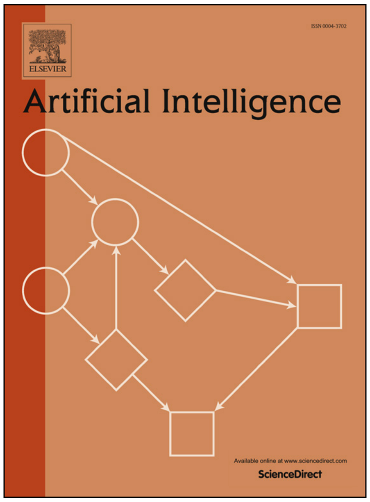

Matching Papers and Reviewers at Large Conferences

Kevin Leyton-Brown, Mausam, Yatin Nandwani, Hedayat Zarkoob, Chris Cameron et al.

PII: S0004-3702(24)00055-9
DOI: https://doi.org/10.1016/j.artint.2024.104119
Reference: ARTINT 104119

To appear in: *Artificial Intelligence*

Received date: 5 August 2022
Revised date: 11 March 2024
Accepted date: 16 March 2024

Please cite this article as: K. Leyton-Brown, Mausam, Y. Nandwani et al., Matching Papers and Reviewers at Large Conferences, *Artificial Intelligence*, 104119, doi: https://doi.org/10.1016/j.artint.2024.104119.

Matching Papers and Reviewers at Large Conferences

Kevin Leyton-Brown[a], Mausam[b], Yatin Nandwani[b,1], Hedayat Zarkoob[a,*,1], Chris Cameron[a], Neil Newman[a] and Dinesh Raghu[b,c]

[a] University of British Columbia, Vancouver, BC, Canada
[b] Indian Institute of Technology Delhi, New Delhi, India
[c] IBM Research, New Delhi, India

ARTICLE INFO

Keywords:
Reviewer-paper matching
Two-phase reviewing process
Conference organization

ABSTRACT

Peer-reviewed conferences, the main publication venues in CS, rely critically on matching highly qualified reviewers for each paper. Because of the growing scale of these conferences, the tight timelines on which they operate, and a recent surge in explicitly dishonest behavior, there is now no alternative to performing this matching in an automated way. This paper introduces *Large Conference Matching (LCM)*, a novel reviewer–paper matching approach that was recently deployed in the 35th AAAI Conference on Artificial Intelligence (AAAI 2021), and has since been adopted (wholly or partially) by other conferences including ICML 2022, AAAI 2022-2024, and IJCAI 2022-2024. LCM has three main elements: (1) collecting and processing input data to identify problematic matches and generate reviewer–paper scores; (2) formulating and solving an optimization problem to find good reviewer–paper matchings; and (3) a two-phase reviewing process that shifts reviewing resources away from papers likely to be rejected and towards papers closer to the decision boundary. This paper also describes an evaluation of these innovations based on an extensive post-hoc analysis on real data—including a comparison with the matching algorithm used in AAAI's previous (2020) iteration—and supplements this with additional numerical experimentation.[†]

1. Introduction

Submissions to prominent AI conferences have grown steadily in recent years. The AAAI Conference on Artificial Intelligence (AAAI) received fewer than 1,000 submissions in each of its first 33 years (1980–2012); fewer than 2,000 submissions annually until 2015; and fewer than 4,000 submissions annually until 2018. There has since been a significant surge in the number of submissions to AAAI conferences: AAAI 2019 received over 7,000 submissions, growing to over 7,500 in 2020 and to over 9,000 in 2021. Program committees have needed to keep pace with this surge; e.g., nearly 10,000 reviewers were involved in the 2021 conference. Given the scale and the tight timeline on which such large conferences operate—roughly three months between paper submission and author notification—it is becoming increasingly challenging to assign reviewers[2] to papers about which they can provide high-quality reviews.

[†] Writing this paper and conducting the experimental analysis described herein took almost a full year beyond the conclusion of AAAI 2021. Nevertheless, this paper is grounded in our experiences as Program Chairs (Mausam, Kevin), Workflow Chairs (Hedayat, Dinesh) and Associate Workflow Chairs (Yatin, Chris, Neil) at AAAI 2021. Many other people contributed to the the conference's design and operation. Notably, Qiang Yang was Conference Chair; Gabriele Röger and Yan Liu were Associate Program Chairs, and Guangneng Hu was a Workflow Chair; all contributed to the design and planning process throughout the year preceding the conference. Even more concretely impacting the content of this paper, Gabi provided draft text on filtering manipulative bids that we adapted into Section 2.2, and Yan made contributions to the MIP formulation described in Section 3. Keshav Sai Kolluru helped in the analysis of the bids, and Shantanu Agarwal helped in coding Subject Area Matching Score (section 2.3.1). The AAAI 2020 team provided us with their matching code, which facilitated the experiments in Section 5.2.1. The CMT support team provided invaluable real-time service and development during the conference which made our introduction of two-phase reviewing process feasible. Carol Hamilton and the AAAI Office provided innumerable forms of behind-the-scenes support. The AAAI 2021 conference and executive committees supported us in experimenting with our unconventional ideas for the conference. Finally, an organization as large as AAAI operates only through the efforts of literally thousands of volunteers in myriad roles; we thank all of them for their continued service to the community.

[*] Corresponding author

Email addresses: kevinlb@cs.ubc.ca (K. Leyton-Brown); mausam@cse.iitd.ac.in (Mausam); yatin.nandwani@cse.iitd.ac.in (Y. Nandwani); hzarkoob@cs.ubc.ca (H. Zarkoob); cchris13@cs.ubc.ca (C. Cameron); newmanne@cs.ubc.ca (N. Newman); diraghu1@in.ibm.com (D. Raghu)

[1] Joint lead student authors.

[2] Modern conference reviewing is often divided into four hierarchical roles, though terminology can vary. At AAAI, program committee members (PCs) review submissions; senior program committee members (SPCs) facilitate discussions among PCs and write meta-reviews; area chairs (ACs) oversee the review process, communicate with SPCs about final decisions, and recommend accept/reject decisions; Program Chairs design and oversee the whole review process and make final decisions for each submission. Additionally, AAAI 2021 introduced the role of associate program

Some key subproblems are estimating how qualified a given reviewer is to review a particular paper; tractably finding good matchings; determining which papers can be rejected without full review; and identifying reviewers who have bid, reviewed, or scored papers in an attempt to manipulate outcomes.

We address these challenges with a novel automated pipeline for reviewer–paper matching, which we name *Large Conference Matching (LCM)*. LCM was first deployed at AAAI 2021, and has been used at more than half a dozen other conferences since. The LCM pipeline consists of three main elements: (1) collecting and processing input data to identify problematic matches and generate reviewer–paper scores; (2) formulating and solving a constrained matching problem; and (3) splitting the reviewing process into two phases to enable matching more reviewers with stronger papers. We briefly describe each of these in turn (and then devote a full section to each in what follows). Open-source code allowing new conferences to deploy LCM is available at https://github.com/ChrisCameron1/LargeConferenceMatching.

Data Collection and Processing. Program Chairs need to transform a reviewer's profile into an estimate of how qualified that reviewer is to review each paper. Profiles can be as detailed as Program Chairs are creative, containing features such as past email addresses, DBLP/Semantic Scholar profiles, self-reported reviewing proficiency via a bidding phase, self-declared conflicts of interest (COIs), and texts of previously written papers. A key question that conference organizers need to answer is how to process this data and transform it into a numerical score assessing the quality of every plausible reviewer–paper pair. Two notable examples of such numerical scores are TPMS [Charlin and Zemel, 2013] and ACL scores [Neubig et al., 2020]. Both of these scores are computed based on textual similarity between a submitted paper and reviewer's prior publications. However, they are not always available; TPMS scores can only be computed for reviewers whose prior publications are known and ACL scores can only be computed for reviewers whose past publication abstracts are accessible, e.g., via a known Semantic Scholar profile. Therefore, LCM needs to be robust to missing TPMS or Semantic Scholar profiles. LCM aggregates all such data along with the subject area matching (SAM) scores, calculated based on the keyword match between papers and reviewers, to obtain final reviewer-paper scores between 0 and 1. These scores are normalized so that their values are unbiased by missing data. We refer to the resulting scoring function as *aggscore*. In AAAI 2021, aggscore leveraged TPMS and ACL scores, reviewers' and papers' primary and secondary subject areas, and explicit bids from reviewers.

Input data is only as useful as it is trustworthy: a growing concern at large-scale conferences is that some participants may self-report information to direct their submission to a specific reviewer or to ensure they review a specific paper [Jecmen et al., 2020]. To mitigate these risks, we construct relationship graphs based on each user's publication history to identify unreported conflicts (which, of course, can also arise via mistakes or oversights from non-malicious users). We also describe a set of heuristics for identifying suspicious patterns of self-reported conflicts and of bidding for papers to review. In both cases, we advocate erring on the side of caution, forbidding all such suspicious assignments.

Formulating a Constrained Matching Problem. Once scores have been assigned, the next key problem is to come up with an actual assignment. LCM formulates this problem as a Mixed-Integer Programming (MIP) problem. We are far from the first to use a MIP formulation for reviewer–paper matching [Taylor, 2008, Flach et al., 2010, Garg et al., 2010, Tang et al., 2010, Charlin et al., 2011, Charlin and Zemel, 2013, Lian et al., 2018, Kobren et al., 2019]. However, LCM goes well beyond the vanilla "find a score-maximizing assignment" formulation by including a wide range of soft constraints describing additional desiderata. For example, it enables penalizing the assignment of coauthors to review the same paper, rewarding geographic diversity in assignments, and penalizing matching pairs of reviewers who each bid positively on each other's papers. These soft constraints increase the odds that each paper is reviewed by an intellectually diverse set of reviewers and make potential collusion rings (reviewers conspiring to review each other's papers) [Littman, 2021] harder to sustain. Separately, LCM includes a seniority constraint to encourage distributing experienced reviewers across papers. We show in our experimental analysis of AAAI 2021 data that despite these advantages, soft constraints only slightly impacted the mean aggregate score and hence did not meaningfully degrade average match quality.

To ensure that the resulting MIP formulation can be solved within a reasonable amount of time, LCM sparsifies the space of possible assignments by creating variables representing the possibility of assigning a given paper to a given reviewer only when the pairing had a sufficiently high score; it then uses column generation as necessary to identify further candidate reviewers for hard-to-match papers. Coauthorship constraints will be too large to specify in memory for all but the smallest conferences; however, LCM leverages row generation as necessary (i.e., identifying an initial

chairs, who shadowed the program chairs and took primary responsibility for various discrete tasks. In this paper, we sometimes use the term reviewer to refer exclusively to PCs; in other cases, we refer to PCs, SPCs, and ACs. Distinctions between the two cases should nevertheless be clear from the context.

solution without coauthorship constraints and then iteratively adding individual violated constraints and rerunning the optimization). See Figure 1 for an illustration of information flow through LCM.

Two-Phase Reviewer Assignment. The quality of papers submitted to large and prestigious conferences varies vastly. There is a widespread perception that the submissions at the lowest end of this quality distribution do not merit the same investment of reviewing effort as the rest of the distribution.

Perhaps the most natural response to this situation is a mechanism for summarily rejecting weak papers without full reviews. This is typically operationalized as a quick perusal of each paper by an Associate Editor or Area Chair, and is widely employed across journals and conferences (e.g., in the AI community, it was used at IJCAI 2020 [Bessiere, 2020]). However, such approaches can be difficult to implement fairly: Area Chairs may not have strong opinions after a quick perusal; some papers will be missed; and such an approach would still impose a huge workload across many thousands of submissions. Even when it works well, such approaches still produce poorly explained decisions that many authors find frustrating: a quick perusal does not result in a detailed justification for the decision or suggestions about how work can be improved.

This is further complicated by the fact that reviews can be sufficiently noisy as to produce unreliable aggregate recommendations, even for those papers that *do* receive full consideration at large conferences. This was perhaps most convincingly demonstrated via an influential NeurIPS experiment [Lawrence and Cortes, 2014], which split reviewing between two independent program committees. The experiment found that while the strongest and weakest papers could be reliably identified, many papers close to the decision boundary were accepted by one program committee and rejected by the other.

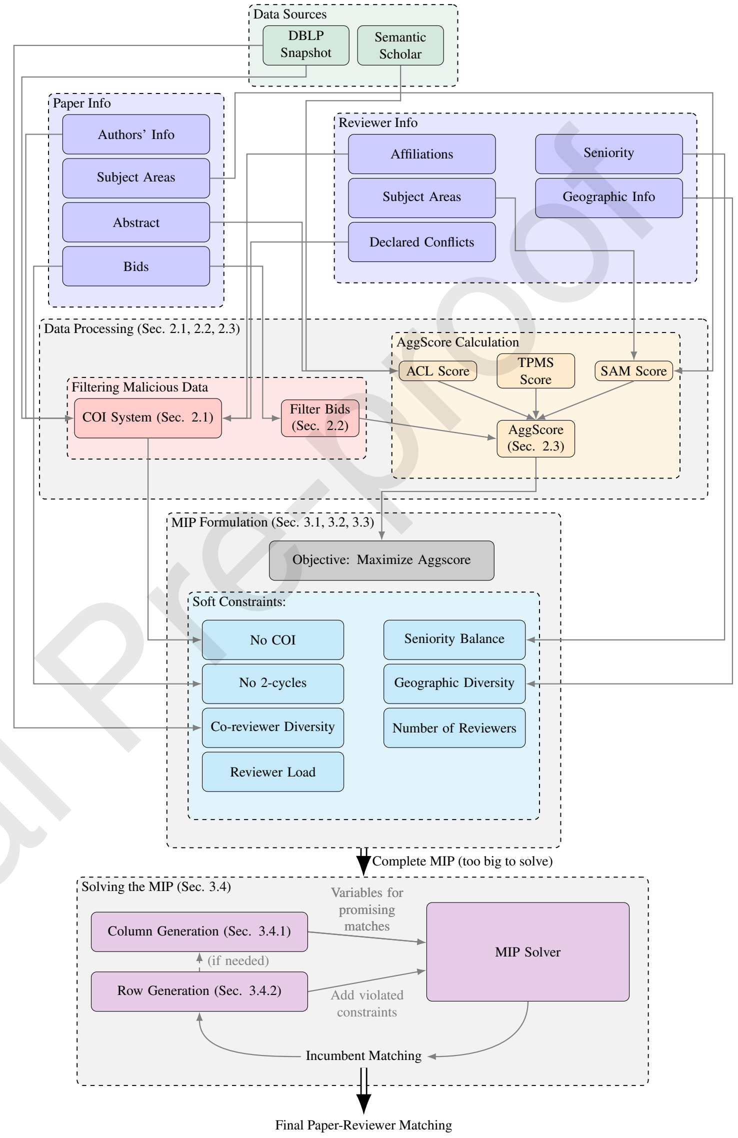

Figure 1: Schematic diagram showing the flow of information in our system.

The LCM pipeline aims to improve decision-making in such variable-variance settings via a *Two-Phase* mechanism for matching reviewers to papers.[3] Relative to a baseline that assigns the same number of reviews to every paper (say, 3), LCM rejects papers that are clearly below threshold after they have received a reduced number of reviews (say, 2), and improves aggregate review quantity for the papers that remain by increasing the number of reviews that each receives (say, to 4). Beyond its key property of variance reduction for borderline papers, this approach offers various other advantages. As compared to summary rejection, it gives meaningful feedback (two full reviews) to authors of rejected Phase 1 papers and avoids the overhead of a separate summary rejection phase by treating

[3] This approach has various historical antecedents; we defer a survey of these to Section 4.

all papers in the same way. It gives Program Chairs a second opportunity to assign additional reviewers for papers identified as problematic or for which Phase 1 reviewers went missing. Finally, it facilitates mechanisms like AAAI's "NeurIPS/EMNLP FastTrack", in which high-scoring rejected papers from sister conferences with late notification dates are allowed to proceed directly to Phase 2, with their reviews from the sister conference taking the place of Phase 1 reviews. We show in our analysis of AAAI 2021 data that LCM's two-phase matching approach enabled the rejection of 2,615 papers with only two reviews, which were redirected to higher-scoring papers. Analysis of an (inadvertent) natural experiment allowed us to estimate that these Phase I rejections included few false negatives: that only 2.9% would ultimately have been accepted if they had been promoted to Phase II (see Section 5.3.1). Two-Phase reviewing also enabled a FastTrack mechanism that attracted 688 full submissions (of which 268 were ultimately accepted).

The remainder of this paper is organized as follows. We begin by describing LCM's mechanisms for data collection and processing in Section 2. Then, we present LCM's mixed-integer programming formulation of the reviewer–paper matching problem in Section 3. We describe LCM's Two-Phase matching scheme in Section 4. Section 5 consists of an exhaustive experimental analysis of data from AAAI 2021, where LCM was first deployed. Finally, we summarize our contributions in Section 6. We note that much related work has studied different aspects of the reviewer–paper matching problem. To streamline exposition, we discuss related work in each corresponding section.

2. Data Collection and Processing

In this section, we describe LCM's techniques for collecting and processing raw data about reviewers and papers into an aggregated score for reviewer–paper matching.

2.1. Calculating Conflicts of Interests

A conflict of interest (COI) exists between a reviewer and a paper if the reviewer has a relationship with one or more of the paper's authors that may give them a personal interest in the paper's acceptance or rejection. For example, most of us would struggle to provide a fully unbiased opinion about a paper authored by a former supervisor. Clearly, reviewers should not be matched to papers with which they have a COI.

2.1.1. Filtering Manipulative Conflicts

Existing reviewer–paper matching systems detect COIs by asking authors and reviewers to disclose all of their current and past affiliations as well as to declare any additional explicit conflicts. This creates the possibility for a malicious author to provide very long lists of potential conflicts, either in an attempt to lead the system to choose a desired reviewer or simply to avoid expert assessment of their work. LCM identifies COI declarations as suspicious if they satisfy any of the following criteria: (1) a user listed an unusually large number of email domains as affiliations (e.g., 8 or more); (2) a user listed an unusually large number of non co-authors as COIs (e.g., 15 or more); (3) a user self-reported an unusually large number of COIs for which the corresponding reported users did not self-report a reciprocal COI (e.g., 10 or more). In applying LCM to AAAI 2021, we manually examined all such cases and removed all self-declared COIs that did not appear justified on inspecting the user's publicly available profile. The exact parameters listed above were chosen empirically to capture only extreme behavior: the three conditions respectively captured only (1) 1.1% of users who reported more than one conflict domain; (2) 0.14% of users who self-reported at least one conflict; and (3) 0.44% of users with at least one asymmetric conflict. In some cases, we observed evidence of quite egregious behavior: one user reported 35 conflict domains; another reported 57 asymmetric conflicts.

2.1.2. Predicting Unreported Conflicts of Interest

Many conference management systems ask users to self-report information such as email domain names for organizations with which the user is currently or was formerly associated. Unfortunately, such user-provided information is often incomplete. Users can purposefully withhold information; they can forget to include a past affiliation; and even if they remember every one, they can mistype a domain or fail to report all of their organization's aliases (*e.g.*, *iitd.ac.in* vs *cse.iitd.ac.in*). We recommend that conference organizers perform domain normalization and data cleaning (*e.g.*, identifying common domain prefixes and suffixes, like *.cse, .edu*, and *.ac*, and unifying different aliases of the same organization). LCM contains a starting point that we found useful for AAAI 2021.

Obviously, this approach cannot detect unreported or incorrect information. We thus augmented user-reported conflicts with additional inferred conflicts based on coauthorship information. We recommend that conference organizers require authors and reviewers to supply DBLP profiles. Given these profiles, LCM can download the full DBLP database[4] and thereby identify coauthorship relations without the need to disambiguate similar names. LCM is also able to augment this database with current conference submissions and build a weighted coauthorship graph, with nodes corresponding to authors, edges existing between pairs of coauthors, and weights representing the number of publications coauthored by the pair.

One way to prevent COIs would be to prohibit a reviewer from reviewing a paper submitted by a previous coauthor. However, we consider it too strict to categorically forbid matching coauthors who may not have interacted in years. (In Section 3.1, we describe softer ways in which LCM can discourage the selection of such matchings.) LCM thus considers a reviewer r_j to have a COI with paper p_i if any of the following conditions are satisfied for some author a of p_i:

1. r_j coauthored a current conference submission with a;
2. r_j coauthored a published paper with a over the previous 5 years; or
3. r_j and a coauthored more than 6 published papers over any time period.

LCM further adds conflicts between reviewer r_j and papers by an author a when it infers that a supervised r_j or that both r_j and a were students of the same supervisor. We note that advisor–advisee relationships between users are unknown; hence they must be extracted from the coauthorship graph. LCM infers that a was one of r_j's supervisors if all of the following conditions hold:

1. a began publishing considerably earlier than r_j (at least 5 years);
2. a published at least 10 more papers than r_j; and
3. a coauthored many of r_j's early papers (at least 3 out of the first 10).

These inferences do not need to be perfect: each COI simply prevents a particular reviewer–paper matching, and a few false positive COIs are unlikely to be harmful. Indeed, as we show later in Figure 4 (Section 5.1), at AAAI 2021 most papers had a large number of highly qualified reviewers, indicating that occasionally forbidding one of them was unlikely to materially affect the quality of the matching.

We observe that we are not the first to use DBLP to infer COIs; Long et al. [2013] describe methods for inferring coauthor, colleague, advisor–advisee, and competitor relationships. Like us, they used DBLP data to identify coauthor relationships. However, they relied on users' public web pages to identify colleague and advisor–advisee relationships. However, the latter approach requires all users to have reliable and accessible public web pages, which dramatically complicates its application to large conferences.

2.2. Filtering Manipulative Bids

Bids carry important information about reviewers' preferences that can significantly affect reviewers' satisfaction with an assignment. Unfortunately, reviewers can also maliciously submit bids with the aim of affecting a paper's chances of being accepted or rejected [Noothigattu et al., 2021, Wu et al., 2021]. Traditional approaches for detecting such misbehavior are via Area Chair (AC) oversight, spot checking of discussions, and whistle-blower reports. All of these approaches are labor intensive and scale poorly to huge conferences.

Jecmen et al. [2020] proposed alleviating this issue by using a randomized algorithm for the reviewer assignment problem, decreasing the impact of malicious behavior by controlling the probability that certain (suspicious) reviewers or pairs of reviewers are matched to a paper. However, such randomization can impose significant costs in terms of match quality and also still allows suspicious matches uncomfortably often. We nevertheless strongly agree with Jecmen et al. that malicious behavior is better prevented at the matching phase than detected afterwards. We found that the vast majority of papers at AAAI 2021 received a large number of bids by qualified reviewers. In such cases, as we have already argued for COIs, the abundance of qualified reviewers makes it relatively harmless to forbid or downgrade potentially suspicious matches even in the likely event that many of the suspicion-raising bids were not actually attempts at manipulation. LCM thus adjusts a reviewer's bids when they appear suspicious, which can happen in several ways. In what follows, we refer to the bid levels used by AAAI 2021—*not willing*, *in a pinch*, *willing*, and *eager*—but these, along with the constants that follow, can of course be adjusted in the software.

[4] The full DBLP database is publicly available at http://dblp.uni-trier.de/xml/dblp.xml.gz.

First, a reviewer might place only a few bids to manipulate the likelihood that they would be assigned particular papers (in the most extreme example, bidding positively on only a single paper). To mitigate this risk, LCM discards *all* of a Program Committee (PC) member's bidding information if they placed fewer than 9 positive bids overall (counting *in-a-pinch* bids as half). It employs a similar strategy at other levels of enthusiasm: if a bidder placed fewer than 4 *eager* bids, they are transformed into *willing* bids; then fewer than 4 *willing bids* are transformed into *in-a-pinch*. LCM performs a similar procedure for Senior Program Committee (SPC) members, setting a higher threshold of 10 to reflect the higher number of required positive bids.

Second, a reviewer can increase their chances of being assigned a particular paper by bidding negatively on many other papers for which they are qualified. LCM therefore discards all *not willing* bids of PCs and SPCs if the number of *not willing* bids exceeds six times the number of *willing* and *eager* bids.

Third, and most complex, *collusion rings* are arrangements under which two or more reviewers agree to positively review each other's papers, a dismaying phenomenon that is reportedly on the rise [Littman, 2021]. LCM defines two criteria to judge whether a bid by a reviewer r on a paper p is consistent with participation in a collusion ring: (1) at least 40% of r's positive bids are on papers authored by a specific author of p; (2) there exists some author r^* of p such that both r and r^* bid positively on each other's papers. Specifically, the sum of the number of r's positive bids on papers authored by r^* and the number of r^*'s positive bids on papers authored by r is at least 60% of the total number of positive bids by r and r^*. For both (1) and (2) above, LCM considers both *eager* and *willing* bids as positive. These criteria are not perfect; e.g., (1) can occur in small subcommunities where a few researchers focus on a given topic, and (2) can wrongly flag reviewers with a few submissions (e.g., two authors with one paper each bid on each other's paper). We consider it acceptable to generate some false positives (for the same reasons outlined above), but aim to reduce the number of such mistakes by requiring a substantial pattern of suspicious bidding across all of a reviewer's bids. In the assignment process LCM discards *all* of a reviewer's bids when either (1) or (2) suggest the possibility that the reviewer belonged to a collusion ring.

When deploying LCM at AAAI 2021, we conducted an extensive manual examination of bids, using various heuristic criteria. In all cases that we judged problematic, LCM had already eliminated the appropriate bids.

2.3. Scoring Matches

A key question in automated reviewer–paper matching is quantifying the value of matching a reviewer r_j with a submitted paper p_i, which we denote as s_{ij}. It is critical to get this right.[5] In brief, LCM considers a match to be good if the reviewer both has expertise in the paper's subject matter and is interested in the paper. We assess expertise by aggregating the degree of match between the paper's primary and secondary subject area keywords and those of the reviewers along whatever external signals are available (e.g., TPMS; ACL). We assess degree of interest via bids, assuming a baseline level of interest for papers about which a reviewer is predicted to have expertise and for which the reviewer did not submit any explicit (positive or negative) bid.

Much existing work has leveraged similar signals, much of which is focused on automatically discovering latent topics of a submission and comparing it with latent topics of a set of papers authored by a given reviewer [Mimno and McCallum, 2007, Long et al., 2013, Anjum et al., 2019, Kobren et al., 2019, Wieting et al., 2019]. Three further lines of related work deserve detailed discussion, since we either extended them (the first) or leveraged them directly (the second and third). First, Conry et al. [2009] proposed a score based on the primary and secondary subject areas of both a paper and a reviewer. The fundamental idea was to embed both reviewers and papers into a discrete vector space with each discrete dimension quantifying the affinity of either the reviewer or the paper to the corresponding topic. Second, the TPMS score [Charlin and Zemel, 2013] is computed based on the similarity between the text of a submitted paper and a reviewer's prior publications. Similarity is quantified as the dot product between a paper vector and a reviewer vector. The paper vector is constructed using the topic proportions in the text of the submitted paper identified by Latent Dirichlet Allocation (LDA) [Blei et al., 2003]. The reviewer vector is computed by taking an average of the topic proportions in each of the reviewer's prior publications. TPMS scores can only be computed for reviewers whose prior publications are known. Third, the ACL matching score [Neubig et al., 2020] compares abstracts of a submitted paper and the abstracts of a reviewer's prior publications in a dense vector space. The abstract representation is computed by averaging the embeddings of subwords generated by the SentencePiece tokenizer [Kudo and Richardson, 2018]. A score is then computed for each of the reviewer's prior publications with respect to the paper. The top three scores

[5] Indeed, a few readers may remember that our initial matching at AAAI 2021 had to be changed shortly after we released it. The reason was that we accidentally miscalibrated our scoring function. Based on the emails we received, reviewers definitely noticed flaws in the inital matching!

are aggregated to compute the ACL score. ACL scores can only be computed for reviewers whose past publication abstracts are accessible, e.g., via a known Semantic Scholar profile.

The rest of this section is organized as follows. We first describe how LCM computes Subject Area Matching (SAM) scores, a continuous extension of discrete affinity scores. Then, we describe how LCM normalizes other raw scores, such as TPMS and ACL, to ensure that they have the same scale as SAM scores. We then explain how it merges all of these scores to make the result robust to missing data. This can be a real concern at large conferences: e.g., in AAAI 2021, we needed to rely on reviewers to manually provide the information required to compute these scores and around 40% of reviewers had at least one missing entry. Next, we explain how LCM adjusts these scores to take into account reviewers' bids. Finally, we describe an approach for pruning low-scoring matches, which LCM uses to avoid the possibility of wildly inappropriate matches.

2.3.1. Subject Area Matching Score (SAM)

LCM scores the keyword match between papers and reviewers via an extension of the affinity scoring idea described earlier [Conry et al., 2009], which embeds each reviewer and paper in a vector space. LCM's approach goes beyond explicitly specified subject areas by learning correlations between different keywords. This is important because the approach of Conry et al. [2009] assigns a weight of 0 to keywords that are not specified explicitly, even when these have a strong degree of semantic similarity with keywords that *are* specified. For example, keywords *“Humans and AI -> Human-Agent Negotiation"* and *“Game Theory and Economic Paradigms -> Negotiation and Contract-Based Systems"* were highly correlated at AAAI 2021, and thus it might have been inaccurate to assign a weight of 1 to only one of the two when the other was absent.

LCM assumes that authors describe their submissions via one primary keyword and a variable number of secondary keywords; likewise reviewers characterize their area(s) of expertise.[6] Let $\Psi = \{\psi_1, \cdots \psi_{N_\psi}\}$ be the set of subject areas. LCM represents each paper p_i using two vectors based on their associated subject areas: a primary vector $\mathbf{i}_p \in \{0,1\}^{N_\psi}$ and secondary vector $\mathbf{i}_s \in \{0,1\}^{N_\psi}$. $\mathbf{i}_p$ is a one-hot vector that represents the primary subject area and $\mathbf{i}_s$ is a multi-hot vector that represents the secondary subject areas. LCM represents each reviewer r_j using a vector $\mathbf{j} \in [0,1]^{N_\psi}$. LCM sets the dimensions corresponding to the reviewer's primary subject area to 1 and the reviewer's secondary subject areas to 0.5. Constructing a sparse vector representation using just the primary and secondary areas will fail to capture reviewers' expertise in semantically similar areas that were not explicitly mentioned. To alleviate this problem, LCM converts the sparse $\mathbf{j}$ vector to a dense vector by inferring the reviewer's expertise in subject areas other than primary and secondary.

Let $\psi' \in \Psi$ be a subject area which is neither in reviewer's primary or secondary areas. LCM first identifies multiple sources that can help infer the expertise value $j_{\psi'} \in [0,1]$ for ψ'. It then infers an expertise value using each source one at a time and aggregates them. We now discuss each source and how LCM infers the expertise value using them.

Co-occurrence: Given that a reviewer is an expert in ψ, LCM computes conditional expertise in ψ' as $P_e(\psi'|\psi) = n(\psi,\psi')/n(\psi)$, where $n(\psi,\psi')$ is the number of reviewers and papers with both ψ and ψ' in their subject areas and $n(\psi)$ is the number of reviewers and papers with ψ in their subject area. The intuition behind this approximation is twofold: (1) P_e uses the current trend to capture correlation among the subject areas and (2) the asymmetry in P_e reflects the fact that expertise in a specialized area can indicate expertise in corresponding generic areas but not vice-versa. Given a reviewer's subject area ψ, the expertise value of ψ' is $\rho * P_e(\psi'|\psi)$. LCM sets ρ to 1 when ψ is a primary area and 0.5 when ψ is a secondary area. It considers the primary and the secondary areas as different sources and computes an expertise value for each source separately.

Paper Subject Areas: The subject areas of papers submitted by j (if any) that are not already in the reviewer's subject areas are set to a value (0.2) lower than the one used for secondary areas. This helps capture expertise of reviewers in areas they may have just started exploring.

Common Parent: Each subject area ψ can be expressed as $parent(\psi) \rightarrow child(\psi)$. For example, in *NLP → Dialog Systems*, *NLP* is the parent and *Dialog Systems* is the child. Each parent can have multiple children. Given a reviewer's subject area ψ, LCM assigns a score of $\rho * 0.4$ to ψ' if $parent(\psi')$ is same as $parent(\psi)$ and 0 otherwise. ρ is same as the one used for co-occurrence computation.

For each subject area ψ', LCM computes the expertise value from different sources, aggregates them using the max operator and assigns it to $j_{\psi'}$. The max operator helps pick the source that best models ψ'. Finally, LCM computes the *Subject Area Matching (SAM)* score using sparse paper vectors ($\mathbf{i}_p$ and $\mathbf{i}_s$) for paper p_i and dense reviewer vector $\mathbf{j}$ for

[6] Of course, this can easily be adjusted in the LCM code, but doing so would require the logic that follows to be adapted accordingly.

reviewer r_j as $\mathrm{SAM}(i,j) = \frac{1}{Z}\left(\mathbf{i}_p \cdot \mathbf{j} + \lambda \cdot \mathit{Sorted}(\mathbf{i}_s \circ \mathbf{j})\right)$, where the operator $\circ$ indicates Hadamard product, the function $\mathit{Sorted}(\mathbf{x})$ sorts the elements in $\mathbf{x}$ in decreasing order and $\lambda = [0.5, 0.5^2, \cdots, 0.5^{N_\psi}]$. The normalizer $Z = 1 + \sum_{m=1}^{M} 0.5^m$, where M is the number of non-zero values in $\mathbf{i}_s$. The scoring function ensures (1) paper's primary area contributes more than all the secondary areas, (2) the more the number of secondary areas match, the higher the score, and (3) papers do not get penalized for providing fewer (or zero) secondary areas.

Some intuitive properties of SAM scores are: it considers the primary keyword to be more important than all of the secondary keywords; gives each individual keyword less weight when more keywords are listed; and imputes additional keywords (with lower weights) for reviewers based on keyword co-occurrence patterns across the whole conference.

2.3.2. Normalizing Scores

LCM requires all final scores to lie in the range [0, 1] so that it can use consistent units to set the parameters that penalize violations of soft constraints (e.g., we might penalize violation of a soft constraint by 0.1 units, meaning that we would accept a match that scored 0.09 units worse in order to preserve the constraint, but not a match that scored 0.11 units worse). The three component scores (TPMS, ACL and SAM) are computed independently, and since they can differ enormously in their numerical ranges, we recommend first normalizing them each individually before taking their weighted average as the final matching score. We achieved this for AAAI 2021 via a procedure that should be applicable to most conferences. For each score, we independently and randomly selected many (paper, reviewer) pairs. A Program Chair manually examined each pair (reading the paper's abstract and the reviewer's profiles) and annotated each pair (p_i, r_j) as $y_{ij} \in \{1, 0.75, 0.5, 0\}$, corresponding to *excellent*, *good*, *in-a-pinch*, and *bad* matches respectively. We then performed a linear regression mapping the raw score component to y_{ij} and clipped the resulting function to the range [0, 1].

2.3.3. Aggregating Scores

The next task performed by LCM is to aggregate scores into a single number between 0 and 1, in a way that is robust to missing TPMS and ACL scores. Although these weights can be changed, we propose to give equal weight to keyword-based scores and to the set of all automatically inferred scores (TPMS and ACL), since keyword scores depend on information provided explicitly by authors and reviewers. When both TPMS and ACL are present, LCM thus computes TPMS + ACL + 2 * SAM and renormalizes; when only one of TPMS and ACL is present, LCM sums the available score and SAM score and renormalizes; when only one score is present, we use it as the base aggregated score:

$$\text{Base Aggregated Score} = \begin{cases} \frac{1}{4}\text{TPMS} + \frac{1}{4}\text{ACL} + \frac{1}{2}\text{SAM} & \text{if all data is present} \\ \frac{1}{2}\text{ACL} + \frac{1}{2}\text{SAM} & \text{if TPMS is missing} \\ \frac{1}{2}\text{TPMS} + \frac{1}{2}\text{SAM} & \text{if ACL is missing} \\ \text{SAM} & \text{if both ACL and TPMS are missing} \end{cases}$$

2.3.4. Accounting for bids

Given this quantification of reviewer *expertise*, conference organizers still need to account for reviewer *interest* as expressed through bidding to obtain a final result we call *aggscore*. LCM computes aggscore by upweighting/downweighting the base aggregated score based on the bid. LCM's approach to combine expertise and interests does not allow bids to override the base aggregated scores. It achieves this by computing aggscore = (Base Aggregated Score)$^{\text{bidscore}}$. Note that this maintains scores in the range [0, 1]. Using exponentiation (as compared e.g., to multiplying by a bid-specific constant, as in AAAI 2020, or to adding a constant and renormalizing) means that bids influence aggscore more with increasing reviewer expertise; reviewers can not influence the system to give them papers for which they are manifestly unqualified. On the other hand, the presence or absence of a bid can make a significant difference to the scores of sufficiently qualified reviewers. LCM's default values (used at AAAI 2021) map bids onto bidscores as $20, 1, 0.67, 0.4, 0.25$ corresponding to bids of *not-willing, not-entered, in-a-pinch, willing, and eager*, respectively. An exponent of 20 for *not-willing* makes the reviewer–paper match almost impossible unless it was very highly scored and no alternatives exist. An exponent of 1 for papers that received no explicit bid leaves those scores unchanged. Exponents less than 1 (for *in-a-pinch, willing,* and *eager*) boost scores; we chose these values based on spot checking potential matches at AAAI.

2.3.5. Cleaning up low-scoring matches

The final step in generating scores in LCM is to clean up low-quality scores as much as possible to avoid the possibility of highly inappropriate matches. In AAAI 2021, through spotchecking some of the scores, we observed that while high ACL or TPMS scores indicated good matches, low values ($s_{ij} < 0.15$) carried much less signal than low SAM scores. Hence, when aggscore was less than 0.15, we used the minimum of 0.15 and the SAM score exponentiated by the reviewer's bid. This transformation ensured that whenever a low-scoring match was selected, it was at least from the same subject area. This avoided the possibility of wildly inappropriate matches and helped reviewers see some rationale for our decisions to assign poorly matching papers.

3. Assigning Reviews

LCM formulates the problem of maximizing its scoring function subject to a set of both hard and soft constraints as a Mixed-Integer Programming (MIP) problem. It is quite natural to formulate the reviewer–paper matching problem as one of maximizing an aggregate weighted matching score subject to reviewer and paper capacity constraints and avoiding COIs, and much past work has done so [Taylor, 2008, Flach et al., 2010, Garg et al., 2010, Tang et al., 2010, Charlin et al., 2011, Charlin and Zemel, 2013, Lian et al., 2018, Kobren et al., 2019]. Indeed, with only COI and capacity constraints, this formulation corresponds to a network flow problem and is thus solvable in polynomial time [Taylor, 2008]. Unfortunately, that formulation can yield an inequitable distribution of work across reviewers and of scores across papers. To address this, Stelmakh et al. [2019] and Kobren et al. [2019] focused on finding more balanced matchings. Specifically, instead of the common objective of maximizing the aggregate weighted matching score, Stelmakh et al. [2019] look into finding matches that maximize the minimum matching score across all papers. Because the resulting optimization problem is NP-hard, they design an approximation algorithm, which finds a suboptimal solution by solving a series of network flow problems. Like us, Kobren et al. [2019] aim to maximize an aggregate weighted matching score, but took a somewhat different approach. Specifically, they augmented an Integer Linear Program with lower and upper bounds on the assignments per reviewer and also added local fairness constraints to ensure that the total matching score for each paper would exceed a threshold; they propose an approximation algorithm that relaxes integrality constraints to solve their assignment problem. LCM uses slack variables to soften hard constraints instead of relaxing the integer constraints, resulting in a mixed-integer program. It also introduces a wide range of additional constraints. We now describe this approach in more detail. For a formal specification of the LCM's mixed-integer program, please see Appendix A.

3.1. Formulating the Review Assignment Problem

It would not suffice simply to identify a score-maximizing matching: the best reviewers could be given hundreds of papers and different papers could receive very different treatment, leading to an unfair assignment. For example, a paper could be assigned all junior reviewers from one research group. We thus imposed various *constraints* on reviewer–paper matchings.

Constraint 1 (COI). *No reviewer should be assigned to a paper with which they are in conflict.*

Constraint 2 (Number of Reviewers). *Each paper must receive a specified number of reviews.*

This number of reviews is very often three, as it has historically been at AAAI and many other AI conferences. In LCM's two-phase reviewing system (see Section 4) the reviewer–paper matching problem is solved separately in each of the phases, assigning two reviewers for each paper in Phase 1, and adding additional reviewers in Phase 2 such that each paper not rejected in Phase 1 got at least four reviews across the two phases. (That is, LCM adds extra reviewers in Phase 2 when reviewers did not submit their reviews at the end of Phase 1.)

Constraint 3 (Reviewer Load). *The number of papers assigned to any reviewer should not exceed some maximum.*

LCM has a hard limit of three papers per reviewer per phase, and (as we will discuss later) penalizes matchings that assign more than two papers per reviewer in a given phase.

Constraint 4 (Seniority). *Each paper is assigned at least one experienced reviewer.*

Senior reviewers often have busy schedules, making their reviews a scarce resource, but their years of experience in the field can lead to particularly reliable feedback to authors and also contribute to the development of novice reviewers.

The constraint is thus motivated by the following goals: (1) to ensure that experienced reviewers are distributed fairly across papers; (2) to reduce variance in reviews; and (3) to ensure that each discussion has at least one experienced participant.

A second set of constraints serves both to ensure that each paper is reviewed by an intellectually diverse set of reviewers and to make collusion rings harder to sustain. Collusion is only possible amongst reviewers who can somehow communicate with one another. By preventing pairs of co-authors from reviewing the same papers, preventing arbitrary pairs of reviewers from bidding to review each other's papers, and by choosing reviewers from as diverse a set of geographic regions as possible, LCM limits opportunities for gaming the system.

Constraint 5 (Coauthorship Distance)**.** *No two reviewers assigned to the same paper have small distance in the coauthorship graph.*

Constraint 6 (Geographic Diversity)**.** *No two reviewers assigned to the same paper belong to the same geographic region.*

Constraint 7 (No 2-Cycles)**.** *No pair of reviewers who both bid positively on each other's papers may review each other's submissions.*

It would not be desirable to satisfy this full set of constraints at all costs. In AAAI 2021, it was easy to find qualified reviewers satisfying all of these constraints for many papers; however, for others imposing all of these constraints would dramatically have degraded reviewer quality or even made it impossible to find the required number of reviewers. Furthermore, these constraints are often overkill: e.g., qualified reviewers often bid on each other's papers without harboring malicious intent. Out of caution, LCM adopts the principle of avoiding such matchings when reasonable alternatives exist but allowing them otherwise, expressing all but *COI*, *Number of Reviewers*, and PC's *Reviewer Load* as soft constraints. More specifically, for each constraint, it identifies a constant expressing its importance and penalizes the objective function by this constant each time the constraint is violated for a reviewer–paper pair. Observe that the task of identifying these constants is dramatically simplified by the fact that the scoring function is normalized between 0 (worst possible reviewer–paper matching) and 1 (best possible matching). For example, expressing a coauthorship distance penalty of 0.1 would mean that we would prefer to accept a reviewer–paper matching scoring up to 0.1 points lower in exchange for satisfying the constraint. The full mixed-integer programming formulation of our solution to the review assignment problem is given in Appendix A.

3.2. Solving the Review Assignment Problem

Let x_{ij} be binary variables that denote whether paper i is assigned to reviewer r_j. The formulation just described instantiates a variable x_{ij} for each possible (reviewer j, paper i) combination (in addition to a variety of associated slack variables and a huge number of constraints). This can easily lead to a MIP too large to store in memory, let alone to solve. In such cases, LCM supports the application of column (variable) generation and row (constraint) generation.

3.2.1. Generation of Assignment Variables (Columns)

LCM only creates x_{ij} variables that are likely to be relevant (thereby restricting the space of possible assignments; implicitly, non-generated variables are set to 0). There is little point in considering variables that correspond to (i, j) pairs with low matching scores, as these would represent poor matches. In rare cases, this may result in papers receiving fewer than the desired number of reviews. When the MIP is infeasible in this way, LCM can iterate (perform column generation) by generating more matching variables for the corresponding papers. At AAAI 2021, we assigned matches to such papers manually, as the algorithm is unlikely to do much better than a random assignment when given only low-scoring alternatives. We did not create variables for any reviewer–paper pair (i, j) with $s_{ij} < 0.15$.[7]

LCM creates variables as follows: for each paper, it creates a variable for the k highest scoring PC, SPC, and AC reviewers. Similarly, for each PC/SPC/AC member, it creates a variable (if it did not exist already) for their highest scoring $k/5k/10k$ papers. k can be iteratively expanded as time allows (including in targeted ways, such as increasing k for underutilized reviewers) until the dense matrix is recovered. In AAAI 2021, LCM's MIP was computationally feasible at high enough k that further increases to k offered only negligible improvements, so we did not leverage LCM's ability to perform column generation iteratively.

[7] We note that this suggests a refinement to the bidding procedure: it is only necessary to show reviewers papers for which a sufficiently positive bid could bring their score above the threshold. This could result in large time savings for reviewers in the bidding phase, as it can be hard to sift through an entire conference worth of papers.

3.2.2. Generation of Coauthor Constraints (Rows)

Even given a reduced k, there still may be too many coauthor constraints to include (the number scales with the product of the number of (reviewer, paper) combinations and the number of reviewers). Instead, LCM begins by solving the problem with no coauthor constraints. After reaching an initial solution, it identifies violated coauthor constraints and adds them explicitly to the MIP. LCM repeats this process until it is satisfied with the number of violated constraints. At AAAI 2021 the number of violations plateaued after eight iterations. Pseudocode is given as Algorithm 1.

Algorithm 1: Assignment Variable Sparsification and Constraint Generation

```
Let $V_{\text{master}}$ contain all coauthor constraints
Let $\mathbf{X}_k$ be the set of $x_{ij}$ variables obtained by sampling a constant multiple of the top $k$ papers for every reviewer and top $k$ reviewers for every paper
$i \leftarrow 0$
$V_i \leftarrow \{\}$ {Store of violated coauthor constraints up to iteration $i$, initially empty}
while True do
    Solve the MIP defined by $\mathbf{X}_k$ and $V_i$. Store the solution in $\mathbf{X}^*$
    Identify all new coauthor violations $V_{\text{new}}$ in $\mathbf{X}^*$ using $V_{\text{master}}$
    if $V_{\text{new}}$ is empty or time has run out then
        Return $\mathbf{X}^*$
    end if
    $V_{i+1} \leftarrow V_i \cup V_{\text{new}}$
    $i \leftarrow i + 1$
end while
```

Lastly, to reduce computation time, LCM does not attempt to find an optimal solution to the MIP. Instead, it sets an absolute MIP-gap tolerance of 20. Since aggscore is on a scale between 0 and 1, in a large conference where on the order of tens of thousands of assignments must be made, being suboptimal by at most 20 objective function units is quite benign. In AAAI 2021, the result was that we found a very nearly optimal solution much more quickly.

4. Two-Phase Matching

The quality of papers submitted to large conferences varies vastly. An influential experiment by organizers of the 2014 NeurIPS conference [Lawrence and Cortes, 2014] split reviewing between two independent program committees and had them both review 10% of submissions. It found that the strongest papers (a small set) and the weakest papers (a much larger set) could be reliably identified, but that many papers close to the decision boundary were accepted by one program committee and rejected by the other. In the years since, conference organizers have sought to reallocate reviewing resources from papers that are nearly certain to be rejected to papers that have a realistic chance of acceptance, to improve review quality for the latter group. A popular approach is to employ simple heuristics designed to detect low-quality papers, a process known as “summary rejection” or “desk rejection”. For example, in IJCAI 2020, Area Chairs were asked to spend a short amount of time skimming each paper to decide whether it deserved a more careful review [Bessiere, 2020]. Neurips 2020 employed a similar system: over three weeks, Area Chairs skimmed over 9,000 papers to identify obvious rejects and senior Area Chairs cross-checked these choices; 11% of submissions received summary rejects [Yuan, 2020]. Given the size of AI conferences, such summary rejection procedures are time consuming for Area Chairs. Furthermore, they are likely to be noisy and also may reflect unconscious biases against superficial properties of a paper, meaning that they may not be reliable enough to eliminate more than a relatively small fraction of papers. Finally, they tend to be unpopular with authors, who dislike having their paper rejected via an opaque process that produces no reviews. For these reasons, NeurIPS decided not to employ such an approach in 2021 on the basis of negative feedback received in 2020.

LCM supports an alternative early-rejection approach, pioneered at AAAI 2021, that simultaneously concentrates the conference’s reviewing budget on papers close to the decision boundary; provides meaningful feedback to authors of early-rejected papers; and early-rejects few papers that could ultimately have been accepted (see our evaluation below). The key idea is to break reviewing into two phases. In Phase 1 each paper is assigned 2 reviewers. Papers that receive two sufficiently-high-confidence reviews recommending rejection are rejected at this stage, with the authors immediately receiving these full reviews and being offered no opportunity for rebuttal. The process then proceeds to Phase 2, where two or more additional reviewers are assigned to each of the papers that remain. After the second round of reviews, rebuttals are solicited from authors and reviewers from both phases are asked to read rebuttals and each other's reviews, to engage in a discussion mediated by SPCs and ACs, and ultimately to revise their reviews accordingly. Program Chairs then make decisions based on recommendations from ACs.

The key rationale for this system is that, given the low acceptance rates at large conferences, papers for which the first two reviews are both confident and negative have a very low chance of eventual acceptance. Because reviews are much more careful than quick perusals by ACs, this system is also able to reject a larger fraction of papers than the summary rejection approaches previously employed by IJCAI, NeurIPS, and other conferences. The resulting reviewing resources can be devoted to papers with a larger chance of acceptance, reducing variance in these recommendations. In the experimental section below, we offer evidence both that AAAI 2021 papers rejected in Phase 1 were indeed likely to be rejected if given more reviews, and that additional reviews performed in Phase 2 indeed reduced variance in confidence-weighted average scores.

It is also worth mentioning three additional benefits of the Two-Phase approach. First, inevitably some reviewers fail to complete their assignments, others report low confidence, and still others point out a need for additional reviews by specialists in particular topics. When this occurs in Phase 1, new reviewers can be found without needing to resort to ad hoc, manual approaches that scale badly to large conferences. Second, the existence of a second reviewer–paper matching period at the beginning of Phase 2 makes it possible to fast-track high-quality rejected submissions from another conference, using their previous reviews and some numerical constraint on their average scores to take the place of the reviews that would have been obtained in Phase 1. We deployed such a "fast-track" in AAAI 2021 for papers that received NeurIPS and EMNLP reviews that scored them roughly in the top half of submissions. Third, many authors appreciate receiving rejection notifications quickly so that they can resubmit their work to another venue without waiting for the whole review process to conclude.

Finally, we note some of the drawbacks of Two-Phase reviewing: reviewers have to be chased twice, increasing workload for Program Chairs, ACs, and SPCs; and authors rejected in Phase 1 have no opportunity to rebut reviews that they consider erroneous. Moreover, running a two-phase process has the potential to lengthen the overall review process, although we note that this was not a major consequence of the change at AAAI; as compared to previous years, reviewers were given roughly half the amount of time to review roughly half as many papers in each of two phases.

While we designed LCM's Two-Phase reviewing system independently, we have since learned that versions of the idea were previously implemented at other CS conferences. To the best of our knowledge, the first conference that split reviewing into two phases was ICML 2009 [Bottou and Littman, 2009]. According to the program chairs, the motivation was not increasing the number of reviews assigned to strong papers, as it was for us, but rather a concern that papers would be rejected by reviewers who did not fully understand them, due to exponential growth in submissions that had led to a pool of papers larger than the pool of trusted reviewers. ICML09's system differed from ours in at least two key ways: in Phase 1, authors in ICML 2009 were asked to indicate a preference for an Area Chair who would oversee their paper in Phase 2 and help to choose reviewers; papers were allowed to submit a rebuttal in response to Phase 1 reviews. Two-Phase reviewing was discontinued at ICML in 2011, apparently because the highly manual process became both logistically unwieldy and excessively time consuming as the conference continued to grow.

We are aware of two other (non-AI) conferences that also employed Two-Phase reviewing: ASPLOS, and OOPSLA [Adve and Hoe, 2010, Lopes, 2014]. Notably, in OOPSLA13 [Lopes, 2014], authors of accepted Phase 1 papers were asked to revise their papers based on Phase 1 reviews; Phase 2 reviews then assessed the revised versions. ASPLOS21 [Kozyrakis and Berger, 2021] required authors to submit a two-page extended abstract in Phase 1; about 55% of the papers were then promoted to Phase 2 and asked to submit a full paper. In addition to these conferences, Boehm et al. [2021] describe a Two-Phase reviewing process for PLDI conferences, in which Phase 1 decisions were made based on at least three reviews and authors of the papers that were candidates for Phase 1 rejection were invited to submit author responses before a final decision ass made for their papers. The author response period for the rest of the papers happened at the end of Phase 2.

AAAI 2021's successful deployment of Two-Phase reviewing has already begun to show some impact of its own. On the practical side, our Two-Phase design was maintained by AAAI 2022-2024 and was newly adopted by IJCAI 2022 and 2023, ICML 2022, and ACM-EC 2023. The design has also received academic study. A recent paper by Jecmen et al. [2021] investigated how reviewers should be divided across the two phases to maximize the overall matching score in a conference, showing that the problem is NP-hard and empirically verifying that a simplified *random-split* strategy of randomly reserving reviewers for Phase 2 gives a near-optimal assignment on real conference data.

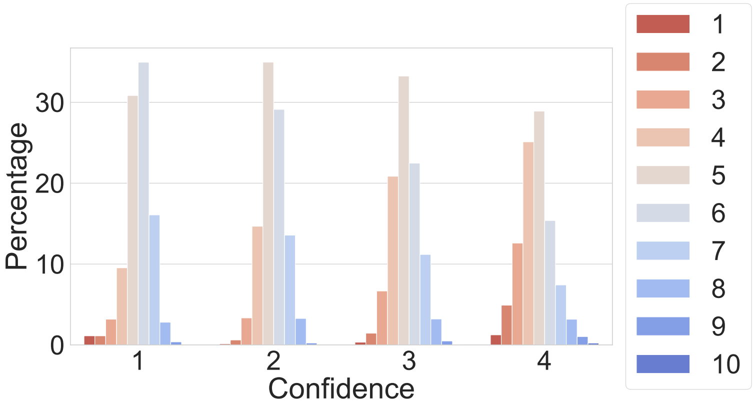

Figure 2: Paper score distribution by confidence of reviewer. More confident reviewers gave fewer 5s and 6s.

5. Evaluations on AAAI 2021 Data

We deployed LCM at AAAI 2021. In this section, we analyze a dataset of 6,729 submissions,[8] the specific reviewer matching we obtained for them, and the resulting paper reviews from 8,072 reviewers in the main conference's PC.[9] You can find the complete list of parameter choices we used to obtain the AAAI 2021 matching in Appendix A.2.

5.1. Analysis of Match Quality

We begin by asking whether LCM made good matches in the real conference. Of course, evaluating match quality is conceptually tricky—if ground truth data existed, we would have used it to improve the matching! The gold standard is to ask Area Chairs to manually ensure that matchings make sense; it is also possible to monitor social media for outrage [Lawrence, 2014]. Our focus here was to go beyond such subjective measures, performing a data-driven and quantitative evaluation along as many dimensions as possible. What follows is, to our knowledge, the most comprehensive, publicly available post-hoc evaluation of a Computer Science conference's match quality. Indeed, we note that such analyses are usually limited only to the graphs that appear in Program Chairs' presentations at conference business meetings. Of course, the confidentiality of reviewing data typically prevents others from conducting further, in-depth analyses. In this vein, we regret that we are similarly unable to release any of the data underlying the analyses in this paper.

5.1.1. Did LCM's scoring function capture reviewer expertise?

First, we investigated whether reviewers actually bid on papers for which we assigned them high "expertise" scores, leveraging the fact that reviewers tend to bid positively on papers that they are capable of reviewing. Considering the set of reviewers who submitted at least one bid, we looked at all matched reviewer–paper pairs that we assigned base scores (i.e., aggscores before taking bids into account) greater than 0.75. We found that 29% of these pairs were accompanied by *eager* bids, 38% *willing*, and 3% *in a pinch* (the remainder no bids). We see this as evidence that high base scores indeed correlated with high match quality. However, the bidding interface is a possible confounding factor here—e.g., users could view papers in descending order of their TPMS scores (one of the inputs to aggscore). We also note that in the future it would be even better to allow users to sort by aggscore.

Second, we observe that the distribution of overall scores given by reviewers to papers (which ranged between 1 and 10) noticeably changed as a function of self-reported confidence.[10] Figure 2 shows that more confident reviewers tended to give more informative paper scores (fewer borderline 5 and 6 scores). Given that higher confidence reviews were more informative, we examined the relationship between our scoring function and reported reviewer confidence.

[8] For this analysis we removed papers that were withdrawn, desk rejected, or rejected as dual submissions. We also excluded all papers from the AI for Social Impact track as these papers were matched via a somewhat different process. We included papers that were only submitted in the second phase via the fast track.

[9] In AAAI 2021, Program Chairs, ACs, and SPCs manually assigned about 4.5% of the final reviewer assignment. In fact, many of these were suggested by area chairs in Phase 2 based on the reviews in Phase 1. In our evaluations of match quality, we only considered matches that were automatically assigned to avoid confounding factors of manual matches.

[10] Confidence levels could be reported as "Reviewer made an educated guess", "Reviewer is broadly knowledgeable, but not an expert in the area", "Reviewer is knowledgeable in the area of the paper", or "Reviewer is an expert in the specific topic of the paper".

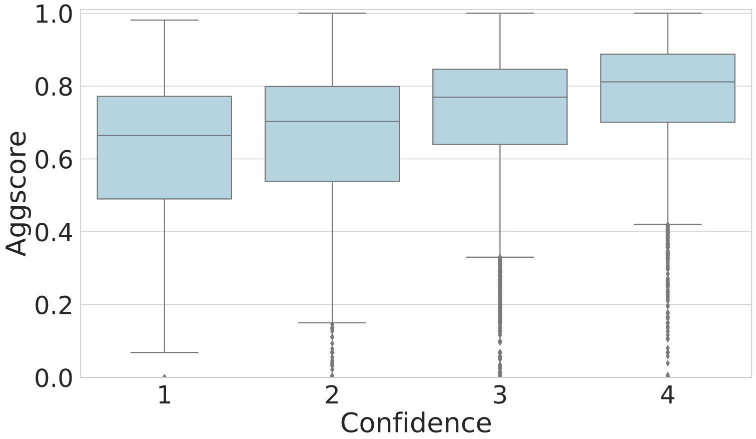

Figure 3: Reported confidence of an eventual review vs. predicted aggscore for all reviewer–paper pairs matched at AAAI 2021.

Figure 3 shows that our scoring function was positively correlated with confidence: as confidence increased, so did the 25th, 50th, and 75th percentile of scores.

5.1.2. *Assuming that LCM's scoring function predicted review quality, how close did it come on average to assigning the best possible reviewer to each paper?*

We considered the ordinal ranking of reviewers assigned to each paper in terms of aggscore, where rank 1 corresponds to the reviewer with the highest aggscore for that paper. We then looked at the mean reviewer ranking for each paper with respect to the ranks of all of the paper's reviewers. The first / second / third quartiles of mean reviewer rankings across all papers were 7 / 12 / 28. This shows that papers were often assigned very highly qualified reviewers; note that these averages are taken across all of a paper's reviewers (median 4) and that a single reviewer with a poor rank is able to cause the average ranking to grow large. For example, a paper which received its top four reviewers would receive an average rank of 2.5; a paper that was assigned reviewers ranking 1, 5, 13, 29 would receive an average rank of 12. In comparison, the median paper had 1446 "qualified" reviewers ($s_{ij} \geq 0.15$).

5.1.3. *Was LCM's aggscore computation stable under missing data?*

Recall that LCM's aggscore is a combination of three constituent scores: TPMS, ACL, and SAM (see Section 2.3.3). At AAAI 2021 these constituent scores were available for 60.25%, 96.64%, and 100% of papers respectively. Here we evaluate the stability of our aggscores in cases where ACL and TPMS scores were missing. We began with all "qualified" (paper, reviewer) pairs ($s_{ij} \geq 0.15$) for which all of the three scores were available. We then computed two proxy aggregate scores, representing the scores we would have computed if either of the TPMS or ACL scores had been missing.

The standard deviation of the changes in aggscores caused by missing data were 0.049 and 0.048 respectively, whereas the mean standard deviation of the complete-data score for a given paper was 0.127. Since the variance of the differences between the hidden data aggscores and the full-data aggscores was much smaller than the average variance in aggscores across reviewers for a paper, the aggscore was still likely to have made a similar assessment of reviewer–paper pairings even in the presence of missing data.

5.1.4. *How many conflicts were identified by LCM's COI detection?*

We consider self-reported conflict domains, explicitly stated conflicts, and conflicts between coauthors of papers submitted to AAAI 2021 to be 'trivial conflicts'. Out of the total $2,674,372$ conflicts that LCM found at AAAI 2021, 96.4% were trivial. Of the remaining 3.6% conflicts, 2.8% were due to unreported coauthorship relationships we verified via DBLP. The remaining 0.8% of conflicts were between predicted student–supervisor pairs or predicted students of the same supervisor. Overall, LCM added at least one nontrivial conflict to a large majority (78.8%) of submissions.

5.2. Analysis of MIP Solving

We now turn to an evaluation of LCM's matching algorithm (described in Section 3), including a comparison to the matching algorithm used at AAAI 2020, the previous year's conference. For our experiments, we created a full conference dataset containing all submitted AAAI 2021 papers and reviewers analyzed above. We augmented the dataset to include all additional reviewers who were not ultimately assigned any reviews at AAAI 2021, bringing the total number of reviewers up to 8,964. Except where otherwise noted, we evaluated LCM's algorithm in terms of the single-phase matching problem, as this allowed us to focus on matching quality and facilitates comparison with prior work. In the same spirit, we required each paper to be assigned four reviews ($\gamma_{\text{pc}} = 4$) and set a limit of five reviews per reviewer ($c_j = 5$). We ran row generation (as described in Section 3.2.2) for 10 iterations. We ran all of our experiments on a 16-core machine with Intel Silver 4216 Cascade Lake 2.1GHz processors and 96 GB RAM. For MIP solving, we used CPLEX version 12.9.0.0 with default parameters.

We note a subtlety that arises when comparing different assignments. LCM's MIP formulation uses an inequality instead of a hard constraint on the number of matches each paper receives, allowing it to assign fewer than γ_{pc} reviewers to a paper. This can occur for papers for which there are no qualified reviewers or only very poor matches conditional on the rest of the assignment. A consequence for experimental analysis is that we can find ourselves needing to make comparisons across assignments in which different numbers of reviewer–paper matches exist. In these cases, we padded all matchings with reviewer–paper matches that contributed nothing to the objective function, as though we had matched the paper to a reviewer having an aggscore of 0 who imposed neither rewards or penalties via soft constraints. Such zero-contribution matches accounted for at most 1% of the total reviews in any of our matchings reported below. While in reality we addressed all such cases at AAAI 2021 manually, we note that many AAAI 2021 reviewers were ultimately assigned zero papers because they either did not have large enough aggscores or had too many conflicts, meaning that a large pool of available and unused reviewers did exist.

5.2.1. *How did LCM's matching algorithm compare to the algorithm used in the previous iteration of the conference?*

We compared LCM's matching algorithm to the method used in the conference's previous iteration (AAAI 2020), a flow-based matching algorithm. Rather than simple capacity constraints, each paper and reviewer had a number of weighted *slots* that linearly scaled the contribution of each assignment to prioritize each paper getting some good reviewers and each reviewer getting some good papers. Based on advice from the previous Program Chairs, we set the paper slot weights to be 8/4/1/1 and reviewer slot weights to be 8/2/1/1/1. Their reviewer–paper scoring function differed from LCM's in two main ways: (1) it did not use ACL scores, and (2) bids scaled scores multiplicatively rather than exponentially. Beyond the scoring function, their matching algorithm did not attempt to satisfy any soft constraints.

Nevertheless, we investigated how effective last year's approach was at limiting soft constraint violations despite not optimizing for them (i.e., we tested how frequently these violations occurred organically at the optimum of a simpler matching problem). We ran both algorithms and evaluated the number of soft constraint violations compared to LCM's approach. For reference, we also performed a similar experiment that compared LCM's algorithm to a version of the same algorithm that omitted all soft constraints. We summarize the results in Table 1, reporting the percent reduction in the number of: (a) papers without a senior reviewer (Constraint 4); (b) coauthor violations (Constraint 5); (c) papers with reviewers from the same region (Constraint 6); and (d) cycle violations (Constraint 7). We observed that AAAI 2020's approach led to substantially more violations of three of the four soft constraints. Most notably, LCM's algorithm assigned to the same paper 76.8% fewer pairs (118 compared to 509) of reviewers who were previous coauthors. Conversely, LCM's approach led to a 90% *increase* in the number of violations of our no-2-cycles constraint, prohibiting pairs of reviewers who bid positively on each other's submissions from being assigned to each other's papers. However, such matchings were quite rare overall (22 for AAAI 2020 and 42 for LCM's approach out of 26,916 reviewer–paper matchings in total). We believe that the AAAI 2020 method led to fewer 2-cycles because their scoring function weighs positive bids less heavily than LCM's does and therefore is less likely to match reviewers to papers where a reviewer bid positively. When using LCM's scoring function, we were able to reduce 2-cycle violations by 84% using LCM's soft constraint over the baseline version with no soft constraints.

LCM's algorithm had approximately the same memory footprint as the flow-based approach (96 GB vs 95 GB) despite the additional soft constraints that LCM considered: the flow-based method creates an edge for every reviewer–paper pair whereas LCM only created variables for a subset of qualified reviewers. In terms of runtime, running LCM's algorithm with 10 iterations of row generation required 52.0 wall-clock hours compared to 0.8 needed for the AAAI 2020 approach. Note that LCM's approach is anytime: it can terminate after any number of iterations of

Soft constraint	Baseline (no soft constraints)	AAAI 2020
Constraint 4 (Seniority)	56.1	9.5
Constraint 5 (Coauthorship Distance)	88.1	76.8
Constraint 6 (Geographic Diversity)	24.1	23.3
Constraint 7 (No 2-Cycles)	84.6	-90.9

Table 1
Percent reduction of soft constraint violations using LCM's approach relative to (1) LCM's approach without soft constraints and (2) last year's matching algorithm for a simulation of a single-phase version of AAAI 2021.

Soft constraint	% Aggscore reduction
Constraint 4 (Seniority)	0.05
Constraint 5 (Coauthorship Distance)	1.17
Constraint 6 (Geographic Diversity)	0.85
Constraint 7 (No 2-Cycles)	0.01
All constraints	2.04

Table 2
Percent reduction in aggscore when each soft constraint was turned on individually relative to a base MIP having no soft constraints. The final row displays the reduction when all soft constraints were present.

row generation. One iteration of LCM's algorithm took 0.9 wall-clock hours and five iterations took 7.3 wall-clock hours. We were willing to devote a few days to computation, so we find these numbers reasonable. Nevertheless, in what follows we identify some ways to reduce runtimes without sacrificing much in performance (e.g., 5× speedup via decomposing the matching into two stages; 1.7× speedup via reducing k to further sparsify the constraint matrix).

5.2.2. What did introducing soft constraints cost us in matching aggscore?

Table 2 shows that for the most part, with our specific settings of the soft constraint parameters, we were able to have our cake and eat it too. With all of LCM's soft constraints turned on, we only lost 2.04% mean aggscore. With each soft constraint turned on individually, the Coauthorship Distance and Geographic Diversity constraints reduced mean aggscore the most (1.17% and 0.85% respectively) while the Seniority and No 2-Cycles constraints reduced mean aggscore the least (0.05% and 0.01% respectively). The sum of the aggscore reductions across matchings where each soft constraint was turned on individually was 2.08% (sum of first four rows in Table 2), very similar to the 2.04% reduction in mean aggscore when all constraints were turned on simultaneously. This suggests that there was very little complementarity across different constraints (e.g., when we reduced Coauthorship Distance violations, we rarely also increased Geographic Diversity as a side effect).

5.2.3. How well did the row generation of coauthorship distance constraints perform?

Ideally we would evaluate the performance of the row generation of coauthorship distance constraints by comparing performance after each number of iterations to the performance of the optimal approach that held all rows of the constraint matrix in memory. Unfortunately, we were unable to compute the optimal objective value (OPT) for our dataset: this is why we needed row generation in the first place! Nevertheless, we can compare to an upper bound on OPT. Since each additional coauthorship constraint can only penalize the objective function, the optimal value of the objective function (without coauthorship penalty from ungenerated constraints) can only weakly decrease as these soft constraints are added. Therefore, the objective value we observe after the last iteration of row generation is an upper bound on OPT. After 10 iterations, we closed the gap between the initial solution before row generation (obtained with V_0 in Algorithm 1) and the upper bound on OPT by 77.5% and 1,308 units in the objective function.

5.2.4. How did the choice of our column generation parameter k impact the objective?

Recall that to reduce the size of the MIP, LCM creates variables only for the most-promising paper–reviewer pairs: for each reviewer, we sampled the k best-scoring papers, and for every paper, we sampled the k best-scoring reviewers.

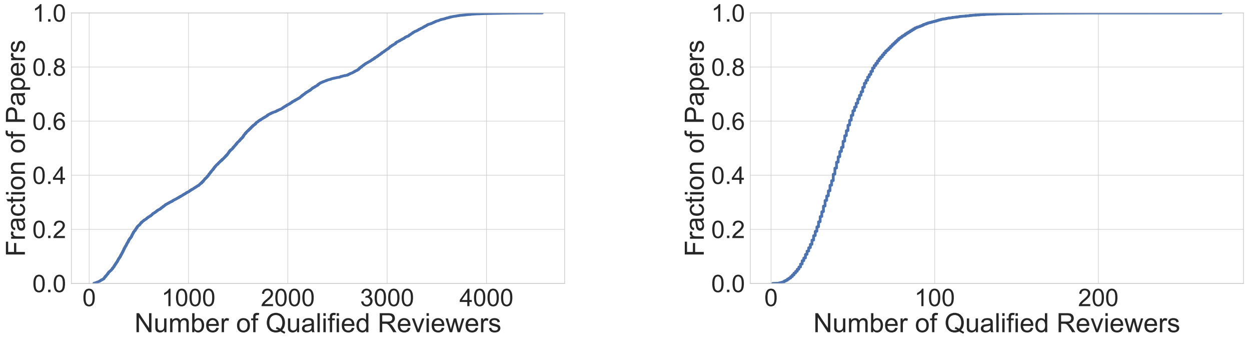

Figure 4: ECDF of the number of “qualified” reviewers for each paper (i.e., reviewers with $s_{ij} \geq 0.15$), considering both all reviewers (Left) and only reviewers who also made a positive bid for the paper (Right). The median paper had 1446 qualified reviewers and 43 qualified reviewers who also placed a positive bid.

Number of Phases	Objective Reduction	Speedup
2	0.02%	5.4×
4	0.33%	28.1×

Table 3
Sacrifice in objective function vs. computational savings of multi-phase conferences relative to single-phase conference.

We then filtered out any paper–reviewer pairs scoring below a threshold, keeping only qualified reviewers. We set $k = 50$ when creating the match for AAAI 2021. Figure 4 shows that after this filtering, the median paper had ≈1,500 qualified reviewers. Expanding to include variables for all possible qualified assignments (unbounded k) would have required much more memory than was available via our computing resources.

We investigated the impact of k on the quality of the resulting matching. Running LCM's algorithm with a much smaller k value ($k = 10$), we observed only a 1.8% loss in the objective but a 1.7 times speedup in performing 10 iterations of row generation, as compared to $k = 50$. Increasing k to 100 had a negligible impact on the objective (0.04%). We thus consider it unlikely that significant additional gains would have been achieved by increasing k further.

5.2.5. *What were the tradeoffs between objective value and computational savings with multiple phases?*

In a two-phase conference, about half of the total reviews are assigned in each phase, so the Phase 1 and Phase 2 MIPs have half as many free variables as the MIP for a corresponding single-phase conference. We should expect solving the two smaller problems to be much faster than solving the larger one, since the running time of MIP solvers tends to scale superlinearly with problem size. However, of course, the solution obtained by solving two phases optimally in succession will yield weakly lower objective value than the optimal solution to the single-phase problem. For example, imagine that each stage assigns two reviews per paper and that for some paper, there are only four qualified reviewers. The two-phase algorithm will not consider that it may need to save two qualified reviewers for the second phase, and may assign these highly specialized reviewers elsewhere, consuming their capacity. We thus empirically evaluated loss in objective value due to running a two-phase version of our conference. In these experiments, we assigned two reviewers per paper in Phase 1, fixed those assignments, and then assigned two more reviewers per paper in Phase 2. We found that the resulting matching had objective value only 0.02% lower, as compared to the single-phase conference. On the other hand, decomposing the problem in this way led to a 5.4× speedup.

It occurred to us that this approach of incrementally building up a MIP solution reviewer by reviewer did not require that the conference actually did perform two-phase reviewing. For example, a single-phase conference could use a similar k-phase strategy, when adding k reviewers per paper simply to speed up MIP solving, which could make a significant computational difference for conferences significantly larger than AAAI 2021. To investigate this idea, we evaluated a four-phase approach that assigned one reviewer per paper, fixed those assignments, and repeated four times until each paper had four reviewers. As before, we observed very encouraging results: the objective value decreased by only 0.33% relative to a single-phase conference, whereas MIP solving accelerated by 28.1× (see Table 3)!

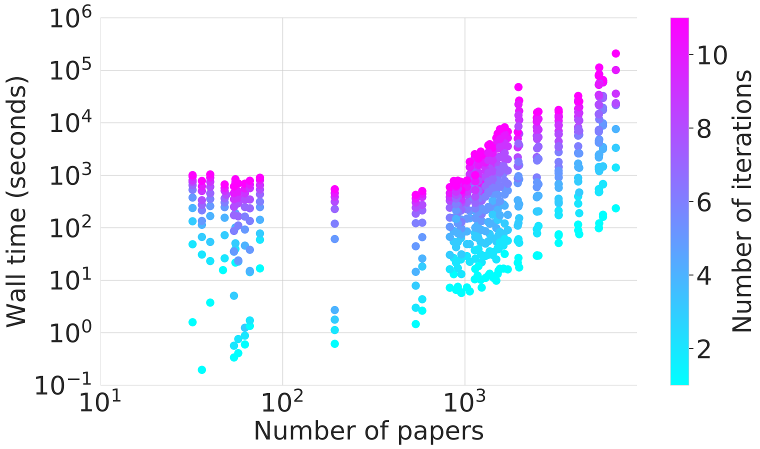

Figure 5: Running time vs. conference size (number of papers); points represent (conference, iteration) pairs.

5.2.6. *How did LCM scale?*

To understand how LCM's performance depended on the size of the conference, we generated 77 synthetic conferences of different sizes by subsampling papers and reviewers (without replacement) from our dataset. In an attempt to maximize the realism of the structure of our simulated conferences, we constructed them based on related keywords. At AAAI 2021, keywords were divided into top-level categories (e.g., Machine Learning) and bottom-level categories (e.g., Learning Theory); papers and reviewers could select one top-level and multiple bottom-level keywords.

Our generator works as follows. We begin with an initial bottom-level keyword and then initialize our conference to contain all papers associated with that keyword and a set of reviewers associated with the corresponding top-level keyword proportional to the fraction of papers belonging to the bottom-level keyword within that top-level area. We store a set of keywords associated with our conference as the generator runs. In each iteration, we add to our set of keywords the keyword belonging to most papers in our current set of papers that is not yet part of our tracked set of keywords. Papers corresponding to this keyword and additional reviewers are then added accordingly. Every time the number of papers grew by a factor of 1.3, we created a new synthetic conference containing the current set of papers and reviewers and added it to our set of samples. For diversity, we ran our generator on three trajectories, each seeded with a different keyword: one from game theory, one from computer vision, and one from constraint satisfaction, because each of these areas involves a substantially distinct set of reviewers and authors and each has its own dedicated conferences. We also included the full AAAI 2021 dataset as our final datapoint, giving us a total of 78 simulated conferences.

We then turned to studying how LCM's row generation approach scaled. Figure 5 shows cumulative wall-clock time for each (conference, iteration) point, with points coloured by iteration. For conferences with fewer than 1,000 papers, we could always complete 10 iterations in less than an hour. Interestingly, the smallest conferences were harder to optimize than medium-sized conferences. We speculate that this occurred because some soft constraints (e.g., coauthor / cycle constraints) were harder to satisfy when the set of papers and reviewers were more densely connected. Running time appeared to scale exponentially as the number of papers grew beyond 1,000, with the biggest size taking a few hours per iteration.

5.2.7. *How did LCM's results on optimality and aggscore reduction translate to our generated conferences?*

Figure 6 shows ECDFs of how much of the gap we closed between the initial solution before row generation and the upper bound on OPT described earlier in Section 5.2.3. After just one iteration of row generation, we closed at least 25% of the gap for 80% of the generated conferences. After 10 iterations, we closed 75% of the gap for over 90% of conferences. The trend in which these ECDFs become closer and closer together after successive iterations led us to believe that we were approaching optimality, and that much of the remaining gap after 10 iterations was due to looseness in our upper bound on OPT.

Similar to the results previously reported about our full conference in Section 5.2.2, we observed that introducing soft constraints led to only small reductions in aggscore across our 77 variable-sized conferences. For 60% of these

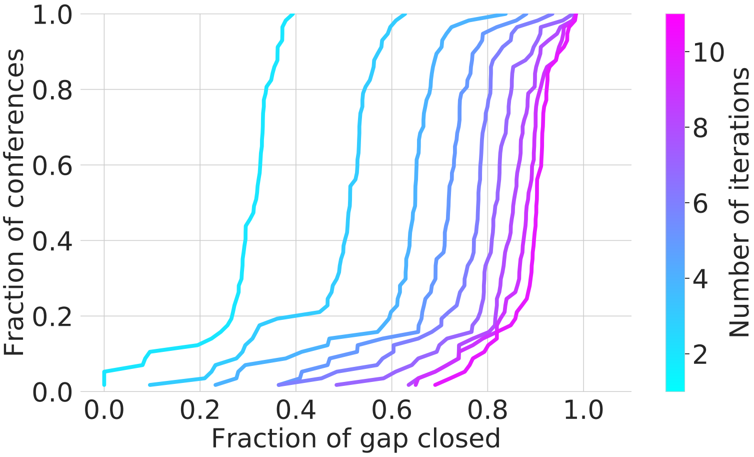

Figure 6: ECDF of % of gap closed (relative to our upper bound) between the first and k^{th} iteration of row generation.

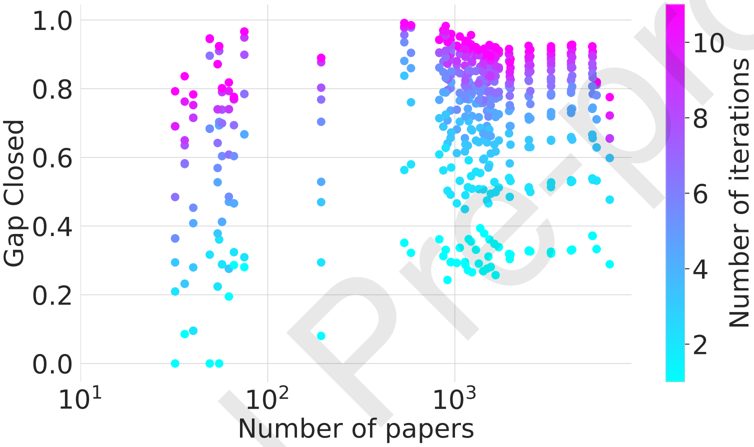

Figure 7: Conference size (number of papers) vs. % of gap closed (relative to our upper bound) between the first and k^{th} iteration of row generation.

conferences, we lost less than 2% in aggscore; at worst, we lost 5%. Figure 8 shows an ECDF of the reduction in aggscore across all the generated conferences.

5.3. Evaluation of LCM's Two-Phase Design

5.3.1. How prevalent were false negatives?

The risk we were most concerned about arising from LCM's Two-Phase design was the possibility that papers rejected in Phase 1 might have been accepted if they had received additional reviews, an opportunity for rebuttal, and/or discussion; we call such papers "false negatives". The cleanest way to estimate the probability with which we falsely rejected such papers in Phase 1 would have been for us to conduct a randomized experiment: randomly promoting some fraction of papers that would otherwise have been rejected in Phase 1 and observing the result. We did not perform this experiment, but the real data provides a very similar natural experiment. Specifically, we promoted papers to Phase 2 without considering their scores in two cases: (1) when one or more reviews were missing; and (2) when their reviewers had low confidence. In total, 231 papers were promoted in this manner, 16 of which were eventually accepted. Each of these papers eventually received four or more high-confidence reviews, allowing us to calculate the probability that two randomly selected reviews would have met the criteria for Phase 1 rejection. This allowed us to estimate what fraction of Phase 1 rejections might eventually have been accepted. In our data this probability was 2.9%, suggesting that Phase 1 rejections included very few false negatives.

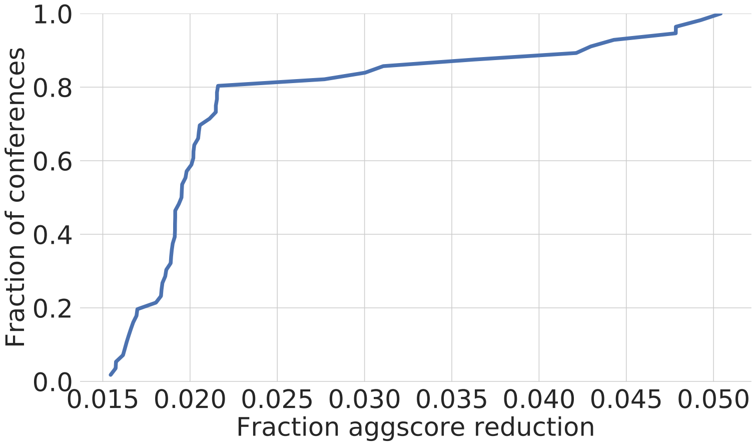

Figure 8: ECDF of the fractional reduction in mean aggscore upon adding soft constraints.

5.3.2. Did Phase 1 reviewers participate in discussion after Phase 2?

There was a relatively long delay between the end of Phase 1 and the start of the discussion period (in our case, almost a month), leading to some public speculation that Phase 1 reviewers might have tended to be less active in the discussion phase [e.g., Kambhampati, 2021]. We found little evidence of such a trend: Phase 2 papers received a discussion post from 55% of Phase 1 reviewers and likewise 55% of Phase 2 reviewers. We did find that reviewers who reviewed only in Phase 1 (who were responsible for a relatively small fraction of reviews overall) had a slightly smaller (52%) rate of participation in the discussion. We further note that Two-Phase reviewing is not the only possible explanation for this small discrepancy: e.g., this group may also have contained a smaller fraction of highly qualified reviewers.

5.3.3. How many additional reviews were gained?

AAAI 2021 received 7,133 full paper submissions to the first phase. We were able to reject 2,615 (about 37%) of the submitted papers in Phase 1, leaving us with a surplus of 2,615 reviews (relative to AAAI's previous practice of assigning 3 reviews to each paper) to spread amongst the remaining papers. This contributed towards assigning at least four reviewers to every main track paper in Phase 2 and at least 3 reviews to the 737 fast track submissions, out of which 721 submissions received more than 4 reviews.

5.3.4. How important was it to have additional reviews?

For every Phase 2 paper, we sampled every subset of 3 reviews that it might have received if a subset of the same reviewers had been assigned in a single-phase conference and calculated the confidence-weighted average score that the paper would have received in this scenario. (Confidence ranged between 1 (low) and 4 (high); corresponding weights were 0.25, 0.5, 0.75, and 1.) We plot the result in Figure 9. Each point corresponds to a 3-review scenario for a single paper. The x axis gives the paper's score in the AAAI 2021 conference (where it received more than 3 reviews); the y axis gives the spectrum of 3-review scores. The decision boundary for acceptance fell around 6.4, though many papers below this threshold were accepted and many above this threshold were rejected. Nevertheless, variance in the score is a good proxy for variance in reviewer support for the paper. Figure 9 shows that for papers that are close to the decision boundary, the weighted average score was particularly high variance, varying by up to 2 points around the threshold depending on which 3 reviewers were picked. Particularly for papers that are close to the decision boundary, we conclude that the additional reviews enabled by two-phase reviewing helped Program Chairs to make more accurate decisions.

5.3.5. Did the adoption of Two-Phase reviewing result in a higher bar for acceptance by ACs/SPCs?

One concern that has been publicly raised about two-phase reviewing is that it might depress acceptance rates because ACs and SPCs may aim to accept the same fraction of Phase 2 papers as in previous years, not considering the large fraction of papers already rejected in Phase 1 [e.g., Walsh, 2022]. Our analysis of AC/SPC recommendations shows that this was not the case in AAAI 2021. ACs submitted 3,825 ratings on a scale of *Reject, Lean Reject, Lean*

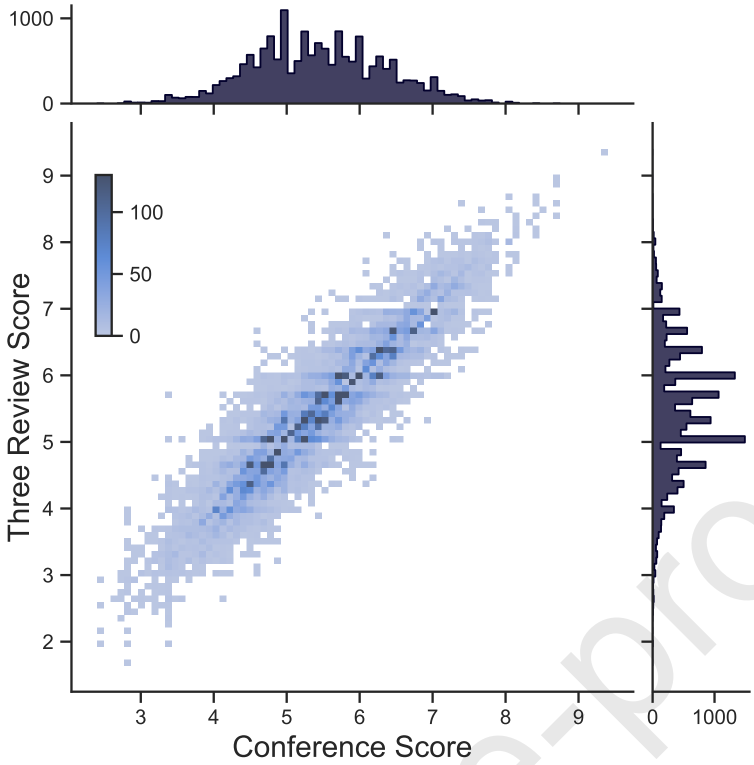

Figure 9: Confidence-weighted average scores for Phase 2 papers based on all reviews vs. all combinations of 3 reviews. Marginal densities are plotted on the sidelines.

Accept, Weak Accept, Accept, and Best Paper. Considering the last four scores as recommendations for acceptance, we found that ACs recommended accepting 37.6% of Phase 2 papers, corresponding to an overall acceptance rate of 22.2%. Similarly, SPCs submitted 3,873 ratings on a scale from 1 to 9.[11] Considering scores of 6 and above as accept recommendations, SPCs recommended accepting 37.5% of Phase 2 papers, corresponding to an overall acceptance rate of 22.3%. Both of these percentages were higher than the conference's actual acceptance rate of 21% and almost the same as the average acceptance rate of 22.9% over the last 5 years [OpenResearch, 2021]. We therefore see no evidence to support the claim that introducing two-phase reviewing led to lower overall acceptance rates in 2021.

5.3.6. *Did reviewers behave differently in Phase 2?*

In AAAI 2021, Phase 2 reviews tended to be (slightly) more negative than Phase 1 reviews, with Phase 1 pre-rebuttal scores averaging 0.29 higher than Phase 2 pre-rebuttal scores on the 10-point scale (see Figure 10 for an ECDF). We used the Wilcoxon signed-rank test [Rey and Neuhäuser, 2011] to check whether pairs of Phase 1 and Phase 2 reviews given to the same paper came from the same distribution (null hypothesis) or whether Phase 2 reviews tended to be more negative (alternative hypothesis). We ran the test with a significance level of 0.01 on a dataset whose entries were every possible combination of a Phase 1 review score and a Phase 2 review score for the same paper. (Thus, we only considered papers which were reviewed in Phase 2.) We used pre-rebuttal scores to make sure that reviewers' scores were not influenced by reading author responses and other reviews, after which review scores tended to become more correlated). Over 6160 pairs of scores in our dataset, the test had a p-value of $7.57 \times e^{-69}$, causing us to reject the null hypothesis that reviewing was identical across phases.

There are various reasons why reviewers might have been more negative in Phase 2: they may have applied a higher standard, knowing this was the final reviewing round; they may have changed their perceptions about quality thresholds

[11] The text prompts given for each score were as follows: (1) Trivial, wrong, or known; (2) Strong reject: will fight to get it rejected; (3) Clear reject; (4) Reject; (5) Below threshold of acceptance; (6) Above threshold of acceptance; (7) Accept; (8) Clear Accept (Top 50% accepted papers (est.)); and (9) Accept: will fight to get it accepted.

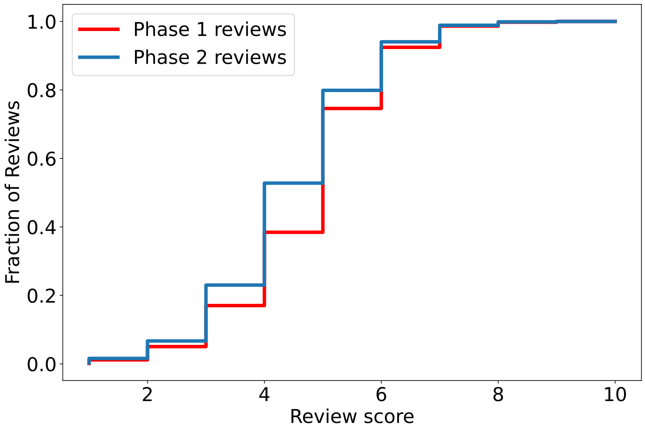

Figure 10: ECDF of the review scores by review phase. Phase 2 reviews tend to be more negative than Phase 1 reviews.

after their experiences reviewing in Phase 1; more qualified reviewers might have been identified in Phase 2 and these reviewers may have had more critical views.

However, a fourth explanation occurred to us, which the data can tell us something about: if reviewers behaved exactly as they did in Phase 1, some change in scores should be expected due to selection bias arising from the Two-Phase structure. To see how this would happen, imagine that each reviewer's score is an unbiased sample from a Gaussian centered on the "true score" of the paper. While reviewing in this model is unbiased on expectation, performing reviewing in two phases adds bias. That is, if both reviewers in Phase 1 draw samples above the paper's true score, the paper will get more reviews in Phase 2, which on expectation will be more negative (reverting to the mean). Conversely, if both reviewers in Phase 1 draw samples well below the paper's true score, we will not see additional reviews (which again would revert to the mean) because the paper will be rejected in Phase 1. This asymmetry means that we should expect reviews in the second phase to be at least somewhat more negative on average for purely statistical reasons; observing such bias is not necessarily cause for alarm.

To investigate the impact of this selection bias effect, we considered a refined dataset of papers that had at most 1 negative review (review score of 5 or below), and hence would not have been rejected in Phase 1 regardless of the order in which their reviewers were chosen; there turned out to be 1583 papers. We again ran the Wilcoxon signed-rank test comparing Phase 1 scores to Phase 2 scores and obtained a p-value of 0.57: we did not have enough evidence to reject the null hypothesis that the distributions of Phase 1 scores and Phase 2 scores were the same. This indicates that these papers were unlikely to have been promoted from Phase 1 purely due to noise.

We now turn to a more quantitative analysis. To see how much of the 0.29-point average score gap between reviews in the two phases could plausibly be explained by selection bias, we needed to estimate the variability of reviewers' process of scoring papers. We thus constructed a simple statistical model of our setting. We assumed that each paper p has a true score $s_p \in \mathbb{R}$ drawn from a Gaussian distribution $\mathcal{N}(\mu_s, \sigma_s^2)$ where μ_s and σ_s^2 indicate our prior belief on the average and variance of true scores. When asked to review paper p, reviewer r draws an unbiased sample o_r^p from a normal distribution $\mathcal{N}(s_p, \sigma^2)$ around s_p, the true score of paper p, and with the variance of σ^2. We note that when we use thousands of observations to fit a handful of parameters, priors are overwhelmed and hyperparameter choices become nearly irrelevant. Nevertheless, performing Bayesian inference requires the selection of priors. We therefore estimated $\mu_s = 5$ and $\sigma_s^2 = 1$ as the empirical mean and variance of overall paper scores and assumed that the prior over $\frac{1}{\sigma}$ was Gamma$(\alpha_\tau, \beta_\tau)$, setting $\alpha_\tau = \beta_\tau = 1$.

We used Gibbs sampling to estimate σ for reviewers and s_p for every p. After taking the average over the Gibbs samples, our inference estimated σ to be 1.3. We then ran a simulation in which we used our graphical model to generate

6,723 true paper scores[12] from $\mathcal{N}(5, 1)$ and review scores that followed the noise model defined by $o_p^r \sim \mathcal{N}(s_p, 1.3^2)$. We then simulated Phase 2 by filtering out any paper that received two reject recommendations (received two review scores below 4.5) from its first two sampled review scores (in Phase 1). Considering the remaining (Phase 2) papers in our synthetic experiment, we observed that Phase 1 reviews were 0.18 higher than Phase 2 reviews on average. Overall, this (simplistic) analysis explains about two thirds of the gap that we saw in our real data. We conclude that selection bias likely made reviewing in the two phases appear more different than it really was, but that it is furthermore plausible that reviewers did behave at least somewhat differently in the two phases.

6. Conclusion

This paper has presented Large Conference Matching (LCM), a novel method for reviewer–paper matching that is scalable to large conferences and more robust to malicious behavior. LCM is based on a mixed-integer program that combines a scoring function (which itself combines three match scores and reviewer bids) with various soft constraints encouraging the optimizer to pick, for each paper, a set of reviewers that is geographically diverse, includes at least one senior reviewer, and does not include coauthors. Several preprocessing steps predict new conflicts of interest alongside known ones, and reviewer bids are also processed to undermine malicious behavior. A Two-Phase reviewing scheme uses available reviewing resources more efficiently by shifting reviews away from papers that have low acceptance probability. We performed extensive evaluation and post-hoc analysis on data from AAAI 2021 to demonstrate the value of LCM. We have publicly released our implementation of LCM for further use by other conferences; indeed, it has been used by ICML 2022 and ACM-EC 2023. Furthermore, the two-phase reviewing methodology and various other novel elements of the design described here have been adopted by AAAI 2022, 2023 and 2024, IJCAI 2022, 2023, and 2024, and ICML 2022.

7. Ethics Statement

Since our study involves sensitive data from human participants, we obtained formal ethics approval from the University of British Columbia's Behavioural Research Ethics Board (BREB #H23-01087).

More broadly, publishing the details of a reviewer-paper matching system such as LCM can have long-term implications for future program chairs. Making the system more transparent to the community increases trust and confidence in the system, but also potentially opens it up to strategic behavior. While we maintain that obscuring the details of the matching process is not the right way to dissuade would-be manipulators, we also acknowledge that our system does not come with any guarantees that rule out misuse (e.g., formal claims of incentive compatibility). Despite the potential risks, an open and transparent discussion of a reviewer-paper matching pipeline allows the research community to collectively work towards developing fair and unbiased systems that enhance the dissemination of knowledge. Encouraging more research and discussion on this topic can lead to further advancements in the field, benefiting authors, reviewers, and the scientific community as a whole. In light of this, future chairs who use our system may need to be more watchful of malicious behavior, knowing that system details are publicly accessible (e.g., perhaps they should watch for the formation of international collusion rings among authors). We note that our pipeline contains many parameters, which chairs of future conferences are free to select their own based on the specific conference's requirements. This flexibility makes the mechanism harder to attack without knowledge of these exact parameters. On the other hand, failure to calibrate these constants carefully can create problems that can disproportionately affect some authors; e.g., if the geographical diversity constant is set too high, papers on topics primarily studied in one geographic region could be adversely affected.

References

Laurent Charlin and Richard S Zemel. The Toronto Paper Matching System : An automated paper-reviewer assignment system. 2013.

Graham Neubig, John Wieting, Arya McCarthy, Amanda Stent, Natalie Schluter, and Trevor Cohn. ACL reviewer matching code. https://github.com/acl-org/reviewer-paper-matching, 2020.

Steven Jecmen, Hanrui Zhang, Ryan Liu, Nihar B Shah, Vincent Conitzer, and Fei Fang. Mitigating manipulation in peer review via randomized reviewer assignments. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 12533–12545, 2020. URL https://proceedings.neurips.cc/paper/2020/file/93fb39474c51b8a82a68413e2a5ae17a-Paper.pdf.

[12] 6,723 is the total number of papers in our database that were either rejected or accepted based on their reviews in Phase 1 or Phase 2—i.e., excluding papers that left the system for other reasons, such as being withdrawn by their authors or being desk rejected because of double submission, violation of author anonymity, exceeding page limits, etc.

Camillo J Taylor. On the optimal assignment of conference papers to reviewers. 2008.

Peter A Flach, Sebastian Spiegler, Bruno Golénia, Simon Price, John Guiver, Ralf Herbrich, Thore Graepel, and Mohammed J Zaki. Novel tools to streamline the conference review process: Experiences from SIGKDD'09. *ACM SIGKDD Explorations Newsletter*, 11(2):63–67, 2010.

Naveen Garg, Telikepalli Kavitha, Amit Kumar, Kurt Mehlhorn, and Julián Mestre. Assigning papers to referees. *Algorithmica*, 58(1):119–136, 2010. doi: 10.1007/s00453-009-9386-0. URL https://doi.org/10.1007/s00453-009-9386-0.

Wenbin Tang, Jie Tang, and Chenhao Tan. Expertise matching via constraint-based optimization. In *IEEE/WIC/ACM International Conference on Web Intelligence*, pages 34–41, 2010. doi: 10.1109/WI-IAT.2010.133. URL https://doi.org/10.1109/WI-IAT.2010.133.

Laurent Charlin, Richard Zemel, and Craig Boutilier. A framework for optimizing paper matching. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 86–95, 2011.

Jing Wu Lian, Nicholas Mattei, Renee Noble, and Toby Walsh. The conference paper assignment problem: Using order weighted averages to assign indivisible goods. In *AAAI Conference on Artificial Intelligence*, pages 1138–1145, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17396.

Ari Kobren, Barna Saha, and Andrew McCallum. Paper matching with local fairness constraints. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD*, pages 1247–1257, 2019. doi: 10.1145/3292500.3330899. URL https://doi.org/10.1145/3292500.3330899.

Michael L Littman. Collusion rings threaten the integrity of computer science research. *Communications of the ACM*, 64(6):43–44, 2021. URL https://cacm.acm.org/magazines/2021/6/252840-collusion-rings-threaten-the-integrity-of-computer-science-research/fulltext.

Christian Bessiere. [IJCAI-PRICAI 2020] discussions about summary reject. http://www.lirmm.fr/~bessiere/Site/Media/SR-clarification, 2020.

Neil Lawrence and Corinna Cortes. The NIPS experiment. http://inverseprobability.com/2014/12/16/the-nips-experiment/, 2014.

Cheng Long, Raymond Chi-Wing Wong, Yu Peng, and Liangliang Ye. On good and fair paper-reviewer assignment. In *IEEE International Conference on Data Mining*, pages 1145–1150, 2013.

Ritesh Noothigattu, Nihar B Shah, and Ariel Procaccia. Loss functions, axioms, and peer review. *Journal of Artificial Intelligence Research*, 70: 1481–1515, 2021.

Ruihan Wu, Chuan Guo, Felix Wu, Rahul Kidambi, Laurens Van Der Maaten, and Kilian Weinberger. Making paper reviewing robust to bid manipulation attacks. In *International Conference on Machine Learning*, pages 11240–11250. PMLR, 2021.

David M. Mimno and Andrew McCallum. Expertise modeling for matching papers with reviewers. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 500–509, 2007. doi: 10.1145/1281192.1281247. URL https://doi.org/10.1145/1281192.1281247.

Omer Anjum, Hongyu Gong, Suma Bhat, Wen-Mei Hwu, and Jinjun Xiong. PaRe: A paper-reviewer matching approach using a common topic space. In *Empirical Methods in Natural Language Processing*, pages 518–528, 2019. doi: 10.18653/v1/D19-1049. URL https://doi.org/10.18653/v1/D19-1049.

John Wieting, Kevin Gimpel, Graham Neubig, and Taylor Berg-Kirkpatrick. Simple and effective paraphrastic similarity from parallel translations. In *Conference of the Association for Computational Linguistics, ACL*, pages 4602–4608, 2019. doi: 10.18653/v1/p19-1453. URL https://doi.org/10.18653/v1/p19-1453.

Don Conry, Yehuda Koren, and Naren Ramakrishnan. Recommender systems for the conference paper assignment problem. In *ACM conference on Recommender systems*, pages 357–360, 2009.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL https://aclanthology.org/D18-2012.

Ivan Stelmakh, Nihar B Shah, and Aarti Singh. PeerReview4All: Fair and accurate reviewer assignment in peer review. In *Algorithmic Learning Theory*, pages 828–856. PMLR, 2019.

Yuan Yuan. Poorly-explained NeurIPS 2020 desk-rejects peeve ML researchers. https://syncedreview.com/2020/07/16/poorly-explained-neurips-2020-desk-rejects-peeve-ml-researchers/, 2020.

Leon Bottou and Michael Littman. ICML 2009's call for paper. https://icml.cc/Conferences/2009/cfp.html, 2009.

Vikram S. Adve and James C. Hoe. ASPLOS XV: Proceedings of the fifteenth international conference on architectural support for programming languages and operating systems. https://dl.acm.org/doi/proceedings/10.1145/1736020, 2010.

Cristina V. Lopes. The oopsla two-phase review process. *SIGPLAN Not.*, 49(4S):27–32, jul 2014. ISSN 0362-1340. doi: 10.1145/2641638.2641648. URL https://doi.org/10.1145/2641638.2641648.

Christos Kozyrakis and Emery Berger. Using extended abstracts to improve peer review. https://www.sigarch.org/using-extended-abstracts-to-improve-peer-review/, 2021.

Hans Boehm, Jack Davidson, Kathleen Fisher, Cormac Flanagan, Jeffrey S. Foster, Jeremy Gibbons, Mary Hall, Graham Hutton, David Padua, Frank Tip, Jan Vitek, Philip Wadler, Keshav Pingali, Michael O'Boyle, Steve Blackburn, and Emery Berger. Practices of PLDI. https://www.sigplan.org/sites/default/files/PracticesofPLDI.pdf, 2021.

Steven Jecmen, Hanrui Zhang, Ryan Liu, Fei Fang, Vincent Conitzer, and Nihar B Shah. Near-optimal reviewer splitting in two-phase paper reviewing and conference experiment design, 2021. URL https://arxiv.org/abs/2108.06371.

Neil Lawrence. Paper allocation for NIPS, Jun 2014. URL https://inverseprobability.com/2014/06/28/paper-allocation-for-nips.

Subbarao Kambhampati. [Twitter 2021] tweet about two-phase. https://twitter.com/rao2z/status/1466408324908216320?s=20, 2021.

Toby Walsh. [Twitter 2022] tweet about two-phase. https://twitter.com/TobyWalsh/status/1497058084384276484, 2022.

OpenResearch. AAAI acceptance rate. https://www.openresearch.org/wiki/AAAI, 2021.

Denise Rey and Markus Neuhäuser. *Wilcoxon-Signed-Rank Test*, pages 1658–1659. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-04898-2. doi: 10.1007/978-3-642-04898-2_616. URL https://doi.org/10.1007/978-3-642-04898-2_616.

A. MIP Problem Formulation

LCM's mixed-integer program is formally defined as follows. Let us denote the set of N papers by $\mathcal{P}$, the set of Program Committee members (PCs) by $\mathcal{PC}$, the set of SPCs (Senior Program Committee members) by $\mathcal{SPC}$, and the set of Area Chairs (ACs) by $\mathcal{AC}$. We define the set of reviewers to be $\mathcal{R} = \mathcal{PC} \cup \mathcal{SPC} \cup \mathcal{AC}$ and use the term *reviewer* to refer to any of a PC, SPC or AC.

Our goal is to assign γ_{pc} PCs, γ_{spc} SPCs, and γ_{ac} ACs to each paper p_i, for $1 \leq i \leq N$, such that the total reviewer–paper matching score is maximized subject to a set of constraints denoted by C.

Our overall objective function not only includes the total matching score, but also various penalties via the slack variables, due to violation of corresponding hard constraints.

We initialize the overall objective function O as the total matching score and then keep on adding terms to it as we relax the various constraints in the next section. We create binary variables x_{ij} that denote whether paper p_i is assigned to reviewer r_j.[13] We can write:

$$\text{Initial Objective: } O^{\text{match}} = \sum_{i\in\mathcal{P}, j\in\mathcal{R}} s_{ij}x_{ij}. \tag{1}$$

Recall that $s_{ij} \in [0, 1]$ is the aggscore, the matching score between reviewer r_j and paper p_i as discussed in Section 2.3. Next, we formulate all the (soft) constraints in C as described earlier.

A.1. Formulation of Constraints in C

Constraint 1 (COI): Let $\text{coi}_{ij} = 1$, if there exists a conflict of interest between reviewer j and any of the authors of the paper i, and 0 otherwise. Reviewer j cannot review paper i if $\text{coi}_{ij} = 1$:

$$x_{ij} = 0, \text{ for each } i \in \mathcal{P}, j \in \mathcal{R}, \text{ such that } \text{coi}_{ij} = 1. \tag{2}$$

Constraint 2 (Number of reviewers): For each paper p_i, we need to ensure that it gets the desired number of reviewers:

$$\sum_{j\in\mathcal{PC}} x_{ij} = \gamma_{\text{pc}} \;\; ; \;\; \sum_{j\in\mathcal{SPC}} x_{ij} = \gamma_{\text{spc}} \;\; ; \;\; \sum_{j\in\mathcal{AC}} x_{ij} = \gamma_{\text{ac}}$$

With the equality constraints above, the MIP may become infeasible. Hence, we replace it with inequality constraints:

$$\sum_{j\in\mathcal{PC}} x_{ij} \leq \gamma_{\text{pc}} \;\; ; \;\; \sum_{j\in\mathcal{SPC}} x_{ij} \leq \gamma_{\text{spc}} \;\; ; \;\; \sum_{j\in\mathcal{AC}} x_{ij} \leq \gamma_{\text{ac}} \tag{3}$$

These constraints would be tight for a maximization problem where each match contributes positively to the objective. Note that it is possible to have a case in which conditional on the rest of the assignment, the only remaining matches do not positively contribute to the objective function. In such a case, the optimizer will not assign those papers their full capacity of reviewers since such assignments would be of poor quality. We manually reviewed such cases.

Constraint 3 (Reviewer Load): The total number of papers assigned to a reviewer j should not exceed their capacity $c_j \in \mathbb{Z}^+$:

$$\sum_{i\in\mathcal{P}} x_{ij} \leq c_j, \forall j \in \mathcal{PC} \tag{4}$$

$$\sum_{i\in\mathcal{P}} x_{ij} \leq c_j, \forall j \in \mathcal{SPC} \cup \mathcal{AC}$$

Just ensuring the above upper bound may lead to an inequitable assignment of papers to reviewers, especially ACs and SPCs, many of whom may end up with no assignment at all. To resolve this, one option is to add a lower bound l_j on the number of papers assigned to ACs and SPCs, as suggested by Kobren et al. [2019]:

$$\sum_{i\in\mathcal{P}} x_{ij} \geq l_j, \forall j \in \mathcal{SPC} \cup \mathcal{AC}.$$

[13] In Phase 2, we set $x_{ij} = 1$ if r_j was already assigned to p_i in Phase 1 to ensure that the matching from Phase 2 is a superset of Phase 1 matching. The capacity limits on reviewers were increased accordingly.

However, the lower bound constraint only ensures that no SPC or AC is unassigned, rather than preventing skewed matchings. To promote an equitable distribution of papers amongst reviewers, we introduce W intermediate capacity levels. For each $w \in \{1, 2, \dots, W\}$, we define c_{jw} to be the intermediate capacity for reviewer j and set $c_{j1} = l_j$ and $c_{jW} = c_j$. For ACs and SPCs, we replace the hard capacity constraints with a sequence of soft capacity constraints, one for each capacity level w. These constraints promote equitable distribution of papers as additional penalties are incurred each time an assignment crosses a new capacity level, making it very expensive to overload an individual.

To do so, we introduce new slack variables, $s^{\text{cap}}_{jw} \geq 0, \forall j \in \mathcal{SPC} \cup \mathcal{AC}, \forall w \in \{1 \dots W\}$, each of which relax the capacity c_{jw} to $s^{\text{cap}}_{jw} + c_{jw}$, but penalize the objective by $p^{\text{cap}}_{w}(\text{type}_j)$ for each additional assignment above c_{jw}, where type_j denotes the reviewer category (SPC or AC). Then, for each $w \in \{1 \dots W\}$:

$$\sum_{i \in \mathcal{P}} x_{ij} \leq s^{\text{cap}}_{jw} + c_{jw}, \;\; \forall j \in \mathcal{SPC} \cup \mathcal{AC} \text{ and } \forall w \in \{1 \dots W\}; \tag{5}$$

$$O^{\text{cap}} = \sum_{w \in W} \sum_{j \in \mathcal{SPC} \cup \mathcal{AC}} p^{\text{cap}}_{w}(\text{type}_j) s^{\text{cap}}_{jw}. \tag{6}$$

Constraint 4 (Seniority): We first assign $\text{seniority}_j \in \{0, 1, 2, 3\}$ to each reviewer j, such that 3 is the senior-most category and 0 is the junior-most category. In AAAI 2021, we chose to assign seniority levels as follows: Reviewers who had previously reviewed for at least 3 relevant conferences[14] or had published 10 or more papers in relevant conferences were assigned seniority level 3. Remaining reviewers having between 4 and 9 published papers were assigned seniority level 2. Remaining reviewers having 2 to 4 papers or with experience reviewing at one or two previous conferences were assigned seniority level 1. All other reviewers were assigned a seniority level of 0.

We then reward each paper for maximizing its seniority level, up to some maximum reward. We introduce slack variables, $s^{\text{sen}}_{i} \geq 0 \; \forall i \in \mathcal{P}$.

$$s^{\text{sen}}_{i} \leq \sum_{j \in \mathcal{PC}} \text{seniority}_j x_{ij}, \forall i \in \mathcal{P} \tag{7}$$

$$s^{\text{sen}}_{i} \leq \text{TargetSeniority}, \forall i \in \mathcal{P} \tag{8}$$

$$s^{\text{sen}}_{i} \geq \text{MinSeniority}, \forall i \in \mathcal{P} \tag{9}$$

$$O^{\text{sen}} = \sum_{i \in \mathcal{P}} \text{Reward}^{\text{sen}} s^{\text{sen}}_{i} \tag{10}$$

Note that setting MinSeniority greater than 0 can result in possible infeasibility (i.e., it may not be possible to assign a minimum level of seniority simultaneously to every paper). Setting MinSeniority = 0 keeps the constraint soft. Papers are rewarded for seniority up to a maximum seniority level given by TargetSeniority. Setting this target allows us to express a preference for spreading seniority out over many papers rather than concentrating it on a few papers, as the objective only gets a bonus for the first TargetSeniority units of seniority per paper.

Constraint 5 (Coauthorship Distance): This set of constraints penalizes the assignment of pairs of reviewers who have ever coauthored a paper together to the same paper. This reduces the chances of papers' reviewers knowing each other, promoting review diversity. To do so, for each pair of reviewers in $V = \{(j, j') | j < j', j, j' \in \mathcal{PC} \cup \mathcal{SPC}\}$, we define a new variable $\text{cad}_{jj'} \geq 0$ such that it takes the value of 1 if there exists a common paper assigned to both j and j', and 0 otherwise. We exclude ACs from this set of constraints as they are few in number, highly trusted, and are not directly involved in discussions with the PCs. The following constraints define $\text{cad}_{jj'}$:

$$\text{cad}_{jj'} \geq 0, \forall j, j' \in V; \tag{11}$$

$$\text{cad}_{jj'} \geq x_{ij} + x_{ij'} - 1, \;\; \forall i \in \mathcal{P} \text{ and } \forall j, j' \in V. \tag{12}$$

Next, let $d_{jj'}$ denote the edge distance between two reviewers j and j' in a coauthorship graph with reviewers as nodes. An edge exists between j and j' iff they have ever published a paper together (based on DBLP data, as described

[14] Reviewers were asked: "How many times have you been part of the Program Committee (as PC/SPC/AC, etc) of AAAI, IJCAI, NeurIPS, ACL, SIGIR, WWW, RSS, NAACL, KDD, IROS, ICRA, ICML, ICCV, EMNLP, EC, CVPR, AAMAS, HCOMP, HRI, ICAPS, ICDM, ICLR, ICWSM, IUI, KR, SAT, WSDM, UAI, AISTATS, COLT, CORL, CP, CPAIOR, ECAI, or ECML in the past?"

in Section 2.1.2). To discourage coauthors being reviewers of the same paper, we add a penalty that is bigger when coauthorship distance is small:

$$O^{\text{co}} = \sum_{(j,j')\in V} p^{co}(d_{jj'})\text{cad}_{jj'}. \tag{13}$$

$p^{co}(d)$ is the penalty coefficient, which depends on coauthorship edge distance d. We set $|p^{co}(1)| > |p^{co}(2)| > 0$ and $p^{co}(d) = 0, \forall d \geq 3$.

Constraint 6 (Geographic Diversity): For each paper i, let variable reg_i count the number of distinct geographic regions of its assigned PCs and SPC. We inferred regions based on academic affiliations and email addresses provided by the reviewers. To encourage assignment of reviewers from different geographic regions, we add a reward proportional to reg_i in the overall objective function so that reg_i is maximized. To count reg_i, we need to introduce auxiliary indicator variables, $v_{ir} \leq 1$, that take the value of 1 only if paper i is assigned at least 1 reviewer from region $r \in$ Regions. Let t_{jr} indicate if reviewer j belongs to region r or not. The following expressions define the overall regional constraints for each paper.

$$\text{reg}_i \leq \sum_{r\in\text{Regions}} v_{ir},\ \forall i \in \mathcal{P} \tag{14}$$

$$v_{ir} \leq \sum_{j\in\mathcal{PC}\cup\mathcal{SPC}} t_{jr}x_{ij}, \forall r \in \text{Regions and } \forall i \in \mathcal{P} \tag{15}$$

$$v_{ir} \leq 1, \forall r \in \text{Regions and } \forall i \in \mathcal{P} \tag{16}$$

$$O^{\text{reg}} = \sum_{i\in\mathcal{P}} \text{Reward}^{\text{reg}}\text{reg}_i \tag{17}$$

Constraint 7 (No 2-Cycles): These constraints aim to avoid scenarios where two reviewers both bid positively on each other's papers and are assigned these papers to review. Similar to Constraint 5 (Coauthorship Distance) and 6 (Geographic Diversity), we exclude ACs from this constraint for similar reasons. We first identify all such bidding cycles, where reviewer j bids positively on paper i authored by reviewer j' and j' bids positively on paper i' authored by reviewer j, with $j, j' \in \mathcal{PC} \cup \mathcal{SPC}$. Denote such a bidding cycle by a 4-tuple (j, j', i, i'), and the set of all such cycles as CY. We then aim to avoid any of the assignments in the bidding cycle (j, j', i, i'). We introduce slack variables, $s^{\text{cy}}_{jj'ii'} \geq 0$, for each bidding cycle, and add a penalty p^{cy} in Equation (19) for every assignment that belongs to a bidding cycle.[15]

$$x_{ij} \leq s^{\text{cy}}_{jj'ii'}, \forall j, j', i, i' \in CY \tag{18}$$

$$O^{\text{cy}} = \sum_{j,j',i,i'\in CY} p^{\text{cy}} s^{\text{cy}}_{jj'ii'} \tag{19}$$

Finally, our overall assignment problem can be stated as:

$$\max O = O^{\text{match}} + O^{\text{cap}} + O^{\text{sen}} + O^{\text{co}} + O^{\text{reg}} + O^{\text{cy}}$$
$$\text{subject to } C.$$

A.2. AAAI 2021 MIP parameters

In AAAI 2021, we used the following parameters: $\text{Reward}^{\text{reg}} = 0.1$, $|\text{Regions}| = 5$, $p^{\text{cy}} = -0.05$, $p^{\text{co}}(1) = -0.3$, $p^{\text{co}}(2) = -0.2$, $\text{Reward}^{\text{sen}} = 0.1$, TargetSeniority = 4, MinSeniority = 0. In Phase 1, we set reviewer capacities to 3 for PCs, and set $\gamma_{\text{pc}} = 2$, $\gamma_{\text{spc}} = \gamma_{\text{ac}} = 1$; we increased PC capacity to 4, and set $\gamma_{\text{pc}} = 4$, $\gamma_{\text{spc}} = \gamma_{\text{ac}} = 0$ in Phase 2. We only made assignments to SPCs and ACs in Phase 1 except for the fast track papers for which we set $\gamma_{\text{spc}} = \gamma_{\text{ac}} = 1$. We set reviewer capacities to 24 and 60 for SPCs and ACs respectively. We set $W = 5$ with $c_{j\cdot} = [8, 12, 16, 20, 24]$

[15] In the conference and this paper's experiments, we used an alternate formulation of the constraint which only penalized bidding cycles that led to assignment cycles. In this formulation, slack variables $s^{\text{cy}}_{jj'}$ were created for each pair of reviewers (j, j') involved in a bidding cycle, and the constraint was enforced with $x_{ij} + x_{i'j'} \leq (1 + s^{\text{cy}}_{jj'})$. We present the above formulation instead because, on reflection, we recommend penalizing both halves of a cycle separately.

for all $j \in \mathcal{SPC}$ and $c_{j\cdot} = [20, 30, 40, 50, 60]$ for all $j \in \mathcal{AC}$, with $p_w^{\text{cap}}(\cdot) = -0.05, 1 \leq w \leq (W - 1)$ and a very high penalty of -0.5 for $p_W^{\text{cap}}(\cdot)$ for both SPCs and ACs, so that the system never assigned beyond $c_{jW} = c_j$.

The values of these parameters express tradeoffs between constraint violations and match quality. They reflect Program Chairs' subjective assessments of the relative importance of allocating senior reviewers to each paper or the risk of authors forming bidding cycles. For example, in AAAI 2021, by setting $\text{Reward}^{\text{reg}} = 0.1$, we specified that, all things being equal, adding a reviewer from a new region was worth an aggscore reduction of 0.1. We note however that the experiments in Section 5.2.2 indicate that the average, aggregate aggscore costs of each soft constraint were much smaller than these constants. In the case of our AAAI 2021 dataset, this means that it was usually possible to satisfy these constraints at much smaller aggscore penalties and hence most of the matching was robust to the specific values of these constants. This paper does not experimentally investigate varying these parameter settings. We expect that other conference organizers will want to set them differently based on their own assessments of their conferences' needs.

Declaration of interests

☐ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☒ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Kevin Leyton-Brown reports financial support was provided by NSERC Discovery Grant, a DNDNSERC Discovery Grant Supplement, a CIFAR Canada AI Research Chair (Alberta Machine Intelligence Institute), a Compute Canada RAC Allocation, and awards from Facebook Research and Amazon Research. Mausam reports financial support was provided by grants from Google, Bloomberg, 1MG, IBM AIHN, Jai Gupta chair fellowship by IIT Delhi, and the Visvesvaraya Young Faculty Fellowship by Govt. of India. Yatin Nandwani reports financial support was provided by grant from AAAI as a workflow chair and a scholarship from Indian Institute of Technology Delhi. Hedayat Zarkoob reports financial support was provided by grant from AAAI as a workflow chair. Dinesh Raghu reports financial support was provided by grant from AAAI as a workflow chair. co-author Kevin Leyton-Brown was previously an associate editor of the Artificial Intelligence Journal (AIJ) and the Journal of AI Research (JAIR). co-author Mausam is an associate editor with the Artificial Intelligence Journal (AIJ).