

University of Luxembourg

Team meeting 2017

Emmanuel Kieffer



Bi-level Optimization

- Emmanuel Kieffer
- Phd candidate since February 2015
- Phd topic “Bi-level optimization”
- Background:
 - DUT in “Statistiques et Informatique décisionnelle”
 - Bachelor in Computer Sciences
 - Master in Computer Sciences: “Optimisation et algorithmique”
- Research interests:
 - Evolutionary Computing
 - Machine Learning
 - Matheuristics

Bi-level Optimization

- Bi-level problems generalize Stackelberg Games
- Introduced as “mathematical programs with optimization problems in the constraints” by Bracken
- Possible constraints at both levels

$$\begin{aligned} \min \quad & F(x, y) \\ \text{s.t.} \quad & G(x, y) \leq 0 \\ & \min \quad f(x, y) \\ & \text{s.t.} \quad g(x, y) \leq 0 \\ & x, y \geq 0 \end{aligned}$$

where $F, f : \mathbf{R}^n \times \mathbf{R}^m \rightarrow \mathbf{R}, G : \mathbf{R}^n \times \mathbf{R}^m \rightarrow \mathbf{R}^p$ and $\mathbf{R}^n \times \mathbf{R}^m \rightarrow \mathbf{R}^q$.

- NP-hard even for linear levels

From highly constrained problems to Bi-level problems

- Study Highly constrained optimization problems
- Propose novel approaches

- 3 conf. papers (GECCO16, META16, IPDPS17)
- 1 conf. paper under review (SWEVO)
- Contrib: new co-evolution mechanism

- Model real applications

- 2 journal papers under review (4OR, BMC bioinfo)
- 3 conf. papers (IAIT15, AUSSOIS16, SSCI16)
- Contrib: Bi-level Clustering, Bi-level Cloud Pricing

- Propose new approaches to tackle Bi-level optimization problems

- 1 conf. papers (GECCO17)
- 1 journal under review
- Contrib: Bayesian optimization, GP hyper-heuristics

- Tackle modelled applications

- Writing journal paper on Bi-level Cloud Pricing

Timeline



2015

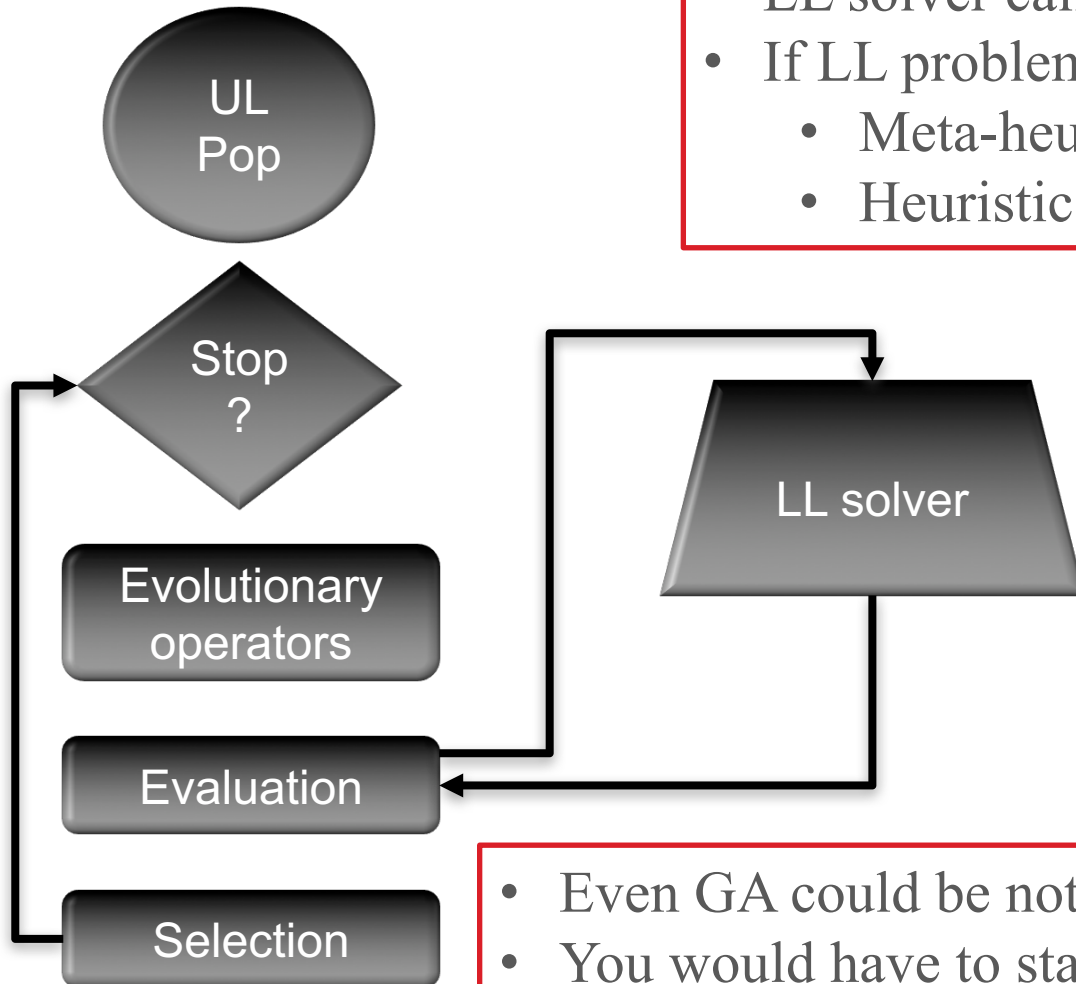
2017

Bi-level resolution

- Two ways to approach Bi-level resolution:
 - Avoiding to compute lower-level solution/fitness → surrogate approaches
 - Learning how to solve faster the lower-level
- Both involve Machine Learning
- Accepted paper to GECCO 2017 in Berlin (SAEOPT):
 - “Bayesian optimization approach of general bi-level problems”
 - Surrogate techniques very efficient for continuous Bi-level
 - We challenge one of the best existing algorithms → BLEAQ
- Now on Discrete/Combinatorial problems → Cloud Pricing

Challenges: Bi-level GA

- LL solver calls = UL fitness evaluations
- If LL problem is NP-hard:
 - Meta-heuristics
 - Heuristics



WARNING

- Even GA could be not suitable as LL solver
- You would have to start as many evolutions as UL fitness evaluations

Heuristics as LL solvers

- We would like fast solvers → heuristics
- Best, we can do is $O(n \log n)$ for greedy heuristics
- Generally, greedy heuristics can give poor results
- Can we design better heuristics ? Keep good time processing, increase accuracy
- Can we do it automatically ? Learn and build heuristics
- Idea for Bi-level Discrete Problem:
 - Identify LL problems
 - Generate instances by variation of the upper-level variables
 - Learn and generate heuristics (greedy to keep $O(n \log n)$)
 - Start Bi-level GA with heuristics as LL solvers

GP hyper-heuristics

- In the past, hyper-heuristics → selection of relevant heuristics during the search
- Limitations for new designed problems ? Maybe no existing heuristics
- New hyper-heuristics based on heuristic generation
- Heuristics are generated from scratch using machine learning
- GP hyper-heuristics based on Genetic Programming
- Learn to create the heuristics using the evolution of syntax trees

Ref : “A Genetic Programming Hyper-Heuristic for the Multidimensional Knapsack Problem” by Drake et al. in 2014

GP hyper-heuristics

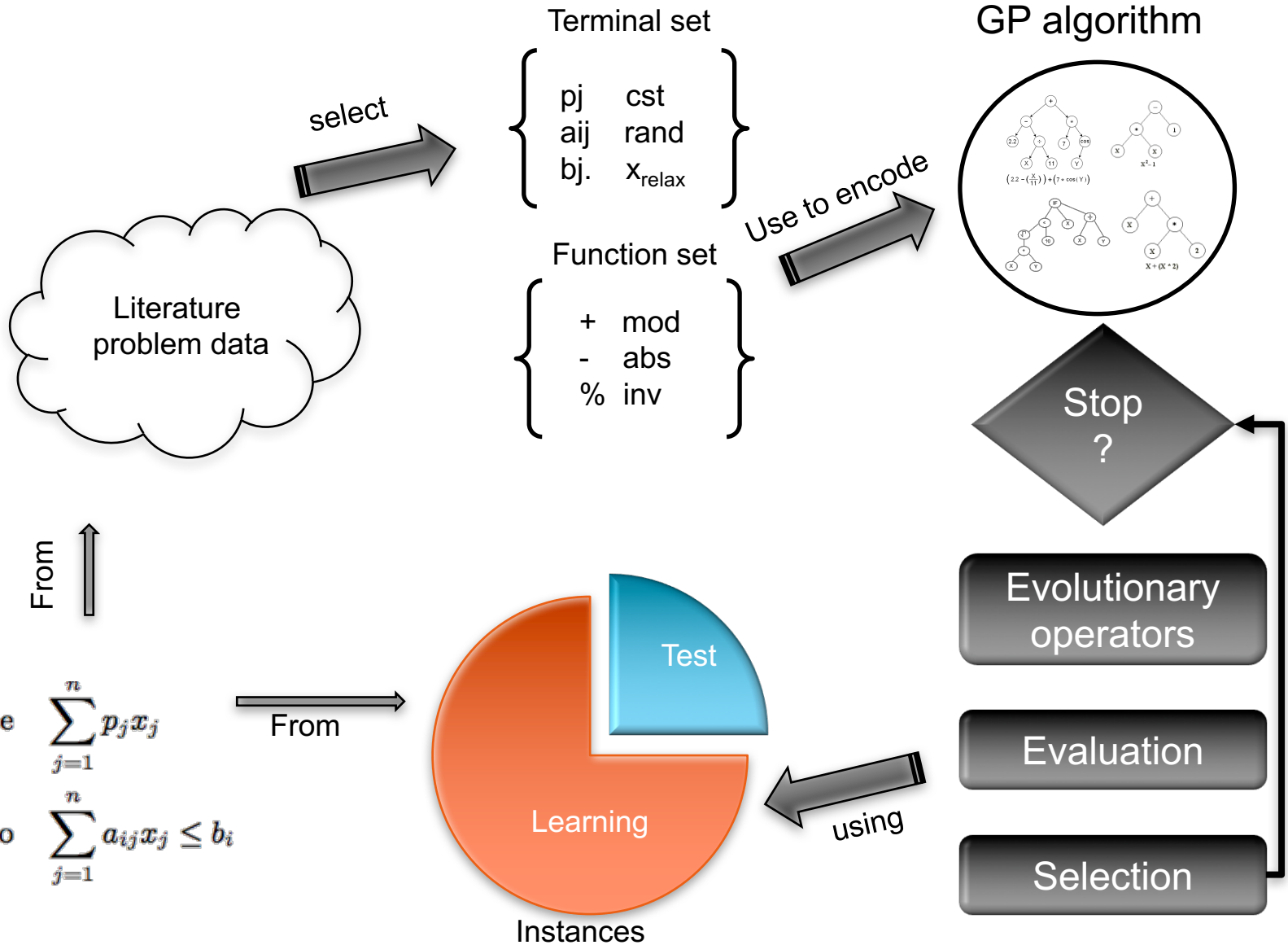
- Work on the space of heuristics rather than space of solutions
- Try to counter-balance the “curse of dimensionality”
- Learn full heuristics or some parts
 - Ex: learn the scoring function used for sorting items
 - The remain part is static \Leftrightarrow template of heuristic (see below)

Algorithm 3 greedy_heuristic(instance,function)

```
1: value  $\leftarrow$  0
2: solution  $\leftarrow$  [0,0,...,0]
3: sacks  $\leftarrow$  [0,0,...,0]
4: indexes  $\leftarrow$  sort(items,func)
5: while indexes  $\neq \emptyset$  do
6:   index  $\leftarrow$  indexes.pop_head()
7:   if sacksi + Ai,index  $\leq$  rhsi  $\forall i \in \{1, \dots, m\}$  then
8:     solutionindex  $\leftarrow$  1
9:     value  $\leftarrow$  value + pindex
10:    for i  $\in \{1, \dots, m\}$  do
11:      sacksi  $\leftarrow$  sacksi + Ai,index
12:    end for
13:  end if
14: end while
15: return value
```

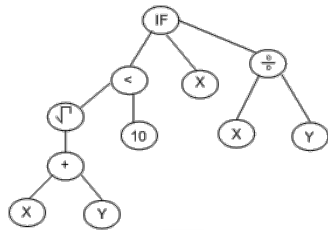
Ex: Multi-dimensional knapsack

Work-flow



Evaluation and validation

Evaluation

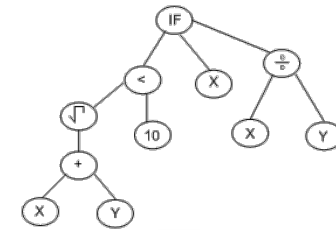


solves

Learning

$$fitness(H) = \sum_{I \in L} H(I)$$

Validation



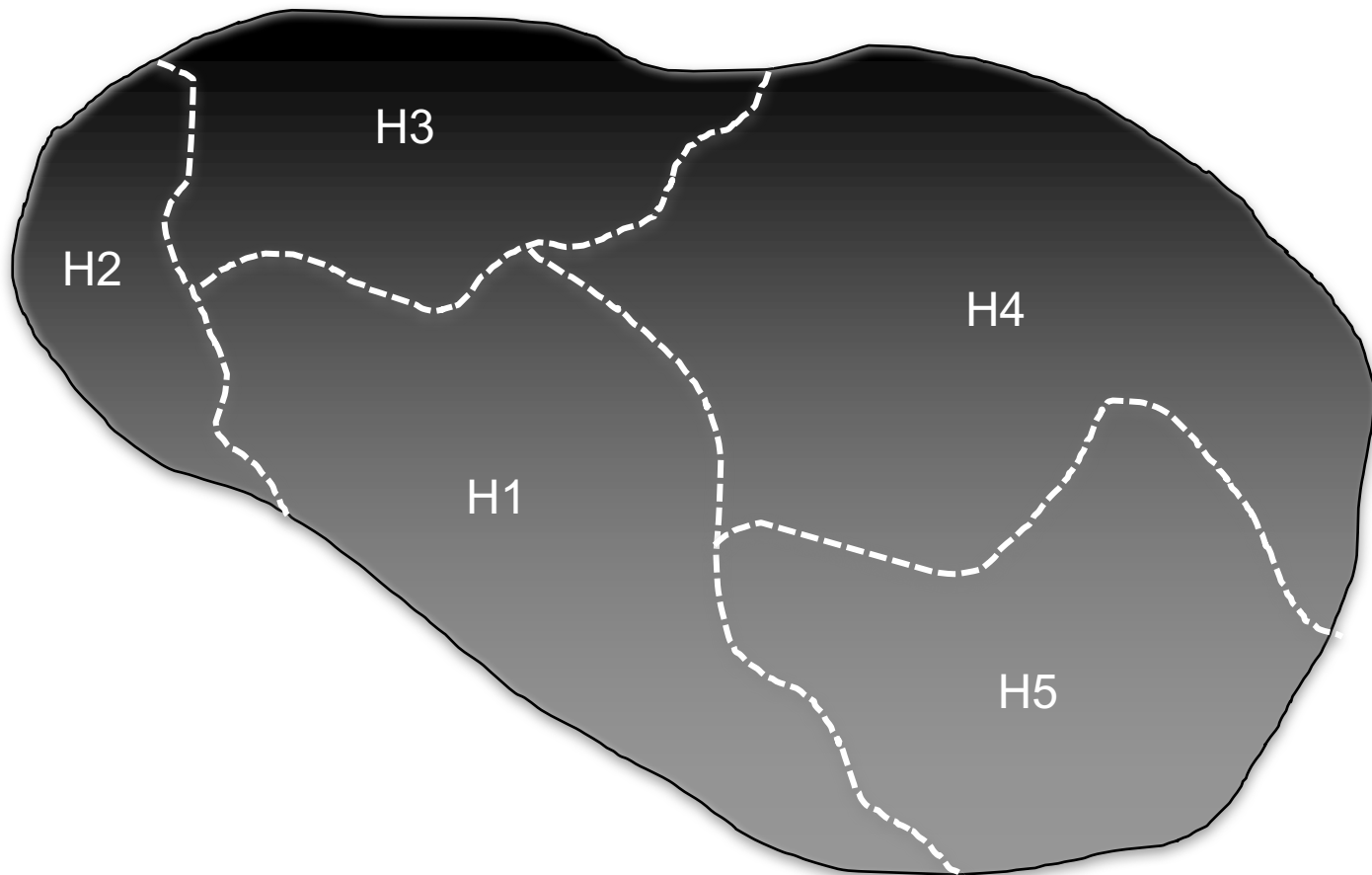
solves

Test

$$\% - gap(H, I) = 100 * \frac{\bar{I}_{lp} - H(I)}{\bar{I}_{lp}}$$

Future works

- Search space cartography using unsupervised learning
- Provide a map of automatically generated heuristics for a given problem



Thank you

Questions ?