Objective landscapes in CP Optimizer

! Ongoing work !

Philippe Laborie
IBM Analytics

October 3, 2016

Overview

- Introduction and notations
- Objective landscape definition
- Objective landscape properties
- Objective landscape computation
- Objective landscape exploitation
- Preliminary results

Introduction

Objective

Improve performance of CP Optimizer by exploiting some information on the relation between decision variables and objective value

Introduction

Given a current solution S in the LNS, we would like to **estimate**:

- How much a variable x_i contributes to the cost
- The impact on the cost to modify the value of x_i by ϵ (reduced cost)
- The ideal values (with respect to the cost) in the current domain of x_i

This is useful to:

- Decide which fragment to relax
- Guide the completion by :
  - Enforcing improvement/non-degradation of the cost for some variables
  - Selecting next variable to fix
  - Fixing variables to values close to their ideal ones

Introduction

- We need to **automatically** compute how decision variables are related with the objective function
- It must work for **continuous** variables with large domains (typically start/end of interval variables)
- This computation has to be **generic** (work for any objective function)
- This computation has to be **cheap**

$\leadsto$ **Objective landscape**

Introduction

Comparison with **impacts** (on cost):

- **Impacts** are computed by changing the domain of a variable x_i and measuring how it impacts the cost function
- **Objective landscapes** work the other way round: the cost function is changed and it measures the impact on the domain of variables x_i

Introduction

Comparison with **impacts** (on cost):

- **Impacts** are mostly designed for discrete variables with reasonably small domains
- **Objective landscapes** are mostly designed for continuous domains; their complexity is not affected by domain size

Introduction

Comparison with **impacts** (on cost):

- **Impacts** are continuously updated during the search and make a kind of *average* assumption about the value of other variables
- **Objective landscapes** are computed once and for all at the beginning of the search and make a kind of *optimistic* assumption about the value of other variables

Introduction

Comparison with **temporal linear relaxation** :

- **Temporal linear relaxation**
  - Does not give the contribution of an individual variable to the cost
  - Does not give reduced costs for all variables
- **Objective landscapes** do (in a certain sense)

Introduction

Comparison with **temporal linear relaxation** :

- **Temporal linear relaxation** heavily exploit the constraints of the problem (their linearization) and is computed at each LNS move
- **Objective landscapes** focus on the cost function and are computed once and for all at the beginning of the search

Introduction

Comparison with **temporal linear relaxation** :

- **Temporal linear relaxation** need to convexify the objective terms
- **Objective landscapes** make less assumption about convexity of objective terms and is more related to the notion of quasiconvexity (see later)

Introduction

Comparison with **temporal linear relaxation** :

- **Temporal linear relaxation** needs building and solving an LP
- **Objective landscapes** are much lighter to build

Introduction: notations

We suppose an optimization problem:

$$\begin{aligned} \text{minimize} \quad & z = f(x) \\ \text{subject to} \quad & c(x, y) \end{aligned}$$

Variables $x = [x_1, ..., x_n], y = [y_1, ..., y_m]$ are **decision variables** in the general sense, that is they can represent:

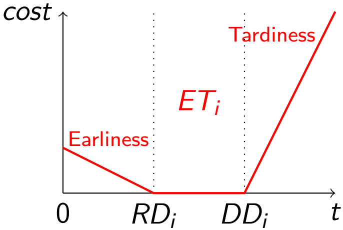
- Actual integer variables (`IloIntVar`) of the model
- Attributes of an interval variable (presence, start, end, size, length)
- Contribution of interval variables to cumul functions (`IloHeightAtXXX`)

The objective function z functionally depends on a set of **objective decision variables** x

Introduction: example

Flow-shop scheduling problem with earliness/tardiness cost:

$$\begin{aligned} \textbf{minimize} \quad & z = \sum_{i=1}^{n} \mathrm{endEval}(o_{i,m}, ET_i) \\ \textbf{subject to} \quad & \forall i \in [1,n], \forall j \in [1, m-1], \mathrm{endBeforeStart}(o_{i,j}, o_{i,j+1}) \\ & \forall j \in [1,m], \mathrm{noOverlap}(\{o_{i,j}\}_{i \in [1,n]}) \end{aligned}$$



Objective decision variables are the end values of $o_{i,m}$

Introduction: assumptions

- Focus on objective variables **without holes** in the domain:
  - Interval variables attributes (start, end, size, length, presence, heights)
  - Boolean variables
  - Integer variables with a “continuous” semantics

  $\leadsto$ Domain of a variable is a range $[L, U]$
- Single objective function (no multi-objective for now)

Landscape definition: notations

Let x_i be a decision variable and S a feasible solution
We denote:

- D_i the initial domain of variable x_i
- $D = D_1 \times ... \times D_n$
- X_i^L a lower bound on the value of variable x_i
- X_i^U an upper bound on the value of variable x_i
- X_i^S the value of variable x_i in solution S
- Z^L a lower bound on the objective function
- Z^U an upper bound on the objective function

Landscape definition: notations

Let x_i be an objective variable, function $f_i^* : D_i \to \mathbb{R}$ denotes the optimal objective value one can obtain when $x_i = v$. That is:

$$f_i^*(v) = \min_{x \in D \text{ s.t. } x_i = v} f(x)$$

Idea

The idea of objective landscapes is to build, for each decision variable x_i, a function that approximates f_i^*

Landscape definition: notations

We suppose the existence of a given propagation algorithm that is able to propagate constraints like $f(x) \leq z$ in order to reduce the domain of variables x_i

Landscape definition: notations

We suppose the existence of a given propagation algorithm that is able to propagate constraints like $f(x) \leq z$ in order to reduce the domain of variables x_i

Let x_i be an objective variable, we define:

- Z^L the smallest value of $z \in \mathbb{R}$ such that the propagation of objective cut $f(x) \leq z$ does not fail
- $X_i^L(z)$ for $z \geq Z^L$ as the lower bound on variable x_i obtained after the propagation of an objective cut $f(x) \leq z$
- $X_i^U(z)$ for $z \geq Z^L$ as the upper bound on variable x_i obtained after the propagation of an objective cut $f(x) \leq z$

Landscape definition: notations

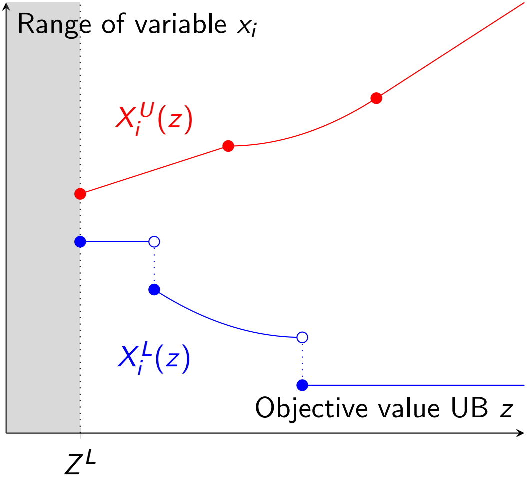
We suppose the existence of a given propagation algorithm that is able to propagate constraints like $f(x) \leq z$ in order to reduce the domain of variables x_i

Let x_i be an objective variable, we define:

- Z^L the smallest value of $z \in \mathbb{R}$ such that the propagation of objective cut $f(x) \leq z$ does not fail
- $X_i^L(z)$ for $z \geq Z^L$ as the lower bound on variable x_i obtained after the propagation of an objective cut $f(x) \leq z$
- $X_i^U(z)$ for $z \geq Z^L$ as the upper bound on variable x_i obtained after the propagation of an objective cut $f(x) \leq z$

Clearly, by monotonicity of the propagation, $X_i^L(z)$ (resp. $X_i^U(z)$) is a non-increasing (resp. non-decreasing) function of z and $X_i^L(z) \leq X_i^U(z)$

Landscape definition

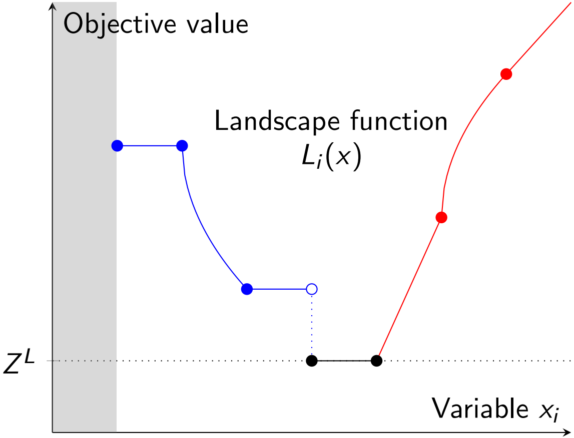


Landscape definition

Objective landscape (informal definition)

The **objective landscape function** of an objective variable x_i is a function L_i whose graph is the 90^o rotate of the union of the graphs of the two functions $X_i^L(z)$ and $X_i^U(z)$.

Landscape definition



Landscape definition

Objective landscape (formal definition)

The **objective landscape function** of an objective variable x_i is a function $L_i : D_i \rightarrow [Z^L, +\infty)$ defined as follow:

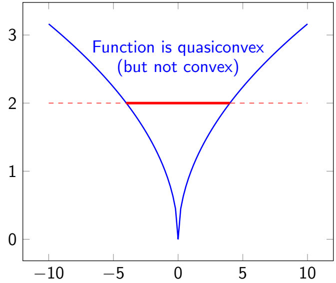
- For $v \in [X_i^L(Z^L), X_i^U(Z^L)] : L_i(v) = Z^L$
- For $v < X_i^L(Z^L) : L_i(v) = \min\{z \mid X_i^L(z) \leq v\}$
- For $v > X_i^U(Z^L) : L_i(v) = \min\{z \mid X_i^U(z) \geq v\}$

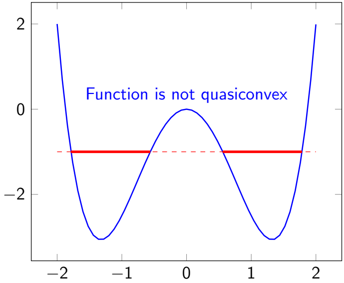
Landscape properties: quasiconvex functions

Definition

A function $f : S \to \mathbb{R}$ defined on a convex subset S of a real vector space is **quasiconvex** if for all $x, y \in S$ and $\lambda \in [0, 1]$ we have
$f(\lambda x + (1 - \lambda)y) \leq \max\left\{f(x), f(y)\right\}$

Informally, along any stretch of the curve the highest point is one of the endpoints. Example of one-variable functions:





Landscape properties

Property 1 (Quasiconvexity of landscape functions)

For any objective variable x_i, L_i is quasiconvex

Proof is a direct consequence of the fact $X_i^L(z)$ (resp. $X_i^U(z)$) is a non-increasing (resp. non-decreasing) function of z and $X_i^L(z) \leq X_i^U(z)$ □

Landscape properties

Property 2 (Landscape functions as lower bounds)

For any objective variable x_i, $L_i \leq f_i^*$

Proof is a direct consequence of the soundness of propagation $\square$

Landscape properties

Definition (Exact landscape function)

A landscape function L_i for an objective variable x_i is said to be **exact** if and only if $L_i = f_i^*$

Landscape properties: a particular (but common) case

Assumption

Let's suppose:

1. For all objective variable x_i, function f_i^* is quasiconvex
2. Propagation of $f(x) \leq z$ performs bound-consistency on the x_i's

Landscape properties: a particular (but common) case

Assumption

Let's suppose:

1. For all objective variable x_i, function f_i^* is quasiconvex
2. Propagation of $f(x) \leq z$ performs bound-consistency on the x_i's

This is for instance the case if:

- f is the sum of individual terms: $f(x_1, ..., x_n) = \sum_i f_i(x_i)$ and
- Each function f_i is quasiconvex

Landscape properties: a particular (but common) case

Assumption

Let's suppose:

1. For all objective variable x_i, function f_i^* is quasiconvex
2. Propagation of $f(x) \leq z$ performs bound-consistency on the x_i's

This is for instance the case if:

- f is the sum of individual terms: $f(x_1, ..., x_n) = \sum_i f_i(x_i)$ and
- Each function f_i is quasiconvex

It also hold more generally for any combination of sum, min, max, product (of non-negative terms) of quasiconvex individual terms $f_i(x_i)$, as far as x_i appears only once in the expression, for instance:
$x^2\sqrt{y} + \max(u^3, \min(v, w))$

Landscape properties: a particular but common case

Property 3

If function f_i^* is quasiconvex and if the propagation of $f(x) \leq z$ performs bound-consistency on x_i's then L_i is exact ($L_i = f_i^*$)

Proof is not very complex. By property 2, we know $L_i \leq f_i^*$. The proof that $f_i^* \leq L_i$ exploits bound-consistency (and quasiconvexity of f_i^* where bounding functions $X_i^L(z)$ and $X_i^U(z)$ are discontinuous). The formal proof is available in a separate document □

Landscape properties

Theorem (Characterization of exact landcapes)

A landscape function L_i is exact if and only if function f_i^* is quasiconvex and the propagation of $f(x) \leq z$ performs bound-consistency on x_i's

Proof is easy by combining properties 1 and 3 and the fact that if propagation does not perform bound-consistency, one can easily exhibit cases where for a given v, $L_i(v) > f_i^*(v)$ □

Landscape computation

Given an upper-bound Z^U on the objective (e.g. from an existing initial solution) and a lower bound Z^L (e.g. from root propagation), landscape functions can be computed by discretizing the range of possible objective values $[Z^L, Z^U]$ as a discrete set $\{z_1 = Z^L, z_2, ..., z_j, ..., z_{m-1}, z_m = Z^U\}$ and storing the min/max bounds of each decision variables x_i for each z_j

Basic algorithm for landscape computation

for j in m..1 **do**
  post($f(x) \leq z_j$)
  for i in 1..n **do**
    $X_i^L(z_j) \leftarrow$ current lower bound of x_i
    $X_i^U(z_j) \leftarrow$ current upper bound of x_i

Landscape computation

Questions:

1. Beside objective expression, should we also propagate constraints ?
2. How to best discretize objective values ?
3. How to interpolate landscapes ?

Landscape computation

Question: should we also propagate constraints ?

Pros:

- It is a way to (modestly) handle constraints in the landscapes
- The algorithm to compute landscape can work directly on the complete problem as extracted in the engine
- The landscapes can easily be extended so that they are also computed for **non-objective** decision variables, we just need to record the bounds for these variables too

Landscape computation

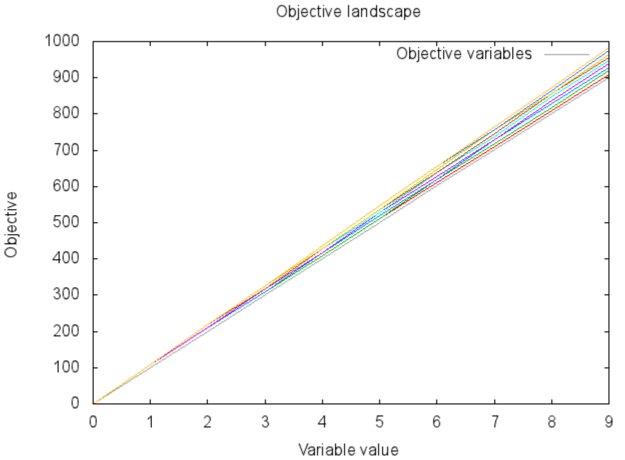
Question: should we also propagate constraints ?

Cons:

- It introduces dependencies between the objective variables and messes-up the theoretical framework about exactness of landscapes
- More importantly: it hides some important information

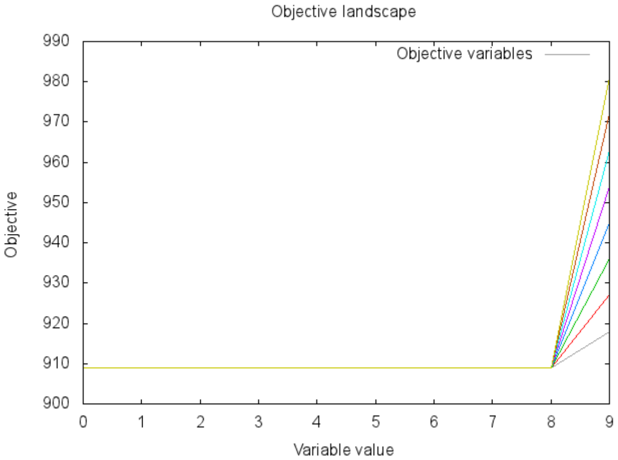
Landscape computation

Example: $i \in [0, 10), x_i \in [0, 10)$: minimize $\sum_i (100 + i) x_i$ subject to allDiff$\{x_i\}$



Landscape computation

Example: $i \in [0, 10), x_i \in [0, 10)$: minimize $\sum_i (100 + i)x_i$ subject to allDiff$\{x_i\}$



Landscape computation

Question: How to best discretize objective values ?

Ideally, we would like to select objective values $\{z_1 = Z^L, z_2, ..., z_j, ..., z_{m-1}, z_m = Z^U\}$ that result in a good sampling of the typical values of x_i met in the solutions (X_i^S) as the landscape function will be accessed at these values

Note:

- The basic algorithm for landscape computation can of course be adapted to select z_i on the fly given the variable bounds computed with the previous $z_j, j < i$
- We could even think of collecting the typical values of variables in solutions and then have a step that refines the landscape function around those values

Landscape computation

Question: How to interpolate landscapes ?

- Interpolation with lower bounding step functions preserves the lower bounding property of landscapes function
- Linear interpolation may be fine too

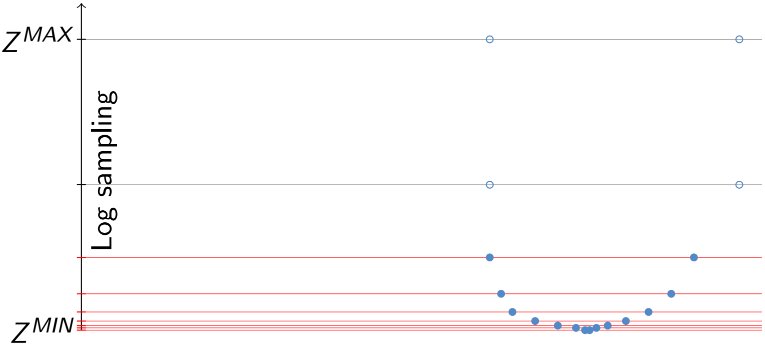
Landscape computation

Prototype implementation:

- Landscapes are computed at the end of presolve, in a transformer that converts only the objective expression (plus some functional constraints like: $z - x_1 - ... - x_n == K$)
- Let Z^{MIN} and Z^{MAX} denote the objective range (in particular after propagation-based transformer)

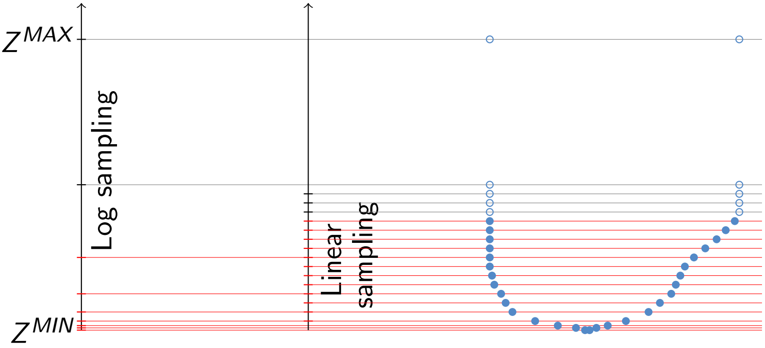
Landscape computation

Prototypical implementation:



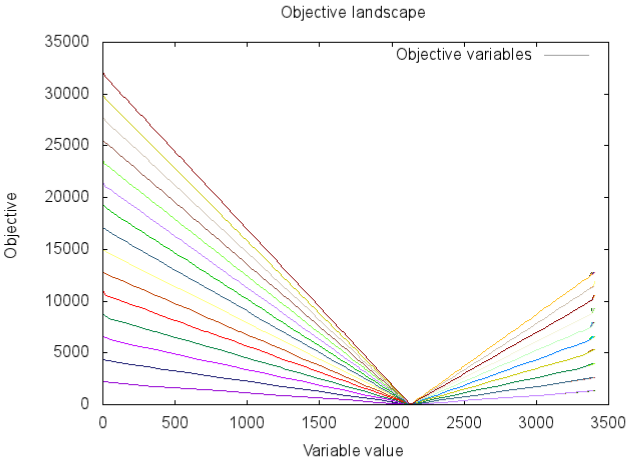
Landscape computation

Prototypical implementation:



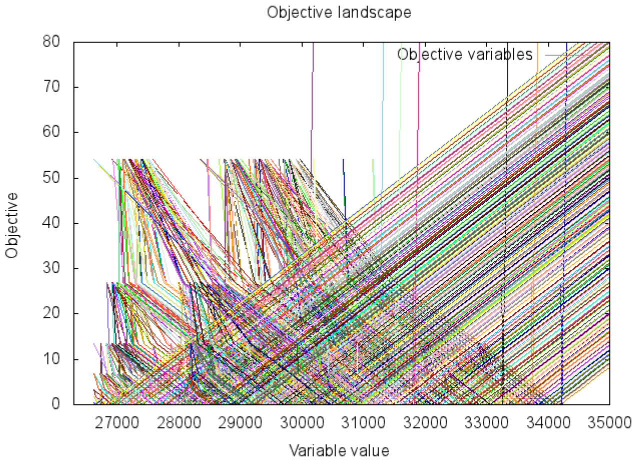
Landscape computation

Example: an instance of the CommonDueDate family



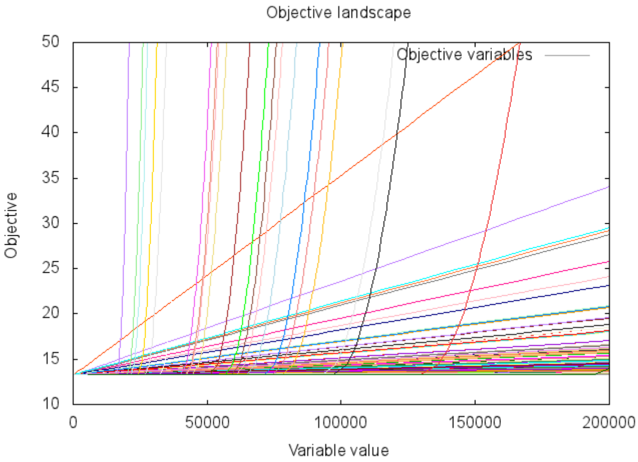
Landscape computation

Example: all 1816 landscape functions for FLETC instance

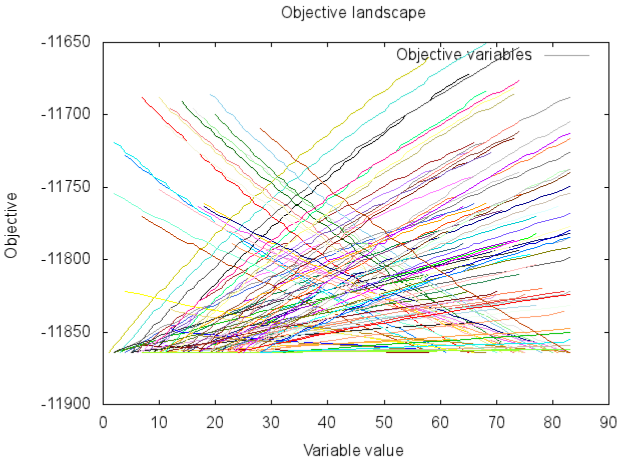


Landscape computation

Example: all 601 landscape functions for an ST Micro instance



Example: RCPSP with discounted cash flow



Landscape exploitation

1. Preventing cost degradation for some variables in LNS move
2. Defining new types of neighborhoods
3. Selecting variables and values in completion goals
4. Integration in BOLIDE meta-heuristics
5. ... any other idea welcomed!

Landscape exploitation: Preventing cost degradation

At the root node of an LNS move, after restoration of the LNS fragment, for each objective variable x_i, we know:

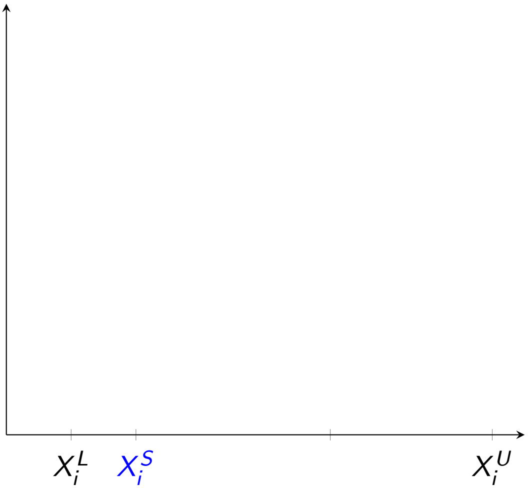
- X_i^S: the value of x_i in the incumbent solution
- X_i^U: the current upper-bound on x_i
- X_i^L: the current lower-bound on x_i
- L_i: the landscape function of x_i
- F_i: whether or not x_i belongs to the LNS fragment

Landscape exploitation: Preventing cost degradation

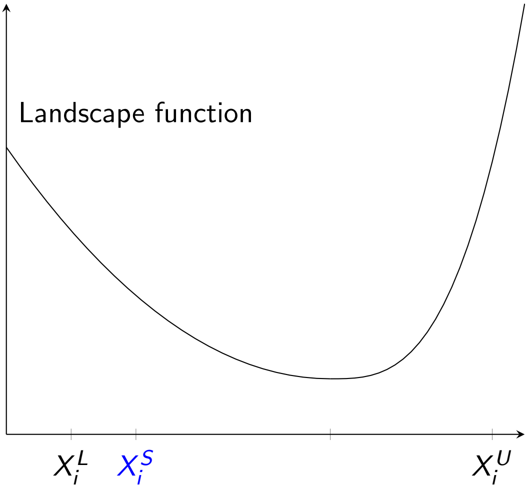
Very simple idea. At a given LNS move:

- Use or not the strategy below depending on a Boolean learner
- If strategy is used, select a certain ratio of objective variables (learned, currently possible values are: 5%, 10%, 20%, 50%, 75%, 100%), according to some **criterion**, on which to enforce non-degradation
- Non-degradation constraint: for selected variables x_i, restrict their domain to the range of values v such that $L_i(v) \leq L_i(X_i^S)$

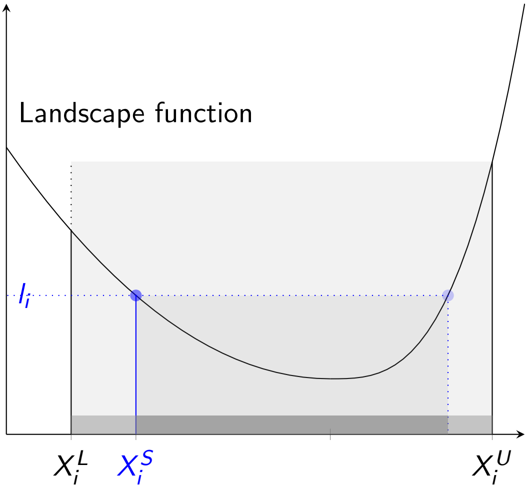
Landscape exploitation: Preventing cost degradation



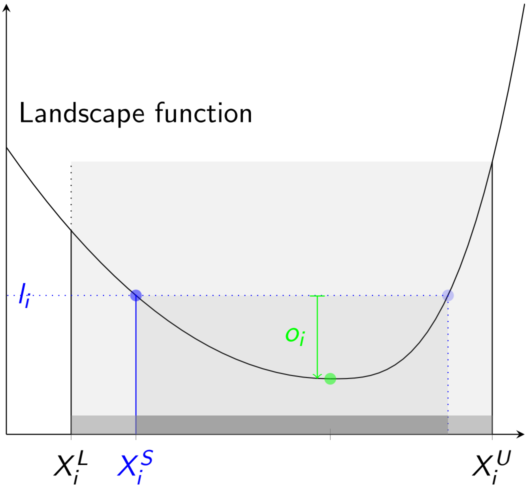
Landscape exploitation: Preventing cost degradation



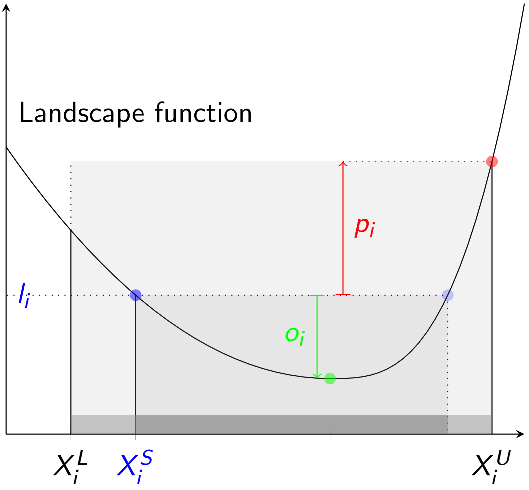
Landscape exploitation: Preventing cost degradation



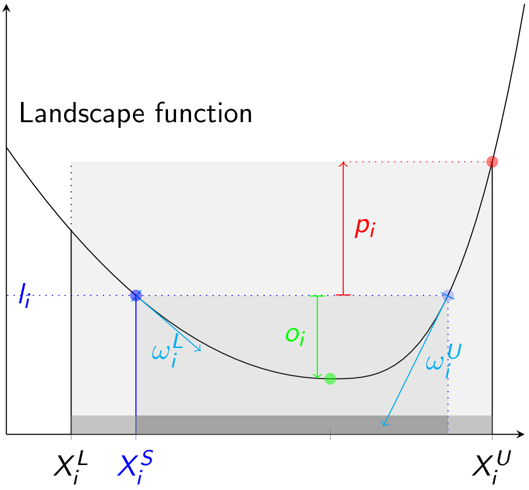
Landscape exploitation: Preventing cost degradation



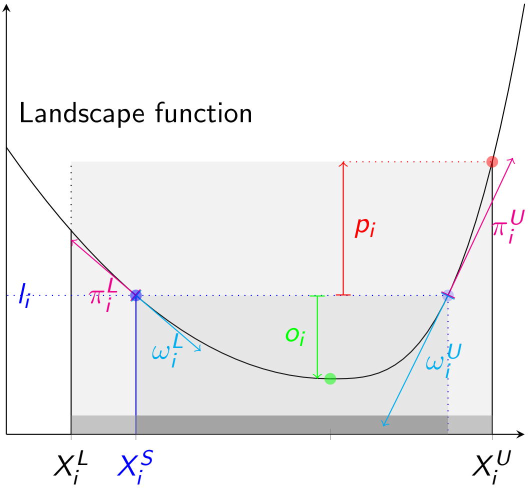
Landscape exploitation: Preventing cost degradation



Landscape exploitation: Preventing cost degradation



Landscape exploitation: Preventing cost degradation



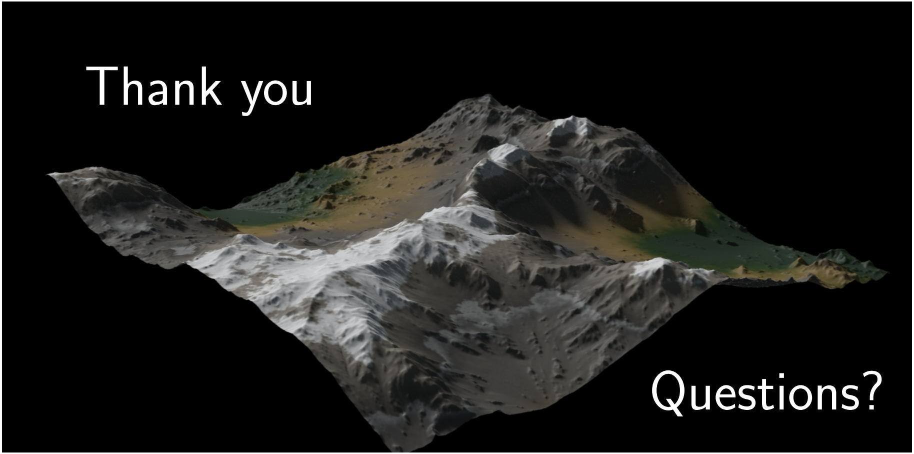
Preliminary results

Comparison using the same code (on Head, similar to 12.7) with 1 worker with/without the usage of landscapes. Average over all scheduling tests (3146 instances, 129 families). Average speed-up ratio uses *family average*.

Criterion (argmax)	Speedup
1 l_i	1.24
2 $\max(o_i, p_i)$	1.22
3 $\lvert\pi_i + \omega_i\rvert$	1.27
4 random	**1.35**
5 o_i	1.24
6 p_i	1.21

This is very early work ...

1. Preventing cost degradation in LNS moves
2. Defining new types of neighborhoods
3. Selecting variables and values in completion goals
4. Integration in BOLIDE meta-heuristics
5. ... any other idea welcomed!


Thank you
Questions?