# Generation of Near-Optimal Solutions using ILP-Guided Sampling

**A. Srinivasan [1], G Shroff [2], L. Vig [2] and S. Saikia [2]**

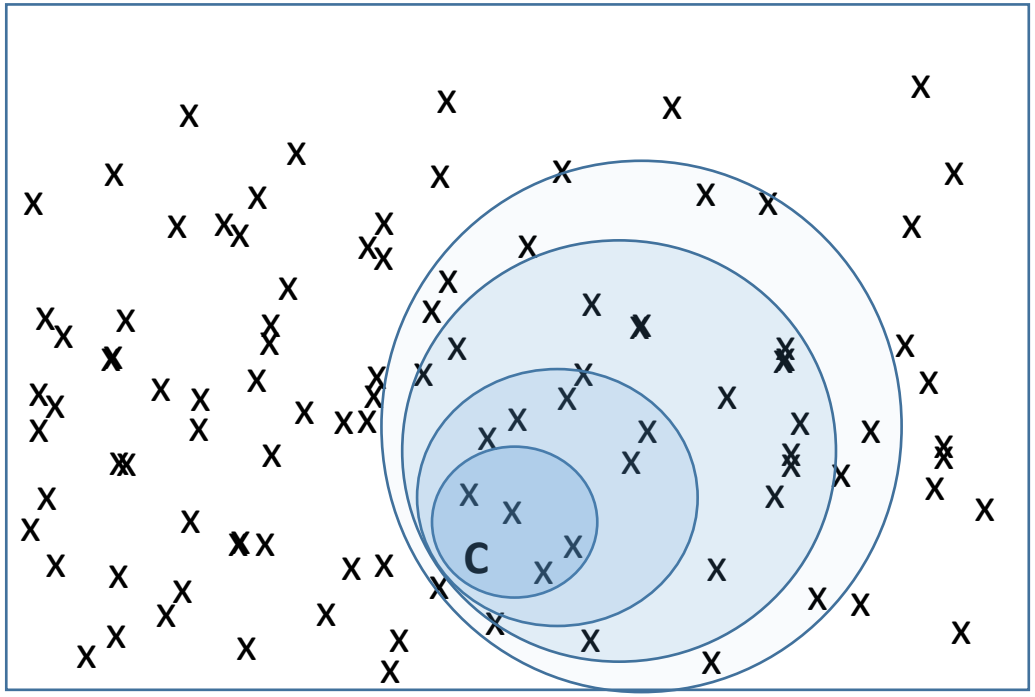
[1]BITS Pilani, Goa and [2]TCS Research

# What We Are Interested In

Real-world optimisation problems for which:

1. The space of  potential solutions is very large
2. There are very few optimal or near-optimal solutions
3. It is computationally expensive to obtain the cost of an instance
4. Analytical or numerical methods are not  easy to formulate, but there is domain-knowledge in the form of heuristics, rules-of-thumb  and so on, relevant to low-cost solutions

# The Main Idea

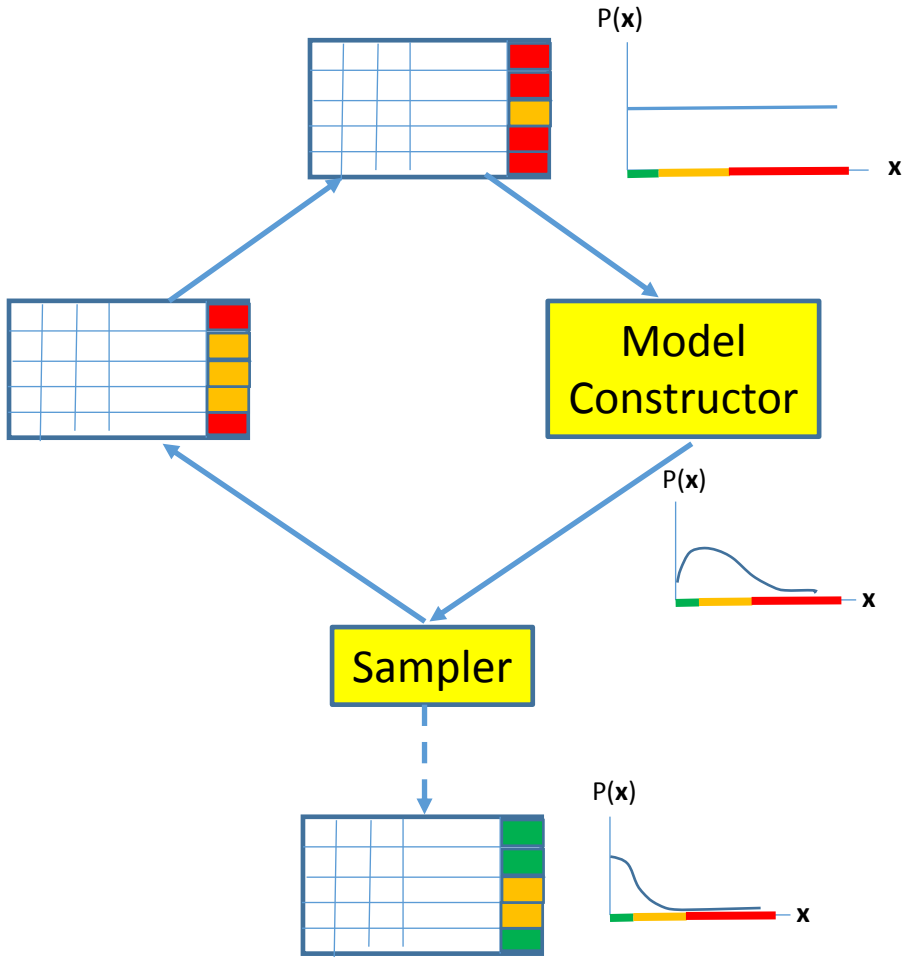



Adopt a sampling-based approach: we would like to generate instances in **C** (optimal, or near-optimal solutions)

We use domain-knowledge and ILP-constructed theories to sample from a sequence of concepts $C_1$ , $C_2$ …, $C_n$ = **C** that get progressively closer to the optimal set

For the full story:
https://arxiv.org/pdf/1608.01093.pdf

# Estimation of Distribution Algorithms (EDAs)



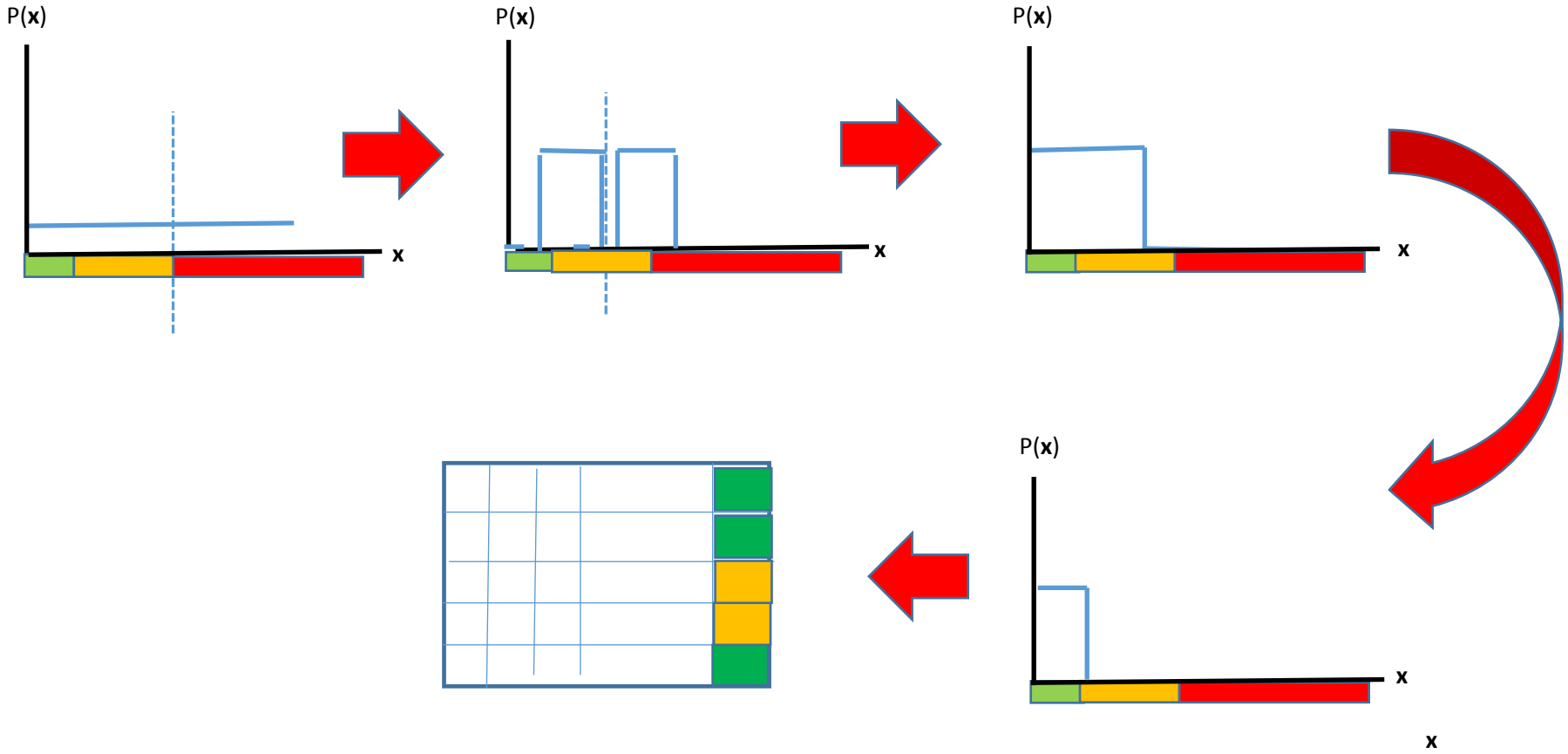

The model constructor builds a model that for progressively better solutions

The sampler uses the model to perform a form of “theory-guided sampling”

After each iteration, the probability of getting  better solutions increases (and worse solutions decreases)

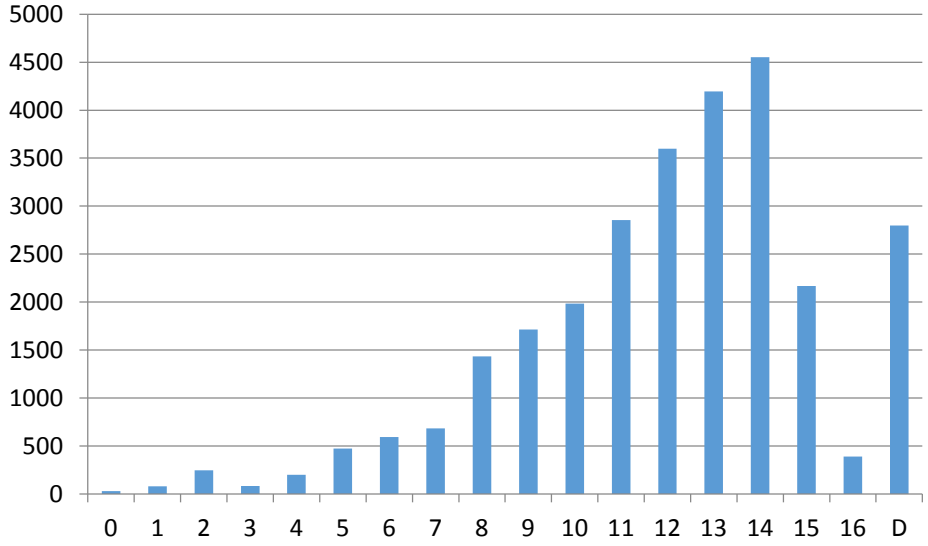
Finally, we should be sampling from a distribution that is largely skewed towards optimal or near-optimal solutions

# An ILP-Assisted EDA for Optimisation



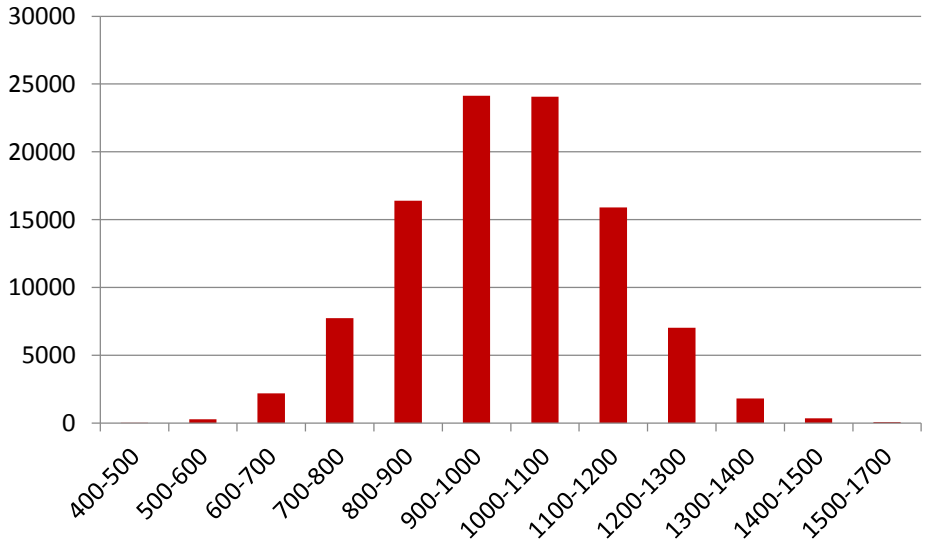
# Empirical Evaluation: Datasets

**KRK-win**




Total Instances: 28,056
Optimal: Depth=0 (1/1000)

**5x5 JobShop**




Total Instances: 100,000
(Near) Optimal: Cost=400-600 (3/1000)

# Background Knowledge

**KRK-win**

a. Piece distances
b. File/rank distances
c. “Alignment” distance
d. Adjacency patterns
e. “Between” patterns
f. Distance to closest edge
g. Distance to closest corner
h. Inter-piece patterns (kings in opposition, kings almost in opposition, L-shape)

**JobShop**

a. Schedule job J “early” on M
b. Schedule job J “late” on M
c. Job J has the fastest task for M
d. Job J has the slowest task for M
e. Job J has a fast task for M
f. Job J has a slow task for M
g. Total wait time
h. Time before executing a task on M

# Results

## KRK-win

| Threshold | Random Sampling (A) | With ILP Theory (B) | Ratio B:A |
|---|---|---|---|
| Depth ≤ 8 | 0.135 | 0.721 | 5.3 |
| Depth ≤ 4 | 0.022 | 0.555 | 25.2 |
| Depth = 0 | 0.001 | 0.25 | 250.0 |

| Threshold | Optimal Solutions (Random) | Optimal Solutions (ILP) |
|---|---|---|
| Depth ≤ 8 | 1 | 21 |
| Depth ≤ 4 | 2 | 27 |
| Depth = 0 | 3 | 27 |

## JobShop

| Threshold | Random Sampling (A) | With ILP Theory (B) | Ratio B:A |
|---|---|---|---|
| Cost ≤ 1000 | 0.507 | 0.647 | 1.3 |
| Cost ≤ 750 | 0.025 | 0.171 | 6.8 |
| Cost ≤ 600 | 0.003 | 0.08 | 26.7 |

| Threshold | Optimal Solutions (Random) | Optimal Solutions (ILP) |
|---|---|---|
| Cost ≤ 1000 | 3 | 6 |
| Cost ≤ 750 | 6 | 28 |
| Cost ≤ 600 | 9 | 36 |

# Some Questions

## Why are the results better on KRK-win?

- Background used for KRK-win is much more relevant for low-cost solutions than that for JobShop
- Results for KRK-win with background that is not relevant to low-cost solutions are not as good.
- CONJECTURE: Results on JobShop will improve with better background knowledge

## How can we sample from ILP-constructed models?

- Rejection sampling (generate solution, check if covered by model)
- Generative background knowledge + SLD resolution
- Importance sampling/MCMC  (on-going with J. Cussens)

## What if we had no instances of optimal solutions in the training data?

- Results are still good (new results, not in the paper)

## What next?

- Railway scheduling and Port loading