Crowdsourcing Backdoor Identification for Combinatorial Optimization

Ronan Le Bras[1], **Richard Bernstein**[1]
Carla P. Gomes[1], **Bart Selman**[1], and **R. Bruce van Dover**[2]

[1] Computer Science Dept.
[2] Materials Science and Engr. Dept.
Cornell University, Ithaca, NY

Abstract

We will show how human computation insights can be key to identifying so-called backdoor variables in combinatorial optimization problems. Backdoor variables can be used to obtain dramatic speed-ups in combinatorial search. Our approach leverages the complementary strength of human input, based on a visual identification of problem structure, crowdsourcing, and the power of combinatorial solvers to exploit complex constraints. We describe our work in the context of the domain of materials discovery. The motivation for considering the materials discovery domain comes from the fact that new materials can provide solutions for key challenges in sustainability, e.g., in energy, new catalysts for more efficient fuel cell technology.

1 Introduction

Over the last decade, we have seen dramatic improvements in combinatorial solvers. State-of-the-art mixed integer programming (MIP), satisfiability (SAT) and SAT Modulo Theory (SMT) solvers can handle practical problem instances with up to several hundreds of thousands of variables and up to a million constraints. Such dramatic scale-up has led to a range of new applications, such as bounded-model checking for verification, AI planning, and scheduling. The good performance of current solvers can be traced back to their ability to discover and exploit hidden problem structure in the application instances. The notion of backdoor variables provides useful insights into this phenomenon. Backdoor variables capture much of the practical complexity of problem instances, since after setting backdoor variables, a problem instance simplifies to a tractable subclass [Williams *et al.*, 2003b]. It was 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].

The problem of finding a small backdoor set has been shown to be worst-case intractable. However, in practice, rapid restart techniques and variable selection heuristics enable solvers to find small backdoor sets relatively quickly [Williams *et al.*, 2003a; Gomes *et al.*, 1998]. Intuitively backdoor variables represent critically constrained resources in the original problem formulation. However, in actual problem encodings, it has been difficult to find a clear semantic interpretation of such variables [Hoffmann *et al.*, 2007, e.g.].

In this paper, we present an application domain with an SMT encoding, where we are able to identify useful classes of backdoor variables in a semantically meaningful way. We then show how by setting a subset of such variables, one can speed up the SMT solution process by several orders of magnitude. For example, we present an instance that required 46,816 seconds (13 hours) to solve without explicit backdoor variable information. With such information, a few dozen variables can be pre-assigned, and the SMT solver can solve the remaining instance in 129 seconds. For other results, see table 1.

A key issue is how to obtain information on the backdoor set for a specific problem instance. As we will see, the required backdoor information captures certain global structure of the problem instance. We will show how a human problem solver, familiar with the problem domain, but at a novice level, can identify the backdoor information from a relatively basic visual representation of a problem instance. Once such information is provided to the SMT solver, it solves the instance in 129 seconds. Essential to this process is a global inspection of a series of visual patterns. We subsequently show how this process can be further simplified by dividing up the inspection task into a series of local patterns that can be crowdsourced for inspection on Amazon Mechanical Turk. The Turkers, looking at these visual patterns, have no knowledge of the original combinatorial optimization problem in question. For the instance mentioned earlier, around 10 Turkers, inspecting, on average, 30 patterns each, provided enough information on the backdoor set to reduce the SMT solution time to 350 seconds (down from the original time of 13 hrs), a dramatic speed up of over two orders of magnitude.

This work complements the findings of the seminal FoldIt project [Khatib *et al.*, 2011], which showed how human gamers can find good protein folds, including certain new folds not found with fully automated methods. Our results show that a hybrid human-computer strategy can provide yet further speed ups, outperforming pure human and computer strategies. In our approach, the concept of backdoor variables guides the design of such a hybrid strategy. The decomposition of the task of finding useful backdoor variables into a sequence of smaller "local" problems (each solvable in about 30 seconds by non-experts) enabled us to crowdsource the discovery process. Interestingly, this approach also is reminiscent of some of the earliest uses of "human computation," where groups of individuals (mostly women) performed many relatively small calculations that were then put together to obtain a global overall result [McLennan and Gainer, 2012]. Of course, those original calculations are now trivially automated. In contrast, in our work, it is an interesting research question whether certain visual processing methods combined with machine learning tools could possibly replace the tasks performed by the Turkers. Overall though, it seems likely that our hybrid human-computer strategy for combinatorial optimization will be of use in a range of combinatorially hard application domains.

Below we will first describe our computational sustainability application: a challenge problem in combinatorial materials discovery. The overall goal is the analysis of high-intensity X-ray images to search for new crystalline phases of inorganic compounds. Given that our challenge domain involves a real-world application, the details of the problem and the SMT model we developed are rather involved. Our description below is meant to provide sufficient detail for the reader to understand the domain sufficiently well to appreciate the hybrid human-computer experiments that follow.

2 Motivating Application

Combinatorial materials discovery involves the rapid, high-throughput synthesis, measurement, and analysis of a large number of different but structurally related materials. In combinatorial materials discovery, materials scientists search for intermetallic compounds with desirable physical properties by obtaining measurements on hundreds of samples from a *thin film* composition spread. This approach has been successfully applied for example to speed up the discovery of new materials with improved catalytic activity for fuel cell applications [Van Dover *et al.*, 1998; Gregoire *et al.*, 2010]. Determining the structure of the materials (or *phase map*) formed in a composition spread is key to understanding composition and property relations and can potentially result in a breakthrough discovery. In the set-up we consider in this paper, scientists run several experiments at the Cornell High Energy Synchrotron Source (CHESS) for about one week per year (at an experimentation cost of about $1M) and spend the rest of the year analyzing the data. The goal is to reduce the processing time of much of the data interpretation task to a timeframe of hours. Such rapid analysis will enable scientists to dynamically optimize their experiments over the days that they have access to the synchrotron, thereby reducing overall experimentation time and significantly accelerating the discovery cycle.

The motivation for considering the materials discovery problem comes from the fact that new materials provide a fundamental basis for solutions to some of the most pressing issues in energy generation, transport, and utilization as well as more general issues in sustainability. In many cases, long-term solutions will depend on breakthrough innovations in materials, such as the development of new materials for more efficient fuel cells, solar cell arrays, or for wind turbines.

Combinatorial materials discovery, in particular the problem of ternary phase-field identification addressed in this paper, provides unique computational and modeling challenges. While statistical methods and machine learning are important components to address this challenge, they fail to incorporate relationships that are inherent due to the basic physics and chemistry of the underlying materials. In fact, a successful approach to materials discovery requires *a tight integration of* ***statistical machine learning methods****, to deal with noise and uncertainty in the measurement data*, and ***optimization and inference techniques****, to incorporate a rich set of constraints arising from the underlying materials physics and chemistry.* [Ermon *et al.*, 2012] showed that a constraint reasoning and optimization approach, as performed by a state-of-the-art Satisfiable Modulo Theory (SMT) solver, can effectively solve small- to medium-scale synthetic instances.The challenge we consider here is how to significantly scale up this approach, including the significant measurement noise level present in real-world data. In particular, our ultimate objective is to obtain an analysis turn-around time of under 12 hours. This would enable us have the analysis guide further experiments during the time period scheduled for experimentation.

The broader underlying question that we consider is whether human input can be used to significantly boost the performance of combinatorial reasoning and optimization methods. The project is close in spirit to the seminal FoldIt project [Cooper *et al.*, 2010] for protein folding. In FoldIt, human computation is the main driving force, complemented with a limited amount of local computation (e.g., "shaking" of structures). We are proposing a much tighter integration between our computational framework and the human computation [Law and von Ahn, 2011] component. In our approach, the human input and the SMT solver are highly complementary: the complex local physical constraints require a sophisticated optimization approach, whereas the global human insights are used to guide the solver. In particular, we will show how we can boost the performance of the SMT solver by providing additional information from human input. The task at hand, which involves the in-

terpretation of complex high-intensity X-ray diffraction patterns, appears to be well-suited for a human computation approach. As we will see, the human input provides useful *global guidance* to the solver, by identifying the setting of backdoor variables in the SMT model, critical variables that when assigned a value, enable highly efficient constraint reasoning and inference, leading to orders of magnitude speedups for the SMT solver. Overall, the results show that our hybrid human-computer approach presents us with unique opportunities for tackling hard combinatorial optimization challenges.

3 Problem Description

In the composition spread approach, a *thin film* is obtained by depositing three metals onto a silicon wafer using guns pointed at three distinct locations (see Fig. 1). Different locations (or samples) on the thin film correspond to different concentrations of the sputtered materials, based on their distance to the gunpoints. X-ray diffraction (XRD) is then used to characterize a number of samples on the thin film. For each sample point, it provides the intensity of the electromagnetic waves as a function of the scattering angle. The observed diffraction pattern is closely related to the underlying crystal structure, which provides important insights into the chemical and physical properties of the corresponding composite material.

The goal of the *phase-map identification problem* is to identify regions of the thin film that share the same underlying crystal structure. Intuitively, the XRD patterns observed across the *thin film* can be explained as combinations of a small set of basis patterns called *phases*. Finding the phase map corresponds to identifying these *phases* as well as their concentration on the thin film. The main challenge is to model the complex crystallographic process that these phases are subject to (such as the expansion of the lattice, which results in a 'scaling' of the XRD pattern), while taking into account the imperfection of the silicon wafer as well as experimental noise of the data.

While it is natural to study the phase-map identification problem on the basis of full XRD curves, constructive interference of the scattered X-rays occurs, by nature, at *specific* angles and creates spikes (or *peaks*) of intensity. In addition, experimental noise combined with variations of the Silicon substrate make the measured intensity of the beam unreliable. As a result, materials scientists mostly rely on peak angles when tackling the phase-map identification problem.

Similarly, we adopt a peak-based approach (as presented in [Le Bras *et al.*, 2011]) in which we use a peak detection algorithm to extract the angles of the *peaks* $\mathcal{Q}(p)$ in the XRD pattern of a point p. The goal is then to find a set of peaks $\{\mathcal{E}_k\}_{k=0}^{K-1}$, coefficients $a_{p,k} \in \mathbb{R}$, and scaling factors $s_{p,k} \in \mathbb{R}$ for K basis patterns that can explain the observed sets of peaks $\{\mathcal{Q}(p)\}_{p=0}^{P-1}$. The scaling factor models the potential expansion of the crystal lattice with changing

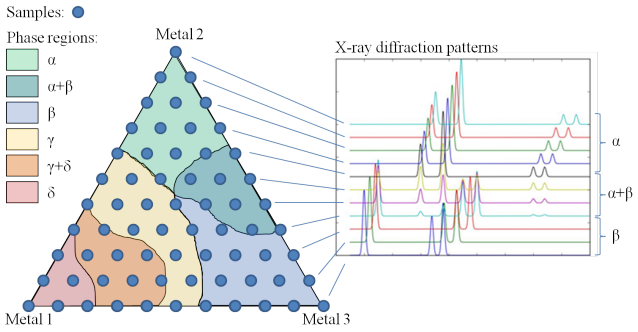

Figure 1: Left: Depiction of the problem, showing a set of sampled points on a *thin film*. Each sample corresponds to a different composition, and has an associated measured x-ray diffraction pattern. Colors correspond to different combinations of the basis patterns $\alpha, \beta, \gamma, \delta$. Right: Scaling (shifting) of the diffraction patterns as one moves from one point to a neighboring one.

composition, which is observed as a shift in the scattering angle at which each diffraction peak is observed. For each peak $c \in \mathcal{Q}(p)$ we want to have at least one peak $e \in \mathcal{E}_k$ that can explain it, i.e. $\forall c \in \mathcal{Q}(p) \exists e \in \mathcal{E}_k \;\; s.t. \;\; (a_{p,k} > 0 \wedge \; |c - s_{p,k} \cdot e| \leq \epsilon)$ where ϵ is a parameter that depends on the accuracy of the peak-detection algorithm. The objective is therefore to minimize $\sum_{p=0}^{P-1} \sum_{k=0}^{K-1} \mathbb{1}_{[a_{p,k}>0]} \sum_{e \in \mathcal{E}_k} \mathbb{1}_{[\forall c \in \mathcal{Q}(p), |c - s_{p,k} \cdot e| > \epsilon]}$, which corresponds to the total number of missing peaks.

Other constraints must also be met. The coefficients $a_{p,k}$ must be non-negative and satisfy $|\{k | a_{p,k} > 0\}| \leq M$ (i.e. no more than M basis patterns can be used to explain a sample p). The subgraph induced by $\{p | a_{p,k} > 0\}$ must be connected in order for the basis patterns to appear in contiguous locations on the *thin film*. And for each basis pattern k, the corresponding scaling coefficients $s_{i,k}$ must be continuous and monotonic as a function of the corresponding location i on the *thin film*.

Note that one can avoid the use of expensive nonlinear arithmetic by using a logarithmic scale for the x-ray data, so that multiplicative scalings become linear operations. We refer to these effects (corresponding to the scalings in the original problem formulation) as *shifts*. Namely, we define a set $\mathcal{A}(p) = \{\log q, q \in \mathcal{Q}(p)\}$ of peak positions in log-scale and represent the positions of the peaks of the basis patterns using the same logarithmic scale.

4 Backdoor Identification

In this section, we first present a Satisfiability Modulo Theories (SMT) encoding of the problem, and then study the impact of various variable assignments on the performance of the approach.

As described in [Ermon *et al.*, 2012], the phase-map identification problem can be formulated as an SMT encoding as follows. Let P be the number of samples and L the maximum number of peaks per point, i.e.

$L = \max_p |\mathcal{A}_p|$. An upper and lower bound e_{max} and e_{min} for the positions of the peaks are computed based on the observed patterns. Moreover, ϵ represents a tolerance level such that two peaks within an interval of size 2ϵ are considered to be overlapping, while S_{max} is a bound on the maximum possible shift.

Variables Boolean variables $r_{p,k}$, for $0 \leq p \leq P-1, 0 \leq k \leq K-1$, indicate whether phase (basis pattern) k appears in point p (i.e., $a_{p,k} > 0$). The *Integer* variables $e_{k,\ell} \in [e_{min}, e_{max}]$ represents the position of the ℓ-th peak of the k-th basis pattern while $S_{p,k} \in [-S_{max}, S_{max}]$ represents the shift of the k-th basis pattern at point p. The variables $I_{p,k}$ are Integer indicators for the Boolean variables $r_{p,k}$ (i.e.$r_{p,k} \Leftrightarrow (I_{p,k} = 1)$) and used to count the number of phases involved at point p. The variables t_p represent the number of unexplained peaks at point p, i.e. the number of missing peaks at point p. These are peaks that should appear according to $\{r_{p,k}\}_{k=0}^{K-1}, \{e_{k,\ell}\}_{\ell=0}^{L-1}$, and $\{S_{p,k}\}_{k=0}^{K-1}$, but were not observed, i.e. do not belong to $\mathcal{Q}(p)$.

Constraints Every peak $a \in \mathcal{A}(p)$ in a point p must be explained by at least one peak belonging to a phase k, which can appear shifted by $S_{p,k}$:

$$\bigvee_{k=0}^{K-1} \bigvee_{\ell=0}^{L-1} \left(r_{p,k} \wedge (|e_{k,\ell} + S_{p,k} - a| \leq \epsilon)\right) \forall p, \forall a \in \mathcal{A}(p)$$

The number of missing peaks in sample p is defined as follows:

$$t_p = \sum_{k=0}^{K-1} \sum_{\ell=0}^{L-1} ITE(r_{p,k} \wedge \neg(\bigvee_{a \in \mathcal{A}(p)} |e_{k,\ell} + S_{p,k} - a| \leq \epsilon), 1, 0),$$

where ITE is an if-then-else expression. Here we assume that each phase contains at least one peak, but since peaks can be overlapping (e.g., $e_{k,\ell} = e_{k,\ell+1}$) a basis pattern is allowed to contain less than L distinct peaks.

The objective function is to minimize $\sum_{p=0}^{P-1} t_p$, the number of total missing peaks across all samples. Equivalently, we define a threshold T such that $\sum_{p=0}^{P-1} t_p \leq T$ and search for the lowest admissible value of T.

Finally, additional constraints on phase usage, shift continuity and monotonicity, and phase connectivity can be found in [Ermon *et al.*, 2012].

Next, we study the impact of various variable assignments on the running time of the approach. To determine the effectiveness of setting different types of variables, we set random subsets of each type of variables according to their correct values, and measure the resulting relative reduction in runtime (Fig. 2). These experiments are based on synthetic instances for which the ground truth is known, and allows us to correctly preassign any arbitrary variable of the model. For each of the 4 types of variables, we ran the solver with around 25 different levels of variable assignment sizes (from 0% to 100%), and 10 randomly-selected variable subsets. The results, as illustrated in Fig. 2, show that the $e_{k,l}$ variables, which correspond to the locations of the peaks in each phase, are the variables whose assignments trigger the best reduction in runtime. Conversely, the S_{pk} variables (i.e. the shifts of the phases in the samples) exhibit the poorest improvement. Therefore, this analysis advocates a discrepancy in the usefulness of any potential human input about variable values, and suggests that the information about the peak location of a phase is more valuable than, for example, how the phase shifts across the thin film.

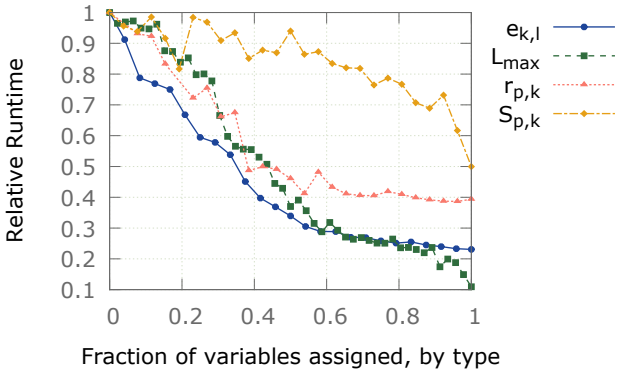

Figure 2: Improvement of the SMT solver runtime as a function of the number of pre-assigned variables, for each type of variable. The assignment fraction for maximum peaks per phase (L_{max}) is relative to the difference between the default and correct bounds.

In addition, this has to be put into perspective and consider the actual human effort required to provide such an input. In order to determine either the maximum peaks per phase (L_{max}) or the shift values ($S_{p,k}$), one must first carefully identify the peaks present in each phase. Furthermore, phase presence information ($r_{p,k}$) and shift values are tedious to communicate, and difficult to decompose into multiple tasks. Finally, some information on the maximum peaks per phase and phase presence can be inferred from human selection of peaks belonging to particular phases. We focus on human input to set peak locations as the performance benefit exceeds that of the other variable classes, with the intention of deducing additional variable settings where possible. How to collect this human input is the subject of the next section.

5 Human-Computation Component

To gather human computation input in order to identify the values of backdoor variables, our approach involves the following steps: 1) enumerating and visualizing the sets of XRD patterns that are likely to make the phases most apparent, 2) identifying subsets of peaks that define partial phases, and 3) assigning the variables according to the user input and running the SMT solver.

We compare two human computation approaches for

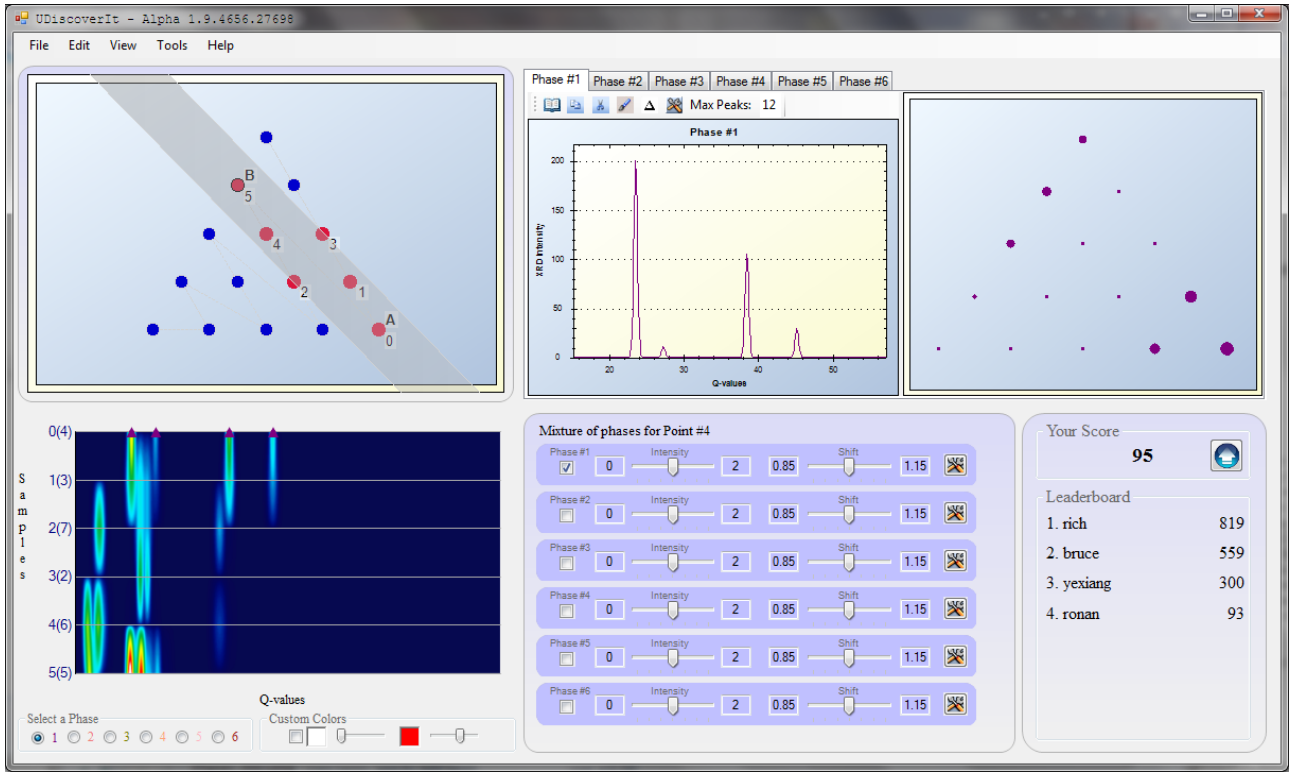

Figure 3: Snapshot of UDiscoverIt, a graphical user interface for providing human input to an SMT solver for the phase-map identification problem.

determining variable assignments, which differ in the level of background and effort required to contribute, as well as the scope of the human computation tasks. In the first approach, individual problem solvers explore entire problem instances and use this global context to provide self-contained partial solutions for the solver. In the second method, we decompose the problem into smaller human computation tasks to be completed quickly by many contributors on the Amazon Mechanical Turk.

Individual Problem Solver Approach Our first approach is designed for individual problem solvers who are familiar with the problem domain, but does not require expert knowledge. The UDiscoverIt interface, depicted in Figure 3, helps the user visualize the XRD patterns and provide insightful input about the underlying basis patterns. The features and controls of the interface reflect the underlying physics, including the combination and scaling of the basis patterns (Fig. 3: Bottom-right), and guide the user towards physically meaningful partial solutions. A user proceeds as follows. First, the user selects one or more slices (linear subsets) of XRD patterns to analyze (Fig. 3: Top-left). Next, the user identifies peaks that behave jointly among these patterns if such a relationship is clear from the selected slice. These peaks are assumed to belong to the same phase, and are used to build a partial phase (Fig. 3: Top-right). Namely, with respect to the SMT encoding, this subset of peaks will be used to initialize the $e_{k,l}$ variables of the corresponding phase k. In addition, given that this partial phase has been built up from the peaks of a given sample p, this input allows us to infer values about the variables $r_{p,k}$ and $S_{p,k}$ as well.

At this point, the user is invited to submit this partial solution to the server. Then, the user proceeds iteratively, with new slices of XRD patterns, until the expected number of phases has been reached.

Crowdsourcing Approach The process described above can be decomposed into simple human computation tasks and completed quickly by many Turkers, without context or background knowledge of the problem. For each problem instance, we generate images corresponding to 12 slices using simple geometric rules, without assuming any prior knowledge about the instance. These slices include most of the sample points, and illustrate a variety of composition gradients. A separate task is then created for each sample point in each slice, resulting in 40-100 images per instance, including between one and five images per sample point. In future work, we are planning to reduce number of individual tasks required by using measurable relationships between the XRD patterns to predict the most informative slices.

Tasks are embedded in an interactive interface and submitted to the Amazon Mechanical Turk (Fig. 4). In each task, Turkers visually identify lines forming pat-

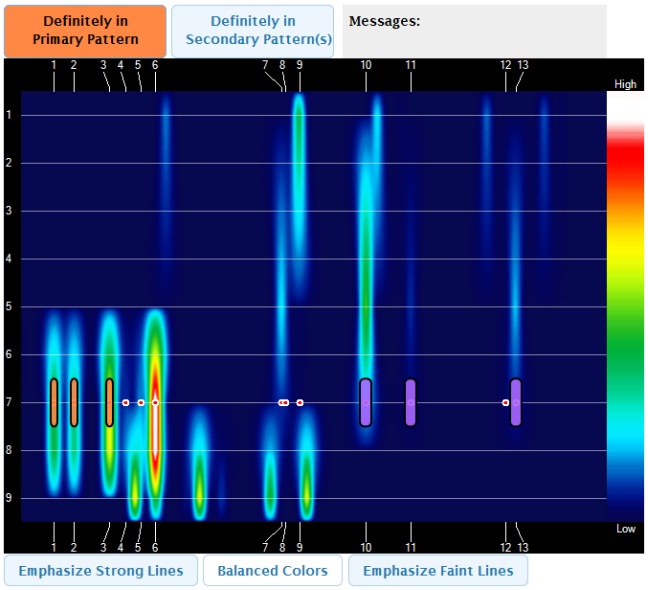

Figure 4: Example of a completed Mechanical Turk task. The three leftmost lines are marked in orange as a primary pattern, and three more towards the right are marked in purple as secondary patterns. The others are less clear and have been left unmarked.

terns according to specific criteria, and click to mark them. Turkers are required to complete a 20-minute tutorial and qualification test to learn the features that constitute a pattern. They are not provided with an explanation of the application purpose or the meaning of the patterns, which are simply treated as abstract visual patterns. Turkers typically complete each task in less than one minute.

Each task is assigned to five different Turkers, and 5-25 answers are aggregated by sample point. For each sample point, we enumerate a set of candidate patterns to use as partial definitions for phases, rank them according to criteria representing the level of agreement among submitted answers, and select the best non-conflicting patterns.

This is, however, a non-trivial process. Indeed, answers can incorrectly include peaks from two separate phases, which can naturally be solved with a voting mechanism. Furthermore, Turkers can choose to provide answers based on any phase present, and a majority of votes is therefore required to guarantee that all included peaks belong to the same phase. In addition, Turkers often include only a small subset of the relevant peaks, potentially with little agreement on which peaks are included. As a result, it is likely that for the more complex sample points, no subset of peaks will meet the threshold for inclusion, in order to constitute a partial phase.

In the following, we describe how to aggregate the input from the Turkers and to overcome the previously mentioned limitations. For each sample point, we generate an edge-weighted graph $G = (V, E)$ where V is the set of peaks, E is the set of all pairs of peaks (defining a complete graph), and the weight of an edge $e = (i, j) \in E$ corresponds to the co-occurrence count of peaks i and j in the submitted answers. At this point, we enumerate the unique, non-trivial components that result from partitioning the graph when cutting the edges whose weight is below a range of threshold values. Finally, for each component we generate one candidate pattern based on majority vote, only taking into account the edge weights in this component.

Next, we propose to rank these candidate patterns according to the degree agreement in the submissions. In practice, individual tasks vary in clarity, difficulty, and number of phases present. Turkers are expected to have the strongest agreement on answers corresponding to the clearest phase patterns, at the sample points where they are most prominent. Overall, the distinct patterns exhibiting the strongest agreement represent the most likely subsets of peaks in each phase. We score each candidate pattern on the level of agreement among the submissions, as well as the amount of information collected. A good candidate for such a score is the normalized sum of the ratio of internal edges to total of internal and external edges, the ratio of internal edges to the maximum possible internal edges, and the average degree of the induced subgraph. Although one might consider a different measure for ranking the candidate patterns, this proposed score combines the notions of internal and external graph densities, typically defined in graph clustering [Schaeffer, 2007, e.g.].

Finally, the final aggregation step involves a human computation task in which we select the highest-ranked candidate patterns that correspond to different phases. Each subsequent pattern in rank order is selected if and only if it is visually determined to represent a distinct phase from those previously selected. In future work, we expect to perform a unified aggregation across all sample points, leveraging structure between neighboring sample points, and avoiding the need for this manual step. Furthermore, we infer the maximum number of peaks in each phase from the number of peaks in the sample point in which the phase is defined.

Overall, this process allows us to partially assigned the $e_{k,l}$ variables with the selected candidate patterns, as well as the maximum number of peaks L_{max} in each phase, and we submit the resulting variable settings to the SMT solver.

6 Empirical Results

The instances we used to evaluate our approach are synthetic instances, generated as described in [Ermon *et al.*, 2012]. As opposed to real instances for which the solution is unknown, synthetic instances allow to validate the proposed approach. Nevertheless, these instances are realistic in size and generated from actual physical/crystallographic models.

First, we evaluated the Mechanical Turk task accuracy using ground truth values, which can be calculated

Dataset				Without user input			Single user input		Crowdsourced input	
System	P	L^*	K	*#var*	*#cst*	**Time** *(s)*	**Time** *(s)*	**#peaks set**	**Time** *(s)*	**#peaks set**
A1	36	8	4	408	2095	3502	**153**	15	**545**	6
A2	60	8	4	624	3369	17345	**262**	14	**1178**	6
B1	15	6	6	267	1009	79	**28**	6	**4**	8
C1	28	6	6	436	1864	346	**83**	12	**63**	7
C2	28	8	6	490	2131	10076	**517**	17	**140**	17
C3	28	10	6	526	2309	28170	**332**	17	**421**	20
D1	45	7	6	693	3281	18882	**107**	22	**196**	13
D2	45	8	6	711	3410	46816	**129**	24	**350**	15

Table 1: Runtime (seconds) of the SMT solver with and without user input. 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.

for the synthetically generated instances. Overall, 16% of answers only included a single peak, suggesting the Turkers could not confidently define a clear pattern and that the images vary in pattern clarity and complexity. Nevertheless, only 1% of non-trivial answers incorrectly included peaks from two or more different phases. In addition, 36% of answers included a secondary pattern, which workers were instructed to include when they were nearly as clear as the primary pattern. Out of these answers, 10% incorrectly included peaks from the same phase as the primary pattern. In some circumstances, workers tended to systematically interpret a single phase as multiple patterns. We had intended to use the secondary patterns to infer the maximum number of peaks in each phase; however, it is essential not to underestimate this value. As a result, a more sophisticated approach would be necessary to incorporate these answers, and we excluded them from our analysis.

Next, we performed experiments to evaluate the reduction in runtime resulting from the human-provided variable assignments. All experiments were run on a Linux (version 2.6.18) cluster where each node has an Intel Xeon Processor X5670, with dual-CPU, hex-core @2.93GHz, 12M Cache, 48GB RAM. The SMT solver used in these experiments was *Z3* [De Moura and Bjørner, 2008].

Table 1 shows the runtime of the SMT solver on 8 instances of various sizes, with and without user input from both human computation approaches. It shows that both user input allow a significant improvement in runtime on each instance, between about one to two orders of magnitude of improvement on all instances. For example, on the D2 dataset, the running time was reduced from 13 hours down to 2 to 6 minutes. In terms of solution time, one should also consider the time spent by humans when completing the tasks. In the case of the individual problem solver approach, each instance requires on average about 5 min. In the crowdsourcing approach, Turkers spent about 20 seconds on each of the approximately 50 HITs per instance, for a total of about 15 min per instance. Nonetheless, given its simple HIT selection scheme, our method clearly collects redundant information and could greatly benefit from a more sophisticated active learning approach. Further minimizing the amount of input needed from the crowd will be the focus of future work.

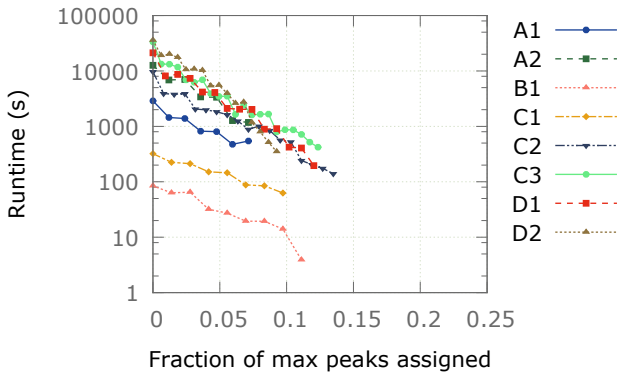

Figure 5: Runtime of the SMT solver as a function of the number of crowd-selected peaks on each system.

We also measure the incremental impact of the amount of user input incorporated into the SMT solver for both approaches, as illustrated in Figs. 6 and 5. These results show that the amount of user input has diminishing absolute returns, and suggest that a minimal user input dramatically speeds up the search.

Interestingly, the level of user input needed to reach such performance is quite minimal with respect to the instance size. The input corresponds to the assignments of about 20 variables, which represents barely 5% of all the variables of the SMT encoding.

7 Conclusion

Our experiments show how human computation and crowdsourcing insights can be key to identifying backdoor variables in combinatorial optimization problems, dramatically speeding up the performance of combinatorial solvers. Our approach leverages the complementary strength of human input, providing global insights into problem structure, and the power of combinatorial solvers to exploit complex local constraints. In this work,

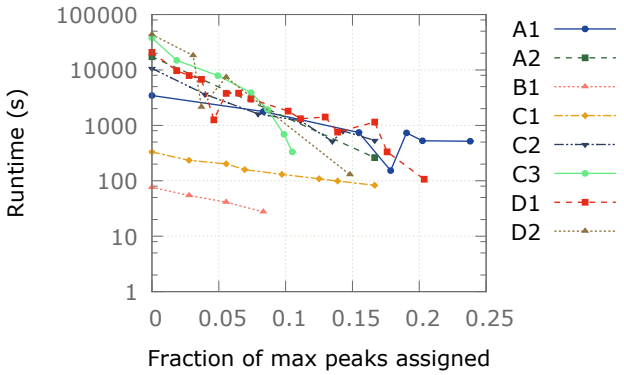

Figure 6: Runtime of the SMT solver as a function of the number of individual-selected peaks on each system.

we also show how the identification of backdoor variables can be translated into abstract pattern visualization tasks, with no information about the original combinatorial optimization problem in question, in a way that allowed for crowdsourcing. We plan to further pursue this line of research considering different crowdsourcing strategies. We described our work in the context of the domain of materials discovery. We believe there are many other combinatorial domains for which a similar approach holds promise.

Acknowledgments

We are thankful to the anonymous reviewers for their constructive feedback. Also, we thank Stefano Ermon for his insights into the SMT formulation, and Yexiang Xue and Bruce van Dover's students for providing user input. This work was supported by the National Science Foundation (NSF Expeditions in Computing award for Computational Sustainability, grant 0832782). The experiments were run on an infrastructure supported by the NSF Computing research infrastructure for Computational Sustainability grant (grant 1059284).

References

[Cooper *et al.*, 2010] S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, Z. Popović, and Foldit Players. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756–760, August 2010.

[De Moura and Bjørner, 2008] L. De Moura and N. Bjørner. Z3: An efficient smt solver. *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, 2008.

[Dilkina *et al.*, 2009] B. Dilkina, C. Gomes, Y. Malitsky, A. Sabharwal, and M. Sellmann. Backdoors to combinatorial optimization: Feasibility and optimality. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 56–70, 2009.

[Ermon *et al.*, 2012] S. Ermon, R. Le Bras, C. P. Gomes, B. Selman, and R. B. van Dover. Smt-aided combinatorial materials discovery. In *Proc. of the Conference on Theory and Applications of Satisfiability Testing*, SAT'12, 2012.

[Fischetti and Monaci, 2011] M. Fischetti and M. Monaci. Backdoor branching. *Integer Programming and Combinatoral Optimization*, pages 183–191, 2011.

[Gaspers and Szeider, 2012] S. Gaspers and S. Szeider. Backdoors to satisfaction. *The Multivariate Algorithmic Revolution and Beyond*, pages 287–317, 2012.

[Gomes *et al.*, 1998] C.P. Gomes, B. Selman, H. Kautz, et al. Boosting combinatorial search through randomization. In *Proceedings of the National Conference on Artificial Intelligence*, pages 431–437. JOHN WILEY & SONS LTD, 1998.

[Gregoire *et al.*, 2010] J. M. Gregoire, M. E. Tague, S. Cahen, S. Khan, H. D. Abruna, F. J. DiSalvo, and R. B. van Dover. Improved fuel cell oxidation catalysis in pt1-xtax. *Chem. Mater.*, 22(3):1080, 2010.

[Hoffmann *et al.*, 2007] J. Hoffmann, C. Gomes, and B. Selman. Structure and problem hardness: Goal asymmetry and dpll proofs in sat-based planning. *Logical Methods in Computer Science*, 3(1):6, 2007.

[Khatib *et al.*, 2011] F. Khatib, S. Cooper, M.D. Tyka, K. Xu, I. Makedon, Z. Popović, D. Baker, and F. Players. Algorithm discovery by protein folding game players. *Proceedings of the National Academy of Sciences*, 108(47):18949–18953, 2011.

[Kilby *et al.*, 2005] P. Kilby, J. Slaney, S. Thiébaux, and T. Walsh. Backbones and backdoors in satisfiability. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 1368. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[Law and von Ahn, 2011] E. Law and L. von Ahn. *Human Computation*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2011.

[Le Bras *et al.*, 2011] R. Le Bras, T. Damoulas, J. M. Gregoire, A. Sabharwal, C. Gomes, and R. B. van Dover. Constraint reasoning and kernel clustering for pattern decomposition with scaling. In *CP*, 2011.

[McLennan and Gainer, 2012] Sarah McLennan and Mary Gainer. When the Computer Wore a Skirt: Langleys Computers, 19351970. *NASA History Program Office News & Notes*, 29(1):25–32, 2012.

[O'Sullivan, 2010] Barry O'Sullivan. Backdoors to satisfaction. tutorial at cp 2010. In *Proceedings of the Principles and Practice of Constraint Programming (CP)*, 2010.

[Schaeffer, 2007] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[Szeider, 2006] S. Szeider. Backdoor sets for dll subsolvers. *SAT 2005*, pages 73–88, 2006.

[Van Dover *et al.*, 1998] R. B. Van Dover, LF Schneemeyer, and RM Fleming. Discovery of a useful thin-film dielectric using a composition-spread approach. *Nature*, 392(6672):162–164, 1998.

[Williams *et al.*, 2003a] R. Williams, C. Gomes, and B. Selman. On the connections between backdoors, restarts, and heavy-tailedness in combinatorial search. *structure*, 23:4, 2003.

[Williams *et al.*, 2003b] R. Williams, C.P. Gomes, and B. Selman. Backdoors to typical case complexity. In *International Joint Conference on Artificial Intelligence*, volume 18, pages 1173–1178. Citeseer, 2003.