Optimal Task Assignments*

Steve Yeh†

July 30, 2024

Abstract

A principal wields task assignment power over her team of agents working on a project with multiple tasks. Effort spent working on a task generates positive spillovers and substitutes for effort at similar tasks but incurs a convex cost. Assigning agents to more tasks mitigates free-riding and shirking, but raises team effort costs due to convexity. I focus on the class of key-peripheral projects composed of peripheral tasks (linked to one other task) and key tasks (linked to at least two peripheral tasks). The modular assignment, which assigns each agent to a module consisting of a key task and its peripheral tasks, resolves the free-riding problem and implements a unique equilibrium— each agent exerts full effort at his key task. The modular assignment is optimal if every key task is sufficiently specialized. A key task exhibits high specialization if linked to more peripheral tasks than key tasks. Characterizing the optimal task assignment beyond key-peripheral networks with sufficiently specialized key tasks requires additional assumptions on the project structure and cost function.

*I am indebted to Evan Sadler for his encouragement, guidance, and careful readings. I thank Melanie Chen, Zhiming Feng, Navin Kartik, Jacopo Perego, and audiences at the 35th Stony Brook International Conference on Game Theory and Columbia Microeconomic Theory Colloquium for their helpful comments.

†Columbia University— sy3012@columbia.edu.

1 Introduction

A principal and her team of agents work on a project with multiple tasks. Effort spent completing a task generates positive spillovers and is a strategic substitute for effort spent finishing similar tasks. For example, a piece of code written for a particular application in software development can be referenced by team members creating complementary features.[1] Cleaning a critical dataset or annotating an important document facilitates progress on subsequent analyses. Creating a standard manual for sample collection aids different laboratory sub-teams in planning various experiments.

A project can be modeled as a network where links between tasks symbolize effort spillovers and substitutability. Take, for instance, a programming team working on a phone app with the following coding tasks: task A is a call feature, task B is a music playback capability and task C is a sound recording component. Tasks A and B require access to a device's speakers while tasks A and C require control over a device's microphone. Tasks B and C do not have an evident overlap; therefore, the project can be represented as a network with links between tasks A and B, and tasks A and C.

The main novelty of the model is that the principal possesses task assignment power in place of conventional contracting means. In most organizations, a separate compensation and benefits department, not the manager herself, designates salaries and bonus rates for employees. This paper analyzes how a fundamental managerial responsibility in assigning tasks can influence project outcomes.

The incentives for agents are standard. Each agent is evaluated by the number of tasks he completes. An agent finishes a task if he exerts enough effort that together with the sum of efforts at neighboring tasks meets an exogenous completion threshold. Effort provision is subject to a convex cost and for simplicity, I assume perfect effort substitutability between related tasks. A binary evaluation of task completion is practical in many settings. For instance, progress made towards fixing a piece of software that crashes is futile if the principal needs it to be fully debugged for deployment. Cleaning half of a dataset is impractical; the principal needs the whole dataset scrubbed to uphold the integrity of subsequent analyses.

The principal assigns agents to tasks with the objective of completing as many tasks as possible while keeping total team effort costs low. The latter captures the fact that managerial evaluations often assess how well the manager builds, supports, and motivates her team (Hyväri (2006); Chen and Lee (2007)). Managing team effort costs

[1] Code is typically shared on a team-accessible GitHub page.

is critical to a favorable performance review for the principal because they measure employee fatigue, a key contributor to burnout and low morale (Maslach and Leiter (2016)).[2] The main analysis assumes there are as many agents as tasks so the principal does not have to worry about staffing shortages when she assigns tasks.

The fundamental trade-off in the model involves balancing two sources of effort costs: free-riding and convexity. The principal wants agents to exert high effort at central tasks with many links to take advantage of outsized positive effort spillovers, but they will inevitably shirk and free-ride off other agents' efforts at related tasks. This leads to an overall increase in effort costs because every agent assigned to a linked task must increase their efforts to compensate. To combat free-riding and discourage shirking, the principal should assign additional linked tasks to central task assignees so they internalize their spillovers. Nevertheless, this raises effort costs due to convexity. Convex effort costs push in the opposite direction—the principal should instead assign each agent to fewer tasks and encourage a balanced effort distribution among agents. The optimal task assignment depends on how this tension between free-riding and convexity manifests in particular network structures.

I focus on a class of networks with a dichotomous classification of tasks: tasks linked to one other task (peripheral tasks) and tasks linked to at least two peripheral tasks (key tasks). I call this class of networks key-peripheral. Key-peripheral networks are simple project structures and include a variety of commonly studied graphs such as stars, connecting stars, hierarchical structures like trees with two leaves for every internal vertex, etc. They model software development projects particularly well because they embody a basic design mechanism— inheritance. Inheritance champions code reuse in constructing parent classes and methods so that subclasses and auxiliary methods can reference them when building specific functionalities.[3] Creating parent classes and building general libraries of functions are key tasks while coding accessory methods and features correspond to peripheral tasks. Excluding classes and methods that are neither key nor peripheral avoids adding unnecessary design complexity and depth of inheritance that make it difficult to predict how the software behaves (see Chidamber and Kemerer (1994) and Subramanyam and Krishnan (2003)). Key-peripheral task

[2] A 2022 Forbes article reported 67% of survey respondents experiencing some level of burnout resulting from feeling overworked.

[3] See IBM's primer on inheritance. Figure 4 of the guide displays the code architecture for an elementary bank account system, which is key-peripheral with one key task and two peripheral tasks. While the graph is directed to illustrate inheritance relationships, effort spillovers and substitutability between related coding tasks are bidirectional.

structures are prominent in settings beyond software development projects. Cleaning and preparing a dataset is a key task, and analyses that use different subsets of the data are peripheral tasks.[4] In laboratories, developing a standard operating procedure for a laboratory instrument is a key task, and different experiments that use said instrument are peripheral tasks.

As a preliminary analysis, I fully characterize the first-best assignment and effort profile on key-peripheral networks. The principal assigns each agent to one task. The efficient effort level at a key task is strictly less than the completion threshold and increases with the number of linked peripheral tasks. Importantly, it does not depend on the number of neighboring key tasks. The efficient effort level at each peripheral task is strictly positive but lower than that of its key task. Spreading efforts across tasks relieves high convex costs in the first-best.

To tackle optimal assignments (where agents behave strategically), I turn to a familiar concept in organizational economics: modularization.[5] The modular task assignment, which assigns each agent to a module consisting of a key task and its peripheral tasks[6], achieves the delicate balance between mitigating free-riding and convexity. Each agent always exerts full effort at his key task and zero effort at his peripheral tasks, thereby resolving the free-rider problem. The modular design ensures agents are assigned to tasks that are related to each other, enabling them to effectively internalize their own spillovers without having to overexert and face steep convex costs. Modularization performs remarkably well with regards to implementability and optimality.

The first main result is that the modular assignment implements a unique equilibrium where full effort is played at key tasks and zero effort everywhere else. The equilibrium takes a specific graph theoretic structure— the tasks with full effort form a minimum dominating set. A *dominating set* is a set of tasks that together with their neighboring tasks constitute the whole network; a minimum dominating set is a dominating set of minimum size. Minimum dominating effort profiles, which prescribe full effort at every task in a minimum dominating set and zero elsewhere, exploit effort substitutability

[4]Free-riding takes the following form: team members clean sections of the dataset relevant to their respective analyses. The analyst in charge of cleaning and preparing the whole dataset free-rides and just compiles the various cleaned subsets of the data together.

[5]The idea of modularization was first popularized by Simon (1965). Since then, it has been a keystone concept in software development and production design. See Parnas (1972) and Garud, Langlois and Kumaraswamy (2003) for more about the history and prevalence of modularization in software development and production design.

[6]A module is analogous to a *package* in software design which typically consists of a parent class and associated subclasses. See Figure 8 in this IBM guide on class diagrams for a diagram.

and complete the entire project with positive effort levels at the minimum number of tasks. Utilizing this language allows to establish a baseline for what a principal can implement when facing an arbitrary project structure. I provide an algorithm that, given any network, constructs a generalized modular task assignment that implements a minimum dominating effort profile as the unique equilibrium.

The second main result is that the modular assignment is optimal for key-peripheral projects with sufficiently specialized key tasks. The *specialization* of a key task is the ratio of its number of peripheral tasks to the number of key tasks it is linked to. Specialization neatly captures the trade-off between mitigating free-riding and convexity in key-peripheral projects.

When key tasks are highly specialized, alternate assignments that reduce costs via convexity perform poorly because they must leave entire sets of peripheral tasks incomplete in equilibrium. Spreading effort costs involves assigning a different agent to each task within a module and implementing high (but not full) key task effort and low (but nonzero) peripheral task efforts. The total efforts that contribute to completing the key task exceeds the completion threshold, and the assigned agent will free-ride off peripheral efforts and shirk. Free-riding can be avoided only if he is assigned an extra neighboring key task, which I call a supporting key task.[7] The sum of efforts played at the supporting key task and its linked tasks must be *exactly* the completion threshold in equilibrium; otherwise, the agent has a profitable deviation by reducing his effort at the original key task. Now, finishing a peripheral task of the supporting key task implies the efforts played at the peripheral and supporting key task sum to at least the threshold. Then the total efforts that contribute to completing the supporting key task, which includes the effort played at the original key task, is greater than the completion threshold, contradicting equilibrium. All in all, if every key task is sufficiently specialized, then reducing costs via convexity and sacrificing entire sets of peripheral tasks is suboptimal. The modular assignment is optimal.

High key task specialization is analogous to the concept of *loose coupling* in systems design, which entails a low degree of interdependence between modules within a system. Projects with specialized key tasks/loosely coupled modules are common and possess several appealing properties: easy testability, simple scalability, reduced maintenance costs, and increased immunity to catastrophic cascading failures (Stevens, Myers and

[7]That is, if there exists a neighboring key task to support this scheme. Assigning him to neighboring peripheral tasks encourages him to exert full effort at the key task, replicating the modular assignment payoff.

Constantine (1999)).

For projects with low key task specialization, optimality depends on the specific cost function. The modular assignment still resolves the free-rider problem, but the principal could achieve a higher payoff by reducing costs via convexity in the following manner: designate a key task with low specialization to support spreading effort costs at multiple neighboring key tasks while sacrificing its own peripheral tasks. Whether or not this scheme is profitable requires explicit computations. I show that if a project contains a key task with specialization below one, then there exists a cost function that makes the modular assignment optimal and another cost function that makes it suboptimal.

Moving beyond key-peripheral projects, I study the performance of the modular task assignment on a broader class of networks that consist of key, peripheral, and intermediary tasks. Intermediary tasks are helper tasks that aid progress at multiple key tasks. They are linked to at least two key tasks but not any peripheral tasks. An identical null result regarding optimality applies: for any project composed of key, peripheral, and intermediary tasks, there exists a cost function that makes the modular assignment optimal and another cost function that makes it suboptimal. Now, incorporating tasks that are not key, peripheral or intermediary makes the analysis decidedly more involved. Determining what effort profiles an assignment can implement as equilibria becomes a fickle exercise that depends on the exact task structure.[8] All together, this establishes a boundary about the main result. Key-peripheral networks with sufficiently specialized key tasks are, in a sense, a large class of interpretable project structures for which a simple characterization of the optimal task assignment exists *without* additional assumptions regarding the network structure and cost function.

To be transparent, I make two substantive assumptions in the main analysis: perfect substitution of effort between linked tasks and no staffing shortages. Relaxing the first assumption, I show that the modular assignment is optimal (given sufficiently specialized key tasks) so long as the degree of effort substitutability between tasks is sufficiently high. Robustness of the main result is a critical property because related tasks, in practice, may exhibit moderate to high levels of effort substitutability, but not perfect substitutability. With regards to understaffing, the main optimality result still holds so long as the number of agents weakly exceeds the number of key tasks because the modular assignment is still feasible. Nevertheless, an additional trade-off appears when the team size drops below that number. Whether it is profitable to assign each agent more tasks and overwork them or assign each agent fewer tasks and leave tasks incomplete depends

[8] See Example 4 in Section 7.2.

on the specific network structure and cost function.

Related Literature

This paper contributes to the literature on network games, principal-agent problems, and team theory in organizations.[9] In the networks literature, agents are typically treated as nodes in the network. That notion is decoupled in my model: tasks make up the network, and a principal assigns agents to tasks. The set-up of my model is most similar to Bramoullé and Kranton (2007), who study the provision of a public good that cannot be excluded along links in a network. Equilibria feature free-riding and take the shape of maximal independent sets of the network: agents who exert positive effort do not neighbor each other. Introducing task assignments expands the set of equilibria to include minimum dominating effort profiles. Crucially, minimum dominating sets may not always be independent sets. Many recent papers focus on public good provision in networks and discuss the free-rider problem (see Allouch (2015); Kinateder and Merlino (2017, 2023); Elliott and Golub (2019)). In more general analyses, Bramoullé, Kranton and D'Amours (2014) determine the structure of all Nash equilibria in network games that exhibit linear best responses. To tackle the prevalent issue of equilibrium multiplicity, Galeotti et al. (2010) incorporate incomplete information into network games. In a setting with complete information, task assignments can resolve the issue of equilibrium multiplicity. In particular, I show that the modular task assignment implements a unique equilibrium.

Optimal contracts with moral hazard have been the common theme in research on principal-agent problems, starting with the classic papers of Mirrlees (1976) and Holmström (1979). Many variations and extensions have been considered, such as adding a retention rule (Banks and Sundaram (1998)), incorporating dynamics (Holmström (1999)), and endogenizing the information acquisition process (Georgiadis and Szentes (2020)). Holmström (2017) provides an overview of the literature. I do not model uncertainty in my environment; the primary forces that threaten the principal's payoff are free-riding and convex costs. Instead of writing contracts or monitoring agents, the principal can only assign tasks. Task assignments resemble delegation, where the principal

[9] Assignment problems are ubiquitous in operations research, but the treatment does not allow agents to behave strategically nor incorporate incentives and equilibrium analysis. Ahuja, Magnanti and Orlin (1993) provides a comprehensive overview of classical assignment problems. Approaches to solving assignment problems include rewriting them as network flow problems or graph coloring problems.

delegates a set of decisions to the agent from which he makes a choice (e.g. Alonso and Matouschek (2008)). In my model, the decisions an agent faces are related to each other in a network where links indicate effort spillovers and substitutability. In Matouschek, Powell and Reich (2024), a principal designs communication networks for agents trying to match their actions to exogenous local states and coordinate with other agents residing within the same module of production. The authors focus on how a firm's organizational structure, specifically a modular production network, affects the structure of the optimal communication network. In contrast, the principal in my model chooses a particular task assignment, and in doing so, influences the production design rather than taking it as fixed.

Research on team theory has similarly focused on designing optimal incentives in the presence of moral hazard and conflicts of interest (e.g. Groves (1973); Holmström (1982); Che and Yoo (2001); Georgiadis (2015)). Recent work focuses on how a manager with limited commitment chooses project objectives for a team of agents (Georgiadis, Lippman and Tang (2014)), determines the optimal team size when the project is risky (Fu, Subramanian and Venkateswaran (2016)), finds the optimal team composition (Glover and Kim (2021)) and characterizes the optimal incentive structure for agents with production spillovers (Dasaratha, Golub and Shah (2024)). I contribute to the literature on team theory by studying how a manager should optimally assign tasks to a team of agents where task collaboration is modeled via links in the project architecture— the optimal assignment is the modular assignment given sufficiently high key task specialization.

2 Model

Let $N = \{1, ..., n\}$ be a set of agents and $K = \{1, ..., k\}$ be a set of tasks. Tasks are arranged in a network G and links between tasks indicate the presence of effort spillovers and substitutability. Tasks are assigned to agents who decide how much effort to exert from the nonnegative effort space $\mathbb{R}^+$. The principal chooses an assignment $f : N \to 2^K$, where $f(a)$ is the set of tasks assigned to agent a. The set of viable assignment functions is $F = \{f \mid \forall a, d \in N, f(a) \cap f(d) = \emptyset\}$, ensuring no more than one agent is assigned to a task. An assignment f induces an effort game played by agents $\langle M, ((\mathbb{R}^+)^{|f(a)|})_{a \in M}, (\pi_a)_{a \in M} \rangle$:

- $M \subseteq N$ is the set of assigned agents i.e. agents a with $f(a) \neq \emptyset$.

- $(\mathbb{R}^+)^{|f(a)|}$ is the effort space for agent a. He chooses effort levels at each task in $f(a)$; denote e_i as the effort chosen at task i.
- $\pi_a = \sum_{i \in f(a)} \mathbb{1}\{e_i + \sum_{\ell \in G_i} e_\ell \geq b\} - c(\sum_{i \in f(a)} e_i)$ is the payoff for agent a where:
  - b is an exogenous task completion threshold.
  - G_i is the set of tasks that neighbor task i in graph G.
  - $c : \mathbb{R}^+ \to \mathbb{R}^+$ is a C^1, strictly increasing and strictly convex function with $c(0) = 0$ and $c'(0) = 0$.

Agents who remain unassigned receive a payoff of zero. Tasks can also remain unassigned and have zero exerted effort. The total payoff to agent a is the sum of per-task payoffs less the cost of effort, which is a function of his aggregate effort. Completing a task requires at least b units of effort. Tasks are perfect substitutes for each other and the completion threshold can be reached if enough effort is expended on linked tasks. I relax that assumption and consider imperfect effort substitutability in Section 8.

Given a task assignment, agents play a Nash equilibrium in the induced effort game. Every agent chooses effort levels to maximize his payoffs in response to the effort provisions of all other agents. The principal chooses a viable assignment $f \in F$ such that an induced Nash equilibrium $\{\hat{e}_i\}_{i=1}^k$ maximizes the total payoff across the entire network:

$$\max_{f \in F} \sum_{i=1}^{k} \mathbb{1}\{\hat{e}_i + \sum_{\ell \in G_i} \hat{e}_\ell \geq b\} - \sum_{a \in M} c\left(\sum_{i \in f(a)} \hat{e}_i\right)$$

The main assumptions of the model are as follows.

Assumption 1. $c(b) < 1$.

For an agent assigned to exactly one task, his best response is to work just enough to reach the completion threshold.

Assumption 2. *There are at least as many agents as tasks i.e.* $n \geq k$.

The principal does not have to worry about an understaffed team. Whether the principal can implement a certain effort profile depends only on the agents' strategic behavior. This assumption will be relaxed in Section 9.

3 Key-Peripheral Networks

The main analysis focuses on the class of key-peripheral networks.

Definition 1. A **peripheral task** is linked to only one other task. A **key task** has at least two neighboring tasks that are peripheral and the set of all peripheral tasks of a key task is its **peripheral set**. A network is **key-peripheral** if every task is either key or peripheral.

An agent spending effort on a key task expedites progress at many other auxiliary tasks. Peripheral tasks are specialized offshoot tasks— they synergize well with only one other task in the project. Key-peripheral networks, which contain only key and peripheral tasks, model simple task structures. It is a flexible and interpretable class of networks that includes star graphs, connecting star graphs, and tree graphs where every branch node (including the root) has at least two leaf nodes.

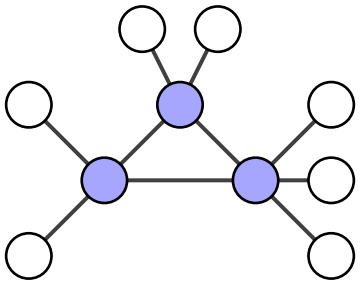


Figure 1: A Key-Peripheral Network Containing a Cycle

Note that the definition of key-peripheral networks does not preclude cycles. In the figure above, the blue task nodes, which induce a cycle, are key tasks while the white task nodes are peripheral.

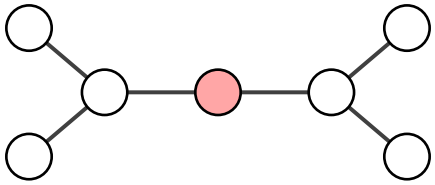


Figure 2: A Non-Key-Peripheral Graph

Key-peripheral networks exclude intermediary tasks that facilitate progress at key tasks but do not have offshoot tasks themselves. The figure above depicts a network

that contains a red task which is neither key nor peripheral. I show in Section 7.2 that characterizing optimal task assignments for networks that are not key-peripheral depends non-trivially on the particular network structure and the cost function.

4 Free-Riding versus Convexity

I illustrate the central tension in the model between free-riding and convex costs with a simple example. Let the net payoff to an agent a assigned to tasks in $f(a)$ is $\sum_{i\in f(a)} \mathbb{1}\{e_i + \sum_{\ell\in G_i} e_\ell \geq \frac{6}{7}\} - (\sum_{i\in f(a)} e_i)^2$. The completion threshold is $\frac{6}{7}$ and the cost of effort is quadratic.

(a) Problem: Free-Riding (b) Problem: Convexity

Figure 3: Free-Riding versus Convexity on the Connecting Star Graph

Each color represents a different assigned agent. In describing assignments and their induced effort profiles, I say that an assignment **implements** an effort profile if it is a Nash equilibrium in the effort game induced by the assignment. The left assignment is the trivial assignment which assigns each agent to one task and the depicted effort profile completes every task while spreading the effort costs among many agents. However, the trivial assignment does not implement it as an equilibrium because of free-riding; key task agents will shirk reduce their efforts. To alleviate free-riding, the principal can assign every task to one agent, as she does in the right assignment. By "selling the project to one agent," the agent now internalizes all of his positive effort spillovers. Unfortunately, convexity now takes hold and prevents the blue agent from completing every task. It is simply too costly to do so.[10] The right assignment implements an equilibrium where the agent plays full effort at only one of the key tasks.

[10]If the cost function was concave, then clearly the principal should always assign all of the tasks to one agent. In this scenario, the first-best coincides with the agent's problem, making the analysis trivial.

The assignments illustrated in Figure 3 can be seen as the two extremes. Intuitively, the optimal task assignment should strike a balance between mitigating free-riding and convexity. The modular assignment, as defined next, achieves exactly that.

Definition 2. A **module** is a set of tasks containing a key task and its peripheral set. On key-peripheral networks, a **modular assignment** assigns each agent to one module within the network (until all modules are assigned).

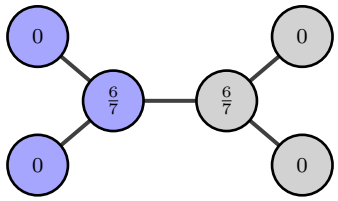


Figure 4: A Modular Assignment on the Connecting Star

Figure 4 shows a modular assignment on same connecting star project. The principal assigns two agents, blue and gray, to modules consisting of a key task and its associated peripheral tasks. Each agent will in turn choose to exert all of his effort on completing the key task regardless of the other agent behaves. The modular design obliges each agent to internalize their spillovers, guaranteeing they do not free-ride. While agents are assigned multiple tasks, they are not assigned dissimilar tasks, saving them from exerting steep levels of effort and enduring high convex costs.

5 The First-Best Assignment and Effort Profile

As a preliminary analysis, I describe the principal's first-best assignment and effort profile on key-peripheral networks ignoring agents' incentives. The first-best effort profile solves a specific cost-minimization problem:

$$\min_{\{e_i\}_{i=1}^k} \sum_{i=1}^{k} c(e_i) \quad \text{s.t. } e_i + \sum_{\ell \in G_i} e_\ell \geq b \text{ and } e_i \geq 0 \text{ for all } i$$

The first-best assignment is the trivial assignment because assigning each agent to one task achieves the lowest cost by convexity.[11] The effort threshold must be satisfied at each task. If not, then the principal is strictly better off by having full effort level b

[11] A convex cost with $c(0) = 0$ satisfies superadditivity, so $c(\sum_{i \in f(a)} e_i) > \sum_{i \in f(a)} c(e_i)$.

exerted at the unfinished task, which increases her total payoff by at least $1 - c(b) > 0$. The following proposition fully characterizes the first-best effort profile on key-peripheral networks.

Proposition 1. *Let i denote a key task and S_i its peripheral set. For all peripheral tasks $h \in S_i$, $c'(e_i) = |S_i| \cdot c'(e_h)$ in the first-best effort profile.*

Proof. The first-order conditions of the cost-minimization problem are:

$$c'(e_i) = \sum_{\ell \in G_i \cup \{i\}} \lambda_\ell + \mu_i \quad ; \quad c'(e_h) = \lambda_i + \lambda_h + \mu_h$$

where λ_ℓ is the Lagrange multiplier on the $e_\ell + \sum_{\ell' \in G_\ell} e_{\ell'} \geq b$ constraint and μ_i is the Lagrange multiplier on the nonnegativity constraint. In the first-best, $e_i + e_h = b$ for all $h \in S_i$, otherwise the total cost is not minimized. Thereby $e_h = e_{h'}$ for all $h, h' \in S_i$. Before proceeding, I state a helpful claim, which I prove in the Appendix.

Claim: For all $h \in S_i$, $e_h > 0$ in the first-best.

By the claim, $e_h > 0$ and so $\mu_h = 0$. Moreover, $e_i + |S_i| \cdot e_h > b$, which implies $\lambda_i = 0$. The first order condition can be simplified:

$$c'(e_h) = \lambda_i + \lambda_h + \mu_h \implies c'(e_h) = \lambda_h$$

$$c'(e_i) = \lambda_i + \sum_{h \in S_i} \lambda_h + \sum_{j \in G_i \setminus S_i} \lambda_j + \mu_i = |S_i| \cdot c'(e_h) + \sum_{j \in G_i \setminus S_i} \lambda_j$$

Since all other neighboring task nodes $j \in G_i \setminus S_i$ are also key tasks, then $\lambda_j = 0$. Then $\sum_{j \in G_i \setminus S_i} \lambda_j = 0$ and $c'(e_i) = |S_i| \cdot c'(e_h)$. □

Proposition 1 pins down the efficient effort level at every task in a key-peripheral network. Key task effort levels solve $c'(e_i) = |S_i| \cdot c'(b - e_i)$ while peripheral task effort levels solve $c'(b - e_h) = |S_i| \cdot c'(e_h)$. The principal wants high (but not full) effort at key tasks to magnify effort spillovers. The efficient effort level at a key task increases with the size of its peripheral set because the ratio of the marginal costs $\frac{c'(e_i)}{c'(b-e_i)}$ is strictly increasing in e_i due to strict convexity. Peripheral task efforts must be low but nonzero because spreading the effort levels across tasks helps keep costs down.

The first-best analysis provides a glimpse at the advantage of modularization in mitigating free-riding. The first-best effort at a key task effort is independent of the efforts played at other key tasks. As we saw in the previous section, the modular

assignment achieves this by inducing each agent to exert full effort at their key tasks regardless of how the other agent acted. In Section 7, I will show that as the size of each peripheral set increases, the modular assignment payoff approaches the first-best.

6 Unique Implementability of Modular Assignments

In this section, I turn to the problem of optimal task assignments where agents behave strategically. I first focus on the unique implementability of modularization.

Theorem 1. *The modular assignment implements a unique effort profile that specifies full effort on key tasks and zero effort everywhere else.*

Proof. The modular assignment assigns each agent to a module. Each assigned agent plays effort b at his assigned key task i and zero effort at every task in the peripheral set S_i. Assume by contradiction that he chooses some other effort level $e_i \in [0, b)$ in equilibrium. Then playing effort $e_h = b - e_i$ at one of his peripheral tasks h ensures the completion of two tasks (i and h) at a cost of $c(e_h + e_i) = c(b)$. But $|S_i| \geq 2$, so there is at least one other assigned task that is left incomplete. Setting $e_i = b$ completes more tasks with a weakly lesser cost. □

The modular assignment rectifies the equilibrium multiplicity problem that is frequently encountered in network games. The unique equilibrium can be implemented by assigning each agent to a key task and two peripheral tasks instead of the entire peripheral set. Nevertheless, assigning agents to the entire peripheral set matters for the extension on imperfect substitutability. As I will show in Section 8, assigning agents to more peripheral tasks within their module allows lower levels of effort substitutability to support optimality. The unique equilibrium takes a particular graph theoretic structure— a minimum dominating effort profile.

Definition 3. For a graph G and a subset D of the vertex set $V(G)$, let $G[D]$ be the set of vertices of G which are in D or adjacent to a vertex in D. If $G[D] = V(G)$, then D is a **dominating set** of G. A dominating set of the smallest size is a **minimum dominating set**; the size of a minimum dominating set is called the **domination number** $\gamma(G)$.[12] A **minimum dominating effort profile** prescribes effort b at tasks

[12]Dominating sets are well-studied by graph theorists. For further reference, see West (2001). In general, finding a minimum dominating set (and therefore the domination number) is an NP-hard problem where a brute force algorithm would have a time complexity of $O(2^n)$. On key-peripheral networks, the unique minimum dominating set is the set of key tasks.

in a minimum dominating set and zero effort everywhere else.

Minimum dominating effort profiles are attractive for two reasons. First, they exploit effort substitutability and complete every task in the project with positive effort exerted at the least number of tasks. Second, minimum dominating effort profiles are universally implementable, not just on key-peripheral networks.

Theorem 2. *Every minimum dominating effort profile of a network is implementable.*

Proof. Fix a network and denote D_{min} as its minimum dominating set. Consider the following constructive algorithm:

Algorithm 1

1: Assign each task in D_{min} to an unassigned agent
2: **for** each assigned agent **do**
3: **if** his assigned task i neighbors another task in D_{min} **then**
4: Assign him to an extra task $j \notin D_{min}$ such that $G_j \cap D_{min} = \{i\}$
5: **end if**
6: **end for**

An agent assigned to a single task will always play effort b to reap a net payoff of $1 - c(b) > 0$. For an agent assigned to two tasks, playing effort b at his assigned task in D_{min} is an equilibrium action. If he decreases his effort at that task, he loses the task completion payoff of the extra task not in D_{min}. Any deviation that keeps the sum of his efforts across the two tasks equal to b yields the same payoff as before. Such an extra task $j \notin D_{min}$ with $G_j \cap D_{min} = \{i\}$ on line 4 of Algorithm 1 always exists. Suppose not, then every neighboring task of i is also linked to some task in $\ell \in D_{min}$ with $\ell \neq i$. Then $D_{min} \setminus \{i\}$ dominates the graph, which contradicts D_{min} being a minimum dominating set. Thus the assignment created from Algorithm 1 implements a minimum dominating effort profile as a Nash equilibrium. □

In a minimum dominating effort profile, agents assigned to neighboring tasks will free-ride. To counteract this, Algorithm 1 assigns each of those agents to an extra task, inducing them to internalize their spillovers and exert full effort at the task in the minimum dominating effort profile. While every minimum dominating effort profile is implementable, they may not be *uniquely* implementable. The assignment constructed via Algorithm 1 can induce other Nash equilibria that are undesirable. Take the following network where the square tasks constitute a minimum dominating effort profile and each color represents a different assigned agent according to Algorithm 1.

(a) Good Equilibrium (b) Bad Equilibrium

Figure 5: A Minimum Dominating Effort Profile (Not Uniquely Implementable)

Unfortunately, this assignment also implements an undesirable equilibrium where every agent exerts effort level b at their assigned circle tasks instead of the square tasks in the minimum dominating set. This leaves the white circle tasks unfinished. Fortunately, with some minor adjustments, we can always find a minimum dominating effort profile of a network that is implementable as the unique Nash equilibrium.

Theorem 3. *There exists a minimum dominating effort profile implementable as the unique Nash equilibrium for any network.*

Proof. Consider Algorithm 2, which takes as input a minimum dominating set and outputs a minimum dominating set.

Algorithm 2

Require: D_{min} is a minimum dominating set
1: Implement D_{min} via Algorithm 1
2: **for** each agent assigned to a pair of tasks $i \in D_{min}$ and $j \notin D_{min}$ **do**
3: **if** $\tilde{D} = (D_{min} \cup \{j\}) \setminus \{i\}$ dominates the network **then**
4: **return** $\tilde{D}$
5: **end if**
6: **end for**
7: **return** D_{min}

Start with an arbitrary minimum dominating set and recursively call Algorithm 2 until a fixed point (a minimum dominating set) is reached. Note that each call to Algorithm 2 returns a minimum dominating set because $\tilde{D}$ within the loop satisfies $|\tilde{D}| = |D_{min}|$. Since task $j \notin D_{min}$ satisfies $G_j \cap D_{min} = \{i\}$ and $\tilde{D} = (D_{min} \cup \{j\}) \setminus \{i\}$, then $G_j \cap \tilde{D} = \emptyset$. Therefore, the set of agents assigned to a pair of tasks shrinks with each recursive step, ensuring a fixed point is reached in a finite number of steps.

Let $\tilde{D}_{min}$ be the minimum dominating set that results from the recursion of Algorithm 2. Now construct the **generalized modular assignment** via Algorithm 3:

Algorithm 3

Assign each task in $\tilde{D}_{min}$ to an unassigned agent
for each assigned agent **do**
    if his assigned task i neighbors another task in D_{min} **then**
        Assign him to tasks $j, k \notin \tilde{D}_{min}$ such that $G_j \cap \tilde{D}_{min} = G_k \cap \tilde{D}_{min} = \{i\}$
    end if
end for

There exists at least one extra task j such that $G_j \cap \tilde{D}_{min} = \{i\}$ by the same argument in Theorem 2. Suppose that there does not exist another task $k \neq j$ with the same property. Then $(\tilde{D}_{min} \cup \{j\}) \setminus \{i\}$ is a minimum dominating set, which contradicts that $\tilde{D}_{min}$ was the output fixed point of Algorithm 2.

Like before, an agent assigned to a single task will always play effort b to reap a net payoff of $1 - c(b) > 0$. Agents assigned to three tasks will always play effort b at his assigned task $i \in \tilde{D}_{min}$ and zero effort at his two other tasks not in $\tilde{D}_{min}$. If he chooses some other effort level $e_i \in [0, b)$ in equilibrium, then playing effort $e_h = b - e_i$ at one of his other tasks h ensures the completion of two tasks (i and h) at a cost of $c(e_h + e_i) = c(b)$. But there is one other assigned task that is left incomplete. Thus $e_i = b$ completes strictly more tasks with a weakly lesser cost for a higher net payoff. □

The proof of Theorem 3 retains the spirit of modularization. First, the recursion of Algorithm 2 builds a minimum dominating set $\tilde{D}_{min}$ with a special property: tasks in $\tilde{D}_{min}$ that neighbor each other are linked to at least two extra tasks with only one neighboring task in $\tilde{D}_{min}$. These additional assigned tasks are similar to peripheral tasks— they are linked to only one task residing in a minimum dominating set. Applying the recursion to the minimum dominating set in Figure 5 ends after one step. The output is shown in Figure 6 where the square tasks constitute the minimum dominating set.

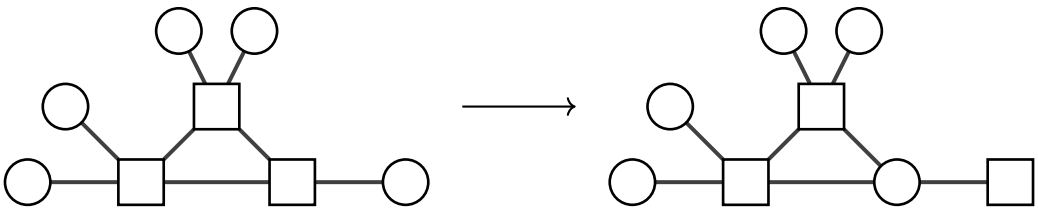


Figure 6: Algorithm 2 Recursion Output

The minimum dominating effort profile associated with $\tilde{D}_{min}$ is implementable as the unique Nash equilibrium. Similar to before, we worry that agents who are assigned to

neighboring tasks will free-ride. The **generalized modular assignment** constructed by Algorithm 3 does the trick by assigning each of those agents to two extra peripheral-like tasks as Figure 7 illustrates.

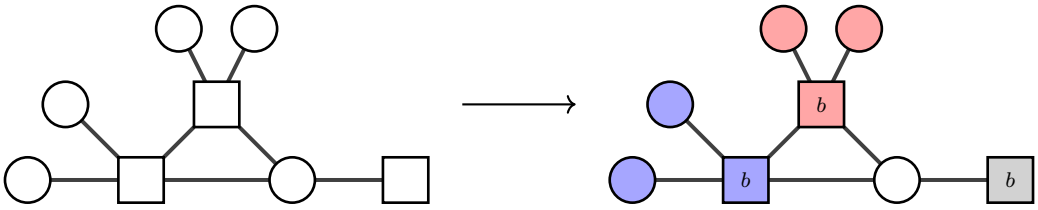


Figure 7: Generalized Modular Assignment via Algorithm 3

7 Optimality of Modular Assignments

As we saw in Theorem 1, the modular assignment implements an effort profile where every agent exerts full effort at his assigned key task and zero elsewhere. Every task in the project is completed. An assignment that achieves a higher net payoff must incur a lower cost by taking advantage of convexity, but doing so comes at a trade-off: some peripheral set of a module must remain unfinished.[13]

Proposition 2. *If an assignment achieves a strictly higher payoff than the modular assignment, then it must implement effort $e_i \in (0, b)$ at some key task i and leave the entire peripheral set of a neighboring key task $j \in G_i$ unfinished.*

Proof. To achieve a strictly higher payoff than modularization, an assignment must have a lower total cost. It must implement an effort profile with effort $e_i \in (0, b)$ at some key task i because implementing effort $e_i = 0$ is strictly worse and $e_i = b$ just replicates the payoff to modularization. The assignment must complete every peripheral task of a key task i with $e_i \in (0, b)$. If not, then the modular assignment achieves a strictly higher payoff by Assumption 1.

Let S_i denote key task i's peripheral set. To complete every task in S_i, each peripheral task must be assigned to agents who each exert effort $b - e_i$. The effort threshold at key task i is at least $e_i + |S_i|(b - e_i) > b$, causing the key task agent to shirk. This can be supported as an equilibrium only if the key task agent is also assigned to some neighboring key task $j \in G_i$ that specifically requires effort level e_i. The completion

[13]This trade-off does not always exist for arbitrary networks. See Example 3 in Section 7.2.

threshold at key task j must bind b i.e. $e_j + \sum_{\ell \in G_j} e_\ell = b$. If not, the agent will again have a profitable deviation by decreasing his effort at task i. Every peripheral task $\ell \in S_j$ has to be left unfinished in equilibrium since $e_\ell + e_j < e_\ell + e_j + e_i \leq e_j + \sum_{k \in G_j} e_k = b$. $\square$

Proposition 2 pares down the space of potentially better assignments to ones implementing effort profiles that are *locally* different from the minimum dominating effort profile. A direct implication is that the modular assignment is optimal on projects with disconnected key tasks. Moreover, the first-best effort profile, which completes every task with key task efforts less than b, is never implementable. The next example applies Proposition 2.

Example 1. Take the key-peripheral network shown below with completion threshold $b = \frac{1}{2}$ and a quadratic cost function. The first-best effort profile prescribes an effort of $\frac{1}{3}$ at each key task and $\frac{1}{6}$ at each peripheral task by Proposition 1. Assigning each agent to a different task can implement those efforts on the diamond module:

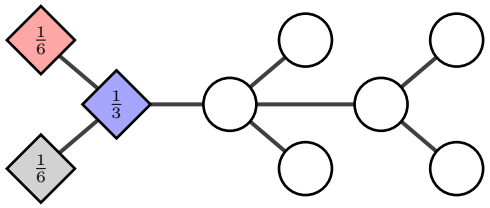


Figure 8: First-Best on the Diamond Module

All tasks within the diamond module are completed at a cost of $2 \cdot (\frac{1}{6})^2 + (\frac{1}{3})^2$, which is lower than the cost of the modular assignment of $(\frac{1}{2})^2$. Observe that the sum of efforts at the diamond key task exceeds the completion threshold; the blue agent will always seek to decrease his effort. The only way to sustain an effort level of $\frac{1}{3}$ in equilibrium is to assign the blue agent to a neighboring key task that *requires* the diamond task effort to be exactly $\frac{1}{2}$.

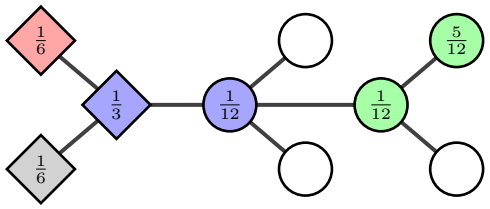


Figure 9: An Equilibrium Where Blue Circle Supports Blue Diamond

Assigning the blue agent to the neighboring circle key task supports the diamond key task effort of $\frac{1}{3}$ in equilibrium but Proposition 2 provides strict equilibrium restrictions. The sum of efforts at the blue circle task must be equal to $\frac{1}{2}$, so its peripheral tasks must remain unfinished in equilibrium.

The central tension in the model lies in resolving free-riding by assigning more tasks to each agent versus alleviating convex costs by assigning fewer tasks to each agent. On one hand, if every module is large, then implementing an equilibrium that exploits convexity is a bad idea. Doing so leaves entire peripheral sets unfinished, making modularization, which completes every task, more attractive. On the other hand, if a key task is linked to a lot of other key tasks, then designating said key task to support spreading effort at multiple neighboring modules could be favorable. Whether this scheme reduces costs enough to compensate for sacrificing a set of peripheral tasks requires additional information regarding the network structure as will be shown in Section 7.1. To capture the trade-off between free-riding and convexity, I introduce the specialization of a key task.

Definition 4. The **specialization of key task** i is $\sigma_i = \frac{|S_i|}{|G_i \setminus S_i|}$ where S_i is its peripheral set.

The specialization of a key task is the ratio of its number of peripheral tasks to its number of linked key tasks. If a key task is not linked to any other key tasks, then it has infinite specialization. Modularization is advantageous if every key task is highly specialized because reducing costs via convexity necessarily leaves large sets of peripheral tasks incomplete. In contrast, if a key task exhibits low specialization, then taking advantage of convexity and leaving tasks incomplete can be beneficial.[14] This leads to the main result concerning the optimality of the modular assignment.

Theorem 4. *Let $\mathcal{K}(G)$ be the set of all key tasks in key-peripheral network G. If $\min_{i \in \mathcal{K}(G)} \sigma_i \geq c(b)$, then the modular assignment is optimal. The optimal number of assigned agents is the domination number $\gamma(G)$.*

Proof. Fix an alternate assignment that implements intermediate effort $e_i \in (0, b)$ at some key task i. Let L be the set of all key tasks with implemented efforts in $(0, b)$ and completed peripheral tasks. Let U be the set of supporting key tasks. U is nonempty and the peripheral sets of key tasks in U are not finished in equilibrium by Proposition

[14] Example 2 shows how this can be profitable for a particular cost function.

2. It is without loss to compare payoffs for modules with key tasks residing in L. The modular assignment does weakly better than the alternate assignment at modules where the key tasks have implemented effort of b or zero. The modular assignment achieves a strictly higher payoff if:

$$\underbrace{\sum_{i\in L}(|S_i|+1)-|L|\cdot c(b)}_{\text{modular assignment payoff}} \geq \underbrace{\sum_{i\in L}(|S_i|+1)+|U|}_{\text{upper bound on alt. assignment payoff}} \iff c(b)\leq \frac{\sum_{i\in U}|S_i|}{|L|}$$

Since every key task in L neighbors a key task in U, we can write:

$$L\subseteq \bigcup_{i\in U}(G_i\setminus S_i) \implies |L|\leq \sum_{i\in U}(|G_i\setminus S_i|)$$

The following inequalities hold:

$$\frac{\sum_{i\in U}|S_i|}{|L|}\geq \frac{\sum_{i\in U}|S_i|}{\sum_{i\in U}(|G_i\setminus S_i|)}\geq \min_{i\in U}\frac{|S_i|}{|G_i\setminus S_i|}\geq \min_{i\in\mathcal{K}(G)}\frac{|S_i|}{|G_i\setminus S_i|}=\min_{i\in\mathcal{K}(G)}\sigma_i$$

If $\min_{i\in\mathcal{K}(G)}\sigma_i\geq c(b)$, then the modular assignment achieves a strictly higher payoff than any assignment that satisfies the necessary condition outlined in Proposition 2. In other words, no assignment achieves a strictly higher payoff than the modular assignment. The modular assignment assigns as many agents as the number of key tasks, which is the graph domination number $\gamma(G)$. □

The modular assignment assigns $\gamma(G)$ agents and if the total number of agents n exceeds $\gamma(G)$, then the principal should keep $n-\gamma(G)$ agents unassigned.[15] The next result relates the modular assignment payoff to that of the first-best. As peripheral sets of each key task gets larger, the modular assignment payoff converges to that of the first-best.

Proposition 3. *The payoff to the modular assignment converges to the first-best as* $|S_i|\to\infty$ *for each key task* i *in a key-peripheral network.*[16]

[15] The principal can still uniquely implement the minimum dominating effort profile by assigning an agent to a key task and two linked peripheral tasks, and so the upper bound on the number of assigned agents is $k-2\cdot\gamma(G)$ where k is the total number of tasks. Assigning more agents threatens the uniqueness of implementability.

[16] $|S_i|\to\infty$ implies $\min_{i\in\mathcal{K}(G)}\sigma_i\to\infty$ but the converse is not true because $\sigma_i=\infty$ by definition whenever $|G_i\setminus S_i|=0$.

Proof. Let e_i^* be the first-best effort at key task i. By definition of the first-best:

$$\underbrace{\sum_{i\in\mathcal{K}(G)} |S_i| + 1 - \sum_{i\in\mathcal{K}(G)} c(e_i^*) - \sum_{i\in\mathcal{K}(G)} |S_i| c(b - e_i^*)}_{\text{first-best payoff}} \geq \underbrace{\sum_{i\in\mathcal{K}(G)} |S_i| + 1 - \gamma(G)c(b)}_{\text{modular payoff}}$$

$$\implies \gamma(G)c(b) \geq \sum_{i\in\mathcal{K}(G)} c(e_i^*) + \sum_{i\in\mathcal{K}(G)} |S_i| c(b - e_i^*)$$

Recall the first-best key task effort solves $c'(e_i^*) = |S_i| \cdot c'(b - e_i^*)$, thus $|S_i| \to \infty \implies e_i^* \to b$, which also implies $\sum_{i\in\mathcal{K}(G)} c(e_i^*) \to \gamma(G)c(b)$ by continuity of c. Since the above inequality always holds for the first-best, then $\lim_{|S_i|\to\infty} \sum_{i\in\mathcal{K}(G)} |S_i| c(b - e_i^*) = 0$. Thereby the difference in payoffs between the first-best and the modular assignment vanishes as $|S_i| \to \infty$. □

In the following corollaries, I detail some scenarios where the modular task assignment is always optimal.

Corollary 1. *If the principal needs to complete every task in a key-peripheral project, then the modular assignment is optimal.*

It is realistic to assume settings where the principal must finish the entire project. Proposition 2 rules out any assignment that implements an effort profile with intermediate key task efforts tasks are left incomplete in equilibrium. Implementing an effort profile with zero effort at key tasks is far from optimal; every peripheral task agent must exert effort b to complete all tasks in the project.

Corollary 2. *If every key task is linked to weakly more peripheral tasks than key tasks, then the modular assignment is optimal.*

If every key task is linked to more peripheral tasks than key tasks, then $\min_{i\in\mathcal{K}(G)} \sigma_i \geq 1$. The sufficient condition of Theorem 4 holds since $c(b) < 1$ by Assumption 1.

Corollary 3. *The modular assignment is optimal for linear costs and fixed capacity costs of the form:*

$$c(e) = \begin{cases} 0 & 0 \leq e \leq b \\ +\infty & e > b \end{cases}$$

With fixed capacity costs, there are no benefits to dividing tasks among different agents. With linear costs, it is actually harmful to divide tasks within a module among

different agents. If an agent assigned to key task i exerts any effort less than b, then the total cost to completing the entire module is at least $c(e_i) + |S_i|c(b - e_i) > c(b)$, which is more expensive than the modular assignment. In either case, the trade-off between free-riding and convexity disappears and the optimal assignment must complete every task. The modular assignment does exactly that. The same reasoning applies to arbitrary networks: the generalized modular assignment, which implements a minimum dominating effort profile as a Nash equilibrium by Theorem 3, is optimal for linear and fixed capacity costs.

7.1 When Sufficiency Fails

Part of the appeal of Theorem 4 is that it is agnostic of the functional form of the cost of effort and relies only on computing $c(b)$ and σ_i. The next result shows that finding the optimal assignment on key-peripheral networks with low specialization relies on explicit computations that depend non-trivially on the shape of the cost function.

Theorem 5. *If* $\min_{i \in \mathcal{K}(G)} \sigma_i < 1$*, then there exists a cost function that makes the modular assignment optimal and another cost function that makes it suboptimal.*

Proof. Fix some network G with minimum specialization $\min_{i \in \mathcal{K}(G)} \sigma_i < 1$. By Theorem 4, the modular assignment is optimal for any cost function with $c(b) \leq \min_{i \in \mathcal{K}(G)} \sigma_i$.

Now I construct an alternate assignment and a suitable cost function that makes the modular assignment suboptimal. A cost function satisfying $c(b) > \min_{i \in \mathcal{K}(G)} \sigma_i$ is not enough as Example 2 will demonstrate; the shape of the cost function matters. Label the key task with lowest specialization as i and consider an alternate task assignment that uses i to support intermediate efforts at every neighboring key task: assign all tasks in $\{i\} \cup (G_i \setminus S_i)$ to one agent (call him agent a), each peripheral task $k \in S_j$ for key task $j \in G_i \setminus S_i$ to a different agent, and all other modules following the modular assignment. This assignment keeps the tasks in S_i unassigned.

Now consider the effort profile where agent a plays effort $\frac{b}{|G_i \setminus S_i|}$ at each task in $G_i \setminus S_i$ and zero effort at task i. Each agent assigned to a peripheral task of a key task residing in $G_i \setminus S_i$ plays effort $\frac{b(|G_i \setminus S_i| - 1)}{|G_i \setminus S_i|}$. Every other agent assigned to a module plays effort b at his respective key task and zero at his peripheral tasks.

Agent a does not have a profitable deviation in lowering his effort at any of his tasks lest he loses task i's completion payoff. Every agent assigned to a peripheral task of a key task in $G_i \setminus S_i$ is playing a best response. The alternate assignment implements said

effort profile as a Nash equilibrium with intermediate effort at key tasks in $\{i\} \cup (G_i \setminus S_i)$ and replicates the modular payoff everywhere else. Thus, it is without loss to compare the payoffs between the alternate assignment and the modular assignment on the key tasks residing in $\{i\} \cup (G_i \setminus S_i)$ and their peripheral tasks. The payoff to the alternate task assignment is:

$$\underbrace{\sum_{j \in G_i \setminus S_i} |S_j| + |\{i\} \cup (G_i \setminus S_i)|}_{\text{total \# of tasks}} - \underbrace{c(0 + (|G_i \setminus S_i|) \cdot \tfrac{b}{|G_i \setminus S_i|})}_{=c(b)\ ;\ \text{agent } a\text{'s cost}} - \sum_{j \in G_i \setminus S_i} |S_j| \cdot \underbrace{c(\tfrac{b(|G_i \setminus S_i|-1)}{|G_i \setminus S_i|})}_{\text{each peripheral agent's cost}}$$

The payoff to the modular assignment is:

$$|S_i| + \sum_{j \in G_i \setminus S_i} |S_j| + |\{i\} \cup (G_i \setminus S_i)| - |\{i\} \cup (G_i \setminus S_i)| \cdot c(b)$$

The alternate assignment achieves a strictly higher payoff if:

$$|G_i \setminus S_i| \cdot c(b) > |S_i| + \sum_{j \in G_i \setminus S_i} |S_j| \cdot c(\tfrac{b(|G_i \setminus S_i|-1)}{|G_i \setminus S_i|})$$

The goal is to construct a cost function $c : \mathbb{R}^+ \to \mathbb{R}^+$ that is C^1, strictly increasing and strictly convex with $c(0) = 0$ and $c'(0) = 0$ that satisfies the above inequality. Set $c(b) = \frac{1}{2}(\frac{|S_i|}{|G_i \setminus S_i|} + 1)$. By assumption, $\min_i \sigma_i = \frac{|S_i|}{|G_i \setminus S_i|} < 1$, so $\min_i \sigma_i < c(b) < 1$ and there exists some $\epsilon \in (0, 1)$ such that $c(b) = \epsilon + \frac{|S_i|}{|G_i \setminus S_i|}$. Set $c(\frac{b(|G_i \setminus S_i|-1)}{|G_i \setminus S_i|}) = \frac{\epsilon \cdot |G_i \setminus S_i|}{2m \sum_{j \in G_i \setminus S_i} |S_j|}$ where m is a natural number chosen such that $\frac{\epsilon \cdot |G_i \setminus S_i|}{2m \sum_{j \in G_i \setminus S_i} |S_j|} < \frac{1}{2}(\frac{|S_i|}{|G_i \setminus S_i|} + 1)(\frac{(|G_i \setminus S_i|-1)}{|G_i \setminus S_i|})$, where the latter term is the secant line between $(0, 0)$ and $(b, \frac{1}{2}(\frac{|S_i|}{|G_i \setminus S_i|} + 1))$ evaluated at $\frac{b(|G_i \setminus S_i|-1)}{|G_i \setminus S_i|}$. This ensures convexity is satisfied and the alternate assignment achieves a strictly higher payoff than the modular assignment. We now have three points necessary to construct a polynomial: $(0, 0)$, $(\frac{b(|G_i \setminus S_i|-1)}{|G_i \setminus S_i|}, \frac{\epsilon \cdot |G_i \setminus S_i|}{2m \sum_{j \in G_i \setminus S_i} |S_j|})$ and $(b, \frac{1}{2}(\frac{|S_i|}{|G_i \setminus S_i|} + 1))$. The desired C^1, strictly monotone and convex cost function with a specified endpoint derivative $c'(0) = 0$ can be generated by the Fritsch-Carlson method for shape-preserving piecewise cubic Hermite polynomial interpolation (see Fritsch and Carlson (1980) and Ferguson and Pruess (1991)). □

Example 2. If a key-peripheral network has minimum specialization below one, then characterizing the optimal task assignment requires knowing the exact cost function. The network in Figure 10 has minimum key task specialization of $\frac{2}{3}$. The alternate assignment constructed in the proof of Theorem 5 designates the key task with the

lowest specialization to support intermediate efforts at neighboring key tasks. Each neighboring key task is assigned to the blue agent. Each peripheral task is assigned to a different agent. The peripheral tasks of the blue circle key task are left incomplete.

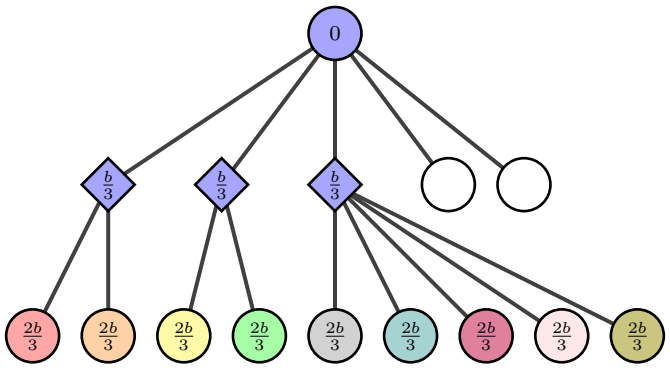


Figure 10: An Alternate Assignment

The total payoff of the implemented equilibrium is $13 - (c(b) + 9 \cdot c(\frac{2b}{3}))$. The payoff from modularization is $15 - 4 \cdot c(b)$, and the alternate assignment does better than the modular assignment if the cost function satisfies $c(b) > \frac{2}{3} + 3 \cdot c(\frac{2b}{3})$. With a completion threshold of $b = 0.99$, the quadratic cost function $c(e) = e^2$ fails the inequality while $c(e) = e^6$ does not. Interestingly, a very convex cost function $c(e) = e^{41}$ fails the inequality yet again because $c(b) = 0.99^{41} \approx 0.6622 < \frac{2}{3}$; the sufficient condition in Theorem 4 is satisfied and the modular assignment is optimal for that particular cost function. Note that the alternate assignment implements multiple equilibria. The general form of an implemented equilibrium is shown in Figure 11 with constraint $e_i + e_j + e_k + e_\ell = b$.

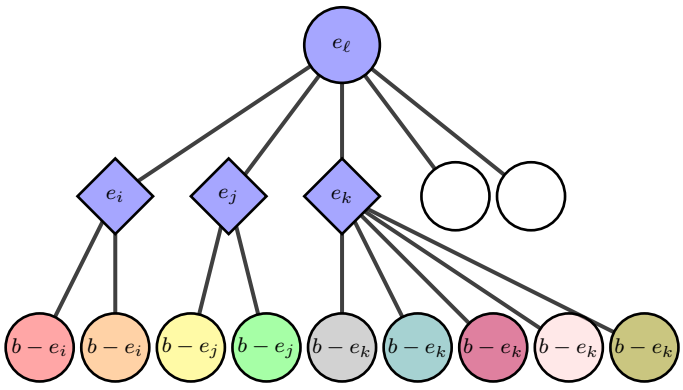


Figure 11: Figure 11: General Form of Equilibria

The optimal equilibrium implemented by the alternate assignment is the solution to the following cost-minimization problem:

$$\min_{e_i,e_j,e_k,e_\ell} c(e_i+e_j+e_k+e_\ell)+2c(b-e_i)+2c(b-e_j)+5c(b-e_k)$$

$$\text{s.t. } e_i+e_j+e_k+e_\ell=b \ ; \ e_i,e_j,e_k,e_\ell \geq 0$$

Of course, the formulation of this optimization problem depended on the particular network structure, and comparing its value to the modular assignment hinges on the exact cost function.

7.2 Beyond Key-Peripheral Networks

A natural step towards a general characterization of optimal task assignments is to include intermediary tasks that act as helper tasks for multiple key tasks.

Definition 5. An **intermediary** task is linked to at least two key tasks but not any peripheral tasks.

For networks that contain intermediary tasks in addition to key and peripheral tasks, local reduction in the space of assignments by Proposition 2 still applies: if an assignment achieves a strictly higher payoff than the modular assignment, then it must implement intermediate effort $e_i \in (0,b)$ at some key task i. However, a null result analogous to the one in the previous subsection holds.

Theorem 6. *For a network composed of key, peripheral and intermediary tasks, there exists a cost function that makes the modular assignment optimal and another cost function that makes it suboptimal.*

Proof. See Appendix. □

I discuss the intuition behind the proof of Theorem 6, but relegate its finer details to the Appendix. To make the modular assignment optimal, the desired cost function should ensure spreading costs via convexity is futile. A cost function that is sufficiently close to being linear, such as $c(e)=(\frac{e}{b\beta})^\alpha$ with β sufficiently large to ensure $c(b)<1$ and α sufficiently close to (but greater than) 1 does the trick. Concerning suboptimality, I construct a cost function and an assignment that beats modularization similar to the proof of Theorem 5. Example 3 illustrates. The main takeaway is that outside the class

of sufficiently specialized key-peripheral networks, characterizing the optimal assignment requires additional assumptions regarding the exact cost function and network structure.

Example 3. The following figure depicts two different assignments on a network with an intermediary task connecting two star graphs.

(a) Modular Assignment (b) Alternate Assignment

Figure 12: A Network with an Intermediary Task

Each color represents a different assigned agent and white tasks remained unassigned. The alternate assignment spreads effort across multiple tasks and achieves a higher payoff if $\frac{c(b)}{4} > c(\frac{b}{2})$. Fixing $b = 0.99$, the cubic cost function $c(e) = e^3$ satisfies this inequality but another cost function $c(e) = e^{4/3}$ fails to do so.

Now, the analysis becomes more fickle upon incorporating tasks that are not key, peripheral or intermediary. Determining whether an effort profile is implementable or not is a delicate exercise that hinges on the exact network structure.

Example 4. With arbitrary project structures, the analysis needs to be tailored to the exact network structure. Take the network from Figure 12 and add an extra task k linked to the intermediary task and one of the key tasks as shown below.

(a) Modular Assignment (b) Tasks to Reduce Costs

Figure 13: A Network with an Extra Task

To beat the modular assignment, an alternate assignment needs to implement intermediate efforts at either key task ℓ or i shown on the right. Let's focus on the first case: implement intermediate effort $e_\ell \in (0, b)$ and assign the peripheral tasks to two different agents who play efforts $b - e_\ell$. As always, the agent assigned to task ℓ needs to be assigned to a supporting neighboring task j with $e_j + \sum_{m \in G_j} e_m = b$ to prevent him from shirking at task ℓ.

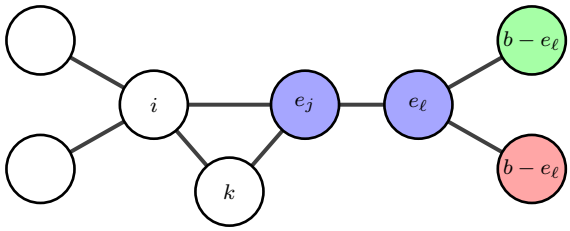
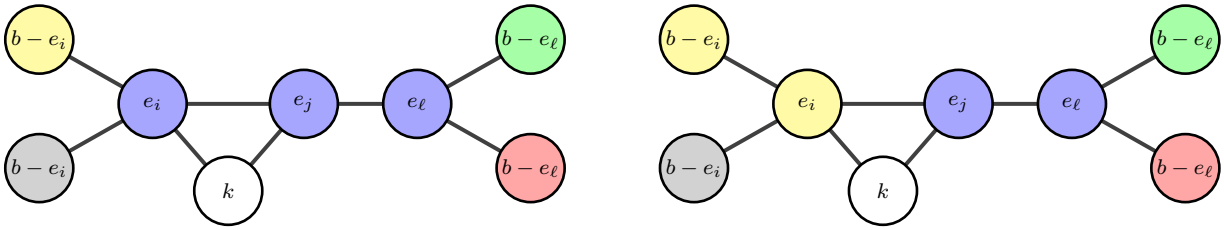


Figure 14: Intermediate Effort $e_\ell \in (0, b)$

Since $G_k \cup \{k\} \subset G_j \cup \{j\}$, we can immediately conclude that $e_j + \sum_{m \in G_j} e_m = b$ implies $e_k + \sum_{m \in G_k} e_m < b$. In other words, task k must be left incomplete and cannot be assigned to any agent in equilibrium, lest that agent will have a profitable deviation to try and complete task k. Moreover, effort e_i must be strictly less than b. This leaves two possible assignments, one where task i is assigned to the blue agent and one where it is assigned to another agent:

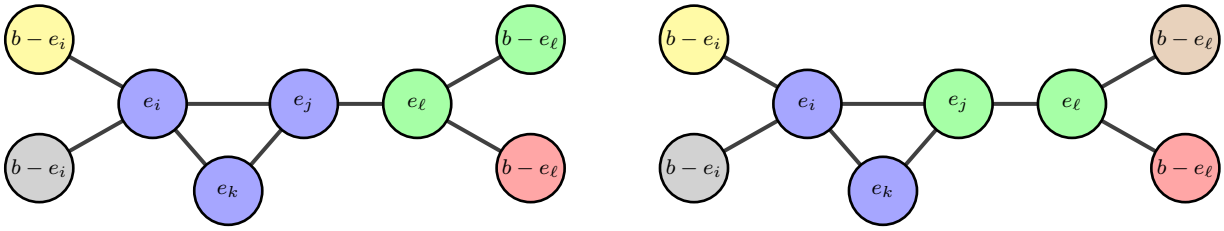


(a) Blue Task i (b) Yellow Task i

Figure 15: Two Possible Assignments

In the left assignment, $e_j + \sum_{m \in G_j} e_m = e_i + e_j + e_\ell = b$ so the blue agent is exerting total effort of b for a cost of $c(b)$. In the right assignment, the yellow agent is assigned to a key task with intermediate effort effort $e_i < b$ and must be assigned to a neighboring peripheral task. The yellow agent will also exert total effort of b for a cost of $c(b)$. Notice that in either assignment, there is one agent who exerts total effort of b for a cost of $c(b)$ and the alternate assignment achieves an upper bound payoff of $7 - c(b)$. The modular assignment achieves a a higher payoff of $8 - 2c(b)$ since $8 - 2c(b) > 7 - c(b) \iff 1 > c(b)$.

Implementing an intermediate effort $e_i \in (0, b)$ is also strictly worse than the modular assignment. The agent assigned to task i must be assigned to a neighboring task. If he is not assigned to task k, then it must remain incomplete and we return to a variation of the assignments shown in Figure 15. Consider the assignments where task k is assigned to that agent.



(a) Blue Task j (b) Green Task j

Figure 16: Two Possible Assignments

The left assignment implements an equilibrium only if $e_i + e_k + e_j = b$ for a total cost of at least $c(e_i + e_k + e_j) + c(e_\ell + b - e_\ell) = 2c(b)$, so it performs strictly worse than the modular assignment. Now let's focus on the right assignment. Since task k is assigned to the blue agent, it will be completed in equilibrium. By the previous analysis, there cannot be an intermediate effort at task ℓ in equilibrium. The right assignment thus accrues a total cost of at least $2c(b)$ in either case of $e_\ell = 0$ or $e_\ell = b$, making it strictly worse than the modular assignment.

We conclude the modular assignment is optimal on this network, albeit it required a long-winded case analysis. It is important to stress that determining which effort profiles are implementable in this network relied specifically on how task k is linked in the network. This changes when we add another task linked to the intermediary task and the other key task.

(a) Modular Assignment (b) Alternate Assignment

Figure 17: Network with Two Extra Tasks

The left assignment depicts the modular assignment. The right assignment achieves a higher payoff if $c(b) > 5 \cdot c(\frac{b}{2})$. The addition of an unrelated task at another region of

the project structure dramatically transformed the space of implementable effort profiles. To conclude, characterizing the optimal task assignment, or even determining whether an effort profile is implementable, relies on exactly how each task is positioned within an arbitrary project structure.

8 Imperfect Substitutability

I introduce an extension of the model that incorporates imperfect substitutability between tasks. The per-task payoff function is given by $\mathbb{1}\{e_i + \delta \sum_{\ell \in G_i} e_\ell \geq b\}$ where δ measures the substitutability of effort at linked tasks. For simplicity, I assume the degree of substitutability is constant across all neighboring tasks. High δ values reflect high levels of effort substitutability where $\delta = 1$ represents perfect substitutability, while low δ values reflect low levels of substitutability. If $\delta = 0$, then links between tasks are superfluous and each task can be treated as a singleton.

Under imperfect substitutability, I require effort $\frac{b}{\delta}$ instead of b at tasks in a minimum dominating set. The performance of the modular assignment depends on the value of δ. If δ is very low, then exerting $\frac{b}{\delta}$ at the key task can be expensive for both the principal and the agent. An adequately high δ alleviates this concern.

Theorem 7. *If* $\min_{i \in \mathcal{K}(G)} \sigma_i \geq c(\frac{b}{\delta})$ *and* δ *is sufficiently high (which can depend on the cost function and network structure), the modular assignment implements the minimum dominating set and is optimal for the principal.*

Proof. See Appendix. □

Example 5. I provide intuition for how δ plays a role in determining the implementability and optimality of modularization. Consider the connecting star graph where one key task is linked to three peripheral tasks and the other is connected to two peripheral tasks.

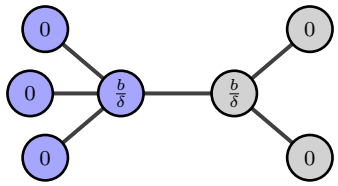


Figure 18: Modular Assignment on the Connecting Star

Implementability: The payoff received by the blue agent is $4 - c(\frac{b}{\delta})$. δ needs to be sufficiently high to prevent the blue agent from having a profitable deviation. We can

inspect all possible deviations the blue agent has while holding his number of completed tasks constant. If the blue agent seeks to complete only one task within his assigned module, then the cheapest way to do so is by shirking and free-riding at the key task; δ must satisfy the incentive-compatible (IC) constraint $1 - c(0) \leq 4 - c(\frac{b}{\delta})$. The least expensive way to complete two tasks within his module is by exerting b at a peripheral task and free-riding at the key task; the corresponding constraint is $2 - c(b) \leq 4 - c(\frac{b}{\delta})$. Continuing this pattern, the modular assignment implements the minimum dominating effort profile for the blue agent if δ satisfies the following IC constraints:

$$4 - c(\tfrac{b}{\delta}) \geq m - c((m-1)b) \text{ for } m \in \{1, 2, 3, 4\}$$

A δ value that satisfies every IC constraint for the blue agent exists since c is continuous and every inequality holds for $\delta = 1$. Under imperfect substitutability, assigning agents to an entire module has its benefits— a greater gross payoff for agents permits lower values of δ. Ostensibly, a higher δ parameter is needed for more convex cost functions. A computation yields that δ needs to be at least 0.469 to satisfy all IC constraints if $b = \frac{3}{4}$ and a quadratic cost. For a cubic cost and the same completion threshold, δ needs to be greater than 0.558.

For the minimum dominating set to be implementable across the entire network, δ must satisfy the IC constraints for all agents. The sufficient level of δ is higher for the gray agent because he is assigned to a smaller module. The smaller the module, the more expensive it is to exert effort $\frac{b}{\delta}$. Thereby, it is sufficient for δ to satisfy the IC constraints for the agent assigned to the smallest module. Repeating the same calculations for the gray agent with $b = \frac{3}{4}$, δ must be at least 0.600 for a quadratic cost, while for a cubic cost, the lower bound for δ is 0.667.

Optimality: I relegate the finer case analysis to the proof in the Appendix. The crucial insight is that if δ is sufficiently large, then the analysis essentially reduces to that of Theorem 4. Modularization is optimal as long as each key task is sufficiently specialized.

9 Understaffing

What kind of task assignments should the principal employ when her team is understaffed? I relax Assumption 2 and allow there to be fewer agents than tasks in the project i.e. $n < k$. How understaffing affects the principal's optimal assignment de-

pends on how big the project is. If $n \geq \gamma(G)$ where $\gamma(G)$ is the domination number of key-peripheral network G, then the modular assignment is still feasible and Theorem 4 holds. However, if $n < \gamma(G)$, then the comparison across assignments depends on the network structure and costs.

Example 6. If $n = 2$, then using an assignment that assigns the blue and gray agents to some modules completes eight tasks out of ten in the project shown in Figure 16.

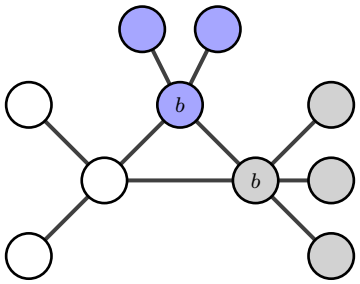


Figure 16: An Assignment with $n = 2$

Assignments that spread effort among tasks within modules reap low payoffs when there is understaffing because they require assigning different agents to peripheral tasks. Instead, the principal should utilize a "coarse" modular assignment and assign agents to multiple modules. This accomplishes more tasks but incurs higher costs.

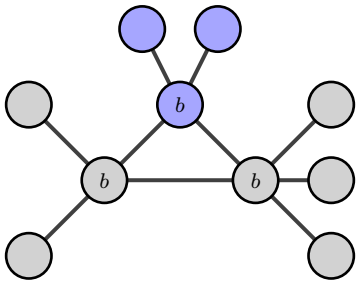


Figure 17: A Coarse Modular Assignment with $n = 2$

If $c(2b) - c(b) \leq 2$, then the gray agent will exert effort level b at both of his key tasks. Moreover, this implies that the coarse modular assignment also obtains a higher payoff than the modular assignment.

In the example above, there is only one coarse modular assignment. In larger projects, how modules are divided among agents is central to determining which assignment is best. For instance, in a project with 6 modules and 3 agents, the payoffs to assigning 2 agents to 3 modules each (where each agent exerts effort level $3b$) or 3 agents to 2 modules each (where each agent exerts effort level $2b$) depends on the values of $c(3b)$ and $c(2b)$. As can be seen, the structure of optimal assignments where the team is acutely understaffed i.e. $n < \gamma(G)$ cannot be generically derived without more assumptions made on the cost function and project structure. Nevertheless, we can recover the optimal assignment if we focus on a special case: the fixed capacity cost function as defined in Corollary 2. Before proceeding, I first formalize the definition of modularization with understaffing.

Definition 6. For key-peripheral network G, let $H(G, n)$ be an induced subgraph of key-peripheral network G with domination number equal to n. Define $H^{max}(G, n)$ to be the largest of such induced subgraphs.[17] The **modular assignment with understaffing** assigns a different agent to the key task residing in the minimum dominating set of $H^{max}(G, n)$ and its peripheral set.

The minimum dominating set of $H^{max}(G, n)$ comprises n key tasks. The peripheral sets of each of these key tasks reside within $H^{max}(G, n)$, lest $H^{max}(G, n)$ would not be a maximum induced subgraph and so the modular assignment with understaffing is well-defined and feasible. The following figure is an example of $H^{max}(G, 1)$. The modular assignment with understaffing for $n = 1$ assigns the sole agent to the blue tasks.

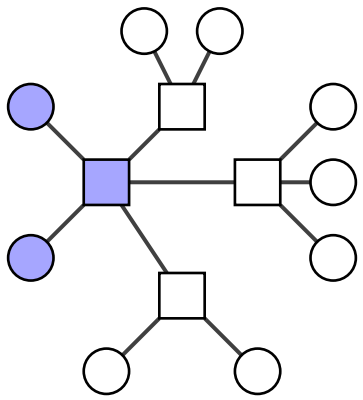


Figure 18: Modular Assignment with Understaffing

[17]Since finding the domination number of a graph is an NP-hard problem, finding all induced subgraphs of a graph with a fixed domination number greater than 1 is also an NP-hard problem.

Corollary 4. *Consider the fixed capacity cost function from Corollary 2 and $n < \gamma(G)$. Then the modular assignment with understaffing is optimal.*

Each agent exerts effort level b at his assigned key task, which makes up the minimum dominating set of $H^{max}(G, n)$. With a fixed capacity cost function, the optimal assignment must complete the most tasks within the project, which is what the modular assignment with understaffing accomplishes.

10 Concluding Remarks

Free-riding is a prevalent issue in network games where links symbolize effort spillovers and substitutability; agents will shirk at central tasks in the project. This paper offers an intuitive solution to the free-rider problem via task assignments. Assigning an agent to a module of related tasks— a key task and its peripheral tasks— ensures that he will prioritize completing the most important task in his work stream. A judicious task assignment in modularization aligns agents' incentives with the principal's objective, thereby solving the free-riding problem. Nevertheless, assigning more tasks to each agent runs into trouble with convex costs. The modular assignment takes care to not assign agents to tasks that are unrelated to each other. The modular assignment is optimal on key-peripheral networks where key tasks are sufficiently specialized. Key-peripheral projects with highly specialized key tasks are common and embrace two important principles in systems design: inheritance and loose coupling.

There are multiple directions for future research on the task assignment problem. First, in my model, I take the project structure as a primitive. Adding an initial project planning period that precedes the task assignment phase can capture that, in practice, the principal may have some control over project design. Second, agents may have different skill sets more suitable for some tasks than others. Incorporating heterogeneous productivity among agents across tasks can add another dimension to the trade-off— in addition to mitigating free-riding and convexity, the principal should try to match agents to tasks that fit their skill sets. Third, projects may contain tasks that have to be completed in sequence. The underlying task structure thus takes the form of a directed network, and the optimal task assignment will likely depend on the direction of effort spillovers and substitutability. Lastly, some tasks may take longer than others to complete. Embedding the task assignment problem in a dynamic setting can introduce time-dependent free-riding behavior where agents wait until linked tasks are completed

before starting their task. Equipping the principal with the ability to set deadlines or project milestones in tandem with task assignment power would be interesting to explore.

References

Ahuja, Ravindra K., Thomas L. Magnanti, and James B. Orlin. 1993. Network Flows: Theory, Algorithms, and Applications. Englewood Cliffs, N.J:Prentice Hall.

Allouch, Nizar. 2015. "On the Private Provision of Public Goods on Networks." Journal of Economic Theory, 157: 527–552.

Alonso, Ricardo, and Niko Matouschek. 2008. "Optimal Delegation." The Review of Economic Studies, 75(1): 259–293.

Banks, Jeffrey S., and Rangarajan K. Sundaram. 1998. "Optimal Retention in Agency Problems." Journal of Economic Theory, 82(2): 293–323.

Bramoullé, Yann, and Rachel Kranton. 2007. "Public Goods in Networks." Journal of Economic Theory, 135(1): 478–494.

Bramoullé, Yann, Rachel Kranton, and Martin D'Amours. 2014. "Strategic Interaction and Networks." American Economic Review, 104(3): 898–930.

Chen, Sheu Hua, and Hong Tau Lee. 2007. "Performance Evaluation Model for Project Managers using Managerial Practices." International Journal of Project Management, 25(6): 543–551.

Che, Yeon-Koo, and Seung-Weon Yoo. 2001. "Optimal Incentives for Teams." American Economic Review, 91(3): 525–541.

Chidamber, S.R., and C.F. Kemerer. 1994. "A Metrics Suite for Object Oriented Design." IEEE Transactions on Software Engineering, 20(6): 476–493.

Dasaratha, Krishna, Benjamin Golub, and Anant Shah. 2024. "Incentive Design With Spillovers." Working Paper.

Elliott, Matthew, and Benjamin Golub. 2019. "A Network Approach to Public Goods." Journal of Political Economy, 127(2): 730–776.

Ferguson, J. C., and S. Pruess. 1991. "Shape-preserving interpolation by parametric piecewise cubic polynomials." Computer-Aided Design, 23(7): 498–505.

Fritsch, F. N., and R. E. Carlson. 1980. "Monotone Piecewise Cubic Interpolation."

SIAM Journal on Numerical Analysis, 17(2): 238–246.

Fu, Richard, Ajay Subramanian, and Anand Venkateswaran. 2016. "Project Characteristics, Incentives, and Team Production." Management Science, 62(3): 785–801.

Galeotti, Andrea, Sanjeev Goyal, Matthew O. Jackson, Fernando Vega-Redondo, and Leeat Yariv. 2010. "Network Games." The Review of Economic Studies, 77(1): 218–244.

Garud, Raghu, Richard Langlois, and Arun Kumaraswamy, ed. 2003. Managing in the Modular Age: New Perspectives on Architectures, Networks and Organizations. Oxford:Blackwell.

Georgiadis, George. 2015. "Projects and Team Dynamics." The Review of Economic Studies, 82(1): 187–218.

Georgiadis, George, and Balazs Szentes. 2020. "Optimal Monitoring Design." Econometrica, 88(5): 2075–2107.

Georgiadis, George, Steven A. Lippman, and Christopher S. Tang. 2014. "Project Design with Limited Commitment and Teams." The RAND Journal of Economics, 45(3): 598–623.

Glover, Jonathan, and Eunhee Kim. 2021. "Optimal Team Composition: Diversity to Foster Implicit Team Incentives." Management Science, 67(9): 5800–5820.

Groves, Theodore. 1973. "Incentives in Teams." Econometrica, 41(4): 617–631.

Holmström, Bengt. 1979. "Moral Hazard and Observability." The Bell Journal of Economics, 10(1): 74–91.

Holmström, Bengt. 1982. "Moral Hazard in Teams." The Bell Journal of Economics, 13(2): 324–340.

Holmström, Bengt. 1999. "Managerial Incentive Problems: A Dynamic Perspective." The Review of Economic Studies, 66(1): 169–182.

Holmström, Bengt. 2017. "Pay for Performance and Beyond." The American Economic Review, 107(7): 1753–1777.

Hyväri, Irja. 2006. "Project Management Effectiveness in Project-Oriented Business Organizations." International Journal of Project Management, 24(3): 216–225.

Kinateder, Markus, and Luca Paolo Merlino. 2017. "Public Goods in Endogenous Networks." American Economic Journal: Microeconomics, 9(3): 187–212.

Kinateder, Markus, and Luca Paolo Merlino. 2023. "Free Riding in Networks." European Economic Review, 152: 104378.

Maslach, Christina, and Michael P. Leiter. 2016. "Understanding the Burnout Experience: Recent Research and its Implications for Psychiatry." World Psychiatry, 15(2): 103–111.

Matouschek, Niko, Michael Powell, and Bryony Reich. 2024. "Organizing Modular Production." Working Paper.

Mirrlees, James A. 1976. "The Optimal Structure of Incentives and Authority within an Organization." The Bell Journal of Economics, 7(1): 105–131.

Parnas, D. L. 1972. "On the Criteria to be used in Decomposing Systems into Modules." Communications of the ACM, 15(12): 1053–1058.

Simon, Herbert A. 1965. "The Architecture of Complexity." General Systems : Yearbook of the Society for the Advancement of General Systems Theory, 10: 63.

Stevens, W. P., G. J. Myers, and L. L. Constantine. 1999. "Structured design." IBM Systems Journal, 38(2/3): 231–256.

Subramanyam, R., and M.S. Krishnan. 2003. "Empirical analysis of CK metrics for object-oriented design complexity: implications for software defects." IEEE Transactions on Software Engineering, 29(4): 297–310.

West, Douglas Brent. 2001. Introduction to Graph Theory. Prentice Hall.

A Appendix

A.1 Proof of Claim in Proposition 1

Proof. Assume by contradiction that there exists $h \in S_i$ such that $e_h = 0$. This implies $e_i = b$ and subsequently $e_h = 0$ for all $h \in S_i$. Define $L := \{\ell \in G_i : e_\ell + \sum_{\ell' \in G_\ell \setminus \{i\}} e_{\ell'} = 0\}$. Note that $S_i \subseteq L$ by assumption, so L is non-empty. All of the tasks in L require $e_i = b$ so their respective effort thresholds are met. Consider the following ϵ-deviation effort profile: increase effort at all tasks in L by some $\epsilon > 0$ and decrease effort at task i by the same ϵ. This deviating effort profile completes the same number of tasks as before, but with a lower cost if ϵ is chosen to satisfy $|L| \cdot c(\epsilon) + c(b - \epsilon) < c(b)$.

Since c is C^1 and $c'(0) = 0$ by assumption, there exists $\epsilon_0 > 0$ such that for all $\epsilon_0' < \epsilon_0$, $\frac{c(\epsilon_0')}{\epsilon_0'} < \frac{c'(0.99b)}{|L|}$. There also exists $\epsilon_1 > 0$ such that for all $\epsilon_1' < \epsilon_1$, $\frac{c(b)-c(b-\epsilon_1')}{\epsilon_1'} > c'(0.99b)$ because c is strictly increasing and convex. Set $\epsilon = \min\{\epsilon_0, \epsilon_1\}$ so that $\frac{|L| \cdot c(\epsilon)}{\epsilon} < \frac{c(b)-c(b-\epsilon)}{\epsilon}$ which implies $|L| \cdot c(\epsilon) + c(b - \epsilon) < c(b)$ and proves the claim. □

A.2 Proof of Theorem 6

Proof. Let G be a network composed of key, peripheral and intermediary tasks.

Cost Function for Optimality: Any assignment that achieves a strictly higher payoff than the modular must implement an equilibrium with intermediate effort $e_i \in (0, b)$ at some key task i and complete all of its peripheral tasks i.e. efforts $b - e_i$ played. For this to be an equilibrium, the agent assigned to key task i, call him agent a, must be assigned to a supporting neighboring task j. This supporting task can be key or intermediary.

Case #1: Task j is key. Then its peripheral tasks must be incomplete, and the modular assignment does better in this case for some cost function with $c(b) < \min_{i \in \mathcal{K}(G)} \sigma_i$. An example is cost function $c(e) = \left(\frac{e}{\beta b}\right)^\alpha$ where β is sufficiently large.

Case #2: Task j is intermediary. As an aside, first notice the first-best key task effort in a network that contains intermediary tasks is exactly the same as that of a standard key-peripheral network i.e. $c'(e_i) = |S_i| c'(b - e_i)$. Now we focus on the class of cost functions $c(e) = \left(\frac{e}{\beta b}\right)^\alpha$ with $\alpha > 1$ and β sufficiently large to ensure $c(b) < 1$. In the first-best:

$$e_i^* = b \left(\frac{\exp\left(\frac{\ln(|S_i|)}{\alpha - 1}\right)}{1 + \exp\left(\frac{\ln(|S_i|)}{\alpha - 1}\right)} \right)$$

By L'Hôpital's rule, $\lim_{\alpha \to 1^+} e_i^* = b$, and so for α sufficiently close to 1, the difference

between the modular assignment and the first-best payoffs is negligible. Now returning to the original inquiry, if task j is intermediary, then there must be at least one key task in $k \in G_j$ different from i by definition. In equilibrium, it must be that $e_k < b$ lest $e_j + \sum_{\ell \in G_j} e_\ell > b$ and agent a will have a profitable deviation by reducing his effort at task i. The upper bound to the payoff of the alternate assignment on tasks in $\{i, j, k\} \cup S_i \cup S_k$ is to complete all tasks at a minimum cost:

$$\min_{e_i, e_j, e_k} c(e_i) + |S_i| c(b - e_i) + c(e_j) + c(e_k) + |S_k| c(b - e_k)$$

$$\text{s.t. } e_j + \sum_{\ell \in G_j} e_\ell = b \; ; e_i, e_j, e_k \geq 0$$

Denote $\hat{e}_i$ and $\hat{e}_k$ to be the solutions to this cost minimization problem. For the class of cost functions $c(e) = e^\alpha$ with $\alpha > 1$, it cannot be that both $\lim_{\alpha \to 1^+} \hat{e}_i = b$ *and* $\lim_{\alpha \to 1^+} \hat{e}_k^* = b$ since the constraint $e_j + \sum_{\ell \in G_j} e_\ell = b$ must be obeyed so that $e_i \in (0, b)$ is an equilibrium effort profile. Thus there is always a nonnegligble wedge between the first-best payoff and that of this alternate assignment, and we can conclude that the modular assignment does better in this case for cost function $c(e) = \left(\frac{e}{\beta b}\right)^\alpha$ with α sufficiently close to 1.

Now to tie both cases together, the modular assignment is optimal for cost function $c(e) = \left(\frac{e}{\beta b}\right)^\alpha$ with sufficiently large β and α sufficiently close to 1.

Cost Function for Suboptimality: Let $\mathcal{I}(G)$ be the set of all intermediary tasks of network G. Choose some intermediary task j such that $G_j \cap D_{min} \not\supset G_k \cap D_{min}$ for all $k \in \mathcal{I}(G) \cap G_j$. Assign intermediary task j and all key tasks $G_j \cap D_{min}$ to some agent, call him agent a. Assign every peripheral task of every key task in $G_j \cap D_{min}$ to a different agent. Define the set $\mathcal{J} := \{k \mid G_k \cap D_{min} \subseteq (G_j \cap D_{min})\}$ and notice that $\mathcal{J} \cap G_j = \emptyset$ by property of task j. Assign the tasks in $\mathcal{J}$ according to Algorithm 4.

Algorithm 4

```
for x ∈ {1, 2, ..., |G_j ∩ D_min|} do
    for each k ∈ J with |G_k ∩ D_min| = x do
        if not linked to a task in J already assigned then
            Assign k to an unassigned agent
        else
            Remain unassigned
        end if
    end for
end for
```

For every other task $\ell \in D_{min}$, assign ℓ and two of its peripheral tasks to a different agent. This assignment implements the following effort profile as a Nash equilibrium. Agent a exerts effort $\frac{b}{|G_j|}$ at every task in G_j and zero effort at task j. An agent assigned to a task $k \in \mathcal{J}$ with $|G_k \cap D_{min}| = x$ best responds with effort $\frac{b(|G_j|-x)}{|G_j|}$. Every other agent plays effort b at his assigned task in D_{min} and zero elsewhere. The nontrivial best response to check is that of agent a. If he shirks at any of his tasks in G_j, then he loses the task completion payoff at task j. $\mathcal{J} \cap G_j = \emptyset$ ensures that any of the positive efforts played at intermediary tasks in $\mathcal{J}$ do not contribute to the total efforts at task j. Playing effort b at task j and zero at every other task in G_j is not profitable because every task in G_j is completed regardless of his effort provision. This is because the tasks in $G_j \cap D_{min}$ neighbor at least two peripheral tasks with efforts $\frac{b(|G_j|-1)}{|G_j|}$ with $|G_j| \geq 2$, so the total neighboring efforts total to at least the completion threshold b. Thereby agent a does not have a profitable deviation to playing effort b at task j.

The payoff to this assignment replicates the modular assignment everywhere besides the tasks in $\{j\} \cup G_j \cup \mathcal{J}$. It is easy to see that the completion threshold is met at tasks in $G_j \cup \{j\}$ and the assigned tasks in $\mathcal{J}$. For the unassigned tasks in $\mathcal{J}$, their neighboring efforts total to at least $\frac{b}{|G_j|} + \frac{b(|G_j|-1)}{|G_j|} = b$, so they are also completed. Thus the alternate assignment completes every task in $\{j\} \cup G_j \cup \mathcal{J}$ with an upper bound cost of:

$$c(|G_j| \cdot \tfrac{b}{|G_j|}) + |\mathcal{J}| \cdot c(\tfrac{b(|G_j|-1)}{|G_j|})$$

The modular assignment counts a cost of $|G_j \cap D_{min}| \cdot c(b)$ on the tasks in $\{j\} \cup G_j \cup \mathcal{J}$. Thus the assignment achieves a strictly higher payoff if:

$$|\mathcal{J}| \cdot c(\tfrac{b(|G_j|-1)}{|G_j|}) < (|G_j \cap D_{min}| - 1)c(b) \iff c(\tfrac{b(|G_j|-1)}{|G_j|}) < \tfrac{|G_j \cap D_{min}|-1}{|\mathcal{J}|}c(b)$$

Now to construct the desired cost function, set $c(b)$ to some value $C \in (0, 1)$. and set $c(\frac{b(|G_j|-1)}{|G_j|}) = \frac{|G_j \cap D_{min}|-1}{2m|\mathcal{J}|}$ where m is a natural number chosen such that $\frac{|G_j \cap D_{min}|-1}{2m|\mathcal{J}|} < C \cdot (\frac{b(|G_j|-1)}{|G_j|})$ to preserve convexity. Again, we appeal to the Fritsch-Carlson method for shape-preserving piecewise cubic Hermite polynomial interpolation to generate the desired cost function.

□

A.3 Proof of Theorem 7

Proof. The proof proceeds in two parts: showing that the modular assignment implements the minimum dominating effort profile under imperfect substitutability for a sufficiently large δ and that the modular assignment is optimal when $c(b) \leq \min_{i \in \mathcal{K}(G)} \sigma_i$ for a sufficiently large δ.

Implementability: The modular assignment assigns each agent to a module; an agent assigned to module $S_i \cup \{i\}$ receives payoff of $|S_i| + 1 - c(\frac{b}{\delta})$ if he plays according to the equilibrium effort profile.

We can inspect all possible deviations the agent has by holding his number of completed tasks in the module constant, which creates $|S_i|$ separate cost minimization problems and generates $|S_i|$ incentive-compatible (IC) constraints. We consider deviations that set $e_i \neq \frac{b}{\delta}$, lest they would not be profitable. The agent's best payoff from completing m tasks within the module $S_i \cup \{i\}$ with $e_i \neq \frac{b}{\delta}$ solves:

$$\min e_i + \sum_{\ell \in S_i} e_\ell$$

$$\text{s.t. } e_h + \delta \sum_{\ell \in G_h} e_\ell \geq b \text{ for exactly } m \text{ tasks and } e_h \geq 0 \text{ for all tasks } h \in S_i \cup \{i\}$$

Since all other agents are playing according to the minimum dominating effort profile, key task i can be completed by free-riding. The solution to the problem above for $m = 1$ is setting $e_i = e_l = 0$ for all $l \in S_i$, for a payoff of 1. To complete 2 tasks, the solution is to set one of the $e_l = b$ and the rest of the efforts equal to zero, for a payoff of $2 - c(b)$. To complete 3 tasks, the solution is to set $e_l = b$ for 2 tasks for a payoff of $3 - c(2b)$. Thereby, for playing $\frac{b}{\delta}$ at key task i to be a best response, the following constraints must hold:

$$|S_i| - c(\tfrac{b}{\delta}) \geq m - c((m-1)b) \text{ for all } m \in \{1, 2, ..., |S_i|\}$$

A sufficiently high δ that is less than 1 that satisfies all constraints exists because c is continuous and $|S_i| \geq 2$. Take the maximum over all δ values that make each agent's best response $e_i = \frac{b}{\delta}$ at their key task, this is the value of δ that makes the modular assignment implement the minimum dominating effort profile under imperfect substitutability.

Optimality: The payoff from the modular assignment is:

$$\sum_{i \in \mathcal{K}(G)} (|S_i| + 1) - |\mathcal{K}(G)| \cdot c(\tfrac{b}{\delta})$$

We break the analysis into cases for an arbitrary module with key task i:

Case #1: A deviation for the principal is to implement effort $e_i = 0$. Then the greatest payoff possible is if the principal assigns all other peripheral tasks to different agents. For modularization to be optimal on module $S_i \cup \{i\}$, the following inequality must hold:

$$|S_i| + 1 - c(b/\delta) \geq |S_i| + 1 - |S_i| \cdot c(b) \implies c(\tfrac{b}{\delta}) \leq |S_i| \cdot c(b)$$

Since the inequality holds when $\delta = 1$ and c is continuous, then there must be some $\delta_1 < 1$ such that the inequality holds for all $\delta > \delta_1$.

Case #2: Another deviation is to implement effort $e_i \in (0, \frac{b}{\delta})$.

Case #2.1: If all of the peripheral task nodes remain unassigned, then obviously none of them are completed and this is sub-optimal compared to the modular assignment. I split this into subcases.

Case #2.2: Assign $m \in \{1, ..., |S_i| - 1\}$ peripheral tasks to different agents. Each agent exerts effort level $e_h = b - \delta e_i$. The maximum payoff of $m + 1 - c(e_i) - m \cdot c(e_h)$ which is bounded from above by $m + 1$, which itself is bounded above by $|S_i|$. There exists δ_2 such that $|S_i| + 1 - c(\frac{b}{\delta_2}) > |S_i|$ again by continuity of c.

Case #2.3: Assign all tasks in S_i to different agents who each exert $e_h = b - \delta e_i$. The total effort level for an agent at task i is:

$$e_i + \delta |S_i|(b - \delta e_i) + \delta \sum_{\ell \in G_i \setminus S_i} e_\ell$$

It would be problematic if the effort threshold does not exceed b, as there would be no equilibrium trade-off for implementing an intermediate effort level: Observe that for any $e_i \in (0, \frac{b}{\delta})$:

$$\delta > \frac{1}{|S_i|} \implies e_i + \delta |S_i|(b - \delta e_i) > b$$

Let δ_4 be greater than $\frac{1}{|S_i|}$. Therefore implementing an intermediate effort level and completing every task within the module is by itself not an equilibrium.

Taking the maximum δ i.e. $\delta = \max\{\delta_1, \delta_2, \delta_3\}$ for the smallest module suffices to fend off profitable deviations in the form of Case #1-#2.3 for *all* modules within the project while guaranteeing that assigning agents as prescribed in Case #2.4 is not an equilibrium. The remainder of the proof of optimality now mirrors that of Theorem 4 assuming the δ is sufficiently high as previously shown except for the following observation.

Implement a key task effort of $e_i \in (0, \frac{b}{\delta})$ and assign all tasks in S_i to different agents. Since δ is sufficiently high, the principal needs to give the agent assigned to key task i a neighboring key task j that supports it in equilibrium. Just like in Proposition 2, the sum of total efforts at key task j must be exactly b i.e. $e_j + \delta \sum_{\ell \in G_j} e_\ell = b$ for this to be an equilibrium. If the principal assigns one of the peripheral tasks ℓ of key task j to an agent, then he will best respond by playing effort $e_\ell = b - \delta e_j$. Note that there exists a sufficiently large δ such that $e_j + \delta e_i + \delta(b - \delta e_j) > b$ since the inequality holds with $\delta = 1$ and the expression is continuous in δ. Thereby if δ is large enough, then none of the peripheral tasks of a supporting key task can be completed in equilibrium. Then all of the conclusions in Theorem 4 follow in the same fashion and modularization is optimal if $c(\frac{b}{\delta}) \leq \min_{i \in \mathcal{K}(G)} \sigma_i$. $\square$