Task allocation in networks

Francis Bloch, Lilian Hartmann, Luca Paolo Merlino and Dotan Persitz*

July 2023

VERY PRELIMINARY AND INCOMPLETE

Abstract

We study dynamic task allocation when there is a fixed bipartite network associating workers to tasks. We analyze two approaches - centralized and decentralized. First, we study the optimal policy of a planner whose objective is to minimize the expected time of completion of all tasks. Second, we analyse a game played by workers who independently choose their tasks and are rewarded each time they complete a task. We show that both the planner's and the worker's problems are NP-hard and characterize networks for which the planner's and workers' policies are time-consistent. When policies are time-consistent the planner prefers the workers to start with the hardest tasks, whereas workers always prefer to start with easier tasks. We show that the two policies only coincide when the bipartite network satisfies a strong symmetry condition on the bipartite network. Differential rewards can be used to implement the planner's optimal task allocation and we show that non-contingent rewards, which are independent of the set of remaining tasks, can be used as long as there is no task that a single agent can complete.

Keywords: Task allocation, Scheduling, Suitability constraints, Bipartite networks.

JEL Codes: D20, L20, K0

*Bloch: Paris School of Economics and Université Paris 1 Panthéon Sorbonne, francis.bloch@univ-paris1.fr; Hartmann: , lilianhartmanndesnos@gmail.com, Merlino:Université Libre de Bruxelles, luca.paolo.merlino@ulb.be, Persitz: Coller School of Management, Tel Aviv University, persitzd@post.tau.ac.il

1 Introduction

Large manufacturing, infrastructure, auditing or research projects can often be decomposed into many complementary tasks which all need to be completed for the success of the project. The allocation of workers to tasks is a key problem faced by organizations responsible for large projects, and optimal task allocation policies have long been a subject of interest in Operations Management and Economics. As an example, CERN in Geneva groups together 12,200 scientists engaged in large-scale high energy physics research projects using the Large Hadron Collider (LHC). Scientists are grouped in seven distinct, overlapping, collaborative teams called "experiments". Each specific research project consists of different tasks, requiring different skills, and tasks are allocated to scientists belonging to the experiments in which the project is held (see Di Stefano and Micheli (2022) for a description of the internal organization at CERN).

In this paper, we consider a task allocation problem for large projects when the suitability of workers to tasks is represented by an exogenous bipartite network. A worker is connected to a task if she possesses the required skill to accomplish the task. Tasks and skills are not in a one-to-one relation, so that any worker with a given skill may accomplish several tasks, and any task can be completed by workers with different skills. The project consists in several tasks, with no precedence order, and is completed whenever all tasks are accomplished. The completion of any task is a stochastic process, driven by an exponential distribution which depends on the number of workers working on the task.

If all workers are able to work on all the tasks, or if the mapping between workers and tasks is bijective, the task allocation problem is trivial, as any allocation results in the same expected completion time of the project. But the feasibility constraints captured by the bipartite suitability network introduce a new, dynamic, aspect to the task allocation problem. Once a task is completed, the set of agents who are able to work on the remaining tasks is given by a new, reduced, bipartite suitability network. The current assignment of workers to tasks affects the probability that any task is completed, hence the distribution over the bipartite suitability networks in the next steps of the project. Therefore, assigning a worker work in the present on a certain task affects the distribution of workers who will be able to work in the future on the remaining tasks. This dynamic effect transforms the task allocation problem into a complex Markov Decision Problem.

We analyze the task allocation problem both in a centralized and decentralized settings. In the *centralized setting*, a single planner allocates tasks to workers in order to minimize expected completion time of the project. In the *decentralized setting* every worker independently chooses which task to work on and is rewarded every time she completes a task. We compute the optimal policy of the planner in the centralized setting (the "planner's problem") and the Nash equilibrium in the decentralized setting (the "worker's game").

We first observe that both the computation of the optimal policy of the planner and of the Markov Perfect Equilibrium strategies of the workers are complex problems. They correspond to the computation of an optimal policy in a recursive, finite-horizon Markov Decision Problem where the state space, which is equal to the number of sub-networks after deletion of single nodes of a given bipartite network, grows exponentially with the number of tasks.

Faced with the complexity of the problem, rather than looking for approximation algorithms, we take a different route. We analyze those bipartite suitability networks for which the optimal policies and Markov Perfect Equilibrium strategies take a simple, time-consistent form, where every worker works on the same task until it is completed.

We first observe that, whenever there are two tasks to accomplish, the optimal policies are easy to characterize. In the planner's problem, the planner always allocates a worker to the "hardest" tasks (the task with the hardest required skills, that the smallest number of workers can accomplish). This is because, whenever the hardest task is accomplished, the set of workers available to work on the remaining task is larger, and hence the expected time of completion of the project is shorter. In the worker's game, we show that workers have weakly dominant strategies. This strategy prescribes that a worker who can work on both tasks always choose to work on the "easiest" task (the task with the easiest required skills, that the largest number of workers can accomplish). This is due to the fact that, once the easiest task is accomplished, the set of agents competing for the reward on the second task is smaller, increasing the expected reward of the worker.

Extending these optimal policies to more than two tasks requires assumptions on the bipartite suitability network. We uncover properties on the bipartite network, *Union Size Invariance*, *Submodularity* and *Increasing Differences* which guarantee that the optimal policy in the planner's problem and the Markov Perfect Equilibrium strategies in the worker's game are time-consistent and prescribe the same choice as in the two-task problem. Union Size Invariance states that, whenever the number of workers who can work on task a is greater than the number of workers who can work on task b, the same ranking holds for the number of agents who can work on task a or any union C of tasks different from a and b, and the number of agents who can work on b or C. Under Union Size Invariance, the ranking of tasks remains invariant throughout the entire project, and the planner optimally chooses to allocate workers to the hardest tasks. Submodularity and Increasing Differences are different properties, which state that the marginal effect of a new task on the number of agents who can achieve the task is lower for tasks with lower number of workers. Submodularity and Increasing Differences, together with Union Size Invariance, to show that in any Markov Perfect Equilibrium of the worker's game, workers always choose to work on the easiest task.

While Union Size Invariance, Submodularity and Increasing Differences can be viewed as stringent requirements on the bipartite suitability network, we provide two interesting families of networks which satisfy both conditions. In the ranked tasks model, tasks are ordered by difficulty in the sense that any worker who can work on task i can work on any task $j > i$. In the generalist-specialists model, the set of workers is divided into specialists who can only work on a single task, and generalists who can work on all tasks.

Our analysis points to a complete reversal of policies in the centralized and decentralized models. In the centralized model, hardest tasks are completed first, whereas they are completed last in the decentralized model. In fact, the incentives of workers in the decentralized model are opposite to the planner's to the point that an increase in the number of workers may result in an increase in the completion time of the project. The only situation where the workers' and planner's incentives are aligned is when they are always indifferent among all the tasks, a situation which only arises when the bipartite suitability network satisfies a strong symmetry condition.

In order to implement the first best task allocation, the planner can use differential rewards. We show that there always exist a system of contingent rewards (which depend on the set of remaining tasks) which can be used to implement the first best. Non-contingent rewards, chosen ex ante and independently of the set of remaining tasks, can be used if and only there is no exclusive task (that a single agent can complete). We provide exact formulas for the differential rewards with two tasks and with three tasks in the ranked tasks model.

We also analyze highlight the difference between the task allocation of the planner and of the workers in two ways. We first provide an example to show that, in the equilibrium task allocation of the workers' game, an increase in the number of workers can harm the planner, resulting in an increase in the expected completion time. Finally, we compute the "price of anarchy" by measuring the expected time of completion of all tasks in the equilibrium task allocation of the workers' game and the optimal solution of the planner. We show that when there are two tasks, the expected completion time in the workers' game is at worst 20% higher than in the optimal solution, and that the price of anarchy increases with the number of tasks at most in a linear way.

The rest of the paper is organized as follows. The next Subsection describes the related literature.

Section 2 presents the model. We then analyze the planner's problem in Section 3 and the workers' game in Section 4. Section 5 discusses the comparison between the two solutions. We conclude in Section 6. All proofs are relegated to the Appendix.

1.1 Related Literature

The task allocation problem we study can be interpreted as a *scheduling* problem where tasks are jobs and workers are servers which process the jobs. There is a known, finite sequence of jobs, processing times are exponential and every job can be processed in parallel by multiple identical servers but jobs are subject to machine eligibility constraints. Following classical terminology of scheduling problems (Pinedo, 2016), we consider an identical parallel machine environment, with equal exponential processing times and machine eligibility restrictions, and either a makespan or total completion time objective. When the machines are nested (the "ranked tasks" model), it is known that the Least Flexible Job-fastest Machine first (LFJ-FM) policy minimizes both the expected makespan and expected sum of completion times among dynamic preemptive policies (Theorem 12.2.7 p. 338 in Pinedo, 2016 and Pinedo and Reed, 2013).[1] The Least-Flexible Job policy is equivalent to our "hardest task policy". It is a nonpreemptive policy—it does not require servers to switch jobs before the completion of a task. As noted in Pinedo and Reed (2013), the policy is closely related to the Least Laxity First rule used for scheduling of multi-processor systems in computer science. (See Davis and Burns, 2011.) Our contribution to the literature in scheduling is two-fold. First, we extend the optimality of the "hardest task" rule beyond the nested machine eligibility environment to a much larger class of machine eligibility restrictions. Second, and most importantly, we contrast the solution of the centralized problem with the equilibrium of the decentralized model, where each server acts independently, and show that the two models may result in opposite incentives.

Our model is also more distantly related to the literature on task allocation and team formation in multi-agent systems—see Rizk, Awad and Tunstel (2019) and Khamis, Hussein and Elmogy (2015) for recent surveys. However, the emphasis of the literature in task allocation in multi-agent systems is very different. It stresses the importance of coordination rather than the dynamic allocation of tasks, and it highlights the importance of skills complementarity in team formation. The methods used are also very different. The literature on Multi-Robot Task Allocation (MRTA), for example, is mostly concerned with approximation algorithms and simulations.

Finally, our paper is related to the literature on sequential search in economics pioneered by Weitzman

[1] Pinedo (2016) notes that it is easier to characterize optimal policies with machine eligibility requirements in the stochastic than in the deterministic model. See Leung and Li (2008) and Leung and Li (2016) for surveys on deterministic scheduling problems with machine eligibility requirements.

(1979)—see Doval (2018), Eliaz, Fershtman and Frug (2021) and Fershtman and Pavan (2022) for recent extensions of the Weitzman model. In Weitzman's model, a single agent chooses both the order in which she will check different alternatives and a stopping rule. The model can be extended to parallel search—see Vishwanath, 1992 for an early study of parallel search and the application to test design in Loch, Terwiesch and Thomke, 2001. There are several differences between the parallel search extension of the Weitzman model and our approach. First we consider a planner whose objective is to complete all tasks rather than select an alternative. Second, we study a stochastic model, with random processing times. Third, we only consider identical agents and tasks, and introduce heterogeneity only in the workers' ability to search for different alternatives.

2 The model

2.1 Workers and tasks

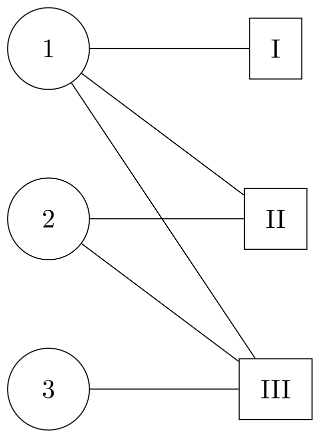
Consider a finite set of n workers, $W = \{w_1, w_2, \ldots, w_n\}$, and a finite set of m tasks, $T = \{t_1, t_2, \ldots, t_m\}$. A bipartite network G between W and T assigns to each worker w_i, the set $H_i \subseteq T$ of tasks on which the worker can work. H_i's cardinality is denoted by h_i. Conversely, for any task t_j, we let $A_j \subseteq W$ denote the set of workers who can work on this task. We denote A_j's cardinality by a_j. We say that task t_j is "harder" than task t_k if $a_j < a_k$.

For any bipartite graph G, we let $T(g)$ (with cardinality $t(G)$ denote the set of tasks and $W(G)$ (with cardinality $w(G)$) denote the set of workers. In addition, let $H(G) = \{H_1, H_2, \ldots, H_n\}$ and $A(G) = \{A_1, A_2, \ldots, A_m\}$ denote the set of sets of tasks and workers. For any subset of tasks $S \subseteq T$, A_S denotes the set of all agents who can work on some task in S. Formally, $A_S = \bigcup_{j \in S} A_j$. We denote A_S's cardinality by a_S.

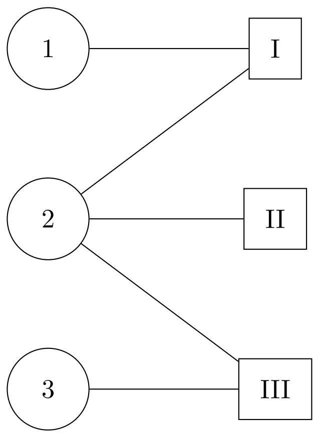
We assume that the graph G is connected. This implies that for any task $t_j \in T$, there exists another task t_k such that $A_j \cap A_k \neq \emptyset$. Note that if the graph is not connected, we can decompose the problem into a series of independent problems, one for each connected component of the graph G.

2.2 Two examples

Example 1: The Ranked tasks Model: Suppose that tasks can be ranked by objective difficulty (task t_j is more difficult than task t_k if $j < k$). In addition, suppose that every agent has a given set of skills which allows her to work on any task below a given difficulty threshold. That is, in the



(a) The "Ranked tasks" model where $q_1 = I$, $q_2 = II$ and $q_3 = III$.



(b) The "Specialists-Generalists" model where $H_1 = \{I\}$, $H_2 = T$ and $H_3 = \{III\}$.

Figure 1: Two examples using three agents ($\{1, 2, 3\}$) and three tasks ($\{I, II, III\}$).

"Ranked tasks" bipartite graph, each worker w_i is characterized by an integer $q_i \in \{1, 2, \ldots, m\}$ such that $H_i = \{t_{q_i}, t_{q_i+1}, \ldots, t_m\}$.[2] Note that in the "Ranked tasks" model, the sets in $A(G)$ are nested, that is, for any $j < k$, $A_j \subseteq A_k$.[3] Figure 1a illustrates a "Ranked tasks" model with three agents and three tasks.

Example 2: The Specialist-Generalists Model: Suppose that there are two types of workers - Specialists who can only work on one task (H_i is a singleton containing one element from T), and Generalists who can work on all tasks ($H_i = T$). Denote the set of specialists that work on task k by $W_k^S = \{w_i \in W : H_i = \{k\}\}$ and the set of generalists by W^G. Therefore, for every task $t_k \in T$, $A_k = W_k^S \cup W^G$. Figure 1b illustrates a "Specialist-Generalists" model with three agents and three tasks.

2.3 Timing

We suppose that the completion of tasks is stochastic. Time is continuous, and every worker draws a success according to a Poisson distribution with fixed homogeneous parameter λ. Therefore, if a team $B \subseteq W$ of cardinality b works on the same task t_j, the first success is drawn according to a Poisson distribution with fixed parameter $b\lambda$. Therefore, the time to success is distributed according to an exponential distribution with mean $\frac{1}{b\lambda}$.

The model is dynamic. We suppose that there is not precedence order on tasks, which can be accomplished in any sequence. We refer to the period between the beginning and the completion of the

[2]Another interpretation of the model is that tasks are ranked by their confidentiality (t_1 is the most confidential task) and each agent w_i is characterized by a security clearance q_i that allows her to work on tasks that are not more confidential than task t_{q_i}.

[3]The bipartite graph corresponding to the "Ranked tasks" model is called a "bipartite chain graph".

first task as phase 1, to the period between the completion of the first task and the completion of the second task as phase 2, etc. Workers can be re-assigned to the set of the remaining tasks at any point in time. However, since the Poisson distribution is memory-less, it is enough to consider re-assignments only between phases.

2.4 Objectives

We consider different preferences for the *planner* and for the *workers*. The objective of the planner is to complete all targets as quickly as possible. He suffers additive waiting cost across phases before the completion of the project. Hence his utility is given by $U_p = -\mathbf{E}\tilde{\tau}$ where $\tilde{\tau}$ is the random total time of completion of all tasks. By contrast, we suppose that the workers receive a unit reward every time they complete a task. Hence, agent w_i receives a utility $U_i = \mathbf{E}\tilde{\kappa}_i$ where $\tilde{\kappa}_i$ is the random number of tasks that the agent has completed by herself.

We thus distinguish between two problems. The (centralized) *planner's problem* is an optimization problem where the planner chooses a policy to assign workers to tasks. The (decentralized) *worker's game* is a stochastic game where workers choose which task to work on at any phase.

3 The planner's problem

The planner's problem is to assign all workers to one of the tasks. Formally, an assignment is a mapping $\mu : W \to T$ such that for any w_i, $\mu(w_i) \in H_i$. Given that all workers are identical except for their position in graph G, an action of the planner can be summarized by a vector $\mathbf{r} = (r_1, .., r_j, .., r_m)$, where r_j denotes the number of workers working on task $j \in T(G)$. The vector $\mathbf{r}$ must of course be obtained by an assignment of workers to tasks: there must exist a mapping μ such that $R_j = \{i | \mu(i) = j\}$.

A policy for the planner is a choice of actions at any graph G that arises over time. We let $v(G, \mathbf{r})$ denote the value to the planner of the action $\mathbf{r}$ at G, which is equal to the negative of the expected time of completion of all tasks in $T(G)$. The value to the planner at the *optimal* action is denoted by $V(G)$.

We first give a simple recursive formula for the value of action $\mathbf{r}$ at G. Define $G - j$ to be the suitability graph obtained from G after deletion of the node t_j. We express the value of the planner at G as a function of the values $V(G - j)$ for any task $j \in T(G)$:

$$v(G, \mathbf{r}) = -\frac{1}{\lambda w(G)} + \sum_{j \in T(G)} \frac{r_j}{w(G)} V(G - j). \tag{1}$$

To understand equation 1, notice that, given the exponential distribution of the time of completion of tasks, the expected time of completion of *any* task in $T(G)$ is equal to $\frac{1}{\lambda w(G)}$. Once any task j is completed, the new suitability graph is $G-j$ and the planner selects the optimal action at $G-j$ resulting in the value $V(G-j)$.

The planner's problem is thus a finite-horizon, recursive, Markov Decision Problem where the set of states is the set of all graphs obtained from the initial suitability graph G_0 after deletion of any subset S of tasks in $\mathcal{T}$. We use the recursive formula to write the Bellman equation for the planner's problem as:

$$V(G) = -\frac{1}{\lambda w(G)} + \max_{\mathbf{r}} \sum_{j \in T(G)} \frac{r_j}{w(G)} V(G-j). \tag{2}$$

The Bellman equation shows that the action of the planner does not affect the reward received at state G, which is always equal to $-\frac{1}{\lambda w(G)}$, but only the probability of transition to the states $G-j$. Hence, the optimal action at G is only driven by the values $V(G-j)$ and we directly obtain the following Lemma:

Lemma 1 *The planner optimally assigns worker i to a task $j \in H_i$ such that*

$$V(G-j) \geq V(G-k) \forall k \in H_i.$$

Lemma 1 shows that the planner's optimal choice is characterized by a ranking of tasks at each and every suitability graph G. This ranking is obtained by considering the values $V(G-j)$ for any task j. If different tasks generate the same value, and this is the highest value among the tasks in H_i, the planner is indifferent between assigning agent i to any of the tasks. We then assume, without loss of generality, that there exists a fixed, exogenous ranking of the tasks and that the planner assigns agent i with highest rank among the tasks which generate the same value.[4]

The ranking of tasks is defined recursively. When there are only two tasks left, $T(G) = \{t_1, t_2\}$, we can easily compute the value $V(G-1)$ and $V(G-2)$, as there is only one task left to complete. Hence $V(G-1) = -\frac{1}{a_2}$ and $V(G-2) = -\frac{1}{a_1}$. The optimal action is then to assign any agent who can work on both tasks to the task with the "hardest" task, namely the task with the smallest cardinal. Formally,

Lemma 2 *Suppose that there are only two tasks, $T(G) = \{1, 2\}$. Then the planner optimally assigns an agent who can work on both tasks to task 1 if $a_1 < a_2$, to task 2 if $a_2 < a_1$ and to either task if $a_1 = a_2$.*

[4]There is no loss of generality in specifying a tie-breaking rule in the Markov Decision Problem because all tie-breaking rules generate the same value $V(G)$. To see this, consider two tasks j and k such that $V(G-j) = V(G-k)$. By equation 2, any two assignments (r_j, r_k) and (r'_j, r'_k) such that $r_j + r_k = r'_j + r'_k$ generate the same value $V(G)$. The argument clearly extends to the case where there are more than two tasks with the same value.

The simple argument underlying Lemma 2 is similar to the argument showing that a server must be assigned to the least flexible job in the scheduling problem with nested machines—see Pinedo and Reed (2013). The objective of the planner is to maximize the number of active workers after the completion of any of the two tasks. The number of active workers will clearly be larger if the hardest task is completed first.

With more than two tasks, the computation of the optimal action of the planner becomes much more complex.[5] It depends not only on the number of workers capable of working on each task, but also on the number of workers capable of working on any subset of tasks. It is well known for a fixed finite state space, the solution to a Markov Decision Problem can be obtained as a solution to a polynomial, Linear Programming problem—see Papadimitriou and Tsitsiklis (1987) for a discussion of the complexity of Markov Decision Problems with finite set space. However, in the dynamic task allocation problem, the state space is the set of all networks which can be generated from G_0 by deleting any subset of tasks, and hence grows exponentially with the number of tasks.

Faced with the high complexity of the planner's problem for general suitability graphs, we next investigate situations where the problem becomes "simple". We define "simple" problems as problems for which the planner always assigns an agent to the same task as long as the task is not completed, independently of the suitability graph.

Definition 1 *The planner's problem is* simple *if the ranking of tasks is independent of the suitability graph G.*

In a simple problem, there exists a single ranking of the tasks in T, and each agent who is capable of working on multiple tasks is assigned to the task with the highest rank (the task j for which the expected value $V(G - j)$ is highest) for any suitability graph G such that $j \in T(G)$. Hence, in a simple problem, the planner does not ask agents to switch tasks, but instead consistently assigns them to work on the same task until it is completed. In other words, using the terminology of scheduling problems, a problem is simple if the optimal policy never involves preemption. We now define a property on the original suitability graph G_0, that we term Union Size Invariance, that guarantees that the problem of the planner is simple.

Definition 2 *The suitability graph G_0 satisfies Union Size Invariance if, for any two tasks j, k,*

- *If $a_j = a_k$, for any subsets of tasks S which does not contain j, k, $a_{S \cup j} = a_{S \cup k}$;*

[5] Already with three tasks, the problem becomes hard to solve. In Appendix B, we propose a detailed analysis of the three-tasks problem.

- *If $a_j < a_k$, for any subsets of tasks S which does not contain j, k, $a_{S \cup j} \leq a_{S \cup k}$.*

Union Size Invariance guarantees that the ranking obtained by measuring the number of workers capable on working on every single task (the ranking generated by a_j) is not reversed when one considers situations where a larger set of tasks remains available. Intuitively, this condition guarantees that the ranking used to assign workers when there is a single task left remains true for any number of remaining tasks. Note that Union Size Invariance treats differently situations where two tasks are ranked at the same level ($a_j = a_k$) or at different levels $a_j \neq a_k$. In the first case, indifference of the planner must remain true for any number of remaining tasks, so that the set of agents who can work on any subset $S \cup j$ must be exactly equal to the set of agents who can work on any subset $S \cup k$. In the second case, the ranking of tasks must not be reversed, but the number of agents who can work on any subsets $S \cup j$ and $S \cup k$ is not exactly pinned down.

It is easy to check that the ranked tasks model (or equivalently the nested machines constraint in Pinedo and Reed, 2013) satisfies Union Size Invariance. Consider two tasks j, k such that $a_j = a_k$. Then by construction, we must have $A_j = A_k$. Hence for any subset of tasks S, $A_{S \cup j} = A_{S \cup k}$. Next suppose $a_j < a_k$. Let S be a set of tasks and l the task with the highest index in S. If $l > k$ then $A_{S \cup j} = A_{S \cup k} = A_l$ and hence $a_{S \cup j} = a_{S \cup k}$. If $k \geq l$ then $A_{S \cup k} = A_k \supseteq A_{j \cup S}$, so that $a_{S \cup k} \geq a_{S \cup j}$.

The generalists-specialists model also clearly satisfies Union Size Invariance. Consider two tasks j, k. Notice that $A_j = SP_j \cup GE, A_k = SP_k \cup GE$ and $SP_j \cap SP_k = \emptyset$. For any subset of tasks S, let SP_S be the number of specialists who can work on any task in S. Then $SP_S \cap SP_j = SP_S \cap S_k = \emptyset$ and $A_S = SP_S \cup GE$. We thus have $a_{S \cup j} = a_j + a_S - g$ and $a_{S \cup k} = a_k + a_S - g$. This implies that if $a_j = a_k$, then $a_{S \cup j} = a_{S \cup k}$ and if $a_k < a_l$ then $a_{S \cup j} < a_{S \cup k}$.

There are of course many other suitability graphs which satisfy Union Size Invariance. For example, suppose that every worker can work on at most two tasks and that all tasks have a different ranking ($a_j \neq a_k$ for all j, k). Then Union Size Invariance is equivalent to the requirement that for any two tasks j, k such that $A_j \cap A_k \neq \emptyset$ and $a_j < a_k$, and any subset of tasks S different from k, l

$$\sum_{l \in S} (|A_k \cap A_l| - |A_j \cap A_l|) \leq (a_k - a_j).$$

This condition is likely to be satisfied if the number of agents who can work on two tasks is small relative to the number of agents who can only work on one task, or if the number of agents who can work on two tasks has small variation across pairs of tasks.

We now state the main Theorem of this Section, showing that the Markov Decision Problem of the planner is simple under Union Size Invariance.

Theorem 1 *Suppose that the initial suitability graph G_0 satisfies Union Size Invariance. Then the planner's dynamic task allocation problem is simple and assigns all workers to the hardest tasks.*

Theorem 1 shows that, under Union Size Invariance, the vector $\mathbf{a} = (a_1, ..., a_m)$ is a sufficient statistic to generate all states in the state space. The ranking of tasks is given by the ranking of the numbers a_j and the problem becomes linear in the number of tasks. The planner always assigns a worker to the "hardest task" (the task with the lowest number a_j) first. The planner prefers to send the workers to the hardest tasks first, in order to maximize the number of active agents in subsequent periods, thereby reducing the expected time of completion of the whole project.

In the proof of Theorem 1, we also show that the expected value of the planner is a strictly increasing function of the number of workers who can work on any task. Hence, when the agents are assigned to targets according to the planner's optimal policy, increasing the number of agents always results in a decrease in the expected completion time.

4 The worker's game

We now turn our attention to the worker's game. At any suitability graph G, agent i chooses a probability distribution p_i over the tasks in $H_i(G)$, with p_i^j denoting the probability that agent i works on task j. Worker i receives a reward normalized to 1 if she completes the task. As all agents succeed according to the same Poisson process with parameter λ, this happens with probability $\frac{1}{w(G)}$.

The model can also be interpreted as a model of team cooperation. If workers form teams to work on a task and share equally the the reward inside teams, the agent's payoff in a team of size r_j would be equal to

$$\frac{r_j}{w(G)} \frac{1}{r_j} = \frac{1}{w(G)}.$$

The worker's game is a finite-horizon, multi-stage game, and we consider Markov Perfect Equilibria (MPE) of the game. As in the planner's problem, the state space is the set of all subgraphs of the initial suitability graph G_0 after deletion of the nodes corresponding to tasks. A strategy for player i is a mapping associating a probability distribution over $H_i(G)$ to any suitability graph G.

Consider a fixed MPE. We let $U_i(G)$ denote the payoff of player i at suitability graph G at equilibrium. We denote by $u_i(G, \mathbf{p})$ the payoff to player i given the strategy profile $\mathbf{p}$ at G, assuming that the MPE is played at all subsequent periods. Using the recursive structure of the game, we compute

$$u_i(G, \mathbf{p}) = \frac{1}{w(G)} + \sum_{j \in T(G)} \frac{\sum_k p_k^j}{w(G)} U_i(G - j), \tag{3}$$

where $\sum_k p_k^j$ denotes the expected number of workers working on task j.

In a MPE, every player chooses p_i to maximize $u_i(G, p_i, p_{-i})$ at any graph G. Because $u_i(G, p_i, p_{-i})$ is a linear function of $\mathbf{p}$, every worker has a dominant strategy. As in the planner's problem, the dominant strategy of the worker is to select a task $j \in H_i(G)$ for which the continuation value $U_i(G - j)$ is highest.

Lemma 3 *In a Markov Perfect Equilibrium, if $p_i^j > 0$, then*

$$U_i(G - j) \geq U_i(G - k) \forall k \in H_i(G).$$

When there are multiple tasks which result in the same continuation value, we assume that agent i chooses her task according to a fixed, exogenous ranking. As opposed to the planner's problem, this tie-breaking rule is not innocuous in the construction of the MPE of the worker's game. It will affect the expected number of workers who choose to work on every task, and hence the value of all workers in the game—including workers who are not indifferent among tasks.

As in the planner's problem, the dominant strategy of a worker is easily computed when there are only two tasks left. When $T(G) = \{t_1, t_2\}$, the continuation values are $U_i(G - 1) = \frac{1}{a_2}$ and $U_i(G - 2) = \frac{1}{a_1}$. The dominant strategy is to work first on the "easiest" task for which the number of workers is highest, and reserve the hardest task for the end.

Lemma 4 *Suppose that there are only two tasks, $T(G) = \{t_1, t_2\}$. Then in a MPE, a worker who can work on both tasks prefers to work on task t_1 if $a_1 > a_2$, on task 2 if $a_2 > a_1$ and on either task if $a_1 = a_2$.*

Lemma 4 shows that the incentives of the planner and the workers are diametrically opposed. The worker's objective is to minimize the number of active workers in the second period in order to limit competition. They will thus choose to work on the easiest task first and keep the hardest task for the last period.

The computation of the dominant strategy of a worker is as complex as the computation of the solution of the planner's optimal policy. As the number of states grows exponentially with the number of tasks, the computation of the MPE of the workers' game also grows exponentially with the number of tasks. As in the planner's problem, we define a simple situation where the strategy of the worker is independent of the suitability graph G

Definition 3 *The strategy of the worker is* simple *if the ranking of tasks is independent of the suitability graph G.*

When they use simple strategies, the workers continue to work on the same task until it is completed. Strategies are thus time-consistent. In the terminology of scheduling, there is no preemption.

The construction of MPE in simple strategies in the workers' game is reminiscent of the construction of a simple optimal policy for the planner, but we highlight two major differences. First, as every worker has a different set of tasks that she can work on at any graph, $T_i(G)$, even when simple strategies are employed, one needs to keep track of the number of agents who can work on every task in the state. Hence the state cannot be summarized by a vector $\mathbf{a}$ describing the number of workers capable of working on any task, independent of the identity of tasks, as in the planner's problem. Second, while the planner can choose the number of agents working on any task, the worker can only rank those tasks that she is capable of working on. Hence the relevant ranking of tasks for worker i is the ranking of the tasks in $H_i(G_0)$ even though her value V depends on the number of workers working on all tasks. This means that, as opposed to the planner's problem, making the ranking of agent i's task independent of the graph G is not sufficient to conclude that the value of the worker is monotonically decreasing in the number of agents who can accomplish any task.

In addition, in order to show that the dominant strategy of the workers in a MPE are simple, we need to impose two additional conditions on the suitability graph G_0 that we term Submodularity and Increasing Differences.

First we order the tasks so that if $a_j < a_k$ then $j < k$ and if $a_j = a_k$, $j < k$ if j precedes k in the exogenous ranking of tasks.

Definition 4 *The initial suitability graph G_0 satisfies Submodularity if, for any two consecutive tasks j and k, and any possible subset of tasks S of rank higher than k,*

$$a_{S \cup k} - a_S \geq a_{S \cup j \cup k} - a_{S \cup k}$$

Definition 5 *The initial suitability graph G_0 satisfies Increasing Differences if the following condition holds. Consider a task l and two tasks j and k of rank higher than l with $j < k$. Then for any possible subset of tasks S of rank higher than l excluding j and k,*

$$a_{S \cup l \cup k} - a_{S \cup k} \geq a_{S \cup l \cup j} - a_{S \cup j}.$$

Submodularity and Increasing Differences are conditions on the marginal effect of the addition of a new task on the set of agents who can perform a subset of easier tasks. Submodularity states that this effect is larger when the set of tasks is smaller. Increasing Differences states that the effect is larger when the subset of easier tasks contains a task which can be completed by a larger number of workers.

We check that the ranked tasks and generalist-specialists models both satisfy Submodularity and Increasing Differences. In the ranked tasks model, if $j < k$ and S is a subset of tasks easier than k, then $A_{S \cup j \cup k} = A_{S \cup j} = A_{S \cup k}$ so that $a_{S \cup k} - a_S = 0 = a_{S \cup j \cup k} - a_{S \cup k}$, and Submodularity holds. If $l < j$ and $l < k$, then $A_l \cup A_j = A_j$ and $A_l \cup A_k = A_k$. Hence $a_{S \cup l \cup j} - a_{S \cup j} = 0 = a_{S \cup l \cup k} - a_{S \cup k}$ for any set S of tasks of rank higher than l and Increasing Differences holds.

In the generalist-specialists model, $a_{S \cup k} = a_S + a_k - g$. So $a_{S \cup k} - a_S = a_k - g$ and $a_{S \cup j \cup k} - a_{S \cup k} = a_j - g$. So

$$a_{S \cup k} - a_S - (a_{S \cup j \cup k} - a_{S \cup k}) = a_k - a_j \geq 0,$$

and Submodularity holds. If $l < j < k$, then

$$\begin{aligned} a_{S \cup l \cup j} - a_{S \cup j} &= a_S + a_l + a_j - 2g - (a_S + a_j - g), \\ &= a_l - g, \\ &= a_S + a_l + a_k - 2g - (a_S + a_k - g), \\ &= a_{S \cup l \cup k} - a_{S \cup k}, \end{aligned}$$

and Increasing Differences holds.

We now prove the main Theorem of the Section, stating conditions under which the workers' game admits a MPE in simple strategies.

Theorem 2 *If the initial suitability graph G_0 satisfies Union Size Invariance, Submodularity and Increasing Differences, the worker's game admits a Markov Perfect Equilibrium where all workers adopt simple strategies and choose to work on the easiest tasks.*

Theorem 2 shows, under Union Size Invariance, Submodularity and Increasing Differences, the existence of a MPE of the worker's game where all workers adopt the same simple strategy. At any state $\mathbf{a} = (a_1, ..., a_m)$ characterized by the number of workers capable of working on the remaining tasks, agents choose to work on the "easiest task".

5 The planner's and workers' task allocations

Theorems 1 and 2 show that, when the conditions for simple solutions are strategies are satisfied, the task choices of the planner and the workers are diametrically opposed: the planner's optimal policy is to assign workers to tasks with the smallest number of workers, whereas in the equilibrium of the workers' game, all workers choose to work on the task with the largest number of workers. In this Section, we discuss the difference between the planner's and the workers' choices by (i) characterizing situations when the choices are identical, (ii) discussing a policy where the planner differentiates the rewards to the tasks, (iii) establishing comparative statics on the effect of the number of workers on the expected completion time and (iv) providing a bound for the price of anarchy in the ranked tasks model.

5.1 Coincidence of the planner's and the workers' task allocations

We first analyze situations where the solution to the planner's problem and the equilibrium strategies in the worker's game are identical. For the two problems to have the same solution, the planner and the workers must be indifferent among all the tasks at every state. This requires the number of agents to be identical across tasks at any state. The following Proposition characterizes those initial suitability graphs G_0 for which the condition holds.

Proposition 1 *The optimal policy of the planner results in the same assignment as an MPE of the planner's game if and only if $a_j = a_k$ for all tasks j, k and for all collections of $k \leq m$ elements $\{A_1, ..., A_k\}$ and $\{B_1, ..., B_k\}$ in $\mathcal{A}$, $| \cap_{j=1}^{k} A_j| = | \cap_{j=1}^{k} B_j|$.*

Proposition 1 shows that the suitability graph G_0 needs to satisfy a very strong symmetry condition for the solution to the planner's problem to be equal to the MPE of the workers' game. This strong symmetry condition is satisfied by the complete bipartite network, where all intersections of k sets are equal to the set of all workers. It is also satisfied in the specialists-generalist model when all tasks have the same number of specialists. In that case, any intersection of k sets is equal to the set of generalists.

5.2 Differential rewards

In this subsection, we study the effect of a policy by which the planner offers different rewards for the different tasks. We first consider *contingent rewards* which are conditional on the current suitability graph G. A contingent reward scheme defines, for every suitability graph G, a vector $(\rho_1(G), ..., \rho_j(G), ..., \rho_{t(G)}(G))$ of rewards for all the tasks $t_j = t_1, ..., t_{t(G)}$ in $T(G)$. In the worker's game, the utility of worker i choosing a probability vector $\mathbf{p}$ is then given by:

$$u_i(G,\mathbf{p}) = \sum_{j \in T(G)} \frac{p_i^j \rho_j(G)}{w(G)} + \sum_{j \in T(G)} \frac{\sum_k p_k^j}{w(G)} U_i(G-j), \tag{4}$$

As the worker's payoff is linear in p_i^j, every worker who can work on multiple tasks has a weakly dominant strategy and chooses the task for which the sum $\rho_j(G) + U_i(G-j)$ is the highest.

Lemma 5 *In a Markov Perfect Equilibrium, if $p_i^j > 0$, then*

$$\rho_j(G) + U_i(G-j) \geq \rho_k(G) + U_i(G-k) \forall k \in H_i(G).$$

Consider a suitability graph for which the solution to the planner and the strategies of the workers are simple, and order the tasks so that $i < j$ implies that $a_i \leq a_j$.

Define the payoffs recursively. When $t(G) = 1, \rho_j(G) = 1$.

For any G with $t(G) > 1$, let $\rho_{t(G)}(G) = 1$ and for any $j = 1, ..., t(G) - 1$, let

$$\rho_j(G) = \max_{k>j] \max_i \ \{j,k \in H_i} \rho_k(G) + U_i(G-k) - U_i(G-j)$$

The definition of the contingent rewards thus follows a double recursion (i) on the number of tasks in $T(G)$ and (ii) on the tasks, starting with the task with the largest number of workers. It is easy to check that, with the given reward scheme, any worker i works on the task with the smallest index. Hence *there exists a contingent reward scheme which guarantees that all workers work on the optimal task chosen by the planner.*

We next look for *non-contingent* rewards, which are chosen ex ante and do not depend on the current suitability graph. A *non-contingent* reward scheme is a vector $(\rho_1, ..., \rho_m)$ of rewards for all tasks in T. We say that task j is *exclusive* if there is a single worker who can complete the task, $a_j = 1$.

Proposition 2 *There exists a non-contingent reward scheme if and only if there is no exclusive task.*

Proposition 2 shows that the planner can give incentives to the workers to select his favorite task allocation, by assigning higher non-contingent rewards to harder tasks. However, Proposition 2 only shows that one can construct a finite reward scheme, where the ratios between the rewards $\frac{\rho_j}{\rho_k}$ are bounded. The proof does not provider any guidance on the mimimal rewards which can be used.

To get a better grasp of the minimal rewards that the planner can use, we turn to the ranked task model, and obtain the following characterization of non-contingent rewards implementing the planner's optimal task allocation when $m = 3$.

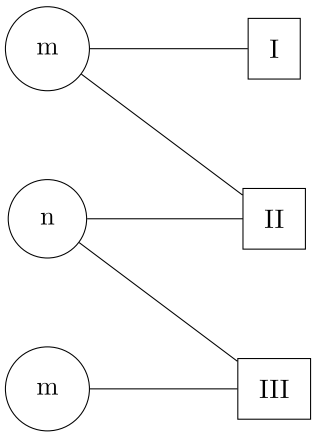


Figure 2: An example of suitability graph

Proposition 3 *Consider the ranked task model with $m = 3$. If task t_1 is not exclusive, the planner can implement her first-best task allocation by selecting*

$$\begin{aligned}\rho_1 &= \rho_2 \frac{a_1(a_2-1)}{a_2(a_1-1)},\\ \rho_2 &= \rho_3 \frac{a_2^2(a_3^2+2a_2-a_1)}{a_3^2(a_2^2+2a_3-a_1)},\\ \rho_3 &= 1.\end{aligned}$$

If task t_1 is exclusive, the planner cannot implement her first-best task allocation.

Proposition 3 provides exact values on the minimal ratios between rewards which give incentives to the workers to select the planner's optimal task allocation. Unfortunately, the computations cannot easily be extended beyond three tasks.

5.3 Comparative statics of the number of workers

We now show that the misalignment of the incentives of the planner and of the workers can lead to counter-intuitive comparative statics effects. While an increase in the number of workers always results in a decrease in the expected time of completion in the planner's problem, *it may lead to an increase in the expected time of completion in the MPE of the workers' game.* This striking result is illustrated in the following example.

Example 1 *Suppose that there are three tasks and the suitability graph is given by Figure 2*

There are m agents who can work on tasks 1 and 2, n agents on tasks 1 and 3 and m agents only on task 3. We have $a_1 = m < a_2 = a_3 = m + n$. We suppose that the exogenous ranking of tasks is given by

$3 \prec 2 \prec 1$.[6]

Consider an agent who can work on tasks 1 and 2. After task 1 is completed, workers who can work on both tasks are indifferent between tasks 2 and 3 and, by the tie-breaking rule, choose to work on task 3. We thus have

$$U_1(-1) = \frac{1}{2m+n} + \frac{m+n}{2m+n}\frac{1}{m+n}.$$

After task 2 is completed, $m+n$ workers work on task 3 and m workers on task 1, and hence the expected utility of a worker who can work on 1 is given by

$$U_1(-2) = \frac{1}{2m+n} + \frac{m+n}{2m+n}\frac{1}{m}.$$

So the workers who can work on 1 and 2 prefer to work on 2.

Consider next a worker who can work both on 2 and 3. After 2 is completed, she gets an expected payoff

$$U_2(-2) = \frac{1}{2m+n} + \frac{m}{2m+n}\frac{1}{n+m} = \frac{1}{n+m},$$

whereas, after 3 is completed, all workers work on task 2 and she obtains an expected payoff

$$U_2(-3) = \frac{1}{n+m}.$$

So, workers are indifferent between working on tasks 2 and 3 and, by the exogenous ranking condition, choose to work on task 2.

The expected time of completion of the project is thus given by:

$$\begin{aligned} V &= \frac{1}{2m+n} + \frac{m+n}{2m+n}[\frac{1}{2m+n} + \frac{m}{2m+n}\frac{1}{m+n} + \frac{m+n}{2m+n}\frac{1}{m}] \\ &+ \frac{m}{2m+n}[\frac{1}{m+n} + \frac{1}{m}] \end{aligned}$$

Next, suppose that there is one additional worker who can work on task 3, so that the suitability graph is now given by Figure 3

We compute the expected utilities of an agent who can work on two tasks after each task has been

[6]Notice that the suitability graph satisfies Union Size Invariance, Submodularity and Increasing Differences.

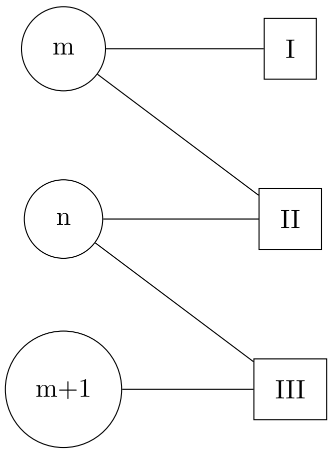


Figure 3: An example of suitability graph with an additional worker

completed as

$$
\begin{aligned}
U_1(-2) &= \frac{1}{2m+n+1} + \frac{m+n+1}{2m+n+1}\frac{1}{m} \\
&> \frac{1}{2m+n+1} + \frac{m+n+1}{2m+n+1}\frac{1}{m+n} \\
&= U_1(-1), \\
U_2(-3) &= \frac{1}{m+n} \\
&> \frac{1}{2m+n+1} + \frac{m}{2m+n+1}\frac{1}{m+n+1} \\
&= U_2(-2)
\end{aligned}
$$

Hence, workers who can work on tasks 1 and 2 still work on 2 while workers who can work both on 2 and 3 are no longer indifferent, and strictly prefer to work on task 3.

This results in an expected time of completion given by

$$
\begin{aligned}
V' &= \frac{1}{2m+n+1} + \frac{m}{2m+n+1}[\frac{1}{2m+n+1} + \frac{m}{2m+n+1}\frac{1}{m+n+1} + \frac{m+n+1}{2m+n+1}\frac{1}{m}] \\
&+ \frac{n+m+1}{2m+n+1}[\frac{1}{m+n+1} + \frac{1}{m}]
\end{aligned}
$$

It is easy to check that for any $m \leq n$, $V' > V$. Increasing the number of agents results in an increase in the expected time of completion of the project and hence a *decrease* in the value of the planner. To understand this result, note that, after task 3 is completed, the planner faces a long expected completion time as all $m+n+1$ remaining workers first work on task 2, and then m workers on task 1. By contrast, if task 2 is completed, the $2m+n+1$ workers simultaneously work on both tasks, m on task 1 and

$m+n+1$ on task 3. The addition of a single worker who can work on 3 yields a drastic change in the behavior of the n workers who move from working on task 2 to working on task 3. Even though there is one additional worker, this drastic change in the workers' behavior results in an increase in the expected time of completion of the entire project.

5.4 Price of anarchy

We can measure the efficiency loss in the worker's game by the *price of anarchy*, the ratio of the expected completion time under the planner's optimal policy and in the workers' game. When there are only two tasks, we let B_1, B_2 and B_{12} denote the set of workers who can work only on task 1, only on task 2 and on both tasks, with cardinality b_1, b_2 and b_{12} respectively, and we let $\beta_i = \frac{b_i}{n}$ denote the proportion of workers in each of the three sets. The price of anarchy is given by

$$PA = \lim_{n\to\infty} \max_{b_1,b_2,b_{12}|b_1+b_2+b_{12}=n} \frac{ET_w(n)}{ET_p(n)}$$

where $ET_p(n)$ and $ET_w(n)$ denote the expected completion time of the project in the planner's optimal policy and in the worker's game. Let $b_1 \geq b_2$. By Lemma 2, the planner assigns workers in B_{12} to work on task 2 while in the workers' game, these workers work on task 1. Hence

$$\frac{ET_w(n)}{ET_p(n)} = \frac{1+(b_1+b_{12})/(b_2+b_{12})+b_2/(b_1+b_{12})}{1+(b_2+b_{12})/(b_1+b_{12})+b_1/(b_2+b_{12})}$$

Proposition 4 *When $m=2$, the price of anarchy is given by*

$$PA = \frac{1}{2(\sqrt{2}-1)} \approx 1.207$$

Proposition 4 shows that the misalignment of the workers and planner's incentives can result at most in an additional expected time of completion of 20% when there are two tasks. When the number of tasks increases, the expected loss goes up, but it becomes impossible to obtain an exact formula for the price of anarchy. However, in the ranked tasks model, we obtain an upper bound on the price of anarchy, for any number m of tasks:

Proposition 5 *In the ranked tasks model, the price of anarchy satisfies*

$$PA \leq \max_{\alpha\in[0,1]} \frac{m-1+\alpha}{\alpha m+(1-\alpha)^m}.$$

The upper bound (which is exact when the number of tasks is equal to 2), is not very tight when the number of tasks increases. However, as $\alpha m + (1-\alpha)^m \geq 1$ for all m, α, we have

$$PA \leq m - 1 + \alpha,$$

establishing that the price of anarchy is linear in the number of tasks. Hence, as the number of tasks grows, the price of anarchy grows at a rate which is at most linear in the number of tasks.

6 Conclusions

In this paper, we study a dynamic stochastic task allocation problem when there is a fixed bipartite network associating workers to tasks. We analyze the optimal policy of a planner whose objective is to minimize the expected time of completion of all tasks, and a game played by workers who independently choose their tasks and are rewarded each time a task is completed. We show that both the planner's and the worker's problems are NP-hard and characterize networks for which the planner's and workers' policies are stationary. When policies are stationary, the planner prefers the workers to start with the hardest tasks, whereas workers always prefer to start with easier tasks. Examples of situations where policies are stationary are nested tasks (where tasks can be ranked by difficulty and workers by skills) and generalist-specialists models where some workers can work on all tasks and other workers are specialized on a single task. We show that the policy of the planner and the outcome of the workers' game only coincide when the bipartite network satisfies a strong symmetry condition. Differential rewards can be used to implement the planner's optimal task allocation and we show that non-contingent rewards, which are independent of the set of remaining tasks, can be used as long as there is no task that a single agent can complete.

Our analysis thus provides a strong contrast between the outcome of the centralized optimization problem of the planner and the outcome of the decentralized game of task choice of the workers. However, we are aware that our analysis relies on strong assumptions on the model. What happens if workers are heterogeneous and have different possibly complementary skills? What if some tasks take more time than others? What happens if, in addition to the task choice, workers endogenously choose their level of effort? We plan to tackle these important questions in future research.

References

Davis, Robert I and Alan Burns (2011) "A survey of hard real-time scheduling for multiprocessor systems," *ACM computing surveys (CSUR)*, Vol. 43, No. 4, pp. 1–44.

Di Stefano, Giada and Maria Rita Micheli (2022) "To Stem the Tide: Organizational Climate and the Locus of Knowledge Transfer," *Organization Science*.

Doval, Laura (2018) "Whether or not to open Pandora's box," *Journal of Economic Theory*, Vol. 175, pp. 127–158.

Eliaz, Kfir, Daniel Fershtman, and Alexander Frug (2021) "On the Optimal Scheduling of Attention,"Technical report, CEPR Discussion Paper No. DP16364.

Fershtman, Daniel and Alessandro Pavan (2022) "Searching for arms: Experimentation with endogenous consideration sets,"Technical report, Mimeo.

Khamis, Alaa, Ahmed Hussein, and Ahmed Elmogy (2015) "Multi-robot task allocation: A review of the state-of-the-art," in Anis Koubâa and J. Ramiro Martínez-de Dios eds. *Cooperative robots and sensor networks 2015*: Springer, pp. 31–51.

Leung, Joseph Y-T and Chung-Lun Li (2008) "Scheduling with processing set restrictions: A survey," *International Journal of Production Economics*, Vol. 116, No. 2, pp. 251–262.

——— (2016) "Scheduling with processing set restrictions: A literature update," *International Journal of Production Economics*, Vol. 175, pp. 1–11.

Loch, Christoph H, Christian Terwiesch, and Stefan Thomke (2001) "Parallel and sequential testing of design alternatives," *Management Science*, Vol. 47, No. 5, pp. 663–678.

Papadimitriou, Christos H and John N Tsitsiklis (1987) "The complexity of Markov decision processes," *Mathematics of operations research*, Vol. 12, No. 3, pp. 441–450.

Pinedo, Michael L (2016) *Scheduling*, Vol. 29: Springer.

Pinedo, Michael and Josh Reed (2013) "The Least flexible job first rule in scheduling and in queueing," *Operations Research Letters*, Vol. 41, No. 6, pp. 618–621.

Rizk, Yara, Mariette Awad, and Edward W Tunstel (2019) "Cooperative heterogeneous multi-robot systems: A survey," *ACM Computing Surveys (CSUR)*, Vol. 52, No. 2, pp. 1–31.

Vishwanath, Tara (1992) “Parallel search for the best alternative,” *Economic Theory*, pp. 495–507.

Weitzman, Martin L (1979) “Optimal search for the best alternative,” *Econometrica*, pp. 641–654.

7 Appendix A: Proofs

Proof of Theorem 1

In the first part of the proof, we show that, under Union Size Invariance, the vector $\mathbf{a} = (a_1, ..., a_m)$, which denotes the number of workers who can work on each task in $T(G_0)$, is sufficient to describe any suitability graph G. Any state in the Markov decision process is generated by a subvector of $\mathbf{a}$. We first prove the following Lemma, showing the existence of a family of functions g_k where k denotes the number of tasks (or the dimension of a subvector of $\mathbf{a}$).

Lemma 6 *Under Union Size Invariance, there exists a family of functions $g_k : \Re^k \to \Re$ for $k \in \{1, ..., m\}$ that are symmetric and weakly increasing in all their arguments such that, for any set of k tasks $S = \{t_1, t_2, .., t_k\}$ in $\mathcal{T}$,*

$$a_S = g_k(a_1, ..., a_k).$$

Proof: Let $\mathcal{A}$ denote the collection of all sets A_j for $j \in T(G_0)$. We first show that the function g_k is uniquely defined for the values $(a_1, ..., a_k) \in \Re^k$ which can be generated by k subsets of $\mathcal{A}$. Formally, let

$$D^k = \{(a_1, ..., a_k) \in N^k | \exists \{A_1, ..., A_k\} \subseteq \mathcal{A}, |A_j| = a_j \forall j \in \{1, .., k\}\}.$$

Hence, for any given G_0, D^k denotes the set of all vectors formed of the cardinals of k sets in $\mathcal{A}$. We will show that g_k is uniquely defined on D^k.

To this end, consider two distinct collections of sets $\{A_1, ..., A_k\} \subseteq \mathcal{A}$, and $\{B_1, ..., B_k\} \subseteq \mathcal{A}$ corresponding to different tasks such that $a_j = b_j$ for all $j = 1, ..., k$. We consider successive changes from $\{a_1, ..., a_k\}$ to $\{b_1, ..., b_k\}$. Consider first $\{a_1, a_2, .., a_k\}$ and $\{b_1, a_2, ..., a_k\}$. Let t_1 and t_2 be the tasks corresponding to a_1 and b_1. Applying Union Size Invariance by taking $a_1 = b_1$ and the union S of tasks corresponding to $a_2, .., a_k$

$$g_k(a_1, .., a_k) = a_{S \cup 1} = a_{S \cup 2} = g_k(b_1, a_2, ..., a_k).$$

Repeating the argument successively, we obtain

$$g_k(a_1, ..., a_k) = g_k(b_1, ..., b_k).$$

This shows that for any two collection of tasks which result in the same vector in D^k, the function g^k assigns the same value.

Next observe that, because the union of sets is commutative, for any permutation σ, $S = \cup_{j=1}^{k} A_j = \cup_{j=1}^{k} A_{\sigma(j)}$ so that

$$g_k(a_1, .., a_k) = a_S = g_k(a_{\sigma(1)}, ..., a_{\sigma(k)}),$$

and the function g_k is symmetric.

Finally, consider two collections of k sets $\{A_1, ..., A_k\}, \{B_1, ..., B_k\}$ where $|A_j| = |B_j|$ for all $j > 1$ and $a_1 < b_1$. Let t_1 be the task corresponding to a_1, t_2 the task corresponding to b_1. By Union Size Invariance, taking the union of tasks $S = \cup_{j \neq J} t_j$, we have

$$g_k(a_1, ..., a_k) = a_{S \cup 1} \leq a_{S \cup 2} = g_k(b_1, a_2, ..., a_k).$$

The argument can be repeated for any argument of the function g^k showing that the function g^k is weakly increasing in all its arguments. ■

Lemma 6 constructs a collection of functions $g_k : \Re^k \to \Re$ which we now use to compute the planner's expected value. Notice that the functions g_k are only uniquely defined for collections of integers which can be generated by the cardinals of the sets in $\mathcal{A}$,

$$D^k = \{(a_1, ..., a_k) \in N^k | \exists \{A_1, ..., A_k\} \subseteq \mathcal{A}, |A_j| = a_j \forall j \in \{1, .., k\}\}.$$

For any vector $(x_1, .., x_k) \in \Re^k \setminus D^k$, Lemma 6 does not place any restriction on the value of the function g_k.

We now use Lemma 6 to simplify the planner's problem and write it only as a function of the number of agents who can work on every target a_k, and not as a function of the number of agents who can work on any collection of targets a_S for $S \in \mathcal{T}$. The planner's policy can thus be conditioned on the vector $\mathbf{a}$ rather than on the feasibility graph G. We now consider as a state space the set of all subvectors of the vector $\mathbf{a}$, and we write the Bellman equation as:

$$V(\mathbf{a}) = -\frac{1}{\lambda g_m(\mathbf{a})} + \max_{\mathbf{r}} \sum_{j \in T(\mathbf{a})} \frac{r_j}{g_m(\mathbf{a})} V(a_{-j}). \tag{5}$$

In the second part of the proof, we will show that the value $V(\mathbf{a})$ is increasing in all the components of the vector $\mathbf{a}$.

We next show that the expected time of completion of all tasks is decreasing after a task has been completed or, alternatively, that the expected value of the planner is always higher after the completion of any task

Lemma 7 *For any vector* $\mathbf{a}$ *and any target* $j \in T(\mathbf{a})$, $V(\mathbf{a}) \leq V(a_{-j})$.

Proof: The proof is by induction on the number of tasks. Clearly if $m = 1$, $V(a_{-1}) = 0 > V(\mathbf{a})$.

Now consider a problem with m tasks and let $(r_1, ..., r_m)$ be the optimal action of the planner. Suppose that task j has been completed, and consider the following action for the planner at state a_{-j}:

- For any task $k \neq j$, let r_k agents work on task j
- All r_j agents who initially worked on task k are now assigned to a "useless" task with value 0.

The value of this action $\mathbf{r}$ is given by

$$v(a_{-j}, \mathbf{r}) = -\frac{1}{\lambda g_m(\mathbf{a})} + \sum_{k \neq j} \frac{r_k}{g_m(\mathbf{a})} V(a_{-k,j}) + \frac{r_j}{g_m(\mathbf{a})} V(a_{-j}).$$

By the induction hypothesis, $V(a_{-k,j}) \leq V(a_{-j})$. Hence

$$\begin{aligned} V(\mathbf{a}) &= -\frac{1}{\lambda g_m(\mathbf{a})} + \sum_j \frac{r_j}{g_m(\mathbf{a})} V(a_{-j}) \\ &\leq -\frac{1}{\lambda g_m(\mathbf{a})} + \sum_{k \neq j} \frac{r_k}{g_m(\mathbf{a})} V(a_{-k,j}) + \frac{r_j}{g_m(\mathbf{a})} V(a_{-j}) \\ &= v(a_{-j}, \mathbf{r}) \\ &\leq V(a_{-j}) \end{aligned}$$

where the last inequality is due to the fact that $\mathbf{r}$ is not necessarily the optimal action of the planner at a_{-j}. ■

We are now ready to prove that $V(\mathbf{a})$ is increasing in all its arguments. The proof is by induction on the number of tasks. If there is only one task

$$V(a) = -\frac{1}{\lambda a}$$

is clearly increasing in a. Now suppose that for any $\mathbf{a}$ with $m - 1$ tasks, $V(\mathbf{a})$ is increasing in all its arguments and consider a problem with m tasks.

After any task j has been completed, the problem faced by the planner is a problem with $m - 1$ tasks, which is thus increasing in all its arguments. Consider then two tasks j, k with $a_j < a_k$. Because

$a_{-j} > a_{-k}$, $V(a_{-j}) > V(a_{-k})$. By Lemma 1, this implies that the planner ranks the tasks according to the number of workers who can perform them and that the optimal policy of the planner is to assign workers to the hardest tasks first

Hence, for any j, r_j is the number of agents who can perform task j but not any of the hardest tasks. Ranking the tasks according to the number of agents who can perform them, with $a_1 \leq a_2 \leq a_3 ... \leq a_m$, we thus obtain

$$r_j = |A_1 \cup ...A_j| - |A_1 \cup ...A_{j-1}| = g_j(a_1, ..a_j) - g_{j-1}(a_1, .., a_{j-1}).$$

We thus obtain

$$V(\mathbf{a}) = -\frac{1}{\lambda g_m(\mathbf{a})} + \sum_{j=1}^{m} \frac{g_j(a_1, ..., a_j) - g_{j-1}(a_1, ..., a_{j-1})}{g_m(\mathbf{a})} V(a_{-j}).$$

Now consider two vectors $\mathbf{a} = (a_1, ..., a_m)$ and $\mathbf{b} = (b_1, ..b_m)$ where $\mathbf{a} < \mathbf{b}$, i.e. such that $a_j \leq b_j$ for all j and $a_k < b_k$ for some k.

$$\begin{aligned}
V(\mathbf{a}) - V(\mathbf{b}) &= -\frac{1}{\lambda g_m(\mathbf{a})} + \frac{1}{\lambda g_m(\mathbf{b})} + \sum_{j=1}^{m} \frac{g_j(a_1, .., a_j) - g_{j-1}(a_1, ..., a_{j-1})}{g_m(\mathbf{a})} V(a_{-j}) \\
&- \sum_{j=1}^{m} \frac{g_j(b_1, .., b_j) - g_{j-1}(b_1, ..., b_{j-1})}{g_m(\mathbf{b})} V(b_{-j}) \\
&= -\frac{1}{\lambda g_m(\mathbf{a})} + \frac{1}{\lambda g_m(\mathbf{b})} + \sum_{j=1}^{m} \frac{g_j(a_1, .., a_j) - g_{j-1}(a_1, ..., a_{j-1})}{g_m(\mathbf{a})} V(a_{-j}) \\
&- \sum_{j=1}^{m} \frac{g_j(a_1, .., a_j) - g_{j-1}(a_1, ..., a_{j-1})}{g_m(\mathbf{b})} V(a_{-j}) + \sum_{j=1}^{m} \frac{g_j(a_1, .., a_j) - g_{j-1}(a_1, ..., a_{j-1})}{g_m(\mathbf{b})} V(a_{-j}) \\
&- \sum_{j=1}^{m} \frac{g_j(b_1, .., b_j) - g_{j-1}(b_1, ..., b_{j-1})}{g_m(\mathbf{b})} V(b_{-j}) \\
&= V(\mathbf{a})[1 - \frac{g_m(\mathbf{a})}{g_m(\mathbf{b})}] + \sum_{j=1}^{m} \frac{g_j(a_1, .., a_j) - g_{j-1}(a_1, ..., a_{j-1})}{g_m(\mathbf{b})} V(a_{-j}) \\
&- \sum_{j=1}^{m} \frac{g_j(b_1, .., b_j) - g_{j-1}(b_1, ..., b_{j-1})}{g_m(\mathbf{b})} V(b_{-j}).
\end{aligned}$$

As $a_{-j} \geq b_{-j}$ for all j (with strict inequality for some k), and a_{-j} and b_{-j} are vectors of dimension

$m-1$, by the induction hypothesis, $V(a_{-j}) \geq V(b_{-j})$ for all j with strict inequality for some k. Hence

$$\begin{aligned} V(\mathbf{a}) - V(\mathbf{b}) &> V(\mathbf{a})[1 - \frac{g_m(\mathbf{a})}{g_m(\mathbf{b})}] + \sum_{j=1}^{m} \frac{g_j(a_1,..,a_j) - g_{j-1}(a_1,...,a_{j-1})}{g_m(\mathbf{b})} V(b_{-j}) \\ &- \sum_{j=1}^{m} \frac{g_j(b_1,..,b_j) - g_{j-1}(b_1,...,b_{j-1})}{g_m(\mathbf{b})} V(b_{-j}) \end{aligned}$$

Next, we develop

$$\sum_{j=1}^{m} \frac{g_j(a_1,..,a_j) - g_{j-1}(a_1,...,a_{j-1})}{g_m(\mathbf{b})} V(b_{-j}) = \sum_{j=1}^{m-1} \frac{g_j(a_1,..,a_j)}{g_m(\mathbf{b})} (V(b_{-j}) - V(b_{-(j+1)}) + \frac{g_m(\mathbf{a})}{g_m(\mathbf{b})} V(b_{-m}),$$

so that

$$\begin{aligned} V(\mathbf{a}) - V(\mathbf{b}) &> V(\mathbf{a})(1 - \frac{g_m(\mathbf{a})}{g_m(\mathbf{b})}) + \sum_{j=1}^{m-1} \frac{(g_j(a_1,..,a_j) - g_j(b_1,...,b_j)}{g_m(\mathbf{b})} (V(b_{-j}) - V(b_{-(j+1)}) \\ &+ \frac{(g_m(\mathbf{a}) - g_m(bfb)}{g_m(\mathbf{b})} v(b_{-m}) \end{aligned}$$

Now, because the function g_j is weakly increasing in all its arguments and $(b_1,...,b_j) \leq (a_1,...,a_j)$ for all j, $g_j(a_1,..,a_j) - g_j(b_1,...,b_j) \geq 0$. In addition, as all tasks are ordered by increasing number of agents who can perform them, $b_{j+1} \geq b_j$ so that $b_{-j} \geq b_{-(j+1)}$. By the induction hypothesis, $V(b_{-j}) \geq V(b_{-(j+1)})$ (with strict inequality for some k), so that

$$V(\mathbf{a}) - V(\mathbf{b}) > V(\mathbf{a})(1 - \frac{g_m(\mathbf{a})}{g_m(\mathbf{b})}) - (1 - \frac{(g_m(\mathbf{a})}{g_m(\mathbf{b})}) V(b_{-m}).$$

Hence

$$V(\mathbf{a}) - V(\mathbf{b}) > (1 - \frac{g_m(\mathbf{a})}{g_m(\mathbf{b})})(V(\mathbf{a}) - V(\mathbf{b}) + V(\mathbf{b}) - V(b_{-m}))$$

so that

$$V(\mathbf{a}) - V(\mathbf{b}) > \frac{\frac{g_m(\mathbf{a})}{g_m(\mathbf{b})} - 1}{1 + \frac{g_m(\mathbf{a})}{g_m(\mathbf{b})}} (V(b_{-m}) - V(\mathbf{b})).$$

By Lemma 7, $V(b_{-m}) - V(\mathbf{b}) > 0$ so that

$$V(\mathbf{a}) - V(\mathbf{b}) > 0,$$

showing that V is increasing in all its arguments. At any state $\mathbf{a}$, if $a_j < a_k$ then $a_{-j} > a_{-k}$ and hence $V(a_{-j}) > V(a_{-k})$. This shows that the optimal action is to assign a worker to the hardest task at any state **a**, and completes the proof of the Theorem. ■

Proof of Theorem 2

The proof of Theorem 2 follows the same structure as the proof of Theorem 1. We first use Union Size Invariance and Strong Union Size Difference Invariance to construct functions $g_k(a_1, ..., a_k)$ which generate the number of workers working on any subset S of k tasks and satisfies two monotonicity properties. In a second step of the proof, we show that the value of a worker can never decrease when the number of tasks increases. Finally, we show that the value of a worker is higher when she completes a task with a larger number of workers. This last step guarantees that a worker prefers to start with the easiest task, as this results in a smaller number of competitors in subsequent periods.

As Union Size Invariance holds, there exist functions g_k which are symmetric and increasing such that, for any collection S of k tasks $S = 1, 2, ..., k$,

$$a_S = g_k(a_1, .., a_k).$$

Submodularity implies that, for any ordered set of tasks $(a_1, ..., a_m)$

$$g_{m-j+1}(a_j, ..., a_m) - g_{m-j}(a_{j+1}, ..., a_m) \leq g_{m-j}(a_{j+1}, ..., a_m) - g_{m-j-1}(a_{j+2}, ..., a_m).$$

By a repeated application of this inequality, for any two tasks j, k with $j < k$,

$$g_{m-j+1}(a_j, ..., a_m) - g_{m-j}(a_{j+1}, ..., a_m) \leq g_{m-k+1}(a_k, ..., a_m) - g_{m-k}(a_{k+1}, ..., a_m).$$

Finally, by Increasing Differences, for any fixed task l, any two tasks j, k with $a_k, a_l \geq a_j$ and any set of s tasks S such that $a_p \geq a_l$ for all $p \in S$, we have:

$$g_{s+2}(a_l, a_j, a_S) - g_{s+1}(a_j, a_S) \leq g_{s+2}(a_l, a_k, a_S) - g_{s+1}(a_k, a_S).$$

By a repeated application of this inequality, for any fixed task l and any two vectors **a**, **b** of dimension k such that $b_p \geq a_p \geq a_j$ for all $p \in S$, we obtain

$$g_{k+1}(a_j, \mathbf{a}) - g_k(\mathbf{a}) \leq g_{k+1}(a_j, \mathbf{b}) - g_k(\mathbf{b}) \tag{6}$$

Next, we consider a worker i and, in order to ease notation, denote her value U dispensing with the index

i. We first show that, if workers play simple strategies, the workers' utility is weakly higher at G than after one task has been completed.

Lemma 8 *For any suitability graph G and any target $j \in T(G)$, when all workers play simple strategies at G, $U(G) \geq U(G-j)$.*

Proof: The proof is by induction on the number of tasks. If G has only one task, $U(G-1) = 0 \leq U(G)$. Suppose that G has m tasks and that the inequality holds for any G with less than m tasks.
If the worker is inactive at $G-j$, then $U(G) \geq 0 = U(G-j)$. So suppose that the worker is active at $G-j$. Let $(r_1, ..., r_m)$ denote the number of workers who work on each task in the MPE at G. Furthermore let $r_{j,k}$ denote the number of workers who initially work on task j and move to task k after j has been completed. Because all strategies are simple, the ranking of tasks at any suitability graph G is given by the ranking when there are only two tasks left. Hence, at G, every agent works on the easiest task. We conclude that, if a worker works on task $k \neq j$ at G, she will continue to work on task k after j has been completed at $G-j$. The number of agents working on task $k \neq j$ at $G-j$ is thus $r_k + r_{j,k}$. We now compute

$$\begin{aligned}
U(G) - U(G-j) &= \frac{1}{g_m(\mathbf{a})} + \sum_k \frac{r_k}{g_m(\mathbf{a})} U(G-k) - \frac{1}{g(a_{-j})} - \sum_{k \neq j} \frac{r_k + r_{j,k}}{g_{m-1}(a_{-j})} U(G-j,k), \\
&= U(G)(1 - \frac{g_m(\mathbf{a})}{g_{m-1}(a_{-j})}) + \frac{1}{g_{m-1}(a_{-j})} + \sum_k \frac{r_k}{g_{m-1}(a_{-j})} U(G-k) \\
&- \frac{1}{g(a_{-j})} - \sum_{k \neq j} \frac{r_k + r_{j,k}}{g_{m-1}(a_{-j})} U(G-j,k) \\
&= U(G)(1 - \frac{g_m(\mathbf{a})}{g_{m-1}(a_{-j})}) + \sum_k \frac{r_k}{g_{m-1}(a_{-j})} U(G-k) + \frac{r_j}{g_{m-1}(a_{-j})} U(G-j) \\
&- \sum_{k \neq j} \frac{r_k}{g_{m-1}(a_{-j})} U(G-j,k) - \sum_{k \neq j} \frac{r_{j,k}}{g_{m-1}(a_{-j})} U(G-j,k).
\end{aligned}$$

By the induction hypothesis, $U(G-k) \geq U(G-k,j)$ for all k. Hence

$$U(G) - U(G-j) \geq U(G)(1 - \frac{g_m(\mathbf{a})}{g_{m-1}(a_{-j})}) + \frac{r_j}{g_{m-1}(a_{-j})} U(G-j) - \sum_{k \neq j} \frac{r_{j,k}}{g_{m-1}(a_{-j})} U(G-j,k).$$

Applying again the induction hypothesis,

$$U(G) - U(G-j) \geq U(G)(1 - \frac{g_m(\mathbf{a})}{g_{m-1}(a_{-j})}) + \frac{r_j - \sum_{j \neq k} r_{j,k}}{g_{m-1}(a_{-k})} U(G-j).$$

Now notice that the number of workers who become inactive after task j is completed is given by

$$g_m(\mathbf{a}) - g_{m-1}(a_{-j}) = r_j - \sum_{k \neq j} r_{j,k},$$

Hence

$$U(G) - U(G-j) \geq U(G)(1 - \frac{g_m(\mathbf{a})}{g_{m-1}(a_{-j})}) - (1 - \frac{g_m(\mathbf{a})}{g_{m-1}(a_{-j})})U(G-j).$$

so that

$$(U(G) - U(G-j)) frac g_m(\mathbf{a}) g_{m-1}(a_{-j})) \geq 0,$$

completing the proof of the Lemma. ■

Because workers can work on different tasks, we need to keep track of those tasks on which worker i can work in any suitability graph. Hence, as opposed to the planner's problem, the relevant state for worker i at G is not the unordered vector of the number of agents who can work on all remaining tasks, but the ordered number of workers who can work on each and every task. Let $\mathcal{A} = \{A_1, ..., A_m\}$ be the set of workers who can work on each task at G_0. We fix an ordering of tasks. We suppose that $\mathcal{A}$ is ordered so that whenever $a_j < a_k$, $j < k$ and when $a_j = a_k$, $j < k$ if j precedes k in the given, exogenous ranking of tasks, which is used to break ties.

When agents adopt simple strategies and always work on the easiest task first, we can characterize the state (the suitability graph G) by a vector of dimension m, $\mathbf{a}(a_1, ..., a_m)$ where $a_j \in \{0, a_j\}$. We interpret $a_j = 0$ to mean that task j has been completed. The vector $\mathbf{a}$ thus keeps track of the identity of the tasks which are left and the number of workers who can work on each of the remaining tasks. Given the ordering of tasks, whenever $j \leq k$ and $a_j > 0, a_k > 0$ we must have $a_j \leq a_k$. As before, for any task j such that $a_j > 0$, we denote by a_{-j} the suitability graph obtained after task j has been completed, i.e. the vector obtained from $\mathbf{a}$ by replacing a_j with 0.

In order to show that worker i always wants to work on the easiest task at G, we consider two tasks on which worker i can work, j and k with $j < k$ which are consecutive for i, in the sense that there is no other task in $\{j+1, ..k-1\}$ on which worker i can work. The strategy of the proof is to show that $U(a_{-k}) \geq U(a_{-j})$, so that agent i always prefers to work on task k.

The proof is by induction on the number of tasks. Suppose that there are only two tasks. If worker i can work on both tasks, $U(a_{-2}) = U(a_1, 0) = \frac{1}{a_1} \geq \frac{1}{a_2} = U(0, a_2) = U(a_{-1})$. Worker i thus prefers to start with the easiest task.

Now consider a suitability graph G where there are $m+1$ tasks, and suppose that, for any state with m

tasks or less, every player plays a simple strategy and chooses to work on the easiest task first. Consider a worker i and two tasks j and k such that $t_j, t_k \in T_i$ (worker i can work on both tasks) but $t_l \notin T_i$ for all $j < l < k$ (there is no task ranked between j and k on which i can work). After tasks j and k are completed, there are only m tasks left and, by the induction hypothesis, all players play simple strategies and the states $G - j$ and $G - k$ are characterized by the vector $a_{-j} = (a_1, ..., a_{j-1}, 0, a_{j+1}, ..., a_m)$ and $a_{-k} = (a_1, ..., a_{k-1}, 0, a_{k+1}, ..., a_m)$.[7] Our objective is to show $U(a_{-k}) - U(a_{-j}) \geq 0$. We decompose the difference as:

$$\begin{aligned} U(a_{-k}) - U(a_{-j}) &= U(a_1, ..., a_{k-1}, 0, a_{k+1}, ..., a_m) - U(a_1, ..., a_{j-1}, 0, a_{j+1}, ..., a_m), \\ &= U(a_1, ..., a_{k-1}, 0, a_{k+1}, ..., a_m) - U(a_1, ..., a_{j-1}, a_{j+1}, ..., a_{k-1}, a_k, 0, a_{k+1}, a_m) \\ &- +U(a_1, ..., a_{j-1}, a_{j+1}, ..., a_{k-1}, a_k, 0, a_{k+1}, a_m) - U(a_1, ..., a_{j-1}, 0, a_{j+1}, ..., a_m). \end{aligned}$$

The vector $(a_1, ..., a_{j-1}, a_{j+1}, ..., a_{k-1}, a_k, 0, a_{k+1}, a_m)$ is constructed by (i) using the same ordered sequence as in a_{-j}, but (ii) having task k (rather than task j) completed. When comparing the vector $(a_1, ..., a_{j-1}, a_{j+1}, ..., a_{k-1}, a_k, 0, a_{k+1}, a_m)$ with the vector a_{-k}, the set of tasks remains identical, but the number of workers working on each task has changed: the number of workers working on tasks $1, ...j - 1$ and $k+1, ..., m$ remains identical, but the number of workers working on tasks j to $k-1$ has increased, from $(a_j, ..., a_{k-1})$ to $(a_{j+1}, ..., a_k)$. When comparing the vector $(a_1, ..., a_{j-1}, a_{j+1}, ..., a_{k-1}, a_k, 0, a_{k+1}, a_m)$ with the vector a_{-j}, the ordered vector of the number of agents working on each task remains constant, equal to $(a_1, ..., a_{j-1}, a_{j+1}, ..., a_m)$ but the set of tasks has changed: task k has been completed in the vector $(a_1, ..., a_{j-1}, a_{j+1}, ..., a_{k-1}, a_k, 0, a_{k+1}, a_m)$ and task j in the vector a_{-j}. The number of workers working on tasks $1, ..., j-1$ and $k+1, ..., m$ is the same, but in the vector $(a_1, ..., a_{j-1}, a_{j+1}, ..., a_{k-1}, a_k, 0, a_{k+1}, a_m)$, there are a_{l+1} workers working on tasks $l = j, ..., k - 1$.

We finally note that the state $(a_1, ..., a_{j-1}, a_{j+1}, ..., a_{k-1}, a_k, 0, a_{k+1}, a_m)$ does *not belong* to the state space, as it cannot be obtained as a sub-graph of the initial suitability graph G. We thus need to *expand* the state space to include the state $(a_1, ..., a_{j-1}, a_{j+1}, ..., a_{k-1}, a_k, 0, a_{k+1}, a_m)$ and all vectors which can be obtained from $(a_1, ..., a_{j-1}, a_{j+1}, ..., a_{k-1}, a_k, 0, a_{k+1}, a_m)$ after replacing some a_l with 0 as tasks are completed. We assume that, in the expanded state space, all workers play simple strategies, starting with the easiest task.

Claim 1: We have: $U(a_1, ..., a_{k-1}, 0, a_{k+1}, ..., a_m) - U(a_1, ..., a_{j-1}, a_{j+1}, ..., a_{k-1}, a_k, 0, a_{k+1}, a_m) \geq 0$.

[7]The notations are easily adapted if $j = 1$ or $k = m$.

Proof of the Claim:

To simplify notation, we will let $c = (a_1, ..., a_{k-1}, a_{k+1}, ..., a_m)$ and $b = (a_1, ..., a_{j-1}, a_{j+1}, ..., a_{k-1}, a_k, a_{k+1}, a_m)$ denote the two $m-1$ dimensional vectors obtained after elimination of task k.

The proof is by induction on the number of tasks. If we only have one task l and $l \leq j-1$ or $l \geq k+1$, then $U(b) = U(c)$. If $j \leq l \leq k$ then $c_l = a_l$ and $b_l = a_{l+1}$ so $U(c) = \frac{1}{a_l} \geq \frac{1}{a_{l+1}} = U(b)$. We have

$$U(c) - U(b) = \frac{1}{g(c)} + \sum_j \frac{r_j}{g(c)} U(c-j) - \frac{1}{g(b)} - \sum_j \frac{r'_j}{g(b)} U(b-j),$$

where r'_j is the number of workers who work on task j at state b.

Now,

$$\begin{aligned} U(c) - U(b) &= \frac{1}{g(c)} - \frac{1}{g(b)} + \sum_j \frac{r_j}{g(c)} U(c-j) \\ &- \sum_j \frac{r_j}{g(b)} U(c-j) + \sum_j \frac{r_j}{g(b)} U(c-j) - \sum_j \frac{r'_j}{g(b)} U(b-j), \\ &= U(c)(1 - \frac{g(c)}{g(b)}) + \sum_j \frac{r_j}{g(b)} U(c_{-j}) - \sum_j \frac{r'_j}{g(b)} U(b_{-j}), \\ &= U(c)(1 - \frac{g(c)}{g(b)}) + \sum_j \frac{r_j}{g(b)} U(c-j) \\ &- \sum_j \frac{r'_j}{g(b)} U(c-j) + \sum_j \frac{r'_j}{g(b)} U(c-j) - \sum_j \frac{r'_j}{g(b)} U(b-j). \end{aligned}$$

By the inductive hypothesis, $U(c-j) \geq U(b-j)$. Hence,

$$U(c) - U(b) \geq (1 - \frac{g(c)}{g(b)}) U(c) + \sum_j \frac{r_j}{g(b)} U(c-j) - \sum_j \frac{r'_j}{g(\mathbf{b})} U(c-j).$$

Now recall that

$$r_j = g(a_j, ..., a_m) - g(a_{j+1}, ..., a_m),$$

so that

$$\begin{aligned} \sum_j \frac{(r_j - r'_j)}{g(b)} U(c_{-j}) &= \sum_j \frac{g(c_j, ..., c_m) - g(c_{j+1}, ..., c_m)}{g(b)} U(c-j) \\ &- \sum_j \frac{g(b_j, .., b_m) - g(b_{j+1}, ..b_m)}{g(b)} U(c-j) \end{aligned}$$

Now recall that $b = (a_1, ..., a_{k-1}, a_{k+1}, .., a_l) > c = (a_1, ..., a_k, .., a_{l-1})$, so that by Strong Union Difference

Invariance,

$$g(b_j, .., b_m) - g(b_{j+1}, ..b_m) \geq g(c_j, ..., a_m) - g(c_{j+1}, ..., c_m).$$

and as $U(c-j) < U(c)$ by Lemma 8,

$$g(c_j, ..., c_m) - g(c_{j+1}, ..., c_m) - g(b_j, .., b_m) + g(b_{j+1}, ..b_m)g(b)U(c-j) \geq \quad (7)$$

$$\frac{g(c_j, ..., c_m) - g(c_{j+1}, ..., c_m) - g(b_j, .., b_m) + g(b_{j+1}, ..b_m)}{g(b)}U(c). \quad (8)$$

$$(9)$$

Summing up over all j,

$$\sum_j \frac{g(c_j, ..., c_m) - g(c_{j+1}, ..., c_m) - g(b_j, .., b_m) + g(b_{j+1}, ..b_m)}{g(b)}U(c-j) \geq \frac{g(c) - g(b)}{g(b)}U(c-j).$$

so that

$$U(c) - U(b) \geq (1 - \frac{g(c)}{g(b)})U(c) + (\frac{g(c)-1}{g(b)})U(c) = 0,$$

completing the proof of the Claim. ■

Claim 2: We have: $U(a_1, ..., a_{j-1}, a_{j+1}, ..., a_{k-1}, a_k, 0, a_{k+1}, a_m) - U(a_1, ..., a_{j-1}, 0, a_{j+1}, ..., a_m) \geq 0.$

Proof of the Claim: Fix an m dimensional ordered vector $(a_1, a_2, ..., a_m)$ with $a_l \in \{0, a_l\}$, $a_j > 0$ and $a_{k-1} > 0$. The proof will be an induction on the number of tasks $l \neq k, j$ for which $a_l \neq 0$ (i.e. the number of tasks different from j and k on which agent i can work.) We construct different $m+1$-dimensional vectors based on the vector $\mathbf{a} = (a_1, ..., a_m)$.

First, let $b(\mathbf{a}) = (a_1, a_2, ..., a_{k-1}, 0, a_k, ..., a_m)$, namely $b_l = a_l$ for $l \leq k-1$, $b_k = 0$, $b_l = a_{l-1}$ for $l \geq k+1$.

Second, let $c(\mathbf{a}) = (a_1, a_2, ..., a_{j-1}, 0, a_j, ..., a_m)$, namely $c_l = a_l$ for $l \leq j-1$, $c_j = 0$, $c_l = a_{l-1}$ for $l \geq j+1$.

The vector b is thus obtained by letting $b_k = 0$ while the vector c is obtained by letting $c_j = 0$. The statement of the Claim can be rewritten as : $U(b) - U(c) \geq 0$ whenever $a_l > 0$ for all l.

Third, let $b^1(\mathbf{a}) = (a_1, a_2, ..., a_{k-2}, 0, 0, a_k, ..., a_{m-1})$, namely $b^1_l = a_l$ for $l \leq k-2$, $b^1_{k-1} = b^1_k = 0$, $b^1_l = a_{l-1}$ for $l \geq k+1$.

Fourth, let $c^1(\mathbf{a}) = (a_1, a_2, ..., a_{j-1}, 0, a_j, ..., a_{k-2}, 0, a_k, ..., a_{m-1})$, namely $c^1_l = a_l$ for $l \leq j-1$, $c^1_j = 0$,

$c^1_l = a_{l-1}$ for $j+1 \leq l \leq k-1$, $c^1_k = 0$, $c^1_l = a_{l-1}$ for $l \geq k+1$. [8]

The vector b^1 is obtained from the vector b after elimination of the task $k-1$ while the vector c^1 is obtained from the vector c after elimination of the task k.

Fifth, let $b^2(\mathbf{a}) = b(\mathbf{a}) = (a_1, a_2, ..., a_{j-1}, 0, a_{j+1}, ..., a_{k-1}, 0, a_k, ..., a_m)$, namely $b^2_l = a_l$ for $l \leq j-1$, $b^2_j = 0$, $b^2_l = a_l$ for $j+1 \leq l \leq k-1$, $b^k_2 = 0$ and $b^2_l = a_{l-1}$ for $l \geq k+1$.[9]

Sixth, let $c^2(\mathbf{a}) = (a_1, a_2, ..., a_{j-1}, 0, 0, a_{j+1}, ..., a_m)$, namely $c^2_l = a_l$ for $l \leq j-1$, $c^2_j = c^2_{j+1} = 0$, $c^2_l = a_{l-1}$ for $l \geq j+2$

The vector b^2 is obtained from the vector b after elimination of the task j while the vector c^2 is obtained from the vector c after elimination of the task $j+1$.

Seventh, let $\hat{b}(\mathbf{a}) = (a_1, a_2, ..., a_{j-1}, 0, a_{j+1}, ..., a_{k-2}, 0, 0, a_k, ..., a_{m-1})$, namely $\hat{b}_l = a_l$ for $l \leq j-1$, $\hat{b}_j = 0$,$\hat{b}_l = a_l$ for $j+1 \leq l \leq k-2$, $\hat{b}_{k-1} = \hat{b}_k = 0$ $\hat{b}_l = a_{l-1}$ for $l \geq k+1$.[10]

Eighth, let $\hat{c}(\mathbf{a}) = (a_1, a_2, ..., a_{j-1}, 0, 0, a_{j+1}, ..., a_{k-2}, 0, a_k, ..., a_{m-1})$, namely $\hat{c}_l = a_l$ for $l \leq j-1$, $\hat{c}_j = \hat{c}_{j+1} = 0$,$\hat{c}_l = a_{l-1}$ for $j+2 \leq l \leq k-1$, $\hat{c}_k = 0$ $\hat{c}_l = a_{l-1}$ for $l \geq k+1$.[11]

The vector $\hat{b}$ is obtained from the vector b after elimination of tasks j and $k-1$ while the vector $\hat{c}$ is obtained from the vector c after elimination of tasks k and $j+1$.

We now prove that $U(b) \geq U(c)$ in several steps.

Step 1: For any two vectors $\hat{b}$ and $\hat{c}$ generated from $\mathbf{a}$, $U(\hat{b}(\mathbf{a})) = U(\hat{c}(\mathbf{a}))$.

By induction on the number of tasks in $\mathbf{a}$ (different from a_j, a_{k-1})) for which $a_l > 0$. If there is no task different from a_j, a_{k-1}), we immediately have: $\hat{b} = \hat{c} = (0, 0, ..., 0)$ Now suppose that $\hat{b} = \hat{c}$ for all $\mathbf{a}$ with less than m active tasks. Now notice that $\hat{b}_l = \hat{b}_l$ for $l \leq j-1$ and $l \geq k+1$, that $\hat{b}_j = \hat{b}_k = \hat{c}_j = \hat{c}_l = 0$. Hence, $\hat{b}$ and $\hat{c}$ only differ in the number of workers for tasks that i cannot work on. This implies that either i can work on some task both at $\hat{b}$ and $\hat{c}$ or she cannot work on any task at $\hat{b}$ and $\hat{c}$. In the latter case, $U(\hat{b}(\mathbf{a})) = 0 = U(\hat{c}(\mathbf{a}))$. In the former case, using the recursive formula:

$$\begin{aligned} U(\hat{b}(\mathbf{a})) - U(\hat{c}(\mathbf{a})) &= \sum_{l \leq j-1} r_l (U(\hat{b}(\mathbf{a}) - l) - U(\hat{c}(\mathbf{a}) - l)) + \sum_{j+1 \leq l \leq k-2} r_l (U(\hat{b}(\mathbf{a}) - l) - U(\hat{c}(\mathbf{a})) - (l+1)) \\ &+ \sum_{l \geq k} r_l (U(\hat{b}(\mathbf{a}) - (l+1)) - U(\hat{c}(\mathbf{a}) - (l+1))) \end{aligned}$$

[8] If $j+1 > k-1$, then clearly there is no task l such that $j+1 \leq l \leq k-1$ and the definition of the vector must be adapted accordingly.

[9] Again, if $j+1 > k-1$, then there is no task l such that $j+1 \leq l \leq k-1$ and the definition of the vector must be adapted accordingly.

[10] Whenever $j+1 > k-2$, the definition must be adapted. If $j = k-1$, then $\hat{b}_j = \hat{b}_{k-1} = 0$, if $j = k-2$, then there is no l such that $j+1 \leq l \leq k-2$.

[11] Whenever $j+2 > k-1$, the definition must be adapted. If $j = k-1$, then $\hat{c}_{j+1} = \hat{c}_k = 0$, if $j = k-2$, then there is no l such that $j+2 \leq l \leq k-1$.

But, by construction, for any $l \leq j-1$, $\hat{b}(\mathbf{a}-l) = \hat{b}(a_{-l})$ and $\hat{c}(\mathbf{a}-l) = \hat{c}(a_{-l})$. For any $j+1 \leq l \leq k-2$, $\hat{b}(\mathbf{a}-l) = \hat{b}(a_{-l})$ and $\hat{c}(\mathbf{a}-(l+1)) = \hat{c}(a_{-l})$. For any $l \geq k$, $\hat{b}(\mathbf{a}-(l+1)) = \hat{b}(a_{-l})$ and $\hat{c}(\mathbf{a}-(l+1)) = \hat{c}(a_{-l})$. Hence,

$$U(\hat{b}(\mathbf{a})) - U(\hat{c}(\mathbf{a})) = \sum_{l \neq j, k-1} r_l(U(\hat{b}(a_{-l}) - U(\hat{c}(a_{-l})).$$

By the induction hypothesis, $U(\hat{b}(a_{-l}) - U(\hat{c}(a_{-l}) = 0$, establishing that

$$U(\hat{b}(\mathbf{a})) = U(\hat{c}(\mathbf{a})),$$

completing the proof of Step 1. ■

Step 2: For any vectors $b^1(\mathbf{a}), c^1(\mathbf{a}), b^2(\mathbf{a})$ and $c^2(\mathbf{a})$ generated from $\mathbf{a}$, we have $U(b^1) - U(c^1) > 0$ and

$$U(b^1) - U(c^1) + U(b^2) - U(c^2) \geq 0.$$

The proof is again by induction on the number of components of the vector $\mathbf{a}$, different from a_j and a_{k-1} which are different from 0.

Suppose first that all components of $\mathbf{a}$ different from a_j and a_{k-1} have value 0. Then $b^1_l = 0$ for all $l \neq j$ and $b^1_j = a_j$, $c^1_l = 0$ for all $l \neq j+1$ and $c^1_{j+1} = a_j$ $b^2_l = 0$ for all $l \neq k-1$ and $b^2_{k-1} = a_{k-1}$, $c^2_l = 0$ for all $l \neq k$ and $c_k = a_{k-1}$. Because worker i can work on tasks j and k but not on tasks $j+1$ and $k-1$, we have: $U(b^1) = \frac{1}{a_j}, U(c^1) = 0, U(b^2) = 0$ and $U(c^2) = \frac{1}{a_{k-1}}$. Hence $U(b^1) - U(c^1) = \frac{1}{a_j} > 0$ and

$$U(b^1) - U(c^1) + U(b^2) - U(c^2) = \frac{1}{a_j} - \frac{1}{a_{k-1}} \geq 0.$$

Now, suppose that $U(b^1) - U(c^1) + U(b^2) - U(c^2) \geq 0$ for any $\mathbf{a}$ with less than m tasks different from a_j, a_{k-1} with a nonzero value. Consider a vector $\mathbf{a}$ with m tasks and compute the difference $U(b^1\mathbf{a}) - U(c^1\mathbf{a})$. Notice that because $b^1_j = a_j \neq 0$, worker i will always be active at $b^1(\mathbf{a})$ (and may or may not be active at $c^1(\mathbf{a})$.) Hence

$$\begin{aligned} U(b^1(\mathbf{a})) - U(c^1(\mathbf{a})) \quad &\geq \quad \frac{1}{g(a_{-(k-1)})} + \sum_{l \leq j-1} r_l(U(b^1(\mathbf{a}) - l) - U(c^1(\mathbf{a}) - l) \\ &+ \quad r_j((U(b^1(\mathbf{a}) - j) - U(c^1(\mathbf{a}) - (j+1)) + \sum_{j+1 \leq l \leq k-2} r_l((U(b^1(\mathbf{a}) - l) - U(c^1(\mathbf{a}) - (l+1)) \\ &+ \quad \sum_{l \geq k} r_l(U(b^1(\mathbf{a}) - (l+1)) - U(c^1(\mathbf{a}) - (l+1))). \end{aligned}$$

Now, by construction, for $l \leq j-1$, $b^1(\mathbf{a}) - l = b^1(a_{-l})$ and $c^1(\mathbf{a}) - l = c^1(a_{-l})$. We also have $b^1(\mathbf{a}) - j = \hat{b}(\mathbf{a})$ and $c^1(\mathbf{a} - (j+1)) = \hat{c}(\mathbf{a})$. For $j+1 \leq k-2$, $b^1(\mathbf{a}) - l = b^1(a_{-l})$ and $c^1(\mathbf{a}) - (l+1) = c^1(a_{-l})$. Finally, for $l \geq k$, $b^1(\mathbf{a}) - (l+1) = b^1(a_{-l})$ and $c^1(\mathbf{a}) - (l+1) = c^1(a_{-l})$. Hence,

$$U(b^1(\mathbf{a})) - U(c^1(\mathbf{a})) \geq \frac{1}{g(a_{-(k-1)})} + \sum_{l \neq j, k-1} r_l(U(b^1(a_{-l})) - U(c^1(a_{-l}))) + r_j(U(\hat{b}(\mathbf{a})) - U(\hat{c}(\mathbf{a}))).$$

Because, by Step 1, $U(\hat{b}(\mathbf{a})) - U(\hat{c}(\mathbf{a})) = 0$, we have:

$$U(b^1(\mathbf{a})) - U(c^1(\mathbf{a})) \geq \frac{1}{g(a_{-(k-1)})} + \sum_{l \neq j, k-1} r_l(U(b^1(a_{-l})) - U(c^1(a_{-l}))).$$

By the inductive hypothesis, $U(b^1(a_{-l})) - U(c^1(a_{-l})) > 0$ so that $U(b^1(\mathbf{a})) - U(c^1(\mathbf{a})) > 0$

We also compute the difference $U(b^2(\mathbf{a})) - U(c^2(\mathbf{a}))$. Notice that because $c^2_k = a_{k-1} \neq 0$, worker i will always be active at $c^2(\mathbf{a})$ (and may or may not be active at $b^2(\mathbf{a})$.) Hence

$$\begin{aligned} U(b^2(\mathbf{a})) - U(c^2(\mathbf{a})) &\geq - - \frac{1}{g(a_{-j})} + \sum_{l \leq j-1} r_l(U(b^2(\mathbf{a}) - l) - U(c^2(\mathbf{a}) - l)) \\ &+ \sum_{j+1 \leq l \leq k-2} r_l((U(b^2(\mathbf{a}) - l) - U(c^2(\mathbf{a}) - (l+1))) + r_{k-1}(U(b^2(\mathbf{a}) - l) - U(c^2(\mathbf{a}) - (l+1))) \\ &+ \sum_{l \geq k} r_l(U(b^2(\mathbf{a}) - (l+1)) - U(c^2(\mathbf{a}) - (l+1))). \end{aligned}$$

Now, by construction, for $l \leq j-1$, $b^2(\mathbf{a}) - l = b^2(a_{-l})$ and $c^2(\mathbf{a}) - l) = c^1(a_{-l})$. For $j+1 \leq k-2$, $b^2(\mathbf{a}) - l = b^2(a_{-l})$ and $c^2(\mathbf{a}) - (l+1) = c^2(a_{-l})$. We also have $b^2(\mathbf{a} - (k-1)) = \hat{b}(\mathbf{a})$ and $c^2(\mathbf{a} - k = \hat{c}(\mathbf{a})$ Finally, for $l \geq k$, $b^2(\mathbf{a}) - (l+1) = b^2(a_{-l})$ and $c^2(\mathbf{a}) - (l+1) = c^2(a_{-l})$. Hence,

$$U(b^2(\mathbf{a})) - U(c^2(\mathbf{a})) \geq -\frac{1}{g(a_{-j})} + \sum_{l \neq j, k-1} r_l(U(b^2(a_{-l})) - U(c^2(a_{-l}))) + r_{k-1}(U\hat{b}(\mathbf{a}) - U(\hat{c}(\mathbf{a}))).$$

Because, by Step 1, $U\hat{b}(\mathbf{a}) - U(\hat{c}(\mathbf{a})) = 0$, we have:

$$U(b^2(\mathbf{a})) - U(c^2(\mathbf{a})) \geq -\frac{1}{g(a_{-j})} + \sum_{l \neq j, k-1} r_l(U(b^2(a_{-l})) - U(c^2(a_{-l}))).$$

Hence,

$$\begin{aligned}U(b^1(\mathbf{a})) - U(c^1(\mathbf{a})) + U(b^2(\mathbf{a})) - U(c^2(\mathbf{a})) &\geq \frac{1}{g(a_{-(k+1)})} - \frac{1}{g(a_{-j})} \\ &+ \sum_{l \neq j, k-1} r_l(U(b^1(a_{-l})) - U(c^1(a_{-l})) + U(b^1(a_{-l})) - U(c^1(a_{-l}))).\end{aligned}$$

Because $g(\cdot)$ is increasing, and $a_{-(k+1)} \leq a_{-j}$, $g(a_(- k + 1)) \leq g(a_{-j})$ so that $\frac{1}{g(a_{-(k+1)})} \geq \frac{1}{g(a_{-j})}$. Hence

$$U(b^1(\mathbf{a})) - U(c^1(\mathbf{a})) + U(b^2(\mathbf{a})) - U(c^2(\mathbf{a})) \geq \sum_{l \neq j, k-1} r_l(U(b^1(a_{-l})) - U(c^1(a_{-l})) + U(b^1(a_{-l})) - U(c^1(a_{-l}))).$$

By the induction hypothesis, $U(b^1(a_{-l})) - U(c^1(a_{-l})) + U(b^1(a_{-l})) - U(c^1(a_{-l})) \geq 0$, so that

$$U(b^1(\mathbf{a})) - U(c^1(\mathbf{a})) + U(b^2(\mathbf{a})) - U(c^2(\mathbf{a})) \geq 0,$$

completing the proof of Step 2. ■

Step 3: For any vectors $b(\mathbf{a})$ and $c^(\mathbf{a})$ generated from $\mathbf{a}$, we have

$$U(b(\mathbf{a})) \geq U(c(\mathbf{a})).$$

The proof is again by induction on the number of components of $\mathbf{a}$ different from a_j, a_{k-1} which have a non zero value. If there is no component with a non zero value, $U(b) = \frac{1}{a_j} \geq \frac{1}{a_{k-1}} = U(c)$.

Now suppose that for all vectors $\mathbf{a}$ with less than m nonzero components $U(b(\mathbf{a})) \geq U(c(\mathbf{a}))$ and consider a vector with m nonzero components.

First notice that as $b_j = a_j > 0$ and $c_k = a_{k-1} > 0$, player i is active both at $b(\mathbf{a})$ and $c(\mathbf{a})$. Hence,

$$\begin{aligned}U(b(\mathbf{a})) - U(c(\mathbf{a})) &= \sum_{l \leq j-1} r_l(U(b(\mathbf{a}) - l) - U(c(\mathbf{a}) - l)) \\ &+ r_j(U(b(\mathbf{a}) - j) - U(c(\mathbf{a}) - (j+1))) + \sum_{j+1 \leq l \leq k-2} r_l(U(b(\mathbf{a}) - l) - U(c(\mathbf{a}) - (l+1))) \\ &+ r_{k-1}(U(b(\mathbf{a}) - (k-1)) - U(c^2(\mathbf{a}) - k)) + \sum_{l \geq k} r_l(U(b(\mathbf{a}) - (l+1)) - U(c(\mathbf{a}) - (l+1)).\end{aligned}$$

Now, observe that for $l \leq j - 1$, $b(\mathbf{a}) - l = b(a_{-l})$ and $c(\mathbf{a}) - l = c(a_{-l})$. We also have $b(\mathbf{a} - j) = b^2(\mathbf{a})$ and $c(\mathbf{a} - (j+1)) = c^2(\mathbf{a})$. For $j + 1 \leq j \leq k - 2$, $b(\mathbf{a} - l) = b(a_{-l})$ and $c(\mathbf{a} - (l+1)) = c(a_{-l})$. We have $b(\mathbf{a} - (k-1) = b^1(\mathbf{a})$ and $c(\mathbf{a} - k) = c^1(\mathbf{a})$. Finally, for $l \geq k$, $b(\mathbf{a} - (l+1)) = b(a_{-l})$ and

$c(\mathbf{a}) - (l+1) = c(a_{-l})$.

Hence,

$$\begin{aligned} U(b(\mathbf{a})) - U(c(\mathbf{a})) &= \sum_{l \le j-1} r_l(U(b(a_{-l})) - U(c(a_{-l}))) \\ &+ r_j(U(b^2(\mathbf{a})) - U(c^2(\mathbf{a}))) + \sum_{j+1 \le l \le k-2} r_l(U(b(a_{-l})) - U(c(a_{-l}))) \\ &+ r_{k-1}(U(b^1(\mathbf{a})) - U(c^1(\mathbf{a}))) + \sum_{l \ge k} r_l(U(b(a_{-l})) - U(c(a_{-l}))). \end{aligned}$$

Now, by Submodularity, when $j < k-1$, $g(j, j+1, .., m) - g(j+1, ..., m) \le g(k-1, ..., m) - g(k, ..., m)$, so $r_j \le r_{k-1}$. We also recall, by Step 2, that $U(b^1(\mathbf{a})) - U(c^1(\mathbf{a})) > 0$. This implies:

$$U(b(\mathbf{a})) - U(c(\mathbf{a})) \ge \sum_{l \ne j, k-1} r_l(U(b(a_{-l})) - U(c(a_{-l}))) + r_j(U(b^1(\mathbf{a}) - U(c^1(\mathbf{a}) + U(b^2(\mathbf{a})) - U(c^2(\mathbf{a}))).$$

By Step 2, $U(b^1(\mathbf{a}) - U(c^1(\mathbf{a}) + U(b^2(\mathbf{a})) - U(c^2(\mathbf{a})) \ge 0$. By the inductive hypothesis, $U(b(a_{-l})) - U(c(a_{-l})) \ge 0$, so that

$$U(b(\mathbf{a})) - U(c(\mathbf{a})),$$

establishing Step 3 and completing the proof of the Theorem. ■

Proof of Proposition 1:

Suppose that the condition holds. Then, whenever $m = 2$, the optimal policy of the planner and the MPE of the workers' game both have workers who can work on both targets work on the target with the highest exogenous rank.

Suppose that, whenever there are $m-1$ tasks left, all workers who can work on any subset of tasks work on the task with the highest index. If the condition of the Proposition is satisfied, for any task $j = 1, .., m$, the expected number of workers who can work on task j is the same, so that the expected time of completion and expected value of a worker are the same whenever any task is completed, $V(H-j) = V(H-k)$ and $U(H-j) = U(H-k)$ for all j, k. This implies that both the planner and the workers are indifferent among all tasks when there are m tasks left.

Next, we suppose that the condition of the Proposition is not satisfied, and we show that there exists one state at which the choices of the planner and of one agent are different.

If there exist two tasks j and k such that $a_j \ne a_k$, a direct application of Lemmas 2 and 4 shows that the equilibrium strategy of a worker who can work on both tasks (who must exist because the graph is connected) differs from the optimal policy of the planner.

So we next suppose that $a_j = a_k$ for all j, k but that there are two collections of sets $\{A_1, ..., A_k\}$ and $\{B_1, .., B_k\}$ such that $| \cap_{j=1}^k A_j| \neq | \cap_{j=1}^k B_j|$. Pick the smallest value of k for which these collections of sets exist.

First notice that, if $| \cap_{i=1}^k A_j| \neq | \cap_{i=1}^k B_j|$, then there must exist two collections of k sets which only differ in one set $\{A_1, ...A_{k-1}, A_k\}$ and $\{A_1, ..., A_{k-1}, A_l\}$ such that $|\cap_{j=1}^k A_j| \neq |\cap_{j=1}^{k-1} A_j \cap A_l|$. Otherwise, all collections of k sets would necessarily have an intersection of the same cardinal, as any collection of k sets can be obtained from another collection of k sets by permuting one set at a time. Without loss of generality, suppose that $| \cap_{j=1}^k A_j| < | \cap_{j=1}^{k-1} A_j \cap A_l|$

Now consider the planner's problem when there are $k+1$ remaining tasks $i = 1, .., k-1, k, l$. Suppose first that there exists one worker i who can work on tasks k and l, $A_k \cap A_l \neq \emptyset$. Notice that, after any task is completed, all remaining tasks are symmetric (in the sense that all intersections of A_i have the same cardinal) so that

$$\begin{aligned} V(-k) &= -\frac{1}{| \cup j = 1^k A_j|} + V(-k, j), \\ V(-l) &= -\frac{1}{| \cup j = 1^{k-1} A_j \cup A_l|} + V(-l, j) \end{aligned}$$

where $V(-k, j) = V(-l, j)$ is the value of the planner after any pair of tasks has been completed. Now recall that

$$| \cup_{j=1}^k A_j| = \sum_{j=1}^k (-1)^{j+1} \sum_{1 \leq i_1 < .. < i_j \leq k} |A_{i_1} \cap .. \cap A_{i_j}|.$$

and because all intersections of less than k sets in $\mathcal{A}$ have the same cardinal:

$$| \cup_{j=1}^k A_j| - | \cup_{j=1}^{k-1} A_j \cup A_l| = (-1)^{k+1}(| \cap_{j=1}^k A_j| - | \cap_{j=1}^{k-1} A_j \cap A_l|).$$

We deduce that, when k is odd, $| \cup_{j=1}^k A_j| < | \cup_{j=1}^{k-1} A_j \cup A_l|$ and $V(-k) < V(-l)$, whereas when k is even $| \cup_{j=1}^k A_j| > | \cup_{j=1}^{k-1} A_j \cup A_l|$ and $V_(- k) > V(-l)$.

Next observe that, in the decentralized model, worker i computes her expected utilities after tasks k and l are completed as

$$\begin{aligned} U(-k) &= +\frac{1}{| \cup_{j=1}^k A_j|} + U(-k, j), \\ U(-l) &= +\frac{1}{\lambda | \cup_{j=1}^{k-1} A_j \cup A_l|} + U(-k, j), \end{aligned}$$

where we use the fact that $U(-k,j) = U(-l,j)$ for all k,l,j. Using the same reasoning as for the planner, $U(-k) > U(-l)$ if k is odd and $U(-k) < U(-l)$ if k is even. Hence, the rankings of the planner and worker i between the two tasks are different.

Next suppose that $A_k \cap A_l = \emptyset$. Because the suitability graph is connected, there exists j such that $A_k \cap A_j \neq \emptyset$. This implies that for $k = 2$, there exist two collections of sets such that $|A_k \cap A_l| = 0 \neq |A_k \cap A_j|$.

Now, there must exist two tasks j, j' such that $A_k \cap A_j \neq \emptyset, A_j \cap A_{j'} \neq \emptyset$ and $A_k \cap A_{j'} = \emptyset$. To see this, note that, because the graph is connected, there exists a chain of elements connecting A_k and A_l, $i_1 = k, ..., i_K = l$ such that $A_{i_j} \cap A_{i_{j+1}} \neq \emptyset$. But now, pick the smallest j such that $A_{i_1} \cap A_{i_j} = \emptyset$. Then $A_{i_1} \cap A_{i_{j-1}} \neq \emptyset$, and hence we have found three sets $A_{i_1}, A_{i_{j-1}}$ and A_{i_j} with the desired property.

Consider the set of tasks k, j, j' and a worker who can work both on tasks i and j. We compute the planner's payoff

$$\begin{aligned} V(-k) &= -\frac{1}{|A_j \cup A_{j'}|} + V(-k,l), \\ V(-j) &= -\frac{1}{|A_k \cup A_{j'}|} + V(-j,l) \end{aligned}$$

Because all sets A_k have the same cardinal, $V(-k,l) = V(-j,l)$ for all l and, as $A_j \cap A_{j'} \neq \emptyset$ and $A_k \cap A_{j'} = \emptyset$, $|A_j \cup A_{j'}| < |A_k \cup A_{j'}|$, so that $V(-k) < V(-j)$ and the planner strictly prefers to assign a worker to task j.

Similarly, we compute the expected payoff of an worker i who can work on both tasks as

$$\begin{aligned} U(-k) &= +\frac{1}{|A_j \cup A_{j'}|} + U(-k,l), \\ U(-j) &= +\frac{1}{|A_k \cup A_{j'}|} + U(-j,l) \end{aligned}$$

and find that the worker strictly prefers to work on task k, so that the planner and the worker have different incentives. ■

Proof of Proposition 4: We compute

$$\begin{aligned}\frac{ET_w}{ET_p} &= \frac{n(b_2+b_{12})+(b_1+b_{12})^2}{n(b_1+b_{12})+(b_2+b_{12})^2}\\ &= \frac{(\beta_2+\beta_{12})+(\beta_1+\beta_{12})^2}{(\beta_1+\beta_{12})+(\beta_2+\beta_{12})^2}\\ &= \frac{(1-\beta_1)+(1-\beta_2)^2}{(1-\beta_2)+(1-\beta_1)^2}\end{aligned}$$

Relabelling $\alpha_1 = 1-\beta_1, \alpha_2 = 1-\beta_2$, we solve the problem:

$$\max_{\alpha_1,\alpha_2 \; 0\le\alpha_1\le\alpha_2\le 1, 1-\alpha_1\le\alpha_2} h(\alpha_1,\alpha_2) \equiv \frac{\alpha_1+\alpha_2^2}{\alpha_2+\alpha_1^2}$$

We first show that, for $\alpha_2 \geq \max\{\alpha_1, 1-\alpha_1\}$, $\frac{\partial h}{\partial \alpha_2} > 0$. By direct computation we obtain

$$\text{sign}\,\frac{\partial h}{\partial \alpha_2} = \text{ sign } \alpha_2^2 + 2\alpha_2\alpha_1^2 - \alpha_1$$

Now, given that $\alpha_2 \geq 0$, $\alpha_2^2 + 2\alpha_2\alpha_1^2 - \alpha_1 > 0$ if and only if

$$\alpha_2 > -\alpha_1^2 + \sqrt{\alpha_1^4+\alpha_1}$$

If $\alpha_1 > \frac{1}{2}$, then $\alpha_2 \geq \alpha_1 > 1-\alpha_1$. We easily check that for $\alpha_1 > \frac{1}{2}$,

$$\alpha_1 > -\alpha_1^2 + \sqrt{\alpha_1^4+\alpha_1}$$

establishing the result. If now $\alpha_1 < \frac{1}{2}$ then $\alpha_2 \geq 1-\alpha_1 > \alpha_1$, and we easily check that for $\alpha_1 < \frac{1}{2}$,

$$1-\alpha_1 > -\alpha_1^2 + \sqrt{\alpha_1^4+\alpha_1}$$

establishing the result. As $\frac{\partial h}{\partial \alpha_2} > 0$, the optimal solution satisfies $\alpha_2 = 1$ (or $\beta_2 = 0$), so that there is no worker who can only work on task 2. Now focusing on the workers who can work on task 1, we compute, when $\alpha_2 = 1$,

$$\frac{\partial h}{\partial \alpha_1} = -\alpha_1^2 - 2\alpha_1 + 1 = 0,$$

resulting in the value $\alpha_1^* = \sqrt{2}-1$ so that the maximal value of h is given by $\beta_1 = 2-\sqrt{2}, \beta_{12} = \sqrt{2}-1$ and the value

$$h(\alpha_1^*, \alpha_2^*) = \frac{\sqrt{2}}{4-2\sqrt{2}} = \frac{1}{2(\sqrt{2}-1)}.$$

■

Proof of Proposition 5: Let $(a_1, ..., a_{m-1}, n)$ denote the number of agents who can work on tasks $1, 2, ..m-1, m$ in the ranked task model. As the ranked task model satisfies the conditions of Theorems 2, we know that

$$ET_w = \frac{1}{n} + \frac{1}{a_{m-1}} + ... + \frac{1}{a_1}$$

In particular, as $\frac{1}{a_1} \geq \frac{1}{a_k}$ for all $k > 1$

$$ET_w \leq \frac{1}{n} + \frac{m-1}{a_1}$$

which is the expected time of completion of all tasks for the vector $(a_1, ..., a_1, m)$ Next, by the proof of Theorem 1, the expected time of completion is decreasing in the number of workers who can accomplish any task. Hence, the expected time of completion for the vector $(a_1, ..., a_{m-1}, n)$ is greater than the expected time of completion for the vector $(a_1, n, ..., n)$. Hence

$$PA \leq \max_{a \leq n} \frac{ET_w(a, ..., a, n)}{ET_p(a, n, ..., n)}$$

Now we compute

$$ET_w(a, ..., a, n) = \frac{1}{n} + \frac{m-1}{a},$$

and

$$ET_p(a, n, ..., n) = \frac{m-1}{n} + \frac{1}{a}[1 - \frac{a}{n}]^{m-1} + \frac{1}{n}[1 - (1 - \frac{a}{n})^{m-1}].$$

To obtain the preceding formula, note that the only situation where the number of active workers is smaller than n is when tasks $a_2, ..., a_m$ are completed before task a_1, which happens with probability $[1 - \frac{a}{n}]^{m-1}$. In that case, the expected time of completion is equal to $\frac{m-1}{n} + \frac{1}{a}$. For any other realization, the expected time of completion is $\frac{m}{n}$.

Now, let $\alpha = \frac{a}{n}$ denote the fraction of workers who can accomplish tasks $1, ..m-1$ when n grows large. We then have

$$\frac{ET_w(a, ..., a, n)}{ET_p(a, n, ..., n)} = \frac{m - 1 + \alpha}{\alpha m + (1-\alpha)^m}$$

concluding the proof of the Proposition. ■

Proof of Proposition 2 Suppose that there is no exclusive task. The proof is by induction on the number of remaining tasks. Consider first the workers' game when there are two tasks left. For any j, k with $i < j$, the condition guaranteeing that workers choose task j is:

$$\rho_j + \frac{\rho_k}{a_k} \geq \rho_k + \frac{\rho_j}{a_j},$$

yielding

$$\rho_j \geq \rho_k \frac{a_j(a_k - 1)}{a_k(a_j - 1)}$$

As long as this condition is satisfied is satisfied for all pairs (j, k), the workers will choose the planner's optimal task allocation.

Now suppose that there are $t(G)$ remaining tasks. We first show the following lemma.

Lemma 9 *For any agent i, any task j, any suitability graph $G - j$*

$$0 \leq U_i(G - j) \leq \sum_{k \in H_i, k \neq j} \alpha_i^k(G)\rho_k$$

where $0 < \alpha_i^k < 1$.

Proof of the Lemma: Consider a task k. Because the suitability graph $G - j$ has $t(G) - 1$ tasks left, by the inductive hypothesis, all workers work on the task with the smallest number of workers in $G - j$. Pick a task $k \in H_i$. We want to compute the probability that worker i *does not get the reward associated to task* $k \in H_i$,

$$\pi_i^k = \Pr(i does not complete k first))$$

Let σ be a sequence of completed tasks, i.e. a permutation over the set of tasks $T(G - j)$. We have

$$\pi_i^k = \sum_\sigma \Pr(i does not complete k first, the sequence of completed tasks is given by \sigma)$$

Now consider the set of sequences Σ_0 such that task k is completed exactly after all tasks of rank lower than k are completed. We have

$$\pi_i^k \geq \sum_{\sigma \in \Sigma_0} \Pr(i does not complete k first, the sequence of completed tasks is given by \sigma)$$

Now, when task k is completed exactly after all tasks of rank lower than k have been completed, by the induction hypothesis there are exactly a_k workers working on task k, so that the probability that an agent other than k completes the task first is equal to $\frac{a_k-1}{a_k} \in (0,1)$. We thus have

$$\pi_i^k \geq \sum_{\sigma \in \Sigma_0} \frac{a_k - 1}{a_k} \sum_{\sigma \in \Sigma_0}$$

$$= \quad \Pr[\sigma \in \Sigma_0] \frac{a_k - 1}{a_k}$$

where the last equality stems from the fact that the two events that i completes k first and the sequence is given by σ are independent. Now let $\alpha_i^k = 1 - \Pr[\sigma \in \Sigma_0]\frac{a_k-1}{a_k}$. Clearly, $0 < \alpha_i^k < 1$ and the probability that i receives reward k is bounded above by α_i^k, establishing the Lemma. ■

Next to prove the existence of differential rewards, we will fix the ratio $\frac{\rho_j}{\rho_k} = R_{jk}$ for all pairs of tasks (j,k). Notice that the ratios have been constructed for $m = 2$. Now so suppose that the ratios have already been constructed for set of tasks of cardinality smaller than t and let $R_{jk} = \max_{G \ t(g) \leq t} R_{jk}^G$.

Consider a suitability graph G with t tasks. Hence, for any set of tasks smaller than $T(G)$, all workers choose the optimal task allocation of the planner. Consider a worker i for whom j is the smallest index task in H_i, i.e. $k > j \forall j \in H_i, j \neq k$. Worker j prefers to work on j if and only if we have:

$$\rho_j + U_i(G - j) \geq \rho_k + U_i(G - k)$$

for all $k \neq j$. By Lemma 9 $U_i(G-j) > 0$ and $U_i(G-k) \leq a_i^j \rho_j + \sum_{l>j, l \neq k} a_i^l \rho_l$ so that a sufficient condition for the worker to prefer to work on j is

$$\rho_j(1 - a_i^j) \geq \rho_k + \sum_{l>j, l \neq k} a_i^l \rho_l$$

Now consider $a^j = \max_i a^j$, we have a sufficient condition:

$$\rho_j(1 - a^j) \geq \rho_k + \sum_{l>j, l \neq k} a^l \rho_l$$

This gives a recursive formula. For the two tasks with the highest index, we obtain

$$R_{jk}^G = \frac{1}{1 - a^j} > 1$$

So $\rho_j > \rho_k$ for all $j < k$, and we can construct recursively a ratio

$$R^G_{jk} = \frac{1 + \sum_{l>j, l\neq k} a^l R^G_{lk}}{1 - a^j} > 1$$

Whenever $\frac{\rho_j}{\rho_k} > R^G_{jk}$, worker i prefers to work on task j than on task k, establishing the result.

To show that non-exclusivity is a necessary condition, suppose that there is one task j that a single agent can complete. Then, whenever there are two tasks j and k left, *there does not exist a finite ration R_{jk} such that worker i prefers to work on j when $\frac{\rho_j}{\rho_k} > R_{jk}$.* ■

Proof of Proposition 3: When $m = 3$, we need to distinguish between two types of agents: (i) agents who can work on tasks t_2 and t_3 and (ii) agents who can work on all tasks. For agents who can work on tasks t_2 and t_3, the condition reads:

$$\rho_2 + U(13) \geq \rho_3 + U(12)$$

where $U'(13)$ and $U'(12)$ denote the continuation values after tasks 2 and 3 are completed. Now

$$\begin{aligned} U'(13) &= \frac{\rho_3}{a_3} + \frac{(a_3 - a_1)}{a_3}\frac{\rho_3}{a_3} \\ U'(12) &= \frac{\rho_2}{a_2} + \frac{(a_2 - a_1)}{a_2}\frac{\rho_2}{a_2} \end{aligned}$$

giving the condition:

$$\rho_2 \geq \rho_3 \frac{a_2^2(a_3^2 + 2a_3 - a_1)}{a_3^2(a_2^2 + 2a_2 - a_1)}$$

For agents who can work on both tasks, we note that they prefer to work on task t_1 than task t_2 if

$$\rho_1 + U(23) \geq \rho_2 + U(13)$$

and on task t_2 rather than t_3 if

$$\rho_2 + U(13) \geq \rho_3 + U(12)$$

where $U(23), U(13)$ and $U(12)$ denote the continuation values after tasks t_1 t_2 and t_3 are completed. We compute

$$\begin{aligned} U(23) &= \frac{\rho_2}{a_2} + \frac{a_2\rho_3}{a_3^2}, \\ U(13) &= \frac{\rho_1}{a_1} + \frac{a_1\rho_3}{\rho_3^2}, \\ U(12) &= \frac{\rho_1}{a_1} + \frac{a_1\rho_2}{a_2^2} \end{aligned}$$

We finally observe that the condition: $\rho_2 \geq \rho_3 \frac{a_2^2(a_3^2+2a_3-a_1)}{a_3^2(a_2^2+2a_2-a_1)}$ is sufficient to show that the worker prefers to work on task t_2 than task t_3 and that the condition: $\rho_1 \geq \rho_2 \frac{a_1(a_2-1)}{a_2(a_1-1)}$ is sufficient to show that the worker prefers to work on task t_1 than task t_2. ■

8 Appendix B: The three-task model

Unfortunately, Lemma 2 cannot be generalized to more than two tasks. We use the case of three tasks to point out the new considerations that emerge when there are more than two tasks to complete.

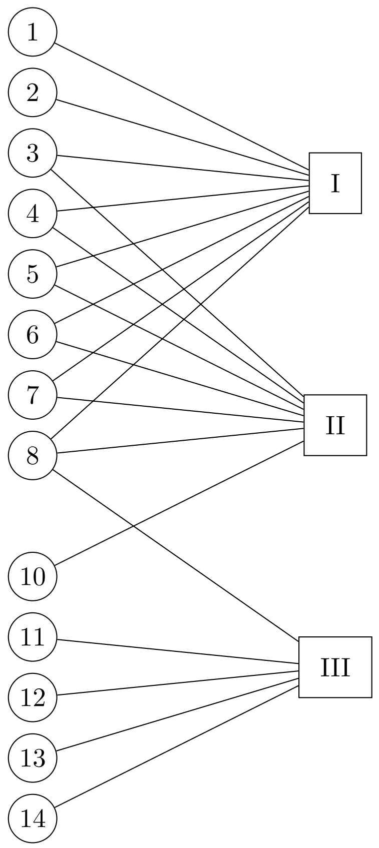
First, Lemma 10 shows that in the case of three tasks, if all workers that can complete the "hardest" task can either complete all tasks or only the "hardest" task, then the planner should follow the principle demonstrated in Lemma 2 by postponing the "easier" tasks as much as possible.

Lemma 10 *Suppose that there are three tasks, $T = \{1,2,3\}$. Suppose that $A_1 \cap A_3 \subseteq A_2$ and $A_2 \cap A_3 \subseteq A_1$. If $a_1 > a_2 > a_3$, optimally, the planner should prefer, for each worker and in each phase, assignment to task 3 over assignments to tasks 1 and 2 and assignment to task 2 over assignment to task 1.*

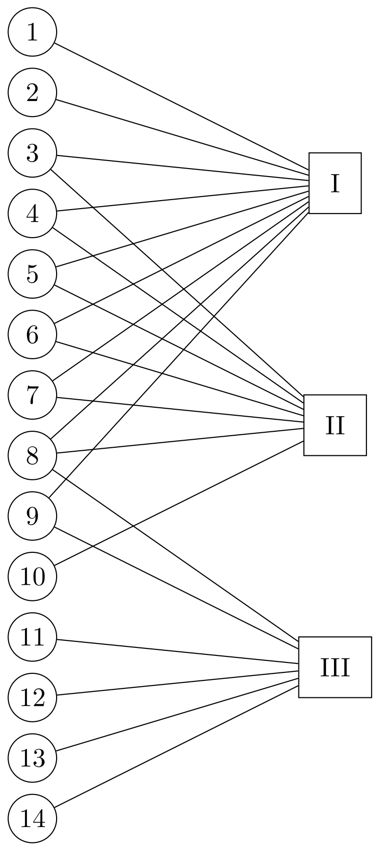
The graph in Figure 4a satisfies Lemma 10 since the agents that can work on the hardest task (task III) can either work only on this task or can work on all tasks. Therefore the planner prefers, for each agent and in each phase, assignment to task III over assignments to tasks I and II and assignment to task II over assignment to task I.[12]

Agent 9, added in Figure 4b, can work on both task I (the "easiest" task) and task III (the "hardest" task), violating the conditions of Lemma 10, without changing the order of difficulty. It turns out that in this case, increasing the probability of having agent 9 available in the final phase, by completing task II first, is more worthwhile than completing the "hardest" task (task III) first and jeopardizing

[12]If task I is completed first, by Lemma 2, in phase 2, agents 3-7 and 10 inspect task II while agents 8 and 11-14 inspect task III. The expected time of completion is $\frac{102}{385} \times \frac{1}{\lambda}$. If task II is completed first, by Lemma 2, in phase 2, agents 1-7 inspect task I while agents 8 and 11-14 inspect task III. The expected time of completion is $\frac{121}{480} \times \frac{1}{\lambda}$. If task III is completed first, by Lemma 2, in phase 2, agents 1 and 2 inspect task I while agents 3-8 and 10 inspect task II. The expected time of completion is $\frac{121}{504} \times \frac{1}{\lambda}$. Hence, it is optimal for the planner to prefer in phase 1 assignment to task III (the "hardest" task) over assignments to tasks I and II and assignment to task II over assignment to task I.



(a) $A_1 \cap A_3 \subseteq A_2$, $A_2 \cap A_3 \subseteq A_1$ and $a_1 > a_2 > a_3$.



(b) $A_2 \cap A_3 \subseteq A_1$ and $a_1 > a_2 > a_3$, but not $A_1 \cap A_3 \subseteq A_2$.

Figure 4: Two examples using 13\14 agents and three tasks. In both, agents 1 and 2 work only on task I, agents 3-7 work on tasks I and II, agent 8 works on all tasks, agent 10 works only on task II and agents 11-14 work only on task III. Agent 9, added in the right-hand-side graph works on tasks I and III.

the availability of agent 9 in the final phase.[13] Interestingly, in this case, the planner's preferences may change across phases. In phase 1, the planner prefers assignments to task II over assignments to task III (and task I). Therefore, agent 8 is optimally assigned to task II. If task I is completed first, by Lemma 2, the planner prefers assignments to task III over assignments to task II. Therefore, the planner will re-assign agent 8 from task II to task III although task II was not completed.

Proof of Lemma 10: Denote by $X_S = \{w_i | H_i = S\}$ the set of workers that are able to complete each task in S and are not able to complete tasks that are not in S. Denote by x_S the cardinality of X_S.

Note that $A_1 \cap A_3 \subseteq A_2$ implies that $X_{13} = \emptyset$ and that $A_2 \cap A_3 \subseteq A_1$ implies that $X_{23} = \emptyset$. Also

[13]If task I is completed first, by Lemma 2, in phase 2, agents 3-7 and 10 inspect task II while agents 8, 9 and 11-14 inspect task III. The expected time of completion is $\frac{5}{21} \times \frac{1}{\lambda}$. If task II is completed first, by Lemma 2, in phase 2, agents 1-7 inspect task I while agents 8, 9 and 11-14 inspect task III. The expected time of completion is $\frac{17}{78} \times \frac{1}{\lambda}$. If task III is completed first, by Lemma 2, in phase 2, agents 1, 2 and 9 inspect task I while agents 3-8 and 10 inspect task II. The expected time of completion is $\frac{139}{630} \times \frac{1}{\lambda}$. Hence, it is optimal for the planner to prefer in phase 1 assignment to task II (not the "hardest" task) over assignments to tasks I and III and assignment to task III over assignment to task I.

note that $a_1 > a_2 > a_3$ implies $x_1 > x_2$ and $x_2 + x_{12} > x_3$ since

$$a_1 = x_1 + x_{12} + x_{13} + x_{123} = x_1 + x_{12} + x_{123}$$

$$a_2 = x_2 + x_{12} + x_{23} + x_{123} = x_2 + x_{12} + x_{123}$$

$$a_3 = x_3 + x_{13} + x_{23} + x_{123} = x_3 + x_{123}$$

We first consider the planner's problem. Suppose that task 1 is completed first. Then, there are $x_{23} + x_{123} = x_{123}$ agents that can work on both tasks, $x_2 + x_{12}$ agents that can work only on task 2 and $x_3 + x_{13} = x_3$ agents that can work only on task 3. The expected completion time given that task 1 was completed first, is a sum of two parts. The first part represents the expected time it takes to complete one of the two tasks (task 2 or task 3): $\frac{1}{\lambda(x_2+x_3+x_{12}+x_{123})}$. The second represents the expected time it takes to complete the final task, given that the workers that can work on both tasks 2 and 3 are allocated in phase 2 to task 3 which is the harder task since $x_2 + x_{12} > x_3$ (Lemma 2). With probability $\frac{x_3+x_{123}}{x_2+x_3+x_{12}+x_{123}}$, task 3 is completed in the second phase, and the expected time to complete the final task, task 2, is $\frac{1}{\lambda(x_2+x_{12}+x_{123})}$. With probability $\frac{x_2+x_{12}}{x_2+x_3+x_{12}+x_{123}}$, task 2 is completed in the second phase and the expected time to complete the final task, task 3, is $\frac{1}{\lambda(x_3+x_{123})}$. Thus, the expected time to complete task 2 and task 3 is

$$\frac{1}{\lambda(x_2 + x_3 + x_{12} + x_{123})}\left[1 + \frac{x_3 + x_{123}}{x_2 + x_{12} + x_{123}} + \frac{x_2 + x_{12}}{x_3 + x_{123}}\right] =$$

$$\frac{1}{\lambda}\left[\frac{1}{(x_3 + x_{123})} + \frac{x_3 + x_{123}}{(x_2 + x_3 + x_{12} + x_{123})(x_2 + x_{12} + x_{123})}\right]$$

Now, suppose that task 2 is completed first. Then, there are $x_{13} + x_{123} = x_{123}$ agents that can work on both tasks, $x_1 + x_{12}$ agents that can work only on task 1 and $x_3 + x_{23} = x_3$ agents that can work only on task 3. The expected completion time given that task 2 was completed first, is a sum of two parts. The first part represents the expected time it takes to complete one of the two tasks (task 1 or task 3): $\frac{1}{\lambda(x_1+x_3+x_{12}+x_{123})}$. The second represents the expected time it takes to complete the final task, given that the workers that can work on both tasks 1 and 3 are allocated in phase 2 to task 3 which is the "harder" task since $x_1 + x_{12} > x_3$ (Lemma 2). With probability $\frac{x_3+x_{123}}{x_1+x_3+x_{12}+x_{123}}$, task 3 is completed in the second phase, and the expected time to complete the final task, task 1, is $\frac{1}{\lambda(x_1+x_{12}+x_{123})}$. With probability $\frac{x_1+x_{12}}{x_1+x_3+x_{12}+x_{123}}$, task 2 is completed in the second phase and the expected time to complete the final task, task 3, is $\frac{1}{\lambda(x_3+x_{123})}$. Thus, the expected time to complete task 1 and task 3 is

$$\frac{1}{\lambda(x_1+x_3+x_{12}+x_{123})}\left[1+\frac{x_3+x_{123}}{x_1+x_{12}+x_{123}}+\frac{x_1+x_{12}}{x_3+x_{123}}\right]=$$

$$\frac{1}{\lambda}\left[\frac{1}{(x_3+x_{123})}+\frac{x_3+x_{123}}{(x_1+x_3+x_{12}+x_{123})(x_1+x_{12}+x_{123})}\right]$$

Finally, suppose that task 3 is completed first. Then, there are $x_{12}+x_{123}$ agents that can work on both tasks, $x_1+x_{13}=x_1$ agents that can work only on task 1 and $x_2+x_{23}=x_2$ agents that can work only on task 2. The expected completion time given that task 3 was completed first, is a sum of two parts. The first part represents the expected time that it takes to complete one of the two tasks (task 1 or task 2): $\frac{1}{\lambda(x_1+x_2+x_{12}+x_{123})}$. The second represents the expected time it takes to complete the final task, given that the workers that can work on both tasks 1 and 2 are allocated in phase 2 to task 2 which is the "harder" task since $x_1 > x_2$ (Lemma 2). With probability $\frac{x_2+x_{12}+x_{123}}{x_1+x_2+x_{12}+x_{123}}$, task 2 is completed in the second phase, and the expected time to complete the final task, task 1, is $\frac{1}{\lambda(x_1+x_{12}+x_{123})}$. With probability $\frac{x_1}{x_1+x_2+x_{12}+x_{123}}$, task 1 is completed in the second phase and the expected time to complete the final task, task 2, is $\frac{1}{\lambda(x_2+x_{12}+x_{123})}$. Thus, the expected time to complete task 1 and task 2 is

$$\frac{1}{\lambda(x_1+x_2+x_{12}+x_{123})}\left[1+\frac{x_2+x_{12}+x_{123}}{x_1+x_{12}+x_{123}}+\frac{x_1}{x_2+x_{12}+x_{123}}\right]=$$

$$\frac{1}{\lambda}\left[\frac{1}{(x_2+x_{12}+x_{123})}+\frac{x_2+x_{12}+x_{123}}{(x_1+x_3+x_{12}+x_{123})(x_1+x_{12}+x_{123})}\right]$$

Comparing the three expected times expressions for the final two phases, it is clear that

$$\frac{1}{\lambda}\left[\frac{1}{(x_3+x_{123})}+\frac{x_3+x_{123}}{(x_2+x_3+x_{12}+x_{123})(x_2+x_{12}+x_{123})}\right]>$$

$$\frac{1}{\lambda}\left[\frac{1}{(x_3+x_{123})}+\frac{x_3+x_{123}}{(x_1+x_3+x_{12}+x_{123})(x_1+x_{12}+x_{123})}\right]$$

That is, the expected time it takes to complete the final two phases is longer if task 1 is completed first than if task 2 is completed first.

Also,

$$\frac{1}{\lambda}\left[\frac{1}{(x_3+x_{123})}+\frac{x_3+x_{123}}{(x_1+x_3+x_{12}+x_{123})(x_1+x_{12}+x_{123})}\right]>$$

$$\frac{1}{\lambda}\left[\frac{1}{(x_2+x_{12}+x_{123})}+\frac{x_2+x_{12}+x_{123}}{(x_1+x_3+x_{12}+x_{123})(x_1+x_{12}+x_{123})}\right]$$

since

$$\frac{x_2 + x_{12} - x_3}{(x_3 + x_{123})(x_2 + x_{12} + x_{123})} > \frac{x_2 + x_{12} - x_3}{(x_1 + x_3 + x_{12} + x_{123})(x_1 + x_{12} + x_{123})}$$

That is, the expected time it takes to complete the final two phases is longer if task 2 is completed first than if task 3 is completed first.

Therefore, since the planner wishes to complete the three tasks as fast as possible, he should prefer, for each worker in phase 1, assignment to task 3 over assignments to tasks 1 and 2 and assignment to task 2 over assignment to task 1. By Lemma 2 this is true also for phase 2 and it is trivially true for phase 3 since there is only one task left.

We now move to the workers' game, focusing on the non-trivial optimal strategies of the agents that can perform all tasks and on the agents that can perform task 1 and task 2. We solve the game backwards based on Lemma 2.

Suppose that task 1 is completed first. Then, there are $x_{23} + x_{123} = x_{123}$ agents that can work on both tasks, $x_2 + x_{12}$ agents that can work only on task 2 and $x_3 + x_{13} = x_3$ agents that can work only on task 3. That is, the probability to be the winner of phase 2 is $\frac{1}{x_2+x_3+x_{12}+x_{123}}$. By Lemma 2, the probability that task 2 (the easier task, since $x_2 + x_{12} > x_3$) is completed in phase 2 is $\frac{x_2+x_{12}+x_{123}}{x_2+x_3+x_{12}+x_{123}}$ while the probability that task 3 is the completed task in phase 2 is $\frac{x_3}{x_2+x_3+x_{12}+x_{123}}$. If task 2 was the one completed in phase 2 the probability of winning phase 3 is $\frac{1}{x_3+x_{123}}$. If task 3 was the one completed in phase 2 the probability of winning phase 3 is $\frac{1}{x_2+x_{12}+x_{123}}$. Therefore, the expected number of completions of agents that can perform all tasks is

$$\frac{1}{x_2 + x_3 + x_{12} + x_{123}} + \frac{x_2 + x_{12} + x_{123}}{x_2 + x_3 + x_{12} + x_{123}} \times \frac{1}{x_3 + x_{123}} + \frac{x_3}{x_2 + x_3 + x_{12} + x_{123}} \times \frac{1}{x_2 + x_{12} + x_{123}} =$$

$$\frac{1}{x_2 + x_3 + x_{12} + x_{123}} \left[1 + \frac{x_2 + x_{12} + x_{123}}{x_3 + x_{123}} + \frac{x_3}{x_2 + x_{12} + x_{123}} \right] =$$

$$\frac{1}{x_2 + x_{12} + x_{123}} + \frac{x_2 + x_{12} + x_{123}}{(x_3 + x_{123})(x_2 + x_3 + x_{12} + x_{123})}$$

The expected number of completions of agents that can perform task 1 and task 2.

$$\frac{1}{x_2 + x_3 + x_{12} + x_{123}} + \frac{x_2 + x_{12} + x_{123}}{x_2 + x_3 + x_{12} + x_{123}} \times 0 + \frac{x_3}{x_2 + x_3 + x_{12} + x_{123}} \times \frac{1}{x_2 + x_{12} + x_{123}} =$$

$$\frac{1}{x_2 + x_3 + x_{12} + x_{123}} \left[1 + \frac{x_3}{x_2 + x_{12} + x_{123}} \right] = \frac{1}{x_2 + x_{12} + x_{123}}$$

Suppose that task 2 is completed first. Then, there are $x_{13} + x_{123} = x_{123}$ agents that can work on both tasks, $x_1 + x_{12}$ agents that can work only on task 1 and $x_3 + x_{23} = x_3$ agents that can work only on task 3. That is, the probability to be the winner of phase 2 is $\frac{1}{x_1+x_3+x_{12}+x_{123}}$. By Lemma 2, the probability that task 1 (the easier task, since $x_1 + x_{12} > x_3$) is completed in phase 2 is $\frac{x_1+x_{12}+x_{123}}{x_1+x_3+x_{12}+x_{123}}$ while the probability that task 3 is the completed task in phase 2 is $\frac{x_3}{x_1+x_3+x_{12}+x_{123}}$. If task 1 was the one completed in phase 2 the probability of winning phase 3 is $\frac{1}{x_3+x_{123}}$. If task 3 is the one completed in phase 2 the probability of winning phase 3 is $\frac{1}{x_1+x_{12}+x_{123}}$. Therefore, the expected number of completions of agents that can perform all tasks is

$$\frac{1}{x_1+x_3+x_{12}+x_{123}} + \frac{x_1+x_{12}+x_{123}}{x_1+x_3+x_{12}+x_{123}} \times \frac{1}{x_3+x_{123}} + \frac{x_3}{x_1+x_3+x_{12}+x_{123}} \times \frac{1}{x_1+x_{12}+x_{123}} =$$

$$\frac{1}{x_1+x_3+x_{12}+x_{123}}\left[1+\frac{x_1+x_{12}+x_{123}}{x_3+x_{123}}+\frac{x_3}{x_1+x_{12}+x_{123}}\right] =$$

$$\frac{1}{x_1+x_{12}+x_{123}} + \frac{x_1+x_{12}+x_{123}}{(x_3+x_{123})(x_1+x_3+x_{12}+x_{123})}$$

The expected number of completions of agents that can perform task 1 and task 2.

$$\frac{1}{x_1+x_3+x_{12}+x_{123}} + \frac{x_1+x_{12}+x_{123}}{x_1+x_3+x_{12}+x_{123}} \times 0 + \frac{x_3}{x_1+x_3+x_{12}+x_{123}} \times \frac{1}{x_1+x_{12}+x_{123}} =$$

Suppose that task 3 is completed first. Then, there are $x_{12} + x_{123}$ agents that can work on both tasks, x_1 agents that can work only on task 1 and x_2 agents that can work only on task 2. That is, the probability to be the winner of phase 2 is $\frac{1}{x_1+x_2+x_{12}+x_{123}}$. By Lemma 2, the probability that task 1 (the easier task, since $x_1 > x_2$) is completed in phase 2 is $\frac{x_1+x_{12}+x_{123}}{x_1+x_2+x_{12}+x_{123}}$ while the probability that task 2 is the completed task in phase 2 is $\frac{x_2}{x_1+x_2+x_{12}+x_{123}}$. If task 1 was the one completed in phase 2 the probability of winning phase 3 is $\frac{1}{x_2+x_{12}+x_{123}}$. If task 2 is the one completed in phase 2 the probability of winning phase 3 is $\frac{1}{x_1+x_{12}+x_{123}}$. Therefore, the expected number of completions of agents that can perform all tasks is

$$\frac{1}{x_1+x_2+x_{12}+x_{123}} + \frac{x_1+x_{12}+x_{123}}{x_1+x_2+x_{12}+x_{123}} \times \frac{1}{x_2+x_{12}+x_{123}} + \frac{x_2}{x_1+x_2+x_{12}+x_{123}} \times \frac{1}{x_1+x_{12}+x_{123}} =$$

$$\frac{1}{x_1+x_2+x_{12}+x_{123}}\left[1+\frac{x_1+x_{12}+x_{123}}{x_2+x_{12}+x_{123}}+\frac{x_2}{x_1+x_{12}+x_{123}}\right] =$$

$$\frac{1}{x_1+x_{12}+x_{123}} + \frac{x_1+x_{12}+x_{123}}{(x_2+x_{12}+x_{123})(x_1+x_2+x_{12}+x_{123})}$$

Since $x_1 > x_2$ we get that $\frac{1}{x_2+x_{12}+x_{123}} > \frac{1}{x_1+x_{12}+x_{123}}$. This means that agents that can perform task 1 and task 2 (but not task 3) have a higher expected number of completions if task 1 is completed in phase 1 compared to the case where task 2 is completed in phase 1. Since they cannot affect their winning probability in phase 1 and the probability that task 3 is completed in phase 1, it is optimal for these agents to engage with the easier task (task 1) in phase 1.

To prove that for agents that can perform all tasks there is a higher expected number of completions if task 1 is completed in phase 1 compared to the case where task 2 is completed in phase 1 we show that the difference is positive:

$$\left[\frac{1}{x_2+x_{12}+x_{123}}+\frac{x_2+x_{12}+x_{123}}{(x_3+x_{123})(x_2+x_3+x_{12}+x_{123})}\right]-$$

$$\left[\frac{1}{x_2+x_{12}+x_{123}}-\frac{1}{x_1+x_{12}+x_{123}}\right]+\frac{1}{x_3+x_{123}}\left[\frac{x_2+x_{12}+x_{123}}{x_2+x_3+x_{12}+x_{123}}-\frac{x_1+x_{12}+x_{123}}{x_1+x_3+x_{12}+x_{123}}\right]=$$

$$\left[\frac{x_1-x_2}{(x_2+x_{12}+x_{123})(x_1+x_{12}+x_{123})}\right]+\frac{1}{x_3+x_{123}}\left[(1-\frac{x_3}{x_2+x_3+x_{12}+x_{123}})-(1-\frac{x_3}{x_1+x_3+x_{12}+x_{123}})\right]=$$

$$\left[\frac{x_1-x_2}{(x_2+x_{12}+x_{123})(x_1+x_{12}+x_{123})}\right]+\frac{x_3}{x_3+x_{123}}\left[\frac{1}{x_1+x_3+x_{12}+x_{123}}-\frac{1}{x_2+x_3+x_{12}+x_{123}}\right]=$$

$$\left[\frac{x_1-x_2}{(x_2+x_{12}+x_{123})(x_1+x_{12}+x_{123})}\right]-\frac{x_3}{x_3+x_{123}}\left[\frac{x_1-x_2}{(x_1+x_3+x_{12}+x_{123})(x_2+x_3+x_{12}+x_{123})}\right]=$$

$$(x_1-x_2)\left[\frac{1}{(x_2+x_{12}+x_{123})(x_1+x_{12}+x_{123})}-\frac{1}{(1+\frac{x_{123}}{x_3})(x_1+x_3+x_{12}+x_{123})(x_2+x_3+x_{12}+x_{123})}\right]$$

Since $x_1 > x_2$ and since the x_ss are non-negative this difference is positive.

Finally, to prove that for agents that can perform all tasks there is a higher expected number of completions if task 1 is completed in phase 1 compared to the case where task 3 is completed in phase 1 we show that the difference is positive:

$$\left[\frac{1}{x_2+x_{12}+x_{123}}+\frac{x_2+x_{12}+x_{123}}{(x_3+x_{123})(x_2+x_3+x_{12}+x_{123})}\right]-\left[\frac{1}{x_1+x_{12}+x_{123}}+\frac{x_1+x_{12}+x_{123}}{(x_2+x_{12}+x_{123})(x_1+x_2+x_{12}+x_{123})}\right]=$$

$$\left[\frac{1}{x_2+x_{12}+x_{123}}-\frac{1}{x_1+x_{12}+x_{123}}\right]+\frac{1}{x_3+x_{123}}\left[1-\frac{x_3}{x_2+x_3+x_{12}+x_{123}}\right]-\frac{1}{x_2+x_{12}+x_{123}}\left[1-\frac{x_2}{x_1+x_2+x_{12}+x_{123}}\right]=$$

$$\left[\frac{1}{x_2+x_{12}+x_{123}}-\frac{1}{x_1+x_{12}+x_{123}}\right]+\left[\frac{1}{x_3+x_{123}}-\frac{1}{x_2+x_{12}+x_{123}}\right]+$$

$$+\frac{x_2}{(x_2+x_{12}+x_{123})(x_1+x_2+x_{12}+x_{123})}-\frac{x_3}{(x_3+x_{123})(x_2+x_3+x_{12}+x_{123})}=$$

$$\frac{1}{x_1 + x_{12} + x_{123}} + \frac{x_1 + x_{12} + x_{123}}{(x_2 + x_{12} + x_{123})(x_1 + x_2 + x_{12} + x_{123})}$$

Comparing the three expected times expressions for the final two phases, it is clear that

$$\frac{1}{\lambda}\left[\frac{1}{(x_3 + x_{123})} + \frac{x_3 + x_{123}}{(x_2 + x_3 + x_{12} + x_{123})(x_2 + x_{12} + x_{123})}\right] >$$

$$\frac{1}{\lambda}\left[\frac{1}{(x_3 + x_{123})} + \frac{x_3 + x_{123}}{(x_1 + x_3 + x_{12} + x_{123})(x_1 + x_{12} + x_{123})}\right]$$

That is, the expected time it takes to complete the final two phases is longer if task 1 is completed first than if task 2 is completed first.

Also,

$$\frac{1}{\lambda}\left[\frac{1}{(x_3 + x_{123})} + \frac{x_3 + x_{123}}{(x_1 + x_3 + x_{12} + x_{123})(x_1 + x_{12} + x_{123})}\right] >$$

$$\frac{1}{\lambda}\left[\frac{1}{(x_2 + x_{12} + x_{123})} + \frac{x_2 + x_{12} + x_{123}}{(x_1 + x_3 + x_{12} + x_{123})(x_1 + x_{12} + x_{123})}\right]$$

since

$$\frac{x_2 + x_{12} - x_3}{(x_3 + x_{123})(x_2 + x_{12} + x_{123})} > \frac{x_2 + x_{12} - x_3}{(x_1 + x_3 + x_{12} + x_{123})(x_1 + x_{12} + x_{123})}$$

That is, the expected time it takes to complete the final two phases is longer if task 2 is completed first than if task 3 is completed first.

Therefore, since the planner wishes to complete the three tasks as fast as possible, he should prefer assignment to task 3 over assignments to tasks 1 and 2 and assignment to task 2 over assignment to task 1. By Lemma 2 this is true also for phase 2 and it is true for phase 3 since there is only one task left. ■