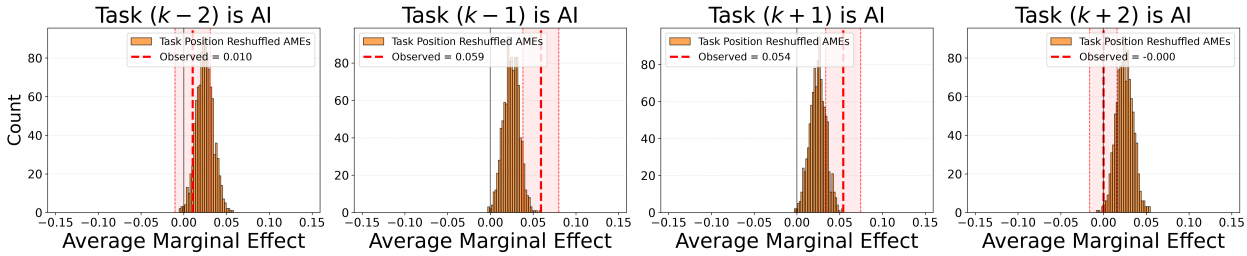
Notes: The two-step workflow of Example 2, the AI-hard and AI-easy steps of Section 4.1, in which $t_1^M = 6$, $t_2^M = 8$, $t_1^A = 3.5$, $t_2^A = 4$, and the AI succeeds at step i with probability $q_i = \alpha^{d_i}$ for $d_1 = 11$ and $d_2 = 1$, so that $q_1 = \alpha^{11}$ and $q_2 = \alpha$. Costs follow from the task time costs of Section 3.2, and $-\mathrm{d}C/\mathrm{d}\alpha$ is the marginal benefit of improving AI quality. The last column gives the range of α over which each configuration is cost-minimizing. Appendix B.3 states and proves the general result the example illustrates. Neither configuration in which step 1 is augmented ever minimizes cost. Augmenting it on its own costs $3.5\alpha^{-11}$, which is prohibitive while AI is unreliable, and by the time AI is reliable enough for that to be affordable, folding step 1 into a chain is cheaper still, because an automated step is never prompted or verified on its own, so its management cost is never paid.

Figure A.1: Neighboring Tasks' AI Execution Status and a Task's AI Execution: Placebo Distributions

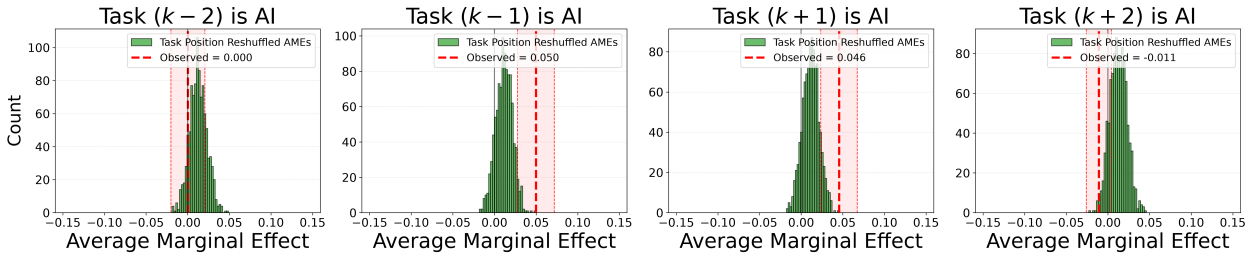
(a) No Fixed Effects



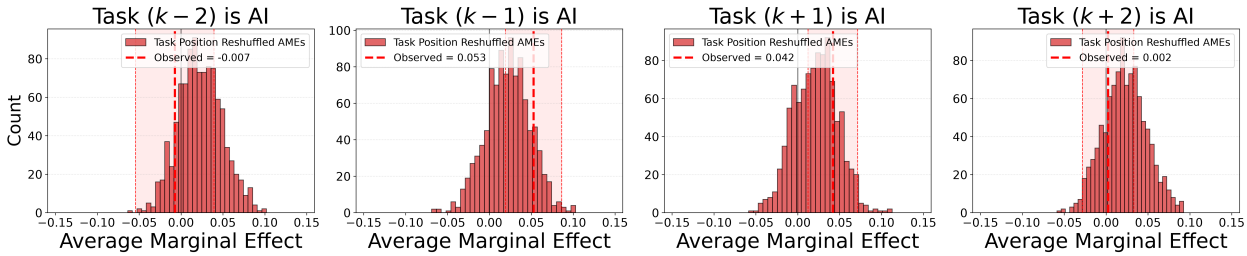
(b) SOC Major Group Fixed Effects



(c) SOC Minor Group Fixed Effects



(d) Detailed Work Activity Fixed Effects



Notes: These graphs show that, among similar tasks appearing in multiple occupations, those sitting in a more AI-executed local context exhibit higher probabilities of being executed by AI, with the association concentrated on the steps immediately adjacent to the task. The red dashed line in each graph indicates the observed average marginal effect of the corresponding variable in the original dataset reported in Table 2. The red shaded area marks the 90% confidence interval around the observed point estimates in the O*NET sample. The histograms plot the distribution of average marginal effects across 1,000 randomized reshuffles of task positions within occupations. Panel (a) reports results from Equation (12), while Panels (b), (c), and (d) augment the baseline specification with SOC major group, SOC minor group, and DWA fixed effects, respectively.

B Omitted Proofs

This appendix collects the proofs omitted from the main text, in the order the results appear there.

B.1 Proof of Proposition 1

Throughout, write $t_i^* = \min\left\{t_i^M, \frac{t_i^A}{q_i}\right\}$ for the cheapest way of executing step i on its own, whether manually or as a standalone augmented step, and take $q_{k-1}, q_k, q_{k+1} \in (0,1)$. Because chains are contiguous and none crosses the boundary of the block $\{k-1,\ k,\ k+1\}$, an arrangement of the block is a partition of it into consecutive runs, each run executed either as a single step at cost t_i^* or as an AI chain augmented at its last step r, at cost $t_r^A / \prod_i q_i$ taken over the run's steps.

Before comparing them, note which of these arrangements can have AI execute step k at all. Executed as a singleton, step k costs $t_k^* = t_k^M$, since $t_k^M < t_k^A/q_k$ by hypothesis, so augmenting it on its own is strictly dominated by performing it manually and never occurs in an optimum. AI therefore executes step k only if step k belongs to a chain of length at least two, and since chains are contiguous and confined to the block, that chain is $\{k,\ k+1\}$, $\{k-1,\ k,\ k+1\}$, or $\{k-1,\ k\}$. In the first two, step k is not the chain's last step, so it is automated, while in the third it is the last step, hence augmented.

Pricing a singleton $\{k\}$ at t_k^M accordingly, the four partitions cost

$$
\begin{aligned}
V_0 &= t_{k-1}^* + t_k^M + t_{k+1}^*, && \text{(no chain covers step } k\text{)} \\
V_1 &= t_{k-1}^* + \frac{t_{k+1}^A}{q_k\, q_{k+1}}, && \text{(chain } \{k,\ k+1\}\text{)} \\
V_2 &= \frac{t_{k+1}^A}{q_{k-1}\, q_k\, q_{k+1}}, && \text{(chain } \{k-1,\ k,\ k+1\}\text{)} \\
V_3 &= \frac{t_k^A}{q_{k-1}\, q_k} + t_{k+1}^*. && \text{(chain } \{k-1,\ k\}\text{)}
\end{aligned}
$$

AI executes step k exactly when $\min\{V_1, V_2, V_3\} < V_0$, ties going to manual execution.

Proof of part (i). Fix (t_k^M, t_k^A, q_k) with $t_k^M < t_k^A/q_k$ and fix $q_{k-1}, q_{k+1} \in (0,1)$. Restrict attention to the two-parameter family of neighbor costs given by $t_{k-1}^M = t_{k-1}^A = \mu > 0$ and $t_{k+1}^M = 1$, $t_{k+1}^A = \tau > 0$, so that $t_{k-1}^* = \mu$ and $t_{k+1}^* = \min\{1, \tau/q_{k+1}\}$. Since $V_0, \ldots, V_3$ are continuous in all of the neighbor parameters, it suffices to exhibit, for each configuration, values of (μ, τ) at which its cost is strictly below the other three; strictness then extends the conclusion to an open neighborhood.

- *Configuration* $\{k,\ k+1\}$*, which automates step* k*.* Set $\tau = q_k\, q_{k+1} t_k^M/2$, so that $V_1 = \mu + t_k^M/2$, and take

$$0 < \mu < \min\left\{\frac{t_k^M}{2}\cdot\frac{1-q_{k-1}}{q_{k-1}},\ \frac{t_k^A}{q_{k-1}\,q_k} - \frac{t_k^M}{2}\right\},$$

a nonempty interval because $q_{k-1} < 1$ and $t_k^A/(q_{k-1}q_k) > t_k^A/q_k > t_k^M$. The three comparisons are then

$$V_0 - V_1 = \frac{t_k^M}{2} + t_{k+1}^*, \quad V_2 - V_1 = \frac{t_k^M}{2}\cdot\frac{1-q_{k-1}}{q_{k-1}} - \mu, \quad V_3 - V_1 = \frac{t_k^A}{q_{k-1}\,q_k} - \frac{t_k^M}{2} - \mu + t_{k+1}^*,$$

and all three are positive: the first because $t_{k+1}^* > 0$, and the second and third by the first and second bounds on μ respectively.

- *Configuration* $\{k-1,\ k,\ k+1\}$*, which also automates step* k*.* Set $\mu = 1$ and take $0 < \tau < q_{k+1}\min\{q_{k-1}\,q_k,\ t_k^A\}$, so that the long chain is cheap. The three comparisons are

$$\begin{aligned} V_0 - V_2 &= 1 + t_k^M + t_{k+1}^* - \frac{\tau}{q_{k-1}\,q_k\,q_{k+1}},\\ V_1 - V_2 &= 1 + \frac{\tau}{q_k\,q_{k+1}} - \frac{\tau}{q_{k-1}\,q_k\,q_{k+1}},\\ V_3 - V_2 &= \frac{t_k^A}{q_{k-1}\,q_k} + t_{k+1}^* - \frac{\tau}{q_{k-1}\,q_k\,q_{k+1}}. \end{aligned}$$

The bound on τ puts $\tau/(q_{k-1}q_kq_{k+1})$ below 1 and $t_k^A/(q_{k-1}q_k)$, so all three are positive.

- *Configuration* $\{k-1,\ k\}$*, which augments step* k*.* Set $\mu = t_k^A/(q_{k-1}q_k)$ and take $\tau > \max\{q_{k+1},\ q_kq_{k+1},\ q_{k-1}q_kq_{k+1}(\mu+1)\}$. The first of these bounds puts τ/q_{k+1} above 1, so that $t_{k+1}^* = 1$ and $V_3 = \mu + 1$, and the comparisons are

$$V_0 - V_3 = t_k^M, \qquad V_1 - V_3 = \frac{\tau}{q_k\,q_{k+1}} - 1, \qquad V_2 - V_3 = \frac{\tau}{q_{k-1}\,q_k\,q_{k+1}} - (\mu+1),$$

positive because $t_k^M > 0$ and by the second and third bounds on τ respectively.

The margin of the human's advantage on step k, $t_k^A/q_k - t_k^M$, was unrestricted throughout, which gives the second claim of part (i). □

Proof of part (ii). Fix the six cost parameters of the three steps and let their success probabilities vary over $Q = (0,1)^3$. Write

$$\mathcal{A} \;=\; \big\{(q_{k-1},\ q_k,\ q_{k+1}) \in Q \;:\; \text{AI executes step } k\big\}$$

for the set of reliabilities at which the cost-minimizing arrangement does not leave step k to the human. Part (ii) is precisely the statement that $\mathcal{A}$ is *upward closed* in the coordinatewise order, meaning that whenever a triple belongs to $\mathcal{A}$, so does every triple at least as large in all three coordinates.

Before the main argument, dispose of the points at which the standing hypothesis lapses. Raising q_k can push t_k^A/q_k below t_k^M, and at any such point step k is cheaper augmented than performed manually even on its own, so AI executes it in every arrangement of the block and the conclusion holds there regardless of the other two coordinates. It is therefore enough to prove the claim when the human still holds the comparative advantage at the larger triple, $t_k^M \le t_k^A/q_k'$. In that case $q_k \le q_k'$ gives $t_k^M \le t_k^A/q_k' \le t_k^A/q_k$, so the advantage is in force at the smaller triple and everywhere between, and the four arrangements priced above describe the block throughout. The proof turns the upward-closure statement into one about the three differences

$$D_j \;=\; V_j - V_0, \qquad j = 1, 2, 3,$$

each of which compares one arrangement that has AI execute step k against the arrangement that does not. Three reductions get us there.

Reduction 1: from $\mathcal{A}$ to one chain at a time. AI executes step k exactly when $\min\{V_1, V_2, V_3\} < V_0$, that is, when $D_j < 0$ for at least one j. Writing $\mathcal{A}_j = \{(q_{k-1}, q_k, q_{k+1}) \in Q : D_j < 0\}$ for the set on which chain j by itself beats manual execution of step k, this says

$$\mathcal{A} \;=\; \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3.$$

A union of upward-closed sets is upward closed, because a triple in the union lies in one of them and every larger triple then lies in that same one. It is therefore enough to show that each $\mathcal{A}_j$ is upward closed, and the three can be handled separately. We need not know which chain is optimal, only whether each alone beats leaving the step to the human.

Reduction 2: from all three coordinates at once to one at a time. Suppose each $\mathcal{A}_j$ is upward closed in each single coordinate, at every fixed value of the other two. Given $(q_{k-1}, q_k, q_{k+1}) \in \mathcal{A}_j$ and a larger triple, raise the coordinates in turn:

$$(q_{k-1}, q_k, q_{k+1}) \implies (q_{k-1}', q_k, q_{k+1}) \implies (q_{k-1}', q_k', q_{k+1}) \implies (q_{k-1}', q_k', q_{k+1}'),$$

each step by closure in the coordinate being raised, holding the other two at the values they have reached. The one-coordinate statements deliver the joint one, provided each holds at every value of the coordinates being held fixed, as they are proved below.

Reduction 3: from upward closure to a property of D_j. With the other two coordinates fixed,

D_j is a function of a single success probability q, and $\mathcal{A}_j$ is the set on which that function is negative. That set is upward closed as soon as D_j, once negative, stays negative as q rises. It suffices for D_j to be nonincreasing in q, which is what we verify in every case but one; in the exceptional case D_j increases, and we show instead that it is negative on the entire range where the increase can occur.

Writing the three differences out,

$$D_1 = \frac{t^A_{k+1}}{q_k\, q_{k+1}} - t^M_k - t^*_{k+1}, \quad D_2 = \frac{t^A_{k+1}}{q_{k-1}\, q_k\, q_{k+1}} - t^*_{k-1} - t^M_k - t^*_{k+1}, \quad D_3 = \frac{t^A_k}{q_{k-1}\, q_k} - t^*_{k-1} - t^M_k,$$

the terms that move with the neighbor reliabilities are the chain costs and the two standalone costs

$$t^*_{k-1} = \min\left\{t^M_{k-1},\, \frac{t^A_{k-1}}{q_{k-1}}\right\}, \qquad t^*_{k+1} = \min\left\{t^M_{k+1},\, \frac{t^A_{k+1}}{q_{k+1}}\right\},$$

of which the first depends on q_{k-1} alone and the second on q_{k+1} alone.

A claim covering both coordinates. Let $m,\ A,\ B,\ c > 0$ and define, for $q \in (0,1)$,

$$D(q) \;=\; \underbrace{\frac{A}{q}}_{\text{a chain through the neighbor}} \;-\; \underbrace{\min\left\{m,\, \frac{B}{q}\right\}}_{\text{the neighbor on its own}} \;-\; \underbrace{c}_{\text{fixed}}.$$

Then $\{q \in (0,1) : D(q) < 0\}$ is upward closed. To see this, note that the neighbor's standalone cost is the smaller of a manual cost m that does not move with q and an augmented cost B/q that falls as q rises, the two crossing at $q^* = B/m$. Splitting there,

$$D(q) \;=\; \begin{cases} \dfrac{A}{q} - m - c, & 0 < q \le q^*, \\[2ex] \dfrac{A - B}{q} - c, & q^* < q < 1, \end{cases}$$

the second line obtained by substituting B/q for the minimum. The two expressions agree at $q = q^*$, where $m = B/q^*$, so D is continuous; and if $q^* \ge 1$ the second branch is empty and only the first arises. On the first branch D is strictly decreasing, since

$$\frac{\mathrm{d}}{\mathrm{d}q}\left(\frac{A}{q} - m - c\right) \;=\; -\frac{A}{q^2} \;<\; 0,$$

so raising q cheapens the chain while leaving untouched the manual cost that the chain displaces. On the second branch the behavior depends on the sign of $A - B$:

- if $A \ge B$, then $(A - B)/q$ is nonincreasing in q, so D is nonincreasing on the branch;

- if $A < B$, then D increases with q, but $(A - B)/q < 0$ and $-c < 0$, so $D(q) < 0$ at *every* point of the branch.

Either way, if D is negative at some point of the second branch it is negative everywhere to the right of it. Now take $q < q'$ in $(0, 1)$ with $D(q) < 0$, and consider the three possible positions of the pair relative to q^*:

- if $q' \le q^*$, both lie on the first branch and $D(q') < D(q) < 0$;
- if $q^* < q$, both lie on the second branch, which was just handled;
- if $q \le q^* < q'$, then $D(q^*) \le D(q) < 0$ by the first branch, and since the second-branch expression agrees with D at q^* and the argument above applies to it on $[q^*, 1)$, we get $D(q') < 0$.

This establishes the claim. Its content is that raising a neighbor's reliability always cheapens a chain that runs through that neighbor, and cheapens the neighbor's own execution only once that execution is an augmented one, in which case both costs carry the same factor $1/q$ and fall in the same proportion, so the comparison between them cannot reverse.

Closure in q_{k-1}. Fix q_k and q_{k+1} and let $q = q_{k-1}$ vary. The difference D_1 does not depend on q at all, because the two arrangements it compares, the chain $\{k, k+1\}$ and manual execution of step k, both leave the predecessor to be executed on its own, so the term t^*_{k-1} appears in V_1 and V_0 alike and cancels. A constant function is nonincreasing, so $\mathcal{A}_1$ is upward closed in this coordinate. The other two differences are instances of the claim, with $B = t^A_{k-1}$ and $m = t^M_{k-1}$ throughout and

$$D_2 : \; A = \frac{t^A_{k+1}}{q_k\, q_{k+1}},\; c = t^M_k + t^*_{k+1}, \qquad D_3 : \; A = \frac{t^A_k}{q_k},\; c = t^M_k,$$

neither of which moves with q. The claim therefore makes $\mathcal{A}_2$ and $\mathcal{A}_3$ upward closed in q_{k-1}. Nothing here ties A to B, so both of the claim's second-branch cases can occur. A predecessor whose own augmented cost falls faster than the cost of the chain it would anchor is exactly the case $A < B$, and it is settled not by monotonicity but by the chain's being cheaper throughout that branch to begin with.

Closure in q_{k+1}. The successor coordinate is the counterpart of the predecessor one, and the milder half of it, so the claim disposes of it directly. The difference D_3 now plays the role D_1 played above and drops out for the mirror-image reason, namely that the chain $\{k-1, k\}$ and manual execution of step k both leave the successor to be executed on its own, so t^*_{k+1} cancels between V_3 and V_0. The remaining two differences are again instances

of the claim, read in $q = q_{k+1}$, with $B = t^A_{k+1}$ and $m = t^M_{k+1}$ throughout and

$$D_1 : \; A = \frac{t^A_{k+1}}{q_k}, \; c = t^M_k, \qquad D_2 : \; A = \frac{t^A_{k+1}}{q_{k-1}\, q_k}, \; c = t^*_{k-1} + t^M_k.$$

The one difference from the predecessor case is that A and B are no longer free of each other. A chain terminating at the successor pays that same verification cost t^A_{k+1}, inflated by the failure risk of the steps ahead of it, so A/B equals $1/q_k$ or $1/(q_{k-1}q_k)$ and is at least 1. Only the case $A \geq B$ of the claim can arise, and $\mathcal{A}_1$ and $\mathcal{A}_2$ are upward closed in q_{k+1}, with the differences nonincreasing outright.

Closure in q_k. This coordinate needs no work beyond the pricing already in hand. With the human's advantage in force, the no-chain arrangement performs step k manually and prices the two neighbors on their own, so V_0 does not involve q_k at all, while each of V_1, V_2, and V_3 carries a factor $1/q_k$ in the cost of the chain covering step k and is therefore strictly decreasing in it. Every D_j is thus strictly decreasing in q_k, and each $\mathcal{A}_j$ is upward closed in that coordinate. This is the sense in which the step's own reliability and its neighbors' enter differently, in that a better neighbor cheapens the chain and the alternative to it at once, and the argument above is what settles the race between them, whereas a better step k cheapens only the chain.

Each $\mathcal{A}_j$ is therefore upward closed in each coordinate separately, hence upward closed in the coordinatewise order by Reduction 2, and $\mathcal{A}$ is upward closed by Reduction 1. □

B.2 Proof of Proposition 2

Throughout we maintain the normalization $t^A_i = 1$ for all s_i and use the notation of Section 4.2: F for the set of steps that fail, $\mathcal{C}$ for the maximal connected components of non-failed steps, $\omega(C)$ for their weights, and (3) for the realized fragmentation whose expectation is the fragmentation index. We additionally write $t^*_i = \min\left\{t^M_i, \frac{1}{q_i}\right\}$ for the minimum cost of implementing step s_i on its own, whether manually or through AI augmentation. For a fixed sequence $\mathcal{S} = (s_1, \ldots, s_m)$ we write FI for its fragmentation index and OPT for the cost of the optimal AI deployment strategy for its steps.

Both bounds compare FI against the cost of a *particular* AI deployment strategy rather than against OPT directly, since OPT is itself a minimum over exponentially many arrangements; the two directions differ in which strategy they use. For the upper bound we bound the realized fragmentation task by task against an arbitrary strategy and then evaluate that bound at the cost-minimizing one, so the quantity being compared to is exactly OPT. Manual tasks contribute the same amount to both sides, and the entire loss is incurred on AI chains, where it is a factor of $\frac{5}{4}$. For the lower bound we instead construct

a greedy strategy that extends a chain for as long as its success probability stays at least 1/2; because that strategy is feasible its cost ALG is at least OPT, so it suffices to bound FI below by a fraction of ALG. A charging argument supplies that fraction, assigning each of the greedy strategy's tasks a share of the realized fragmentation worth at least a quarter of what the task costs, and Lemma B.3 repeats the argument without the assumption $t_i^M \geq 1$ at the cost of a further factor of two. Proposition 2 then combines the three lemmas.

We begin with the upper bound.

Lemma B.1. $FI \leq \frac{5}{4} OPT$.

Proof. Fix an arbitrary task sequence $\mathcal{T} = (T_1, \ldots, T_n)$. Note that FI depends only on the step sequence $\mathcal{S}$. Every step carries its own success probability q_i whatever $\mathcal{T}$ does with it, so the realization of F is unaffected by $\mathcal{T}$. We use $\mathcal{T}$ only to organize the terms of FI, grouping steps so that they can be compared with the cost $\mathcal{T}$ incurs on them. Partition this into two sets of tasks: $\mathcal{T}_M$ containing all (singleton) tasks that are completed manually, and $\mathcal{T}_A$ containing AI chains. We first claim that

$$FI \leq \sum_{(s_i) \in \mathcal{T}_M} t_i^M + \sum_{T_b \in \mathcal{T}_A} \left(1 + \sum_{s_i \in T_b} (1 - \alpha^{d_i})(1 + t_i^*) \right).$$

To see why, first consider a singleton $(s_i) \in \mathcal{T}_M$, which $\mathcal{T}$ executes manually at cost t_i^M. If s_i lies in F, its own term in the realized fragmentation is $t_i^* \leq t_i^M$; otherwise it lies in some connected component C and raises that component's weight $\omega(C)$ by at most t_i^M. Either way it accounts for at most t_i^M of the realized fragmentation, matching what it contributes to the cost of $\mathcal{T}$.

Next consider the AI chains. For any $T_b \in \mathcal{T}_A$, consider the set of steps in T_b that fail in any given realization of F. If no steps in T_b fail, then T_b collectively contributes at most 1 to the realized fragmentation (since all $s_i \in T_b$ lie in the same connected component of non-failed steps). Otherwise, each failed step $s_i \in T_b$ contributes t_i^* to the realized fragmentation for itself plus an additional value of at most 1 for the weight of a potential new connected component of non-failed steps. As each step $s_i \in T_b$ fails independently with probability $(1 - \alpha^{d_i})$, we get that the contribution of the steps in T_b to the fragmentation index is at most $1 + \sum_{s_i \in T_b} (1 - \alpha^{d_i})(1 + t_i^*)$ as claimed.

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

$$OPT = \sum_{(s_i) \in \mathcal{T}_M} t_i^M + \sum_{T_b \in \mathcal{T}_A} \alpha^{-d(T_b)}$$

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

Comparing our expression for OPT with our bound on FI, we see that each task in $\mathcal{T}_M$ contributes the same amount to each, whereas each $T_b \in \mathcal{T}_A$ contributes $1 + \sum_{s_i \in T_b}(1 - \alpha^{d_i})(1 + t_i^*)$ to the latter and $\alpha^{-d(T_b)}$ to the former. We will show that, for each $T_b \in \mathcal{T}_A$, $1 + \sum_{s_i \in T_b}(1 - \alpha^{d_i})(1 + t_i^*) \le \frac{5}{4}\alpha^{-d(T_b)}$, which will complete the proof.

First, since $t_i^* \le \alpha^{-d_i}$ for each $s_i \in \mathcal{S}$, we have $1 + \sum_{s_i \in T_b}(1 - \alpha^{d_i})(1 + t_i^*) \le 1 + \sum_{s_i \in T_b}(1 - \alpha^{d_i})(1 + \alpha^{-d_i})$.

Next note that for any $x, y \in [0, 1]$, $(1 - x)(1 + \frac{1}{x}) + (1 - y)(1 + \frac{1}{y}) \le (1 - xy)(1 + \frac{1}{xy})$. Repeatedly applying this fact, we get that $1 + \sum_{s_i \in T_b}(1 - \alpha^{d_i})(1 + \alpha^{-d_i}) \le 1 + (1 - \alpha^{d(T_b)})(1 + \alpha^{-d(T_b)}) = 1 + \alpha^{-d(T_b)} - \alpha^{d(T_b)}$ for any $T_b \in \mathcal{T}_A$. The ratio between this expression and $\alpha^{-d(T_b)}$ is maximized at $\alpha^{d(T_b)} = 1/2$ (equivalently $\alpha^{-d(T_b)} = 2$), achieving a value of $5/4$ as claimed. □

The bound of $\frac{5}{4}$ in Lemma B.1 is tight, as the following example shows.

Example B.1. Consider a three-step workflow in which the first and last steps always succeed, while the second succeeds with probability $1/2$ and is twice as costly to perform manually:

$$\left(t_i^M,\, t_i^A,\, q_i\right)_{i=1,2,3} = (1,\, 1,\, 1), \quad (2,\, 1,\, \tfrac{1}{2}), \quad (1,\, 1,\, 1).$$

The optimal policy chains all three together for an expected cost of $t_3^A/(q_1 q_2 q_3) = 1/(1 \cdot \frac{1}{2} \cdot 1) = 2$, whereas the fragmentation index is $\frac{1}{2} \times 1 + \frac{1}{2} \times (1 + 2 + 1) = \frac{5}{2}$. The ratio FI/OPT is therefore $\frac{5}{4}$.

We next argue that FI is not much smaller than OPT. We begin with an analysis under the assumption that $t_i^M \ge 1$ for all $s_i \in \mathcal{S}$. That is, for each step s_i, manual completion costs at least as much as verifying one AI attempt at it (with guaranteed success).

Lemma B.2. *Suppose $t_i^M \ge 1$ for all $s_i \in \mathcal{S}$. Then $FI \ge \frac{1}{4} OPT$.*

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 greedy gives up on chaining and executes s_1 on its own, either manually or augmented, whichever is cheaper. We call the resulting singleton task an *individual* task. Otherwise, the sequence $T = (s_1, \ldots, s_\ell)$ is added to the task sequence as an AI chain, and we call it a *non-individual* task. Note that a non-individual task may itself be a singleton, namely when $\ell = 1$; what separates the two labels is not the number of steps a task holds

but whether its run passed the 1/2 test. This procedure 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 $\mathcal{T}_M \subseteq \mathcal{T}_I$ (recalling that $\mathcal{T}_M$ 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 step s_m from $\mathcal{S}$ is contained in some $T_b \in \mathcal{T}_{NI}$, we say that T_b 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 $T_b \in \mathcal{T}_{NI}$, we'll write $\overline{T}_b$ to denote T_b plus the first step that follows T_b. For example, if $T_b = (s_i, \ldots, s_\ell)$, then $\overline{T}_b = (s_i, \ldots, s_\ell, s_{\ell+1})$. The set $\mathcal{T}_{NI}$ then has the property that for each task $T_b \in \mathcal{T}_{NI}$ we have $\alpha^{d(T_b)} \geq 1/2$, and for each non-terminal $T_b \in \mathcal{T}_{NI}$ we have $\alpha^{(d(\overline{T}_b))} < 1/2$.

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

$$ALG = \sum_{(s_i) \in \mathcal{T}_I} t_i^* + \sum_{T_b \in \mathcal{T}_{NI}} \alpha^{-d(T_b)} \leq \sum_{(s_i) \in \mathcal{T}_I} t_i^* + 2N$$

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

Next we bound FI. Note that the assumption $t_i^M \geq 1$ implies that, in any realization of the connected components of non-failed steps $\mathcal{C}$, $\omega(C) = 1$ for all $C \in \mathcal{C}$. This implies that the realized fragmentation is equal to $\mathbf{1}\{s_m \notin F\}$ plus, for each $s_i \in F$ (i.e., each step s_i that fails), a contribution of t_i^* plus an additional 1 in the event that $i > 1$ and the step 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 $FI \geq N/2 + \frac{1}{2}\sum_{(s_i) \in \mathcal{T}_I} t_i^*$. We show this by charging the realized fragmentation to tasks in $\mathcal{T}_{NI}$ and $\mathcal{T}_I$, so that each $T_b \in \mathcal{T}_{NI}$ is charged at least 1 with probability at least 1/2, and each $(s_i) \in \mathcal{T}_I$ is charged at least t_i^* with probability at least 1/2.

To see this, let E_b denote the event that some step in $\overline{T}_b$ fails, for each non-terminal $T_b \in \mathcal{T}_{NI}$. Note that E_b occurs with probability at least 1/2, since $\alpha^{d(\overline{T}_b)} < 1/2$. Also, if E_b occurs, then either some step $s_i \in T_b$ fails, or no step in T_b fails but the step s_j

immediately following T_b does. In the former case, we will charge the $t_i^* \geq 1$ contribution of s_i's failure to T_b. In the latter case the step immediately preceding s_j is the last step of T_b, which did not fail, so s_j's failure creates a new connected component and we charge the additional contribution of 1 that it carries (as described above) to T_b. Additionally, if there is a terminal T_b in $\mathcal{T}_{NI}$, then we charge the baseline $\mathbf{1}\{s_m \notin F\}$ from the calculation of FI to that terminal T_b. That baseline is charged with probability $q_m = \alpha^{d_m}$, and since $d(T_b) \geq d_m$ we have $q_m \geq \alpha^{d(T_b)} \geq 1/2$, so the terminal task too is charged at least 1 with probability at least $1/2$. Aggregating over all these cases, we have that at least 1 is charged to each task $T_b \in \mathcal{T}_{NI}$ 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 t_i^* portion of its contribution to task (s_i). We conclude that $FI \geq N/2 + \frac{1}{2}\sum_{(s_i)\in\mathcal{T}_I} t_i^*$ as claimed.

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

The approximation factor in Lemma B.2 cannot be improved below $\frac{4\sqrt{2}}{9(\sqrt{2}-1)} \approx 1.52$, although we do not have a matching lower bound of 4.

Example B.2. Consider a sequence of m identical steps:

$$\left(t_i^M,\, t_i^A,\, q_i\right) = \left(\sqrt{2},\, 1,\, 1/\sqrt{2}\right) \text{ for all } i.$$

The optimal task sequence chains the steps three at a time, since a chain of k steps costs $2^{k/2}$, or $2^{k/2}/k$ per step, which is minimized at $k = 3$ with value $\frac{2\sqrt{2}}{3} \approx 0.9428$, below both the manual cost $\sqrt{2}$ and the per-step cost of any other chain length, so $OPT \to \frac{2\sqrt{2}}{3}\, m$. The fragmentation index is such that each step contributes $\sqrt{2}$ if it fails, plus 1 if the preceding step did not fail, for a total contribution of $(1 - 1/\sqrt{2})(\sqrt{2} + 1/\sqrt{2}) = \frac{3}{2}(\sqrt{2} - 1) \approx 0.6213$. As m grows large, the ratio between $\frac{2\sqrt{2}}{3}m$ and $0.5 + m \times 0.6213$ approaches $\frac{4\sqrt{2}}{9(\sqrt{2}-1)} \approx 1.52$.

We can relax the assumption in Lemma B.2 that $t_i^M \geq 1$ for all s_i, but at the cost of a worse approximation factor.

Lemma B.3. *For general t_i^M (i.e., allowing $t_i^M < 1$ for some steps s_i), $FI \geq \frac{1}{8}OPT$.*

Proof. The argument follows the proof of Lemma B.2, 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_b has the property that $\sum_{s_i\in T_b} t_i^M < 1$ (which implies it is cheaper to run all steps of T_b manually than as an AI chain that is guaranteed to succeed), we switch to running the steps of T_b manually in the task sequence. In our analysis, we still consider T_b to be a task in $\mathcal{T}_{NI}$; but its cost is taken to be $\sum_{s_i\in T_b} t_i^M$.

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

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

Then, to bound FI, we claim that

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

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 $T_b \in \mathcal{T}_{NI}$, the event E_b occurs with probability at least $1/2$. When it does, we charge to T_b the contribution t_i^* to FI from any $s_i \in F$ that lies in T_b. Furthermore, when E_b occurs, for each connected component C that intersects T_b we additionally charge the contribution $\omega(C)$ from FI to T_b. Crucially, the contribution from each C can be charged to at most two different tasks T_b in this way: once for the leftmost T_b that it intersects, and once for the rightmost T_b that it intersects. (The reason is that any T_b meeting C other than these two has its successor step inside C as well, so no step of $\overline{T}_b$ fails and E_b does not occur. Only the leftmost such task, which may extend past a failed step to the left of C, and the rightmost, whose successor is the failed step that ends C, can trigger E_b.) Moreover, this charging to T_b adds up to a total value of at least $\min\left\{1, \sum_{s_i \in T_b} t_i^M\right\}$. The terminal task, for which E_b is not defined, is charged separately: either some step of it fails, in which case the charging above applies verbatim, or no step of it fails, in which case $T_b \subseteq C$ for some connected component C and $\omega(C) \geq \min\{1, \sum_{s_i \in T_b} t_i^M\}$. Either way the terminal task is charged at least $\min\{1, \sum_{s_i \in T_b} t_i^M\}$, and with probability 1 rather than $1/2$. Since the charging occurs with probability $1/2$ for each $T_b \in \mathcal{T}_{NI}$, and we are at most double-counting each $\omega(C)$, we conclude that the expected sum of all $\omega(C)$, plus t_i^* for all failed steps s_i that lie in some $T_b \in \mathcal{T}_{NI}$, is at least $1/4$ of $\sum_{T_b \in \mathcal{T}_{NI}} \min\left\{1, \sum_{s_i \in T_b} t_i^M\right\}$.

The charging for $(s_i) \in \mathcal{T}_I$ remains unchanged. The value t_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 Lemma B.3 is not tight; the following example shows only that it cannot be improved below $\frac{2}{3}(2 + \sqrt{2}) \approx 2.276$. We do not know whether the assumption $t_i^M \geq 1$ is needed for the sharper factor of 4 in Lemma B.2, as we have not found an instance violating it for which OPT exceeds $4\,FI$.

Example B.3. Consider a sequence that alternates between two step types, with $K > 1$

such pairs (so $2K$ steps in total), where ϵ is vanishingly small:

$$\left(t_i^M,\, t_i^A,\, q_i\right) \;=\; \left(0,\, 1,\, 1/\sqrt{2} - \epsilon\right) \;\text{ for odd } i, \qquad (1,\, 1,\, 1) \;\text{ for even } i.$$

The optimum performs manually, at zero cost, the odd steps that fall between blocks, and chains blocks of five consecutive steps that begin and end on an even step. Such a chain succeeds with probability $(1/\sqrt{2})^2 = 1/2$ and so costs 2, while covering three even steps that would otherwise cost 1 each, giving a total cost of $\frac{2}{3}K$ (ignoring ϵ). The fragmentation index is 1 plus 1 for each of the remaining $K-1$ odd steps that fails, which is $1 + (K-1)(1 - 1/\sqrt{2})$ (again ignoring the impact of ϵ). As K grows large, the ratio tends to $\frac{2}{3}(2+\sqrt{2}) \approx 2.276$.

Proof of Proposition 2. Combining Lemmas B.1, B.2, and B.3 yields Proposition 2. In general $\frac{1}{8}OPT \le FI \le \frac{5}{4}OPT$, and under the additional assumption $t_i^M \ge 1$ for all i, $\frac{1}{4}OPT \le FI \le \frac{5}{4}OPT$. □

B.3 Non-Monotone Gains from AI Improvements: Formal Statement and Proof

This appendix states and proves the result that Example 2 of Section 4.3 illustrates; Table A.3 in Appendix A reports that example's configurations and costs.

To make the argument of Section 4.3 precise, consider a workflow with any number of steps and chains. For any AI strategy $\mathcal{T}$, total cost is a fixed manual term plus a polynomial in $1/\alpha$,

$$C_{\mathcal{T}}(\alpha) \;=\; K_{\mathcal{T}} \;+\; \sum_{c \in \mathcal{K}(\mathcal{T})} t_{r_c}^A \, \alpha^{-D_c}, \tag{B.1}$$

where $K_{\mathcal{T}} = \sum_{\text{manual } i} t_i^M$ is the fixed cost of manually-executed steps and the sum runs over the AI chains of $\mathcal{T}$, denoted $\mathcal{K}(\mathcal{T})$, with each chain c (with augmented endpoint r_c) contributing a term of degree equal to its total difficulty $D_c = \sum_{i \in c} d_i \ge 1$. The cost of $\mathcal{T}$ thus shifts in a continuous and smooth manner as α increases, with monotone marginal gains to technology improvements.

The firm optimizes costs over many possible AI strategies, expressed as the deployment problem in (1). At each level of AI quality α the firm adopts the cheapest available strategy, so its optimal cost is the lower envelope of the strategy-specific cost curves, $C^*(\alpha) = \min_{\mathcal{T}} C_{\mathcal{T}}(\alpha)$. We write $g_{\mathcal{T}}(\alpha) = -\,\mathrm{d}C_{\mathcal{T}}/\mathrm{d}\alpha$ for the marginal benefit of improving AI quality under a fixed strategy $\mathcal{T}$, and $g^*(\alpha) = -\,\mathrm{d}C^*/\mathrm{d}\alpha \ge 0$ for the firm's marginal benefit at the optimum; the function $g^*(\alpha)$ equals $g_{\mathcal{T}}(\alpha)$ on the regime of α values where $\mathcal{T}$ is optimal. As α rises, the optimal strategy changes at finitely many values of α, the reorganization thresholds of Section 4.3, which separate one regime from the next.

Write $\phi = C_{\mathcal{T}'} - C_{\mathcal{T}}$ for the cost difference between the strategies that are optimal just above and just below a threshold α_0, and call the threshold *transversal* if the two curves cross there non-tangentially, that is, if $\phi'(\alpha_0) \neq 0$. Lemma B.4 shows that g^* diminishes within each regime and never falls at a reorganization threshold, and that it jumps upward at every transversal threshold, so the returns to improving AI quality are non-monotone.

Lemma B.4 (Non-monotone Marginal Returns to AI Quality Improvements)**.** *Let $C^*(\alpha) = \min_{\mathcal{T}} C_{\mathcal{T}}(\alpha)$ be the firm's optimal cost over all feasible AI strategies and $g^*(\alpha) = -\mathrm{d}C^*/\mathrm{d}\alpha$ its marginal benefit of improving AI quality. Within each regime (an interval of α on which a single AI strategy is optimal), g^* is non-increasing. At every reorganization threshold (where the optimal strategy changes), g^* never falls, and at every transversal threshold it jumps up discontinuously.*

Proof. Each $C_{\mathcal{T}}$ is a constant plus terms $t^A_{r_c}\,\alpha^{-D_c}$ with $D_c \geq 1$, with derivatives

$$C'_{\mathcal{T}}(\alpha) = -\sum_{c\in\mathcal{K}(\mathcal{T})} t^A_{r_c} D_c\,\alpha^{-D_c-1} \;\leq\; 0,$$
$$C''_{\mathcal{T}}(\alpha) = \sum_{c\in\mathcal{K}(\mathcal{T})} t^A_{r_c} D_c(D_c+1)\,\alpha^{-D_c-2} \;\geq\; 0,$$

so each $C_{\mathcal{T}}$ is non-increasing and convex.

Within a regime where $\mathcal{T}$ is optimal, $g^*(\alpha) = g_{\mathcal{T}}(\alpha) = -C'_{\mathcal{T}}(\alpha) \geq 0$, and the convexity of $C_{\mathcal{T}}$ makes it non-increasing in α (when $\mathcal{T}$ uses no chain $g^*(\alpha) = 0$ but is otherwise strictly decreasing). The returns to a fixed configuration in a regime thus weakly diminish. This proves the first claim of the lemma.

Next, let $\alpha_0 \in (0,1)$ be a threshold where the optimum switches from $\mathcal{T}$ (optimal just below α_0) to $\mathcal{T}'$ (just above α_0).[29] Define

$$\phi(\alpha) = C_{\mathcal{T}'}(\alpha) - C_{\mathcal{T}}(\alpha).$$

The function $\phi(\alpha)$ is the difference of two finite sums of powers of α and hence continuous and differentiable on $(0,1)$. The two costs are equal at α_0, with $C_{\mathcal{T}'}$ the cheaper just above and $C_{\mathcal{T}}$ the cheaper just below. Therefore, by definition ϕ satisfies:

$$\phi(\alpha_0^-) \geq 0, \qquad \phi(\alpha_0) = 0, \qquad \phi(\alpha_0^+) \leq 0.$$

A differentiable function with such a sign change has nonpositive derivative at α_0. So $\phi'(\alpha_0) \leq 0$, or equivalently, $C'_{\mathcal{T}'}(\alpha_0) \leq C'_{\mathcal{T}}(\alpha_0)$. By the definition $g_{\mathcal{T}}(\alpha) = -\mathrm{d}C_{\mathcal{T}}/\mathrm{d}\alpha$, this

[29] We assume this threshold exists. The only scenario where no such threshold exists is the case in which all steps are executed manually for all $\alpha \in (0,1)$. This is the trivial, uninteresting case and is assumed away.

is

$$g_{\mathcal{T}}(\alpha_0) \;=\; -C'_{\mathcal{T}}(\alpha_0) \;\le\; -C'_{\mathcal{T}'}(\alpha_0) \;=\; g_{\mathcal{T}'}(\alpha_0).$$

Since $g^*(\alpha)$ coincides with $g_{\mathcal{T}}(\alpha)$ just below α_0 and with $g_{\mathcal{T}'}(\alpha)$ just above, and both are continuous at α_0, the firm's marginal benefit function $g^*(\alpha)$ does not fall as α crosses α_0. If in addition α_0 is transversal, then $\phi'(\alpha_0) \neq 0$, which together with $\phi'(\alpha_0) \le 0$ gives $\phi'(\alpha_0) < 0$ and hence $g_{\mathcal{T}}(\alpha_0) < g_{\mathcal{T}'}(\alpha_0)$, so the marginal benefit jumps upward discontinuously. Transversality cannot be dispensed with, because the sign change only forces α_0 to be a root of ϕ of odd order, and at a root of order three or higher the envelope's marginal benefit passes through the threshold without a jump.[30] The condition is nonetheless the generic case. A threshold already imposes $\phi(\alpha_0) = 0$, so requiring $\phi'(\alpha_0) = 0$ as well is a second equation in the same unknown, and the time cost vectors (t^M, t^A) for which it holds form a Lebesgue-null set. Both thresholds of Example 2 are transversal, with g^* rising from 0 to 16.0 at $\alpha = 0.50$ and from 4.7 to 134.1 at $\alpha \approx 0.92$. □

B.4 Proof of Proposition 3

Proof. We first note the following recursive formulation of the optimization problem. For all $k \le m$, let $C[k]$ denote the minimum cost of completing a hypothetical workflow that only includes steps 1 through k. Note that, because of the way we defined $C[k]$, step k cannot be automated in any minimum-cost solution that determines $C[k]$; it can only be augmented or performed manually. Note that $C[m]$ is the cost of the 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 work to complete. For $k \ge 1$, $C[k]$ is the lesser of

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

and

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

In other words, we either complete step k manually (at time t_k^M, separately optimizing over steps 1 through $k-1$) or we augment step k, at expected time $t_k^A / \prod_{i=\ell+1}^{k} q_i$. In the latter case, we then optimize over the length of the AI chain that ends with the newly-augmented

[30] With $m = 4$, $d = (2,1,2,1)$, $t^M = (1000, 1000, 56, 1000)$ and $t^A = (102, 65, 1000, 12)$, the strategy $\mathcal{T}$ that chains steps 1–2, performs step 3 manually, and augments step 4 is uniquely optimal just below $\alpha_0 = 1/2$, and the strategy $\mathcal{T}'$ that augments step 1 and chains steps 2–4 is uniquely optimal just above it. Writing $x = 1/\alpha$, their cost difference is $\phi = 12x^4 - 65x^3 + 102x^2 - 12x - 56 = (x-2)^3(12x+7)$, which has a triple root at $x = 2$, and g^* falls through α_0 from 3170.5 to 3165.5 without jumping.

step 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 steps 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 optimal AI strategy) in total time $O(m^2)$. □

B.5 Proof of Proposition 4

In the long-run optimization problem, the firm anticipates the adjustment of worker wages to fit the skill requirements of each job and sets job responsibilities accordingly. This contrasts with the AI deployment problem of Section 3.3, which minimizes the time production takes while holding the assignment of steps to a single worker fixed. The long-run problem additionally optimizes how tasks are bundled into jobs.

The firm's long-run problem is stated in (8) of Section 5.3. It designs both the partition $\mathcal{T}$ of steps into tasks and the partition $\mathcal{J}$ of tasks into jobs, where the choice of $\mathcal{T}$ determines the time and skill requirements of each task and, given $\mathcal{T}$, the choice of $\mathcal{J}$ determines worker wage rates and the time needed per unit of output, including hand-off costs.

What makes this harder than the short-run problem of Proposition 3 is that steps must be assigned to tasks and the resulting tasks to workers at the same time, with hand-off costs at job boundaries counted along the way. Proposition 4 states that the problem nonetheless admits a polynomial-time approximation scheme, which we prove here.

Proof. We first state the cost minimization problem as a recursive problem and then use dynamic programming to approximate the optimal solution.

Recursive formulation. Consider a workflow with m steps $\mathcal{S} = (s_1, s_2, \ldots, s_m)$. Because the wage bill of a job multiplies its total skill by its total time, the running value of either alone never reveals how an added task changes it, so the program carries both and builds the job design up one step at a time. 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.

For job designs whose final job does not necessarily terminate after step i but rather

continues onward to subsequent steps, we need an auxiliary function. For $0 \le i \le m$ and skill and time levels c and t, let $R(i,\ c,\ t)$ denote the minimum cost of completing steps 1 through i when the worker currently accumulating work, whom we call the "active worker," has already been assigned tasks with total skill cost c and total time cost t, where t also includes any hand-off cost for the active worker. The minimum is taken over AI deployment strategies and job designs together. The recursion may either expand the active worker's job or close it and assign the remaining steps to other workers. Throughout, step i is the highest-indexed step that has not yet been assigned. In the base case no steps remain to assign, the active worker's totals are exactly (c, t), and their wage bill is the product of the two,

$$R(0,\ c,\ t) = c\,t.$$

The two functions are linked as follows. In $V(i)$, all job designs must end with a job that completes at step i and hands off to the following worker. Incurring this hand-off cost is equivalent to opening a job for the active worker whose accumulated time already includes t_i^H and whose accumulated skill is still 0, which is the state $(i,\ 0,\ t_i^H)$, and then assigning step i to that worker either manually or as the endpoint of an AI chain. Since a job must contain at least one task by Definition 6, the active worker's job cannot be closed while it is empty, so $V(0) = 0$ and, for $1 \le i \le m$,

$$V(i) = \min\left\{ R\big(i-1,\ c_i^M,\ t_i^H + t_i^M\big),\quad \min_{0 \le r < i} R\left(r,\ c_i^A,\ t_i^H + \frac{t_i^A}{\prod_{i'=r+1}^{i} q_{i'}}\right)\right\}.$$

For each $i \ge 1$, the minimum cost function $R(i,\ c,\ t)$ satisfies the recursive relation

$$R(i,\ c,\ t) = \min\Bigg\{ \underbrace{c\,t + V(i)}_{\text{job closed}},\ \underbrace{R\big(i-1,\ c + c_i^M,\ t + t_i^M\big)}_{\text{Step } i \text{ is manual}},\ \underbrace{\min_{0 \le r < i} R\left(r,\ c + c_i^A,\ t + \frac{t_i^A}{\prod_{i'=r+1}^{i} q_{i'}}\right)}_{\text{Step } i \text{ optimally chained}}\Bigg\}.$$

In words, when assigning step i, the three options the firm considers are:

- **Job closure.** 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). This option is the margin new to the long run. It is available only when the active worker already holds at least one task, that is when $c > 0$. The states with $c = 0$ are exactly the job-opening states $(i,\ 0,\ t_i^H)$, whose value is the $V(i)$ defined above; without this restriction the

recursion evaluated at such a state would read $V(i) = \min\{V(i), \cdot\}$, which any value below the other two branches satisfies, so it determines nothing.

- **Manual step.** The term $R(i-1,\ c + c_i^M,\ t + t_i^M)$ corresponds to designating step i for manual completion, and adding the corresponding (singleton) task. The manual execution costs (c_i^M, t_i^M) 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, and steps 1 through $i-1$ are left to be assigned optimally.

- **AI chain.** The term $\min_{0 \le r < i}\ R\left(r,\ c + c_i^A,\ t + \frac{t_i^A}{\prod_{i'=r+1}^{i} q_{i'}}\right)$ 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 verification costs are added to the active worker's skill and time costs. The cut r is the highest-indexed step that is not added to this AI chain, which can be any value between 0 and $i-1$.

Note that in evaluating the AI chain option, step i can only be augmented (rather than automated), as each step of the recursion adds a full AI chain to the active worker's job; AI strategies in which step i is automated are considered when performing the calculations $R(r,\ c,\ t)$ with $r > i$. The two functions are filled in increasing order of i: first $V(i)$, which calls $R(r,\ \cdot,\ \cdot)$ only at indices $r < i$, and then $R(i,\ c,\ t)$ for $c > 0$, which may call $V(i)$. The cost of the optimal joint AI strategy and job design for the full sequence of m steps is then $V(m)$, recalling that $t_m^H = 0$.

Discretization and Computation Time. We solve the recursion by dynamic programming, filling in the values of $R(i,\ c,\ t)$ for all i, c, and t. Since c and t take continuous values, we discretize both to powers of $(1+\epsilon')$, rounding every running total *up* to the next such power, where $\epsilon > 0$ is the target accuracy and

$$\epsilon' \ = \ \frac{\ln(1+\epsilon)}{2m+1} \ = \ \Theta(\epsilon/m).$$

We show below why ϵ' must be smaller than ϵ by a factor of order m.

Suppose all skill and time costs, and all non-zero hand-off costs, lie in $[1/B, B]$ for some $B > 0$; recall that the final job's hand-off cost t_m^H is zero. Completing every step manually with a separate worker incurs, for each step, a time and hand-off cost of at most B each and a skill cost of at most B, for a total cost of at most $2B^2$ per step, so no solution containing a job with total cost above $2mB^2$ need be considered. Since skill costs combine additively and every non-zero skill or time cost is at least $1/B$, a job's total skill lies in $[1/B,\ mB]$, and

its total time is at most $2mB^2/(1/B) = 2mB^3$. It therefore suffices to tabulate skill costs in $[1/B,\ mB]$ and time costs in $[1/B,\ 2mB^3]$.

Each of these ranges contains $O\big(\epsilon'^{-1}\log(mB)\big) = O\big(m\,\epsilon^{-1}\log(mB)\big)$ powers of $(1+\epsilon')$; rounding upward can carry a total past the bounds above by at most one factor of $(1+\epsilon')$, which changes these counts only by an additive constant. The table is indexed by the step i, which takes $m+1$ values, together with the discretized skill and time levels, each taking $O\big(m\,\epsilon^{-1}\log(mB)\big)$ values, so it holds

$$(m+1)\cdot O\big(m\,\epsilon^{-1}\log(mB)\big)\cdot O\big(m\,\epsilon^{-1}\log(mB)\big) \;=\; O\big(m^3\,\epsilon^{-2}\log^2(mB)\big)$$

entries in total. Each entry is filled in $O(m)$ time by taking the minimum over the job-closure option, the manual option, and the at most m chain cuts r in the recursion above, so filling the whole table takes

$$O\big(m^3\,\epsilon^{-2}\log^2(mB)\big)\cdot O(m) \;=\; O\big(m^4\,\epsilon^{-2}\log^2(mB)\big)$$

time. Recording which option attains each minimum lets the firm read the optimal AI strategy and job design off $V(m) = R(m,\ 0,\ 0)$.

It remains to bound the discretization error. The table is indexed by the rounded totals, so each transition adds an increment to a value that has already been rounded and then rounds again; the distortions therefore compound along a job rather than being incurred once. A job holding k tasks rounds its skill coordinate k times, once for each task added, and its time coordinate $k+1$ times, once when the job opens at the state $(i,\ 0,\ t_i^H)$ and once for each task thereafter. Since we round upward, an induction on the number of accumulations shows that the recorded skill exceeds the exact total by a factor of at most $(1+\epsilon')^k$ and the recorded time by a factor of at most $(1+\epsilon')^{k+1}$. Each job's wage bill is formed by the single multiplication $c\,t$ at the point the job-closure option closes it, so it exceeds the true bill by at most $(1+\epsilon')^{2k+1} \le (1+\epsilon')^{2m+1}$, which our choice of ϵ' makes at most $e^{\ln(1+\epsilon)} = 1+\epsilon$.[31] Every job's bill is therefore overstated by at most this factor and understated by none, so the pair $(\mathcal{T},\ \mathcal{J})$ the table returns has true cost at most $(1+\epsilon)$ times the minimum. □

C CES Representation at Macro Level

Sections 3–5 study a single firm's choices, whereas the automation literature models production directly at the level of the aggregate economy (Acemoglu and Restrepo, 2018, 2022; Acemoglu, 2025). This section shows that, under additional assumptions, the two views connect. A firm's many-input, task-level Leontief technology reduces to a two-

[31] Using $1+x \le e^x$, so that $(1+\epsilon')^{2m+1} \le e^{\epsilon'(2m+1)} = e^{\ln(1+\epsilon)}$.

input Leontief in skill-adjusted AI management labor and skill-adjusted manual labor (Appendix C.1), and aggregating across firms that differ only in how effectively they deploy the same AI technology yields an economy-wide CES production function. This tractably links micro-level organization decisions to the macroeconomic implications of improving AI quality.

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 chains), and (2) skill-adjusted manual labor (to perform manual tasks). These two types of labor demand different compensations, in that executing an AI chain with skill and time cost of one demands w_A, while completing a manual task with similar skill and time requirements commands a compensation of w_M. We call w_A and w_M the *base wage rates* for AI management and manual labor, respectively.

Firms use these two inputs to complete tasks. The skill and time requirements of tasks are set by the AI strategy $\mathcal{T}$, and the skill requirement of a job by $\mathcal{T}$ and the job design $\mathcal{J}$ together: a worker's compensation per unit of time depends on all tasks in the job, so the more tasks a job includes, the higher the compensation required for every task in it. The total compensation of a job is the sum of the skill costs of its tasks, each weighted by the base wage rate of the labor that performs it. Specifically, the compensation for job J will be:

$$w_M \left(\sum_{T_b^M \in J} c_b^M \right) + w_A \left(\sum_{T_b^A \in J} c_b^A \right), \tag{C.1}$$

which is composed of the sum of contributions of skill costs from its manual tasks denoted by set T_b^M (each weighted by w_M) and its AI chains denoted by set T_b^A (each weighted by w_A).[32]

Next, we define skill-adjusted labor, which is the smallest unit of work in our analysis. Let $J(b)$ denote the job to which task b belongs, and $E(b)$ be the mode of execution of task b (i.e., $E(b) = M$ if b is a manual task, or $E(b) = A$ if b is an AI chain). Write d_b for the total difficulty of the steps task b hands to AI, so that $d_b = \sum_{i=\ell}^{r} d_i$ for a chain spanning $(s_\ell, \ldots, s_r)$ and $d_b = 0$ for a manual step. A task's time requirement for a single attempt inherits its mode from $E(b)$ and is written $t_b^{E(b)}$, which is t_b^M when task b is a manual step, and t_b^A, the cost of verifying the chain's augmented endpoint, when it is an AI chain. Recall

[32]This formulation is essentially equivalent to the original definition of wage in (5). If we multiply each skill cost in Equation (C.1) by its respective base wage rate and redefine the task's skill cost as the resulting product, we arrive at the original formulation given by (5). The formulation in (C.1) allows tasks to contribute differently to the job's wage depending on their execution mode and the type of labor performing them.

from Definition 5 that the *expected* time cost of a task is $t_b = t_b^{E(b)} \alpha^{-d_b}$, so that below we write the failure inflation α^{-d_b} explicitly wherever an expected quantity is formed rather than folding it into $t_b^{E(b)}$. Define the skill-adjusted time requirement of task b as

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

Note that the fraction appearing behind $t_b^{E(b)}$ 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 of task b. 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 $w_A \tau_b \alpha^{-d_b}$ if it is an AI chain or as $w_M \tau_b$ if it is executed manually.

C.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.[33] We show that the firm's production function can be represented by a Leontief function with two firm-level aggregate inputs: AI management labor and manual labor.

Suppose the firm solves the long-run cost minimization problem in (8) for a given AI quality level α, and obtains the optimal AI 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. With $\mathcal{T}$ and $\mathcal{J}$ fixed, each step's execution mode, the job boundaries, and each job's total skill requirement are given, so we can work directly with tasks rather than steps.

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 $t^H(J_j)$ and skill requirement equal to the total skill requirement of job J_j.[34] (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 chains and thus require AI management labor, while tasks $k+1$ to

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

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

n, along with the hand-off tasks $n+1$ to $n+p-1$, are executed manually and require manual labor.[35]

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

$$y = \min\left\{\frac{l_1}{\tau_1^A\,\alpha^{-d_1}}, \cdots, \frac{l_k}{\tau_k^A\,\alpha^{-d_k}}, \frac{l_{k+1}}{\tau_{k+1}^M}, \cdots, \frac{l_n}{\tau_n^M}, \frac{l_{n+1}}{\tau^H(J_1)}, \cdots, \frac{l_{n+p-1}}{\tau^H(J_{p-1})}\right\}. \tag{C.2}$$

The y 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 AI strategy and job design are fixed, the required amount of skill-adjusted time to spend on each task in the denominators of Equation (C.2) are fixed, and the firm takes them as given. In equilibrium, allocation of labor to tasks satisfies

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

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:

$$\frac{\partial}{\partial l_z}\left(\frac{\frac{\Delta y}{\Delta l_a}}{\frac{\Delta y}{\Delta l_b}}\right) = 0, \qquad \forall z \neq a, b. \tag{C.3}$$

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 AI management labor and manual 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).

The condition expressed in (C.3) satisfies the aggregation requirement above. Thus, the firm's production function can be represented as:

$$y = \min\left\{\frac{\bar{\alpha}\,l_A}{\tau_A}, \frac{l_M}{\tau_M}\right\}. \tag{C.4}$$

Here, l_A and l_M represent respectively the firm-level aggregate AI management labor and manual labor. Moreover, the aggregate skill-adjusted AI and manual time requirements, τ_A and τ_M, are defined as the sum of skill-adjusted time requirements of tasks in their

[35] The final job p requires no hand-off by definition, so the last hand-off occurs at the end of job $p-1$.

respective groups:

$$\tau_A = \sum_{b=1}^{k} \tau_b^A, \qquad \tau_M = \sum_{j=1}^{p-1} \tau^H(J_j) + \sum_{b=k+1}^{n} \tau_b^M. \tag{C.5}$$

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

$$\bar{\alpha} = \frac{\sum_{b=1}^{k} \tau_b^A}{\sum_{b=1}^{k} \tau_b^A \alpha^{-d_b}}, \tag{C.6}$$

so that the following relationship holds:

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

In short, Equation (C.4) allows us to analyze labor allocation using the simpler two-input Leontief production function for a given AI strategy $\mathcal{T}$ and job design $\mathcal{J}$. This firm-level aggregate Leontief function implies the equilibrium condition:

$$y = \frac{\bar{\alpha}\, l_A}{\tau_A} = \frac{l_M}{\tau_M}. \tag{C.7}$$

C.2 Cross-Firm Aggregation: Leontief to CES

Next, we ask whether the Leontief production functions of many firms can be represented by a single economy-wide production function whose inputs aggregate the corresponding firm-level inputs. The aggregation literature shows that suitable heterogeneity across firms can smooth micro-level production functions into a well-behaved aggregate one (see Sato (1975), Chapters 2 and 4).[36] Most of its existence results do not yield a closed form; we show that under certain conditions on firm heterogeneity, firm-level Leontief production functions aggregate to 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 whose productivity is common across firms, so that firms make their labor decisions conditional on a given capital stock and capital plays no part in the allocation in (C.4).[37] We normalize the economy's aggregate capital stock to 1. Firms do not know their individual effective AI quality level before production occurs; instead, they share a common belief about the

[36] 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.

[37] Since we introduce firm-level heterogeneity through variations in effective AI quality alone, we abstract away from additional sources of heterogeneity across firms.

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 in (8), committing to an AI deployment 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 AI 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.[38] The resulting Leontief production function, determined by the previously selected AI strategy, job design, and the firm's realized effective AI quality, takes a form similar to Equation (C.4).

In what follows, we demonstrate that the heterogeneity in effective AI quality level can be structured so that micro-level Leontief production functions in (C.4) 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:

$$Y = (\theta_A A^\rho + \theta_M M^\rho + (1 - \theta_A - \theta_M) K^\rho)^{\frac{1}{\rho}}, \tag{C.8}$$

where Y represents the (economy-wide) aggregate output, A denotes aggregate AI management labor, M denotes aggregate manual labor, K is aggregate capital, and parameters θ_A and θ_M represent the weights of corresponding inputs. Aggregate capital is normalized to 1 as described above, so the third term is constant at $1 - \theta_A - \theta_M$; its exponent is part of the CES form we posit rather than something the aggregation derives.

To link the macro production function above to the micro production functions, we must relate aggregated firm-level inputs l_A and l_M from Equation (C.4) to their corresponding economy-wide aggregates A and M in Equation (C.8). This requires either determining the macro production function parameters from the distribution of heterogeneity, or specifying the form of heterogeneity given a set of macro production function parameters. We adopt the latter approach and derive the output density function in terms of effective AI quality, denoted by $\phi(\bar{\alpha})$, as a function of θ_A, θ_M, ρ. Throughout, we assume $\rho < 0$, so the elasticity of substitution is $\sigma < 1$, as is usual in this literature.

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

$$w_A l_A + w_M l_M \leq y.$$

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

Substituting for l_M and y from Condition (C.7) into the profitability condition yields:

$$w_A\, l_A + w_M\, \frac{\tau_M\, \bar{\alpha}\, l_A}{\tau_A} \leq \frac{\bar{\alpha}\, l_A}{\tau_A}.$$

Rearranging terms and canceling l_A gives the lower bound of firms' effective AI quality level:[39]

$$\bar{\alpha} \geq \frac{w_A\, \tau_A}{1 - w_M\, \tau_M}.$$

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 (C.6) that for $d_b \geq 0$ the upper bound of $\bar{\alpha}$ is 1 and is achieved when $\alpha \to 1^-$. Therefore:

$$\underline{\alpha} \equiv \frac{w_A\, \tau_A}{1 - w_M\, \tau_M} \leq \bar{\alpha} \leq 1. \tag{C.9}$$

Let $\phi(\bar{\alpha})$ denote output across firms by effective AI quality, normalizing the density of firms over $\bar{\alpha}$ to one, so that a firm of type $\bar{\alpha}$ produces $y = \phi(\bar{\alpha})$ and aggregate output is the integral of ϕ over the producing firms. We refer to ϕ as the distribution of effective AI quality in this sense. From Condition (C.7), it directly follows that the AI management labor and manual labor used by this firm are: $l_A = (\tau_A/\bar{\alpha})\, \phi(\bar{\alpha})$, and $l_M = \tau_M\, \phi(\bar{\alpha})$. Consequently, aggregate output Y, aggregate AI management labor A, and aggregate manual labor M can be expressed as:

$$Y = \int_{\underline{\alpha}}^{1} \phi(\bar{\alpha})\, d\bar{\alpha}, \qquad A = \int_{\underline{\alpha}}^{1} \tau_A \frac{\phi(\bar{\alpha})}{\bar{\alpha}}\, d\bar{\alpha}, \qquad M = \int_{\underline{\alpha}}^{1} \tau_M\, \phi(\bar{\alpha})\, d\bar{\alpha}. \tag{C.10}$$

Substituting the aggregated variables from the micro-level Equation (C.10) into the aggregate production function in (C.8), we obtain:

$$\int_{\underline{\alpha}}^{1} \phi(\bar{\alpha})\, d\bar{\alpha} = \left(\theta_A \left(\tau_A \int_{\underline{\alpha}}^{1} \frac{\phi(\bar{\alpha})}{\bar{\alpha}}\, d\bar{\alpha} \right)^{\rho} + \theta_M \left(\tau_M \int_{\underline{\alpha}}^{1} \phi(\bar{\alpha})\, d\bar{\alpha} \right)^{\rho} + (1 - \theta_A - \theta_M) \right)^{\frac{1}{\rho}}, \tag{C.11}$$

provided that the CES weights and the skill-adjusted time requirements satisfy

$$\theta_A\, \tau_A^{\rho} + \theta_M\, \tau_M^{\rho} = 1, \tag{C.12}$$

which is what the boundary condition at $\bar{\alpha} \to 1$ imposes.

Solving for $\phi(\bar{\alpha})$ in terms of θ_A, θ_M, and ρ then gives the following distribution for

[39] We assume the parameters w_A, w_M, τ_A, τ_M are such that $0 < (w_A\, \tau_A)/(1 - w_M\, \tau_M) < 1$.

effective AI quality:

$$\phi(\bar{\alpha}) = \frac{1}{1-\rho}\left(\frac{1-\theta_A-\theta_M}{\theta_A\,\tau_A^{\rho}}\right)^{\frac{1}{\rho}}(\bar{\alpha})^{\frac{1}{\rho-1}}\left[1-(\bar{\alpha})^{\frac{\rho}{\rho-1}}\right]^{-\frac{1+\rho}{\rho}}. \tag{C.13}$$

Subsection C.3 below derives this distribution.

This completes the production function aggregation procedure.

Two features of this construction are worth making explicit. First, because τ_M is common to all firms, Equation (C.10) gives $M = \tau_M Y$ at every wage vector, while capital is normalized to $K = 1$. As the participation threshold varies, the economy therefore traces a one-dimensional locus along which M/Y and K are both fixed, rather than filling out the space of input combinations. Equation (C.8) should accordingly be read as a representation valid along that locus rather than as an identification of the economy's technology over arbitrary (A, M, K). Second, recovering a genuine three-input CES requires firms to differ in at least two of their input requirements, as in Houthakker (1955) and Levhari (1968). Our two-stage timing rules this out by construction, since every firm commits to the same $\mathcal{T}$ and $\mathcal{J}$ before learning $\bar{\alpha}$. Allowing firms that anticipate deploying AI more effectively to organize work differently would make τ_M vary with $\bar{\alpha}$ and restore the missing dimension, at the cost of a joint distribution over organizational choices and realized quality that would be intractable without imposing further assumptions on the type of heterogeneity and firm behavior.

C.3 Derivation of the Effective AI Quality Heterogeneity Distribution

We derive (C.13) by extending the method of Levhari (1968).

Define $u = (w_A\,\tau_A)/(1 - w_M\,\tau_M)$ and rewrite Equation (C.11) as:

$$\left(\int_u^1 \phi(\bar{\alpha})\,d\bar{\alpha}\right)^{\rho} = \theta_A\left(\tau_A\int_u^1 \frac{\phi(\bar{\alpha})}{\bar{\alpha}}\,d\bar{\alpha}\right)^{\rho} + \theta_M\left(\tau_M\int_u^1 \phi(\bar{\alpha})\,d\bar{\alpha}\right)^{\rho} + (1-\theta_A-\theta_M). \tag{C.14}$$

Define:

$$\Gamma(u) = \int_u^1 \phi(\bar{\alpha})\,d\bar{\alpha}, \tag{C.15}$$

and

$$\Psi(u) = \int_u^1 \frac{\phi(\bar{\alpha})}{\bar{\alpha}}\,d\bar{\alpha}. \tag{C.16}$$

Note that $\Gamma'(u) = -\phi(u)$ and $\Psi'(u) = -\phi(u)/u$. Rewrite Equation (C.14) as:

$$\Gamma^{\rho}(u) = \theta_A \tau_A^{\rho} \Psi^{\rho}(u) + \theta_M \tau_M^{\rho} \Gamma^{\rho}(u) + (1 - \theta_A - \theta_M).$$

Rearranging terms, we obtain:

$$\left(1 - \theta_M \tau_M^{\rho}\right) \Gamma^{\rho}(u) = \theta_A \tau_A^{\rho} \Psi^{\rho}(u) + (1 - \theta_A - \theta_M). \tag{C.17}$$

Differentiating Equation (C.17) with respect to u yields:

$$\left(1 - \theta_M \tau_M^{\rho}\right) \Gamma^{\rho-1}(u) = \theta_A \tau_A^{\rho} \Psi^{\rho-1}(u)\, u^{-1}.$$

Solving for $\Psi^{\rho-1}(u)$ as a function of u and $\Gamma^{\rho-1}(u)$ we get:

$$\Psi^{\rho-1}(u) = \frac{\left(1 - \theta_M \tau_M^{\rho}\right)}{\theta_A \tau_A^{\rho}} \Gamma^{\rho-1}(u)\, u.$$

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

$$\Psi(u) = \left(1 - \theta_M \tau_M^{\rho}\right)^{\frac{1}{\rho-1}} \left(\theta_A \tau_A^{\rho}\right)^{\frac{1}{1-\rho}} u^{\frac{1}{\rho-1}} \Gamma(u). \tag{C.18}$$

Substituting $\Psi(u)$ from Equation (C.18) into Equation (C.17), we have:

$$\left(1 - \theta_M \tau_M^{\rho}\right) \Gamma^{\rho}(u) = \left(1 - \theta_M \tau_M^{\rho}\right)^{\frac{\rho}{\rho-1}} \left(\theta_A \tau_A^{\rho}\right)^{\frac{1}{1-\rho}} u^{\frac{\rho}{\rho-1}} \Gamma^{\rho}(u) + (1 - \theta_A - \theta_M)\,.$$

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

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

The relation $\Gamma' = -\phi$ pins down Γ only up to an additive constant, so the expression above must still be checked against the boundary condition $\Gamma(1) = 0$ that (C.15) imposes by construction. Evaluating the bracket at $u = 1$ leaves

$$1 - \theta_M \tau_M^{\rho} - \left(1 - \theta_M \tau_M^{\rho}\right)^{\frac{\rho}{\rho-1}} \left(\theta_A \tau_A^{\rho}\right)^{\frac{1}{1-\rho}},$$

which vanishes if and only if $1 - \theta_M \tau_M^{\rho} = \theta_A \tau_A^{\rho}$, that is, if and only if Restriction (C.12) holds. Under that restriction the bracket collapses to $\theta_A \tau_A^{\rho}\left[1 - u^{\rho/(\rho-1)}\right]$, so that

$$\Gamma(u) = \left(\frac{1 - \theta_A - \theta_M}{\theta_A \tau_A^{\rho}}\right)^{\frac{1}{\rho}} \left[1 - u^{\frac{\rho}{\rho-1}}\right]^{-\frac{1}{\rho}}, \tag{C.19}$$

which satisfies $\Gamma(1) = 0$ because $-1/\rho > 0$ for $\rho < 0$. Restriction (C.12) is therefore not an auxiliary assumption but the condition under which a density consistent with (C.15)

exists at all. It ties the CES weights to τ_A and τ_M, and hence to the firm's AI strategy and job design. It also delivers $1 - \theta_M \tau_M^\rho = \theta_A \tau_A^\rho > 0$, so the real powers appearing throughout are well defined.

Recall that $\Gamma'(u) = -\phi(u)$. Differentiating (C.19), we obtain:

$$\Gamma'(u) = \frac{1}{\rho - 1} \left(\frac{1 - \theta_A - \theta_M}{\theta_A \tau_A^\rho} \right)^{\frac{1}{\rho}} u^{\frac{1}{\rho-1}} \left[1 - u^{\frac{\rho}{\rho-1}} \right]^{-\frac{1+\rho}{\rho}}.$$

Thus, we have:

$$\phi(\bar{\alpha}) = \frac{1}{1 - \rho} \left(\frac{1 - \theta_A - \theta_M}{\theta_A \tau_A^\rho} \right)^{\frac{1}{\rho}} (\bar{\alpha})^{\frac{1}{\rho-1}} \left[1 - (\bar{\alpha})^{\frac{\rho}{\rho-1}} \right]^{-\frac{1+\rho}{\rho}},$$

which is the formula provided in (C.13) above.

References for Online Appendix

Acemoglu, D. (2025). The Simple Macroeconomics of AI. *Economic Policy 40*(121), 13–58.

Acemoglu, D. and P. Restrepo (2018). The Race between Man and Machine: Implications of Technology for Growth, Factor Shares, and Employment. *American Economic Review 108*(6), 1488–1542.

Acemoglu, D. and P. Restrepo (2022). Tasks, Automation, and the Rise in U.S. Wage Inequality. *Econometrica 90*(5), 1973–2016.

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

Felipe, J. and F. M. Fisher (2003). Aggregation in Production Functions: What Applied Economists Should Know. *Metroeconomica 54*(2-3), 208–262.

Fisher, F. M. (1965). Embodied Technical Change and the Existence of an Aggregate Capital Stock. *The Review of Economic Studies 32*(4), 263–288.

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

Leontief, W. (1947). Introduction to a Theory of the Internal Structure of Functional Relationships. *Econometrica*, 361–373.

Levhari, D. (1968). A Note on Houthakker's Aggregate Production Function in a Multifirm Industry. *Econometrica*, 151–154.

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

Sato, K. (1975). *Production Functions and Aggregation*. North-Holland.

Chaining Tasks, Redefining Work: A Theory of AI Automation

Mert Demirer, John J. Horton, Nicole Immorlica, Brendan Lucier, and Peyman Shahidi

Supplementary Appendix

(Not for Publication)

SA.A Construction Details of the O*NET Sample

This appendix describes how the dataset used in our analyses is constructed from the four sources listed in Section 7:

1. The O*NET 27.3 Database (National Center for O*NET Development, 2023),
2. Human-generated AI exposure labels for O*NET tasks from Eloundou et al. (2024),
3. Realized AI execution labels for O*NET tasks from the Anthropic Economic Index (Handa et al., 2025),
4. A GPT-generated task ordering for each O*NET occupation.

In what follows, we use the terms step, task, and O*NET task interchangeably, as in the main text; likewise, we use job and occupation interchangeably.

We use the May 2023 release of O*NET. This version contains roughly 18,000 tasks assigned to more than 850 U.S. occupations. For AI exposure measures, we use human-generated labels from Eloundou et al. (2024). Throughout the paper, our measure of AI exposure combines their E1 and E2 labels. We treat tasks with a human-assigned E1 or E2 label as exposed to AI and all other tasks as unexposed, which is the same label set from which the empirical fragmentation index of Section 7.3 is constructed.[40]

To obtain AI execution labels, we draw on Anthropic's Economic Index dataset (Handa et al., 2025), which classifies millions of Claude conversations into six categories: *"Validation"*, *"Task Iteration"*, *"Learning"*, *"Directive"*, *"Feedback Loop"*, and *"Filtered."* Conversations in the first three categories (Validation, Task Iteration, Learning) are treated by

[40] Eloundou et al. (2024) label a task E1 if access to an LLM would reduce the time a human needs to complete it by at least half while preserving output quality. E2 tasks reach the same time saving and quality only with additional software or tools built on top of the LLM. Any task that is neither E1 nor E2 is labeled E0.

Anthropic as AI-augmenting activities, whereas those in Directive and Feedback Loop are recognized as AI-automating activities. The sixth category, Filtered, corresponds to conversations that either could not be classified or whose actual category could not be disclosed for privacy reasons.

Anthropic reports the share of matched conversations falling into each of the six categories for a subset of O*NET tasks. In total, conversations are linked to 3,364 O*NET task statements, of which 1,066 have 100% of their conversations filtered, leaving 2,298 with at least one non-filtered conversation. Following Anthropic's definitions of AI-augmenting and AI-automating activities, we assign each of these 2,298 task statements an AI execution label based on the majority share of its non-filtered conversations across the augmenting versus automating categories.[41] Of the 2,298, 1,830 match a task statement in our O*NET sample; since the same statement can be listed under several occupations, these labels attach to 2,347 of the 17,925 occupation-task records in the final sample (1,626 augmentation and 721 automation). Table SA.A.1 provides several example tasks labeled using this procedure.

Finally, we treat tasks that do not appear in the Anthropic dataset, as well as those with 100% filtered conversations, as manual tasks. Figure SA.A.1 shows the distribution of task execution modes in our dataset. Of the 872 occupations, 809 (93%) contain at least one AI-exposed task and 555 (64%) contain at least one AI-executed task. Figure SA.A.2 shows the distribution of occupations by the fraction of their tasks that are exposed to AI (left) and executed by AI (right).

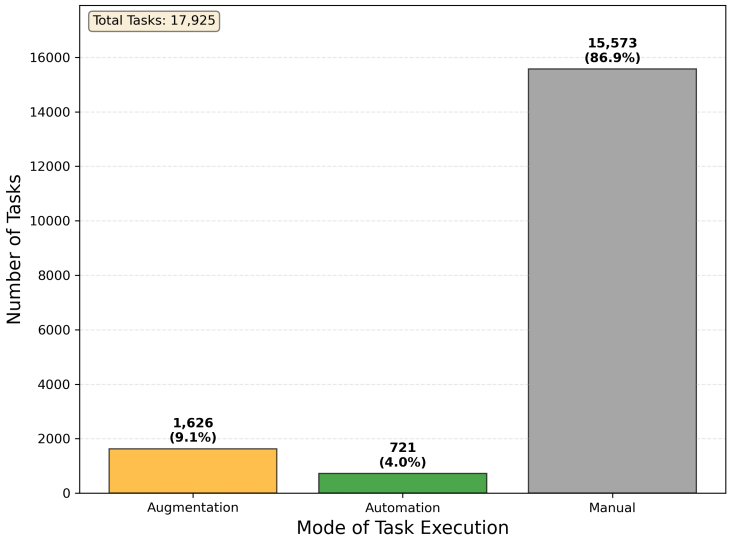
As the final input into our analyses, we generate a workflow sequence for every O*NET occupation. Specifically, we prompt GPT-5-mini through Expected Parrot (Horton and Horton, 2024) with the complete set of tasks within each occupation and ask it to determine the most reasonable sequential ordering in which these tasks would be performed in practice.[42] Figures SA.A.3, SA.A.4, and SA.A.5 show the resulting task sequences for three occupations: *Computer Programmers* with a high share of its tasks executed by AI, *Public Relations Specialists* with a moderate share, and *Electronic Equipment Installers and Repairers, Motor Vehicles* with no tasks executed by AI. We validate our approach in working with O*NET occupation task sequences in a variety of ways, discussed in Section 7.

Before closing, it is worth emphasizing a few points about the dataset we construct. First, note that we use a conservative measure of AI involvement in task execution. Recall

[41] For 23 of the 2,298 statements the two shares are exactly equal, and we break the tie toward automation. These statements account for 28 of the 721 automation records in the final sample. Because our analyses treat the two labels alike as indicators of AI execution, the tie-breaking convention does not affect any reported result.

[42] The prompt used to generate task sequences is given in Appendix SA.C.

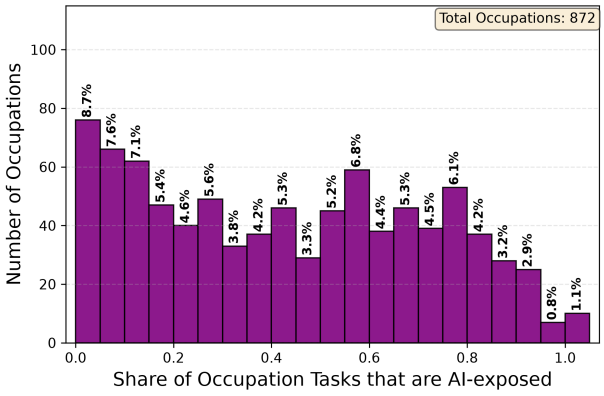
Figure SA.A.1: Distribution of Modes of Task Execution in the Dataset



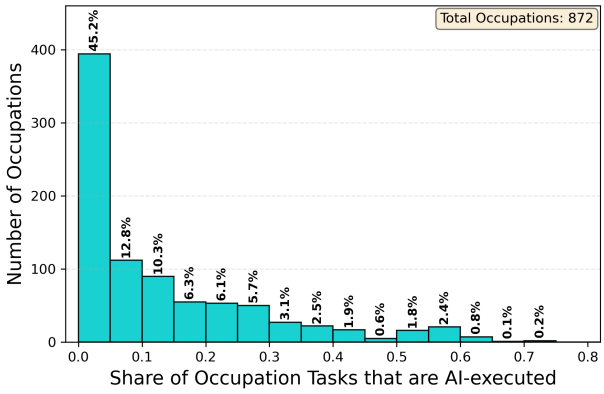
Notes: The automation and augmentation labels are drawn from Anthropic's Economic Index dataset, and the universe of tasks comes from the May 2023 O*NET release.

Figure SA.A.2: Distribution of Share of Occupation Tasks Exposed to and Executed by AI

(a) AI Exposure



(b) AI Execution



Notes: Percentages above bars denote fraction of observations corresponding to that bar out of all 872 occupations.

that we label tasks with fully filtered conversations as manual, even though we know that all of them were executed in some way using Claude. These tasks account for roughly one third of all Anthropic-mapped tasks, and excluding them limits our statistical power. Nevertheless, the remaining two thirds prove sufficient for revealing the patterns predicted by our framework, suggesting that what we report should be viewed as a conservative signal of potentially much stronger effects.

A second concern is that workers may use different AI tools for different purposes, for

example Claude for writing and ChatGPT for coding, which could potentially undermine our results. While we cannot directly rule out such heterogeneous tool use given that we only observe Claude usage, our analyses are not tied to particular industries, occupations, or tasks, which makes them less vulnerable to missing isolated pockets of tool-specific usage.[43] Nevertheless, wherever applicable, we verify that our results are not driven by any narrow subset of occupations or tasks. The patterns we document appear consistently across broad occupation groups, even if somewhat stronger in some groups than others, reflecting heterogeneous impacts of AI. While we do not claim that these relationships are causal, taken together they provide strong suggestive evidence that the mechanisms highlighted by our model are relevant in practice and that our findings reflect broad patterns of AI execution rather than Claude-specific usage.

Third, the realized execution modes in the constructed dataset are broadly consistent with, but do not perfectly satisfy, the model's definition of an AI chain. In the model an automated step can only be followed by another AI-executed step, either automated or augmented, since an automated step's output has to feed into a subsequent AI step rather than into a human. In the data we occasionally observe an automated task followed by a manual one. Task 5 in *Computer Programmers* (Figure SA.A.3) and Task 3 in *Public Relations Specialists* (Figure SA.A.4) are two such instances.

This is a definitional difference rather than a contradiction. The automated versus augmented distinction in our model is defined by production sequencing and task dependencies, whereas the Anthropic classifications are based on task-level feasibility and are agnostic to workflow position. A task is labeled automated if the conversations covering it are predominantly directive, regardless of what follows it in the occupation's workflow. Tasks labeled automated in the data may therefore be followed by manual steps, producing apparent deviations from the model's sequencing restriction without contradicting its underlying logic. For this reason the empirical analysis in Section 7 does not distinguish tasks labeled automated from those labeled augmented, and treats both as indicating AI execution.

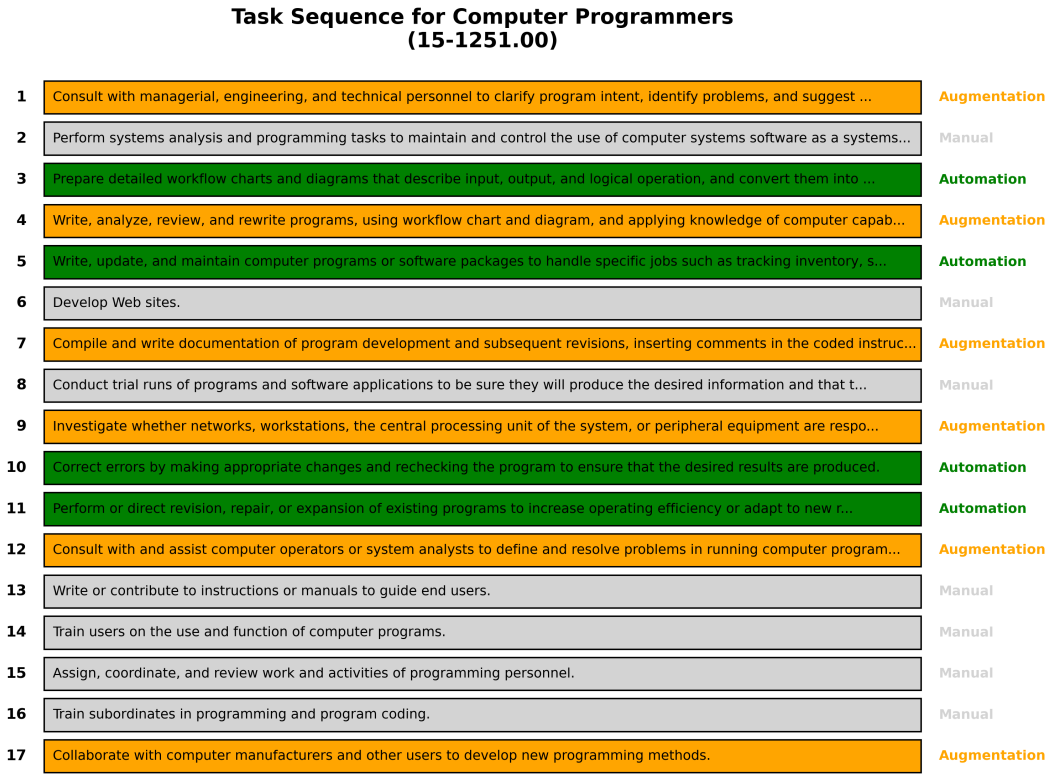
[43] For our results to be entirely spurious, one would have to believe not only that no one used Claude for a large subset of tasks during Anthropic's sample period but also that this missing class of tasks has distinctive characteristics, such as workflow position or AI-suitability, that differ from other tasks for which Claude is used across the economy. We do not have a reason to believe this is the case. If anything, specialization across tools would imply that Anthropic's data might understate the true set of AI-executed tasks in the economy, which would primarily reduce our statistical power rather than generate artificial patterns.

Table SA.A.1: Example O*NET Tasks Associated with Claude Conversations, and Calculated Augmentation vs. Automation Outcomes

O*NET Task	Share of Claude Conversations (%)								Label
	Validation	Task Iteration	Learning	Directive	Feedback Loop	Filtered	Augmentation	Automation	
Provide road information to assist motorists.	0.00	11.47	44.39	33.42	5.74	4.99	**55.86**	39.15	Augmentation
Calculate costs of orders, and charge or forward invoices to appropriate accounts.	0.00	0.00	0.00	68.00	0.00	32.00	0.00	**68.00**	Automation
Process data for analysis, using computers.	4.23	18.60	31.40	25.79	16.38	3.59	**54.23**	42.18	Augmentation
Adapt text to accommodate musical requirements of composers and singers.	0.00	40.00	0.00	58.26	0.00	1.74	40.00	**58.26**	Automation
Provide system design and integration recommendations.	2.13	29.75	39.98	21.23	5.80	1.11	**71.87**	27.02	Augmentation
Communicate traffic and crossing rules and other information to students and adults.	0.00	0.00	0.00	65.52	0.00	34.48	0.00	**65.52**	Automation
Prepare, manipulate, and manage extensive databases.	0.00	25.89	22.34	23.40	24.47	3.90	**48.23**	47.87	Augmentation
Prepare, administer, and grade tests and assignments to evaluate students' progress.	7.64	20.94	7.39	58.13	3.94	1.97	35.96	**62.07**	Automation
Conduct statistical analyses to quantify risk using statistical analysis software or econometric models.	0.00	25.23	31.53	17.12	18.92	7.21	**56.76**	36.04	Augmentation
Assemble, typeset, scan and produce digital camera-ready art or film negatives and printer's proofs.	0.00	21.88	0.00	77.34	0.00	0.78	21.88	**77.34**	Automation
Conduct searches to find needed information, using such sources as the internet.	1.31	5.53	58.30	23.14	3.57	8.15	**65.14**	26.71	Augmentation
Compile data and create statistical reports on library usage.	0.00	0.00	0.00	64.86	0.00	35.14	0.00	**64.86**	Automation
Create custom illustrations or other graphic elements.	0.00	48.17	3.17	43.82	2.74	2.10	**51.34**	46.56	Augmentation
Confer with clients to obtain and provide information when claims are made on a policy.	0.00	0.00	45.45	0.00	0.00	54.55	**45.45**	0.00	Augmentation

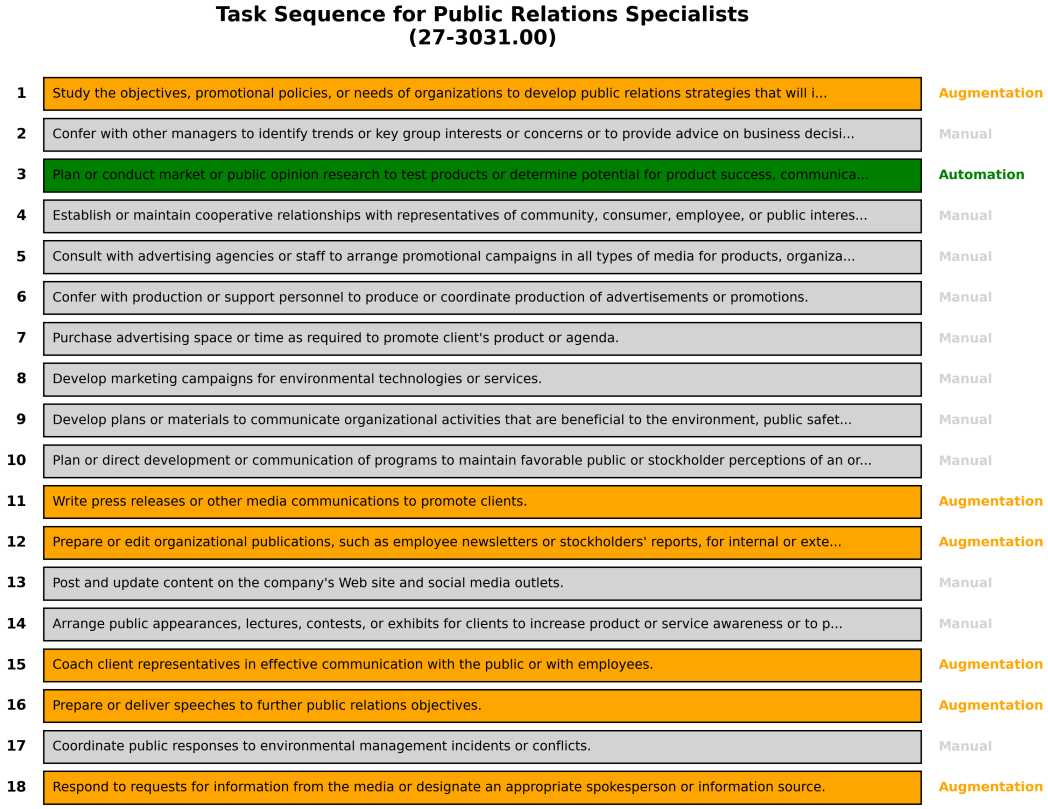
Notes: Augmentation percentage is the sum of Validation, Task Iteration, and Learning types of conversations whereas the automation percentage is the sum of Directive and Feedback Loop types of conversations. Whichever mode between Augmentation and Automation has a higher share is selected as the task's mode of AI execution, ignoring the Filtered share.

Figure SA.A.3: Task Sequence of Computer Programmers Occupation



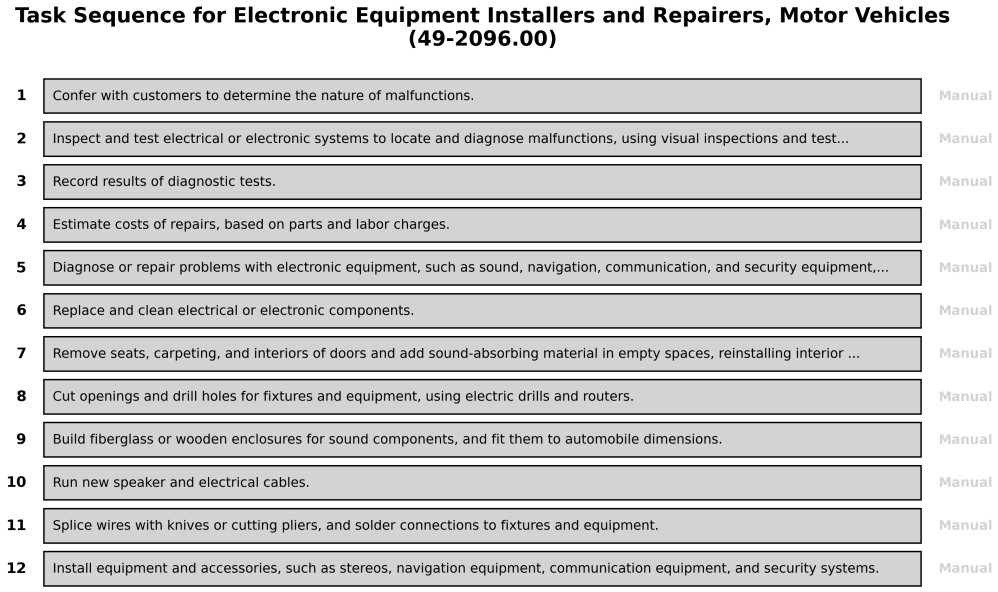
Notes: This figure shows the task sequence for an occupation with a high share of its tasks (about two thirds) executed by AI. The task ordering is generated by GPT-5-mini. The execution labels come from the Anthropic Economic Index. The O*NET code for this occupation is 15-1251.

Figure SA.A.4: Task Sequence of Public Relations Specialists Occupation



Notes: This figure shows the task sequence for an occupation with a moderate share of its tasks (about one third) executed by AI. Ordering and execution labels are as in Figure SA.A.3. The O*NET code for this occupation is 27-3031.

Figure SA.A.5: Task Sequence of Electronic Equipment Installers and Repairers, Motor Vehicles



Notes: This figure shows the task sequence for an occupation with none of its tasks executed by AI. Ordering and execution labels are as in Figure SA.A.3. The O*NET code for this occupation is 49-2096.

SA.B Alternative Definitions of Step Similarity and AI Execution

This appendix re-estimates Prediction #2 of Section 7 under alternative definitions of the variables it is built from, holding the task sequences fixed. It tightens what it means for two tasks to be the same step across occupations, and replaces AI execution with the stricter outcome of AI automation.

Although DWAs are intended to group conceptually similar tasks, one might argue that even within a given DWA tasks may still somewhat differ in their objectives, execution nature, or required skills. To address this concern, we re-estimate Equation (12) using a more restrictive definition of task similarity within DWAs. Specifically, for each DWA, we ask GPT-5-mini to select the subset of tasks that are most similar in terms of skill requirements and execution complexity.[44] This procedure yields, for each DWA, a set of highly comparable tasks that appear across different occupational contexts. Panel (a) of Table SA.B.4 presents example DWAs and their associated O*NET tasks across occupations, along with execution modes, while Panel (b) reports the subset of tasks retained after applying the similarity filter.

In this robustness test we treat these tasks as instances of the same underlying step appearing across occupations and re-estimate Equation (12) on this restricted sample. The resulting estimates, reported in Table SA.B.1 and Figure SA.B.1, closely mirror the baseline patterns of the main specification in both sign and magnitude.

Finally, we examine a stricter outcome that is closer to the model's notion of automation, under which automated steps can only precede other AI-executed steps, whether automated or augmented. Although the Anthropic automation label is not defined by workflow position (Appendix SA.A), we repeat the analysis focusing specifically on AI automation rather than AI execution more broadly, replacing the dependent variable in Equation (12) with an indicator for whether step k is AI-automated (is_automated_k) instead of AI-executed (is_ai_k). The corresponding estimates for the O*NET sample are reported in Table SA.B.2 and Figure SA.B.2, while results for the GPT-filtered sample are reported in Table SA.B.3 and Figure SA.B.3.[45] Under this stricter specification, the estimated effects are smaller in magnitude and often statistically insignificant, reflecting the more limited prevalence of AI automation in the data. Nevertheless, the signs and qualitative patterns of the estimates remain consistent with those obtained for AI execution.

[44] The prompt used to identify similar DWA tasks is provided in Appendix SA.C.

[45] Because relatively few tasks in the data are AI-automated, and because we further restrict attention to highly similar tasks across occupations, we retain tasks associated with multiple DWAs in the original O*NET dataset and randomly assign each such task to one of its associated DWAs for these exercises.

Table SA.B.1: Neighboring Tasks' AI Execution Status and a Task's AI Execution (GPT-filtered Sample)

	Probability that Focal Task (k) is AI-executed					
	(1)	(2)	(3)	(4)	(5)	(6)
Task ($k-2$) is AI-executed	0.04	0.01	0.00	0.04	0.04	0.04
	(0.03)	(0.02)	(0.02)	(0.03)	(0.04)	(0.03)
Task ($k-1$) is AI-executed	0.11***	0.06**	0.06**	0.06**	0.10***	0.06**
	(0.04)	(0.03)	(0.03)	(0.03)	(0.04)	(0.03)
Task ($k+1$) is AI-executed	0.11***	0.08***	0.07***	0.09***	0.10***	0.09***
	(0.03)	(0.03)	(0.02)	(0.03)	(0.03)	(0.03)
Task ($k+2$) is AI-executed	0.01	-0.02	-0.03	-0.00	0.00	-0.00
	(0.03)	(0.02)	(0.02)	(0.03)	(0.04)	(0.03)
Pseudo R^2	0.032	0.056	0.065	0.199	0.020	0.199
Observations	3,689	3,689	3,567	2,544	2,544	2,544
SOC Group FE		Major	Minor			
DWA FE				✓		✓
NumTasks in DWA-Occupation Control					✓	✓

Notes: Clustered standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1. This table reports average marginal effects from estimating Equation (12). The dependent variable in all specifications is an indicator for whether task k is AI-executed (is_ai$_k$). The sample is restricted to similar tasks across occupations, identified by GPT-5-mini as tasks belonging to the same DWA with similar execution nature and skill characteristics. Controls and standard errors are as in Table 2.

Table SA.B.2: Neighboring Tasks' AI Execution Status and a Task's AI Automation

	Probability that Focal Task (k) is AI-automated					
	(1)	(2)	(3)	(4)	(5)	(6)
Task ($k-2$) is AI-executed	0.02**	-0.01	-0.01	-0.01	0.02	-0.01
	(0.01)	(0.01)	(0.01)	(0.02)	(0.03)	(0.02)
Task ($k-1$) is AI-executed	0.04***	0.01*	0.01	0.01	0.08**	0.02
	(0.02)	(0.01)	(0.01)	(0.02)	(0.03)	(0.02)
Task ($k+1$) is AI-executed	0.05***	0.02***	0.02**	0.01	0.09**	0.01
	(0.01)	(0.01)	(0.01)	(0.02)	(0.03)	(0.02)
Task ($k+2$) is AI-executed	0.03***	0.00	-0.00	0.02	0.06*	0.02
	(0.01)	(0.01)	(0.01)	(0.02)	(0.03)	(0.02)
Pseudo R^2	0.102	0.192	0.173	0.242	0.036	0.242
Observations	13,786	12,777	10,390	2,815	2,815	2,815
SOC Group FE		Major	Minor			
DWA FE				✓		✓
NumTasks in DWA-Occupation Control					✓	✓

Notes: Clustered standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1. This table reports average marginal effects from estimating Equation (12) but with dependent variable (is_automated$_k$) instead of (is_ai$_k$). The sample is restricted to similar tasks across occupations, identified as tasks belonging to the same DWA in the O*NET dataset. Tasks that are mapped to multiple DWAs are retained, with one associated DWA assigned randomly for estimation to preserve sufficient variation in the data. Controls and standard errors are as in Table 2.

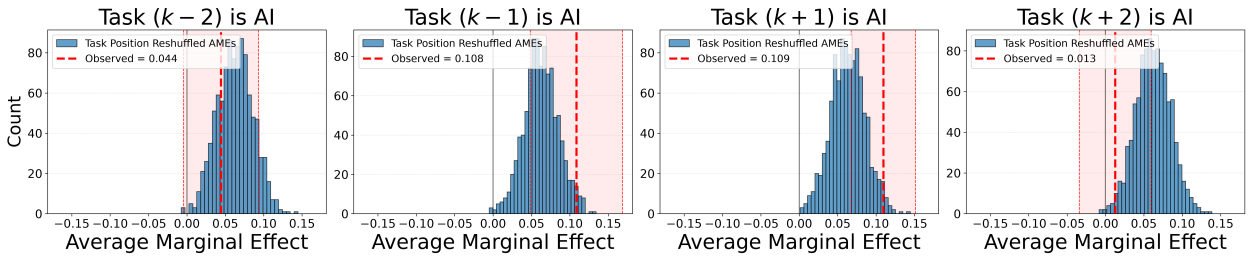
Table SA.B.3: Neighboring Tasks' AI Execution Status and a Task's AI Automation (GPT-filtered Sample)

	Probability that Focal Task (k) is AI-automated					
	(1)	(2)	(3)	(4)	(5)	(6)
Task ($k-2$) is AI-executed	0.01	-0.02	-0.02	0.00	0.01	0.00
	(0.01)	(0.01)	(0.02)	(0.03)	(0.03)	(0.03)
Task ($k-1$) is AI-executed	0.06**	0.02	0.02	0.03	0.11**	0.03
	(0.03)	(0.01)	(0.01)	(0.03)	(0.04)	(0.03)
Task ($k+1$) is AI-executed	0.06**	0.03*	0.03	0.02	0.10**	0.02
	(0.03)	(0.02)	(0.02)	(0.03)	(0.05)	(0.03)
Task ($k+2$) is AI-executed	0.01	-0.02	-0.02	0.01	0.02	0.01
	(0.02)	(0.01)	(0.02)	(0.02)	(0.04)	(0.03)
Pseudo R^2	0.042	0.125	0.118	0.244	0.033	0.244
Observations	5,156	5,077	4,490	1,771	1,771	1,771
SOC Group FE		Major	Minor			
DWA FE				✓		✓
NumTasks in DWA-Occupation Control					✓	✓

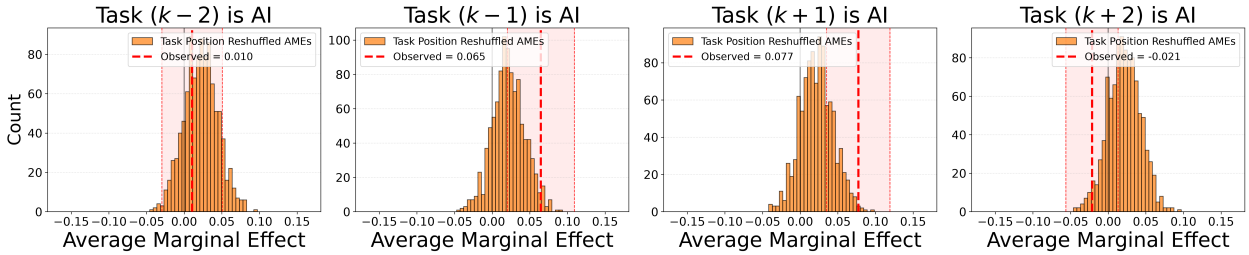
Notes: Clustered standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1. This table reports average marginal effects from estimating Equation (12) but with dependent variable (is_automated$_k$) instead of (is_ai$_k$). The sample is restricted to similar tasks across occupations, identified by GPT-5-mini as tasks belonging to the same DWA with similar execution nature and skill characteristics. Tasks that are mapped to multiple DWAs are retained, with one associated DWA assigned randomly for estimation to preserve sufficient variation in the data. Controls and standard errors are as in Table 2.

Figure SA.B.1: Neighboring Tasks' AI Execution Status and a Task's AI Execution (GPT-filtered Sample): Placebo Distributions

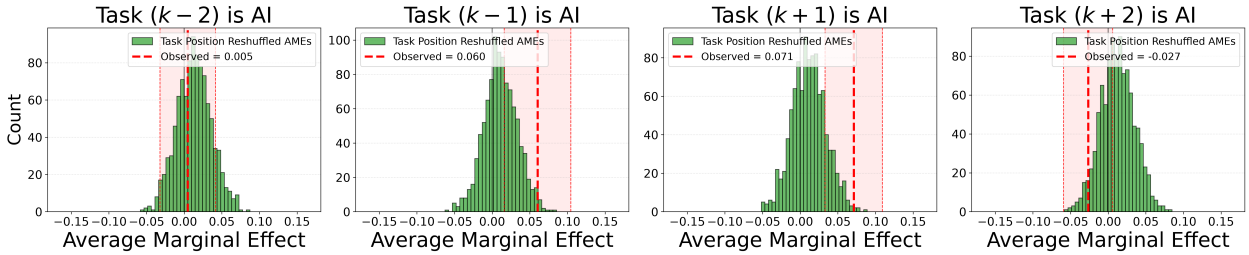
(a) No Fixed Effects



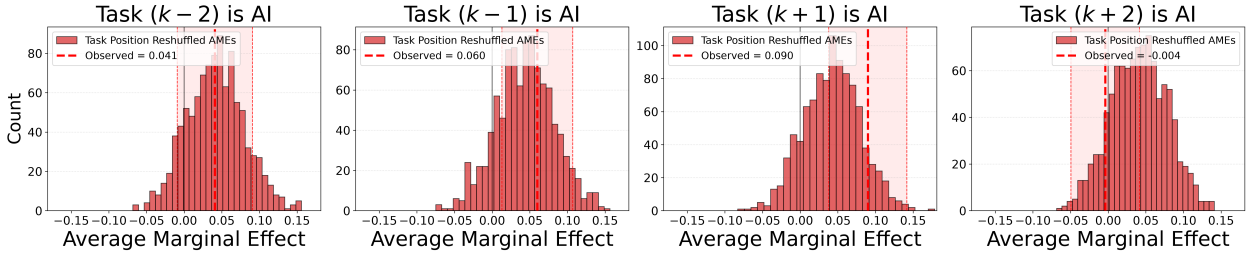
(b) SOC Major Group Fixed Effects



(c) SOC Minor Group Fixed Effects



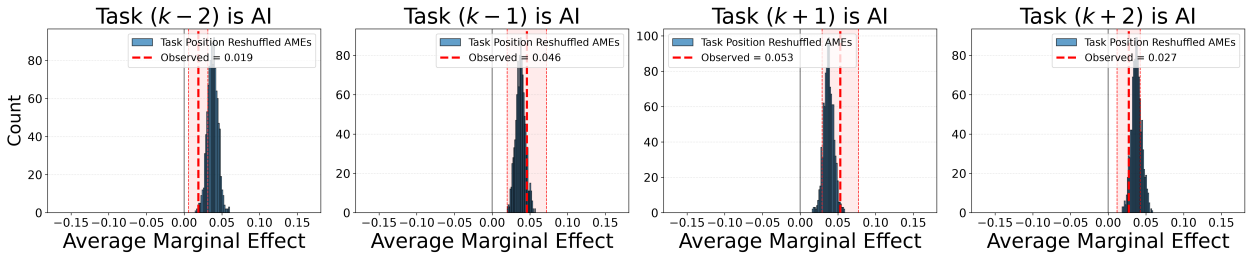
(d) Detailed Work Activity Fixed Effects



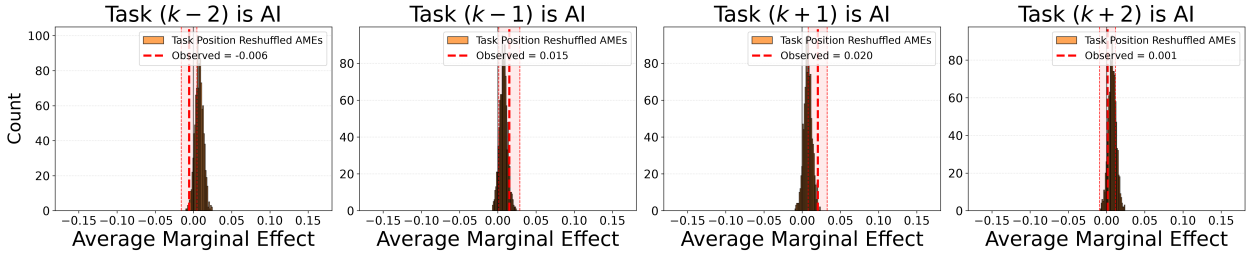
Notes: These graphs show that, among similar tasks appearing in multiple occupations, those sitting in a more AI-executed local context exhibit higher probabilities of being executed by AI, with the association concentrated on the steps immediately adjacent to the task. The dashed line, shaded band, and histograms are constructed as in Figure A.1, with the observed estimates taken from Table SA.B.1. Panel (a) reports results from Equation (12), while Panels (b), (c), and (d) augment the baseline specification with SOC major group, SOC minor group, and DWA fixed effects, respectively.

Figure SA.B.2: Neighboring Tasks' AI Execution Status and a Task's AI Automation: Placebo Distributions

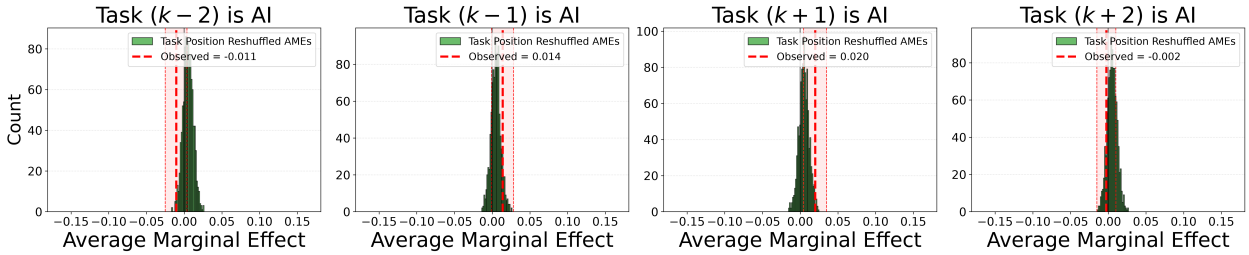
(a) No Fixed Effects



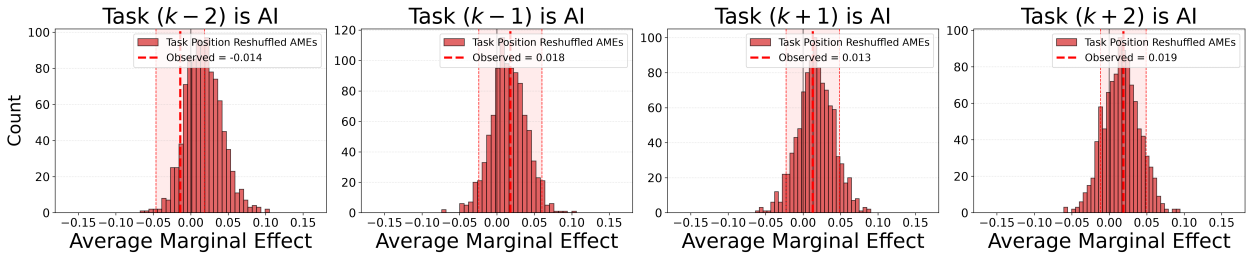
(b) SOC Major Group Fixed Effects



(c) SOC Minor Group Fixed Effects



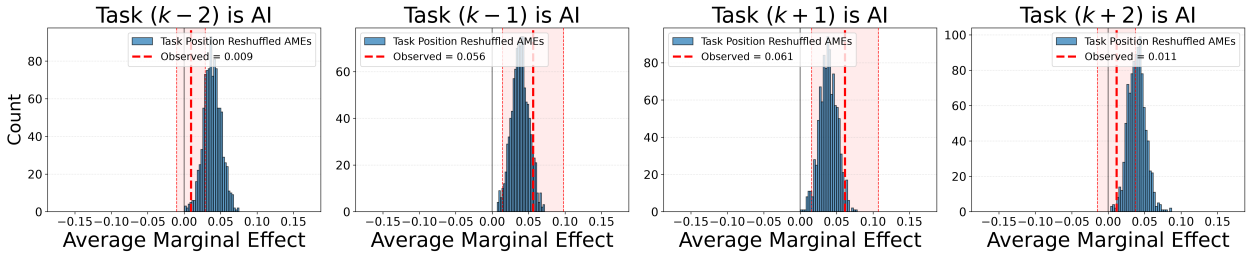
(d) Detailed Work Activity Fixed Effects



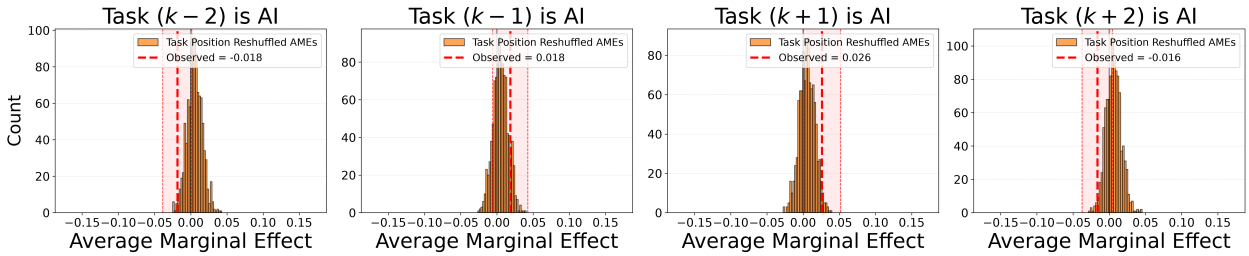
Notes: These graphs show that, among similar tasks appearing in multiple occupations, those sitting in a more AI-executed local context exhibit higher probabilities of being automated by AI, with the association concentrated on the steps immediately adjacent to the task. The dashed line, shaded band, and histograms are constructed as in Figure A.1, with the observed estimates taken from Table SA.B.2. Panel (a) reports results from Equation (12) but with dependent variable (is_automated$_k$) instead of (is_ai$_k$), while Panels (b), (c), and (d) augment the baseline specification with SOC major group, SOC minor group, and DWA fixed effects, respectively.

Figure SA.B.3: Neighboring Tasks' AI Execution Status and a Task's AI Automation (GPT-filtered Sample): Placebo Distributions

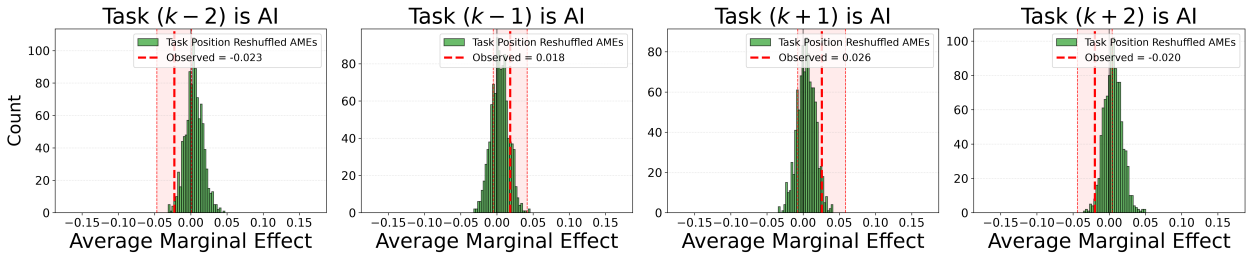
(a) No Fixed Effects



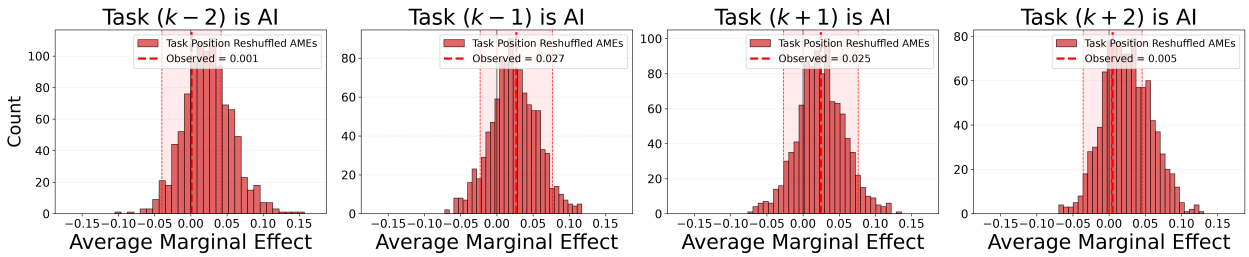
(b) SOC Major Group Fixed Effects



(c) SOC Minor Group Fixed Effects



(d) Detailed Work Activity Fixed Effects



Notes: The pattern is the same as in Figure SA.B.2, now on the GPT-filtered sample. The dashed line, shaded band, and histograms are constructed as in Figure A.1, with the observed estimates taken from Table SA.B.3. Panel (a) reports results from Equation (12) but with dependent variable (is_automated$_k$) instead of (is_ai$_k$), while Panels (b), (c), and (d) augment the baseline specification with SOC major group, SOC minor group, and DWA fixed effects, respectively.

Table SA.B.4: Example DWAs with Tasks in Multiple Occupations and the Tasks Surviving the "Finding Similar Tasks" Procedure

Panel (a)

O*NET DWA Title	O*NET Task Title	O*NET Occupation Title	Execution Label
Analyze operational or research data.	Analyze or manipulate bioinformatics data using software packages, statistical applications, or data mining techniques.	Bioinformatics Technicians	Augmentation
	Conduct quality analyses of data inputs and resulting analyses or predictions.		Augmentation
	Analyze research data to determine its significance, using computers.	Astronomers	Augmentation
	Collect and analyze data, such as studying old records, tallying the number of outpatients entering each day or week, or participating in federal, state, or local population surveys as a Census Enumerator.	Interviewers, Except Eligibility and Loan	Manual
	Analyze data to determine answers to questions from customers or members of the public.	Receptionists and Information Clerks	Augmentation
	Compute and analyze data, using statistical formulas and computers or calculators.	Statistical Assistants	Augmentation
Prepare research or technical reports on environmental issues.	Write reports or articles for Web sites or newsletters related to environmental engineering issues.	Environmental Engineers	Manual
	Prepare technical and research reports, such as environmental impact reports, and communicate the results to individuals in industry, government, or the general public.	Biologists	Manual
	Prepare scientific atmospheric or climate reports, articles, or texts.	Atmospheric and Space Scientists	Automation
	Prepare charts or graphs from data samples, providing summary information on the environmental relevance of the data.	Environmental Scientists and Specialists, Including Health	Augmentation
	Write reports or academic papers to communicate findings of climate-related studies.	Climate Change Policy Analysts	Manual
	Prepare study reports, memoranda, briefs, testimonies, or other written materials to inform government or environmental groups on environmental issues, such as climate change.		Manual
	Prepare technical and research reports, such as environmental impact reports, and communicate the results to individuals in industry, government, or the general public.	Industrial Ecologists	Manual
	Produce environmental documents, such as environmental assessments or environmental impact statements.	Transportation Planners	Manual
Document events or evidence using photographic or audiovisual equipment.	Take photographs and motion pictures for use in lectures and publications and to develop displays.	Park Naturalists	Manual
	Create data records for use in describing and analyzing social patterns and processes, using photography, videography, and audio recordings.	Anthropologists and Archeologists	Automation
	Create photographic recordings of information, using equipment.	Geological Technicians, Except Hydrologic Technicians	Automation
	Use photographic or video equipment to document evidence or crime scenes.	Forensic Science Technicians	Manual

Panel (b)

O*NET DWA Title	O*NET Task Title	O*NET Occupation Title	Execution Label
Analyze operational or research data.	Analyze or manipulate bioinformatics data using software packages, statistical applications, or data mining techniques.	Bioinformatics Technicians	Augmentation
	Analyze research data to determine its significance, using computers.	Astronomers	Augmentation
	Compute and analyze data, using statistical formulas and computers or calculators.	Statistical Assistants	Augmentation
Prepare research or technical reports on environmental issues.	Prepare technical and research reports, such as environmental impact reports, and communicate the results to individuals in industry, government, or the general public.	Biologists	Manual
	Prepare scientific atmospheric or climate reports, articles, or texts.	Atmospheric and Space Scientists	Automation
	Prepare charts or graphs from data samples, providing summary information on the environmental relevance of the data.	Environmental Scientists and Specialists, Including Health	Augmentation
	Prepare study reports, memoranda, briefs, testimonies, or other written materials to inform government or environmental groups on environmental issues, such as climate change.	Climate Change Policy Analysts	Manual
	Prepare technical and research reports, such as environmental impact reports, and communicate the results to individuals in industry, government, or the general public.	Industrial Ecologists	Manual
	Produce environmental documents, such as environmental assessments or environmental impact statements.	Transportation Planners	Manual
Document events or evidence using photographic or audiovisual equipment.	Take photographs and motion pictures for use in lectures and publications and to develop displays.	Park Naturalists	Manual
	Create data records for use in describing and analyzing social patterns and processes, using photography, videography, and audio recordings.	Anthropologists and Archeologists	Automation
	Create photographic recordings of information, using equipment.	Geological Technicians, Except Hydrologic Technicians	Automation

Notes: Panel (a) shows three example detailed work activities (DWAs) and tasks associated with them in the O*NET dataset. Entries highlighted with a red font are those that are dropped in the "finding similar tasks" procedure described in text. Panel (b) shows the set of tasks for DWAs of Panel (a) that successfully survive the "finding similar tasks" procedure.

SA.C GPT-5-mini Prompts

This Appendix provides the GPT-5-mini prompts used to generate a task sequence for each occupation (Prompt 1), and for finding similar tasks across occupations for each DWA (Prompt 2).

Prompt #1: GPT-5-mini Prompt for Ordering of Tasks in Occupation

```
You are an expert in workflow analysis for the occupation: {{ occupation }}.
Below is a list of {{ num_tasks }} tasks that are part of this occupation:
{{ tasks_list }}

Provide the typical sequential order in which these tasks are performed in a
real-world workflow.

Return your answer as a JSON array where each element has:
- "Task Position": the sequence number (1, 2, 3, etc.).
- "Task Title": the exact task text from the list above.
Format: [{"Task Position": 1, "Task Title": "..."},
         {"Task Position": 2, "Task Title": "..."},
         ...]
Only return the JSON array, nothing else.
```

Prompt #2: GPT-5-mini Prompt for Finding Similar Tasks in Each DWA

```
You are an expert in workflow analysis for the detailed work activity:
{{ detailed_work_activity }}.

Below is a list of {{ num_tasks }} task IDs and titles that belong to this
detailed work activity and appear across similar or different occupations
(tasks and occupations are ordered such that the first task belongs to the
first occupation, the second task belongs to the second occupation, etc.).
Tasks IDs: {{ tasks_ids }}
Tasks list: {{ tasks_list }}
Occupations list: {{ occupations_list }}
Occupation Codes list: {{ occupation_codes_list }}

Determine which tasks are similar in nature and in terms of their objectives,
methods, or required skills. There may be more than one task associated with
an occupation. Return only the most relevant task for every occupation.
Only look for tasks that are actually similar. Do not feel obliged to return
all occupations.

Return the task-occupation pairs you determine as similar as a JSON array
where each element has:
- "Task ID": the exact task ID from the list of task IDs above
- "Task Title": the exact task text from the list of tasks above
- "O*NET-SOC Code": the exact occupation code text from the list of occupation
   codes above
- "Occupation Title": the exact occupation text from the list of occupations
   above
Format: [{"Task ID": 1234, "Task Title": "...",
          "O*NET-SOC Code": "...", "Occupation Title": "..."},
          {"Task ID": 5678, "Task Title": "...",
          "O*NET-SOC Code": "...", "Occupation Title": "..."},
          ...]
Only return the JSON array, nothing else.
```

SA.D Robustness to Alternative GPT Prompts

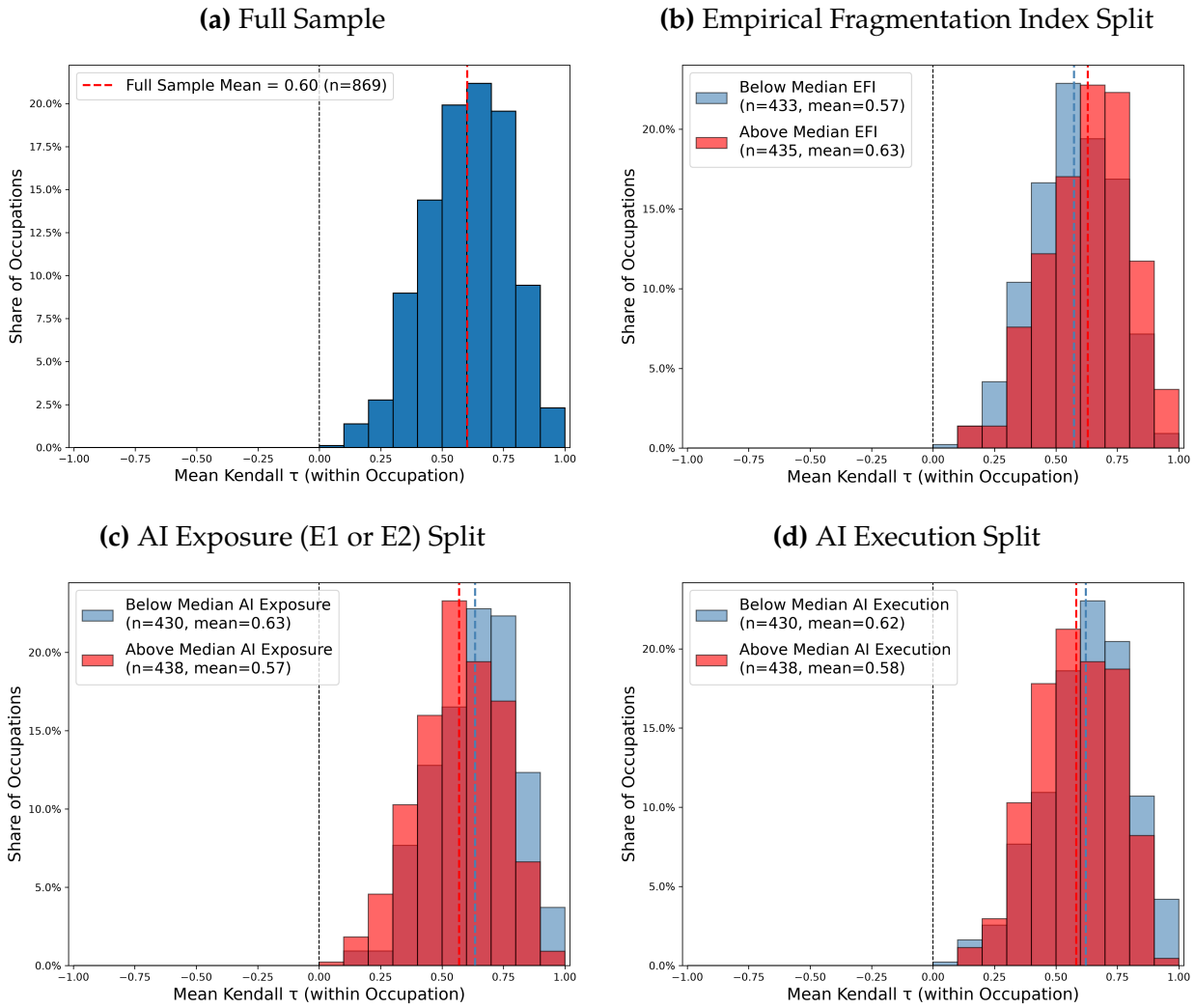
In this appendix, we assess whether our results are sensitive to the specific GPT prompt used to order tasks within occupations. Although alternative prompts can generate different task sequences, we show that these differences do not exhibit systematic patterns across subsets of occupations, and that our headline results are robust to prompt formulation.

We begin by generating task orderings for each occupation using 10 alternative prompts in addition to the baseline prompt. We use the same structure as the first prompt in Appendix SA.C, but only change the sentence starting with "*Provide the typical sequential...*" with an alternative from the list below:

1. *Imagine a typical workday for this occupation. As the day unfolds, tasks arise and are completed as needed. Order the tasks in the sequence they most naturally occur.*
2. *For each task, consider its inputs and outputs. Order tasks so outputs of earlier tasks plausibly feed into later tasks. If tasks are parallel, place the more upstream task first.*
3. *Order tasks to minimize rework, waiting, and unnecessary handoffs. Assume an experienced worker executing the workflow efficiently.*
4. *Think about what must ultimately be produced in this occupation and what needs to happen before that. Use this reasoning to produce a natural forward sequence of tasks.*
5. *Identify which tasks logically depend on others, then order the tasks in a single sequence consistent with those dependencies and typical practice.*
6. *Order tasks according to how information is generated, transformed, and used over the course of the work.*
7. *Order tasks based on when mistakes would be most costly, placing tasks that prevent or constrain downstream errors earlier.*
8. *Order tasks so that tasks informing important decisions tend to occur before tasks that rely on those decisions.*
9. *Order the tasks as an experienced practitioner would intuitively carry them out, without explicitly planning or formalizing the workflow.*
10. *Order the tasks to reflect how the work is most commonly carried out in practice, rather than how it is formally described.*

Notice that these prompts are not simple re-wordings of the same instruction. Instead, they approach the task ordering problem from different perspectives, so that, in principle,

Figure SA.D.1: Kendall's τ Distribution Across GPT Prompt Task Orderings

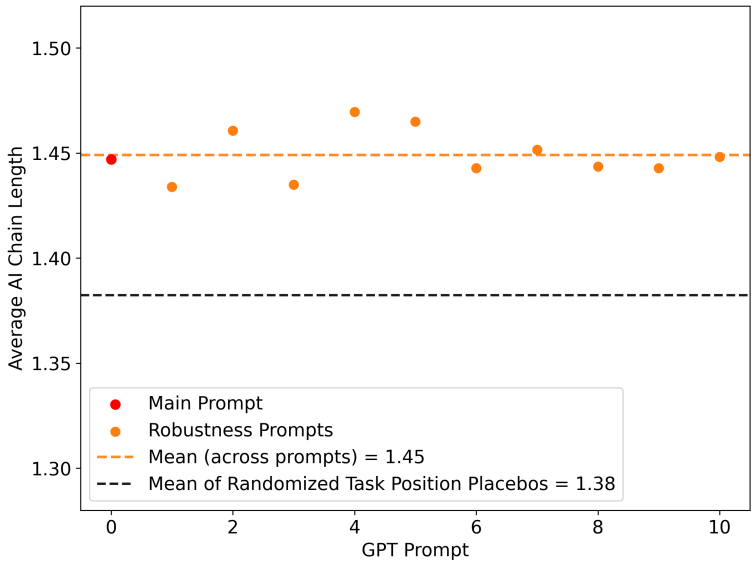


Notes: Each panel shows the distribution of mean Kendall's τ computed across 11 prompts (main + ten alternatives) within occupations. Split figures in Panels (b)–(d) compare occupations above (red) and below (blue) the median of the indicated measure, computed in the sample generated by the main prompt.

the resulting task sequences could differ. Our aim is to quantify how much variation these alternative prompt formulations induce in practice, and whether such variation poses a concern for our purposes.

To measure the degree of overlap between task orderings, we compute pairwise Kendall's τ within each occupation across all pairs of prompts. Across all occupations and prompt pairs, the average Kendall's τ is 0.6, with the full distribution shown in Panel (a) of Figure SA.D.1. This value implies that task orderings are not identical across prompts, but that there is a reasonably high degree of overlap given the diversity of prompt formulations considered. More importantly for our analyses, we find no evidence that GPT generates systematically different task orderings across different types of occupations.

Figure SA.D.2: Robustness of AI Chain Length to GPT Prompts



Notes: This figure shows that the average AI chain length measure is robust to alternative prompt formulations. The figure reports the average AI chain length computed separately under each prompt. Prompt 0 corresponds to the baseline prompt used in the main text. The dashed black reference line is the mean of the same statistic across 1,000 randomized task-position reshuffle placebos shown in Figure 9. The dashed colored line reports the mean of the statistic across all 11 prompts.

Specifically, we split occupations based on whether they fall above or below the median in the *main prompt sample* along three dimensions discussed in Subsection 7.3: the empirical fragmentation index, the share of occupation tasks exposed to AI (E1 or E2), and the share of tasks executed by AI. Panels (b)–(d) of Figure SA.D.1 show that the mean Kendall's τ is very similar for above- and below-median occupations in all cases.

In the following Subsections, we show that we obtain the same results for each of our headline analyses using these alternative prompts.

SA.D.1 Robustness of Prediction #1 Results

First, we examine how sensitive the average AI chain length statistic is to the choice of GPT prompt. Figure SA.D.2 reports the average AI chain length computed under each prompt. Across prompts, the statistic remains tightly clustered around the baseline value implied by the main prompt. In particular, the mean across all 11 prompts (the dashed colored line) matches the O*NET sample estimate from the main prompt, which lies in the extreme tail of the placebo distributions in Figure 9.

SA.D.2 Robustness of Prediction #2 Results

Second, we assess the robustness of the association between neighboring tasks' AI execution and the focal task's likelihood of AI execution.

For each alternative prompt, we re-estimate Equation (12) on the corresponding O*NET sample under four specifications: baseline, SOC major group fixed effects, SOC minor group fixed effects, and DWA fixed effects. Figure SA.D.4 reports the equivalent of average marginal effects reported in columns (1)–(4) of Table 2 for all prompts. The layout and color coding of specifications in this figure mirror those in Figure A.1 from the reshuffled task positions robustness exercise.

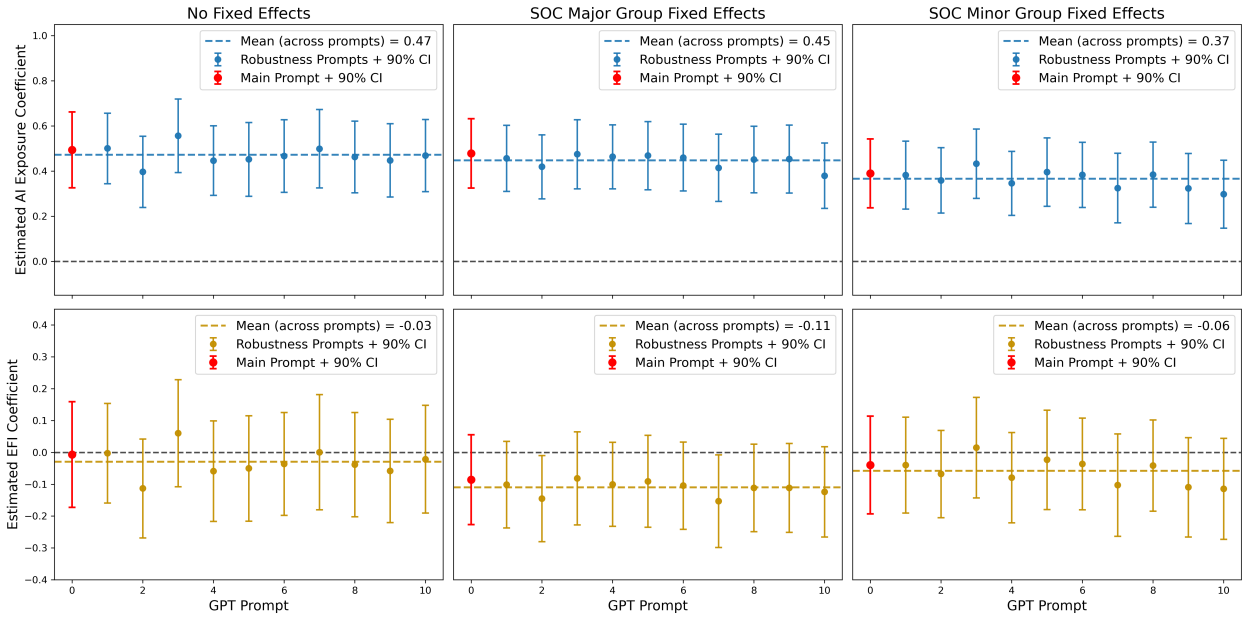
We find that even though the average effect of steps two positions away is slightly larger across alternative prompts than under the main prompt, the substantive conclusion remains unchanged. The two adjacent steps (middle columns) still have larger and more precisely estimated effects than the steps two positions away (side columns), and the latter attenuate and become less distinguishable from zero as additional fixed effects are included in Panels (b)–(d).

SA.D.3 Robustness of Prediction #3 Results

Finally, we re-estimate Equation (14) using task orderings generated by the alternative prompts and report the analogs of the estimates in Table 3.

Figure SA.D.3 plots the equivalents of estimated coefficients reported in Table 3 for the main variables across the three specifications (baseline, SOC major group fixed effects, and SOC minor group fixed effects) for the exposure-based empirical fragmentation index. Across all specifications the estimates obtained under alternative prompts fall inside the confidence interval of the main prompt estimate. The estimates on occupation-level AI exposure are positive and statistically different from zero throughout. The EFI coefficients keep the predicted negative sign in 30 of the 33 cells, but none of them is significant at the 5% level and only two clear the 10% level, mirroring the null reported in Table 3. The dispersion of the EFI coefficient across prompts is small next to the sampling noise in any one estimate, so what the prompts deliver is a stable null rather than a fragile one.

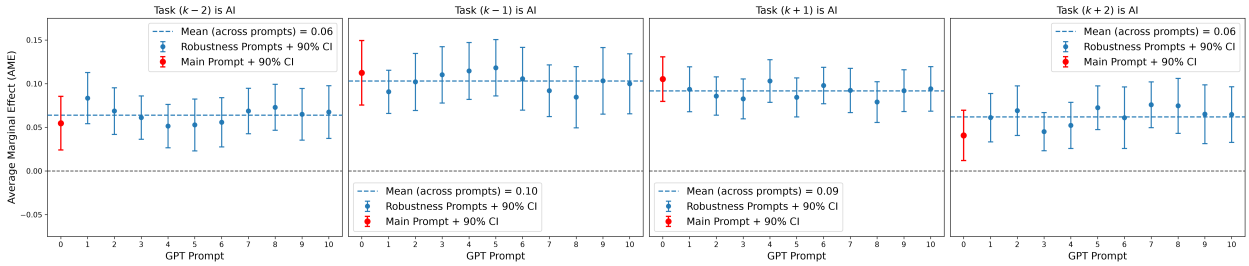
Figure SA.D.3: Robustness of Table 3 Estimates to GPT Prompts



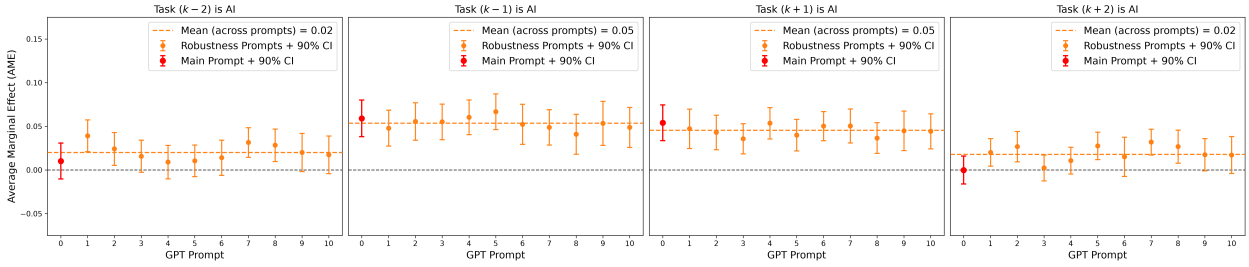
Notes: This figure shows that the estimated relationships between occupation-level AI execution and both AI exposure and the empirical fragmentation index are robust to alternative prompt formulations. The graphs report coefficients on AI exposure (top row, blue) and EFI (bottom row, gold) for the exposure-based EFI. Error bars denote 90% confidence intervals. The dashed colored lines report the mean across all 11 prompts. In each graph, the Prompt 0 estimate is highlighted in red and corresponds to the estimate reported in Table 3 using the main prompt. The graphs correspond to columns (1)–(3) of Table 3.

Figure SA.D.4: Robustness of Table 2 Estimates to GPT Prompts

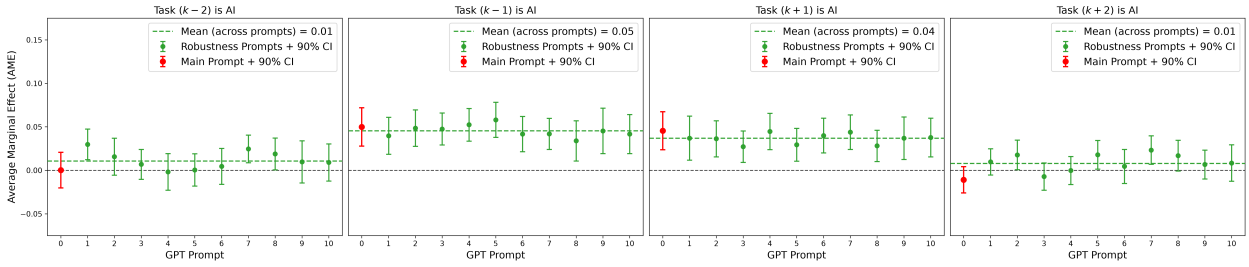
(a) No Fixed Effects



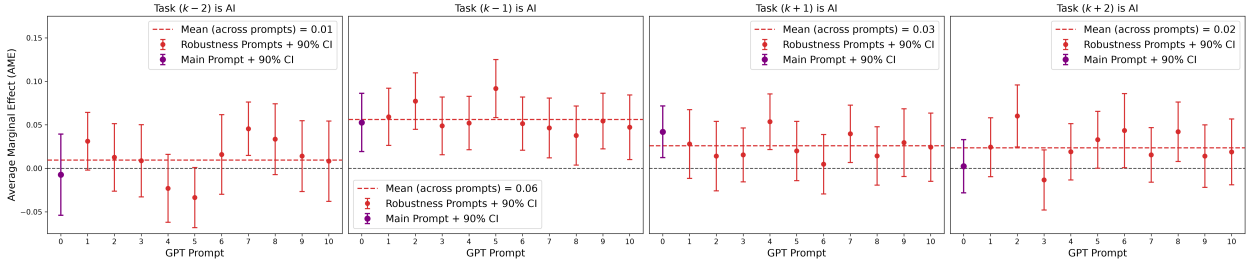
(b) SOC Major Group Fixed Effects



(c) SOC Minor Group Fixed Effects



(d) Detailed Work Activity Fixed Effects



Notes: This figure shows that the association between neighboring tasks' AI execution and the focal task's AI execution is robust to alternative prompt formulations. Across prompts, the two adjacent steps (middle columns) have larger and more precisely estimated effects than the steps two positions away (side columns). The effects of steps two positions away attenuate toward zero and become less distinguishable from zero as additional fixed effects are included. Panels (a)–(d) report the prompt-specific analogs of the estimates in columns (1)–(4) of Table 2. Error bars denote 90% confidence intervals. The dashed colored lines report the mean across all 11 prompts. In each graph, the Prompt 0 estimate corresponds to the estimate reported in Table 2 using the main prompt.

SA.E Robustness to Frequently-Executed Tasks Sample Restriction

The task sequences underlying our empirical analysis combine, for each occupation, the full set of O*NET tasks regardless of how often a task is actually performed on the job. This raises the concern that rarely-executed tasks, which a worker might perform only rarely, lengthen the workflow and insert "filler" steps between the tasks that structure day-to-day production. If they do, the average length of the AI chains we document in Prediction #1, the adjacency between AI-executed steps we exploit in Prediction #2, and the dispersion of AI-exposed steps we measure in Prediction #3 may partly reflect infrequent tasks rather than the operative structure of work. This appendix re-examines all three predictions on samples restricted to the tasks that workers report performing frequently. The chain-length and neighbor results keep their sign under pruning, though they are estimated less precisely as the samples shrink, and the fragmentation null of Subsection 7.3 is likewise unchanged.

Rather than re-querying the language model to re-sequence each occupation's surviving tasks, we prune the workflows already generated for the main analysis. Within each occupation we drop the tasks that fail the frequency filter (discussed below), keep the surviving tasks in their original order, and recompute the average AI chain length, the empirical fragmentation index, and the neighbor-execution flags on the pruned sequence before re-estimating the corresponding statistics. Pruning the existing sequences rather than re-prompting for a fresh ordering isolates the question of interest, namely whether infrequent tasks contaminate our measures, while holding the LLM's sequencing logic fixed.

To keep the sample definition consistent across the three predictions, we restrict each cut to the occupations whose frequency-pruned workflow retains at least five tasks, and we evaluate all three predictions on this common set of occupations. The five-task floor is applied to the pruned workflow, that is, to the steps that survive the frequency filter. For Predictions #1 and #3 the chain length and fragmentation index are computed over each occupation's full pruned workflow; for Prediction #2 the neighbor regression uses the eligible tasks of the same occupations, namely those with two neighbors on either side, which a five-task workflow always admits, and among these only the tasks whose DWA appears in more than one occupation, as in Section 7.2, so fewer occupations contribute to the neighbor estimates than to the other two.[46]

[46]For comparability across cuts, the fragmentation regressions z-score all three variables (the AI-execution share, the empirical fragmentation index, and AI exposure) within each cut, so the reported coefficients are standardized betas, where a one-standard-deviation change in fragmentation maps to a β-standard-deviation change in the AI-execution share. For the neighbor logits we report average marginal effects with analytic robust standard errors clustered at the DWA level, rather than the bootstrap of Table 2. Point

SA.E.1 Constructing the Frequent-Task Samples

We measure how often a task is performed using the O*NET *Frequency of Task* (FT) module. O*NET asks the incumbent workers it surveys in each occupation how often they perform each of the occupation's tasks, offering seven answer categories ordered from least to most frequent: *"yearly or less," "more than yearly," "more than monthly," "more than weekly," "daily," "several times daily,"* and *"hourly or more,"* and it reports the percentage of respondents choosing each category. A task thus comes not with a single frequency but with a distribution of workers' responses over the seven categories, so restricting the sample to frequently-performed tasks requires two choices: (1) which categories count as "frequent," and (2) how large the share of workers reporting the task in those categories must be. We keep a task when the combined share of respondents in the designated frequent categories meets a threshold.

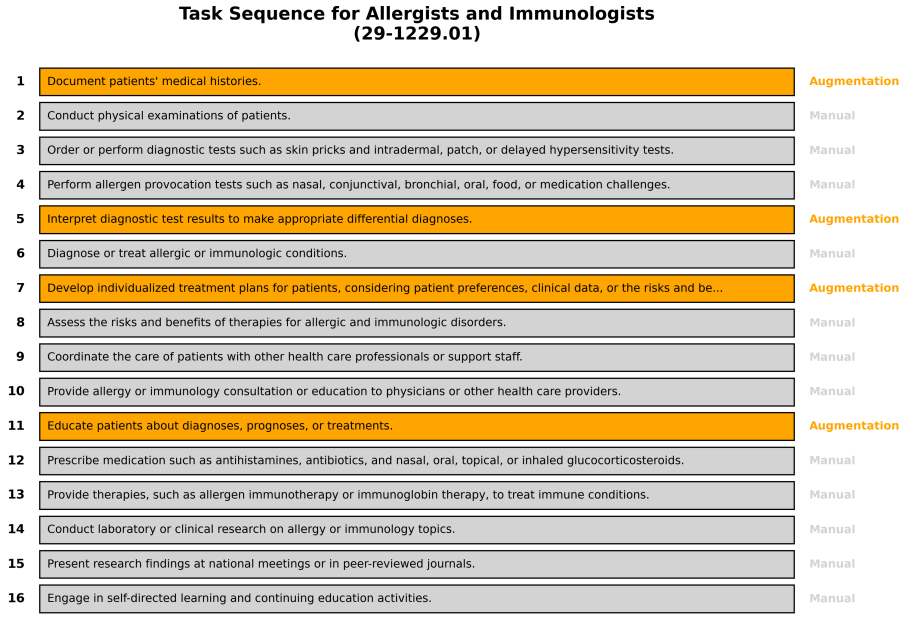
We consider three increasingly demanding notions of "frequent." *Daily+* counts tasks performed daily, several times daily, or hourly or more; *SeveralDaily+* counts those performed several times daily or hourly or more; and *Hourly+* counts only those performed hourly or more. Each notion is evaluated at four thresholds: 20%, 35%, 50%, and 65% of incumbent worker responses. A task is retained under the "Hourly+ $\geq$ 20%" cut, for example, if at least one in five workers reports performing it hourly or more. To see how the cuts relate to one another, consider a task that 60% of surveyed workers report performing several times daily, with the remaining responses spread over less frequent categories. The task survives the SeveralDaily+ cuts at the 20%, 35%, and 50% thresholds, and because responses in a stricter category also count toward every looser logic, it survives the corresponding Daily+ cuts as well. Nothing stricter retains it, in either direction. No worker reports the task hourly or more, so it fails every Hourly+ cut, and its frequent share falls short of 65%, so it fails the strictest threshold under both remaining logics. Each logic-threshold pair therefore defines its own subsample of tasks and occupations. We estimate every prediction separately on each subsample and present the results for all twelve cuts together in a grid, three logics by four thresholds, for ease of comparison.

Figure SA.E.1 makes the pruning procedure concrete for a single occupation, Allergists and Immunologists (SOC 29-1229.01). Panel (a) shows the occupation's full sequenced workflow, the object used throughout the main analysis: sixteen O*NET tasks in the order the language model places them, with the AI-executed steps shaded. Panel (b) shows what survives the Hourly+ $\geq$ 50% cut, one of the most demanding in the grid: seven of

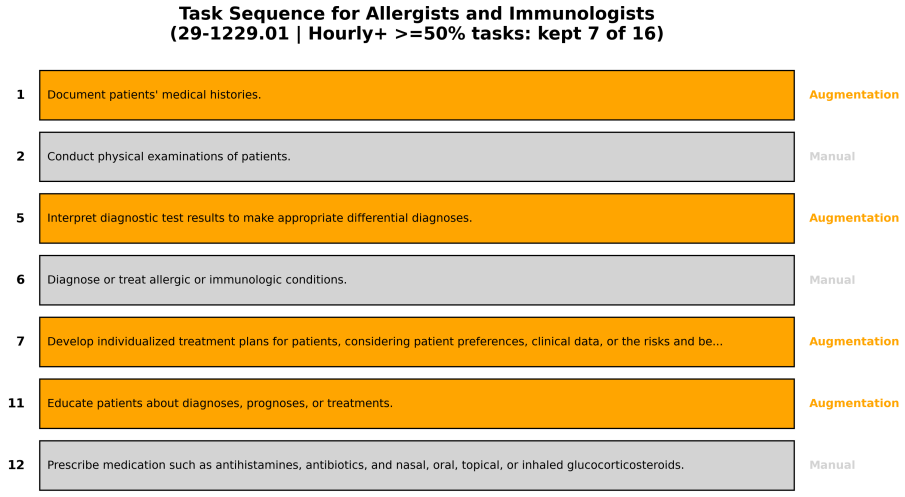
estimates for the "all tasks" specification of Predictions #1 and #2 are identical to those reported in the main text. For Prediction #3, the five-task floor drops one of the 872 occupations, which moves the SOC minor-group estimate from -0.04 to -0.05.

Figure SA.E.1: Frequency Pruning of a Single Occupation's Workflow: Allergists and Immunologists

(a) Full Workflow (All Tasks)



(b) Pruned Workflow (Hourly+ $\geq$ 50%)



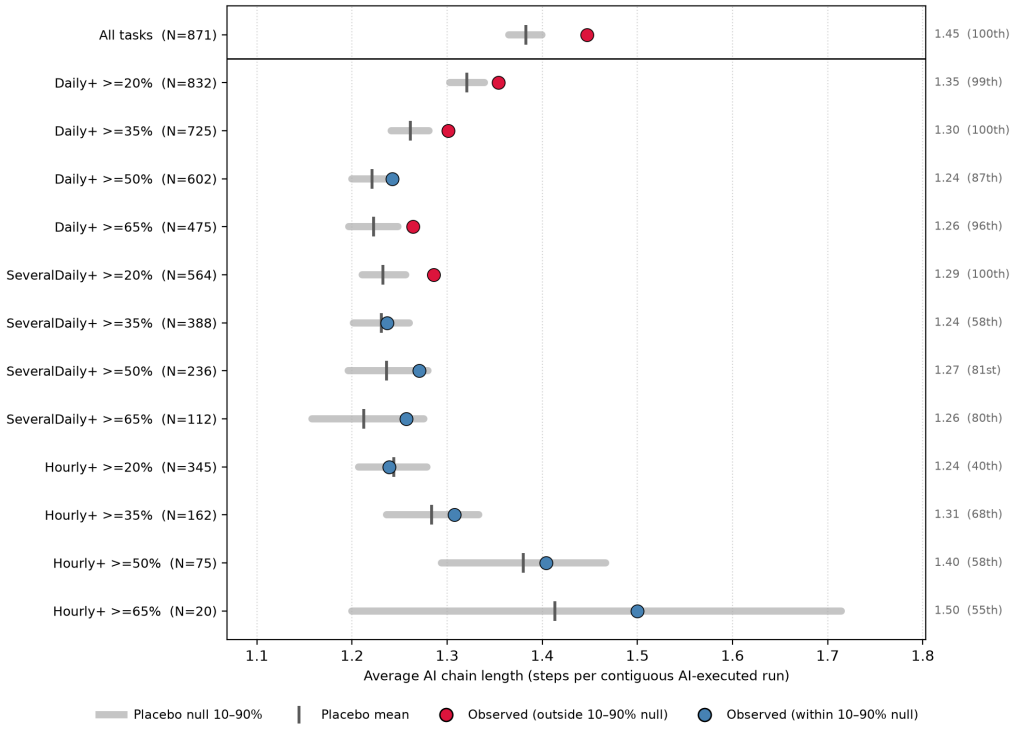
Notes: Panel (a) reports the full sequenced workflow for Allergists and Immunologists (SOC 29-1229.01), the object used in the main analysis: the occupation's sixteen O*NET tasks in the order assigned by the language model. Panel (b) reports the workflow that survives the Hourly+ $\geq$ 50% frequency cut, which retains a task only if at least half of the surveyed incumbent workers report performing it hourly or more. Surviving tasks keep their original relative order and their position in the full workflow, so the numbers skipped in Panel (b) are the pruned steps. Shaded rows are AI-executed steps (augmentation or automation in the Anthropic Economic Index); unshaded rows are manual. Pruning keeps all four AI-executed steps but drops the manual steps separating positions 7 and 11, merging two isolated AI-executed steps into a chain of length two.

the sixteen tasks, retained in their original relative order and still carrying their original position numbers so that the dropped steps can be read off directly. What the filter removes is what one would expect it to remove. The specialized diagnostic procedures (steps 3 and 4), the coordination and consultation activities (steps 9 and 10), and the research and continuing-education tasks (steps 14 through 16) are performed episodically rather than within the hour, and they drop out; what remains is the recurring clinical loop of documenting histories, examining patients, interpreting results, planning treatment, educating patients, and prescribing. The figure also shows why pruning is not innocuous for our measures, which is precisely the concern this appendix addresses. All four AI-executed steps survive the filter, but in the full workflow each is isolated between manual steps, so the occupation contributes four chains of length one and an average AI chain length of 1.00; once the intervening manual steps 8 through 10 are dropped, steps 7 and 11 become adjacent and the same four AI-executed tasks now form three chains of lengths 1, 1, and 2, raising the occupation's average to 1.33. Pruning can thus both shorten workflows and concentrate the surviving AI-executed steps, and the exercises that follow ask whether our three results survive once this is done for every occupation. We take the three predictions in turn.

SA.E.2 Robustness of Prediction #1 Results

Prediction #1 concerns the formation of AI chains. We measure it, as in Section 7.1, by the average length of a maximal run of contiguous AI-executed steps, where a step is AI-executed if it is labeled augmented or automated, pooled across all chains in the sample. A longer-than-random average chain could, however, be delivered mechanically by the composition of each pruned sample. We therefore benchmark every cut against a within-occupation task position-reshuffle placebo, where we redraw 1,000 reshuffles of task positions within occupations, re-apply the frequency filter, and recompute the statistic, producing a null distribution that holds the composition of the sample fixed while destroying the workflow ordering. Figure SA.E.2 reports the observed average against this null as a forest plot: each row is a frequency cut, with the "all tasks" baseline analysis in the main manuscript on top; the grey bar spans the 10th to 90th percentiles of the reshuffle null, the vertical tick marks the null mean, and the dot is the observed estimate, red when it falls outside the null band and blue when inside, with the observed percentile printed in the right margin. The top all-tasks row reproduces the full-sample value of 1.45. Chains shorten modestly as the sample is restricted to frequent tasks, to between 1.24 and 1.35 in every cut except the two sparsest Hourly+ corners, where they tick back up (1.40 and 1.50, with 75 and 20 occupations). In the full sample the observed value sits

Figure SA.E.2: Average AI Chain Length Versus Its Task Position-Reshuffle Null Across Frequency Cuts



Notes: Each row is a frequency cut, with the "All tasks" baseline from the main manuscript on top. The left labels report the number of occupations in each cut. The dot is the observed average AI chain length for that cut: the mean length of a maximal run of contiguous AI-executed (augmented or automated) steps. The grey bar spans the 10th to 90th percentiles of the within-occupation task position-reshuffle null (1,000 draws), and the vertical tick marks the null mean. The dot is red if it falls outside the null band and blue if inside. The right margin reports the observed value and its percentile in the null.

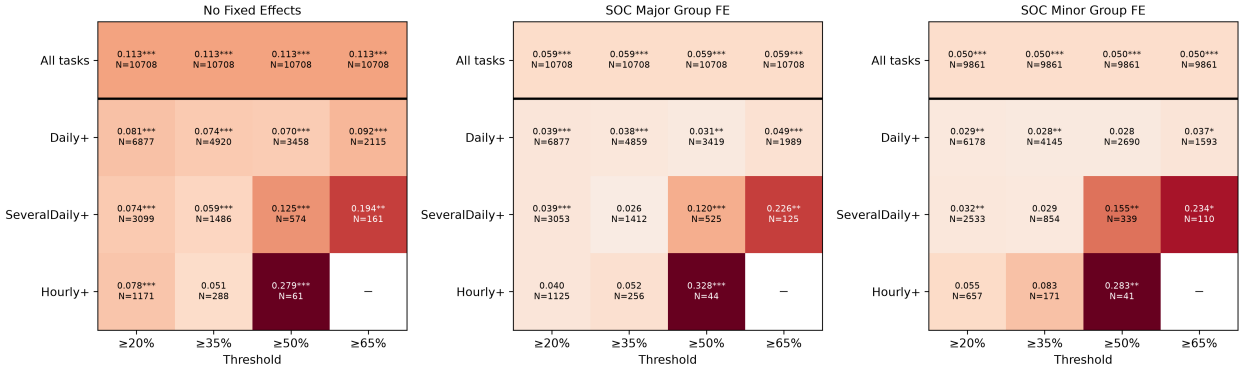
at the 100th percentile of its null. AI-executed tasks form longer contiguous chains than random reordering would produce, so the chaining pattern is not a compositional artifact. Pruning preserves the direction of the result but not its precision. The observed chain length lies above its null mean in eleven of the twelve pruned cuts and clears the 10-to-90 null band in four of them, all under the inclusive logics, namely Daily+ at the 20%, 35%, and 65% thresholds (the 99th, 100th, and 96th percentiles) and SeveralDaily+ at 20% (the 100th). In the remaining cuts it sits inside the band, at the 58th to 87th percentiles under Daily+ and SeveralDaily+ and at the 40th to 68th under Hourly+, where the samples are smallest and the null bands widest.

SA.E.3 Robustness of Prediction #2 Results

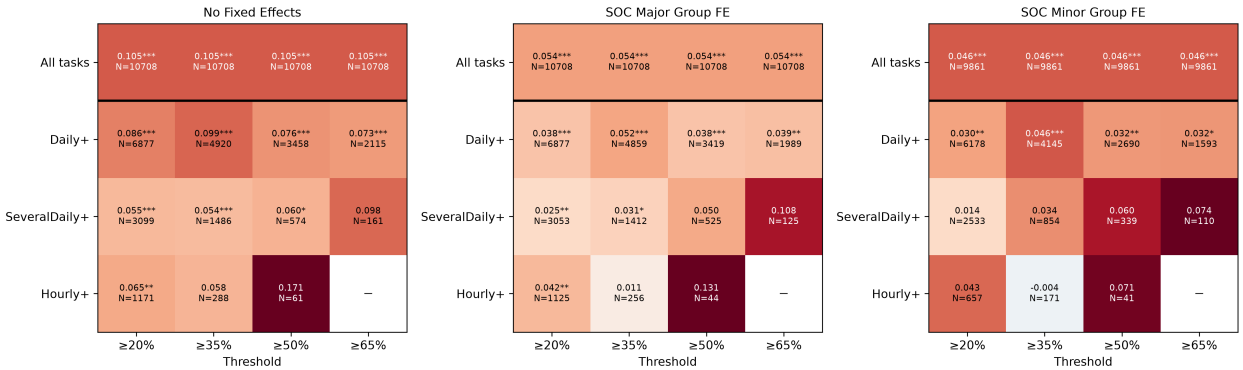
Figure SA.E.3 repeats the exercise for the neighbor prediction. Each cell reports the average marginal effect, from re-estimating Equation (12), of having the immediately preceding ($k-1$, Panel (a)) or following ($k+1$, Panel (b)) task be AI-executed on the focal

Figure SA.E.3: Neighbor AI Execution and Focal-Task AI Execution Across Frequency Cuts

(a) Previous Task ($k-1$)



(b) Next Task ($k+1$)



Notes: Each cell reports the average marginal effect of having the immediately previous task ($k-1$, Panel (a)) or next task ($k+1$, Panel (b)) be AI-executed on the focal task's probability of AI execution, from re-estimating Equation (12) on the corresponding frequency cut. The heatmap layout is as in Figure SA.E.5, except that the number beneath each estimate is the number of task observations and the top "All tasks" row is the unpruned specification of Section 7.2. Standard errors are clustered at the DWA level, and significance stars come from the clustered coefficient test. *** p<0.01, ** p<0.05, * p<0.1. Red denotes a positive effect (the predicted sign) and blue a negative one. A "–" marks a cut with too few observations to estimate.

task's probability of AI execution. A red cell denotes a positive estimate, the sign found in the full sample. The top row reproduces the full-sample adjacent-step estimates of +0.11 for both the previous and the next task under no fixed effects, which attenuate to between +0.05 and +0.06 once SOC fixed effects absorb cross-family differences. The estimate is positive in 65 of the 66 estimable pruned cells. Under the inclusive Daily+ logic it is somewhat smaller than in the full sample, at +0.07 to +0.10 without fixed effects and +0.03 to +0.05 with SOC fixed effects, and it remains significant in 23 of the 24 cells. Under the stricter SeveralDaily+ and Hourly+ logics the estimates are noisier. The previous-task estimate is largest in the sparsest cuts (SeveralDaily+ $\geq$ 50% and $\geq$ 65%, and Hourly+ $\geq$ 50%, with 41 to 574 task observations), at +0.12 to +0.33 and significant

throughout, while the next-task estimate in the same cells ranges from +0.05 to +0.17 and is mostly insignificant. Outside those cuts, significance is lost mainly in the intermediate SeveralDaily+ and Hourly+ cuts, and most often with SOC minor-group fixed effects. The effects of steps two positions away remain small and, under fixed effects, are statistically indistinguishable from zero across the grid, in line with the main text. The part of a step's local context that predicts its execution mode is the part a chain of average length 1.45 can actually span.[47]

Figure SA.E.4 benchmarks the previous-task ($k-1$, Panel (a)) and next-task ($k+1$, Panel (b)) effects against their position-reshuffle nulls, one column per specification. Both estimates sit far in the upper tail of their nulls in the full sample, at the 100th percentile in all three specifications, and mostly stay there under the Daily+ logic. The previous-task estimate lies outside its 10-to-90 null band in ten of the twelve Daily+ cells and the next-task estimate in nine, at the 87th to 97th and the 77th to 100th percentiles across those cells, respectively. Under the stricter logics the two separate. The previous-task estimate still escapes its null in seven of the twelve SeveralDaily+ cells, including all three at the $\geq$ 50% threshold, but in only one of the nine Hourly+ cells, while the next-task estimate stays inside its null band in every SeveralDaily+ and Hourly+ cell, at the 28th to 86th percentiles. The positive association between adjacent-step AI execution and the focal step's AI execution thus persists under the inclusive cuts and loses precision, but rarely its sign, under the strictest thresholds, where the surviving task observations number in the low hundreds and the reshuffle bands are correspondingly wide.

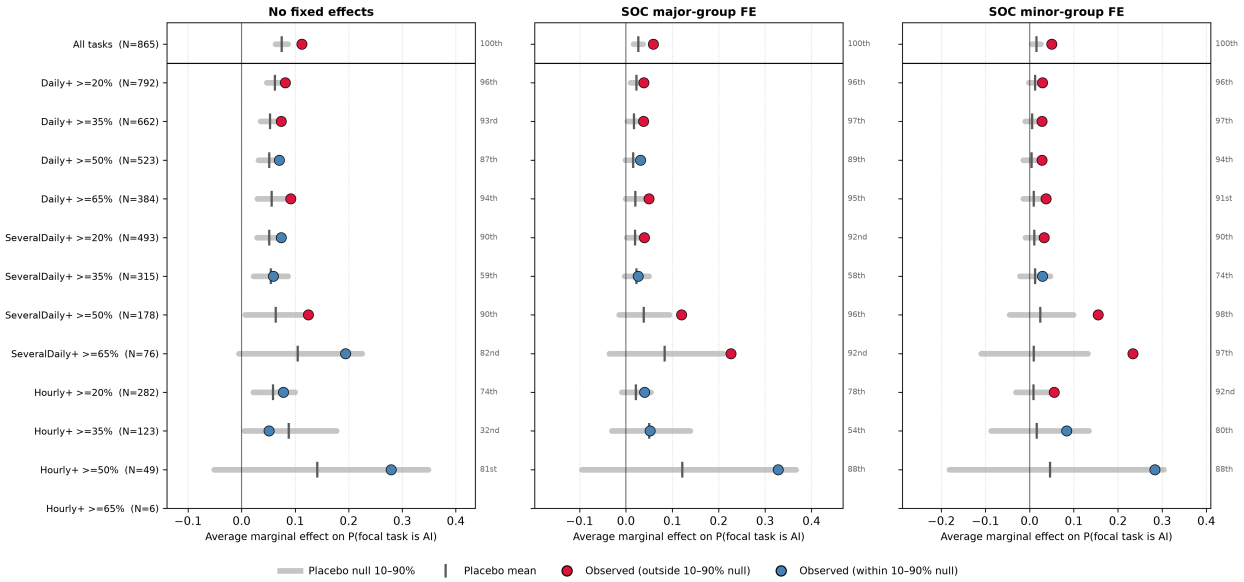
SA.E.4 Robustness of Prediction #3 Results

Figure SA.E.5 summarizes the fragmentation regressions as a heatmap over the logic-by-threshold grid, with the unpruned "all tasks" specification from the main analysis in the top row, so that the effect of progressively dropping infrequent tasks can be read down the rows, which impose a more demanding frequency logic, and across the columns, which raise the threshold. Each cell reports the standardized coefficient on the empirical fragmentation index from re-estimating Equation (14) for the corresponding cut, with the number of occupations printed beneath (no cell is significant at the 10% level, so no stars appear); as in the main text, every specification controls for the number of AI-exposed steps in the occupation. We use our preferred exposure-based measure throughout (as defined in Section 7.3, under which both E1- and E2-exposed tasks may form AI chains), and the three heatmaps correspond to the no-fixed-effects, SOC major-group, and SOC

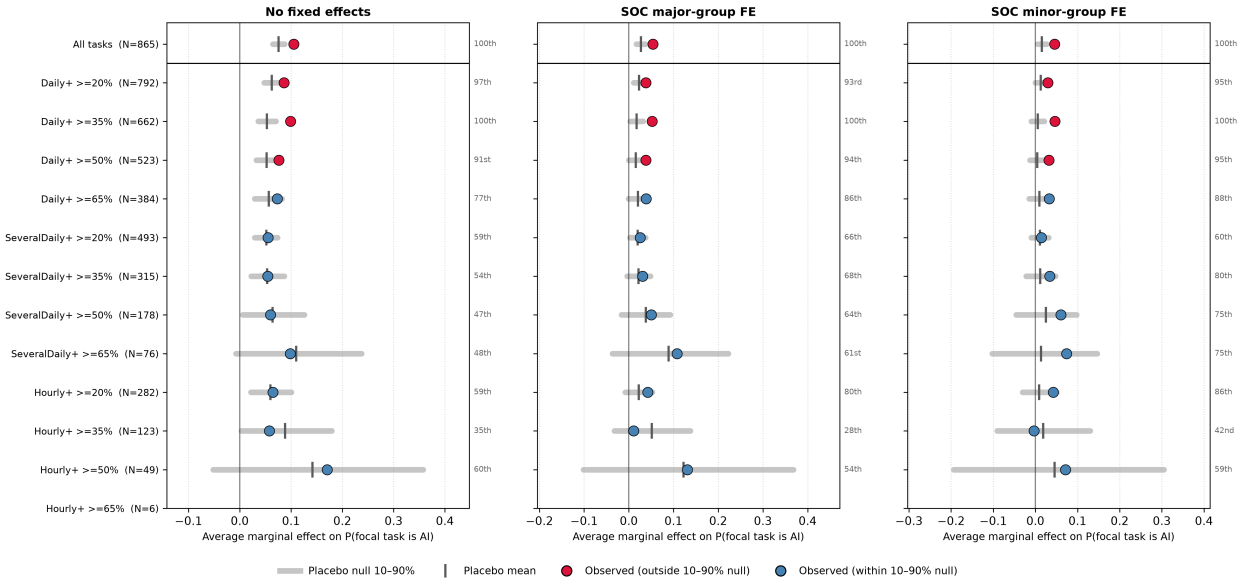
[47] To save space, we do not report the results on steps two position away from the focal step and only focus on the outcomes for immediate neighbors.

Figure SA.E.4: Immediate-Neighbor Associations: Observed Average Marginal Effects Versus Their Position-Reshuffle Nulls Across Frequency Cuts

(a) Previous Task ($k-1$)



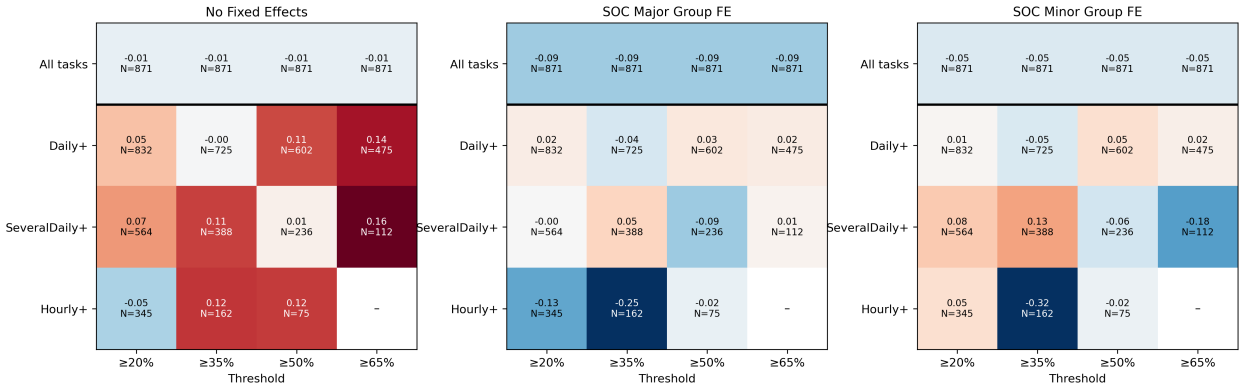
(b) Next Task ($k+1$)



Notes: Panel (a) reports the average marginal effect of having the previous task ($k-1$) be AI-executed, and Panel (b) the corresponding effect for the next task ($k+1$). Rows, the null band, the dot coloring, and the margin labels are as in Figure SA.E.2, with the dot now the observed average marginal effect; the three columns within each panel are the no-fixed-effects, SOC major-group, and SOC minor-group specifications. A blank row denotes a cut with too few neighbored task observations to estimate.

minor-group specifications. A blue cell denotes a negative coefficient, the sign the model predicts, and a red cell a positive one; darker shading indicates a larger magnitude. The

Figure SA.E.5: Empirical Fragmentation Index and AI Execution Across Frequency Cuts



Notes: Each cell reports the standardized coefficient on the empirical fragmentation index from re-estimating Equation (14) on the corresponding frequency cut. The index is the preferred exposure-based measure. The number of occupations appears beneath each estimate. No cell in the grid is significant at the 10% level, so no significance stars are shown. Following the neighbor heatmaps of Subsection SA.E.3, the Hourly+ and ≥ 65% cut is left blank, since the 20 occupations it retains are too few for this specification to estimate. The three heatmaps report the no-fixed-effects, SOC major-group, and SOC minor-group specifications. Within each heatmap, rows correspond to the three frequency logics and columns to the four thresholds. The top "All tasks" row is the unpruned specification of Section 7.3. Blue denotes a negative coefficient (the predicted sign) and red a positive one. Each panel is scaled to its own largest absolute coefficient, so shading is comparable within a panel but not across panels.

top all-tasks row gives the full-sample standardized estimates of −0.01, −0.09, and −0.05, none of them distinguishable from zero, here estimated on the 871 occupations that retain at least five tasks and closely matching the headline estimates of Table 3. Pruning does not recover the relationship. Across the 33 pruned cells the coefficient is negative in 13, the median absolute estimate is 0.05, and no cell in the grid reaches significance at the 10% level under any of the three specifications. We do not read the pruned cells as ruling the channel out either, since 20 of the 33 retain confidence intervals wide enough to admit a moderate negative effect. What the grid establishes is that the null of Subsection 7.3 is a feature of the O*NET sample rather than an artifact of counting rarely-performed tasks as steps.

In short, the chain-length and neighbor patterns keep their sign among frequently performed tasks, and neither they nor the fragmentation null of the O*NET sample appear to be artifacts of the rarely performed activities catalogued in O*NET.

SA.F External Validation of the Sequencing Approach and Results

The O*NET task sequences underlying our empirical analysis are not observed. O*NET catalogues the tasks that belong to an occupation but not the order in which they are performed, so we construct that order by prompting a language model (GPT-5-mini). Every adjacency-based object we compute on the O*NET sample rests on that imputation, which raises two concerns.

The first is that the task orderings may be artifacts of how LLMs reason rather than reconstructions of how work is done. An LLM asked to arrange a list of tasks may place semantically similar items next to one another, encoding topical similarity instead of production logic. Since AI exposure and execution are themselves correlated with what a task is about, such an ordering could manufacture the contiguity we document without any underlying sequence and contaminate our other exercises. Appendix SA.D showed our orderings are stable across prompts, but stability is not accuracy, since an instrument can be consistently wrong.

The second is that not all O*NET tasks may be performed as part of a single workflow sequence. O*NET is a catalogue of what an occupation involves, not a process map, and if its task lists do not describe work that proceeds in an order, no ordering procedure, however accurate, could deliver the objects our predictions require. Appendix SA.E pruned the O*NET list of tasks to those that are highly frequently performed, which makes the resulting task sequences more workflow-like, but the caveat still applies.

This appendix addresses both concerns with external data whose work sequences a language model did not impute. Two corpora serve. The first is the *Process Classification Framework* (PCF) of the American Productivity & Quality Center (APQC), whose member working groups have recorded since 2004 the activities and tasks that make up a firm's business processes, in the order practitioners perform them (APQC, 2026). The second is a collection of process-mining event logs from the 4TU.ResearchData repository (de Leoni and Mannhardt, 2015; Mannhardt, 2016; van Dongen, 2017, 2020), which record case by case the activities organizations actually executed and when, so that the order is observed rather than asserted.

The bulk of the appendix establishes that the LLM can recover an ordering of activities at all, against corpora whose order is already known. Appendix SA.F.1.1 shuffles the steps of 345 documented process branches and asks the LLM to order them in a sequence, under the main analysis's prompt and each of its ten alternatives. It recovers the practitioners' order to a substantial degree, and does so despite the presence of branches whose activities are mutually exclusive options for which no true order exists. Appendix SA.F.1.2 repeats

the exercise against the event logs, where the data reveal which precedences are real and the comparison can be restricted to them. The two benchmarks agree closely about how much sequence structure the instrument recovers.

Subsection SA.F.2 then turns the PCF corpus on the second concern. It documents how AI exposure and AI execution labels are carried onto PCF steps with natural-language embeddings, which is the construction behind columns (4)–(6) of Table 3, where the PCF estimates of the fragmentation channel of Prediction #3 are reported alongside the O*NET sample and discussed in Subsection 7.3. It also gives in full the Prediction #1 exercise on those documented sequences that Subsection 7.1 summarizes, where AI-executed steps again form chains longer than chance allows. Prediction #2 is the one test this corpus cannot carry, for the reason given in Subsection 7.2 and with the counts reported below.

Taken together, the exercises conducted in this appendix indicate that our findings are artifacts neither of how LLMs order a list of tasks in a sequence nor of how O*NET catalogues tasks in occupations.

SA.F.1 Concern #1: LLM Recovering Real-World Task Orderings

We benchmark the sequencing procedure twice, against two corpora that differ in where their order comes from. The first records the order practitioners say they work in; the second records the order in which work was actually carried out.

SA.F.1.1 Benchmark #1: Practitioner-Documented Orderings (PCF)

We apply our task-ordering approach, exactly as used in the main analysis, to the APQC corpus, and measure how much of the order practitioners recorded it recovers. The benchmark is the *Process Classification Framework* (PCF) of APQC in its Cross-Industry version 8.0. The PCF decomposes work into five nested levels, from 13 Categories through 74 Process Groups, 362 Processes, 1,353 Activities, and 215 Tasks, for 2,017 process elements in total, each with a persistent identifier, a name, and a short description. What makes it useful here is that within a parent element the children are listed in the order in which they are performed whenever applicable. Ordering steps in "Order materials and services," for instance, runs from *"process/review requisitions"* through *"approve requisitions," "solicit supplier quotes," "create/distribute purchase orders," "expedite orders and satisfy inquiries," "reconcile purchase orders,"* and finally *"research/resolve order exceptions."* APQC also publishes industry-specific versions of the framework, which Subsection SA.F.2 draws on; Table SA.F.1 shows one documented process group from each of two of them.[48]

[48]Our results are not particular to the cross-industry framework. While there is some degree of heterogeneity in how the LLM recovers sequences from business processes in different industries, the magnitudes of the

Table SA.F.1: Example Process Groups From Two Industry Frameworks

Step	**Aerospace and Defense** Manage product recalls and regulatory audits	**Automotive** Manage parts
1	Initiate recall	Manage inventory after sale
2	Assess the likelihood and consequences of occurrence of any hazards	Manage electronic parts catalog
3	Manage recall related communications	Exchange parts/locate parts
4	Submit regulatory reports	Manage returns
5	Monitor and audit recall effectiveness	Rebuild part
6	Manage recall termination	Manage parts retail operations

Process elements appear in the order APQC's working groups document them.

Notes: Each column shows one PCF Process Group with its constituent process elements, in the order APQC documents them. The groups are Hierarchy ID 6.4 of the Aerospace and Defense framework (PCF ID 20110) and Hierarchy ID 6.11 of the Automotive framework (PCF ID 12685).

We take as our unit of analysis a *process branch*: a parent element together with its ordered children. Restricting attention to branches with at least three children leaves 345 branches covering 1,917 process elements, with a median of five steps per branch. Relative to O*NET, the corpus is small and is weighted toward enterprise process work such as finance, procurement, human resources, information technology, and supply chain.

Practitioner orderings are not immutable ground truth. Before using APQC's listings as a benchmark, we register a caveat about what they are. The PCF is a documentation convention produced by committees, not a record of observed execution, and for some branches the listed order is one defensible arrangement among several rather than a statement about precedence.

Two examples make the point. The branch "Analyze deal options" lists four children: *"evaluate acquisition options,"* *"evaluate merger options,"* *"evaluate de-merger options,"* and *"evaluate divestiture options."* These are mutually exclusive alternatives that a firm considers not necessarily in the presented order. Reversing the order, from divestiture to acquisition, would be an equally coherent way to write the same branch down. Similarly, "Process accounts payable and expense reimbursements" lists *"process accounts payable,"* then *"process expense reimbursements,"* then *"manage corporate credit cards."* These are three distinct disbursement streams that can be run concurrently, and nothing in the process dictates which is documented first. For branches of this kind any ordering an LLM produces will disagree with APQC roughly half the time by construction, and disagreement carries no information about whether the LLM understands sequence.

We make no attempt to purge such branches from the sample, since doing so would require the very judgment about true sequence that the exercise is meant to test. We proceed with the full set of branches, which makes the measured overlap a conservative

difference are not large and we do not report them in detail here for brevity.

Table SA.F.2: Overlap Between GPT-Generated and Practitioner-Documented Process Orderings

Mean Kendall τ against APQC's published process order	
Main prompt (Prompt 0)	+0.53
Average of 10 alternative prompts	+0.51
Other overlap measures	
Adjacent-pair agreement, main prompt (null 0.5)	0.70
Adjacent-pair agreement, 10-prompt average	0.70
Exact-order recovery, main prompt	0.24
Share of branches with $\tau > 0$, main prompt	0.86
Share of branches with $\tau > 0$, 10-prompt average	0.90
Process branches	345
Process elements ordered	1,917

Notes: This table reports the overlap between the orderings GPT-5-mini produces for APQC process branches and the orderings APQC's practitioner working groups published, pooled across all 345 branches. "Adjacent-pair agreement" is the share of APQC-adjacent step pairs the model places in the correct direction, with a null value of 0.5. "Exact-order recovery" is the share of branches whose full published sequence the model reproduces.

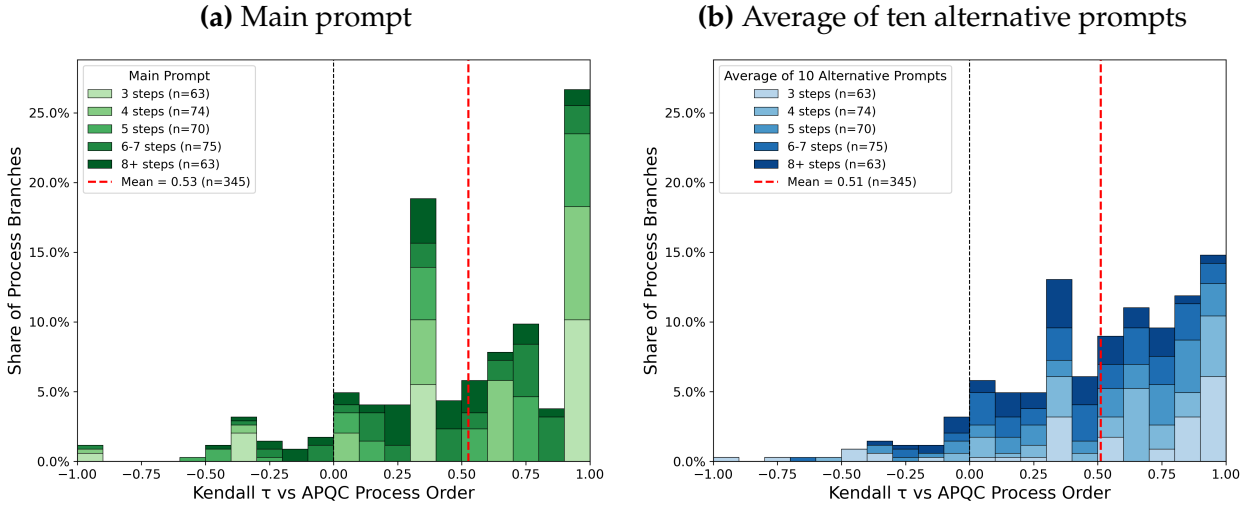
one.

Design. For each of the 345 branches we take APQC's published child order as the target, shuffle the children under a fixed seed so that the LLM never observes the published sequence, and ask it to order them. The prompt is the one used to sequence O*NET tasks in the main analysis (Prompt #1 of Appendix SA.C), carried over verbatim with a single mechanical substitution that maps the occupation domain onto the process domain, replacing "occupation" with "business process" and "task" with "step." We repeat the exercise for each of the ten alternative prompts of Appendix SA.D, applying the identical substitution to all eleven so that no prompt is privileged by hand-tuning, and we use the same model and settings as in the main analysis (GPT-5-mini, temperature 0).

We measure overlap between the LLM's ordering and APQC's with Kendall's τ, computed within each branch and averaged across branches. We also report the share of APQC-adjacent step pairs that the model places in the correct direction, whose null value is 0.5, and the share of branches whose full sequence the model reproduces exactly. Statistical significance is assessed against a permutation null of 1,000 random orderings per branch, which centers τ at zero. Of the 3,795 orderings requested, 3,793 returned a complete and parseable permutation of the branch.

Results. Table SA.F.2 reports the main results and Figure SA.F.1 the underlying distributions. Under the main prompt the average Kendall's τ between the LLM's ordering and APQC's is 0.53, with a median of 0.60; 86% of branches have positive τ. The LLM places 70% of APQC-adjacent step pairs in the correct direction against a null of 50%, and

Figure SA.F.1: Overlap between GPT-orderings and Original APQC-PCF Order by Branch Length



Notes: Each panel plots the distribution across the 345 APQC process branches of the within-branch Kendall's τ between GPT-5-mini's ordering and APQC's published ordering, with each bin decomposed by the number of steps in the branch; darker shading denotes longer branches. Panel (a) uses the main prompt and panel (b) averages, within each branch, the τ obtained under the ten alternative prompts of Appendix SA.D. The red dashed line marks the sample mean and the black dashed line the value a random ordering would deliver in expectation. Kendall's τ is coarse on short branches, which is why the extreme bins at both ends are dominated by branches of three and four steps.

reproduces the practitioner sequence *exactly* in 24% of branches. The last of these is the most demanding, because exact recovery by chance is one in 120 for a five-step branch and one in 40,320 for an eight-step branch. Against the permutation null the average τ corresponds to $z = 22.6$ ($p = 0.001$). The LLM is not reproducing the order in which the steps were shown to it. The average τ between its output and the shuffled input list is 0.11.

Running the ten alternative prompts leaves this essentially unchanged. Prompt-specific averages span a narrow range, from 0.49 to 0.53, so the result is not an artifact of the particular phrasing used in the main prompt. Averaging the ten alternatives within each branch, which removes the sampling noise of any single formulation, leaves the mean at 0.51 while raising the share of branches with positive τ from 86% to 90% and lowering the cross-branch standard deviation from 0.42 to 0.37; twelve of the branches that score negative under the main prompt turn positive once averaged. Prompt choice therefore affects individual branches but not the aggregate picture.

Overlap is higher where the benchmark is a stronger claim about sequence. Across PCF categories the average τ ranges from 0.66 in "Deliver Services," 0.69 in "Manage Customer Service," and 0.65 in "Manage Supply Chain for Physical Products," which are transactional and genuinely sequential, down to 0.25 in "Develop and Manage Products and Services" and 0.27 in "Manage External Relationships," whose children are largely

mutually exclusive options of the kind discussed above.

Reading the left tail. A tenth of branches record a negative τ under the main prompt, and it is worth being clear about what they are, since the sign of τ overstates the disagreement on short branches. Kendall's τ is coarse when there are few steps to order. On a three-step branch it can only take the values $\{-1, -1/3, +1/3, +1\}$, so a single inverted pair already registers as a large negative number, whereas on a ten-step branch the same inversion costs 0.02 and leaves τ near +1. Figure SA.F.1 decomposes the distribution by branch length and shows the consequence directly. The mass at $\tau = -1$ consists almost entirely of three- and four-step branches (three of its four branches), and so does most of the spike at $\tau = +1$, where 56% of three-step branches sit against 10% of the branches with five steps or more.

The tail is not, however, purely an artifact of short branches. Negative values are about as common among the longest branches as among the shortest, 14% in both cases, and what differs is their magnitude, in that short branches produce reversals while long branches produce values between -0.03 and -0.20, that is, a handful of misordered pairs within an otherwise correct sequence. Inspecting the branches that remain negative after averaging over all ten alternative prompts, the recurring case is the one anticipated above: parents whose children are parallel streams or mutually exclusive alternatives, for which no true ordering exists and the sign of τ is close to a coin flip.

SA.F.1.2 Benchmark #2: Observed Process Execution (4TU.ResearchData)

The benchmark above has a limitation it cannot overcome. A documented ordering is an editorial convention, so where a process has no true sequence, disagreement says nothing about the instrument. Moving from documented to *observed* order removes it.

A process-mining event log records, case by case, the activities an organization actually executed and when. Two things follow. First, the ordering is a fact about production rather than an assertion about it. Second, and more useful here, the data reveal *which precedences are real*. If activity x precedes y in 97% of cases the precedence is genuine, whereas a 50/50 split means the two run concurrently and no ordering could be correct. We can therefore condition on the pairs that have a true order before asking whether the model recovers it, which is the step the documented benchmark does not permit.

These logs record processes carried out by several people, with the case passing between them, and that property is what makes them the right benchmark here rather than the wrong one. Read against the short run alone it would be a mismatch, since Section 3.3 assigns the whole sequence to a single worker. But the long-run problem of Section 5.3

is about exactly this object: a workflow partitioned into jobs, with a hand-off incurred wherever the work passes from one worker to the next. What the model holds fixed across both horizons is the order of the steps, not who performs them, so an event log observes the sequence our imputation is meant to recover, at the level of aggregation the long run works with.

Data. We use eight publicly available event logs from the 4TU.ResearchData repository (de Leoni and Mannhardt, 2015; Mannhardt, 2016; van Dongen, 2017, 2020), listed in Table SA.F.3: fine collection by a local police force, sepsis care in a hospital emergency department, consumer lending at a financial institution, and five university travel and reimbursement processes. Together they cover 215,715 cases and 2,048,949 recorded events. Their activity sets are considerably larger than the APQC process branches, between 8 and 28 activities against a median of five, so they also test the instrument on long sequences.

From each log we recover one trace per case, the sequence of activity labels in timestamp order, and keep the activities appearing in at least 2% of cases so that the target is not driven by rare exceptions. We use each activity's *first* occurrence within a case, which handles the loops present in every real log such as repeated laboratory tests or repeated approval rounds. This yields the two objects we need. The *observed ordering* ranks activities by their mean normalized first-occurrence position across cases; this is the analogue of APQC's published order. The *pairwise precedence matrix* records, for every pair, the share of cases containing both in which one precedes the other. We call a pair *determinate* when one direction holds in at least 80% of cases. Across the eight logs, 1,124 of 1,178 activity pairs (95.4%) are determinate, so the benchmark is a statement about sequence for almost all of the comparisons we make.

Design. The procedure mirrors that of Appendix SA.F.1.1. For each log we shuffle the activity labels under a fixed seed, so the model never sees the observed sequence, and ask it to order them, supplying a one-line description of the process in place of an occupation title. We use all eleven prompts, the main specification of Appendix SA.C and the ten alternatives of Appendix SA.D, carried over with the same mechanical substitution that maps the occupation domain onto the process domain. No log is told which activity starts or ends a case.

We report two levels of analysis. At the *log level* we compute Kendall's τ between the model's ordering and the observed ordering, which is directly comparable to the APQC numbers, benchmarked against a permutation null of 1,000 random orderings per log. At the *pair level* we ask only whether the model places a pair in the direction the traces

Table SA.F.3: Event Logs Used for Validation

Log	Process	Cases	Events	Activities	Determinate pairs
Road traffic fines	Fine management, local police	150,370	561,470	8	27 of 28
Sepsis	Emergency department sepsis care	1,050	15,214	15	91 of 98
BPI 2017	Loan application handling	31,509	1,202,267	23	238 of 245
BPI 2020	Domestic travel declarations	10,366	56,303	10	41 of 44
BPI 2020	International travel declarations	6,449	72,151	22	218 of 224
BPI 2020	Requests for payment	6,814	36,724	10	39 of 42
BPI 2020	Prepaid travel costs	2,092	18,239	17	128 of 129
BPI 2020	Travel permits and declarations	7,065	86,581	28	342 of 368
Total		215,715	2,048,949	133	1,124 of 1,178

Notes: All logs are openly available from the 4TU.ResearchData repository (van Dongen, 2017, 2020; de Leoni and Mannhardt, 2015; Mannhardt, 2016). "Activities" counts those appearing in at least 2% of a log's cases, which is the set handed to the model. A pair of activities is "determinate" when one of the two orders holds in at least 80% of the cases containing both.

show. The pair level is the more informative of the two here, both because it is where the determinacy conditioning bites and because it is closer to what the paper's predictions actually use, since Predictions #1 and #2 depend on adjacency rather than on the global ordering.

Results. Table SA.F.4 reports the results and Figure SA.F.2 the underlying distribution.

Against a null of 0.5, the model places 78.8% of all activity pairs in the direction the traces show. Restricting to the determinate pairs, those the data confirm have a real order, gives 78.9%, and weighting each pair by how determinate it is gives 78.8%. That these three numbers coincide is itself informative. The model does no better on the pairs that a real organization orders strictly than on pairs in general, so the aggregate is not being propped up by easy cases. At the log level the mean Kendall's τ is 0.48 with a median of 0.56, against a permutation null centered at 0.00 ($z = 6.8$, $p = 0.001$). The model is not echoing the list it was shown. The average τ between its output and the shuffled input is 0.007.

The ten alternative prompts again leave the picture unchanged. Determinate-pair accuracy under individual prompts spans 0.767 to 0.817 and log-level mean τ spans 0.433 to 0.559, so no conclusion here depends on the phrasing used in the main analysis.

The result that matters for the paper is the comparison with Appendix SA.F.1.1. Restricting both exercises to *adjacent* pairs, the statistic they share, the documented benchmark gives 0.700 under the main prompt and 0.697 averaged over the alternatives, while the observed benchmark gives 0.679 and 0.699. Two independent benchmarks, one written down by practitioners and one recovered from what organizations actually did, agree to

Table SA.F.4: Overlap Between GPT-Generated Orderings and Observed Process Execution

	Main prompt	10 alternative prompts
	(1)	(2)
Pairwise precedence accuracy against observed execution (null 0.5)		
All activity pairs	0.79	0.79
Determinate pairs only	0.79	0.79
Weighted by determinacy	0.79	0.79
Adjacent pairs only	0.68	0.70
Log-level Kendall τ against the observed ordering		
Mean across logs	+0.48	+0.49
Median across logs	+0.56	+0.54
Random ordering (null)	0.00	0.00
Event logs	8	8
Activity pairs	1,178	11,780
Determinate pairs	1,124	11,240
Cases	215,715	215,715

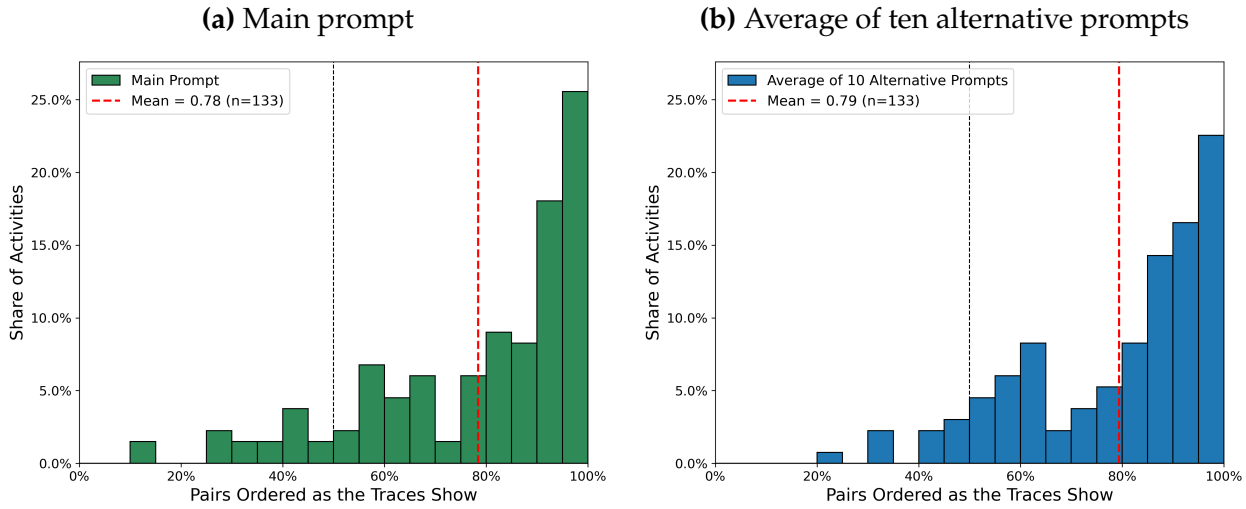
Notes: Pairwise precedence accuracy is the share of activity pairs the model places in the direction the event traces show, with a null value of 0.5. "Determinate pairs" are those for which one order holds in at least 80% of the cases containing both activities. "Adjacent pairs" are consecutive in the observed ordering and are the statistic directly comparable to the adjacent-pair agreement of Table SA.F.2. Column (2) pools the ten alternative prompts of Appendix SA.D, so its pair counts are ten times those of column (1). The random-ordering row is the mean of 1,000 random permutations per log.

within roughly two percentage points about how much sequence structure the instrument recovers.

Where the instrument fails, and why it understates its own accuracy. Performance varies sharply across logs, and the variation is diagnostic. Determinate-pair accuracy is 0.96 for road traffic fines, 0.95 for sepsis care, and 0.91 for international declarations, but 0.56 for requests for payment and 0.49 for domestic declarations. Inspecting the errors, the failures are overwhelmingly about *reading the labels*, not about understanding sequence.

Three examples carry most of the damage. In the domestic declaration log the model ranks `Request Payment` first, reading it as the employee's initiating request; in the data it is the second-to-last step, the finance system requesting disbursement. In the loan log the model treats `A_Accepted` as terminal, whereas in the data it is an early state meaning the application entered processing, preceding validation, offer creation, and completion; seven of the twelve most determinate errors are this single activity. In the sepsis log the model places `Return ER` mid-visit, while in the data it denotes a patient returning to the

Figure SA.F.2: Distribution of Overlap With Observed Execution Across Activities



Notes: The unit is an activity (133 across the eight logs). For each activity we take the determinate pairs it belongs to and plot the share the model ordered as the traces show. The black dashed line marks the coin flip and the red dashed line the mean. Averaging over prompts raises the share of activities above the coin flip from 86.5% to 91.7% while leaving the mean almost unchanged, the same pattern as in Figure SA.F.1. Because activities within a log share one ordering and each pair contributes to two activities, this distribution is descriptive; the inferential statistics are the pooled pair-level accuracy and the log-level permutation test.

emergency department after discharge and is genuinely last.

This matters for how the number should be read. Activity labels in an event log are the strings an information system emits, not descriptions of work. Some are terse (`A_Concept`), and many encode the actor rather than the action (`Declaration APPROVED by ADMINISTRATION`). The model is therefore working with substantially less context than an O*NET task statement provides, and the errors we observe are of a kind that richer task descriptions would largely prevent. Accuracy here is, in that specific sense, a conservative estimate of what the procedure achieves on the task statements the paper actually uses.

SA.F.2 Concern #2: Transferring the Labels onto Documented Sequences

Estimating anything on APQC sequences requires transferring AI exposure and execution labels, since those are measured over O*NET tasks and have no counterpart defined on PCF elements. We pool the Cross-Industry framework with APQC's seventeen industry-specific frameworks, discard duplicate process groups, embed each leaf element and each labeled O*NET task with the language embedding model `all-mpnet-base-v2`, and match every step to its nearest task by cosine similarity. Matching process steps to task statements is lossy, so we set a similarity threshold of 0.71 and carry a label across only when the

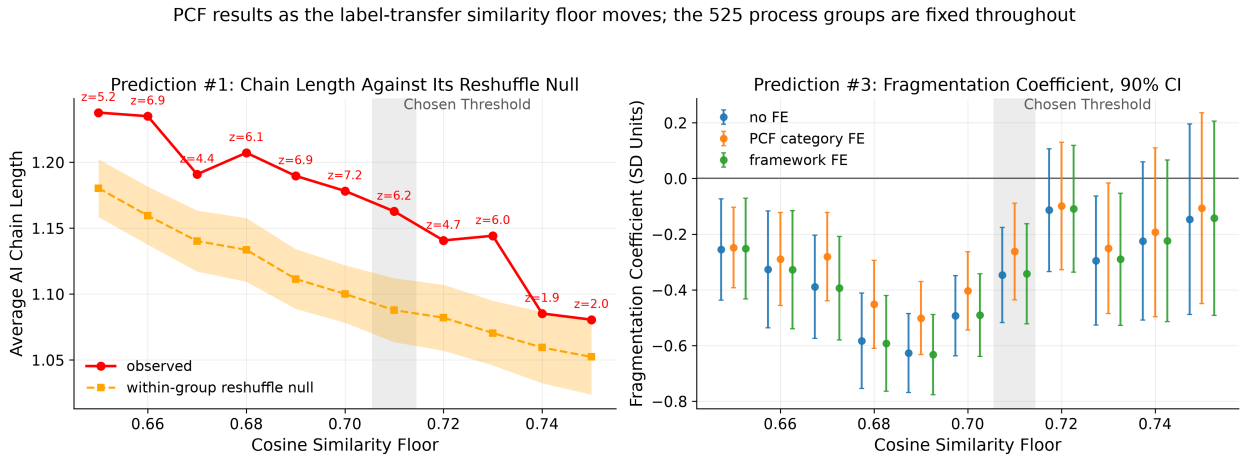
match clears it.[49] A step whose best match falls below the threshold is not discarded. Doing so would shorten the very sequences the exercise is about, and the shortening would not be random, since the steps that fail to match are concentrated in the operational and industry-specific parts of the frameworks. We instead keep every step in place and code the sub-threshold ones as neither AI-exposed nor AI-executed, which preserves the documented ordering intact and treats an unverifiable label as an absent one. Process groups with fewer than five steps are dropped, leaving 13,482 steps in 525 process groups and 26 steps per group, against 20 tasks per occupation in the O*NET sample.

What survives the threshold. The 0.71 cosine similarity threshold is demanding, and it should be read as such before any estimate is. Only 2,067 steps, 15% of the corpus, clear it, at a mean cosine similarity of 0.75. Because every step below it is coded manual and unexposed, the resulting sample is very thin in AI content, where 12% of steps are AI-exposed and 4% are AI-executed, against 44% and 13% in the O*NET task universe. Anything estimated on these sequences therefore asks whether a prediction holds on a corpus carrying roughly a third of the O*NET sample's AI density, with every label that does survive verified at high similarity. The caveat applies to the fragmentation estimates in columns (4)–(6) of Table 3 as much as to the chain-length test below.

How much the similarity floor matters. The floor is a judgement call, so we vary it from 0.65 to 0.75 in steps of 0.01 and re-run both exercises at each value. The 525 process groups are held fixed throughout, since the five-step minimum applies to the raw step count rather than to the labels, so raising the floor only relabels steps as unexposed and unexecuted. Figure SA.F.3 reports the result. Both signs survive the whole range. The average AI chain exceeds its reshuffle null at all eleven floors, and the fragmentation coefficient is negative at all eleven under each of the three fixed-effect specifications, reaching the 5% level in eight of them without fixed effects. The magnitude is not monotone, however, running from -0.26 at 0.65 to -0.63 at 0.69 and back to -0.11 at 0.72, and the standard error roughly doubles across the range as the labels thin out. Neighboring floors can therefore fall on opposite sides of a significance threshold without the underlying estimates differing, as at 0.72 and 0.73, whose point estimates each lie inside the other's confidence interval, so the range matters more than any single cell. We therefore choose the floor on the density of the transferred labels rather than on the estimates it produces. Raising it thins both the AI exposure and the AI execution content of the corpus, and exposure, the more common of the two labels, is the binding one. At 0.71 the corpus retains 12% of its steps

[49] We chose the 0.71 similarity threshold after manual inspection of some of the matches. The paragraph on the similarity floor below reports how the results move as it is varied.

Figure SA.F.3: PCF Results as the Label-Transfer Similarity Floor Varies



Notes: Each point re-runs the label transfer at the indicated cosine similarity floor and re-estimates on the resulting labels. The 525 process groups are the same at every floor, since the five-step minimum is applied to the raw step count and not to the labels. The left panel reports the average AI chain length in red against the mean of 1,000 within-group step-order reshuffles in orange, with the orange band covering two standard deviations of that null and the annotation giving the z-statistic. The right panel reports the coefficient on the empirical fragmentation index from Equation (14) with 90% confidence intervals, under each of the three fixed-effect specifications of columns (4)–(6) of Table 3, which are drawn in the same order of colors as the placebo panels of Figure A.1 and are offset horizontally at each floor so that their intervals do not overlap. The grey vertical band marks the 0.71 floor used throughout the paper, and covers every estimate made at that floor.

as AI-exposed and 4% as AI-executed, and every floor above it drops the exposure share below 10%, which is why we settled there.

Prediction #1. Subsection 7.1 reports this exercise alongside the O*NET one; we give it here in full. The average AI chain in the PCF dataset spans 1.16 steps, against 1.45 in the main text for O*NET. The magnitude is small, and mechanically so, because with only 4% of steps AI-executed, there is little opportunity for long runs to form, and the ceiling on the statistic is far below the O*NET sample's. What the exercise can still ask is whether the runs that do form are longer than the same labels arranged at random along the same sequences, and they clearly are. Benchmarked as in Section 7.1 against 1,000 draws of a within-group step-order reshuffle, the null averages 1.09 with a standard deviation of 0.01, and the observed value sits at the 100th percentile ($z = 6.2$); against a within-category reassignment that preserves group size, the null averages 1.06 and the observed value again sits at the 100th percentile ($z = 10.2$). The economic magnitude of AI chain length is therefore modest, but the co-occurrence of AI-executed content behind it is unambiguous. Since no language model placed these steps in order, the contiguity is a property of the documented workflow rather than of the ordering procedure.

Prediction #2. We do not attempt a counterpart of the neighbor-execution regression, since that test requires the same step to appear in several workflows with *different* ex-

ecution outcomes, while transferred labels are a deterministic function of the matched task. Restricting to steps that clear the threshold and whose matched task maps to a single DWA appearing in more than one process group leaves 981 observations across 137 DWAs, of which only 9 contain both an AI-executed and a non-executed step, against 10,708 tasks across 1,748 DWAs in the baseline specification of Table 2, so the matched corpus is underpowered for that exercise by roughly an order of magnitude.

Reading the PCF fragmentation coefficient. The fragmentation regression of Equation (14) is estimated on this corpus in columns (4)–(6) of Table 3 and discussed in Subsection 7.3, which also discusses why the workflow-level channel is detected on this corpus and not on the O*NET sample. One feature of the transfer belongs here rather than in the main text. All variables are z-scored against their mean and standard deviation within the PCF sample, and the index varies far less across PCF process groups than across O*NET occupations, with a standard deviation of 0.06 against 0.26. A one-standard-deviation move is thus a much smaller move in the index here, and the slope per unit of the index is correspondingly steeper.

SA.F.3 Summary and Takeaways

The benchmarks answer the objection that motivated them. An instrument that merely clustered semantically similar items would have no reason to place seven in ten consecutive step pairs in the practitioners' direction, or to reproduce a full documented sequence one time in four, on a corpus that working groups wrote down with no language model involved at any stage. Nor would it place four in five activity pairs in the direction real organizations executed them, on pairs the traces confirm have a true order. That two independent benchmarks, one documented and one observed, agree to within roughly two percentage points on the statistic they share is the strongest evidence here that what the instrument recovers is sequence rather than topic.

The overlap is nonetheless imperfect, and two considerations bound how it should be read. On one side, some of the residual disagreement is indeterminacy in the benchmark rather than error in the language model's logic, so the measured overlap understates agreement with genuine workflow order. On the other, the shortfall is not merely noise. Across the same branches the model's orderings agree with *each other* at an average pairwise τ of 0.79, well above the 0.53 they achieve against APQC, so the instrument is considerably more self-consistent than it is accurate.

Three limitations qualify what these exercises establish. First, they validate the ordering *procedure* rather than the specific O*NET orderings used in the paper, since neither the PCF nor the event logs records the order in which O*NET occupations perform their tasks.

Second, the PCF is not a random sample of work, since it is weighted toward enterprise process work such as finance, procurement, human resources, information technology, and supply chain. Third, the PCF carries no AI exposure or execution labels of its own, since our AI exposure labels come from Eloundou et al. (2024) and our AI execution labels from the Anthropic Economic Index (Handa et al., 2025), both defined over O*NET tasks. The labels we use on the PCF sample are transferred from the nearest O*NET task (Appendix SA.F.2), and the lossy match leaves that sample thin in AI content, which is why we treat the two samples as complements and report them side by side rather than relying on either alone. Working with O*NET keeps our measures comparable to the rest of the literature that uses these now-standard objects, at the cost of not observing workflow order directly; this appendix is our attempt to bound that cost.

References for Supplementary Appendix

APQC (2026). Process Classification Framework (PCF) – Cross Industry, Version 8.0. https://www.apqc.org/process-frameworks. American Productivity and Quality Center. The industry-specific frameworks are drawn from the same collection.

de Leoni, M. and F. Mannhardt (2015). Road Traffic Fine Management Process. https://doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5. 4TU.ResearchData.

Eloundou, T., S. Manning, P. Mishkin, and D. Rock (2024). GPTs are GPTs: Labor Market Impact Potential of LLMs. *Science 384*(6702), 1306–1308.

Handa, K., A. Tamkin, M. McCain, S. Huang, E. Durmus, S. Heck, J. Mueller, J. Hong, S. Ritchie, T. Belonax, K. K. Troy, D. Amodei, J. Kaplan, J. Clark, and D. Ganguli (2025). Which Economic Tasks are Performed with AI? Evidence from Millions of Claude Conversations. *arXiv preprint arXiv:2503.04761*.

Horton, J. and R. Horton (2024). EDSL: Expected Parrot Domain Specific Language for AI Powered Social Science. *Expected Parrot Working Paper*.

Mannhardt, F. (2016). Sepsis Cases – Event Log. https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460. 4TU.ResearchData.

National Center for O*NET Development (2023). O*NET 27.3 Database. https://www.onetcenter.org/database.html. U.S. Department of Labor, Employment and Training Administration.

van Dongen, B. (2017). BPI Challenge 2017. https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b. 4TU.ResearchData. Loan application handling at a Dutch financial institution.

van Dongen, B. (2020). BPI Challenge 2020. https://doi.org/10.4121/uuid:52fb97d4-4588-43c9-9d04-3604d4613b51. 4TU.ResearchData. Collection of five university travel and reimbursement event logs.