

A Human Computation Framework for Boosting Combinatorial Solvers

Ronan Le Bras, Yexiang Xue, Richard Bernstein, Carla P. Gomes, Bart Selman

Computer Science Department
Cornell University
Ithaca, NY 14853

Abstract

We propose a general framework for boosting combinatorial solvers through human computation. Our framework combines insights from human workers with the power of combinatorial optimization. The combinatorial solver is also used to guide requests for the workers, and thereby obtain the most useful human feedback quickly. Our approach also incorporates a problem decomposition approach with a general strategy for discarding incorrect human input. We apply this framework in the domain of materials discovery, and demonstrate a speedup of over an order of magnitude.

Introduction

The past decade has witnessed the rapid emergence of the field of human computation, along with numerous successful applications. Human computation is motivated by problems for which automated algorithms cannot yet exceed human performance (Von Ahn 2005). Indeed, some tasks are naturally and truly easy for humans, while they remain surprisingly challenging for machines. These problems typically involve a perceptual or cognitive component. For example, successful applications with a strong visual recognition component include the ESP game (Von Ahn and Dabbish 2004), Peekaboom (Von Ahn, Liu, and Blum 2006), and Eyespy (Bell et al. 2009), while TagATune (Law and Von Ahn 2009) and the Listen Game (Turnbull et al. 2007) make extensive use of the human ability to perceive and recognize sounds. In addition, human computation applications might exploit the implicit, background or commonsense knowledge of humans, as it is the case for Verbosity (Von Ahn, Kedia, and Blum 2006) and the Common Consensus system (Lieberman, Smith, and Teeters 2007). Recent developments have also demonstrated how to successfully exploit the wisdom of crowds by combining user annotations for image labeling (Welinder et al. 2010; Zhou et al. 2012) or clustering tasks (Gomes et al. 2011; Yi et al. 2012; Chen et al. 2010).

The work described in this paper is motivated by application domains that involve visual and audio tasks. In particular, we focus on a central problem in combinatorial materials discovery. New materials will help us address some

of the key challenges our society faces today, in terms of finding a path towards a sustainable planet (White 2012; Patel 2011). In combinatorial materials discovery, scientists experimentally explore large numbers of combinations of different elements with the hope of finding new compounds with interesting properties, e.g., for efficient fuel cells or solar cell arrays. We are collaborating with two teams of materials scientists, the Energy Materials Center at Cornell (emc2) and the Joint Center for Artificial Photosynthesis (JCAP) at Caltech. An overall goal is to develop the capability of analyzing data from over one million new materials samples per day. Automated data analysis tools, boosted with a human computation component, will be key to the success of this project.

We consider a central task in combinatorial materials discovery, namely the problem of identifying the crystalline phases of inorganic compounds based on an analysis of high-intensity X-ray patterns. In our approach, we integrate a state-of-the-art optimization framework based on constraint reasoning with a human computation component. This hybrid framework reduces our analysis time by orders of magnitude compared to running the constraint solver by itself. The human input also helps us improve the quality of solutions. For the human computation component, we developed a relatively simple and appealing visual representation of the X-ray images using heat maps (i.e. color-coded graphical representations of 2-D real-valued data matrices), which allows us to decompose the problem into manageable Human Intelligence Tasks (HITs), involving the identification of simple visual patterns, requiring no prior knowledge about materials science.

This work is part of our broader research agenda focused on *harnessing human insights to solve hard combinatorial problems*. Our work is close in spirit to the seminal FoldIt project for protein folding (Cooper et al. 2010). In FoldIt, human gamers are the main driving force for finding new protein folds, complemented with a limited amount of local computation (e.g., “shaking” of structures). We are proposing a framework that provides a much *tighter integration of a combinatorial solver with human insights*. Our approach takes advantage of the dramatic improvements in combinatorial solvers in recent years, allowing us to handle practical instances with millions of variables and constraints. Our objective is also to minimize the amount of required user in-

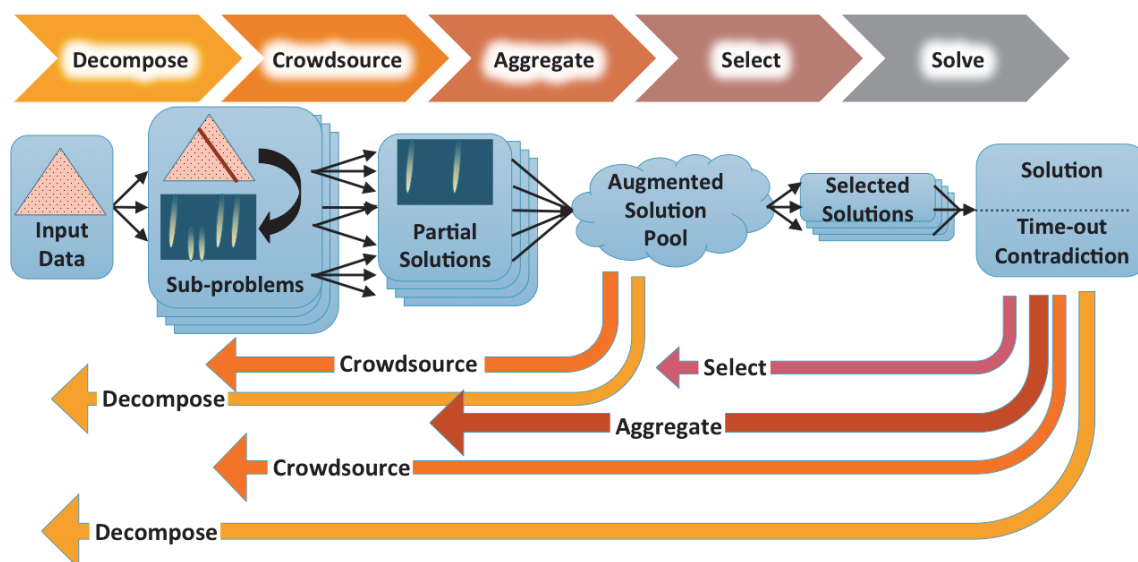


Figure 1: Diagram of the proposed framework.

put. In FoldIt, human gamers spent significant amounts of time discovering improved and interesting folds for a single given protein sequence. In our setting, our aim is to analyze data coming from over a million compounds per day. We therefore need to develop a setting where the required human input per data set is minimal. In our framework, we aim to identify only those cases where fully automatic analysis does not reach a high quality interpretation and where human input can make a real difference. Another work investigates how humans can guide heuristic search algorithms. (Acuña and Parada 2010) show that humans provide good solutions to the computationally intractable *traveling salesman problem*. Also, when modifying their solutions, they tend to modify only the sub-optimal components of their solutions. Similarly, in human-guided search (Anderson et al. 2000; Klau et al. 2002), humans visualize the state of a combinatorial optimizer, and leverage their pattern recognition skills to identify and guide the solver towards promising solution subspaces. This work shows that human guidance can improve the performance of heuristic searches. While that work uses human insights to guide local search algorithms in an interactive optimization framework, our work a human task in order to provide global insights about the problem structure.

Our proposed algorithmic framework tightly integrates human-computation with state-of-the-art combinatorial solvers using problem decomposition, and an incremental recursive strategy. The approach extends the solvers by directly incorporating subtle human insight into the combinatorial search process. Current combinatorial solvers can solve problem instances with millions of variables and constraints. This development has been rather surprising given that the worst-case complexity of the underlying constrained optimization problems is exponential (assuming P not equal NP). The explanation of the good practical performance of such solvers is that they exploit *hidden problem structure*.

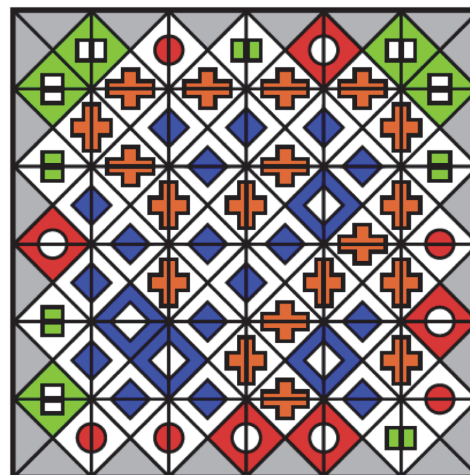


Figure 2: An edge-matching puzzle (Ansótegui et al. 2013).

One example of hidden structure is captured by the notion of a backdoor variable set. Given a constraint satisfaction problem instance, a set of variables is a backdoor if assigning values to those variables leads to a problem that can be solved in polynomial time by using polytime propagation techniques. Intuitively, small backdoor sets capture the combinatorics of the problem and explain how current solvers can be so efficient. (Randomized restarts are often used to find runs with small backdoors.) It has been found that many practical problems have relatively small backdoor sets, on the order of a few percent of the total number of variables (Kilby et al. 2005; Szeider 2006; Dilkina et al. 2009; O’Sullivan 2010; Fischetti and Monaci 2011; Gaspers and Szeider 2012). Current solvers use heuristics to find backdoor variables, but there is still clear room for improvement

in a solver’s ability to focus in on the backdoor variables. This is where human computation can provide additional insights. As we will show, when presented with parts of the overall combinatorial task in a visual form (“heat-maps”), humans can provide useful insights into the setting of hidden backdoor variables. Subsequently asserting these variable settings can significantly boost the performance of the solvers.

Figure 1 illustrates our overall framework. In order to solicit human insights, we need to decompose the overall task into a series of relatively small subproblems that can be viewed effectively for human input. At a high-level, the idea is to decompose the underlying constraint optimization problem into a set of sub-problems with overlapping variables and constraints. Human insights can then help solve the subtasks, providing potential partial solutions to the overall problem. By cycling through many partial solutions of the subproblems, we can search for a coherent overall interpretation with optimal or near optimal utility. Another key issue we have to deal with is possible inconsistent or erroneous human input. Such input will result in an overall inconsistent interpretation of our problem, which can be detected by the aggregation/selection procedures or by the complete solver. Interestingly, we can use our complete constraint solver to identify parts of the user feedback that cause the overall inconsistency. This information can be extracted from what is called *unsat cores*, which identify minimal sets of inconsistent constraints. We can then use this information to reject some of the user human input, potentially asking for new human input, thus repeating the analysis cycle. The back arrows in Fig. 1 capture the potential feedback loops from the aggregator and complete solver to previous phases when inconsistency is detected or time-out occurs.

In the next section we provide a high-level description of our framework and how it applies to the edge-matching puzzle problem, an intuitive visual constraint satisfaction problem. We then describe our motivating materials discovery problem and the different components of our framework for solving it. We present an empirical evaluation of our framework applied to the materials discovery problem that shows a significant reduction in data analysis and interpretation times. Overall, our results show that human computation and fully automated constraint optimization methods can work in a hybrid complementary setting, leading to significant overall performance gains.

Framework

In this section, we provide a high-level description of our framework and how it applies to an intuitive constraint satisfaction problem, the edge-matching puzzle problem, while the following sections describe each of its components in more details.

We consider a general setting where the problem at hand is a constraint satisfaction problem (CSP). Namely, given a set of variables and a set of constraints on those variables, the problem is to find an assignment (“solution”) that satisfies all the constraints. In addition, one can define an optimization criterion by introducing a cost measure for each potential solution. Given a problem instance, a CSP encodes

the instance as a set of variables and constraints, and outputs a solution satisfying these constraints. As an example we consider the edge-matching puzzle problem (see Fig. 2).

The edge-matching puzzle is a tiling puzzle involving tiling an area, typically with regular polygons (e.g., squares), whose edges are differentiated with different colors or patterns, in such a way that the edges of adjacent tiles match. Edge-matching puzzles are challenging since there is no picture to guide the puzzler and the fact that two pieces go together is not a guarantee that they should be together. In order to guarantee the correctness of any local partial solution it is required to complete the entire puzzle. This problem has been shown to be NP Complete (Demaine and Demaine 2007).

The edge-matching puzzle problem can be formulated as a constraint satisfaction problem. In particular the problem can be encoded as a Satisfiability (SAT) problem. Each Boolean variable denotes a given puzzle piece, with a given rotation, being placed at a given cell (i, j) . The constraints of this problem can be encoded as clauses, stating: (1) a cell has one puzzle piece assigned to it; (2) a puzzle piece is assigned to a single cell; (3) a piece matches its neighbors; and (4) for the case of puzzles with specific pieces for the border (say a specific color for the frame) border pieces are not allowed to be assigned to internal cells. We can also consider another “dual” encoding, in which the dual variables represent how the edges of the puzzle are colored. While the encodings are redundant, often it is advantageous to combine them to enforce a greater level of constraint propagation that typically speeds up search (Ansótegui et al. 2013).

Combinatorial solvers have been developed to solve CSPs (Rossi, Van Beek, and Walsh 2006) and exploit advanced techniques to speed up the solution runtime. Yet, some problem instances may not be solved in any reasonable amount of time. In order to boost combinatorial solvers, we propose a framework that combines human computation with state-of-the-art combinatorial solvers into an incremental and recursive process.

Figure 1 illustrates the components of this framework, as described below.

Decomposition: The problem instance is decomposed into overlapping subproblems. Combinatorial problem decomposition typically involves subproblems that are substantially faster to solve or computationally tractable. In our case, we are interested in subproblems that, although might still be hard for computers, become tractable for humans. As in many human computation tasks, the challenge is to present the worker with a partial view of the problem for which he or she will be able to provide valuable insights. *Overlapping* subproblems allow for better consistency in the presence of interdependent structure, which differs from other human computation tasks, such as image labeling, which tend to have more independence. In the edge-matching puzzle we decompose the problem into sub-problems by splitting the set of puzzle pieces into non-disjoint subsets. The decomposition of the puzzle pieces into overlapping subsets may be based on visual features of the pieces, for example grouping pieces by predominant colors. A key aspect of the decomposition procedure is that each

puzzle piece should be assigned to multiple subproblems and sub-problems are selected using some locality notion from the problem domain, e.g., the piece predominant colors.

Crowdsourcing: Each subproblem is solved independently multiple times and results in some candidate solutions. In the case of the edge-matching puzzle, a worker provides a small set of partial solutions, i.e., small sets of partial puzzle patterns. Note that we do not assume that these solutions are correct for the original problem, or even the subproblem. However, we expect that typically, many of the subproblem solutions are reasonable considering only the subproblem, and some of them are consistent or nearly consistent with the full problem.

Aggregation: This step combines partial solutions to produce new candidates that provide information about a larger segment of the original problem than the individual responses. Generally, this involves designing one or more problem-specific operations that describe the portions of two or more partial solutions that are compatible with each other, followed by their recursive application. These combination operations can be more complex than well-known schemes like voting, agreement or iterative improvement. In our edge-matching puzzle problem, as the subproblems overlap, partial puzzle patterns provided by the workers can be aggregated to form augmented (partial) puzzle patterns when they agree on the overlapping components. This problem can be solved as a constraint satisfaction problem, considering the constraints of the puzzle, using a local search method or a complete solver, if the problems are small.

Candidate selection: A subset of the augmented candidate solutions is selected, such that they are mutually consistent and jointly maximize the amount of information they provide. In our case, a combinatorial solver selects a consistent subset of the augmented (partial) puzzle patterns so as to cover the entire puzzle area as much as possible. If the selection procedure fails (e.g., due to detected inconsistency), we can ask for new human input, potentially re-decomposing the problem. See Fig. 1.

Solution Procedure: We run a complete solver, using the selected partial solutions as either constraints or initial state. When no solution is found, we return to a previous step and exploit the information accompanying failure. Indeed, constraint solvers can identify parts of the user feedback that caused inconsistency. As described above, the edge-matching puzzle problem can be encoded as a SAT problem. For complete SAT solvers the basic underlying solution strategy is backtrack search, enhanced by several techniques, such as non-chronological backtracking, fast pruning and propagation methods, nogood (or clause) learning, and randomization and restarts. In addition, modern complete SAT solvers can provide an UNSAT certificate when a contradiction is detected. In the case of the edge-matching puzzle problem, an UNSAT certificate corresponds to identifying which of the selected candidates (i.e. partial puzzle patterns) were responsible for the contradiction. This information can be used to eliminate the UNSAT set of clauses from the set of selected partial solutions provided to the final solver, select a different subset, or generate additional

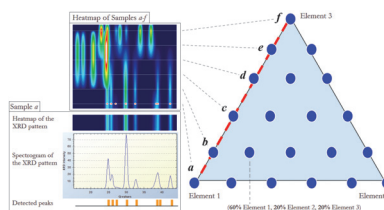


Figure 3: Problem input. A set of samples (blue dots) in composition space, i.e. for different composition of the elements 1, 2 and 3. Each sample is characterized by an X-ray diffraction (XRD) pattern, representing the beam intensity as a function of the diffraction angle, as well as a list of detected peaks. The XRD pattern can be represented as a spectrogram or a heat map, i.e. a color-coded representation of the pattern. The heat map can conveniently represent a list of samples, where the heat maps of the samples are stacked and interpolated between the samples.

subproblems for human workers.

Motivating Application in Materials Discovery

Many industrial and technological innovations, from steam engines to silicon circuits and solar panels, have been enabled through the discovery of advanced materials. Accelerating the pace of the discovery cycle of new materials is essential to fostering innovative advances, improving human welfare and achieving sustainable development.

In order to effectively assess many candidate materials, materials scientists have developed high-throughput deposition techniques capable of quickly generating mixture of chemical elements referred to as composition-spread libraries (Takeuchi, Dover, and Koinuma 2002). Once synthesized, the promising libraries are characterized through X-ray diffraction and fluorescence (Gregoire et al. 2009). The goal of this characterization is to map the composition and the structure of each library. This is called the *phase map identification problem* and is the motivating application of our work. This problem aims to provide composition and structure maps that can be correlated with interesting physical properties within an inorganic library, such as conductivity, catalytic properties or light absorbency. Solving this problem remains a laborious time-consuming manual task that relies on experts in materials science. The contribution of this work is to propose a principled approach to solving this problem, accelerating the pace of analysis of the composition libraries and alleviating the need for human experts.

Fig. 3 depicts the input data of the problem. A sample (a blue dot) corresponds to a given material composition of different chemical elements (Elements 1, 2, and 3). In Fig. 3, the samples uniformly cover the composition space (i.e. the simplex formed by the 3 elements), yet no assumption is made about the distribution of the samples in composition space, i.e., the actual percentage of each material in each sample. In addition, each sample is characterized by an X-ray diffraction (XRD) pattern (bottom-left), which can be represented either as a spectrogram or a heat map. The x-axis corresponds to the angle of diffraction of the X-rays,

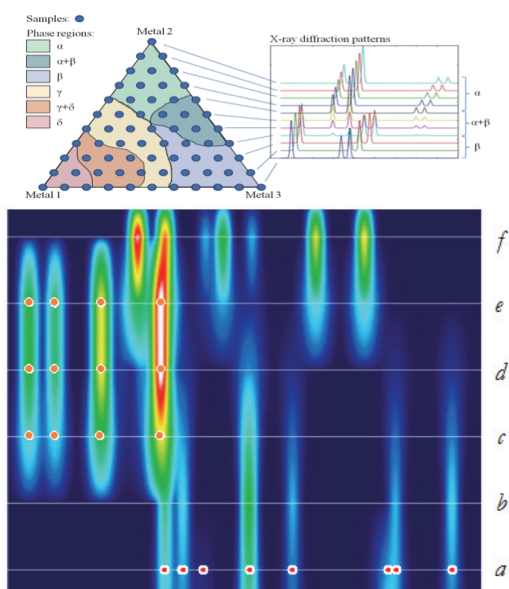


Figure 4: Top: Pictorial depiction of pure phase regions and mixture phase regions in composition space (Left) and XRD patterns of one slice as stacked spectrograms (Right). Bottom: Pictorial depiction of phases for a single slice. The orange dots on the samples c-d-e represent peaks of the X-ray diffraction curves, and characterize a phase, as the 4 left-most vertical blobs vary similarly in intensity on the slice a-f. Overall, this phase has 4 X-ray diffraction peaks and spans the samples c-d-e.

while the y-axis (spectrogram) or the color (heat map) indicate the intensity of the diffracted beam. As described below, in order to visualize multiple XRD patterns at the same time, the heat maps of different samples can be combined by stacking the individual heat maps and interpolating the intensities in between them. Moreover, each X-ray pattern is characterized as a list of peak locations in the diffraction curve.

In the *phase map identification problem*, given the observed diffraction patterns of a composition-spread library, the goal is to determine a set of phases that obey the underlying crystallographic behavior and explain the diffraction patterns. From a materials scientist’s perspective, a phase corresponds to a specific crystal structure, as a particular arrangement of atoms and characterizing lattice parameters. From the aggregation algorithm’s standpoint, a phase is characterized by a set of X-ray diffraction peaks as well as the set of samples where the phase is involved. From a worker’s perspective, a phase is simply a visual pattern of vertical lines/blobs that behave in a similar fashion. See Fig. 4.

From an algorithmic point of view, the *phase map identification problem* can be defined as follows:

Given A set of X-ray diffraction patterns representing different material compositions and a set of detected peaks for each pattern; and K , the expected number of material phases present.

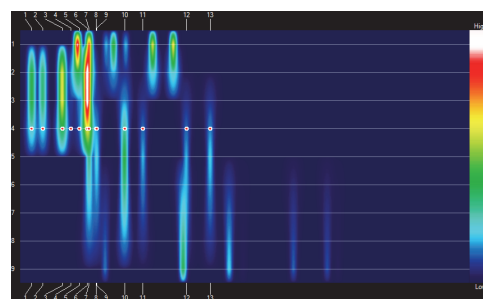


Figure 5: Example of a starting point of a HIT. This heat map represents the beam intensity of 9 sample points (1 through 9), and the intensities in between samples is obtained by interpolation according to the color scale on the right. A worker is asked to identify patterns of similar vertical lines that intersect with sample 4 (whose detected intensity peaks are marked with red dots). We note that another HIT could include the same set of samples, but the worker would be asked to identify patterns of similar vertical lines that intersect with another sample, say sample 7 (in which case its detected intensity peaks would be marked with red dots).

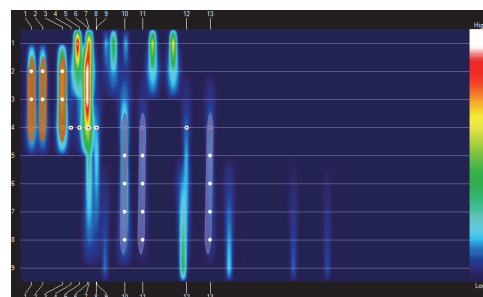


Figure 6: Example of a completed HIT. The three leftmost vertical lines are marked in orange as one pattern spanning three sample points upwards (2 through 4), while three more towards the right are marked in blue as a separate pattern, spanning five sample points downwards (4 through 8). The others are less clear and because of their ambiguity have been left unmarked by the worker, which is the correct decision, since workers are told to be conservative. Contrast this figure with Fig. 4 in which the phase in the top left of the picture includes the fourth vertical line on the right as revealed by a complete solver.

Find A set of K phases characterized as a set of peaks and the sample points they are involved in satisfying the physical constraints that govern the underlying crystallographic process.

Decomposition and Crowdsourcing

Decomposition The decomposition we used to generate Human Intelligence Tasks (HITs) for the phase map identification problem is based on three main criteria: 1) the solutions to subproblems should provide insight into the global solution, 2) each subproblem must be tractable for a human worker, and 3) the number of subproblems for sufficient

problem coverage should scale reasonably.

Domain experts try to solve this problem typically by looking for common patterns in each XRD signal that can be traced through the different sample points. Therefore our subproblems consist of the entire XRD curves for a subset of the sample points.

In order to make the subproblems tractable for human workers, we generate HITs that exploit human visual pattern recognition, and tend to present as few extraneous variations in these patterns as possible. The patterns in the XRD signals that are structurally important tend to change linearly along gradients in the chemical composition, so we choose subsets of sample points that are close to a linear slice through composition space. For example, in Fig. 3, the samples $a-f$ correspond to a slice going through samples a,b,c,d,e,f . Overall, this linear slice defines a chain of adjacent sample points. Namely, it represents a totally-ordered sequence of X-ray diffraction curves where the visual patterns tend to vary continuously along that sequence. We generate a heat-map representation of each such slice so that the patterns are more visually apparent than in alternate representations, such as plots or a discrete representation of detected signal peak locations. Note that the y -axis of the heat map represents the sample points and is therefore categorical, and the beam intensity value in between samples is obtained by interpolation. While this interpolation smooths the heat map in the y -direction and may introduce some visual artifacts, it allows, on the other hand, to better perceive unexpected variations in diffracted beam intensity. In any case, the order of the X-ray diffraction curves on the heat map is key. Indeed, when considering linear slices in composition space, the patterns are expected to vary continuously on the heat map, and they become more apparent than in alternate representations.

With this decomposition structure, the number of possible tasks is cubic in the number of sample points, because assuming a fixed distance threshold, there is at most one unique slice for each pair of sample points, and to guarantee solution coverage, we generate a separate HITs for each sample point in the slice. In practice, we only use slices with orientations that exactly intersect many points, resulting in a number of HITs that scales quadratically.

Human Intelligence Tasks In each HIT (Fig. 5), the worker is presented with a heat-map image representing a single slice, with horizontal grid lines marking the signal for each sample point that is included. *The worker is instructed to look for patterns involving one particular sample point, whose detected peaks are marked with red dots (the target line).* The perceptual challenge is to identify what vertical blobs constitute a pattern, and this is the main focus of a 40-minute tutorial and qualification test (available at <http://www.udiscover.it>). The task is then to select a representative set of the peaks (vertical lines) belonging to one pattern, and to stretch the selection to indicate the adjacent sample points where it is present as well as the variation that occurs across sample points (Fig. 6). The worker repeats this process for a second and third patterns, if they exist. Again,

in this paper, we refer to the set of peaks selected for one particular pattern, including their locations at adjacent points, as a *partial solution* or *partial phase*. Indicating the relationship of a partial phase across adjacent sample points is key, providing a basis for aggregating responses from different HITs and slices.

We control quality in several ways. Workers are required to complete a tutorial and qualification test that covers the rules that describe valid patterns, the interface, and response criteria. Workers are also instructed to be conservative in their submissions, excluding more ambiguous patterns or peaks within those patterns; this helps to reduce the contradictions within and among responses. The user interface itself enforces many of the structural properties that result from the underlying physics of the problem, increasing the chances that each individual response is at least internally feasible. Finally, each HIT is assigned to at least 5 workers. In addition, once the final solution is identified, submissions can be evaluated based on correctness according to this solution; in the event that no acceptable solution is found, responses contributing to the failure can be similarly evaluated. This information allows us to directly evaluate the performance of the workers.

Aggregation and Selection

In this section, we provide the intuition behind the aggregation and the selection steps. We refer the reader to Appendix A for the formal definitions.

Informally, as illustrated in Fig. 4, we define a *peak* as a set of pairs (sample, diffraction angle). For example in Fig. 4, the leftmost vertical blob spanning the samples c-d-e corresponds to a peak that covers 3 samples. Moreover, we define a *phase* as a set of peaks involved in the same sample points (again, see Fig. 4). Finally, a *partial phase* refers to a subset of the peaks of a phase and/or a subset of the sample points.

We translate the output of each HIT into a set of partial phases. Namely, each worker has provided up to 3 partial phases on the samples of the slice of the HIT. Formally, suppose $B = B_1 \cup B_2 \cup \dots \cup B_L$ is the set of all (partial) phases identified by the workers, where B_i is the set of (partial) phases identified in task i . In addition, let K be a positive integer representing the target number of phases.

Intuitively, the aggregation step takes as input the responses B from all workers, and generates a set \bar{B} of augmented phases, while the candidate selection step extracts a subset of K partial phases from \bar{B} , in order to feed and boost the combinatorial solver.

The key intuition is that many partial phases can be combined into larger ones. Figure 7 provides an example. Basically, two phases A and B may be combined into a new phase C , which contains the subset of peaks from A and B , whose diffraction angles match across all the sample locations they both belong to. For this combination operator, we use the notation $C = A \circ B$. Note even though C contains a subset of peaks from A and B , the peaks in C span across the union of all sample points in A and in B . Therefore, we can use this combination operator to extend one partial phase to a larger set of sample locations. We also denote \bar{S}

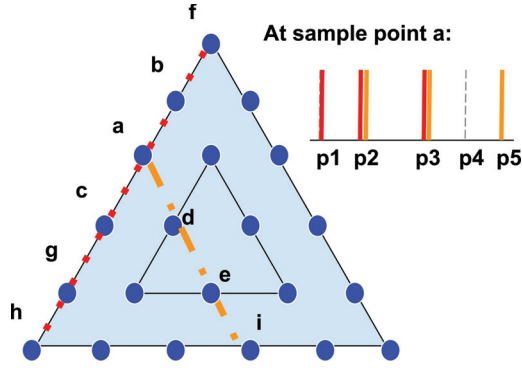


Figure 7: An example showing two partial phases that can be combined into an augmented candidate phase. Suppose worker A is given the indicated slice between sample points f and h , and worker B is given the one between sample points a and i . Worker A annotates phase P_A , which spans over sample points a, b, c , and its peaks at sample point a are p_1, p_2, p_3 . Worker B annotates phase P_B , which spans over sample points a, d , and has peaks p_2, p_3, p_5 at sample point a . The responses from worker A and B can be combined into an augmented candidate phase P_C , which spans sample points a, b, c, d , and has peaks p_2 and p_3 . P_C contains all the peaks (p_2 and p_3) from P_A and P_B whose diffraction angles match across sample locations that P_A and P_B both belong to (sample point a).

the closure of a set of phases S according to that operator, which generates all possible combinations of the phases in S .

Therefore, we define the *aggregation problem* as:

- **Given:** $B = \{B_1, B_2, \dots, B_L\}$,
 - **Find:** \bar{B} ;
- and the *candidate selection problem* as:
- **Given:** \bar{B} , and a positive integer K ,
 - **Find:** K phases from \bar{B} , that are mutually compatible (defined in Appendix A) and maximize the objective $Obj_{sel}(P_1, \dots, P_K)$, where the objective function $Obj_{sel}(P_1, \dots, P_K)$ corresponds to the total number of observed X-ray diffraction peaks explained by the set of selected phases P_1, \dots, P_K .

Because we are aggregating thousands of workers' input, \bar{B} is a large space and we are unlikely to be able to enumerate all items in \bar{B} to find an exact solution. Instead, we first expand B to a larger set $B' \subseteq \bar{B}$ using a greedy algorithm. Then we employ a Mixed-Integer Program (MIP) formulation that selects K phases from B' , covering the largest number of X-ray diffraction peaks, to be given to a complete solver. The following two subsections provide details on the greedy expansion algorithm and the MIP encoding.

Aggregation Algorithm: Greedy Expansion

The greedy algorithm is shown in algorithm 1. The algorithm works by repeatedly calling a subroutine,

$\text{Expansion}(B, P_0, t)$, each time with a different initial phase P_0 from B . Each time the subroutine Expansion generates a set of new phases, which is merged into the final set B' . The subroutine Expansion starts with one initial partial phase P_0 , and iteratively combines the current phase with another phase P_{next} in B , to generate new phases (lines 5 and 6).

Note, after combination, the number of peaks in P either stays the same or monotonically decreases. However, the new phase after combination in general expands to more sample points. In order to keep it tractable for the selection procedure, the algorithm does not collect all the new phases encountered along the way: it only collects the first phase it encounters for the same set of peaks (lines 7 and 8). The cardinality of P , denoted $|P|$, in algorithm 1 measures the number of peaks in phase P .

There is still one question left, which is how to select the next phase P_{next} from B to be combined with the current phase (line 4). Experimentally, we find it is often better to favor those phases that result in the largest number of total peaks after combination, while keeping certain degree of randomness. Hence, we select the next phase with probability proportional to $\text{Softmax}(Obj_{sel}(P), t) = e^{-t \cdot (|P| - |P_{next} \circ P|)}$. In our experiment, the temperature t is set to take value 3.

Algorithm 1: The greedy expansion algorithm.

Data: phase set B .

Result: An extended set of phases $B' \supseteq B$.

- 1 $B' \leftarrow \emptyset$;
 - 2 **for** $P \in B$ **do**
 - 3 | $B' \leftarrow B' \cup \text{Expansion}(B, P, t_0)$;
 - 4 **end**
 - 5 **return** (B')
-

Function $\text{Expansion}(B, P_0, t)$

Data: phase set B ; A phase $P_0 \in B$ as the starting point; temperature t ;

Result: An extended set of phases $B' \supseteq \{P_0\}$.

- 1 $P \leftarrow P_0$;
 - 2 $B' \leftarrow P_0$;
 - 3 **while** $|P| > 0$ **do**
 - 4 | select P_{next} from B with probability proportional to $e^{-t \cdot (|P| - |P_{next} \circ P|)}$;
 - 5 | $P_{prev} \leftarrow P$;
 - 6 | $P \leftarrow P \circ P_{next}$;
 - 7 | **if** $|P| < |P_{prev}|$ **then**
 - 8 | | $B' \leftarrow B' \cup \{P\}$;
 - 9 | **end**
 - 10 | $B \leftarrow B \setminus \{P_{next}\}$;
 - 11 **end**
 - 12 **return** B'
-

Candidate Selection Algorithm: MIP Formulation

After obtaining the extended set B' from the previous greedy phase, we use a Mixed-Integer Program (MIP) encoding to select a set Y of K phases in B' that covers as many peaks as possible. We briefly outline the MIP formulation in this section, while Appendix B provides a detailed formal definition.

Firstly, we enforce that we select exactly k elements from B' and that any element of B' can be selected at most once. Secondly, at thermodynamical equilibrium, the underlying crystallographic process requires that no more than three phases appear in any single sample point. This is obtained by limiting the number of selected phases among the ones involved in any given point. Finally, the objective of the MIP is to minimize the number of unexplained peaks, which can be modeled by counting how many peaks are left uncovered by the selected phases Y .

Solving and Feedback

In this section, we describe how we boost a combinatorial solver using the output from the selection task described in the previous section, namely the selected partial phases Y . We first formulate the *phase map identification problem* as a Satisfiability Modulo Theories (SMT) model, following the description provided in (Ermon et al. 2012; Le Bras et al. 2013). Basically, the main variables represent the locations of the peaks in the phases, and their domain corresponds to a discretized space of diffraction angles. Then, quantifier-free linear arithmetic theory is used to model the behavior of the underlying crystallographic process. In addition to that encoding, we formally translate the output of the aggregation step into additional constraints. Namely, we formulate each selected partial phase $Y_i \in Y$ as an additional hard constraint $\phi(Y_i)$, where $\phi(Y_i)$ forces the partial assignment of the i^{th} phase to Y_i . Moreover, we add an indicator variable z_i corresponding to whether we impose this selected partial phase when solving the overall problem. Namely, we define: $z_i \rightarrow \phi(Y_i)$. Therefore, when assuming all selected partial phases, we impose that $\bigwedge_{i=1}^K z_i$ holds true.

In order to solve the problem we use a complete SMT solver. The basic underlying solution strategy is backtrack search enhanced by several sophisticated techniques, such as fast pruning and propagation methods, nogood (or clause) learning, and randomization and restarts. An important issue that we have to consider is potentially inconsistent or erroneous input in the worker submissions. If we identify the inconsistent input, we can relax the corresponding z_i propositions. State-of-the-art combinatorial solvers can provide an unsat core whenever an instance is unsatisfiable. In that case, the solver would return a subset of the assumptions $\{z_i | i = 1..K\}$ that led to the contradiction. Partially removing elements in that set either leads to a satisfiable instance, or allows us to generate a new unsat core. This approach is formally described in Algorithm 2. The lines 5-13 identify, within the unsat core U , the smallest subset of assumptions Z that has not yet been considered. In line 15, the program rechecks the feasibility after removing the assumptions Z

from the original set Y . Overall, this process starts by removing a single phase from the unsat core. If every single phase in the unsat core has been considered individually, the algorithm attempts to remove pairs of phases, and so on, until it finds a correct set of phases whose removal leads to a feasible solution.

Algorithm 2: The unsat core-based solving algorithm.

```
Data: problem instance  $X$ ; preselected phases  $Y$   
Result: selected phases  $P$ ; solution  $S$  of  $X$   
1  $Z \leftarrow \emptyset$ ;  $C \leftarrow \{\emptyset\}$ ; // Conflicts  
2 assert( $X$ );  
3 status = check( $Y$ );  
4 while status == unsat do  
5    $U = \text{unsatcore}()$ ;  
6   for  $k = 1..|Y|$  do  
7     for  $Z \subseteq U \wedge |Z| = k$  do  
8       if  $Z \notin C$  then  
9          $C = C \cup Z$ ;  
10        goto next;  
11       end  
12     end  
13   end  
14   next;  
15   status = check( $Y \setminus Z$ );  
16 end  
17  $P = Y \setminus Z$ ;  $S = \text{getsolution}()$ ;  
18 return ( $P, S$ )
```

Note that these iterative steps benefit from an incremental solving inside the combinatorial solver, as any constraint reasoning that was performed on the initial assertions (the problem instance) remains valid at each iteration, and limits the overhead of keeping track of the unsat core.

Interestingly, once we solve the problem, we obtain as a by-product the accuracy of each worker submission. Indeed, we can evaluate the quality of each input by cross-checking the partial phases in the submission with the overall solution of the problem. This opens up new research directions in terms of user feedback and contribution-based rewards to the workers.

Experimental results

In this section, we provide empirical results of our approach on the phase map identification problem.

The instances we use for evaluation are synthetic instances that were generated as described in (Le Bras et al. 2014). The advantage of synthetic instances is that they allow us to compare the results with the ground truth, and evaluate the proposed approach. Nevertheless, the generated instances are based on a well-studied real physical system, and comparable in size and complexity to the real systems. For our experiments we considered 8 different systems (all of them involving three different metals): A1, A2, B1, C1, C2, C2, C3, D1, and D2.

In terms of the crowdsourcing tasks, we used the Amazon Mechanical Turk (AMT) platform to recruit workers.

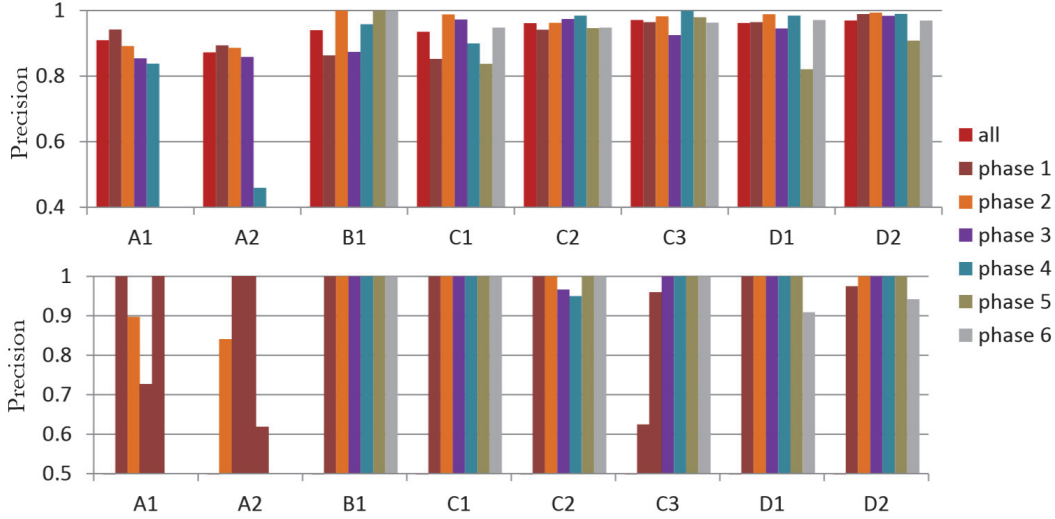


Figure 8: Quality of the workers’ submissions and of the aggregated phase by system (or instance). We considered eight different systems (all of them involving three different metals): A1, A2, B1, C1, C2, C2, C3, D1, and D2. (Upper) The average precision of workers’ submissions towards ground truth phases; (Lower) The precision of the aggregated phases towards ground truth phases. The color of the bars denote the ground-truth phase they are comparing to.

System	Dataset			Solver only	Solver with Human Computation input				
	P	L^*	K	Time (s)	Overall Time (s)	Aggregation (s)	Backtrack (s)	#backtracks	
A1	36	8	4	3502	859	17	300	4	
A2	60	8	4	17345	4377	29	272	2	
B1	15	6	6	79	4	0.07		0	
C1	28	6	6	346	62	0.5		0	
C2	28	8	6	10076	271	4		0	
C3	28	10	6	28170	1163	6	105	1	
D1	45	7	6	18882	596	7		0	
D2	45	8	6	46816	1003	13		0	

Table 1: Comparison of the runtimes of the solver with and without the human component for the different systems. P is the number of sample points, L^* is the average number of peaks per phase, K is the number of basis patterns, $\#var$ is the number of variables and $\#cst$ is the number of constraints.

We provided a 40-minute qualifying test, which also acted as a tutorial (available at <http://www.udiscover.it>). Once a worker passed the test, he was sent to our own interface to complete the actual tasks. We randomly clustered the tasks into groups of 25 tasks. Namely, each time a worker was assigned a task on AMT, he or she had to complete 25 tasks on our interface before getting the confirmation code to feed back to the AMT website. The average completion time per task was approximately 15 seconds, and a worker was rewarded \$1 for completing a group of 25 tasks. Thus, the average rate was about \$9.6 per hour, and contributed to the high worker attrition in the experiment. We generated 660 tasks, and each one had to be completed by 5 different workers. Overall, the tasks were performed by the active participation of 25 human workers. Indeed, on average, a worker completed over 20% of all the tasks that were available to him/her, and all the tasks were completed within 48 hours after being posted on AMT. In the following, we study the quality of the workers and of the output of the aggregation

step, and evaluate how it effectively speeds up the combinatorial solver.

First of all, we study the quality of the workers input, as shown in Fig. 8. We use the average precision as a measure of quality. Formally, the precision of one phase P relative to a ground truth phase Q is defined as the percentage of the number of peaks in P which also belong to Q , i.e. $prec(P, Q) = \frac{|P \cap Q|}{|P|}$. The upper chart of figure 8 shows the average precision of a submitted phase relative to a ground truth phase, which is denoted by

$$prec(Q) = \frac{1}{|P \in sub, P \cap Q \neq \emptyset|} \sum_{P \in sub, P \cap Q \neq \emptyset} prec(P, Q).$$

where sub denotes the set of submitted phases and the all bar denotes the average precision of a submitted phase relative to any ground truth phase:

$$prec_{all} = \sum_{Q \in ground.truth} \frac{|P \in sub, P \cap Q \neq \emptyset|}{\sum_Q |P \in sub, P \cap Q \neq \emptyset|} prec(Q).$$

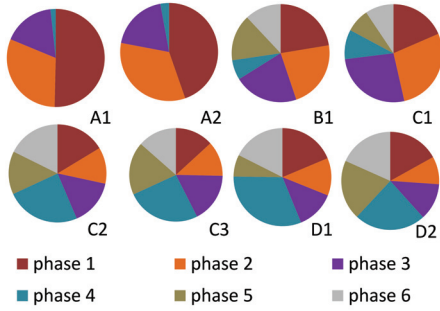


Figure 9: Percentage of workers’ annotations that belong to each ground truth phase, for the different systems (A1, A2, B1, C1, C2, C2, C3, D1, and D2). The percentage related to ground truth phase Q is defined as $\frac{|P \in \text{sub}, P \cap Q \neq \emptyset|}{\sum_Q |P \in \text{sub}, P \cap Q \neq \emptyset|}$. We use the same color as in Fig. 8 for the ground truth phases.

As we can see, human workers provide very high quality input. Many partial phases identified by the workers belong to the actual ground-truth phase as a whole. The average precision score is almost always greater than 90%. More importantly, the phases identified by human workers span nicely over all phases in the ground truth set, except for the A2 instance. Note this allows the later aggregation step to pick up all the K phases.

Regarding the low precision for one phase in the A2 system, it is worth noting that the peaks of that phase have relative low visual intensity. Since the workers were explicitly told in the description of the tasks to only annotate peaks they are most confident about, it is natural that most workers skipped annotating any peak for that phase. A similar yet less obvious problem occurs in A1. Figure 9 illustrates the percentage of workers submissions that belong to each phase in the ground-truth. We observe very few annotations of the fourth phase for A1 and A2.

Next we study the quality of the output of the aggregation step. The lower chart of figure 8 shows the precision of the K aggregated phases relative to the K ground-truth phases. As shown, in most instances the aggregation algorithm is able to select all ground truth phases, except for the instance of A1, A2 and C3. For these 3 systems, the aggregation algorithm selects a partial phase of the same ground truth phase multiple times. Note that A1, A2 and C3 actually correspond to the cases where the combinatorial solver needs to backtrack in order to find the complete solution, due to inconsistent human input. In order to understand why these 3 systems seemed more problematic, we analyzed the ground truth solutions of these problems. In addition to the low visual intensity case in A1 and A2, several ground truth phases in A1 and C3 have many overlapping peaks, which confused the workers during the annotation, as well as the aggregation solver which falsely combined two distinct phase into one.

Finally, we study the speed-up of applying the aggregated information to the complete solver. We feed the aggregated phases into the SMT solver, and study the time the solver

takes to find a complete solution with and without initialization from human input. The result is shown in table 1. We run on a machine running 12-core Intel Xeron5690 3.46GHz CPU, and 48 gigabytes of memory. As shown in this table, aggregated crowdsourced information dramatically boosts the combinatorial solver. When considering the largest instance D2 for example, *the solver cannot find the solution in 13 hours without human input, while it only takes about 15 minutes with human input.*

We observe that the solver can find the complete solution in 5 out of 8 instances, including the biggest two D1 and D2, without even having to reconsider the choices made by the selection algorithm. In these 5 instances, the speed-up corresponds to several orders of magnitude. Furthermore, a worker typically spends approximately 15 seconds annotating one slice. On the eight instances we solve, the number of tasks per system ranges from 34 (B1) to 172 (D2). Thus even if one human worker takes over the entire job of annotating phases for one system, it takes him or her less than 45 minutes to annotate, which is relatively small compared to the original solving time of 13 hours.

On the other 3 instances, we need to backtrack, due to inconsistent human input, in order to find the complete solution. Nevertheless, the overhead of computation time due to the backtracks is relatively small, which suggests that finding inconsistencies among workers input is relatively easy. In any case, it still performs at least an order of magnitude better in terms of computation time. The reason for backtracking is mainly due to the fact that the visual clue is not enough for the human worker to tell apart between two distinct phases, which in turn confuses the aggregation. Nonetheless, the solver is able to find out the complete solution with minimal number of backtracks.

Conclusions

In this paper, we propose a framework that combines state-of-the-art optimization solvers based on constraint reasoning with a human computation component. The experimental results show that this hybrid framework reduces our analysis time by orders of magnitude compared to running the constraint solver without human input, when applied to an important materials discovery problem.

Overall, we show that human computation and fully automated constraint optimization approaches can work in a hybrid complementary setting, leading to significant overall performance gains. Our decomposition and solution aggregation strategy provides a general framework for decomposing large, complex tasks into units suitable for human feedback. Moreover, the solver provides feedback on the human input and allows us to resample and correct for “noise” from the human feedback.

Future research directions include active learning methods to identify, in an online setting, which human tasks need to be performed, as well as to provide feedback to the workers, actively evaluate their quality and provide incentives based on worker accuracy.

Acknowledgments

The authors are grateful to the anonymous reviewers for their positive feedback and constructive comments. We thank Shuo Chen for helping us improve the readability of the original version of this paper. This work was supported by the National Science Foundation (NSF Expeditions in Computing award for Computational Sustainability, grant 0832782, NSF Inspire grant 1344201 and NSF Eager grant 1258330). The experiments were run on an infrastructure supported by the NSF Computing research infrastructure for Computational Sustainability grant (grant 1059284).

Appendix A

We first formally define the notions of peaks and phases.

Definition (peak) A peak q is a set of (sample point, location) pairs: $q = \{(s_i, l_i) | i = 1, \dots, n_q\}$, where $\{s_i | i = 1, \dots, n_q\}$ is a set of sample points where q is present, and l_i is the location of peak q at sample point s_i , respectively. We call $\{s_i | i = 1, \dots, n_q\}$ the realization set of q , denoted as $Rel(q)$. We also use $q(s_i)$ to denote l_i – the peak location at s_i .

When non-ambiguous, we also use the term “peak” to refer to a particular realization of a peak within one sample point as well. A phase corresponds to a set of peaks that share a common realization set, subject to certain physical constraints on their variation. Formally, we define:

Definition (phase) A phase (or a partial phase) P is composed of a set of peaks $\{q_1, q_2, \dots, q_{n_P}\}$, with all peaks q_j sharing a common realization set S . We call S the realization set of P , denoted as $Rel(P)$. $|P|$ is used to denote the number of peaks in a phase.

We use lower-case letters p, q, r, \dots to represent peaks, and use upper-case letters P, Q, R, \dots to represent phases.

Suppose P and Q are two partial phases whose realization sets intersect, $\{p_1, \dots, p_k\}$ are matched peaks between P and Q on all the sample points where P and Q co-exist. It is not hard to see that peaks $\{p_1, \dots, p_k\}$ form a new valid phase which spans over $Rel(P) \cup Rel(Q)$. Based on this idea, we can formally define the combination of two peaks and two phases as follows.

Definition (Combination of peaks) Suppose we have peak $q_1 = \{(s_i^1, l_i^1) | i = 1, \dots, n_1\}$ and $q_2 = \{(s_i^2, l_i^2) | i = 1, \dots, n_2\}$. The combination of q_1 and q_2 , denoted as $q = q_1 \circ q_2$, is a peak, and,

- If $Rel(q_1) \cap Rel(q_2) = \emptyset$, then $q_1 \circ q_2 = \perp$.
- Otherwise, $q = q_1 \circ q_2$ is a peak which exists on sample points $Rel(q) = Rel(q_1) \cap Rel(q_2)$, and

$$q(s) = \begin{cases} q_1(s) & \text{for } s \in Rel(q_1) \\ q_2(s) & \text{for } s \in Rel(q_2) \setminus Rel(q_1). \end{cases}$$

We denote \perp a special null peak and define $\forall q : q \circ \perp = \perp$.

Intuitively speaking, the previous definition says if two peaks p and q match on all sample points they coexist, then their combination is the peak that spans over the union of the realization set of the two peaks. In all other cases, $p \circ q = \perp$. Now we define the combination of two phases.

Definition (Combination of phases) Suppose A, B are two phases with peaks $\{q_{A,i} | i = 1, \dots, m\}$ and $\{q_{B,j} | j = 1, \dots, n\}$, respectively. The combined phase C is then:

$$C = A \circ B = \{q' \circ q'' | \forall q' \in A, q'' \in B\} \setminus \{\perp\}.$$

As the combination of two peaks extends the realization set of one peak p to the realization of another peak q if p and q match on their shared points, the combination of two phases extends all peaks from one phase to the matched peaks of the other.

Definition (Closure) For a set of phases $S = \{P_1, P_2, \dots, P_n\}$. The closure of S , denoted as \bar{S} , is defined as the minimal set that satisfies,

- $S \subseteq \bar{S}$.
- For $P_i, P_j \in \bar{S}$, $P_i \circ P_j \in \bar{S}$.

Appendix B

In this appendix, we present the MIP formulation that models the *candidate selection problem*. Let $E_{i,k}$ be binary indicator variables, where $E_{i,k} = 1$ iff the i -th phase in B' is selected as the k -th phase ($i = 1, \dots, n$ and $k = 1, \dots, K$).

The first constraint is that each phase in B' can be selected at most once, i.e.,

$$\forall i, \quad \sum_{k=1}^K E_{i,k} \leq 1.$$

Next, there is exactly one phase from B' that is selected as the k -th phase.

$$\forall k, \quad \sum_{i=1}^n E_{i,k} = 1.$$

Let $O(j) \subseteq B'$ be all the phases existing at sample point j . There is a physical constraint that no more than three phases can be present at one sample point:

$$\forall j, \quad \sum_{i \in O(j)} \sum_{k=1}^K E_{i,k} \leq 3.$$

Let w_l be the binary variable indicating whether peak l is not covered by any selected phase, and $C(l) \subseteq B'$ be the set of all phases that cover peak l . Then the following constraint holds:

$$w_l + \left(\sum_{i \in C(l)} \sum_{k=1}^K E_{i,k} \right) \geq 1 \text{ for all } l.$$

Here, w_l will only be forced to take value 1 if none of the $E_{i,k}$ variables covering l takes value 1. Finally, we want to find the set Y of phases that minimizes the number of uncovered peaks. Namely, we have:

$$Y = \arg \min_{\{B_1, \dots, B_K\} \subset B'} \sum_{l \in L} w_l.$$

References

- Acuña, D. E., and Parada, V. 2010. People efficiently explore the solution space of the computationally intractable traveling salesman problem to find near-optimal tours. *PLoS one* 5(7):e11685.
- Anderson, D.; Anderson, E.; Lesh, N.; Marks, J.; Mirtich, B.; Ratajczak, D.; and Ryall, K. 2000. Human-guided simple search. In *AAAI/IAAI*, 209–216.
- Ansótegui, C.; Béjar, R.; Fernánandez, C.; and Mateu, C. 2013. On the hardness of solving edge matching puzzles as sat or csp problems. *Constraints* 18(1):7–37.
- Bell, M.; Reeves, S.; Brown, B.; Sherwood, S.; MacMillan, D.; Ferguson, J.; and Chalmers, M. 2009. Eyespy: supporting navigation through play. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 123–132. ACM.
- Chen, S.; Zhang, J.; Chen, G.; and Zhang, C. 2010. What if the irresponsible teachers are dominating? a method of training on samples and clustering on teachers. In *24th AAAI Conference*, 419–424.
- Cooper, S.; Khatib, F.; Treuille, A.; Barbero, J.; Lee, J.; Breen, M.; Leaver-Fay, A.; Baker, D.; Popovic, Z.; and players, F. 2010. Predicting protein structures with a multiplayer online game. *Nature* 466(7307):756–760.
- Demaine, E., and Demaine, M. 2007. Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity. *Graphs and Combinatorics* 23(s1):195.
- Dilkina, B.; Gomes, C.; Malitsky, Y.; Sabharwal, A.; and Sellmann, M. 2009. Backdoors to combinatorial optimization: Feasibility and optimality. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems* 56–70.
- Ermon, S.; Le Bras, R.; Gomes, C. P.; Selman, B.; and van Dover, R. B. 2012. Smt-aided combinatorial materials discovery. In *Proceedings of the 15th International Conference on Theory and Applications of Satisfiability Testing (SAT)*, SAT’12, 172–185.
- Fischetti, M., and Monaci, M. 2011. Backdoor branching. *Integer Programming and Combinatorial Optimization* 183–191.
- Gaspers, S., and Szeider, S. 2012. Backdoors to satisfaction. *The Multivariate Algorithmic Revolution and Beyond* 287–317.
- Gomes, R.; Welinder, P.; Krause, A.; and Perona, P. 2011. Crowdclustering. In *Proc. Neural Information Processing Systems (NIPS)*.
- Gregoire, J. M.; Dale, D.; Kazimirov, A.; DiSalvo, F. J.; and van Dover, R. B. 2009. High energy x-ray diffraction/x-ray fluorescence spectroscopy for high-throughput analysis of composition spread thin films. *Review of Scientific Instruments* 80(12):123905–123905.
- Kilby, P.; Slaney, J.; Thiébaux, S.; and Walsh, T. 2005. Backbones and backdoors in satisfiability. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, 1368. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.
- Klau, G. W.; Lesh, N.; Marks, J.; and Mitzenmacher, M. 2002. Human-guided tabu search. In *AAAI/IAAI*, 41–47.
- Law, E., and Von Ahn, L. 2009. Input-agreement: a new mechanism for collecting data using human computation games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1197–1206. ACM.
- Le Bras, R.; Bernstein, R.; Gomes, C. P.; and Selman, B. 2013. Crowdsourcing backdoor identification for combinatorial optimization. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI’13*.
- Le Bras, R.; Bernstein, R.; Gregoire, J. M.; Suram, S. K.; Gomes, C. P.; Selman, B.; and van Dover, R. B. 2014. A computational challenge problem in materials discovery: Synthetic problem generator and real-world datasets. In *Proceedings of the 28th International Conference on Artificial Intelligence, AAAI’14*.
- Lieberman, H.; Smith, D.; and Teeters, A. 2007. Common consensus: a web-based game for collecting commonsense goals. In *ACM Workshop on Common Sense for Intelligent Interfaces*.
- O’Sullivan, B. 2010. Backdoors to satisfaction. tutorial at cp 2010. In *Proceedings of the Principles and Practice of Constraint Programming (CP)*.
- Patel, P. 2011. Materials genome initiative and energy. *MRS bulletin* 36(12):964–966.
- Rossi, F.; Van Beek, P.; and Walsh, T. 2006. *Handbook of constraint programming*. Elsevier.
- Szeider, S. 2006. Backdoor sets for dll subsolvers. *SAT 2005* 73–88.
- Takeuchi, I.; Dover, R. B. v.; and Koinuma, H. 2002. Combinatorial synthesis and evaluation of functional inorganic materials using thin-film techniques. *MRS bulletin* 27(04):301–308.
- Turnbull, D.; Liu, R.; Barrington, L.; and Lanckriet, G. R. 2007. A game-based approach for collecting semantic annotations of music. In *ISMIR*, volume 7, 535–538.
- Von Ahn, L., and Dabbish, L. 2004. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 319–326. ACM.
- Von Ahn, L.; Kedia, M.; and Blum, M. 2006. Verbosity: a game for collecting common-sense facts. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, 75–78. ACM.
- Von Ahn, L.; Liu, R.; and Blum, M. 2006. Peekaboom: a game for locating objects in images. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, 55–64. ACM.
- Von Ahn, L. 2005. *Human Computation*. Ph.D. Dissertation, Pittsburgh, PA, USA. AAI3205378.
- Welinder, P.; Branson, S.; Belongie, S.; and Perona, P. 2010. The multidimensional wisdom of crowds. *Advances in Neural Information Processing Systems* 23:2424–2432.
- White, A. 2012. The materials genome initiative: One year on. *MRS Bulletin* 37(08):715–716.
- Yi, J.; Jin, R.; Jain, A.; Jain, S.; and Yang, T. 2012. Semi-crowdsourced clustering: Generalizing crowd labeling by robust distance metric learning. In *Advances in Neural Information Processing Systems* 25, 1781–1789.
- Zhou, D.; Platt, J.; Basu, S.; and Mao, Y. 2012. Learning from the wisdom of crowds by minimax entropy. In *Advances in Neural Information Processing Systems* 25, 2204–2212.