Applying Predicate Detection to Discrete Optimization Problems

Vijay K. Garg

Parallel and Distributed Systems Lab,
Department of Electrical and Computer Engineering,
The University of Texas at Austin.

Motivation

Consider the following problems:

- Shortest Path Problem:
  Input: a weighted directed graph and a source vertex
  Output: Least Cost of reaching any vertex i
  Dijkstra's algorithm for graph with non-negative weights,
  Bellman-Ford algorithm for graphs with no negative cycles
- Stable Marriage Problem:
  Input: ordered preferences of n men and n women
  Output: Man-optimal stable marriage
  Gale-Shapley's algorithm
- Assignment Problem:
  Input: n items, n bidders with valuation for items
  Output: Least market clearing prices
  Hungarian Algorithm (or Gale-Demange-Sotomayor's Auction)

Could there be a single algorithm that solves all of these problems?

Lattice-Linear Predicate (LLP) Algorithm

Other Applications of LLP

- Housing Allocation Problem:
  Input: n agents, n houses, initial endowment, preference list of agents
  Output: allocation of houses such that there is no blocking group
  Gale's Top Trading Cycle Algorithm
- Minimum Spanning Tree Problem:
  Input: undirected weighted graph
  Output: spanning tree with that minimizes sum of weight of edges
  Prim's Algorithm, Boruvka' Algorithm
- Horn Satisfiability:
  Input: A boolean formula in Horn form
  Output: Least satisfying assignment, if any
  Horn's Satisfiability Algorithm

Outline of the Talk

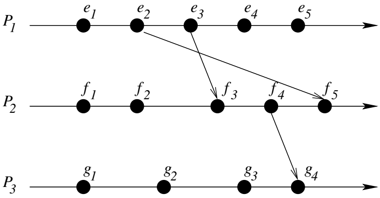
- What are Lattice-Linear Predicates (LLP)?
- LLP Detection Algorithm
- Applications
- Enumerating All Satisfying Global States

Steps of Using LLP Algorithm

- Step 1: Model the underlying search space. A Distributive Lattice of State Vectors. The order on the lattice is based on the optimization objective of the problem.
- Step 2: Define the feasibility predicate B. An element is feasible if it satisfies constraints of the problem
- Step 3: Check whether the feasibility predicate B is Lattice-Linear. If B is lattice-linear, LLP Algorithm will return the optimal feasible solution.

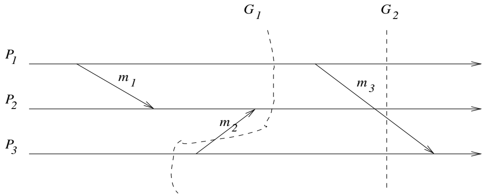
Step 1: Modeling the underlying search space

Model the problem as n processes choosing their component in a vector of size n. The choice for a single process is total ordered.



computation: poset $(E, \rightarrow)$
candidate solution: a possible global state of the system.

Consistent Global State

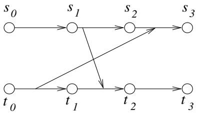


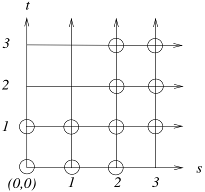
A subset G of E is a consistent global state if

$$\forall e, f \in E : (f \in G) \land (e \rightarrow f) \Rightarrow (e \in G)$$

The set of all consistent global states forms a finite distributive lattice.

Step 1: Order on the underlying space





$(L, \leq)$: Underlying Search Space
L : set of all consistent global state vectors
Order on Global State $G \leq H$ iff for all $i : G[i] \leq H[i]$.
meet of two global states: $K = G \sqcap H = \min(G, H)$
join of two global states: $K = G \sqcup H = \max(G, H)$
meet distributes over join.
$(L, \leq)$ is a distributive lattice.

Step 1: Examples

G: Global State Vector where $G[i]$ is the component for process i.

- Shortest Path: $G[i]$: cost of reaching vertex i from the source vertex
  initially 0
- Stable Marriage: $G[i]$: index in the preference list for man i
  initially 1 // top choice
- Market Clearing Prices: $G[i]$: price of item i
  initially 0

Step 2: Defining Feasibility Predicate B

A global state G satisfies B iff G represents a feasible solution.

- Shortest Path: All nodes most have a parent node. For every vertex j (except source): there exists a vertex i such that $G[j] \geq G[i] + w[i, j]$.
- Stable Marriage: Every man must be matched to a different woman and there must not be any blocking pair.
- Market Clearing Prices: There is no overdemanded item at that pricing vector.

Step 2: Defining Feasibility Predicate Formally

- Shortest Path: Every non-source node has a parent. For any node $j \neq 0$,

$$\exists i \in pre(j) : G[j] \geq G[i] + w[i,j]$$

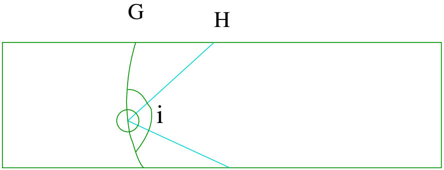
- Stable Marriage: Every man must be matched to a different woman and there must not be any blocking pair. For any man j, let $z = mpref[j][G[j]]$; //current woman assigned to man j

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

- Market Clearing Prices: There is no overdemanded item at that pricing vector. For any item j,

$$\neg\exists J : minimalOverDemanded(J, G) \wedge (j \in J)$$

Lattice-Linearity for Predicate Detection



Forbidden State The state at P_i is forbidden at G with respect to B if unless P_i is advanced B cannot become true.

$$\text{forbidden}(G, i, B) \equiv \forall H : G \subseteq H : (G[i] = H[i]) \Rightarrow \neg B(H)$$

Lattice-Linear Predicates A predicate B is lattice-linear if for all consistent cuts G,

$$\neg B(G) \Rightarrow \exists i : \text{forbidden}(G, i, B).$$

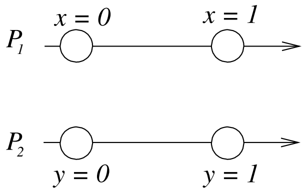
Examples: Conjunctive Predicates: $l_1 \wedge l_2 \wedge ... \wedge l_n$, Feasible Path, Stable Matching, Market Clearing Prices, Minimum Spanning Tree, Housing Core, Horn Formulas

Examples of Lattice-Linear Predicates

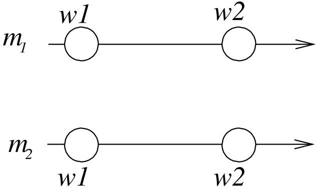
- A conjunctive predicate
  $l_1 \wedge l_2 \wedge \ldots \wedge l_n$, where l_i is local to P_i.
  Suppose G is not feasible. Then, there exists j such that l_j is false in G. The index j is forbidden in G.
- Shortest Path
  Any j such that v_j does not have a parent,
  $(\forall i \in pre(j) : G[j] < G[i] + w[i,j])$ is forbidden in G.
- Stable Marriage
  j is forbidden in G if
  $\exists i : \exists k \leq G[i] : (z = mpref[i][k]) \wedge (rank[z][i] < rank[z][j]))$
- Market Clearing Price
  $(\neg \exists J : minimalOverDemanded(J, G) \wedge (j \in J))$
  Any j in a minimal overDemanded set is forbidden.

Example of Predicates that are not Lattice-Linear

Example 1: $B(G) \equiv x + y \geq 1$



Example 2: $B(G) \equiv G$ is a matching.



Outline of the Talk

- What are Lattice-Linear Predicates (LLP)?
- LLP Detection Algorithm
- Applications
- Enumerating All Satisfying Global States

Detecting Lattice-Linear Predicates

(Advancement Property) There exists an efficient (polynomial time) algorithm to determine the forbidden state.

Theorem

[Chase and Garg 95] Any lattice-linear predicate that satisfies advancement property can be detected efficiently.

LLP Algorithm

How much to advance: j is forbidden in G until α iff

$$\forall H \in L : H \geq G : (H[j] < \alpha) \Rightarrow \neg B(H).$$

```
 vector function getLeastFeasible(T: vector, B: predicate)
//T: top element of the lattice
 var G: vector of reals initially ∀i : G[i] = 0;
 while ∃j: forbidden(G, j, B)  do
      for all j such that forbidden(G, j, B) in parallel:
           if (α(G, j, B) > T[j]) then return null;
           else G[j] := α(G, j, B);
 endwhile;
 return G; // the optimal solution
```

LLP Algorithm: Stable Marriage Problem

P_j:

var G: array[1..n] of 1..n;

input: $mpref[i,k]$: int for all i,k; // men preferences

$rank[k][i]$: int for all k,i; // women ranking

init: $G[j] := 1$;

always: $w = mpref[j][G[j]]$;

forbidden:

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

advance: $G[j] := G[j] + 1$;

Slightly more general than Gale-Shapley Algorithm:
instead of starting from $(1, 1, \ldots, 1)$, can start from any choice vector.

LLP Algorithm: Shortest Path Problem

input: $pre(j)$: list of $1..n$;
$w[i,j]$: positive int for all $i \in pre(j)$
$s : 1..n$; // source node;
init: $G[j] := 0$;
always:
$parent[j,i] = (i \in pre(j)) \wedge (G[j] \geq G[i] + w[i,j])$;
$fixed[j] = (j = s) \vee (\exists i : parent[j,i] \wedge fixed[i])$
$Q = \{(G[i] + w[i,k]) | (i \in pre(k)) \wedge fixed(i) \wedge \neg fixed(k)\}$;

forbidden: $\neg fixed[j]$
advance: $G[j] := \max\{\min Q, \min\{G[i] + w[i,j] \mid i \in pre(j)\}\}$

By ignoring the second part of advance, we can get Dijkstra's algorithm.

LLP Algorithm: Shortest Path Problem Revisited

Assume no negative cost cycle.

input: $pre(j)$: list of $1..n$;
    $w[i,j]$: int for all $i \in pre(j)$
init: if $(j = s)$ then $G[j] := 0$ else $G[j] :=$ maxint;
forbidden: $G[j] > \min\{G[i] + w[i,j] \mid i \in pre(j)\}$
advance: $G[j] := \min\{G[i] + w[i,j] \mid i \in pre(j)\}$

Lattice is reversed: the bottom element is $(maxint, maxint, \ldots, maxint)$
This is just Bellman-Ford's algorithm.

LLP Algorithm: Market Clearing Prices

input: $v[b, i]$: int for all b, i
init: $G[j] := 0;$
always: $E = \{(k, b) \mid \forall i : (v[b, k] - G[k]) \geq (v[b, i] - G[i])\};$
$\quad demand(U') = \{k \mid \exists b \in U' : (k, b) \in E\};$
$\quad overDemanded(J) \equiv \exists U' \subseteq U : (demand(U') = J) \wedge (|J| < |U'|)$

forbidden: $\exists J : minimal - OverDemanded(J) \wedge (j \in J)$
advance: $G[j] := G[j] + 1;$

This is just Demange-Gale-Sotomayor exact auction algorithm.

Properties of LLP Predicates

Lemma

Let B be any boolean predicate defined on a lattice L of vectors.

- *Let $f : L \to \mathrm{R}_{\geq 0}$ be any monotone function defined on the lattice L of vectors of $\mathrm{R}_{\geq 0}$. Consider the predicate $B \equiv G[i] \geq f(G)$ for some fixed i. Then, B is lattice-linear.*
- *Let L_B be the subset of the lattice L of the elements that satisfy B. If B is lattice-linear then L_B is closed under meets.*
- *If B_1 and B_2 are lattice-linear then $B_1 \wedge B_2$ is also lattice-linear.*

Constrained Optimization

- least stable marriage such that regret of Peter is less than or equal to regret of John
- least feasible path such that the cost of reaching x equals cost of reaching y
- least clearing prices such that $item_1$ is priced at least 5 more than $item_2$.

All of the additional constraints are also lattice-linear.

Lemma

LLP can be adapted to find the least vector G that satisfies $B_1 \wedge B_2$ for any lattice-linear predicates B_1, B_2.

Proof.

The algorithm LLP can be used with the following changes: $forbidden(G, j, B_1 \wedge B_2) \equiv forbidden(G, j, B_1) \vee forbidden(G, j, B_2)$, and $\alpha(G, j, B_1 \wedge B_2) = \max\{\alpha(G, j, B_1), \alpha(G, j, B_2)\}$.

□

Meet Closure of Feasible Predicates

Theorem

[Chase and Garg 95] A predicate B is lattice-linear implies that it is meet-closed (in the lattice of all consistent cuts).

Lattice-Linearity implies

- If G and H are feasible cost vectors, then so is $G \sqcap H$.
- If G and H are stable marriage choice vectors, then so is $G \sqcap H$.
- If G and H are market clearing prices, then so is $G \sqcap H$.

Dual of Lattice-Linearity

reverse-forbidden$(G, i, B) \equiv \forall H \in L : H \leq G : (G[i] = H[i]) \Rightarrow \neg B(H)$.
B is *dual-lattice-linear* iff:
$\forall G \in L : \neg B(G) \;\Rightarrow\; \exists i :$ reverse-forbidden(G, i, B).
$B_{stableMarriage}$ and $B_{marketClearing}$ are also dual-lattice-linear.
$\Rightarrow$

- the set of stable marriages and market clearing prices are also closed under joins
- one can traverse the lattice backwards to find the woman-optimal stable marriage or the greatest market clearing prices.

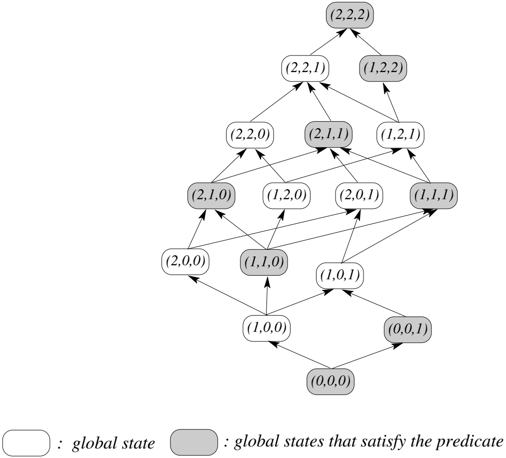
Note: $B_{shortestPath}$ is not dual-lattice-linear.

Outline of the Talk

- What are Lattice-Linear Predicates (LLP)?
- LLP Detection Algorithm
- Applications
- Enumerating All Satisfying Global States

Enumerating All Feasible Solutions

Assume that B is lattice-linear as well as dual-lattice-linear.

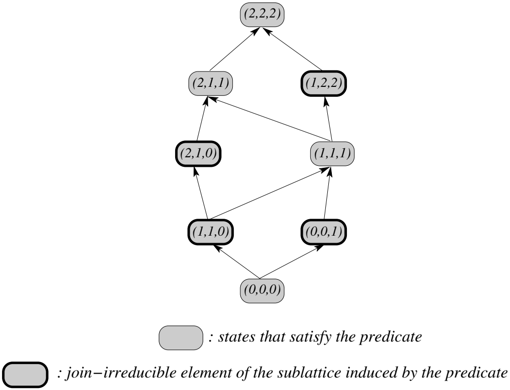


- L_B (the subset of elements in L that satisfy B) forms a sublattice of L
- L_B is a distributive lattice.

Slicing: Can we represent L_B concisely? [Mittal and Garg 01]

Join-irreducible Elements

join-irreducible element: cannot be represented as join of two other elements



Theorem

[Birkhoff's Representation Theorem] A distributive lattice can be recovered exactly *from the set of its join-irreducible elements.*

Algorithm to find All Join-Irreducible Elements

for all $e \in E$:
 compute $J(B, e)$

$J(B, e)$: the minimum global state of $(E, \leq)$ that

- satisfies B, and
- contains e

Feasible predicate: $B_e(G) \equiv B(G) \wedge (e \in G)$
Observation: B_e is a conjunction of two lattice-linear predicates.
We can use LLP algorithm to find the least global state satisfying $J(B, e)$

Applications of Slicing

- Constrained Stable Marriages: We get a generalization of rotation poset [Irving and Gusfield].
- Constrained Market Clearing Prices: A poset that captures all integral market clearing prices.

Conclusions

How to Solve Many Combinatorial Optimization Problems
Find the least feasible element

- View State space as the set of consistent global states
- Each process starts with the most desirable choice and moves to less desirable
- Define a "feasibility" predicate B
- Check if B satisfies the lattice-linearity condition

Other algorithms as special cases of the LLP Algorithm:

- Gale's Top Trading Cycle Algorithm,
- Prim's MST Algorithm,
- Horn's satisfiability algorithm,
- Johnson's algorithm to transform graphs with negative cost edges

Future Work

- Techniques when the feasibility predicate is not lattice-linear.