Resource Optimization for Program Committee Members: A Subreview Article

Michael Bender, Samuel Mccauley, Bertrand Simon, Shikha Singh, Frédéric Vivien

Resource Optimization for Program Committee Members: A Subreview Article*

Michael A. Bender[1], Samuel McCauley[1], Bertrand Simon[2], Shikha Singh[1], and Frédéric Vivien[2]

1 **Stony Brook University, Stony Brook, NY 11794-4400 USA.**
Email: {bender, smccauley,shiksingh}@cs.stonybrook.edu
2 **Univ Lyon, LIP, CNRS – ENS de Lyon – INRIA, Lyon 69007 France.**
Email: {bertrand.simon,frederic.vivien}@inria.fr

Abstract

This paper formalizes a resource-allocation problem that is all too familiar to the seasoned program-committee member. For each submission j that the PC member has the honor of reviewing, there is a choice. The PC member can spend the time to review submission j in detail on his/her own at a cost of C_i. Alternatively, the PC member can spend the time to identify and contact peers, hoping to recruit them as subreviewers, at a cost of 1 per subreviewer. These potential subreviewers have a certain probability of rejecting each review request, and this probability increases as time goes on. Once the PC member runs out of time or unasked experts, he/she is forced to review the paper without outside assistance.

This paper gives optimal solutions to several variations of the scheduling-reviewers problem. Most of the solutions from this paper are based on an iterated log function of C_i. In particular, with k rounds, the optimal solution sends the k-iterated log of C_i requests in the first round, the $(k-1)$-iterated log in the second round, and so forth. One of the contributions of this paper is solving this problem exactly, even when rejection probabilities may increase.

Naturally, PC members must make an integral number of subreview requests. This paper gives, as an intermediate result, a linear-time algorithm to transform the artificial problem in which one can send fractional requests into the less-artificial problem in which one sends an integral number of requests. Finally, this paper considers the case where the PC member knows nothing about the probability that a potential subreviewer agrees to review the paper. This paper gives an approximation algorithm for this case, whose bounds improve as the number of rounds increases.

Keywords and phrases Scheduling, Delegation, Subreviews

Digital Object Identifier 10.4230/LIPIcs.FUN.2016.7

1 Introduction

Serving on the program committee (PC) for a flagship theory conference is rewarding for the PC members, but it is intense and time-consuming work. Each submitted manuscript is typically assigned to three or four PC members. As a result, when you serve on a PC, you are given somewhere between 10 and 60 manuscripts to review, depending on the committee. Reviews need to be completed rapidly—on the order of one month. For many of these papers,

* This work was partially supported by NSF Grants CNS 1408695, CCF 1439084, IIS 1247726, IIS 1251137, CCF 1217708, and Sandia National Laboratories. This research was conducted during Shikha Singh's stay at IBISC, University of Evry Val d'Essonne, Evry 91000, France supported by a Chateaubriand Fellowship, and Samuel McCauley's stay at ENS de Lyon, supported by INRIA and a Chateaubriand Fellowship.

you have only a casual familiarity, meaning that doing a high-quality review will consume large amounts of time and may be fraught with uncertainty.

Consequently, most PC members rely on subreviewers to provide outside reviews. The subreviewer mechanism works as follows. When you serve on a PC, you decide which manuscripts require outside assistance. Then you scrape the Internet for subreviewers, sending request emails to friends, calling in favors, and searching out new domain experts.

Many of these subreview requests are denied. It is not uncommon that you need to send 3-8 review requests before you find a subreviewer that is not reading the paper for another PC member, already overburdened with reviews from the same conference, also serving on the PC, advising or being advised by one of the paper authors, having a baby, married to one of the paper authors, or just taking a well needed break from responding to email.

Thus, even the administrative process of collecting subreviews is work consuming and worth optimizing. One resource optimization, familiar to seasoned PC members, is the race to send out review requests the moment that the submissions have been meted out to the PC. The longer you wait to ask for a review from someone, the less likely it is that that person will comply. It is common that you miss out on collecting another subreviewer by just a few minutes—someone else's request showed up in the mail queue first.

Another optimization is the use of parallelism: send requests to several potential subreviewers in parallel to increase the probability of a positive answer. But every request takes time, and if multiple people say "yes," this leads to redundant work for the community and the PC member.

This paper studies this class of resource-allocation problem, which overburdened PC members naturally face while discharging their PC duties. Specifically, this paper studies the randomized resource-allocation problem of how to find subreviewers for the minimum expected cost. More generally, this paper explores how to delegate work to outside sources, when delegation has a cost, a nontrivial probability of failing, and the delegator must take this into account in his/her decisions.

Scheduling Model

There are N papers to review, denoted $1, \ldots, N$. Reviewing paper j yourself costs C_j. This cost is paper specific, because some submissions are easier for you to judge yourself, and others are harder. Each attempt to recruit a new subreviewer costs 1. The unit cost models the overhead of searching for email addresses, sending review requests, review reminders, thank you notes, requests for alternative reviewers, and all the other delegation overhead. Without loss of generality, $C_j > 1$; otherwise, it is not cost-effective to recruit subreviewers.

There is a (round-dependent and paper specific) probability of $p_{i,j}$ $(0 \leq p_{i,j} \leq 1)$ that at round i the candidate subreviewer declines to review paper j. Probability $p_{i,j}$ is monotonically increasing in i, since as time goes on, any candidate subreviewer is increasingly likely to be reviewing the paper for someone else or already has all of his/her time accounted for.

The reviewing process proceeds in $k+1$ rounds. In the first round, you email out a bolus of subreview requests for papers $1, 2, \ldots, N$. Some of these papers get successfully adopted by subreviewers. For those that do not, you send a second bolus of review requests in the second round, and so forth. In the last round $k+1$, you need to review those stray papers that did not find a subreviewer owner in the first k rounds.

The objective is to minimize the total expected cost to review all N papers. Because we are dealing with expected values, we can analyze the subreviewer-finding strategy for each submission in isolation. Henceforth, we focus on the case of a single paper, and the multiple-paper case follows by linearity of expectation. We simplify notation by dropping

the subscripts that identify papers, writing C and p_i. The main exception is Section 3.3, where a request budget must be distributed among the N papers.

Modeling the process by a series of rounds faithfully captures how many PC members (including the authors on this paper) act when they serve on PCs. Round 1 opens with a flurry of energy and enthusiasm. It ends with a quiescent period filled with hope, despair, gratefulness, and occasional sleep. Round 2, 3, and so forth proceed similarly, as the PC members work up the gumption to renew soliciting reviews, after getting more declines from potential subreviewers.

Although we describe this problem using the colorful language of beating the bushes for subreviewers, this model similarly captures a natural parallel scheduling problem. There are N tasks (the papers). The objective is to schedule all tasks while minimizing the expected cost. A task j can be executed locally (you write the review) or remotely (you assign the review to a subreviewer). A local execution is expensive. A remote execution is cheaper, but even launching a job has a cost and fails with a certain probability.

Results

We first explore the case where the rejection probabilities are known.

- For the case of two rounds, we give a closed-form optimal solution for minimizing the expected reviewing cost when the number of requests is not required to be an integer (e.g., we are allowed to send 3/5th of a review request, if we want).
- We give a closed-form optimal solution for multiple rounds when the number of requests in each round is not required to be an integer. This closed-form solution gives intuition about how requests should be distributed over rounds, and is an important building block for our further results.
- Using the closed form solution, we construct an optimal algorithm for sending integral requests; the running time is linear in the number of rounds.
- We then consider the bounded-reviewers case. Specifically, we further generalize to the (common) case where a submission may have only a bounded number R of knowledgeable experts. Once all R experts decline to review the paper, all resources are exhausted and the beleaguered committee member has no choice but to review the submission without help. We give an optimal algorithm if the probabilities remain constant, an approximation algorithm when they are monotonically increasing, and a pseudo-polynomial algorithm when papers have to share a budget R.

We also study the scheduling subreviewer problem for the case where the probability that a subreviewer rejects the subreview request is constant but unknown to the PC member.

- We give a two round strategy which is a $(4\sqrt{C/\ln C}+2)$-approximation for any subreview cost $C \geq 2$.
- We further generalize to multiple rounds and obtain a k-round strategy that is a $k(C^{1/k}+1)$-approximation.

Related Work

The problem of optimizing the conference-review process through a careful assignment of papers to PC members has been studied in various guises [4, 14, 21, 25, 28]. Goldsmith and Sloan [13] address the problem of assigning papers to PC members in order to minimize "whingeing." Merelo-Guervós et al. [23] give evolutionary algorithms to optimize this assignment problem. Assigning conference papers to review has also been studied as a multi-agent

fair-allocation problem [20]. Data mining and information-retrieval approaches have been used to model a reviewer's interest and build reviewer-recommender systems to assist paper assignment [2, 11, 16, 24]. Wang et al. [26, 27] present a survey on the various stages of the reviewer-assignment problem consolidating approaches from the areas of artificial intelligence, operations research, information retrieval, and algorithms.

Cormode [9] gives a fun guide on how *not* to review papers.

Probabilistic Scheduling. In many papers in scheduling and distributed computing, processors randomly choose which task to execute from a pool of tasks. Because multiple processors may choose the same task, there is some probability of wasted work. This is the case in the so-called do-all or write-all literature (among a very large literature, see for instance [1, 8, 19, 22]), as well as other contexts where processors randomly choose tasks to execute and may fail to find one that was not already chosen [3, 17, 18].

Fault tolerance. The subreviewer scheduling problem is related to the problem of executing tasks on failure-prone platforms, and particularly to the work on replication for fault-tolerance [6, 7]. There is an important difference, however. Most work on fault-tolerance for failure-prone HPC platforms assume that failures are rare [15]; typically the mean time between failures of a component is expressed in tens of years. This means that it is almost always better in practice only to duplicate failed tasks and not be proactive by replicating a priori. Furthermore, the low failure rate allows for first-order approximations [5, 10], which would be invalid in the current context of high rejection (i.e., failure) rates.

Preliminary Notions and Definitions

A ***strategy*** $\mathcal{S}$ is defined as the vector of requests sent in each round, that is, $\mathcal{S} = (R_1, R_2, \ldots, R_k)$. The probability of rejection is ***constant*** if $p_i = p$ for all rounds i, $1 \leq i \leq k$. The probability of rejection is ***round-dependent*** if p_i is monotonically increasing with respect to round i, that is, for any rounds i and j such that $1 \leq i < j \leq k$ we have $p_i \leq p_j$. We also consider the case of ***unknown probability*** where rejection probabilities are constant but not known to the PC member.

A round i ***succeeds*** if at least one request in round i is ***accepted***, that is, some subreviewer in the round agrees to subreview the paper. If a round i succeeds, then no more requests are sent in the later rounds. A round i ***fails*** if all requests sent in round i are ***rejected***, that is, none of the subreviewers asked want to subreview the paper. Round i fails with probability $p_i^{R_i}$. If rounds 1 through k fail, the $(k+1)$th round is reached. The PC member is then forced to review the paper on their own at a cost of C. We call C the ***self-reviewing cost***.

We devise strategies that send nonnegative ***integral number of requests***, that is, $R_i \in \mathbb{N}^0$ for $1 \leq i \leq k$. As an intermediate step, we construct strategies that send ***nonintegral*** or ***real number of requests***, that is, $R_i \in \mathbb{R}^+$ for $1 \leq i \leq k$. For example, in round i the PC member could send 3/5ths of a request, meaning that the rejection probability is $p_i^{3/5}$. As a shorthand, we call a strategy ***integral*** when the number of requests R_i for all rounds i is an integer. Otherwise, the strategy is a ***nonintegral*** or ***real strategy***.

Finally, we consider bounds on the total number of requests a reviewer can make. First, we consider the setting where the PC member can make an ***unbounded total number of requests*** to an arbitrarily large pool of potential subreviewers. Second, we consider the situation where there is a ***bounded total number R of requests*** that the PC member can make, which models the situation that there are a limited number of experts capable of reviewing the submission.

2 Optimal Strategy to Schedule Subreviews

In this section, we give an optimal $(k+1)$-round strategy for sending out subreview requests. We consider the case where there is an arbitrary pool of potential subreviewers, that is, there is no upper limit on the total number of requests that the PC member is allowed to make.

As an intermediate step, we give an optimal nonintegral strategy to find a closed-form solution. Then we give a linear-time algorithm that uses the closed-form solution and finds the optimal integral solution.

We start by explaining the high-level idea behind the optimal strategy. Let R_i denote the number of requests we send out in round i. Then our total expected cost is the following:

$$E(R_1, ..., R_k) = R_1 + p_1^{R_1}\left(R_2 + p_2^{R_2}\left(R_3 + ...\left(R_i + p_i^{R_i}\left(R_{i+1} ... + p_{k-1}^{R_{k-1}}\left(R_k + p_k^{R_k} C\right)\right)\right)\right)\right).$$

Note that the expected reviewing cost *conditioned on reaching a particular round i* is independent of the past requests $R_1, R_2, \ldots, R_{i-1}$; call this cost E_i.

$$E_i(R_i, \ldots, R_k) = R_i + p_i^{R_i}\left(R_{i+1} + p_{i+1}^{R_{i+1}}\left(\ldots p_{k-1}^{R_{k-1}}\left(R_k + p_k^{R_k} C\right)\right)\right).$$

Thus, we can determine the request sequence $R_i, R_{i+1}, \ldots, R_k$ that minimizes E_i by working backwards from the last round.

2.1 Optimal Nonintegral Strategy

The expected reviewing cost conditioned on reaching round k only depends on the self-reviewing cost C. We first compute the optimal number of requests to be made in the last round R_k. This is equivalent to devising an optimal two-round strategy.

▶ **Lemma 1.** *A two-round optimal nonintegral strategy sends R requests, where,*

- $R = 0$ *if* $C \leq \frac{1}{\ln \frac{1}{p}}$*; then the total expected cost is C;*
- $R = \frac{1}{\ln \frac{1}{p}} \ln\left(C \ln \frac{1}{p}\right)$ *if* $C > \frac{1}{\ln \frac{1}{p}}$*; then the total expected cost is* $\dfrac{1}{\ln \frac{1}{p}}\left(1 + \ln\left(C \ln \dfrac{1}{p}\right)\right)$.

We generalize this notion to obtain a complete $(k+1)$-round strategy that sends real requests and is optimal. To simplify analysis, we first present the case where the probability of rejection is constant across all rounds.

▶ **Theorem 2.** *An optimal $(k+1)$-round strategy for sending real requests when the rejection probability p_i is constant, $p_i = p$, is to send R_i requests in round i for $1 \leq i \leq k$, where,*

- $R_i = 0$ *if* $C \leq \frac{1}{\ln(1/p)}$ *(no subreview requests) with a total expected cost of $E = C$.*
- *Otherwise,*

$$R_i = \frac{1}{\ln \frac{1}{p}} \underbrace{\ln\left(1 + \ln\left(1 + \ln\left(1 + \ldots \ln\left(1 + \ln\left(C \ln \frac{1}{p}\right)\right) \ldots\right)\right)\right)}_{(k-i) \text{ times}}$$

with a total expected cost of $E = R_1 + \frac{1}{\ln(1/p)}$.

Theorem 2 generalizes to the case with monotonically increasing rejection probabilities.

▶ **Theorem 3.** *Let κ be the largest value in $\{1,..,k\}$ such that $C \geq \frac{1}{\ln \frac{1}{p_\kappa}}$ (with $\kappa = 0$ if $C < \frac{1}{\ln \frac{1}{p_1}}$). Then, when rejection probabilities are monotonically increasing, the optimal $(k+1)$-round strategy for sending real requests is to send R_i requests in round i where,*

$$\begin{cases} R_{\kappa+1} = ... = R_k = 0\ , \\ R_\kappa = \dfrac{1}{\ln \frac{1}{p_\kappa}} \ln\left(C \ln \dfrac{1}{p_\kappa}\right), \\ R_i = \frac{1}{\ln \frac{1}{p_i}} \ln\left(E_{i+1}^{\mathrm{OPT}} \ln \frac{1}{p_i}\right) \text{ where } E_i^{\mathrm{OPT}} = R_i + \dfrac{1}{\ln \frac{1}{p_i}} \text{ for } 1 \leq i < \kappa\ . \end{cases}$$

The total expected cost $E = C$ if $C < \frac{1}{\ln \frac{1}{p_1}}$. Otherwise, $E = E_1^{\mathrm{OPT}} = R_1 + 1/\ln(1/p_1)$, where

$$R_1 = \frac{1}{\ln \frac{1}{p_1}} \underbrace{\ln\left(\frac{\ln p_1}{\ln p_2}\left(1 + \ln\left(\frac{\ln p_2}{\ln p_3}\left(1 + \ldots \ln\left(\frac{\ln p_{\kappa-1}}{\ln p_\kappa}\left(1 + \ln\left(C \ln \frac{1}{p_\kappa}\right)\right)\right) \ldots\right)\right)\right)\right)}_{(\kappa-1) \text{ times}}.$$

2.2 Optimal Integral Strategy

We use Theorem 3 to construct an algorithm to determine an optimal integral $(k+1)$-round strategy. We first calculate the real request R_i for round i, and then check whether rounding R_i up or down leads to a better expected review cost conditioned on having reached round i; see Algorithm 1. Then we work backwards to compute R_{i-1} through R_1.

▶ **Theorem 4.** *There exists a linear-time algorithm for finding an optimal $(k+1)$-round integral strategy for the case when the rejection probabilities are monotonically increasing.*

Algorithm 1: Computes an optimal $(k+1)$-round integral strategy.

1 $\kappa \leftarrow \max\left\{i \in [1,k] \;\middle|\; C \geq \frac{1}{\ln \frac{1}{p_i}}\right\}$, or $\kappa \leftarrow 0$ if this set is empty
2 $R_i \leftarrow 0$ for all $i \in [1,k]$ `// R_i is the number of requests sent at round i`
3 $E \leftarrow C$ `// E is the current expected cost according to the values of R_i`
4 **for** *i from κ to 1* **do**
5 $\quad R_i \leftarrow \left\lfloor \ln_{\frac{1}{p_i}}(E \ln \frac{1}{p_i}) \right\rfloor$ `// R_i is rounded down by default`
6 $\quad$ **if** $p_i^{R_i} E > 1 + p_i^{R_i+1} E$ **then** $R_i \leftarrow R_i + 1$ `// Round up R_i if beneficial`
7 $\quad E \leftarrow R_i + p_i^{R_i} E$ `// Update E for the next iteration`
8 In each round i, send R_i requests. The expected cost is E.

3 Bounded Number of Requests

A PC member may know only a limited number of experts to ask for a subreview. Let R denote the budget of total number of requests a PC member is allowed to make. In this section, we construct strategies that make a total of at most R requests.

3.1 Constant Rejection Probabilities

We give an optimal algorithm for the case when the probability of rejection is constant across rounds, that is, $p_i = p$ for all i $(1 \leq i \leq k)$.

Optimal Strategy for Unbounded Rounds

We first consider the case when there is no limit on the number of rounds. Regardless of the bound R on total number of requests, there exists an optimal strategy that never sends more than one request per round; see Algorithm 2. This strategy is also optimal for the case when the number of rounds is limited to k and the budget on total requests $R \leq k$.

▶ **Theorem 5.** *When a maximum of R requests can be made over unlimited rounds and the rejection probabilities are constant ($p_i = p$), then the optimal integral strategy is to*
- *self-review immediately if $C \leq 1/(1-p)$, or*
- *make exactly one request per round if $C > 1/(1-p)$.*

The optimal expected cost is $\min(C, \frac{1}{1-p})$.

Algorithm 2: Optimal strategy when the rejection probabilities are constant and there is no bound on the total number of rounds.

```
if C < 1/(1−p) then PC member reviews on their own (and does not send any request)
if C > 1/(1−p) then
    while no request accepted and the number of requests sent is below the limit do
        PC member sends a single request
    if no request accepted then PC member reviews the paper
if C = 1/(1−p) then
    while no request accepted and the number of requests sent is below the limit do
        PC member chooses either to send a single request or review the paper
    if no request accepted then
        PC member reviews the paper
```

Optimal Algorithm for a Bounded Number of Rounds

We now devise a $(k+1)$-round algorithm for the case when we are only allowed to send R requests in total where $R > k$. (If $R \leq k$, then we can use Theorem 5.)

For a given self-reviewing cost C, we determine the optimal integral strategy $\mathcal{S}^*(C)$ that is allowed to make an unbounded number of requests. Let the total number of requests sent by $\mathcal{S}^*(C)$ be R^*. If $R^* \leq R$, our budget is sufficient and we execute strategy $\mathcal{S}^*(C)$. If $R^* > R$, there exists an optimal algorithm with budget R that uses all R requests; see Lemma 6. Thus, the challenge is to determine how to distribute $R < R^*$ requests across k rounds to minimize the total expected review cost.

▶ **Lemma 6.** *For a given self-reviewing cost C, let R^* be the total number of requests sent by an optimal integral strategy $\mathcal{S}^*(C)$ without a budget. When a budget of only R requests is allowed, where $R < R^*$, there exists an optimal strategy $\mathcal{S}^*_R(C)$ that sends exactly R requests.*

Let $R_1, R_2, \ldots, R_k$ be a partition of the budget R across the k rounds, that is, $\sum_{i=1}^{n} R_i = R$. Then the total expected cost is

$$E(R_1, \ldots, R_k) = R_1 + p^{R_1} R_2 + \ldots p^{R_1 + \cdots + R_{k-1}} R_k + p^R C. \tag{1}$$

Note that the final term, $p^R C$ (the expected self-reviewing cost incurred when all requests fail), stays the same regardless of how requests are partitioned among rounds. As long as C is

sufficiently large, further increasing C does not change the optimal partition of R among the k rounds. On the other hand, if C is sufficiently small that the unbounded-request solution does not need more than R requests, we already have a solution—Theorem 4.

With this observation, our goal is to find a self-reviewing cost C_R that uses exactly R requests. The partition determined by this solution is exactly what we are looking for.

The proof proceeds as follows. First, we assume that such a C_R exists and can be found efficiently. Given this assumption, we show that our algorithm computes an optimal strategy. Then, we complete the proof, that is, we show that an appropriate C_R always exists and can be computed efficiently.

▶ **Theorem 7.** *There exists an algorithm that computes an optimal $(k+1)$-round strategy for the case when the total number of requests allowed is bounded by R.*

Algorithm 3: Optimal strategy when the rejection probabilities are constant and when requests and rounds are limited.

```
S*(C) ← an optimal strategy for k+1 rounds and unlimited requests (use Algorithm 1)
R* ← number of requests required by S*(C)
if R* ≤ R then follow strategy S*(C)
else
    C_R ← self-reviewing cost for which an optimal strategy sends R requests
    S*(C_R) ← strategy computed using Algorithm 1 for cost C_R sending R requests
    follow strategy S*(C_R)
```

At first glance, it may seem immediate that such a C_R exists and can be found (perhaps using a binary search). However, the following example illustrates two complications. First, there may be several optimal strategies, each making different total numbers of requests. Second, arbitrarily small perturbations in C_R may change the optimal number of requests—in particular, there may only be a single value C_R that leads to exactly R requests.

▶ **Remark 8.** An example configuration where several strategies are optimal: let $p = 1/2$, $C = 8$ and consider 2 rounds plus the self-reviewing round. The strategies $(R_1, R_2) \in \{(1,2),(1,3),(2,2),(2,3)\}$ all achieve an expected cost of 3, and use between 3 and 5 requests.

In this example, $C = 8$ is the only value of C that uses exactly 4 requests. If $C = 8 - \varepsilon$, the optimal strategy uses 3 requests, and if $C = 8 + \varepsilon$ it uses 5.

Showing An Appropriate Self-Reviewing Cost Exists. If a strategy is allowed to make nonintegral requests, then showing that an appropriate self-review cost C_R exists is straightforward. The total number of requests as a function of the self-review cost C for nonintegral strategies is continuous and increasing; see Theorem 2. Thus, such a C_R always exists.

Now we show that an appropriate C_R exists even for integral strategies. We begin with a structural remark.

▶ **Lemma 9.** *When the self reviewing cost increases, the optimal expected cost does not decrease.*

The next two structural lemmas focus on a single round i.

First, we show that when the self-reviewing cost is increased, the number of requests sent in i (in any optimal solution) cannot decrease. For nonintegral requests, this statement follows directly from Theorem 2. The subtlety here is that the optimal strategy for integer

requests, as described in Algorithm 1, performs a rounding to compute each value R_i. We prove that our result still holds after this rounding.

▶ **Lemma 10.** *Given two integral strategies $\mathcal{S}_C$ and $\mathcal{S}_D$ that are optimal for a self-reviewing cost of C and D respectively, with $C < D$, the number of requests sent in any round i by $\mathcal{S}_C$ is not larger than the number of requests sent in round i by $\mathcal{S}_D$.*

Proof. Let $R_i^*(x)$ denote the (not necessarily integral) optimal number of requests sent at round i with self-reviewing cost x. We know by Theorem 3 that $R_i^*(x)$ is monotonically increasing in x; in particular, $R_i^*(C) \leq R_i^*(D)$.

We show the result by induction on i.

First, for $i = k$, we proceed by contradiction. Assume to obtain a contradiction that $\mathcal{S}_C$ (resp. $\mathcal{S}_D$) sends A requests (resp. B) at round k, with $A > B$. By definition, either $A = \lfloor R_i^*(C) \rfloor$ or $A = \lceil R_i^*(C) \rceil$; similarly $B = \lfloor R_i^*(D) \rfloor$ or $B = \lceil R_i^*(D) \rceil$. Since $R_i^*(C) \leq R_i^*(D)$ and $A > B$, we must have $A = B + 1$. Recall that, as proved in Theorem 3, $R_k^*(\alpha)$ is equal to the value of x that minimizes $x + p_k^x \alpha$.

Then, due to the optimality of A and B with a self-reviewing cost of C and D, respectively, we have (first substituting $A = B + 1$, then rearranging):

$$A + p_k^A C \leq B + p_k^B C \quad \Leftrightarrow \quad 1 + p_k p_k^B C \leq p_k^B C \quad \Leftrightarrow \quad C \geq \frac{1}{p_k^B (1 - p_k)}.$$

Similarly,

$$B + p_k^B D \leq A + p_k^A D \quad \Leftrightarrow \quad p_k^B D \leq 1 + p_k p_k^B D \quad \Leftrightarrow \quad D \leq \frac{1}{p_k^B (1 - p_k)}.$$

Since by definition we have $C < D$, we obtain $D \leq C < D$; hence, a contradiction.

Now assume this result holds true for any j larger than some i; we will show that it also holds for i. As proved in Theorem 3, $R_i^*(C)$ minimizes $x + p_i^x E_{i+1}$, where E_{i+1} is the optimal expected cost for the rounds after round i. When the self-reviewing cost increases, we know by Lemma 9 that E_{i+1} is nondecreasing. Then, we can apply the same reasoning as above and complete the induction. ◀

Now we show that as C increases, the requests sent at i increase continuously as well.

▶ **Lemma 11.** *Given a round i and an integer q, there exists a self-reviewing cost for which an optimal integral strategy sends exactly q requests at round i.*

Proof. We begin with some definitions. We define $\bar{E}_i(C)$ as the optimal expected cost, with integral numbers of requests, generated by rounds i through $k + 1$, conditioned on reaching round i (i.e., all requests at rounds 1 through $i - 1$ failed). To begin, we let $\bar{E}_{k+1}(C) = C$ to account for the self-reviewing round. We then recursively define $E_i(x, C) = x + p_i^x \bar{E}_{i+1}(C)$ as the expected cost generated by rounds i through $k + 1$ (conditioned on reaching round i) when x requests are sent at round i. Therefore, we have $\bar{E}_i(C) = \min_{x \in \mathbb{N}} E_i(x, C)$. We finally define $R_i(C) = \arg\min_{x \in \mathbb{N}} E_i(x, C)$: this is the smallest number of requests that can be sent at round i while retaining optimality.

We want to show that $R_i(C)$ spans all integers when C varies. Recall from Theorem 3 that the real value of x minimizing E_i is equal to $R_i^*(C) = \ln_{\frac{1}{p_i}} \left(\bar{E}_{i+1}(C) \ln \frac{1}{p_i} \right)$. Then if $\bar{E}_{i+1}$ is a continuous and strictly increasing function of C, R_i^* is also a continuous and strictly increasing function of C, and R_i^* spans all integers (for all $q \in \mathbb{N}$ there exists a value C_q such that $R_i^*(C_q) = q$). From Theorem 4, $R_i(C) \in \{ \lfloor R_i^*(C) \rfloor, \lceil R_i^*(C) \rceil \}$. Thus, since R_i^* spans all integers, R_i spans all integers as well ($R_i(C_q) = q$).

Therefore, it suffices to show that $\bar{E}_i(C)$ is a continuous and (strictly) increasing function of C. We prove this property by induction on i from $k+1$ to 1. We begin with the base case $i = k+1$. Then $\bar{E}_{k+1}(C) = C$, which is continuous and (strictly) increasing.

Suppose now that the inductive hypothesis holds for all $j > i$: $\bar{E}_j(C)$ is continuous and strictly increasing and, thus, R_j spans all integers. We must have $\bar{E}_i(C) \leq \bar{E}_{k+1}(C) = C$, therefore, $R_i(C) \leq C$ and $\bar{E}_i(C) = \min_{0 \leq x \leq C}(x + p_i^x \bar{E}_{i+1}(C))$. Then, $\bar{E}_i(C)$ is a continuous and strictly increasing function. ◀

With this structure in mind, we are ready to prove the existence of C_R.

▶ **Theorem 12.** *Given an integer number of requests R, there exists a self reviewing cost $C_R \in \mathbb{R}$ such that an optimal strategy uses R requests.*

Proof. As the self-reviewing cost increases from 1 to infinity, each R_i spans all the integers in increasing order, see Lemmas 10 and 11. Therefore, if C increases by a sufficiently small ε, the number of requests in each round either 1) stays the same, or 2) increases by 1. We refer to these increases as ***jumps***.

If only one R_i jumps for each small increase in C, then the theorem is proved. However, it may be that for all $\varepsilon > 0$, there may be two rounds R_i and $R_{i'}$ that both jump. See Remark 8 for an example. The remainder of the proof handles this case.

Because the expectations E_i^* (see Lemma 11) are continuous and strictly increasing, for each value S of R_i, there exists a unique value C such that both $R_i = S$ and $R_i = S+1$ are optimal. In other words, when R_i jumps, there is a value C such that both values of R_i are optimal.

For a self-reviewing cost C, let $R(C)$ be the minimum number of requests sent in an optimal solution.

Let C be the largest self-reviewing cost such that $R(C) \leq R$, and let $\mathcal{S}$ be an optimal strategy using $R(C)$ requests. If $R = R(C)$, we have the result. Otherwise, we let $\delta = \lim_{\varepsilon \to 0^+} R(C+\varepsilon) - R(C)$ be the minimum number of jumps when C increases by any ε. By definition of δ, we have $\delta \geq R - R(C)$.

Recall that at the point where R_i jumps, both R_i and $R_i + 1$ lead to the same expected cost. Then, there exist δ rounds in which we can add a request in $\mathcal{S}$ without modifying the expected cost. Choose $R - R(C)$ of these rounds, and increment the number of requests. We have shown that this retains the (optimal) total expected cost. Thus for any R, we get a C that has an optimal strategy with exactly R requests. ◀

Computing the Appropriate Self-Reviewing Cost. The main idea behind our algorithm for finding C_R is to use binary search. However, as mentioned in Remark 8, there may only be a single correct value of C_R. In fact, it may not be an integer, or even a rational number. Thus, once we are within 1 of the correct value of C_R, our algorithm jumps to the correct C_R more directly.

We begin with some definitions. We call a self reviewing cost C' a ***point of discontinuity*** at round i if there are two optimal strategies for cost C' that give a different number of requests in round i. (Note that these are similar to the "jumps" in the proof of Theorem 12.) Let $R_i(C)$ (resp. $R_i^+(C)$) be the minimum (resp. maximum) number of requests at round i that ensures optimality for a self-reviewing cost of C. Then we formally define $R(C)$ as the sum over all rounds of $R_i(C)$, and $R^+(C)$ likewise as the sum over $R_i^+(C)$. Note that these quantities can be computed by Theorem 7.

Our algorithm begins with a binary search, starting with the interval $[\frac{1}{1-p}, C]$, and ending on an interval $[L, U]$ which has $R_i(U) - R_i(L) \leq 1$ for all rounds i. In particular, we stop here because each round has at most one point of discontinuity in $[L, U]$.

Then, we refine the search within $[L, U]$. The algorithm finds, for each round i, the point of discontinuity within $[L, U]$ for round i; call it C_i. We then update L to be the smallest such C_i. We repeat until $R^+(L) = R$, at which point L is the desired value of C_R

Algorithm 4: Computing a self-reviewing cost C_R given R.

```
1  L ← 1/(1−p); U ← C_0                          // Lower and upper bounds on C_R
2  while ∃i, R_i(U) − R_i(L) ≥ 2 do               // binary search
3      x ← (1/2)(L + U)
4      if R ∈ [R(x), R⁺(x)] then return x
5      else if R(x) < R then L ← x
6      else U ← x
7  while R⁺(L) < R do
8      I ← {i | R_i(U) = R_i(L) + 1}
       // Value of the expectation before the first discontinuity in [L, U]
9      E(R_1, ..., R_k, C) ← R_1 + p^{R_1} R_2 + ··· + p^{R_1+···+R_k} C
10     R_i ← R_i(L) for each i
11     foreach i ∈ I do
12         C_i ← solution of E(R_1, ..., R_i, ..., R_k, C) = E(R_1, ..., R_i + 1, ..., R_k, C)
       // The minimum value corresponds to the first discontinuity
13     L ← min_{i∈I} C_i
14 return L
```

▶ **Theorem 13.** *Given a request target R, and a self-reviewing cost C (such that all optimal solutions for C require more than R requests) Algorithm 4 computes an appropriate self-reviewing cost C_R which has an optimal solution with exactly R requests.*

Proof. First, Theorem 12 ensures the existence of a self-reviewing cost C_R for which an optimal strategy with R requests exists.

By Lemma 10, we know that $R_i(C)$ and $R(C)$ are nondecreasing. Moreover, by Theorem 5, we know that $R(\frac{1}{1-p}) = 0$, and our theorem assumes that $R(C_0) > R$. Therefore, throughout the first loop of Algorithm 4, we know that there exists $C_R \in [L, U]$. Then, by Lemma 11, we know that each $R_i(C)$ spans every integer, so the first loop terminates. Thus, when the second loop is initiated, $[L, U]$ is in an interval containing C_R, and in which each round has at most one point of discontinuity.

Now, we study a given iteration of the second loop. Let j be the index of the round with the first discontinuity in $[L, U]$. First, note that a round i encounters a discontinuity at the point C if and only if C meets the requirement of Line 12, where the optimal numbers of requests sent at each round for a self-reviewing cost of C are given by $R_1, \ldots, R_k$. Therefore, C_j is indeed the smallest point of discontinuity in $[L, U]$, i.e., we have $R(L) = R(C_j) < R^+(C_j)$, and for every other i, we have $R(L) = R(C_i)$. Thus, during the execution of the second loop, the value of L is increased without skipping any point of discontinuity. This proves the termination and the correctness of this loop.

Finally, we show that this algorithm is efficient, by bounding the number of iterations of the first loop. Note that if a round i has a discontinuity at a self-reviewing cost C_i^1, the

next discontinuity C_i^2 satisfies $C_i^2 \geq C_i^1/p$, see below. Indeed, this inequality is actually an equality for the last round as the self-reviewing cost that has optimal strategies in which round k sends R_k or $R_k + 1$ requests is equal to $\frac{1}{p^{R_k}(1-p)}$. Then, as the previous rounds increase with a smaller rate, the inequality is proved.

Therefore, the first loop contains $O(\log_{1/p} C_0)$ iterations. ◀

3.2 Round-Dependent Rejection Probabilities

In this section we consider the case where rejection probabilities are not constant but are monotonically increasing.

The optimal strategy in Section 3.1 for constant rejection probabilities is based on the fact that when only an insufficient budget of R requests is available, then the optimal partition of R across the k rounds is independent of C. However, when the rejection probabilities are monotonically increasing this is no longer true. For rejection probabilities p_i for round i, the last term in the expression of expected review cost (Equation 1) becomes $p^{\Sigma_i R_i} C$ and thus the optimal review cost depends on the partition $R_1, \ldots, R_k$ of the budget R.

For the case of round-dependent rejection probabilities, we give a greedy strategy which achieves a approximation ratio dependent on the number of rounds. Roughly speaking, this strategy sends $R_i = \left\lfloor OPT \Big/ \prod_{j=1}^{i-1} p_j^{R_j} \right\rfloor$ requests in the ith round, finding the value of OPT through a binary search.

▶ **Theorem 14.** *There exists a greedy strategy whose expected cost is at most* $1 + (k+1)OPT$ *where* OPT *is the cost of the integral optimal strategy using at most* R *requests and* k *rounds.*

3.3 Multiple Papers Sharing a Common Request Budget

In this section we consider the case where the PC member can send a total of at most R requests and these requests have to be distributed among the N papers the PC member has been assigned to review. We first show that there exists a pseudo-polynomial-time algorithm to compute the optimal strategy for one paper using at most R requests. We then extend it to the optimal strategy for N papers, sharing a total budget R, with round-dependent rejection probabilities.

▶ **Theorem 15.** *There exists a pseudo-polynomial time algorithm computing an optimal strategy in* $O(kR^2)$ *time for any paper* j *such that the rejection probabilities with rounds are monotonically increasing and the total number of requests is bounded by* R*.*

Proof. The algorithm uses dynamic programming to build a table $E_j[i, r]$, for $1 \leq i \leq k+1$ and $0 \leq r \leq R$. $E_j[i, r]$ is the optimal expected cost when exactly r requests are spread among the rounds $R_i, R_{i+1} \ldots R_k$. $E_j[i, r]$ is constructed using the following equation:

$$\forall i \in [1, k],\ \forall r \in [0, R], \quad E_j[i, r] = \min_{0 \leq x \leq r} \left(x + p_{i,j}^x E_j[i+1, r-x] \right).$$

The algorithm proceeds by decreasing the value of round i, starting at round $i = k+1$. The table E_j is initialized by setting $E_j[k+1, r] = C_j$ for all r. For each round and each value of r, the algorithm has a cost of $O(R)$ to update the table E_j; hence, the overall complexity is $O(kR^2)$. The space used is $O(kR)$. The optimal strategy is determined by finding the value $\mathcal{R}$ that minimizes $E_j[1, \mathcal{R}]$ (for $0 \leq \mathcal{R} \leq R$). ◀

We use this result to build a solution for the general case for a total of N papers.

▶ **Theorem 16.** *There exists a pseudo-polynomial time algorithm computing an optimal strategy in $O(NkR^2)$ time for sending subreview requests for a total of N different papers such that the rejection probabilities are monotonically increasing with rounds and the total number of requests across the N papers is bounded by R.*

Proof. The algorithm uses dynamic programming to build a table $\mathbb{E}[j, r]$, for $0 \leq j \leq N$ and $0 \leq r \leq R$. $\mathbb{E}[j, r]$ is the optimal expected cost when exactly r requests are spread among the first j papers. $\mathbb{E}[j, r]$ is constructed using the following equation:

$$\forall r \in [0, R],\ \mathbb{E}[j, r] = \min_{0 \leq x \leq r} \left(E_j[1, x] + \mathbb{E}[j-1, r-x]\right),$$

where the value of $E_j[1, x]$ is given by Theorem 15. The algorithm proceeds by increasing values of j, from 1 to N. The overall complexity is $O(NkR^2)$. ◀

4 Unknown Probability of Rejection

In this section we consider the case in which the probability that a subreviewer rejects a subreview request is constant but unknown to the PC member.

We begin with an observation. Suppose we have an unbounded number of rounds for sending requests. Our result for unbounded rounds (see Algorithm 2) states that an optimal algorithm either takes the cost of C immediately, or performs one request per round indefinitely. Even when we do not know the value of p, we can mirror this idea using an approach similar to the classic *ski-rental problem* (see e.g. [12]).

▶ **Observation 17.** Consider a strategy $\mathcal{S}$ that makes one subreview request for C rounds then reviews for a self-reviewing cost C. Then, $\mathcal{S}$ is a 2-approximation (regardless of the value of the rejection probability p).

Proof. Strategy $\mathcal{S}$ has cost at most $2C$. If $C \leq 1/(1-p)$, the optimal strategy is to self-review at cost C and $\mathcal{S}$ is a 2-approximation. The cost of $\mathcal{S}$ is bounded by $1+p+\ldots+p^{C-1}+p^C C \leq 1/(1-p)+p^C C$. Note that $p^C C = p^C + p^C + \ldots p^C \leq 1 + p^2 + \ldots p^C \leq 1/(1-p)$. Thus the total cost is at most $2/(1-p)$. So for $C > 1/(1-p)$, $\mathcal{S}$ is also a 2-approximation. ◀

Now we consider the case when the total number of rounds used to make subreview requests is bounded. We first give a 2-round strategy (one round of making requests, followed by a self-review if all requests failed) and then generalize to a $(k+1)$-round strategy.

Two Rounds. We start with a 2-round strategy that achieves an approximation ratio of $(2\sqrt{C}+2)$. Then we improve the approximation ratio asymptotically; see Theorem 19.

To analyze our strategies we prove the following structural lemma. Roughly speaking, it shows that as long as the optimal cost is not too big, the cost of sending subreview requests is greater than the expected self-reviewing cost.

▶ **Lemma 18.** *Assume the cost of the optimal nonintegral strategy for two rounds with self-reviewing cost C is $\mathrm{OPT} \leq \sqrt{C \ln C}$. Then if OPT sends R requests, $p^R C \leq R$.*

Proof. By contradiction. Note that $\mathrm{OPT} \leq \sqrt{C \ln C}$ implies that $\mathrm{OPT} < C$ since $C \geq 1$. Then by Theorem 2, we have $R = \log_{\frac{1}{p}}(C \ln \frac{1}{p})$, and $p^R C = \frac{1}{\ln \frac{1}{p}}$. With some algebra, if the lemma statement is false, then we must have $\ln \frac{1}{p} < e/C$. Note that $\ln(Cx)/x$ is minimized

when x is maximized; thus $\ln \frac{1}{p} = e/C$ is a lower bound on the cost of OPT. Replacing back into the cost of OPT, we have

$$\sqrt{C \ln C} \geq \text{OPT} = \frac{\ln C \ln \frac{1}{p}}{\ln \frac{1}{p}} + \frac{1}{\ln \frac{1}{p}} \geq \frac{2C}{e}.$$

Note that $\sqrt{C \ln C}/C$ is maximized at $C = e$ (for $C > 1$), at which point $\sqrt{C \ln C} =< 2C/e$. Thus $\sqrt{C \ln C} < 2C/e$ for any $C > 1$. ◀

Consider the 2-round strategy $\mathcal{S}$ that sends $\left\lceil \sqrt{C} \right\rceil$ requests in the subreview round. If all requests fail, a self-reviewing cost C is incurred in the next round.

If the optimal strategy sends more than $\sqrt{C}$ requests, or sends zero requests and incurs self-reviewing cost C, then $\mathcal{S}$ achieves an approximation ratio of $\sqrt{C} + 2$ since $\mathcal{S}$ has total review cost at most $\left\lceil \sqrt{C} \right\rceil + C$. If the optimal strategy sends $R < \sqrt{C}$ requests, then $p^R C \leq R < \sqrt{C}$ by Lemma 18, and we have $p^{\sqrt{C}} C < \sqrt{C}$. Thus, $\mathcal{S}$ has total review cost at most $2\left\lceil \sqrt{C} \right\rceil$ and the optimal strategy has cost at least 1.

Next, we improve the approximation ratio. Note that for small C ($C \leq e^4 < 55$), the warm-up strategy leads to better guarantees.

▶ **Theorem 19.** *Consider a 2-round strategy $\mathcal{S}$ that sends $\left\lceil \sqrt{C/\ln C} \right\rceil$ requests. This strategy is a $(4\sqrt{C/\ln C} + 2)$-approximation for any $C \geq 2$.*

Proof. We compare our algorithm's performance to an optimal strategy $\mathcal{S}^*$. To simplify analysis we do not require $\mathcal{S}^*$ to be integral; however, we do require that $\mathcal{S}^*$ either sends zero requests, or sends at least one request. This lower bounds the cost of an optimal integral strategy. Let OPT be the cost of $\mathcal{S}^*$. We divide the proof into several cases based on the value of OPT.

Case 1 (Large OPT): Assume $\text{OPT} \geq \sqrt{C \ln C}$. Since $\mathcal{S}$ has cost at most $\left\lceil \sqrt{C/\ln C} \right\rceil + C$, the approximation ratio is at most

$$\frac{\left\lceil \sqrt{C/\ln C} \right\rceil + C}{\sqrt{C \ln C}} \leq \frac{\sqrt{C/\ln C} + 1 + C}{\sqrt{C \ln C}} \leq \frac{1}{\ln C} + \frac{1}{\sqrt{C \ln C}} + \frac{\sqrt{C}}{\sqrt{\ln C}} \leq \frac{2\sqrt{C}}{\sqrt{\ln C}} + 1 \text{ for } C > 2.$$

Thus, in this case $\mathcal{S}$ is $(2\sqrt{C/\ln C} + 1)$-competitive. Note that $\sqrt{C/\ln C} > 1$ for all $C > 1$ and $1/\sqrt{C \ln C} < 1$ when $C^C > e$, which holds for all $C \geq 2$.

Case 2 (Small OPT): Assume $\text{OPT} \leq \sqrt{C/\ln C}$. If $\mathcal{S}^*$ sends R requests, its cost is $R + p^R C$. Since $R \leq \text{OPT} \leq \sqrt{C/\ln C}$, $p^R \geq p^{\sqrt{C/\ln C}}$. Note that $p^R C \leq \text{OPT}$. Then the cost of our algorithm is $\left\lceil \sqrt{C/\ln C} \right\rceil + p^{\sqrt{C/\ln C}} C \leq \left\lceil \sqrt{C/\ln C} \right\rceil + \text{OPT}$. Since the optimal algorithm has cost at least 1, this results in a $\left(2\sqrt{C/\ln C} + 1\right)$-approximation algorithm.

Case 3 (Medium OPT): Assume that $\mathcal{S}^*$ has cost OPT, where $\sqrt{C/\ln C} < \text{OPT} < \sqrt{C \ln C}$. By Lemma 18, if $\mathcal{S}^*$ makes R requests, then $p^R C \leq R$. Rearranging, we obtain $p \leq e^{(\ln R)/R - (\ln C)/R}$; taking the derivative with respect to R we get $(R/C)^{1/R}((\ln C)/R^2 + (1 - \ln R)/R^2) \geq 0$ since $R \leq C$. Thus $(R/C)^{1/R}$ is increasing with respect to R; since $R < \text{OPT}$, we obtain $p \leq (R/C)^{1/R} < (\text{OPT}/C)^{1/\text{OPT}}$.

Substituting, the approximation ratio is at most

$$\frac{\left\lceil \sqrt{C/\ln C} \right\rceil + p^{\left\lceil \sqrt{C/\ln C} \right\rceil} C}{\text{OPT}} \leq \frac{1 + \sqrt{\frac{C}{\ln C}} + \left(\frac{\text{OPT}}{C}\right)^{\frac{\sqrt{C/\ln C}}{\text{OPT}}} C}{\text{OPT}} \leq 2 + \left(\frac{C}{\text{OPT}}\right)^{1 - \frac{\sqrt{C/\ln C}}{\text{OPT}}}.$$

Let $x = \sqrt{C/\ln C}/\mathrm{OPT}$, so $x \in (1/\ln C, 1)$. Then the ratio is

$$2 + \left(x\sqrt{C \ln C}\right)^{1-x} = 2 + e^{(1-x)\ln(x\sqrt{C \ln C})}$$

Taking the derivative, we get

$$e^{(1-x)\ln(x\sqrt{C \ln C})}\left(\frac{1-x}{x} - \ln(x\sqrt{C \ln C})\right).$$

This is equal to zero only when $1/x = \ln(x\sqrt{C \ln C}) + 1$. Substituting into the original equation, we are bounded by:

$$2 + \left(x\sqrt{C \ln C}\right)^{1-x} = 2 + \left(\frac{\sqrt{C \ln C}}{\ln(x\sqrt{C \ln C}) + 1}\right)^{1-x} \leq 2 + \left(\frac{\sqrt{C \ln C}}{\ln \sqrt{C/\ln C}}\right)^{1-x}.$$

Note that $\ln C \leq C^{1/e}$ for all C. Then $\sqrt{C/\ln C} \geq C^{1/4}$, so

$$2 + \left(\frac{\sqrt{C \ln C}}{\ln \sqrt{C/\ln C}}\right)^{1-x} \leq 2 + 4\sqrt{\frac{C}{\ln C}}.$$

Finally, we test the boundary cases. When $x = 1$, $2 + \left(x\sqrt{C \ln C}\right)^{1-x} = 2 < 2 + 4\sqrt{C/\ln C}$. When $x = 1/\ln C$, then $2 + \left(x\sqrt{C \ln C}\right)^{1-x} \leq 2 + \sqrt{C/\ln C}$. Thus, the maximum is reached when the derivative is zero and we have the theorem. ◀

Multiple Rounds. We generalize to k rounds and give a $kC^{1/k}$-competitive algorithm. We begin with a useful lemma.

▶ **Lemma 20.** *Assume the optimal nonintegral algorithm has total expected cost* $\mathrm{OPT} < C$. *Then* $p^i i \leq \mathrm{OPT}$ *for all* $i \geq \mathrm{OPT}$.

Proof. The lemma is trivially satisfied if $i = \mathrm{OPT}$, since $p^{\mathrm{OPT}}\mathrm{OPT} \leq \mathrm{OPT}$. Secondly, note that by Theorem 2, $\mathrm{OPT} \geq 1/\ln \frac{1}{p}$.

Taking the derivative, we have

$$\frac{d}{di}p^i i = p^i\left(1 - i\ln\frac{1}{p}\right) < 0,$$

where the second inequality follows from the fact that $i > \mathrm{OPT} \geq 1/\ln \frac{1}{p}$. Thus, $p^i i$ decreases as i increases. In other words, $p^i i \leq p^{\mathrm{OPT}}\mathrm{OPT} \leq \mathrm{OPT}$. ◀

▶ **Theorem 21.** *Let* $\mathcal{S}$ *be a* $(k+1)$*-round strategy with unknown rejection probability* p *where:*

- *self-review immediately if* $\lceil C^{(i-1)/k} \rceil > C$, *or*
- *send* $\lceil C^{(i-1)/k} \rceil$ *requests at round* i.

Then $\mathcal{S}$ *is a* $k(C^{1/k} + 1)$*-approximation strategy.*

Proof. If $\mathrm{OPT} = C$, the strategy $\mathcal{S}$ sends at most kC requests, and is k-competitive. Now assume $\mathrm{OPT} < C$. We define a variable a which roughly characterizes the performance of OPT in terms of $\mathcal{S}$. Let a be such that $\lceil C^{a/k} \rceil \leq \mathrm{OPT} < \lceil C^{(a+1)/k} \rceil$; in other words,

$a+1$ is the last round where $\mathcal{S}$ makes fewer requests than the total cost of OPT. Note that $0 \le a \le k$. The cost of $\mathcal{S}$ in rounds 1 to a is bounded by

$$\sum_{i=1}^{a} \left\lceil C^{i/k} \right\rceil \le a \left\lceil C^{a/k} \right\rceil \le a \cdot \mathrm{OPT}.$$

Consider round $i \ge a+1$. The cost of $\mathcal{S}$ at round i is

$$p^{\sum_{j=1}^{i-1} C^{j/k}} \left\lceil C^{i/k} \right\rceil \le p^{C^{i/k}} C^{i/k} C^{1/k} + p^{C^{i/k}} \le \mathrm{OPT} \cdot C^{1/k} + 1,$$

where the final inequality follows from Lemma 20. Thus our algorithm has total cost at most $a \cdot \mathrm{OPT} + (k-a)(\mathrm{OPT} \cdot C^{1/k} + 1) \le \mathrm{OPT}(kC^{1/k} + k)$. ◀

5 Conclusion

Many of our structural lemmas in this paper compartmentalize the subreviewer problem into independent tractable subproblems, which are much easier to analyze. For example, we can optimize the subreviewing for each paper independently, thanks to the comforting magic of linearity of expectation. Similarly, we can analyze the later rounds in the reviewing process independently of preceding rounds, thanks to the magic of doing probability theory correctly.

Natural open problems abound, generalizing the subreviewer optimization problem from this paper. But in many of these generalizations, this independence and comfortable compartmentalizing gets thrown out the window.

For example, an assiduous PC member may want to obtain *two* independent subreviews for some controversial submission. Now what happens in later rounds depends on previous rounds, and we cannot immediately apply the "work-backward method" given in Section 2.

Similarly, different papers may share potential subreviewers—a PC member may have ten experts to ask about paper 1 and twelve to ask about paper 2, but five of these experts are the same. Now optimizing the subreviewing for these two papers probably cannot be done independently. We also lose independence if we have a limited pool of subreviewers, and each subreviewer has a paper-dependent and time-dependent rejection probability.

References

1. Dan Alistarh, Michael A. Bender, Seth Gilbert, and Rachi Guerraoui. How to allocate tasks asynchronously. In *Proc. of the 53nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 331–340, 2012.
2. Chumki Basu, Haym Hirsh, William W. Cohen, and Craig Nevill-Manning. Recommending papers by mining the web. In *Proc. of the IJCAI Workshop on Learning about Users*, 1999.
3. Michael A. Bender and Cynthia A. Phillips. Scheduling DAGs on asynchronous processors. In *Proc. of the 19th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 35–45, 2007.
4. Salem Benferhat and Jérôme Lang. Conference paper assignment. *International Journal of Intelligent Systems*, 16(10):1183–1192, 2001.
5. George Bosilca, Aurélien Bouteiller, Élisabeth Brunet, Franck Cappello, Jack Dongarra, Amina Guermouche, Thomas Hérault, Yves Robert, Frédéric Vivien, and Dounia Zaidouni. Unified Model for Assessing Checkpointing Protocols at Extreme-Scale. *Concurrency and Computation: Practice and Experience*, 26(17):2727–2810, 2013.
6. Henri Casanova, Fanny Dufossé, Yves Robert, and Frédéric Vivien. Mapping applications on volatile resources. *International Journal of High Performance Computing Applications*, 29(1):19, 2015.

7 Henri Casanova, Dounia Zaidouni, and Frédéric Vivien. Using replication for resilience on exascale systems. In Thomas Hérault and Yves Robert, editors, *Fault-Tolerance Techniques for High-Performance Computing*, page 50. Springer, 2015.

8 Bogdan S. Chlebus and Dariusz R. Kowalski. Cooperative asynchronous update of shared memory. In *Proc. of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 733–739, 2005.

9 Graham Cormode. How not to review a paper: The tools and techniques of the adversarial reviewer. *ACM SIGMOD Record*, 37(4):100–104, 2009.

10 John T. Daly. A higher order estimate of the optimum checkpoint interval for restart dumps. *Future Generation Computer Systems*, 22(3):303–312, 2004.

11 Susan T. Dumais and Jakob Nielsen. Automating the assignment of submitted manuscripts to reviewers. In *Proc. of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 233–244, 1992.

12 Hiroshi Fujiwara and Kazuo Iwama. Average-case competitive analyses for ski-rental problems. *Algorithmica*, 42(1):95–107, 2005.

13 Judy Goldsmith and Robert H. Sloan. The AI conference paper assignment problem. In *Proc. of the AAAI Workshop on Preference Handling for Artificial Intelligence, Vancouver*, pages 53–57, 2007.

14 David Hartvigsen, Jerry C. Wei, and Richard Czuchlewski. The conference paper-reviewer assignment problem. *Decision Sciences*, 30(3):865–876, 1999.

15 Thomas Hérault and Yves Robert. *Fault-Tolerance Techniques for High-Performance Computing*. Springer, 2015.

16 Seth Hettich and Michael J. Pazzani. Mining for proposal reviewers: lessons learned at the national science foundation. In *Proc. of the 12th Annual ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 862–871, 2006.

17 Z. M. Kedem, K. V. Palem, M. O. Rabin, and A. Raghunathan. Efficient program transformation for resilient parallel computation via randomization. In *Proc. of the 24th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 306–317, 1992.

18 Zvi M. Kedem, Krishna V. Palem, and Paul G. Spirakis. Efficient robust parallel computations. In *Proc. of the 22rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 138–148, 1990.

19 Dariusz R. Kowalski and Alexander A. Shvartsman. Writing-all deterministically and optimally using a nontrivial number of asynchronous processors. *ACM Transactions on Algorithms*, 4:33:1–33:22, 2008.

20 Julien Lesca and Patrice Perny. LP solvable models for multiagent fair allocation problems. In *Proc. of the 19th European Conference on Artificial Intelligence*, pages 393–398. IOS Press, 2010.

21 Xinlian Li and Toyohide Watanabe. Automatic paper-to-reviewer assignment, based on the matching degree of the reviewers. *Procedia Computer Science*, 22:633–642, 2013.

22 Charles Martel and Ramesh Subramonian. On the complexity of certified write-all algorithms. *Journal of Algorithms*, 16:361–387, 1994.

23 Juan Julián Merelo-Guervós and Pedro Castillo-Valdivieso. Conference paper assignment using a combined greedy/evolutionary algorithm. In *Parallel Problem Solving from Nature-PPSN VIII*, pages 602–611, 2004.

24 David Mimno and Andrew McCallum. Expertise modeling for matching papers with reviewers. In *Proc. of the 13th Annual ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 500–509, 2007.

25 Jaroslaw Protasiewicz. A support system for selection of reviewers. In *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 3062–3065, 2014.

26 Fan Wang, Ben Chen, and Zhaowei Miao. A survey on reviewer assignment problem. In *New frontiers in applied artificial intelligence*, pages 718–727. Springer, 2008.

27 Fan Wang, Ning Shi, and Ben Chen. A comprehensive survey of the reviewer assignment problem. *International Journal of Information Technology & Decision Making*, 9(04):645–668, 2010.

28 David Yarowsky and Radu Florian. Taking the load off the conference chairs: towards a digital paper-routing assistant. In *Proc. of the Joint SIGDAT Conference on Empirical Methods in NLP and Very-Large Corpora*, 1999.

A Omitted Proofs

Proof of Lemma 1. The expected cost when sending a single round of R requests is $E(R) = R + p^R C = R + e^{R \ln p} C$. Taking the first and second derivatives, we obtain $E'(R) = 1 + (\ln p) p^R C$ and $E''(R) = (\ln p)^2 p^R C$. Thus, E'' is (strictly) positive, and E' is (strictly) increasing. Then, E' is zero when $R = \frac{1}{\ln \frac{1}{p}} \ln\left(C \ln \frac{1}{p}\right)$. If $C \ln \frac{1}{p} \leq 1$, then $R \leq 0$. Because R must be nonnegative and because E' is (strictly) increasing then in that case the optimal solution is to send no requests ($R = 0$) for an expected cost of C. Otherwise, the optimal expected cost is $R + p^R C = R + \frac{1}{\ln \frac{1}{p}} = \frac{1}{\ln \frac{1}{p}} \ln\left(C \ln \frac{1}{p}\right) + \frac{1}{\ln \frac{1}{p}}$. ◀

Proof of Theorem 2. The expression for the total expected review cost can be written as:

$$E(R_1, \ldots, R_k) = \left(R_1 + p^{R_1} R_2 + \cdots + p^{R_1 + \cdots + R_{k-2}} R_{k-1}\right) + p^{R_1 + \cdots + R_{k-1}} (R_k + p^{R_k} C).$$

Let $E_k(R_k) = R_k + p^{R_k} C$. Using Lemma 1, find the optimal value of R_k and E_k.

If $C \leq \frac{1}{\ln \frac{1}{p}}$, then the optimal value for R_k is 0 and thus all the R_is must be 0 and the optimal strategy is to self-review immediately.

For $C > \frac{1}{\ln \frac{1}{p}}$, we do an induction on the round i, starting from $i = k$. The optimal value of R_k is the base case. That is,

$$R_k = \frac{1}{\ln \frac{1}{p}} \ln\left(C \ln \frac{1}{p}\right) \quad \text{and} \quad E_k^{\text{OPT}} = \frac{1}{\ln \frac{1}{p}} \left(C \ln \frac{1}{p}\right) + \frac{1}{\ln \frac{1}{p}}.$$

Let E_j denote the expected cost conditioned on reaching round j, then

$$E_j(R_j, ..., R_k) = R_j + p^{R_j} R_{j+1} + \cdots + p^{R_j + \cdots + R_{k-1}} R_k + p^{R_j + \cdots + R_k} C.$$

Suppose we have determined the optimal values of R_{i+1} to R_k for some value of i. And suppose that for $i + 1 \leq j \leq k$, the expected cost $E_j(R_j, ..., R_k)$ of an optimal solution is equal to $E_j^{\text{OPT}} = R_j + \frac{1}{\ln \frac{1}{p}}$. Now consider R_i. We rewrite the expression of the overall expected cost:

$$\begin{aligned} E(R_1, \ldots, R_k) &= \left(R_1 + \cdots + p^{R_1 + \cdots + R_{i-2}} R_{i-1}\right) \\ &\quad + p^{R_1 + \cdots + R_{i-1}} \left(R_i + p^{R_i} \left(R_{i+1} + \cdots + p^{R_{i+1} + \cdots + R_{k-1}} R_k + p^{R_{i+1} + \cdots + R_k} C\right)\right). \\ &= \left(R_1 + \cdots + p^{R_1 + \cdots + R_{i-2}} R_{i-1}\right) + p^{R_1 + \cdots + R_{i-1}} \left(R_i + p^{R_i} E_{i+1}^{\text{OPT}}\right). \end{aligned}$$

Thus, the optimal value for R_i does not depend on the values of R_1 to R_{i-1} and can be determined by Lemma 1 when substituting E_{i+1}^{OPT} to C. That is,

$$R_i = \frac{1}{\ln \frac{1}{p}} \ln\left(E_{i+1}^{Opt} \ln \frac{1}{p}\right) = \frac{1}{\ln \frac{1}{p}} \ln\left(\left(R_{i+1} + \frac{1}{\ln \frac{1}{p}}\right) \ln \frac{1}{p}\right) = \frac{1}{\ln \frac{1}{p}} \ln\left(1 + \left(R_{i+1} \ln \frac{1}{p}\right)\right).$$

To complete the proof, we use $E_i^{\mathrm{OPT}} = R_i + p^{R_i} E_{i+1}^{\mathrm{OPT}} = R_i + \frac{1}{\ln \frac{1}{p}}$ from Lemma 1. Finally, note that the total expected cost of the strategy is $E = E_1^{\mathrm{OPT}}$. ◀

Proof of Theorem 3. We prove the result by induction on i, starting from $i = k$. The expression for the total expected cost can be written as:

$$E(R_1, \ldots, R_k) = \left(R_1 + p_1^{R_1} R_2 + \cdots + \left(\prod_{j=1}^{k-2} p_j^{R_j} \right) R_{k-1} \right) + \left(\prod_{j=1}^{k-1} p_j^{R_j} \right) \left(R_k + p_k^{R_k} C \right).$$

Let $E_k(R_k) = R_k + p_k^{R_k} C$. The optimal value for R_k and E_k from Lemma 1 is:

$$\begin{cases} R_k = 0 \text{ and } E_k = C \quad \text{if } C < \frac{1}{\ln \frac{1}{p_k}} \\ R_k = \frac{1}{\ln \frac{1}{p_k}} \ln \left(C \ln \frac{1}{p_k} \right) \text{ and } E_k = E_k^{Opt} = R_k + \frac{1}{\ln \frac{1}{p_k}}, \text{ otherwise.} \end{cases}$$

If $C \leq 1/(\ln \frac{1}{p_k})$, then $R_k = 0$ and using induction the theorem follows for the rounds κ through k (recall that the p_i's are non-decreasing).

Let E_j denote the expected cost conditioned on reaching round j. Then

$$E_j(R_j, ..., R_k) = R_j + p_j^{R_j} R_{j+1} + \cdots + \left(\prod_{l=j}^{k-1} p_l^{R_l} \right) R_k + \left(\prod_{l=j}^{k} p_l^{R_l} \right) C.$$

Similar to the proof of Theorem 2, assume that we have optimal values of R_l and E_l, for $i + 1 \leq l \leq \kappa$ (and, thus, $R_{\kappa+1} = ... = R_k = 0$). Consider R_i. Rewriting the expression of the total expected cost we have:

$$\begin{aligned} E(R_1, \ldots, R_k) = & \left(R_1 + \cdots + \left(\prod_{j=1}^{i-2} p_j^{R_j} \right) R_{i-1} \right) \\ & + \prod_{j=1}^{i-1} p_j^{R_j} \left(R_i + p_i^{R_i} \left(R_{i+1} + \cdots + \left(\prod_{j=i+1}^{k-1} p_j^{R_j} \right) R_k + \left(\prod_{j=i+1}^{k} p_j^{R_j} \right) C \right) \right) \\ & = R_1 + \cdots + \left(\prod_{j=1}^{i-2} p_j^{R_j} \right) R_{i-1} + \left(\prod_{j=1}^{i-1} p_j^{R_j} \right) \left(R_i + p_i^{R_i} E_{i+1}(R_{i+1}, ..., R_k) \right). \end{aligned}$$

Since the values of R_{i+1} to R_κ are optimal, and the optimal value for R_i can be determined using Lemma 1 by substituting E_{i+1}^{OPT} to C. Therefore, we obtain the optimal value of R_i:

$$R_i = \frac{1}{\ln \frac{1}{p_i}} \ln \left(E_{i+1}^{\mathrm{OPT}} \ln \frac{1}{p_i} \right) = \frac{1}{\ln \frac{1}{p_i}} \ln \left(\left(R_{i+1} + \frac{1}{\ln \frac{1}{p_{i+1}}} \right) \ln \frac{1}{p_i} \right) = \frac{1}{\ln \frac{1}{p_i}} \ln \left(\frac{\ln p_i}{\ln p_{i+1}} + R_{i+1} \ln \frac{1}{p_i} \right).$$

We have $E_i^{\mathrm{OPT}} = R_i + p^{R_i} E_{i+1}^{\mathrm{OPT}} = R_i + \frac{1}{\ln \frac{1}{p_i}}$ using Lemma 1. Finally, note that the total expected cost of the strategy is $E = E_1^{\mathrm{OPT}}$. ◀

Proof of Theorem 4. For $i > \kappa$, an optimal solution has $R_i = 0$ as in Theorem 3.

We prove that Algorithm 1 is correct and that at the end of Step 7 E equals $\bar{E}_i$ (at iteration i), the optimal expected cost conditioned on reaching round i.

We perform induction on i starting at $i = \kappa$ similar to the proof of Theorem 3. Let $E_\kappa(R_\kappa) = R_\kappa + p_\kappa^{R_\kappa} C$. We know that the nonintegral optimal value of R_κ also minimizes E_κ, i.e., $R_\kappa^* = \ln_{\frac{1}{p_\kappa}}(C \ln \frac{1}{p_\kappa})$. As E_κ decreases then increases, the integral optimal value of R_κ is either $\lfloor R_\kappa^* \rfloor$ or $\lceil R_\kappa^* \rceil$ (or both). This proves the result for $i = \kappa$.

Now, we assume the induction hypothesis for each round $l \in [i+1, \kappa]$. Consider R_i. We rewrite the expression of E when R_l is assigned to its optimal value, for $i + 1 \leq l \leq k$:

$$E(R_1, \ldots, R_i) = R_1 + \cdots + \left(\prod_{j=1}^{i-2} p_j^{R_j} \right) R_{i-1} + \left(\prod_{j=1}^{i-1} p_j^{R_j} \right) \left(R_i + p_i^{R_i} \bar{E}_{i+1} \right).$$

Using Lemma 1 the nonintegral optimal value of R_i is $\ln_{(1/p_i)}(\bar{E}_{i+1} \ln \frac{1}{p_i})$. The integral optimal value of R_i is one of its closest integers. Thus by induction, E has the value of $\bar{E}_{i+1}$ at the of Step 7 Algorithm 1 (at iteration $i + 1$). This proves the correctness for round i. Then, since E is updated to $R_i + p_i^{R_i} \bar{E}_{i+1} = \bar{E}_i$, the induction is complete. ◀