Parallel Algorithms for Equilevel Predicates

Anonymous

ABSTRACT

We define a new class of predicates called *equilevel predicates* on a distributive lattice which eases the analysis of parallel algorithms. Many combinatorial problems such as the vertex cover problem, the bipartite matching problem, and the minimum spanning tree problem can be modeled as detecting an equilevel predicate. The problem of detecting an equilevel problem is NP-complete, but equilevel predicates with the *helpful* property can be detected in polynomial time. Furthermore, the refined *independently helpful* property allows online parallel detection of such predicates in NC.

We also define a special class of equilevel predicates called *solitary* predicates. Unless NP= RP, this class of predicate also does not admit efficient algorithms. Earlier work has shown that solitary predicates with the *efficient advancement* can be detected in polynomial time. We introduce two properties called the *antimonotone advancement* and the *efficient rejection* which yield the detection of solitary predicates in NC. Finally, we identify the minimum spanning tree, the shortest path, and the conjunctive predicate detection as problems satisfying such properties, giving alternative certifications of their NC memberships as a result.

CCS CONCEPTS

• **Theory of computation → Parallel algorithms**.

KEYWORDS

Detecting Predicates, Distributive Lattices, Equilevel Predicates

ACM Reference Format:

Anonymous . 2023. Parallel Algorithms for Equilevel Predicates. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

We introduce a new class of predicates called *equilevel predicates* to model and solve many problems in parallel and distributed computing. These predicates are defined on the elements of a distributive lattice [1]. The distributive lattice is such that the height is small, but its size is large. More concretely, this lattice can be viewed as generated from a poset with at most n chains and m elements per chain. In this case, the height of the lattice is bounded by $O(nm)$ but the cardinality can be as many as n^m elements. In order to gain efficiency, our methods operate on the underlying poset even though we are interested in finding an element in the lattice satisfying the given predicate. This approach allows us to obtain algorithms with complexities efficient in n and m.

A predicate defined on a lattice is equilevel if all the elements of the lattice that satisfy the predicate are on the same level of the lattice. It can be shown that detecting an equilevel predicate is, in general, a hard problem. For example, the minimum vertex cover problem for a graph can be modeled as detecting an equilevel predicate. Even so, we will see that there are many equilevel predicates which can be detected efficiently in parallel, and that this efficiency can be implied by a few simple properties.

As a special case, we also consider the class of equilevel predicates which hold on a single element in the underlying lattice. We call this class *solitary*. This class of predicates is related to the previously studied class of *lattice-linear* predicates [2], which are those predicates closed under meet operations on the underlying lattice. Such predicates have special properties that aid in the design of efficient detection algorithms. Observe that for any lattice-linear predicate B we can define a stronger predicate B' that is satisfied only on the least element in the lattice satisfying B. Then B' is a solitary predicate. For example, consider the distributive lattice of assignments for the stable marriage problem [3, 4]. There may be multiple stable marriages; however, there is a unique man-optimal stable marriage. The predicate that an assignment corresponds to the man-optimal stable marriage is a solitary predicate.

In this paper, we investigate equilevel and solitary predicates, as well as the conditions giving efficient parallel algorithms for their detection. We first show that the problem of equilevel predicate is NP-complete in general. Moreover, the implications of this result are far-reaching in the landscape of lattice-linear predicate detection. For example, we show that slight generalizations of lattice-linear predicate-based problems with efficient solutions, such as the conjunctive predicate detection, are NP-complete. We then define a property on equilevel predicates called the *helpful* property. If any equilevel predicate has the helpful property, then it can be detected efficiently in an online manner. We apply the helpful property to modeling and efficient detection of various problems such as bipartite matching and computing bases which span sets of vectors as equilevel predicates. Furthermore, when this is refined to the stronger *independently helpful* property, there exists a simple NC algorithm for problems modeled as equilevel predicates.

Next, a special class of equilevel predicates, called solitary predicates, is identified. These are those predicates that can only be true on a single element in the lattice, and we show that even with this substantial restriction, detecting these predicates is still hard unless NP=RP. However, prior work implies that solitary predicates can be detected efficiently in sequential contexts when they satisfy *efficient advancement*[2]. We refine the efficient advancement property in a few ways to obtain NC algorithms. In particular, we show that whenever a problem satisfies the *antimonotone* advancement property, it can be solved in NC time when the poset generating the lattice has a small height. For example, it can be shown that the problem of the minimum spanning tree satisfies

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Class of Predicates	Examples	*Detection Algorithm*
General Equilevel	Vertex Cover, k-conjunctive	no efficient algorithm unless P= NP
With Helpful Property	bipartite matching	P
With Independently Helpful Property	minimum spanning tree	NC
General Solitary	USAT	No efficient algorithm unless NP= RP
With Efficient Advancement	man-optimal stable marriage	P
With Antimonotone Advancement	minimum spanning tree with unique weights	NC if the poset has $O(\log n)$ height
With Efficient Rejection	Graph Reachability, Conjunctive Predicates	NC

Figure 1: Various Classes of Predicates. Equilevel predicates are the ones that are true on elements of a lattice at a single level. Solitary predicates are the ones that are true on a single element in the lattice.

antimonotone advancement property, which gives an alternative certification of its membership in NC. Another special class of efficient advancement property giving NC algorithms is the *efficient rejection* property. We show that the shortest path problem [5], and the conjunctive predicate detection in distributed computations [6] satisfy this property.

In summary, this paper makes the following contributions:

- The paper introduces the class of equilevel predicates. Detecting a general equilevel predicate is NP-complete. We show that a slight generalization of the conjunctive predicate detection where one asks for a global state with exactly k events satisfying the conjunctive predicate is also NP-complete.
- We show that any equilevel predicate with a helpful property (defined in this paper) can be detected in polynomial time. The problem of finding the size of a maximum matching in a bipartite graph falls in this class.
- We show that any equilevel predicate with the independently helpful property can be detected in NC. The problem of computing a minimum spanning tree in a weighted undirected graph is shown to be in this class.
- The paper introduces a subclass of equilevel predicates called solitary predicates. It is shown that there is no randomized polynomial time detection algorithm for a solitary predicate unless NP=RP. However, any solitary predicate with the efficient advancement property can be detected in polynomial time.
- The paper introduces two subclasses of solitary predicates: with the *antimonotone* advancement property and with the *efficient rejection* property. Solitary predicate with these properties can be detected in NC.

2 RELATED WORK

Predicates on a distributive lattice whose detection models various combinatorial algorithms have been introduced in [2], [7], [8]. These papers study the class of predicates called lattice-linear predicates which is equivalent to being closed under the meet operation of the lattice. It has been shown that the lattice-linear predicate (LLP) algorithm solves many combinatorial optimization problems such as the shortest path problem, the stable marriage problem, and the market clearing price problem [2]. This method has been applied to other problems such as dynamic programming problems [9], the housing allocation problem [7], the minimum spanning tree problem [10], and generalizations of the stable matching problem [11]. In [8], Gupta and Kulkarni extend LLP algorithms for deriving self-stabilizing algorithms. In [12], Gupta and Kulkarni show that multiplication and modulo operations on natural numbers can also be modeled as LLP algorithms, and in [13] they use lattice-linearity to simplify the analysis of a robot coordination algorithm in an asynchronous deployment setting and improve upon the algorithm's complexity guarantees.

At its core, both the above-mentioned work and our work are based on obtaining solutions to combinatorial optimization and constraint satisfaction problems through predicate detection algorithms. In the context of distributed monitoring, the technique of predicate detection was introduced by Cooper and Marzullo [14] and Garg and Waldecker [15]. Regarding existing classes of predicates introduced for this setting, the detection of conjunctive predicates was examined in [6], lattice-linear predicates were introduced in [16], and regular predicates were introduced in [17]. Moreover, other classes specific to distributed computing settings, such as observer-independent predicates [18], have been investigated as well.

Note that equilevel predicates which hold for multiple elements on the same level cannot be closed under meet. Thus, the LLP algorithm is inapplicable for these predicates. In contrast, when restricted to solitary predicates LLP algorithms can be applied. To advance along this line of work, we identify additional properties that allow parallel NC algorithms to detect solitary predicates.

3 EQUILEVEL PREDICATES

Throughout, we consider problems defined on a distributive lattice representing the domain. To ease the presentation of our definitions, we consider a specific representation. By Birkhoff's theorem [19], any finite distributive lattice $\mathcal{L}$ can be generated by a finite poset P. Let that poset P consist of n chains, i.e., P can be decomposed into the direct product of n chains $P_1, P_2, \ldots P_n$ of distinct elements. Such a poset is called a realizer and always exists [20]. Thus, we can assume $\mathcal{L}$ is represented this way without loss of generality, and every element $G \in \mathcal{L}$ can be viewed as an ideal of P. We let $G[i]$ denote the number of elements in G from P_i, while $|G|$ denotes the cardinality counting all elements in the ideal representing G. Finally, $\bot$ and $\top$ correspond to the bottom and top elements of the lattice respectively. A predicate on $\mathcal{L}$ is an equilevel predicate if all the satisfying elements are on the same level of the lattice.

Definition 3.1 (Equilevel Predicate). A boolean predicate B is *equilevel* with respect to a lattice $\mathcal{L}$ iff

$$\forall G, H \in \mathcal{L} : B(G) \wedge B(H) \implies |G| = |H|.$$

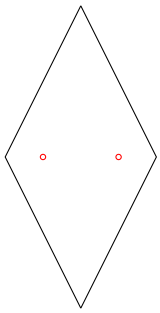

Figure 2: Search space with elements that satisfy the given predicate B. The satisfying solutions are shown in red. They are all at the same level.

Fig. 2 shows a search space (modeled using a distributive lattice) and a predicate and its set of satisfying solutions which are all on the same level of the lattice. We now define the notion of "detecting" a predicate.

Definition 3.2 (Predicate Detection). Given a poset P generating a lattice $\mathcal{L}$, and a Boolean predicate B, if B is true on some element G of lattice $\mathcal{L}$, then decide "yes." Otherwise, decide "no."

For example, consider the minimum vertex cover problem in an undirected graph (V, E). A subset of vertices V' satisfies B if V' is the vertex cover of minimum size. Here, the lattice we consider is the boolean lattice of the vertex set. It is clear that B is an equilevel predicate since there may be multiple minima vertex covers and they must all be of the same size. Then, we observe that detecting an equilevel predicate is NP-complete in general as the minimum vertex-cover decision problem is well known to be NP-complete.

As another example of an equilevel predicate, suppose that we have a bipartite graph (L, R, E) and we are interested in maximum-sized subsets of L that can be matched. The size of maximum-sized subsets of L that can be matched to elements in R is constant, even though there may be multiple matched sets in L. For example, consider $L = \{m_1, m_2, m_3\}$, $R = \{w_1, w_2, w_3\}$ and $E = \{(m_1, w_1), (m_2, w_1), (m_3, w_3)\}$. Then, we can match $\{m_1, m_3\}$ or $\{m_2, m_3\}$. Both sets are of size 2.

Equilevel predicates occur in numerous other contexts. Consider the problem of finding a minimum weight spanning tree of a connected undirected graph (V, E) with n vertices. If the edge weights are not unique, then there may be multiple minimum-spanning trees with equal weight. Let the predicate B be "the set of edges form a spanning tree with minimum weight." It is clear that all sets that satisfy the predicate have $n - 1$ edges. Thus, in the inclusion lattice over the graph's edges, the sets satisfying the predicate are all at the same level.

As a benefit, equilevel predicates are closed under conjunction. If predicates B_1 and B_2 are true for different levels, then the conjunction is false for all elements of the distributive lattice and the predicate $B_1 \wedge B_2$ is trivially equilevel. Otherwise, if they are true for the same level, then $B_1 \wedge B_2$ is also true only at that level (or possibly, always false). However, equilevel predicates are not closed under disjunction or complement because these operations may introduce elements satisfying the new predicate at many different levels of the lattice.

As we have observed, since NP-complete problems such as vertex cover can be modeled as detecting an equilevel predicate, the problem of detecting an arbitrary equilevel predicate is NP-hard. We now show that some problems in distributed computation with efficient algorithms are equilevel predicates when slightly generalized. For example, consider *conjunctive predicates* which are given by a conjunctive $l_1 \wedge l_2 \wedge \ldots l_n$ where each l_i is a function only of state local to the i^{th} process. For this class, there is an efficient algorithm for detecting this predicate in a distributed computation [6]. However, if we generalize the problem by asking for the conjunctive predicate on a particular level of the lattice, then the problem can be modeled as an equilevel predicate. We now show that, in general, asking for a conjunctive predicate on a particular level is NP-complete.

Theorem 3.3. *Given a distributed computation, deciding whether there exists a global state with k events satisfying a given conjunctive predicate is* NP*-complete.*

Proof. We first show that the problem is in NP. The global state itself provides a succinct certificate. We can check that all local predicates are true in that global state and that the global state is at level k.

For hardness, we use the subset sum problem. Given a subset problem on n positive integers, $x_1, x_2, \ldots, x_n$ with the requirement to choose a subset that adds up to k, we create a computation on n processes as follows. Each process P_i has x_i events. The local predicate on P_i is true initially and also after it has executed x_i events. Thus, the local predicate is true at each process exactly twice. The problem asks us if there is a global state with k events in which all local predicates are true. Such a global state, if it exists, would choose for every process either the initial local state or the final local state. All the final states that are chosen correspond to the numbers that have been chosen.

To avoid the expansion of the numbers in binary to unary construction, we encode the representation of events on a process as follows: Since the conjunctive predicates can only be true when the local predicates are true, we keep only local states which satisfy their corresponding local predicate and store the number of local events executed so far with them. This leaves two local states at each process: The initial state with zero events executed until that point, and the final state of the process with a number of events equal to x_i for the i^{th} process. Now, the construction of the computation from the subset sum problem is polynomial in size. □

We now restrict the class of equilevel predicate to a class that can be efficiently detected.

4 EQUILEVEL PREDICATES WITH HELPFUL PROPERTY

One class of equilevel predicates that we can efficiently detect is the set of predicates that satisfy the *helpful* property. An equilevel predicate satisfies the helpful property if, whenever it is false on a global state G, we are guaranteed to be able to compute in polynomial time the indices of the local states which are *helpful*. Unless

the global state G is advanced on *at least one* helpful local state the predicate can never become true.

Formally,

Definition 4.1 (Helpful Property). A Boolean predicate B has the *helpful* property with a polynomial time algorithm $\mathcal{A}$ on a finite distributive lattice $\mathcal{L}$ if, for all $G \in \mathcal{L}$,

$$\neg B(G) \Rightarrow \mathcal{A}(G) \neq \emptyset \wedge \forall i \in \mathcal{A}(G) : helpful(i, G, B).$$

where $helpful(i, G, B)$ is defined as

$$\exists W \in L : B(W) \Rightarrow \exists H > G : (H[i] > G[i]) \wedge B(H).$$

The definition states that whenever the predicate is not true in a global state G, the polynomial time algorithm $\mathcal{A}$ can return a nonempty set of indices $\mathcal{A}(G)$ such that advancing on any of the corresponding local states can be used to detect the predicate. The i^{th} local state is helpful in the global state G if we can advance the global state G on the index i whenever $\neg B(G)$ holds without the risk of missing a satisfying global state H. If there is no global state W that satisfies B, then $helpful(i, G, B)$ holds vacuously.

The helpful property allows us to efficiently detect an equilevel predicate satisfying the property. In all our examples, we also require the algorithm $\mathcal{A}$ to be *online* in the sense that the algorithm $\mathcal{A}$ has access to the poset only on states up to G. Therefore, the computation of helpful indices is based only on the poset of states less than or equal to G. This allows us to use a sequential procedure, given in Algorithm Equilevel, to advance on helpful indices towards a satisfying global state if one exists.

ALGORITHM Equilevel: A Sequential Algorithm to find a state satisfying B

```
function GetSatisfying(B: predicate, L: Lattice)
    var G: vector of int initially ∀i : G[i] = 0;
    while ¬B(G) do
        if (G is the top element of L)
            then return null;
        else
            G[i] := G[i] + 1 where helpful(i, G, B);
    endwhile;
    return G; // an optimal solution
```

Theorem 4.2 gives the correctness of Algorithm Equilevel.

Theorem 4.2. *Let B be any equilevel predicate that satisfies a helpful property with a polynomial time algorithm $\mathcal{A}$. Then, B can be detected in polynomial time.*

Proof. Algorithm Equilevel starts with the least element in the lattice $\mathcal{L}$. The *while* loop can execute at most mn times where n is the number of chains and m is the maximum height of any chain. The algorithm $\mathcal{A}$ to compute $helpful$ has polynomial complexity. Hence, Algorithm Equilevel also has polynomial complexity. □

We show detection of equilevel predicates through the following examples:

Example 1: Maximum Cardinality Bipartite Matching. Consider the problem of bipartite matching in a graph (L, R, E). We are interested in finding a subset $V \subseteq L$ of maximum size such that V can be matched with vertices in R. We use the Boolean lattice on the vertices L. A vertex set $V \subseteq L$ satisfies B if $|V|$ is the maximum number of vertices that can be matched. It is clear that B is an equilevel predicate. Suppose that we have a set of vertices $V \subseteq L$ which is not of the maximum size. Then, we can find a set of helpful set of vertices W such that any element of W can be added to V. It can be checked efficiently that V does not satisfy B (for example, by checking if there exists an alternating path from a vertex in $L - V$ to an unmatched vertex in R). The algorithm to find helpful vertices is simple: any vertex in $L - V$ that has an alternating path to an uncovered vertex in R is a helpful vertex. For efficiency, the algorithm may maintain the set of matched vertices in R. One can use Algorithm Equilevel to find the largest set in L with matching. The predicate $helpful(i, G, B)$ holds whenever the vertex i can be added to the current set of matched vertices G. We note here that [21] discusses parallel algorithms for perfect matching in a bipartite graph. Our goal is to view the problem from the perspective of detecting a global condition.

Example 2: Minimum Spanning Tree in an Undirected Graph. Let B be true on a set of edges of an undirected graph if the edge set forms a spanning tree of the graph. Given a connected undirected graph with n vertices, it is well-known that any edge set that is acyclic and has $n - 1$ edges satisfies B. This predicate is equilevel because all the elements that satisfy the predicate B are at the level $n - 1$ in the boolean lattice of all edges. Given a set of edges, one can efficiently compute whether the set is acyclic. Hence, one can use Algorithm Equilevel to find a spanning tree as follows: The predicate $helpful(i, G, B)$ holds whenever the edge e_i does not form a cycle with edges in G, so at each step Algorithm Equilevel adds any edge which does not make a cycle in the current solution.

Example 3: Basis Set of Vectors. Suppose that we are given a set of vectors S of the same dimension. Our goal is to compute a basis, i.e. find a subset of the biggest size containing linearly independent vectors. The predicate B is true on a set G if all the vectors in G are linearly independent, and there does not exist any vector in $S - G$ that is linearly independent with vectors in G. Here $helpful(i, G, B)$ holds if the vector v_i is independent of all the vectors in G.

Lattice-linear predicates [2] are associated with the concept of *forbidden* local states. In particular, for any G which does not satisfy a lattice-linear predicate B, there exists at least one local state s such that no global state that satisfies B can have the same local state s. This means that one can advance on all forbidden local states of a lattice-linear predicate unconditionally. In particular, if there are multiple local states that are forbidden then the global state can be advanced on all of them in parallel as forbidden local states remain forbidden until they are advanced. In contrast, when it comes to equilevel predicates, advancing on one helpful local state may make the local state on some other process "unhelpful." For example, in computing a spanning tree we may be able to add one of edge a or edge b to our current solution, but adding both of them could create a cycle.

An example of an equilevel predicate without the helpful property is the vertex cover of the least size in a general graph. It is clear, that unless P = NP, this equilevel predicate cannot have the helpful property.

5 EQUILEVEL PREDICATES WITH INDEPENDENTLY HELPFUL PROPERTY

We now define a stronger version of equilevel predicates in which at every step, one evaluates which indices are independently helpful. Two helpful indices are *independently helpful* if it is okay to advance on both of them in parallel. This definition allows us to design parallel algorithms for equilevel predicates. Formally,

Definition 5.1 (Independently Helpful Property). A Boolean predicate B has an independently helpful property with the NC algorithm $\mathcal{A}$ if whenever $\neg B(G)$ and $B(H)$ for some $H > G$, the algorithm $\mathcal{A}$ returns a nonempty index set J satisfying $\forall i \in J : H[i] > G[i]$ such that with a polylogarithmic number of parallel advancements, the predicate is detected.

Take special note of the requirements on the algorithm $\mathcal{A}$: We require it to be of polylogarithmic depth complexity so that we can detect the predicate in NC. Any equilevel predicate with independently helpful property can be detected in NC by using the Algorithm Equilevel-Independence.

Theorem 5.2. *Let B be any equilevel predicate that satisfies an independently helpful property with an* NC *algorithm $\mathcal{A}$. Then, B can be detected in* NC.

Proof. Algorithm Equilevel-Independence starts with the least element in the lattice $\mathcal{L}$. In each step of the *while* loop, it makes one call of helpful and advances on all indices i with $helpful(i, G, B)$. If there can be at most polylogarithmic number of advancements, then the algorithm will detect B in NC. □

Example 1: For the problem of minimum-spanning tree, any set of edges that do not form a cycle with the currently chosen edges is independently helpful. Observe that there may be multiple sets of independently helpful states. We only require the algorithm to return any such set. How do we find an independently helpful set in parallel? At every stage in the algorithm, there is a set of connected components. Each connected component chooses a single outgoing edge such that the edges chosen do not form any cycle. This set of edges can be added in parallel to reach the next stage of the algorithm. The number of connected components decreases by at least a factor of two after every such step. Hence, we would need to make at most polylogarithmic advancements. This leads us to Borůvka's parallel algorithm for the minimum spanning tree [22].

ALGORITHM Equilevel-Independence: A Parallel NC Algorithm to find a state satisfying B

```
vector function GetSatisfying(B: predicate, L: Lattice)
    var G: vector of int initially ∀i : G[i] = 0;
    while ¬B(G) do
        if (G is the top element of L)
            then return null;
        else
            forall i ∈ independent(G) in parallel do
                G[i] := G[i] + 1 where helpful(i, G, B);
    endwhile;
    return G; // an optimal solution
```

6 SOLITARY PREDICATES

A *solitary* predicate is one which is either false on all the elements of the lattice, or is true on a single element in the lattice.

Definition 6.1 (Solitary Predicate). A boolean predicate B is *solitary* with respect to a lattice $\mathcal{L}$ iff

$$\forall G, H \in \mathcal{L} : B(G) \wedge B(H) \Rightarrow (G = H).$$

The above definition states that if the predicate is true for two elements G and H in the lattice, then G must be equal to H. This definition also includes the empty predicate that is not true in any element of the lattice.

Solitary predicates are closely related to the problem of Unique SAT. Unique SAT (USAT) [23] asks for the satisfiability of a boolean expression given the promise that it has either a single satisfying assignment or none. Let $x_1, x_2, \ldots, x_n$ be n boolean variables. Let B be a boolean expression on these variables. The USAT problem requires the algorithm to return 1, if the boolean expression has a unique satisfying assignment, 0, if the boolean expression is not satisfiable and either 0 or 1, if it has multiple satisfying assignments. Valiant and Vazirani [23] have shown the following result.

Theorem 6.2. ***(Valiant-Vazirani**[23]) If there exists a randomized polynomial-time algorithm for solving instances of SAT with at most one satisfying assignment, then* NP=RP.

As a direct application, we get the following result by using the construction in [24] as shown below.

Theorem 6.3. *Given any finite distributive lattice $\mathcal{L}$ generated from a poset P, and a solitary predicate B, there does not exist a randomized polynomial time algorithm to detect B unless* NP=RP.

Proof. Suppose, if possible, there exists a randomized polynomial time algorithm $\mathcal{A}$ to detect a solitary predicate. Given any instance of USAT on n variables $x_1, x_2, \ldots, x_n$, we construct a poset with n events on n processes, $P_1, P_2, \ldots, P_n$ with one event per process. Process P_i hosts the variable x_i which is initially set to false. When the event is executed, the variable is set to true. The distributive lattice generated from this poset has 2^n elements — one for every truth assignment. If the given instance of USAT has zero satisfying assignments, then $\mathcal{A}$ is guaranteed to return $false$. If the given instance of USAT has exactly one satisfying assignment, then $\mathcal{A}$ is guaranteed to return $true$ with a satisfying assignment. Otherwise, we do not care what $\mathcal{A}$ returns. Therefore, any randomized polynomial time algorithm to detect a solitary predicate can be used to solve USAT in polynomial time. □

7 SOLITARY PREDICATES WITH EFFICIENT ADVANCEMENT PROPERTY

Although detecting a general solitary predicate is hard, we can detect the predicate B, whenever it satisfies the efficient advancement property [2]. The efficient advancement property requires an efficient algorithm A such that whenever the predicate is false on an element $G \in L$, the algorithm returns an index i such that the predicate is false for all $H \geq G$ such that $H[i]$ equals $G[i]$.

Definition 7.1 (Efficient Advancement Property [2]). A boolean predicate B has the efficient advancement property with the polytime algorithm $\mathcal{A}$ with respect to a finite distributive lattice $\mathcal{L}$

generated from a poset P if

$$\forall G \in \mathcal{L} : \neg B(G) \Rightarrow \forall H \geq G : H[\mathcal{A}(G)] \neq G[\mathcal{A}(G)] \vee \neg B(H).$$

The above definition states that whenever B is false in G, the algorithm $\mathcal{A}(G)$ returns an index such that any global state H greater than G that matches G on that index also has B false. We will restrict ourselves to online algorithms, i.e., the algorithm $\mathcal{A}$ will only have access to those global states preceding G.

Any problem that can be modeled using a solitary predicate with the efficient advancement property can be solved in polynomial time. We restate the result from [2] in terms of solitary predicates.

Theorem 7.2. *[2] Let B be a solitary predicate with the efficient advancement property on a distributive lattice $\mathcal{L}$. Then B can be detected in polynomial time.*

Observe that the method outlined in Theorem 7.2 is *sequential* and online. However, it is easy to develop a parallel algorithm to detect a solitary predicate with the efficient advancement property. Following [16], we call an index i *forbidden* in G if the efficient advancement property returns i. Observe that multiple indices may be forbidden in G.

ALGORITHM Solitary: An Online Parallel Algorithm for detecting Solitary predicates with the efficient advancement property.

```
var G: element of L initially ⊥;
while ¬B(G) do
    forall i: forbidden(G, i) do in parallel
        if G cannot be advanced on i then return false;
        else advance G on i;
```

Now consider lattice-linear predicates that are true on multiple elements in the lattice. Let B be any lattice-linear predicate. We know that B is closed under the meet operation of the lattice. Suppose that B becomes true in the lattice and the least element that satisfies B is G_{min}. We derive another predicate B' from B that holds only for the element G_{min} whenever it exists. Thus, B' holds for G iff $B(G)$ and for all elements H, if $B(H)$, then $G \leq H$. Therefore, $B'(G)$ is false for all other elements besides G_{min}.

Theorem 7.3. *Let B be any lattice-linear predicate. Let $B'(G)$ be defined as follows:*

$$B'(G) \equiv B(G) \wedge (\forall H : B(H) \Rightarrow G \leq H).$$

Then, B' is a solitary predicate.

Proof. First, consider the case when B does not hold for any element in the lattice. This implies that B' also does not hold for any element in the lattice and is, therefore, solitary. Now, suppose that B is true in the lattice. Since B is an LLP predicate, it is closed under meets. Therefore, there is the smallest element, G_{min}, in the lattice such that $B(G_{min})$ holds. It is readily verified that $B'(G)$ holds iff $G = G_{min}$. □

Thus, all the instances of lattice-linear predicates provide us examples of solitary predicates when we focus on the least element.

Example 1: Man-Optimal Stable Marriage. In this problem, we are given as input n men and n women. We are also given a list of men preferences as $mpref$ where $mpref[i][k]$ denotes k^{th} top choice of man i. The women preferences are more convenient to express as a *rank* array where $rank[i][j]$ is the rank of man j by woman i. A matching between man and woman is stable if there is no *blocking pair*, i.e., a pair of woman and man such that they are not matched and prefer each other to their spouses. The underlying lattice for this example is the set of all n dimensional vectors of $1..n$. We let $G[i]$ be the choice number that man i has proposed to. Initially, $G[i]$ is 1 for all men.

If we now focus on the man-optimal stable marriage, then the predicate "the assignment is a man-optimal stable marriage" is a solitary predicate. The predicate $B_{stableMarriage}$ for the stable marriage is given by,

$$B_{stableMarriage} \equiv \forall j : \neg forbidden(G, j)$$

where $forbidden(G, j)$ is defined as

$$(\exists i : \exists k \leq G[i] : (z = mpref[i][k]) \wedge (rank[z][i] < rank[z][j])),$$

with $z = mpref[j][G[j]]$.

The predicate says that a marriage given by the vector G is stable if none of its index j is forbidden. The index j is forbidden if the woman z corresponding to man j's preference in $G[j]$ is also equal to the preference of man i in G, or a global state before G, and the woman z prefers i to j.

We now define the predicate for the man-optimal stable marriage, $B_{mosm}(G)$ as

$$B_{stableMarriage}(G) \wedge (\forall H : B_{stableMarriage}(H) \Rightarrow G \leq H).$$

By definition, it is clear that B_{mosm} is a solitary predicate. Given a lattice $\mathcal{L}$, we can search for the element satisfying B_{mosm} by searching for $B_{stableMarriage}$. We can use Algorithm Solitary for this purpose.

Example 2: Housing Allocation Problem. As another example of a solitary predicate, consider the housing allocation problem with n agents and n houses proposed by Shapley and Scarf [25]. The housing market is a matching problem with one-sided preferences. Each agent a_i initially owns a house h_i for $i \in \{1, n\}$ and has a completely ranked list of houses. The list of preferences of the agents is given by $pref[i][k]$ which specifies the k^{th} preference of the agent i. The goal is to come up with an optimal house allocation such that each agent has a house and no subset of agents can improve the satisfaction of agents in this subset by exchanging houses within the subset.

It is well-known that the housing market has a unique solution, called the *core* of the game. We model the housing market problem as that of predicate detection in a computation [7]. Each agent proposes to houses in the decreasing order of preferences. These proposals are considered as events executed by n processes representing the agents. Thus, we have n events per process. The global state corresponds to the number of proposals made by each of the agents. Let $G[i]$ be the number of proposals made by the agent i. We assume that in the initial state, every agent has made his first proposal. Thus, the initial global state $G = [1, 1, .., 1]$. We extend the notation of indexing to subsets $J \subseteq [n]$ such that $G[J]$ corresponds to the subvector given by indices in J. A global state G satisfies *matching* if every agent proposes a different house. We generalize *matching* to refer to a subset of agents rather than the

entire set. Let $J \subseteq [n]$. Then, $submatching(G, J)$ iff the houses proposed by agents in J is a permutation of indices in J. Intuitively, if $submatching(G, J)$ holds, then all agents in J can exchange houses within the subset J. For all G, there always exists a nonempty J such that $submatching(G, J)$. Let $B_{housing}(G)$ be defined as G is a matching and

$$(\forall F < G : \forall J \subseteq [n] : submatching(F, J) \Rightarrow F[J] = G[J]).$$

It is easy to show that $B_{housing}(G)$ is a solitary predicate with an efficient advancement property. Hence, the housing allocation problem also can be modeled and solved using the solitary predicates with the efficient advancement property. An agent i is forbidden in the global state G if the agent wishes a house that is part of the submatching in G.

We briefly discuss detection of predicates when the search starts from the top of the lattice in addition to the bottom of the lattice. Many predicates, such as $B_{stableMarriage}$ and conjunctive predicates, satisfy not only the efficient advancement property but also its dual. Equivalently, the set of elements satisfying these predicates are closed not only for the meet operation but also for the join operation. If we are okay with returning either of the elements as our final answer, then searching for any of the element in parallel can speed up the algorithm by a factor of the height of the lattice. The dual of the efficient advancement property can formally be defined as [17] follows.

Definition 7.4 (dual of Efficient Advancement Property). A boolean predicate B has the dual of efficient advancement property with the polytime algorithm $\mathcal{A}$ with respect to a finite distributive lattice $\mathcal{L}$ generated from a poset P if

$$\forall G \in \mathcal{L} : \neg B(G) \Rightarrow \forall H \leq G : H[\mathcal{A}(G)] \neq G[\mathcal{A}(G)] \vee \neg B(H).$$

The above definition states that whenever B is false in G, the algorithm $\mathcal{A}(G)$ returns an index such that any global state H less than G that matches G on that index also has B false. When the efficient advancement as well as its dual are true, one can search for the satisfying element starting from both the bottom and the top of the lattice as shown in Algorithm Solitary2.

ALGORITHM Solitary2: A Parallel Algorithm for detecting Predicates with efficient advancement property and its dual.

```
var G: element of L initially ⊥ (the bottom element of L);
    Z: element of L initially ⊤ (the top element of L);
    while ¬B(G) and ¬B(Z) do
        forall i: forbidden(G, i) do in parallel
            if G cannot be advanced on i then return false;
            else advance G on i;
        forall j: dual-forbidden(Z, j) do in parallel
            if Z cannot be retreated on j then return false;
            else retreat Z on j;
    endwhile;
    if B(G) then return G else return Z; // an optimal solution
```

Applying the idea to the stable marriage problem, one can search for the stable marriage starting from the top choices for all men (the $\perp$ element) of the lattice in parallel with the bottom choices for all men (the $\top$) element of the lattice. This algorithm will traverse the distance in a lattice given by the minimum of the distance of a stable marriage from the top or the bottom. Hence, depending upon the distance of man-optimal and man-pessimal stable marriage, it will return the stable marriage that is closer to the bottom or the top of the lattice.

Similarly, consider the problem of detecting conjunctive predicates which are also closed under the join operation of the lattice. By using the Algorithm Solitary2, we again get an algorithm to detect a conjunctive predicate that will traverse the lattice given by the minimum of the distance of a satisfying global state from the top or the bottom of the lattice. In the above analysis, we are ignoring the factor of 2 penalty that we incur because we run the algorithm both from the bottom and the top of the lattice.

8 SOLITARY PREDICATES WITH NC ADVANCEMENT PROPERTIES

We now define a special case of the efficient advancement property. A predicate B has *the* NC *advancement* property, if (1) there exists an NC algorithm to detect whether an index is forbidden and (2) starting from the initial state and always advancing all the forbidden indices the algorithm either reaches a satisfying state or the top element of the lattice in the polylogarithmic number of steps in the size of the input. Clearly, any predicate that has the efficient NC advancement property can be detected in parallel in NC time with the Algorithm Solitary.

We now give several examples guaranteeing the NC advancement property. We say that a predicate B has an *antimonotone advancement* property if once an index i is not forbidden in G, it stays not forbidden for all $H \geq G$ such that $H[i]$ equals $G[i]$. Formally,

Definition 8.1 (Antimonotone Advancement Property). A boolean predicate B with the advancement property (w.r.t. a poset P) is *antimonotone* if

$$\forall G \in \mathcal{L} : \neg forbidden(G, B, i) \Rightarrow \forall H \geq G \wedge (H[i] = G[i]) : \neg forbidden(H, B, i).$$

Consider the problem of the minimum spanning tree in an undirected graph when all edges have *unique weights*. If the graph is connected, then there is a unique minimum spanning tree. To find this spanning tree, we consider the boolean lattice formed from all edges. We assume that edges are given to us in the increasing order. We define the predicate B on a subset of edges G as true whenever G forms the minimum spanning tree. We use the binary representation of G: The variable $G[i]$ equals 1 iff the i^{th} edge is part of the unique minimum spanning tree. The predicate B is solitary because either there is a unique minimum spanning tree or no spanning tree in such a weighted undirected graph.

The advancement algorithm $\mathcal{A}$ can be defined as follows: We first define a sequence of subsets $E_i = \{e_1, e_2, \ldots e_{i-1}\}$, where e_1, e_2, e_{i-1} are the edges in the graph in the sorted order. Now, let G be any subset of edges and suppose that the predicate B is false on G. Let $G[i]$ correspond to the edge (v_j, v_k). If $G[i]$ equals 0 and the vertices v_j and v_k are not connected with the edges in E_i, then we set $G[i]$ to 1. The algorithm A is in NC since it only checks for connectivity based on a subset of edges which can be done in polylogarithmic depth via computing the transitive closure.

We now show that any solitary predicate with Antimonotone advancement property on a poset with height polylogarithmic in n can be detected in NC.

Theorem 8.2. *Let P be any poset and B be an antimonotone advancement predicate that can be checked in* NC. *If the height of the poset is $O(\log(n^k))$, for any constant k, then the Algorithm Solitary is in* NC.

Proof. The *forall* statement in the algorithm Solitary can run at most $O(1)$ time because if a process is forbidden then it must advance. A process can advance at most $O(\log(n^k))$ times because the height of the poset is $O(\log(n^k))$. If a process is not forbidden, then it stays not forbidden due to the antimonotonicity property. Since checking for the property is in NC, the entire algorithm is in NC. □

When we apply this algorithm to the minimum spanning tree problem, we get the NC algorithm Parallel-MST. The algorithm assumes that edges of the graph are presented in the sorted order similar to Kruskal's algorithm [26] In this example, the predicate

ALGORITHM Parallel-MST: Finding the minimum spanning tree in a graph in NC .

```
var G: array[1..m] of {0, 1} initially ∀j : G[j] := 0;
// Edges G are assumed to be in increasing order of weights
forall edges e_i = (v_j, v_k): do in parallel
    if there is no path from v_j to v_k with edges 1..j − 1 then
        G[j] := 1;
```

B on a set of edges G is defined to be "G forms the minimum spanning tree in the graph." Since all edge weights are unique, B is a solitary predicate. In the boolean lattice of all edges, an edge from v_j to v_k is forbidden if there is no path from v_j to v_k using edges with weight lower than the weight of the edge (v_j, v_k). Such an edge is always included as part of the minimum spanning tree. Furthermore, the predicate satisfies antimonotone advancement property. If an edge (v_j, v_k) is not forbidden, then it continues to stay not forbidden. Finally, the poset corresponding to a Boolean lattice is always $O(1)$ height, and therefore by Theorem 8.2, we have an NC algorithm. Many NC algorithms are already known for the minimum spanning tree problem. Our goal was to show how the notion of solitary predicates in a lattice with $O(1)$ height and *antimonotone* advancement leads to an NC algorithm.

We now discuss another property of predicate that allows us to detect it in NC: *efficient rejection*. A solitary predicate that has the efficient rejection property can be detected in NC even when the height of the poset is not $O(\log(n))$. We show that the problem of finding the least global state that satisfies the conjunctive predicate and the problem of reachability in a directed graph satisfy the efficient rejection property.

As a concrete example, consider *conjunctive predicates*. We are searching for the least global state that satisfies

$$B_{conj} = l_1 \wedge l_2 \wedge \ldots l_n.$$

Since we are only interested in the least global state, we can view it as a solitary predicate with B_{conj} appropriately refined as

$$B_{fconj}(G) = B_{conj}(G) \wedge \forall H : B_{conj}(H) \Rightarrow (G \leq H).$$

We start with the notion of a rejection relation. A state s rejects a state t if whenever all local states less than or equal to s are forbidden, then all states less than or equal to t are also forbidden. If we know the initial forbidden states and the rejection relation, we can compute the states on each process that are forbidden in the initial state or become forbidden when the processes are advanced on the forbidden states. This is done by computing the reflexive transitive closure of the rejection relation. Then, the first local state that is not rejected on every process gives us the least global state that satisfies the predicate. Formally,

Definition 8.3 (Rejection Relation). Given any distributive lattice $\mathcal{L}$ with realizer $P_1, \ldots, P_n$ and a predicate B, for $s \in P_i$ and $t \in P_j$ we define,

$$rejects(s,t) \equiv \forall s' \preceq_i s : \forall G \in \mathcal{L} : forbidden(G, s', B) \Rightarrow \forall t' \preceq_j t : \forall H \in \mathcal{L} : forbidden(H, t', B),$$

where $\preceq_i$ is the reflexive order associated with the chain P_i.

We say that a lattice-linear predicate satisfies *efficient rejection* if

Definition 8.4 (Lattice-linear Predicate with Efficient Rejection). A lattice linear boolean predicate B satisfies *efficient rejection* with respect to a lattice $\mathcal{L}$ if there exists a rejection relation such that $rejects(s,t)$ can be computed in NC.

Algorithm LLPwithRejection detects a lattice-linear predicate with efficient rejection in NC. This algorithm is similar to the one proposed in [27] where the algorithm is proposed for conjunctive predicates and we refer the reader to [27] for details. We show that the algorithm is applicable to any lattice-linear predicate with efficient rejection. Thus, it is also applicable to finding the shortest path in a directed graph.

ALGORITHM LLPwithRejection: An NC algorithm to find the first consistent cut that satisfies an lattice-linear predicate with the rejection relation.

```
Output: Consistent Global State as array G[1 . . . n]
var
    G : array[1 . . . n] of 1 . . . m init 1;
    R : [(1 . . . n, 1 . . . m), (1 . . . n, 1 . . . m)] of 0 . . . 1
            init R[(i, j), (i′, j′)] = 1 iff (i, j) rejects (i′, j′);
    valid : array[[1 . . . n][1 . . . m] of 0 . . . 1 init
 ∀i, j : valid[i][j] := 1;
    RT : array[(1 . . . n, 1 . . . m), (1 . . . n, 1 . . . m)] of 0 . . . 1;
RT := TransitiveClosure(R);
for all (i ∈ 1 . . . n, i′ ∈ 1 . . . n, j′ ∈ 1 . . . m) in parallel do
    if forbidden(i, 1) ∧ (RT[(i, 1), (i′, j′)] = 1) then
 valid[i′][j′] := 0;
for all (i ∈ 1 . . . n) in parallel do
    if ∀j ∈ 1 . . . m : (valid[i][j] = 0) then return false;
    else G[i] := min {j | valid[i][j] = 1};
```

We now show how lattice-linear predicates with efficient rejection can be applied to graph reachability. Suppose that we are given a fixed vertex v_0 in a directed graph (V, E). Our goal is to find all the vertices that are reachable from v_0. We can use simple BFS to find all reachable vertices.

$$B_{traverse} \equiv G[0] \wedge (\forall (v_i, v_j) \in E : G[j] \geq G[i])$$

However, this procedure takes time proportional to the diameter of the graph. By computing the reflexive transitive closure of the binary graph, we can find all reachable vertices in polylogarithmic time. Let A be the graph in the matrix form. We compute G, the reflexive transitive closure of the matrix A, such that $G[i, j]$ equals 1 iff the vertex v_j is reachable from the vertex v_i. We define $B_{closure}$ as $\forall i, j : G[i, j] \geq \max(A[i, j], \max\{G[i, k] \wedge G[k, j] \mid k \in [0..n-1]\})$. For this problem, our poset has n^2 processes and each process has just one event. The process $P_{(i,j)}$ is forbidden if $G[i, j]$ equals 0 and $A[i, j]$ equals 1 or for some k, $G[i, k]$ and $G[k, j]$ are both 1. We now claim that $B_{closure}$ is an LLP with efficient rejection. If for any (i, j), $A[i, j]$ is 1, then the state $G[k, i]$ rejects $G[k, j]$ (i.e., if i is reachable from k, then j is also reachable from k). Thus, whenever all states equal to or prior to $G[k, i] = 0$ are forbidden, then so are the states equal to or prior to $(G[k, j] = 0)$. The rejection relation can be determined in $O(1)$ time since we can compute $A[i, j]$ in $O(1)$ time. Thus, we have that the predicate $B_{closure} \equiv \forall i, j : G[i, j] \geq \max(A[i, j], \max\{G[i, k] \wedge G[k, j] \mid k \in [0..n-1]\})$ is a lattice-linear predicate with efficient rejection.

9 CONCLUSIONS AND OPEN PROBLEMS

We have defined a class of predicates called Equilevel predicates on finite distributive lattices. We have also identified subclasses of equilevel predicates that can be detected efficiently in parallel. There are many problems that are open in this area. Are there other subclasses of equilevel predicates or solitary predicates that admit efficient detection? What other problems can be modeled as equilevel predicate detection and do they introduce any specific algorithmic challenges? On the other end of the spectrum, what properties preclude efficient parallel detection of equilevel predicates? In this work, we made general statements about the NP-hardness of detecting equilevel predicates. However, similar to how we identified properties enabling parallel detection of these predicates, it would be insightful to identify properties that certify P-hardness (see [28] for an introduction) of the detection of corresponding subclasses of equilevel predicates which in turn would rule out the possibility of an efficient parallel solution unless NC=P.

REFERENCES

[1] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, UK, 1990.

[2] Vijay K. Garg. Predicate detection to solve combinatorial optimization problems. In Christian Scheideler and Michael Spear, editors, *SPAA '20: 32nd ACM Symposium on Parallelism in Algorithms and Architectures, Virtual Event, USA, July 15-17, 2020*, pages 235–245. ACM, 2020.

[3] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[4] Dan Gusfield and Robert W Irving. *The stable marriage problem: structure and algorithms*. MIT press, 1989.

[5] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, Dec 1959.

[6] V. K. Garg and B. Waldecker. Detection of weak unstable predicates in distributed programs. *IEEE Trans. on Parallel and Distributed Systems*, 5(3):299–307, March 1994.

[7] Vijay K. Garg. A lattice linear predicate parallel algorithm for the housing market problem. In Colette Johnen, Elad Michael Schiller, and Stefan Schmid, editors, *Stabilization, Safety, and Security of Distributed Systems - 23rd International Symposium, SSS 2021, Virtual Event, November 17-20, 2021, Proceedings*, volume 13046 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2021.

[8] Arya T. Gupta and Sandeep S. Kulkarni. Extending lattice linearity for self-stabilizing algorithms. In Colette Johnen, Elad Michael Schiller, and Stefan Schmid, editors, *Stabilization, Safety, and Security of Distributed Systems - 23rd International Symposium, SSS 2021, Virtual Event, November 17-20, 2021, Proceedings*, volume 13046 of *Lecture Notes in Computer Science*, pages 365–379. Springer, 2021.

[9] Vijay K. Garg. A lattice linear predicate parallel algorithm for the dynamic programming problems. In *Proc. of the Int'l Conf. on Distributed Computing and Networking (ICDCN)*, Delhi, India, 2022. Springer-Verlag.

[10] David R. Alves and Vijay K. Garg. Parallel minimum spanning tree algorithms via lattice linear predicate detection. In *Proc. Parallel and Distributed Combinatorics and Optimization (PDCO), June 2022*, Lyon, France, 2022.

[11] Vijay K. Garg. Keynote talk: Lattice linear predicate algorithms for the constrained stable marriage problem with ties. In *24th International Conference on Distributed Computing and Networking, ICDCN 2023, Kharagpur, India, January 4-7, 2023*, pages 2–11. ACM, 2023.

[12] Arya T. Gupta and Sandeep S. Kulkarni. Multiplication and modulo are lattice linear. *CoRR*, abs/2302.07207, 2023.

[13] Arya Tanmay Gupta and Sandeep S. Kulkarni. Lattice linearity in assembling myopic robots on an infinite triangular grid. *CoRR*, abs/2307.13080, 2023.

[14] Robert Cooper and Keith Marzullo. Consistent detection of global predicates. *ACM SIGPLAN Notices*, 26(12):167–174, 1991.

[15] V. K. Garg and B. Waldecker. Detection of unstable predicates. In *Proc. of the Workshop on Parallel and Distributed Debugging*, Santa Cruz, CA, May 1991.

[16] Craig M. Chase and Vijay K. Garg. Detection of global predicates: Techniques and their limitations. *Distributed Comput.*, 11(4):191–201, 1998.

[17] V. K. Garg and N. Mittal. On slicing a distributed computation. In *21st Intnatl. Conf. on Distributed Computing Systems (ICDCS' 01)*, pages 322–329, Washington - Brussels - Tokyo, April 2001. IEEE.

[18] B. Charron-Bost, C. Delporte-Gallet, and H. Fauconnier. Local and temporal predicates in distributed systems. *ACM Trans. on Programming Languages and Systems*, 17(1):157–179, January 1995.

[19] G. Birkhoff. *Lattice Theory*. Providence, R.I., 1940. first edition.

[20] W.T. Trotter. *Combinatorics and Partially Ordered Sets: Dimension Theory*. The Johns Hopkins University Press, 1992.

[21] Stephen Fenner, Rohit Gurjar, and Thomas Thierauf. A deterministic parallel algorithm for bipartite perfect matching. *Communications of the ACM*, 62(3):109–115, 2019.

[22] Jaroslav Nešetřil, Eva Milková, and Helena Nešetřilová. Otakar Borůvka on minimum spanning tree problem Translation of both the 1926 papers, comments, history. *Discrete Mathematics*, 233(1-3):3–36, 2001.

[23] Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 458–463. ACM, 1985.

[24] Sujatha Kashyap and Vijay K. Garg. Intractability results in predicate detection. *Inf. Process. Lett.*, 94(6):277–282, 2005.

[25] Lloyd Shapley and Herbert Scarf. On cores and indivisibility. *Journal of mathematical economics*, 1(1):23–37, 1974.

[26] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[27] Vijay K. Garg and Rohan Garg. Parallel algorithms for predicate detection. In R. C. Hansdah, Dilip Krishnaswamy, and Nitin H. Vaidya, editors, *Proceedings of the 20th International Conference on Distributed Computing and Networking, ICDCN 2019, Bangalore, India, January 04-07, 2019*, pages 51–60. ACM, 2019.

[28] Raymond Greenlaw, H James Hoover, and Walter L Ruzzo. *Limits to parallel computation: P-completeness theory*. Oxford University Press, USA, 1995.