Algorithms for Recursive Delegation

Juan Afanador [a,*], Murilo S. Baptista [b] and Nir Oren [a]
[a] *Department of Computing Science, University of Aberdeen, Scotland, UK*
E-mails: r01jca16@abdn.ac.uk, n.oren@abdn.ac.uk
[b] *Department of Physics, University of Aberdeen, Scotland, UK*
E-mail: murilo.baptista@abdn.ac.uk

Abstract. Task delegation is essential to many applications, ranging from outsourcing of work to the design of routing protocols. Much research in computational trust has been devoted to the generation of policies to determine which agent should the task be delegated to, given the agent's past behaviour. Such work, however, does not consider the possibility of the agent delegating the task onwards, inducing a chain of delegation events before the task is finally executed. In this paper, we consider the process of delegation chain formation, introducing a new algorithm based on quitting games to cater for recursive scenarios. We evaluate our proposal under various network topologies, consistently demonstrating its superiority with respect to recursive adaptations of existing multi-armed bandit algorithms.

Keywords: Trust, Delegation

1. Introduction

In pursuing a goal, agents may delegate some tasks to others, as the *delegator* may believe that the *delegatee* is more capable of successfully executing the task. While existing work typically assumes that the delegatee would perform the task, this may not always be the case. Instead, the delegatee may act as an intermediary, delegating the task onwards to others who are better suited to executing it. This type of *recursive delegation* has — to our knowledge — rarely been considered in the multi-agent systems community, though it captures a common situation where, for example, projects are repeatedly contracted and subcontracted within or between organisations.

Existing approaches to trust are poorly suited to operating in domains where recursive delegation is possible, as 1) agents within such a system must decide whether to execute a task or delegate it further (as opposed to simply executing a task delegated to them); 2) delegators must learn about the competencies of their neighbours with respect to both delegation and execution (as opposed to learning about competency of execution only), while accounting for the learning process that their neighbours simultaneously undertake; and 3) the topology of the network of possible interactions may change (while most existing approaches focus on delegating only to the most competent neighbouring executor). In these scenarios, the likelihood of a task being successfully executed can change rapidly, meaning that delegation decisions in such domains are difficult to make.

In this work, we propose an algorithm that explicitly considers recursive delegation by building on quitting games [1]. We then compare the performance of this algorithm to several existing techniques, empirically demonstrating its improved behaviour. In this work, we do not consider reputation, but only direct trust observations. Therefore, evaluating our algorithm against many existing trust and reputation-based approaches [2] is inappropriate. Instead, our evaluation employs partner-selection procedures modelled after multi-armed bandits; namely ϵ-greedy [3], UCB1 [4], Thompson Sampling [5], and a numerical implementation of the Gittins Index [6].

The remainder of the paper is structured as follows. We describe our benchmarks in Section 2. In Section 3, we present our new quitting-game based algorithm alongside the general framework for adapting the bandit approaches to the delegation domain —via Brezzi and Lai's numerical approximation to the Gittins Index. We then evaluate the different approaches in Section 4. We discuss our results and situate them within existing work in Section 5, before concluding in Section 6.

*Corresponding author. E-mail: r01jca16@abdn.ac.uk.

2. Background

The problem of task delegation among partners with unknown competencies can be viewed as an exploitation/exploration problem. A delegator must decide whether to delegate a task to a known partner (exploitation), or risk delegating to an unknown partner in the hope of a better outcome (exploration). A common framework for modelling precisely this class of problem is offered by multi-armed bandit (MAB) models, which we describe next.

2.1. Multi-Armed Bandits

A multi-armed bandit problem represents a situation where a single agent must repeatedly select from among several courses of action, and then obtains a reward. The repeated execution of an action can affect the rewards it yields, an effect modelled by a random variable which — whenever the action is performed — can cause a change to occur in the reward state underpinning the action. In the MAB model, each potential action is referred to as an *arm*, while choosing the action is referred to as *pulling an arm*.

Definition 1 (Multi-Armed Bandits — Arms). *An arm A is a tuple $\langle X, r, h, f\rangle$ where X is an ordered list of possible states of the arm, and r is a probability distribution over possible rewards, parameterised by X.*

The history *of the arm, h, is a set of pairs (x_h, l_h) where $l_h \in \mathbb{Z}$ is the number of times the arm was pulled while in the state indexed by x_h. The* current state *of the arm is the state associated with the largest index of the arm's history with a non-zero l_h.*

Denoting the set of all possible histories as H, and the index of the current state of the arm as x, f is a probability distribution over the states $[x_h, x_{h+1}]$ parameterised over H.

Definition 2 (Multi-Armed Bandits — Pulling an arm). *Pulling an arm with current state x_i and history $h = [(x_1, l_1), \ldots, (x_i, l_i), (x_{i+1}, 0), \ldots (x_n, 0)]$ will update the arm's history to h' as follows:*

$$h' = \begin{cases} [(x_1, l_1), \ldots, (x_i, l_i + 1), (x_{i+1}, 0), \ldots (x_n, 0)] & \text{if } f(h) = x \\ [(x_1, l_1), \ldots, (x_i, l_i), (x_{i+1}, 1), \ldots (x_n, 0)] & \text{otherwise} \end{cases}$$

A multi-armed bandit is a set $\mathcal{A}$ of arms. The number of times each arm was pulled starts at zero. Pulling an arm updates the arm as described above, and — given the arm is in state x — yields a reward R with likelihood $r(x, R)$.

A policy specifies which arm should be pulled next. More formally, a policy is a function $\mathcal{S} : [a_1, \ldots, a_n] \times [r_1, \ldots, r_n] \rightarrow \mathcal{A}$, which takes in a sequence of arm-pulls and the rewards obtained so far, and returns the arm to pull. The main problem considered in the MAB literature involves identifying a policy which is in some sense optimal, e.g., which maximises rewards, or minimises regret. It has been long established that if the states of a MAB and the probability distribution of its rewards are known, the Gittins Index can be used to identify the optimal arm to pull [6].

Formally, the Gittins Index for arm i in state x_i, with a discount factor for future rewards of β, is defined as follows:

$$G(x_i) = sup_{\sigma>0} \frac{E[\sum_{t=0}^{\sigma-1} \beta^t r(x_i) | (x_0, 0)]}{E[\sum_{t=0}^{\sigma-1} \beta^t | (x_0, 0)]}$$

The Gittins Index computes the expected reward of pulling arm x_i against the cost of not pulling it, and thus identifies the arm with the highest expected reward as the one that should be pulled. Calculating the Gittins Index is computationally expensive [6], and various numerical approximations have therefore been proposed in the literature [7, 8].

More importantly, in practice, the probability distribution of the rewards and the states of each arm may not be known. In this case, the Gittins Index may be used as a heuristic based on beliefs about rewards and arm states, which means that different ways of calculating these beliefs will result in different procedures with very distinct properties. We now describe several such heuristics addressing the MAB problem, namely UCB1 [4], ϵ-greedy [3], and Thompson Sampling [5]. We will compare the performance of our approach to these heuristics in Section 4.

2.2. MAB Heuristics

We begin this section by briefly describing several well-known MAB heuristics. In Section 3 we detail how these heuristics must be modified to deal with recursive delegation.

UCB1. Rather than simply maximising rewards, *upper confidence bound* (UCB) algorithms, exemplified by UCB1 [4] which we consider in this paper, attempt to minimise decision-theoretic *regret* — the difference between the expected reward obtained had the optimal arm been pulled, and the expected reward of some other arm-pulling policy, where the optimal reward is

the result of the operation of a MAB under complete and perfect information. In turn, the rewards obtained by following the (theoretically) optimal strategy is referred to as the *oracle's prediction*.

UCB1 is simple to implement and works well in practice, achieving logarithmic growth of regret in the number of arm-pulls. For an arm j, UCB1 tracks the average reward obtained from that arm (μ_j), and the number of times the arm has been pulled (n_j), as well as the total number of times an arm-pull has occurred (n). It then picks arm j, so as to maximise an upper bound on the mean expected reward given by the following equation [4]:

$$\mu_j + \sqrt{\frac{2\ln n}{n_j}}$$

This choice guarantees that the probability of deviating from the population mean decays exponentially through time, in accordance with the Chernoff-Hoeffding inequality [9]. Once the arm has been pulled, μ_j, n_j and n are updated to identify the next choice.

Thompson Sampling. This is another simple approach to selecting an arm, which does so by sampling an expected reward based on the arm's history, before selecting the arm whose sample reward is maximal. To perform such sampling, a probability distribution over the arms is required [10]. In this work we consider binary rewards, and we therefore perform our sampling using Beta distributions, whose parameters record the number of times the arm returned a reward, and the number of times it did not. Thompson samples each arm using this probability distribution, and then selects the arm that delivers the highest expected sampled reward.

ϵ-Greedy. This heuristic selects the arm that yields the highest expected reward with likelihood $1-\epsilon$ [3], and picks a random arm otherwise. It is important to note that this heuristic differs from Thompson Sampling in that no sampling over the arms takes place, meaning that the best arm (with regards to their expected rewards) is always selected, unless a random arm is chosen (with likelihood ϵ).

All of the heuristics described above seek to balance exploitation — selecting the arm most likely to give a high reward — with exploration — the contemporaneous learning of an arm's likelihood to deliver the highest expected reward. If the distribution governing the reward an arm provides is stationary, then these heuristics work well and give well-understood convergence guarantees. However, in the case of recursive delegation, agents at each level learn simultaneously, implying the violation of the stationarity assumption — at least until the learning stage ends. It is for this reason that these heuristics function poorly when applied to recursive delegation. In order to apply them to the domain of recursive delegation we must — in some sense — unify the delegation/execution decision. We describe one simple approach to doing so next.

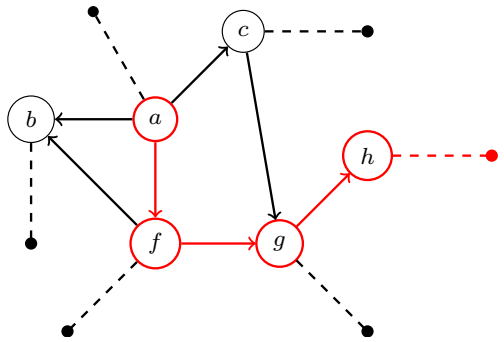


Fig. 1. A network of agents illustrating possible delegation links. Dotted lines indicate links to dummy agents which, when delegated to, execute the task. Red indicates a single *delegation chain* from a to h.

2.3. Applying MAB Heuristics to Recursive Delegation

Agents able to delegate to others must make two choices when tasked with an action, namely whether to execute the action themselves, or delegate it onwards (and in the latter case, must also decide who to delegate to). Each agent has a list of delegatees to which they can delegate a task. By equating the delegatee agents to neighbours of the delegator, we obtain a directed graph over which a path represents a sequence of delegations.

We unify the execution/delegation decision by associating a *dummy agent* with every (nominal) agent in the system. Such an agent acts as the *de facto* delegatee, but has no delegatees of its own. This means that any task reaching the dummy agent must be executed, and we treat this execution as having been performed by the associated nominal agent (which delegated the task to the dummy agent). Figure 1 illustrates a sample delegation network consisting of 6 agents $\{a, b, c, f, g, h\}$, together with their corresponding dummy counterparts as solid nodes. Given this representation, one possible sequence of delegations, or *delegation chain*, is $\langle a, f, g, h\rangle$ in red.

To use the heuristics described above in a recursive context, agents make a *local* delegation decision, choosing whom to pass the task to based only on

their neighbours' potential to become delegatees. If a dummy agent receives the task, then it is executed, and feedback on success or failure is provided to every agent along the delegation chain. From thereon, each agent updates the statistics relevant to its delegation decision with respect to its neighbours, and the process repeats. Clearly, this approach prevents an agent from considering how others within the chain make decisions, and we claim that this affects the effectiveness of MAB heuristics in scenarios of recursive delegation.

2.4. Quitting Games

In the next section, we will formulate an alternative approach to delegation which explicitly considers the actions available to agents through a game-theoretic mechanism based on quitting games [1]. Quitting games are multi-player stochastic games where players are faced with two choices, namely to *continue* (c) or to *quit* (q). The game ends and the players obtain rewards in two situations: whenever a *quit* action occurs, or the game reaches some terminal time. If the game does not end after the players have selected their moves, i.e., they both simultaneously select *continue* actions, another iteration occurs where players act again, repeating this process until termination. Figure 2 illustrates a generic two-player quitting game between agents a and b.

The first entry in each terminal node appearing in Figure 2 corresponds to the reward accrued to a, the other denotes b's reward. Whenever (c_a, q_b) is played, a receives r_{c_a} and b obtains r_{q_b}, whereas (c_a, c_b) leads to yet unrealised rewards denoted by "↻". Agents a and b plan future moves by formulating *strategies* based on the anticipation of potential $\varepsilon-$ *equilibria*.

Definition 3 (**Quitting Game — Strategies**). *At every iteration t within a time horizon T, each player i is provided with a set of actions $A_i = \{c_i, q_i\}$. A* strategy *is a probability measure $x_t^i : T \to [0,1]$ denoting the likelihood of playing c_i at iteration t.*

Definition 4 (**Quitting Game — ε-equilibrium**). *A profile or vector of strategies $\boldsymbol{x}_t$, produces a stream of rewards r_{S_t}, contributed by those players S_t who have chosen not to quit the game, giving rise to an expected reward $v_t^i(\boldsymbol{x}_t) := \boldsymbol{E}_{\boldsymbol{x}}[r_{S_t}]$. A* solution concept *states the criteria for playing a particular profile. ε-equilibrium is the solution concept employed when solving a quitting game. A profile $\boldsymbol{x}_t$ is an ε-equilibrium if the expected reward it yields plus an overhead $\varepsilon_t > 0$, is at least that of any other strategy y_t^i for every player i:*

$$v_t^i(\boldsymbol{x}_t) \geqslant v_t^i(\boldsymbol{x}_t^{-i}, y_t^i) - \varepsilon_t.$$

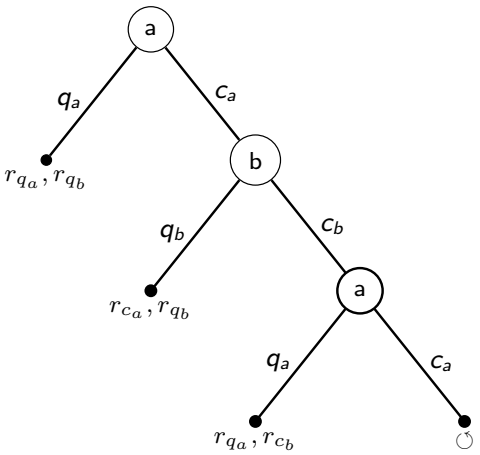


Fig. 2. Quitting Game in Extensive Form

Note that if $\varepsilon_t = 0$, the above expression produces a Nash equilibrium. ε-equilibria can be further qualified as *cyclic* if there exists a point in time $\tau \in T$ when $x_t^i = x_{t+\tau}^i$, or *stationary* if $x_t^i = x_0^i$ for each $t \in T$. For instance, given $r_{q_a} > 0$, $r_{c_a} < r_{q_b}$, $r_{q_a} < r_{c_a}$, and $r_{c_b} \geqslant r_{q_b}$, the stationary profile $(\mathbf{x}^a, c_b)$, $x_t^a \ll 1$ is an ε-equilibrium of the game in Figure 2. More generally, every quitting game where players prefer unilateral termination to indefinite continuation, has a cyclic subgame perfect ε-equilibrium [1], while every two and three-player quitting game has a stationary ε-equilibrium [11].

In the next section, we describe how such quitting games and their associated solution concepts can be used to underpin an algorithm for recursive delegation.

3. Approach

As indicated in Section 2.1, the problem of task delegation can be seen as an exploitation/exploration problem in the spirit of MABs, where delegators choose between delegating the task to competent partners (exploitation) or delegating the task to unknown partners (exploration). In this section we present an algorithm for recursive delegation based on quitting games. After this, we describe how the Gittins Index can be adapted to the recursive delegation domain, allowing us to perform an evaluation of the differing approaches in Section 4.

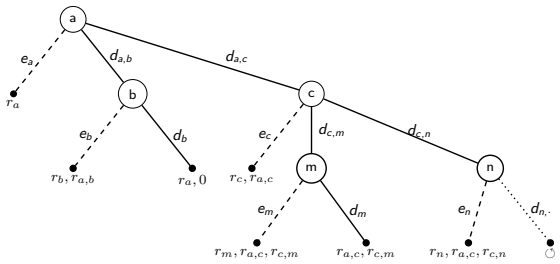


Fig. 3. Delegation Game in Extensive Form

3.1. Delegation as a Quitting Game

Quitting games are readily adaptable to recursive delegation, as they represent the occurrence of self-embedded instances of strategic interaction, resembling the replication of delegation requests along a delegation chain. That is, if a delegator (a) and a potential delegatee (b) were to play a quitting game to determine whether to delegate a task or not, the profile (c_a, c_b) would take them both to a new iteration of the same delegation request. Unlike a standard quitting game, however, a delegation process requires distinct strategic scenarios, where, e.g., b becomes a delegator facing a new delegatee. For this reason we have adjusted quitting games to capture this type of interaction, referring to such a situation as a *delegation game*.

The players of a delegation game have a *delegate* (d) action and an *execute* (e) action, and their rewards depend on future *delegate* actions. Every pair of agents populating each instance of the game consists of one former delegatee acting as delegator, and one new agent serving as potential delegatee. Delegation games can only be prolonged by joint actions (d_i, d_j) for every delegator i and delegatee j — provided there are available delegatees — and are brought to an end whenever an *execute* action occurs, or the terminal time T giving a time horizon to delegation, is reached. Future actions are formulated in terms of *strategies* and the pursuit of $\varepsilon-$ *equilibria*.

Definition 5 (Delegation Game)**.** *A delegation game is a tuple $\langle N, (A_i, u^i, r^i)_{i\in N}\rangle$. All $n \equiv |N|$ agents, or players, pair up with each other in accordance with a particular topology of interaction. A player generating a delegation request will be referred to as the* delegator*, while a player at the receiving end of the delegation request will be termed a* delegatee*. Potential delegatees within the reach of a delegator are said to be the latter's* neighbours*.*

Every iteration of the game comprises several instances of strategic interaction. There are as many instances in a single iteration as there are available delegatees. At every iteration t within a time horizon T, each player i is provided with a set of actions $A_i = \{d_i, e_i\}$.

Definition 6 (Delegation Game — Strategies)**.** *A* strategy *is a probability measure $x_t^i : \mathbb{R} \to [0,1]$ indicating the likelihood of playing d_i at iteration t. Vectors of strategies $\boldsymbol{x}_t$ are termed* profiles*.*

Definition 7 (Delegation Game — Expected Rewards)**.** *The reward obtained from a delegation by player i at iteration t to a set of delegatees D_t is represented by the random variable $r^i_{D_t}$.*

$u_t^i : \boldsymbol{x}_{t-1} \times \mathbb{R} \to \Delta(A_i)$ is a measurable set-valued function that updates each player's strategies once an action e_j occurs or a terminal node is reached based on the rewards obtained. Profiles induce a probability distribution which permits the computation of the expected rewards $v_t^i(\boldsymbol{x}_t) = \boldsymbol{E}_{\boldsymbol{x}}[r_{D_t}]$.

Figure 3 depicts one iteration of a (deterministic) *delegation game*. Agents a, b, c, m and n are arranged in a tree-like structure, where b and c are a's *neighbours*, m and n are c's *neighbours*. b and m have no *neighbours*, and n is linked to another unspecified tree which allows delegation to continue. a has to decide between choosing a delegatee from $\{b, c\}$ or executing the task itself i.e., it has to decide whether to play $d_{a,b}$, $d_{a,c}$ or e_a. Also note, first, that dummy agents appear here as the solid unlabeled nodes where *execute* actions terminate; and second, that the rewarding scheme is treated as exogenous to delegation.

Within the game encoded in Figure 3, a can play e_a and perform the task itself. It can also delegate the task to b, in which case b might accept the task by playing e_b, or not by playing d_b, thus returning the task to a and forcing the occurrence of e_a. In each case, a and b receive $(r_a, 0), (r_{a,b}, r_b)$ and $(r_a, 0)$, respectively. Alternatively, a could delegate to c. If n decides to play e_n, it receives r_n, while c and a obtain $r_{c,n}$ and $r_{a,c}$. The rewards of any agent in the delegation chain emanating from n's *neighbour*, will not be realised until some agent plays an *execute* action, the delegation process reaches a terminal node like b, or the time horizon T is exhausted. Finally, observe that m, being a terminal node like b, can either accept and execute the task by playing e_m, or reject it by invoking d_m as no further delegation can be effected.

When rewards are subject to stochastic processes, the selection of an action has to be expressed in terms

of strategic profiles ($\mathbf{x}_t$), as in Definition 6. The probability distribution that these profiles induce is then used to calculate the expected rewards (v_t^i). By maximising expected rewards in the manner of an ε-equilibrium, *delegators* and *delegatees* select a particular strategy, which once played causes the respective information states to update (u_t^i). This process is formalised in Algorithm 1.

The input to Algorithm 1 is the set of neighbours $ad_i \subset V$ to every agent $a_i \in V$, along with the respective individual rewards obtained from the interactions. These rewards are values of the random variable attached to the probabilities of successful execution which describe an agent's capabilities in Algorithm 2 and, thus, guarantee a common (stochastic) ground for comparison. The resulting initial state allows the computation of individual mixed strategies i.e., the probabilities of delegating, whenever pairs of agents and neighbours engage in a delegation request (line 4). Thus the strategy $x_{i,j}$ would designate the probability of successful delegation that agent a_i imputes to agent a_j, giving rise to the profile $\{x_{i,j}\}_{j\in ad_i} \equiv x_i \equiv [\mathbf{x}_i]_i$ in line 5.

The notation from Figure 3 is preserved except for $r_{\cdot,1}$ and $r_{\cdot,0}$, denoting the rewards of executing the task given a delegatee's willingness to further delegate or not. As long as there are neighbours who have not received such a request, despite holding a positive probability of delegating, the selection of the one with the highest expected pay-off will take place (lines 6 and 7), seeking a Nash equilibrium. A random *state of nature*, $0 < 1-\delta < 1$ gives a stochastic choice as to whether or not the strategy, denoting the probability of playing d, would be realised (line 8). If capable of executing the task, as given by a favourable state of nature i.e., $x_{i,j} > 1-\delta$, the chosen agent will have to weigh up the possibility of passing the task down the delegation chain or attempting its completion, thereby triggering a learning process (lines 9-13).

3.2. Delegation as Nested MABs

We now present an adaptation of MABs to recursive delegation, where each agent makes a local decision regarding how to delegate based on an approximation of the Gittins Index. This heuristic, described in Algorithm 2, exemplifies the general structure of the other bandit algorithms introduced in Section 2.1 later used as benchmarks in Section 4.

Algorithm 2 is initialised in the same manner as Algorithm 1. It implements the Gittins Index through

Algorithm 1 Delegation Game (DIG)

Input: $P_i := \langle a_i, ad_i\rangle$: Tuple of agents and their neighbours, r: Array of sampled rewards per tuple.
Output: S_i: Sequence of agents receiving a delegation request originated in agent a_i, x_i: Agent a_i's array of mixed strategies.

1: **function** DIG(P_i, r_i)
2: $S_i \leftarrow \emptyset, x_i \leftarrow \emptyset$
3: **for** $a_j \in ad_i$ **do**
4: $x_{i,j} = \frac{r_{i,1}-r_{i,0}}{r_{i,j}-r_j}$
5: $x_i \leftarrow x_i \cup \{x_{i,j}\}$
6: **while** $\exists j[(x_{i,j} \neq 0 \wedge ad_j \neq \emptyset)]$ **do**
7: $m \leftarrow argmax_{j\in ad_i}(r_{i,j})$
8: **if** $(1-\delta < x_{i,m})$ **then**
9: **if** $a_m \in S_i$ **then**
10: Update $r_{i,m}, x_{i,m}$
11: **else**
12: $S_i \leftarrow S_i \cup \{a_m\}$
13: **return** LEARN($P_m, r_m; x_i$)
14: **else**
15: a_i executes the task
16: **return** (S_i, x_i)

1: **function** LEARN($P_j, r_j; x_k$)
2: **if** $r_{j,0} \leqslant r_{j,1}$ **then**
3: a_j executes the task
4: Update $x_{k,j}, r_{k,j}$
5: $x_k \leftarrow \{x_{k,l}\}_{l\in ad_k}$
6: **return** (S_k, x_k)
7: **else**
8: **return** DIG(P_j, r_j)

a beta reputation mechanism captured in lines 15-18, which feeds the numerical approximation to the index as specified in lines 7-9. The reputation mechanism is a counter of successful delegation events, acting as a wrapper of the index over recursive calls. This mechanism enables delegators to incorporate information on the chosen delegatees' capabilities to execute the task, as given by their corresponding probability of successful execution $s_{(\cdot)}$ (line 15). By comparing $s_{(\cdot)}$ against the state of nature $1-\delta$, delegators induce a series of binary outcomes resembling line 8 of Algorithm 1.

The main procedure in Algorithm 2 (line 8) is Brezzi and Lai's proposal for computing an optimal MAB policy [7]. More specifically, given a large number of trials, and a time-discounting rate $c \in [0.8, 1]$, sampled from a uniform distribution between 0.8 and 1 before entering Algorithm 2, the following closed-form func-

tion approximates the Gittins Index:

$$G(T) \approx \mu + \sqrt{\frac{\mu(1-\mu)}{T+1}}\psi\left(\frac{1}{(T+1)^c}\right)$$

Here, μ is the mean of the compound distribution of the random variable indicating a successful delegation, and $\psi(t)$ identifies the set of iterations for which it is sub-optimal to continue the exploration of potential delegatees. This latter function is defined as follows:

$$\psi(t) = \begin{cases} \sqrt{t/2} & t \leqslant 0.2 \\ 0.49 - (0.11t)^{-1/2} & t \in (0.2, 1] \\ 0.63 - (0.26t)^{-1/2} & t \in (1, 5] \\ 0.77 - (0.58t)^{-1/2} & t \in (5, 15] \\ \sqrt{2\log(t) - \log\log(t) - \log(16\pi)} & otherwise \end{cases}$$

$\psi : \mathbb{R} \to \mathbb{R}_0^+$ is a non-negative function whose behaviour through time approximates $b(s) = (2^{1/2} + o(1))s$ as $s \to 0$ where s is the sample variance. $b(\cdot)$ is a function that bounds the region where the expected value of the aggregated sampled rewards attains the supremum. It is in this sense that $\psi(\cdot)$ is said to provide an estimate of the optimal stopping boundary [7].

3.3. The Computational Complexity of DIG and DID

As stated in Section 3.1, Algorithm 1 is driven by the stochastic process dictating the successful completion of the task. Upon receiving inputs in the form of sampled expected rewards, the algorithm generates a probability associated with the random variable describing the corresponding strategy. Strategies are then selected by performing the calculations in lines 4–5.

The loop containing these, runs in polynomial time in the number of neighbours $|ad_j|$ for every $j \in N$, but the time complexity of neighbour sampling is $O(nlog(n/\delta))$.

Lemma 1. *Under ε-equilibria, Algorithm 1 displays a neighbour sample complexity of $O(nlog(\frac{n}{\delta}))$, given a state of nature $1-\delta$ for $0 < \delta \ll 1$.*

Proof. Let us denote the mean optimal reward by r^*, i.e., the oracle's prediction, and let $v \equiv s^{-1}\sum_{t\in s} v^t(\mathbf{x}_t) > r^* - \varepsilon$ be the mean reward obtained under an ε-equilibrium. Then we are interested in finding an upper bound on the probability of v differing from the expected value of r^*, i.e., from $v^* = \mathbf{E}_\mathbf{x}[r_D^*]$.

$$\begin{aligned} \mathbf{P}(v > v^*) &\leqslant \mathbf{P}(v > \mathbf{E}_\mathbf{x}(r_D) + \varepsilon/2 \vee v^* < r^* - \varepsilon/2) \\ &\leqslant \mathbf{P}(v > \mathbf{E}_\mathbf{x}(r_D) + \varepsilon/2) + \mathbf{P}(v^* < r^* - \varepsilon/2) \\ &\leqslant 2exp(-s\varepsilon^2/2) \end{aligned}$$

Algorithm 2 Dynamically Indexed Delegation (DID)

Input: $P_i := \langle a_i, ad_i\rangle$: Tuples of agents and their neighbours, s: Array of probabilities of successful execution per tuple, c: Array of time-discounting parameter per tuple.

Output: S_i: Sequence of agents receiving a delegation request originated in agent a_i, ν_i: Agent a_i's array of probabilities of successful delegation.

1: **function** DID($P_i, s_i; c_i$)
2: $S_i \leftarrow \emptyset$, $\nu_i \leftarrow \emptyset$
3: **for** $a_j \in ad_i$ **do**
4: $\alpha_j \leftarrow 0, \beta_j \leftarrow 0$
5: $\mu_j \leftarrow 0$
6: **for** $a_j \in ad_i$ **do**
7: $\mu_j \leftarrow \frac{1}{(1+\beta_j/\alpha_j))}$
8: $G_{i,j} \leftarrow \mu_j + (\frac{\mu_j(1-\mu_j)}{\alpha_j+\beta_j+1})^{1/2}\psi(1/(\alpha_j+\beta_j+1)log(c_i^{-1}))$
9: $m \leftarrow argmax_{k\in ad_i}(G_{i,k})$
10: **if** $a_m \neq a_i$ **then**
11: $S_i \leftarrow S_i \cup \{a_m\}$, $\nu_i \leftarrow \nu_i \cup \{\mu_m\}$
12: **return** $DID(P_m, s_m; c_m)$
13: **else**
14: Self-execute
15: **if** $s_m > 1-\delta$ **then**
16: $\alpha_m \leftarrow \alpha_m + 1$
17: **else**
18: $\beta_m \leftarrow \beta_m + 1$
19: **return** (S_i, ν_i)

The first two lines follow by construction, the third is a direct application of the Hoeffding inequality. Choosing a sample size $s \equiv 2/\varepsilon^2 ln(2n/\delta)$, the probability of the sampled expected reward to deviate from the oracle's is bounded by the ratio between the probability designating the current state of nature and the total number of agents i.e., $\mathbf{P}(v > v^*) \leqslant \frac{\delta}{n}$. Given the loop in lines 6-16 of Algorithm 1 the total time complexity of its sampling procedure is of the order of $O(nlog(\frac{n}{\delta}))$. □

Algorithm 2 is subject to the same stochastic process governing the successful completion of the task. It directly inputs the counters of successful instances into a closed-form representation of the Gittins Index, from which a ranking is obtained and delegatees are chosen. Like DIG, this process also implies the computation of a maximal value running in polynomial time in the number of neighbours $|ad_j|$ for each agent $a_j \in V$.

The time complexity of neighbour sampling is bounded above by the ratio between the deviations from the oracle's prediction and the sample variance:

Lemma 2. *Algorithm 2 displays a neighbour sample complexity of* $O(nlog(log_{p/q}(\frac{n}{\delta})))$ *given* $p \equiv (\alpha\eta + \sigma^2)/(\eta^2+\sigma^2)$ *and* $q \equiv \sigma^2/(\eta^2+\sigma^2)$. $\eta > 0$ *is a measure of convergence for deviations of the Gittins Index as appears in line 8 of Algorithm 2,* $\sigma^2 > 0$ *is the sample variance, and* $1-\delta$ *for* $0 < \delta \ll 1$ *describes the state of nature .*

Proof. By construction $|G_t - G_{t-1}| \leqslant \eta$ for every $t \in \{1,\dots,T\}$ and the sequence $\{G_t, \mathcal{F}_k\}_{k=1}^{T}$ can be considered a Doob Martingale [12]. In this new notation the indexing of agents used in line 8 of Algorithm 2 is replaced with a time index, as our attention is now centered on the likelihood of approaching the oracle's prediction through time. Likewise, $\{\mathcal{F}_k\}_{k=1}^{T}$ is a filtration, or sequence of sub-σ-algebras, defined over the values the Gittins Index takes on.

Given these observations, motivated by the violation of the stationarity assumption, we are interested in the values of the moment-generating function of G_t, conditional on past events. In particular, its expected value and variance $\mathbf{E}[(G_t - G_{t-1})^2|\mathcal{F}_{t-1}] \leqslant \sigma^2$.

Our objective consists in finding an upper bound on the probability of the expected value of the index $G \equiv \mathbf{E}[G_\tau|\mathcal{F}_{\tau-1}]$ deviating from the oracle's prediction $G^* \equiv \mathbf{E}[G^*_\tau|\mathcal{F}_T]$ —as measured in fractions $0 < \alpha \ll 1$ of sample size units s—, for every $\tau \in \{1,\dots,T\}$:

$$\begin{aligned}\mathbf{P}(|G-G^*| \geqslant \alpha s) &\leqslant exp(-\alpha sh)\mathbf{E}[exp(h\sum_{t=1}^{T}|G_\tau - G^*_\tau|)], \quad \forall h \geqslant 0\\ &\leqslant \left(\frac{\sigma^2 exp((\eta-\alpha)h) + exp(-(\sigma^2+\alpha\eta)h/\eta^2)}{(\eta^2+\sigma^2)/\sigma^2}\right)^s\\ &\leqslant exp\left(-sD\left(\frac{\eta\alpha+\sigma^2}{\eta^2+\sigma^2}, \frac{(\eta+\sigma^2)\sigma^2)}{(\eta^2+\sigma^2)^2}\right)\right)\end{aligned}$$

The first line is a direct application of the Chernoff bound to Doob Martingales. The next one follows from a refinement of Bennet's inequality [13], leading in the limit ($h \to \infty$), i.e., when the tightest bound is sought, to the last line. $D(\cdot)$ is the Kullback-Leiber distance between the distributions of its two main arguments [14]. With $p \equiv (\alpha\eta+\sigma^2)/(\eta^2+\sigma^2)$ and $q \equiv \sigma^2/(\eta^2+\sigma^2)$, the rationale applied in Lemma 1 indicates that a sample size $s \propto p^{-1}\frac{ln(2n/\delta)}{ln(p/q)} = p^{-1}log_{p/q}(n/\delta)$ ensures $\mathbf{P}(|G-G^*| \geqslant \alpha s) \leqslant \frac{\delta}{n}$. The rest of the statement in Lemma 2 is similarly obtained. □

The shared state of nature summarised in the probability δ, introduces a common ground for comparing the two algorithms. DIG displaying higher complexity than DID would require the following to hold.

$$\begin{aligned}ln(n/\delta) > log_{p/q}(n/\delta) &= \frac{ln(n/\delta)}{ln(p/q)}\\ &\Rightarrow p/q > e\\ &\Leftrightarrow \alpha\eta > \sigma^2;\end{aligned}$$

This is contradictory for small rates of convergence of the Gittins Index $\eta \ll 1$. In consequence, our adversarial approach offers theoretical guarantees that, under non-stationarity, potential delegatees can be sampled using less computational resources compared to the best performing benchmark i.e., the numerical approximation to the Gittins Index. We leave further investigation of the effects of different values for these parameters on the behaviour of the algorithms, for future work.

4. Evaluation

Having presented our MAB and quitting-game based heuristics, we now turn to evaluating their effectiveness. We begin this section by detailing our experimental setup, following which we describe the experiments and their results.

4.1. Experimental Setup

Our evaluation consisted of running the various heuristics over 1000 trials, i.e., the time horizon was set to $T = 1000$, employing 5 different network configurations assuming 100 different initial states, i.e., 100 different parameter values for the distributions specified throughout this section. These networks were mounted on a graph $G = (V, E)$ whose vertices correspond to a set V of agents organised in m groups $\{K_0,\dots,K_{m-1}\}$, resulting in graphs of $n \equiv |V| = |\times_{i=0}^{m-1} K_i|$ nodes; a number which, as will be explained shortly, was set to $n = 156$.

The groups are also termed *levels*, across which an agent a interacts with its neighbours subject to the criteria dictating the formation of the set of edges E, the topology of the graph and ultimately the structure of the network. As a general convention, the agent who makes the first delegation request is termed the *root*. We considered the following network topologies:

Directed Trees (DT): Agents are arranged in a parent-child relation of precedence, spanning over 4 levels of size 5 . A directed tree is a tuple $\langle G, \preceq\rangle$, where $m = 4$, $|K_l| = 5^l$ and $a \simeq b$ for every $a, b \in K_l$ and $l \in \{0,1,2,3\}$. $a \succ b$ iff $i < j$ for $a \in K_i$ and $b \in K_j$, in which case a and b are treated as delegator and delegatee, respectively.

Random Networks (RN): Agents are allowed to randomly form their own neighbourhoods. A random network is a tuple $\langle G, \mathbf{P} \rangle$, where $\mathbf{P} \in \Delta(A_{(\cdot)})$ is the probability distribution induced by the set of strategic profiles stipulated in Definition 6 and Algorithm 1. In the case of the MAB benchmarks $\mathbf{P} \sim Beta(\alpha, \beta)$, as stated in Section 3.2. Both probabilities indicate the likelihood of choosing a neighbour from the set of agents populating remote levels i.e., those agents not yet chosen as delegatees. The simulations start off with the root agent and 155 potential delegatees, each choosing their neighbours from sampled subsets of size 5.

Regular Lattices (RL): Agents are arranged in regular tilings delineating their corresponding neighbourhoods. A regular lattice is a tuple $\langle G, \mathcal{C} \rangle$. Each agent is assigned a label indicating its location with respect to the root, e.g. $a \rightarrow (a_0, \ldots, a_{m-1})$ such that $0 \leqslant a_i < |K_i|$ for every $0 \leqslant i < m$. $\mathcal{C}$ establishes a relation of connectivity between pairs of agents. $z \rightarrow (a_0, \ldots, a_i + 1, \ldots, a_{n-1})$ is connected to a, or $z\mathcal{C}a$, if $a_i \leqslant K_i - 1$. Likewise a is connected to $b \rightarrow (a_0, \ldots, a_i - 1, \ldots, a_{m-1})$, or $a\mathcal{C}b$, if $a_i \geqslant 0$. For the purpose of the simulations $m = 4$ and $|K_l| = 39$.

Periodic Lattice (PL): Agents are arranged over lattices with identified edges in the shape of a torus. A periodic lattice is a tuple $\langle G, \mathcal{T} \rangle$ functioning with the same mapping of coordinates. The connectivity relation $\mathcal{T}$, however, is characterised by "wraparound" edges connecting every node to $2m$ neighbours, i.e., the representative agent a is connected to nodes $z \rightarrow (a_0, \ldots, i, \ldots, a_{m-1})$ and $b \rightarrow (a_0, \ldots, j, \ldots, a_{m-1})$, where $i \equiv (a_{i-1}) mod |K_i|$ and $j \equiv (a_{i+1}) mod |K_i|$. For the purpose of the simulations $m = 4$ and $|K_l| = 39$.

Scale-Free Networks (SF): Agents interact on a predefined structure where neighbours have been added to the network with probability proportional to the in-degree of already existing agents. A scale-free network is a tuple $\langle G, D, k, \gamma \rangle$, where the degree of a node is defined as $d_a \equiv deg(a)$ for every $a \in \sigma(V)$. That is, d_a counts the number of potential *delegatees* in a's neighbourhood for some permutation $\sigma(V)$. $D = \{d_1, \ldots, d_n\}$ denotes the increasing degree sequence of the graph, satisfying a power law relationship of the form $id_i^{\gamma} = k$ for $k, \gamma > 0$ and $0 \leqslant i \leqslant n - 1$ [15]. For the purpose of the simulations $n = 156$, $\gamma \sim \mathcal{U}(0.41, 0.65)$ and $k \sim \mathcal{U}(0.21, 0.45)$ [16].

Directed Trees offer a structured environment for accommodating agents who establish a relation of precedence upon delegating. Delegators appear as parent nodes of delegatees whose neighbours cannot be directly reached by the former. In this sense, Directed Trees describe hierarchical arrangements of agents suffering from myopic planning — the product of incomplete information about the network structure.

Regular Lattices designate alternative neighbouring patterns with higher degree of connectivity. The resulting mesh-like arrangements of agents preclude the occurrence of circuits and cycles only when the root is placed along the edges of the lattice. A behaviour that cannot be prevented in Periodic Lattices, on account of their toroidal topology achieved through the identification of the opposite edges of the underlying Regular Lattice. Periodic Lattices are suitable for modelling delegation under imperfect and incomplete information — instances where future and past delegatees cannot be known and where their objectives are unclear.

When delegation is inscribed in Scale-Free Networks, agents enjoy greater connectivity accentuating their potential to either perform or delegate a task. The process of growing Scale-Free Networks while preserving the original power law distribution, induces scale-invariant interactions highlighting individual patterns of delegation. It is worth emphasising that in our context this organising principle is presupposed rather than obtained through interaction.

Spontaneous neighbouring is achieved through Random Networks. The root triggers the formation of the network by sampling subsets from V of size equal to the prespecified branching factor. The probability of delegating arising form each algorithm is also used as the probability of spanning an edge from a delegator to a delegatee. To this extent Random Networks are discovered as agents delegate — they are formed *a posteriori*.

We experimented with different parameters for each of the heuristics. For ϵ-greedy, ϵ takes on values between 0.05 and 0.1 [3]. Thompson Sampling was recovered from a Bayesian variation of the same algorithm with no exploration. The discount factor in DID ranged within $[0.8, 1]$ so as to remain consistent with the closed-form approximation to the Gittins Index [6]

The initial probabilities of delegation were sampled from an uninformative Beta distribution. For each heuristic, we measured the probability that a delega-

tion would be successful after the *n*th iteration (averaged over the 100 runs), as well as the regret value for the action. The probabilities of successful execution, employed in Algorithm 2, correspond to the density function induced by random variables describing an agent's capabilities. Once normalised to avoid divisions by zero and negative values, these random variables are used to obtain the mixed strategies in Algorithm 1. Regret is computed as the difference between the probability that a task would be successfully executed, under complete and perfect information, and the final likelihood of successful execution provided by the root's strategies or its MAB delegation criterion.

4.2. Results

Figure 4a shows the performance of the various heuristics over Directed Trees. We observe that the DIG heuristic significantly increases the chance of successful delegation when compared to other approaches. Thompson Sampling appears to outperform the remaining approaches, but takes longer to converge than other techniques.

With regards to regret we observe (Figures 4b and 4c) that DIG minimises regret by maximising the likelihood of successful delegation, and that this relationship holds for the remaining algorithms. The dispersion of regret per time unit (Table 1[1]), indicates that UCB1's and DID's deviations from the oracle's prediction cause greater accumulation of regret as the experiments go past trial 250. It is as though further instances of delegation would prevent these two procedures from keeping up with DIG. However, none of the algorithms obtain levels of regret greater than UCB1's theoretical upper regret bound (Figure 4b and Table 1).

Turning to Random Networks, Figure 5a demonstrates that DIG and DID outperform all other approaches. The rate of convergence for UCB1 significantly lags behind the other approaches. The levels of regret also mirror this behaviour (Figure 5b), with UCB1 approaching its theoretical upper bound.

On account of the difference in the number of neighbours, and the presence of cycles, the variance of marginal regret is less uneven within the corresponding interquartile ranges, but larger on average in the random graph case (Figure 5b). There are more pronounced differences in the levels of regret as new agents are discovered every trial.

Directed Trees and Random Networks represent, perhaps, the most illustrative network configurations encountered while traversing the delegational domain. Trees provide an *a priori* framework — a predefined environment which every algorithm in our pool of heuristics explores at length, securing somewhat similar levels of regret. Random Networks, on the other hand, offer an unstructured setting for delegators to form their own neighbourhoods, thus evidencing the capacity of each procedure to forge delegation chains in ad-hoc environments. These conditions, we conjecture, account for the behaviour reported earlier in this section.

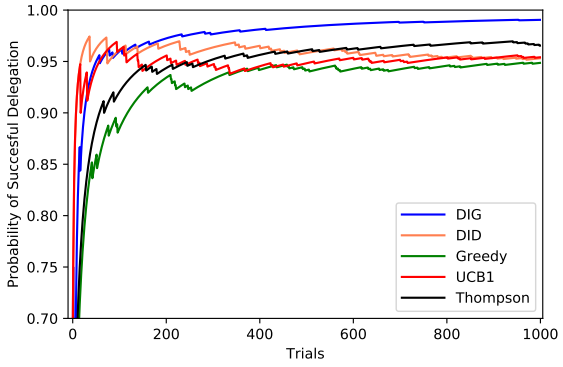
Lattices, tori and Scale-Free networks belong to the same kind of structures as Directed Trees. They are all graphs with predefined edges, obeying different connectivity rules. Our results indicate that the degree of connectivity dictates a pattern of differentiation among the heuristics (Figures 5a-8a), which progressively separates them into two groups: DIG, DID and UCB1 with the highest probabilities of successful delegation and the lowest levels of regret, and Greedy and Thompson at the other extreme.

Agents within Regular Lattices become more successful delegators when using DIG. Not only is the task more likely to be successfully executed, but the agreement with the oracle's choice of delegatee is improved. Thompson sampling and the Greedy approach converge to a relatively low probability of successful delegation, whereas DID and UCB1 attain better results, akin to those displayed by DIG.

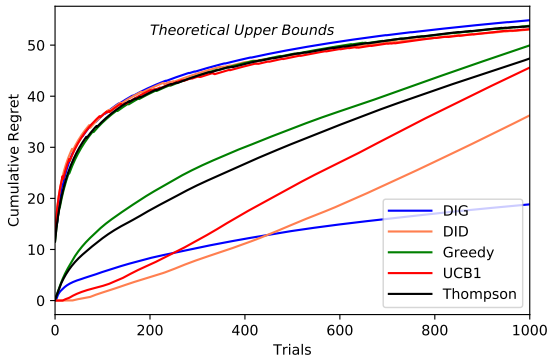
Figure 9 is indicative of DID's and, most notably, UCB1's progressions towards higher probabilities of successful execution within narrower intervals. To corroborate this claim, we tested for homoscedasticity within three different groups: 1) *High*, composed of DIG, DID and UCB1; 2) *Medium*, gathering DID and UCB1; and 3) *Low*, comprising Greedy and Thompson Sampling. Levene's statistic was used to assess the equality of the intra-group variances.

Table 2 reports the values of Levene's statistic alongside the corresponding p-values for each network structure under consideration. It indicates that with a significance level of 5%, the null hypothesis cannot be rejected in the following circumstances: 1) whenever members of *High* are implemented over Scale-Free Networks; 2) for several members of the *Medium* group, namely Directed Trees, Regular Lattices, and Scale-Free Network topologies; and 3) whenever agents use ϵ-greedy or Thompson Sampling in Regular Lattices or Scale-Free Networks.

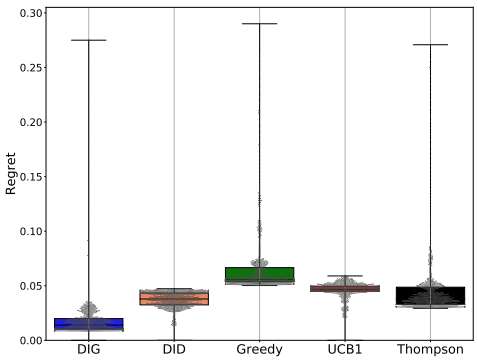
[1] The mean rate of convergence was approximated by the error of deviating from a probability of delegating equal to 1 (e_t), over the first 175 trials i.e., $q \approx \frac{log(e_{t+1}/e_t)}{log(e_t/e_{t-1})}, t \in \{1, \ldots, 175\}$. The cut-off point was obtained through the Welch method [17]



(a) Probabilities of Successful Delegation

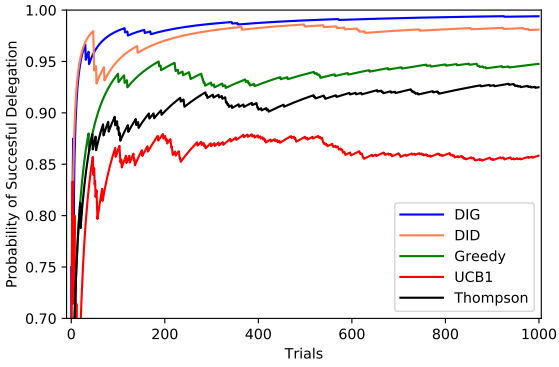


(b) Regret Metrics and Upper Bounds

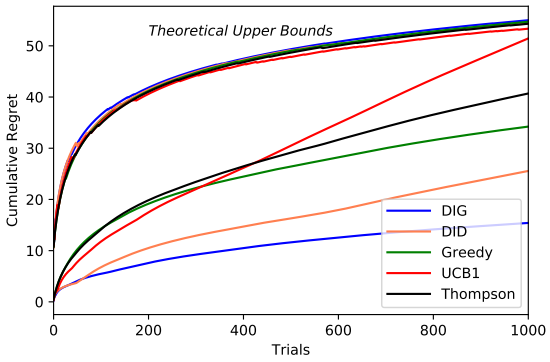


(c) Dispersion of Regret

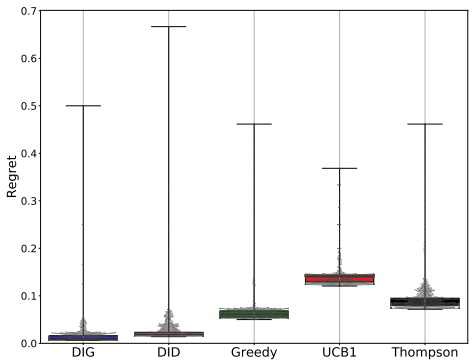
Fig. 4. Comparative Performance over Directed Trees



(a) Probabilities of Successful Delegation

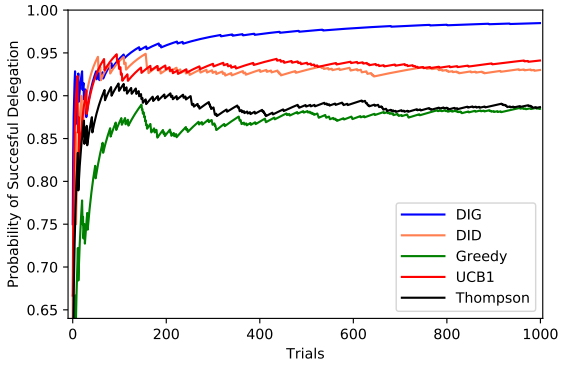


(b) Regret Metrics and Upper Bounds

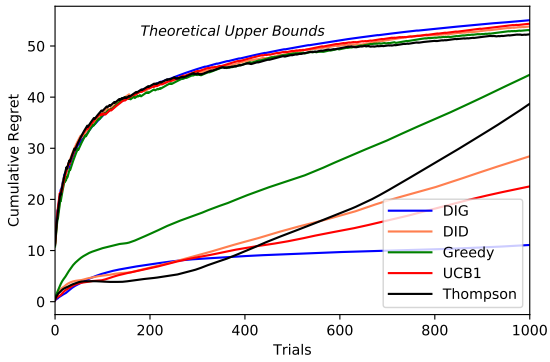


(c) Dispersion of Regret

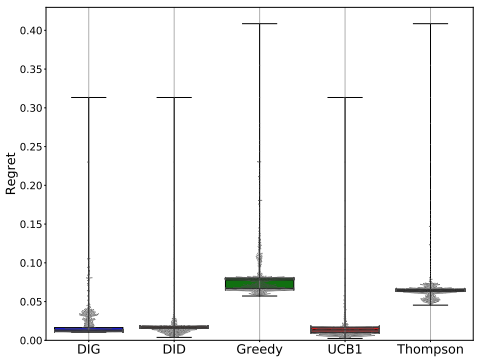
Fig. 5. Comparative Performance over Random Networks



(a) Probabilities of Successful Delegation

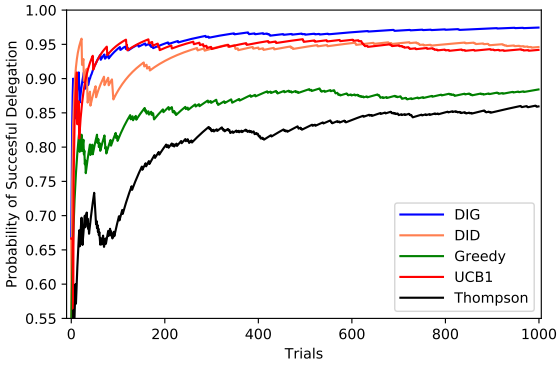


(b) Regret Metrics and Upper Bounds

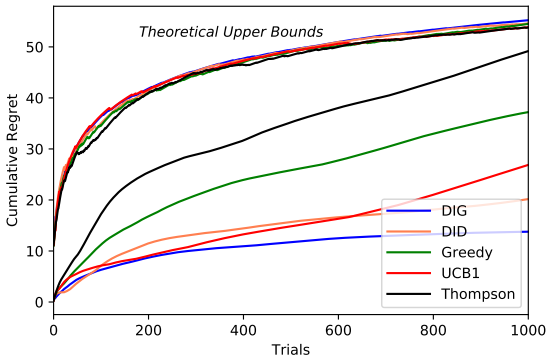


(c) Dispersion of Regret

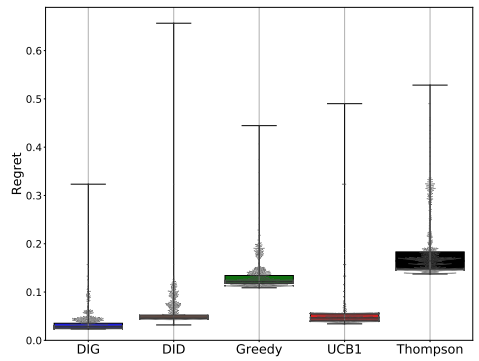
Fig. 6. Comparative Performance over Lattice Networks



(a) Probabilities of Successful Delegation

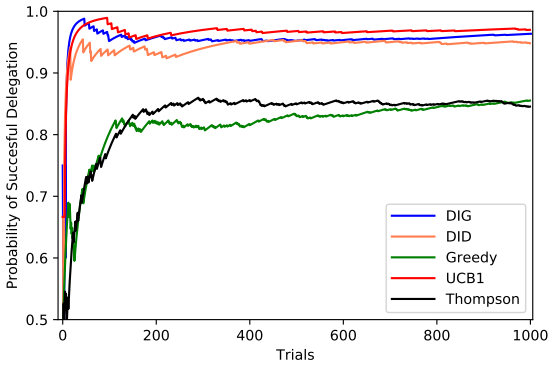


(b) Regret Metrics and Upper Bounds

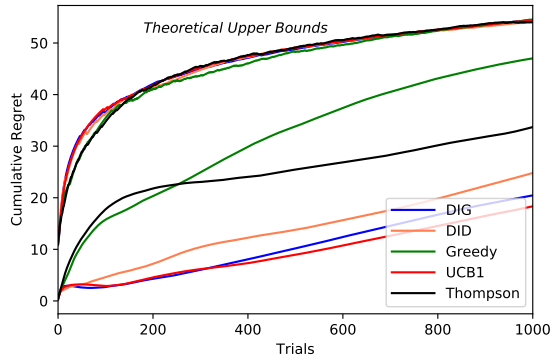


(c) Dispersion of Regret

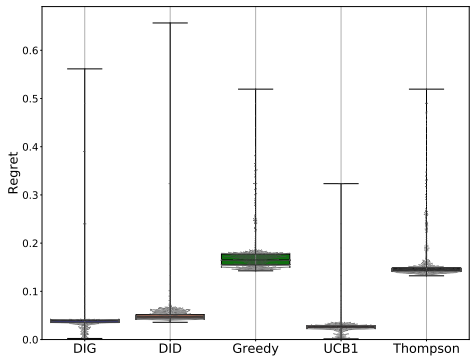
Fig. 7. Comparative Performance over Periodic Lattices



(a) Probabilities of Successful Delegation



(b) Regret Metrics and Upper Bounds



(c) Dispersion of Regret

Fig. 8. Comparative Performance over Scale Free Networks

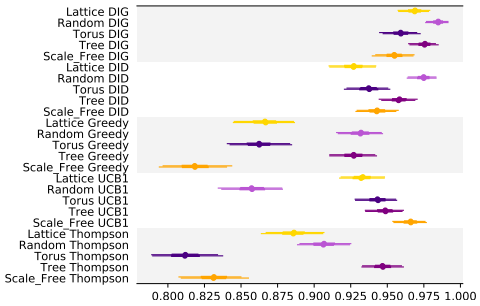


Fig. 9. Probability of Successful Delegation 95% Confidence Intervals

Algorithm	Network Structure	Probability of Successful Delegation	Mean Marginal Regret	Mean Rate of Convergence
DIG	DT	**0.975**	**0.113**	**0.642**
	RL	**0.969**	**0.069**	**0.539**
	RN	**0.985**	**0.031**	**0.648**
	SF	0.954	0.340	0.137
	PL	**0.959**	**0.273**	**0.366**
DID	DT	0.958	0.167	0.101
	RL	0.927	0.209	0.030
	RN	0.974	0.133	0.161
	SF	0.954	0.457	0.078
	PL	0.937	0.492	0.106
ϵ-Greedy	DT	0.927	0.480	0.237
	RL	0.866	0.815	0.131
	RN	0.931	0.437	0.226
	SF	0.818	0.719	**0.172**
	PL	0.863	0.728	0.149
Thompson Sampling	DT	0.947	0.278	0.284
	RL	0.886	0.617	0.002
	RN	0.906	0.686	0.163
	SF	0.831	0.590	0.003
	PL	0.811	1.24	0.223
UCB1	DT	0.948	0.260	0.069
	RL	0.933	0.150	0.060
	RN	0.858	1.173	0.010
	SF	**0.966**	**0.228**	0.025
	PL	0.943	0.434	0.089

Table 1 Relative Performance over Multiple Network Structure

Group	Network Structure	Levene's Statistic	P-Value
High	DT	36.69	1.82e-16
	RL	44.95	5.80e-20
	RN	8.82	1.51e-4
	SF	0.86	**0.42**
	PL	5.06	6.43e-3
Medium	DT	1.50e-3	**0.97**
	RL	9.15e-5	**0.98**
	RN	10.79	1.04e-3
	SF	1.883	**0.17**
	PL	3.146	**0.08**
Low	DT	4.22	0.04
	RL	25.07	**0.065**
	RN	69.50	3.33e-16
	SF	1.71	**0.19**
	PL	65.19	1.05e-14

Table 2 Analysis of Variance between Upper and Lower Groups of Algorithms

Equivalently, there is statistically significant evidence supporting our conjecture on the formation of two groups of heuristics, but accompanied by a third group arising from the transitioning of UCB1 and DID to *High* in environments which follow power-law rules.

Our findings on the probabilities of successful delegation are also mirrored by the distribution of regret over our selection of topologies (Figures 6c, 7c, and 8c. The distribution of regret changes considerably between topologies. It should be noted that for Directed Trees (an *a priori* network with regard to agent structure), the dispersion of regret when agents use UCB1 or DID become highly skewed in the manner of Random Networks, i.e., *a posteriori* networks.

We find considerable concentrations of regret above the median across lattices, tori, and Scale-Free Networks (Figures 6c, 7c, and 8c). Nonetheless all their interquartile ranges are narrow — there is little variability within the central 50% of the data. UCB1, DID, and the rest of MAB benchmarks abandon the distribution patterns displayed over Directed Trees (Figure 4c) in favour of the right-skewed distribution characteristic of DIG implementations.

5. Discussion and Future Work

Our results demonstrate that the DIG algorithm outperforms other approaches when dealing with recursive delegation problems. As future work, we intend to investigate the theoretical properties of the heuristic to further understand its salient features and the conditions behind its performance.

Stationarity offers a common ground for gaining a better insight into the functioning of DIG and the MAB benchmarks. A first step in this direction is made apparent by the similarities our DID heuristic shares with generalised versions of the Gittins Index under weaker forms of stationarity. Evolutionary algorithms have been used to tackle related problems [18], and investigating their performance in the delegation domain offers a potential avenue for future work.

Stationarity — or rather the lack of it — also raises analytical questions of great relevance in their own right. From a MAB perspective, non-stationarity springs from the nesting of bandits within one another and the ensuing processes of contemporaneous learning. Section 3.2 reveals that new delegation problems lie embedded in the constraints restricting every agent's objectives, in turn, dependent on future outcomes. That is, the general delegation problem is, in essence, a multilevel (stochastic) optimisation program with recursive objective functions [19, 20].

From an adversarial point of view, recursive delegation is a type of quitting game — a delegation game. It comprises a series of nested two-person non-zero-sum games, i.e., bimatrix games, subject to non-stationary stochastic perturbations. The delegation game may, in consequence, be similarly reinterpreted as a multilevel (stochastic) bilinear program [21]. This coincidence motivates further investigation into the potential of MAB and game-theoretical approaches to outline solutions to the general multilevel stochastic problem, while prompting the exploration of alternative heuristics based on evolutionary hierarchical genetic algorithms [20] on hierarchical reinforcement learning in non-stationary environments [22, 23].

In this work we have only considered the rewards gained through successful delegation. In the future, we intend to investigate the effects of resource constraints, explicit rewarding schemes, and potential costs to the delegation problem, by borrowing ideas from the principal-agent theory literature [24], and results from coalitional game theory [25].

To our knowledge, the only existing work on trust in the context of recursive delegation within the multi-agents community are [26] and [27]. In the former, the authors consider a supply chain problem and model it via recursive MABs, but focus on budget constraints for each arm, solving local bandit problems in parallel to identify trustworthy suppliers. [27] also consider the problem of recursive delegation, and evaluate how simple algorithms assigning responsibility for task delegation failure along delegation chains, affect the performance of the system.

6. Conclusions

In this paper we described the recursive delegation problem, and empirically demonstrated that a heuristic based on quitting games outperforms different multi-armed bandit based techniques — namely UCB1, ϵ-greedy, Thompson Sampling, and Brezzi and Lai's numerical approximation to the Gittins Index. Our heuristic outperforms these approaches both with regards to regret, and the probability of successful delegation over different graph topologies.

The ability to perform recursive delegation can be directly applied to electronic marketplaces, and has potential applications in areas such as logistics, routing and scheduling. Ultimately, this work serves as a starting point for investigating algorithms for recursive delegation, and a variety of open questions remain.

References

[1] E. Solan and N. Vieille, Quitting games, *Mathematics of Operations Research* **26**(2) (2001), 265–285.

[2] J. Granatyr, V. Botelho, O.R. Lessing, E.E. Scalabrin, J.-P. Barthès and F. Enembreck, Trust and Reputation Models for Multiagent Systems, *ACM Comput. Surv.* **48**(2) (2015), 27–12742, ISSN 0360-0300. doi:10.1145/2816826. http://doi.acm.org/10.1145/2816826.

[3] R.S. Sutton and A.G. Barto, *Reinforcement learning: An introduction*, Cambridge, MA: MIT Press, 2011.

[4] P. Auer and P. Fischer, Finite-time Analysis of the Multiarmed Bandit Problem, *Machine Learning* **47** (2002), 235–256.

[5] O. Chapelle and L. Li, An empirical evaluation of thompson sampling, in: *Advances in neural information processing systems*, 2011, pp. 2249–2257.

[6] J. Gittins, K. Glazebrook and R. Weber, *Multi-armed bandit allocation indices*, John Wiley & Sons, 2011.

[7] M. Brezzi and T.L. Lai, Optimal learning and experimentation in bandit problems, *Journal of Economic Dynamics and Control* **27**(1) (2002), 87–108.

[8] E. Gutin and V. Farias, Optimistic gittins indices, in: *Advances in Neural Information Processing Systems*, 2016, pp. 3153–3161.

[9] W. Hoeffding, Probability inequalities for sums of bounded random variables, *Journal of the American statistical association* **58**(301) (1963), 13–30.

[10] S. Agrawal and N. Goyal, Analysis of thompson sampling for the multi-armed bandit problem, in: *Conference on Learning Theory*, 2012, pp. 39–1.

[11] E. Solan and N. Vieille, Quitting games–An example, *International Journal of Game Theory* **31**(3) (2003), 365–381.

[12] J.S. Rosenthal, *A first look at rigorous probability theory*, World Scientific Publishing Company, 2006.

[13] G. Bennett, Probability inequalities for the sum of independent random variables, *Journal of the American Statistical Association* **57**(297) (1962), 33–45.

[14] R. Dahlhaus, On the Kullback-Leibler Information Divergence of LocallyStationary Processes, *Stochastic processes and their applications* **62**(1) (1996), 139–168.

[15] L. Li, D. Alderson, J.C. Doyle and W. Willinger, Towards a theory of scale-free graphs: Definition, properties, and implications, *Internet Mathematics* **2**(4) (2005), 431–523.

[16] B. Bollobás, C. Borgs, J. Chayes and O. Riordan, Directed scale-free graphs, in: *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, 2003, pp. 132–139.

[17] P.D. Welch, The statistical analysis of simulation results, *The computer performance modeling handbook* **22** (1983), 268–328.

[18] D.E. Koulouriotis and A. Xanthopoulos, Reinforcement learning and evolutionary algorithms for non-stationary multi-armed bandit problems, *Applied Mathematics and Computation* **196**(2) (2008), 913–922.

[19] S. Franke, P. Mehlitz and M. Pilecka, Optimality conditions for the simple convex bilevel programming problem in Banach spaces, *Optimization* **67**(2) (2018), 237–268.

[20] X. He, Y. Zhou and Z. Chen, Evolutionary Bilevel Optimization based on Covariance Matrix Adaptation, *IEEE Transactions on Evolutionary Computation* (2018).

[21] M. Petrik and S. Zilberstein, A Bilinear Programming Approach for Multiagent Planning, *arXiv preprint arXiv:1401.3461* (2014).

[22] T.D. Kulkarni, K. Narasimhan, A. Saeedi and J. Tenenbaum, Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation, in: *Advances in neural information processing systems*, 2016, pp. 3675–3683.

[23] A.S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver and K. Kavukcuoglu, Feudal networks for hierarchical reinforcement learning, *arXiv preprint arXiv:1703.01161* (2017).

[24] H. Zhang and S. Zenios, A dynamic principal-agent model with hidden information: Sequential optimality through truthful state revelation, *Operations Research* **56**(3) (2008), 681–696.

[25] O. Skibski, T.P. Michalak, T. Rahwan and M. Wooldridge, Algorithms for the shapley and myerson values in graph-restricted games, in: *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 197–204.

[26] S. Sen, A. Ridgway and M. Ripley, Adaptive Budgeted Bandit Algorithms for Trust Development in a Supply-Chain, in: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2015, pp. 137–144. ISBN 978-1-4503-3413-6. http://dl.acm.org/citation.cfm?id=2772879.2772900.

[27] C. Burnett and N. Oren, Sub-delegation and trust, in: *AAMAS*, IFAAMAS, 2012, pp. 1359–1360.