Objective Landscapes for Constraint Programming

Philippe Laborie

IBM, 9 rue de Verdun 94250 Gentilly, France
laborie@fr.ibm.com

Abstract. This paper presents the concept of *objective landscape* in the context of Constraint Programming. An objective landscape is a light-weight structure providing some information on the relation between decision variables and objective values, that can be *quickly* computed *once and for all* at the beginning of the resolution and is used to *guide* the search. It is particularly useful on decision variables with large domains and with a continuous semantics, which is typically the case for time or resource quantity variables in scheduling problems. This concept was recently implemented in the automatic search of CP Optimizer and resulted in an average speed-up of about 50% on scheduling problems with up to almost 2 orders of magnitude for some applications.

Keywords: Constraint Programming, scheduling, search, optimization

1 Introduction

Motivations. A recognized weakness of Constraint Programming (CP) for solving combinatorial optimization problems (when compared to Mixed Integer Linear Programming for instance) is the lack of a global vision of the problem and in particular of the influence of decision variables on the objective function.

This paper presents the concept of *objective landscape* in the context of CP. The purpose of *objective landscapes* is to capture some information on the relation between decision variables and objective values. They help answering questions like: How much does a given decision variable contribute to the cost? What is the impact on the cost of modifying the value of a variable? What is the ideal value of a variable with respect to the cost function? An objective landscape is a light-weight structure that can be *quickly* computed *once and for all* at the beginning of the resolution by exploiting constraint propagation and is used to *guide* the search. It is particularly useful on decision variables with large domains and with a continuous semantics, which is typically the case for time or resource quantity variables in scheduling problems.

Although validated mostly on scheduling problems, the concept is generic and in this paper we consider a general combinatorial optimization problem defined as:

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

Variables $x = [x_1, ..., x_n]$, $y = [y_1, ..., y_m]$ are **decision variables**. The objective function f functionally depends on a subset of decision variables x that we will call **objective decision variables**.[1]

Comparison with existing approaches. The idea of measuring the interactions between variables and objective function is not new in CP. Impact-based search (Ibs) [12] incrementally maintains during the search an impact measurement for each possible variable assignment $(x_i = a)$ that estimates the impact of the assignment on the other variables domain. In Activity-based search (Abs) [8], the activity of an assignment $(x_i = a)$ estimates the number of affected variables when it is propagated. As we will see, *objective landscapes* share a common feature with the above techniques in that they are computed by exploiting constraint propagation. But there is also a number of differences:

- In Ibs, *impacts* on cost are measured by modifying the domain of a variable x_i and measuring its impact on the cost whereas *objective landscapes* work the other way round: it modifies the bounds of the cost function and measures the impact on the variables x_i.
- *Impacts* and *activity* measurements are mostly designed for discrete variables with reasonably small domains whereas *objective landscapes* were initially designed for continuous domains: their complexity does not depend on domain size.
- *Impacts* are continuously updated during the search and are based on some *averaging* assumption whereas *objective landscapes* are computed once and for all at the beginning of the search and make some kind of *optimistic* assumption about the values of variables.

Objective landscapes differ from the *cost-based solution densities* introduced in [11] in that (1) they are parameter-free (no parameter ϵ related with the optimality corridor), (2) they do not require specific (and potentially complex) computation algorithms for each constraint, (3) they can scale to large domain size with almost no computational overhead and (4) they estimate the impact of a variable's value on the cost rather than on the number of solutions.

Some approaches in CP use a linear relaxation of the problem to guide the search with a more global view on the objective. In [1], a linear relaxation (Lr) of a sub-problem using only *objective decision variables* is solved at each search node and the solution of this relaxation is used to guide the search. In [9] the Lr is computed with specific algorithms and integrated in a global constraint. The automatic search of CP Optimizer also uses a Lr of the problem at the root node of each Large Neighborhood Search move [7]. As we will see, *objective landscapes* differ from Lr in several points:

- Lr does not give the contribution of different values of a variable to the cost whereas *objective landscapes* do (in a certain sense).

[1] The distinction between objective decision variables x and the other variables y clearly depends on the formulation of the problem and may be considered as a bit arbitrary. This will be discussed in Section 7.

- LR heavily exploits the constraints of the problem (their linearization) and is usually computed several times during the search whereas *objective landscapes* focus on the cost function only and are computed once and for all.
- LR needs to *convexify* the objective function $f(x)$ whereas *objective landscapes* make less assumption about the convexity of objective terms and are related to the more general notion of *quasiconvexity*.
- LR needs building and solving a linear program or running specific algorithms whereas *objective landscapes* are much lighter to compute and exploit.

Example. All along the paper we will use a flow-shop scheduling problem with earliness-tardiness cost inspired from [10] for illustration. Using the scheduling concepts of CP Optimizer, an instance of this problem for n jobs and m machines can be formulated with a set of interval variables $o_{i,j}$ of fixed length (processing time), one for each operation, as follows:

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{i=1}^{n} \text{endEval}(o_{i,m}, ET_i) & \\ \text{subject to} & \text{endBeforeStart}(o_{i,j}, o_{i,j+1}) & \forall i \in [1,n], \forall j \in [1,m-1] \\ & \text{noOverlap}(\{o_{i,j}\}_{i \in [1,n]}) & \forall j \in [1,m] \end{array}$$

In the formulation above, ET_i is a piecewise linear function representing the earliness-tardiness cost for finishing job i at a date t as illustrated on Fig 1. In this example, *objective decision variables* are the end values of interval variables $o_{i,m}$ representing the last operation of job i.

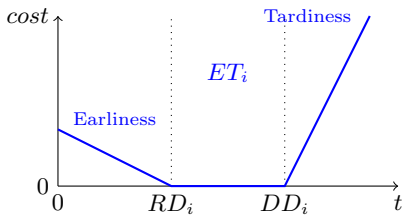

Fig. 1. Example of earliness-tardiness cost function

Paper organization. Objective landscapes are formally defined in Section 2 whereas Section 3 presents some of their properties. From a more practical point of view, Sections 4 and 5 respectively describe how landscapes can be computed and exploited to guide the search. Objective landscapes have been implemented in CP Optimizer (in version 12.7.1). Section 6 presents an experimental study on a large number of scheduling problems showing the practical interest of landscapes.

2 Objective Landscape Definition

We consider a general combinatorial optimization problem defined as:

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

With $x = [x_1, ..., x_n]$, $y = [y_1, ..., y_m]$. Let x_i be a decision variable and S a feasible solution to the problem, we denote:[2]

- 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 f
- Z^U an upper bound on the objective function f

Let x_i be an objective variable, we define function $f_i^* : D_i \to \mathbb{R}$ such that $f_i^*(v)$ denotes the optimal objective value one can obtain when $x_i = v$ if one only considers the objective function f. That is:

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

Value $f_i^*(v)$ gives, in a certain sense, the contribution of the assignment $x_i = v$ to the cost under the optimistic assumption that all the other variables $x_{j,j \neq i}$ are fixed to values minimizing their overall contribution to the cost.

The idea of objective landscapes is to build, for each decision variable x_i, a function that approximates f_i^* and that we will call the objective landscape of variable x_i.

We suppose the existence of a given propagation algorithm $\mathcal{P}$ that is able to propagate constraints like $f(x) \leq z$ in order to reduce the domain of variables x_i. It is important to note that in the context of landscapes, propagation algorithm $\mathcal{P}$ ignores the constraints of the problem $c(x, y)$. We suppose that propagation algorithm $\mathcal{P}$ is *monotonous* (if $z' < z$, the propagation of $f(x) \leq z'$ does not lead to larger domains for variables x_i than the propagation of $f(x) \leq z$) and, of course, that it is *sound* (it does not remove feasible values). Let x_i be an objective variable, we define:[3]

- 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 $f(x) \leq z$

[2] Through the paper, we use lower cases for variables and upper cases for constants.

[3] By abuse of notation, as $\mathcal{P}$ is supposed to be given, we do not use it in the notations of these bounds although they clearly depend on the propagation algorithm.

– $X_i^U(z)$ 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, for any $z \geq Z^L$, $X_i^L(z)$ (resp. $X_i^U(z)$) is a non-increasing (resp. non-decreasing) function of z and $X_i^L(z) \leq X_i^U(z)$. An example of these two functions is shown on the left part of Fig 2.

Informally speaking, 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 $X_i^U(z)$, as illustrated on the right part of Fig. 2.

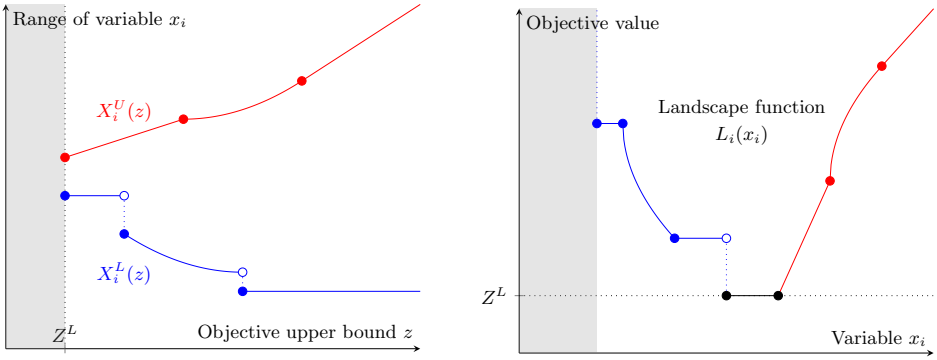

Fig. 2. Left: Decision variable ranges as a function of objective upper-bound z. Right: Objective landscape function of an objective variable x_i

Objective landscapes can now be defined more formally.

Definition 1 (Objective landscape). *Given a propagation algorithm $\mathcal{P}$, the* **objective landscape function** *of an objective variable x_i is a function $L_i : D_i \to [Z^L, +\infty)$ 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* $v > X_i^U(Z^L) : L_i(v) = \min\{z \mid X_i^U(z) \geq v\}$

3 Objective Landscape Properties

We first recap the notion of a quasiconvex function [3].

Definition 2 (Quasiconvex function). *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 $f(\lambda x + (1 - \lambda)y) \leq \max\{f(x), f(y)\}$*

Informally, along any stretch of the curve the highest point is one of the endpoints. Examples of quasi convex and non-quasiconvex functions are shown on Fig 3.

We can now present some properties of objective landscapes.

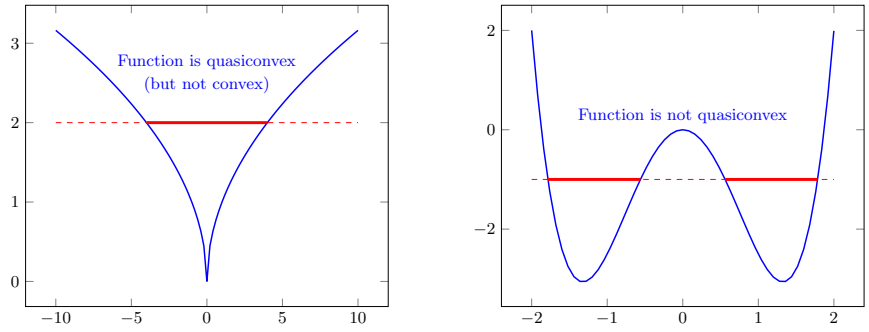

Fig. 3. Examples of quasi convex and non-quasiconvex functions

Proposition 1 (Quasiconvexity of landscape functions). *For any objective variable x_i, landscape function L_i is quasiconvex.*

Proof. 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 $X_i^L(z) \leq X_i^U(z)$. □

Proposition 2 (Landscape functions as lower bounds). *For any objective variable x_i, $L_i \leq f_i^*$.*

Proof. Proof is a direct consequence of the soundness of propagation. □

Definition 3 (Exact landscape function). *A landscape function L_i for an objective variable x_i is said to be* **exact** *if and only if $L_i = f_i^*$.*

The proposition below gives a sufficient condition for a landscape function to be exact.

Proposition 3. *If function f_i^* is quasiconvex and if the propagation of $f(x) \leq z$ performs bound-consistency on variables x then L_i is exact.*

Proof. 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). See Appendix for details. □

Proposition 3 is interesting because its condition holds in many practical cases. In particular, it holds as soon as the objective function is an aggregation $\Theta_{i=1}^n\big(f_i(x_i)\big)$ (where Θ is a *sum*, *min*, *max*, or *product of non-negative terms*) of individual quasiconvex terms $f_i(x_i)$ as it is easy for these expressions to provide a bound-consistent propagation algorithm. The flow-shop scheduling problem with earliness-tardiness cost presented in the introduction is clearly an example as soon as the earliness-tardiness cost function is quasiconvex like the one on Fig. 1. Note that on this example, as the minimal value of the cost functions ET_i is 0, the landscape function L_i of a job i is exactly its earliness-tardiness cost function ET_i.

In fact we can also show that the condition of Proposition 3 is a necessary condition leading to the following theorem that characterizes exact landscapes.

Theorem 1 (Characterization of exact landscapes). *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 variables x.*

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

It is important to keep in mind that objective landscapes are used as a heuristic to guide the search so in practice, even if the conditions of Theorem 1 are not met, the landscape functions can still convey some relevant information and be useful. This is in particular the case when the same variable x_i appears several time in the formulation of objective function f so that the propagation algorithm is not guaranteed to achieve bound consistency.

4 Objective Landscape Computation

Principles of the computation. Given a range $[Z^L, Z^U]$ for the objective function, coming for instance from the propagation at the root node, a simple way of computing landscape functions consists in discretizing the objective values by selecting m values in the domain $z_1 = Z^U > z_2 > \cdots > z_j > \cdots > z_{m-1} > z_m = Z^L$ and recording the bounds $X_i^L(z_j)$ and $X_i^U(z_j)$ for each variable x_i as shown in Algorithm 1. Note that the objective bounds z_j are traversed by decreasing value so, by propagation monotony, the domains of variables x_i will only decrease and no reversibility/backtrack is needed.

Algorithm 1 Objective landscapes basic computation

```
1: for j in 1..m do
2:     propagate(f(x) ≤ z_j)
3:     for i in 1..n do
4:         X_i^L(z_j) ← current lower bound of x_i
5:         X_i^U(z_j) ← current upper bound of x_i
```

The discretized values of the objective landscape functions computed in Algorithm 1 can be interpolated by a linear approximation or by a lower bounding step function that would preserve the lower bound property of Proposition 2.

It is of course important to select some objective values z_j that are representative of the values explored during the search. In a minimization problem, we can expect that small objective values are more useful than large ones, this is why in our implementation in CP Optimizer we decided to use a logarithmic scheme. More precisely, before the model is actually solved (see [6] for an overview of the automatic search), the landscape computation step extracts only the objective function $f(x)$ and performs m_{LOG} logarithmic steps of Algorithm 1 using $z_j = \frac{Z^U - Z^L}{2^{j-1}} + Z^L$. It records the first objective value $Z = z_j$ in the descent

such that the next one (z_{j+1}) has an impact on the bounds of some variable x_i and then performs a new run of Algorithm 1, this time with a linear sampling with m_{LIN} steps of interval $[Z^L, Z]$. This process is illustrated on Fig. 4.

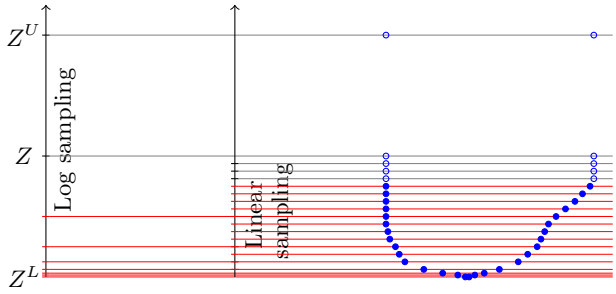

Fig. 4. Objective landscape computation

Complexity. In practice, just a few tens of points (value m in Algorithm 1) are needed to represent a landscape function L_i with sufficient precision. The landscape function structure can be as simple as an array of pairs $[v_j, L_i(v_j)]_{j\in[1..m]}$ and estimating its value $L_i(v)$ for a given v can be performed in $O(\log(m))$ with a binary search. The algorithmic cost of computing the landscapes is in $O(m(n+p))$ if n is the number of objective variables and p the cost of propagating an objective upper bound $f(x) \leq z$. This is negligible compared to the typical resolution time of the optimization problem.

Examples. Some examples of computed objective landscapes using a linear interpolation are illustrated on Fig. 5-8. On these figures, each function describes the landscape of a decision variable of the problem. Note that in these problems, objective decision variables are time variables (start/end of a time interval or temporal delay on a precedence constraint).

- On Fig. 5 the problem is a one-machine problem with V-shape earliness-tardiness costs and a common due date for all activities [2]. We see that the common due date (actual value is 2171 on this instance) as well as the different weights for the earliness-tardiness costs are effectively captured by the landscapes
- On Fig. 6 the problem is the flow-shop scheduling with earliness-tardiness cost of our example with some particular V-shape costs, we observe in particular the different due-dates of the jobs
- Fig. 7 illustrates a semiconductor manufacturing scheduling problem described in [4] whose objective function is the weighted sum of two types of cost: some weighted completion times of jobs (the linear functions) and

some delays on precedence constraints between operations that incur a large quadratic cost
– On Fig. 8 the problem is a resource-constrained project scheduling problem with maximization of net present value [13]. The landscape functions clearly distinguish between the tasks with positive cash flow (increasing landscape function) and the ones with negative one (decreasing landscape function) and reveal the exponential nature of the objective terms ($e^{-\alpha t}$)

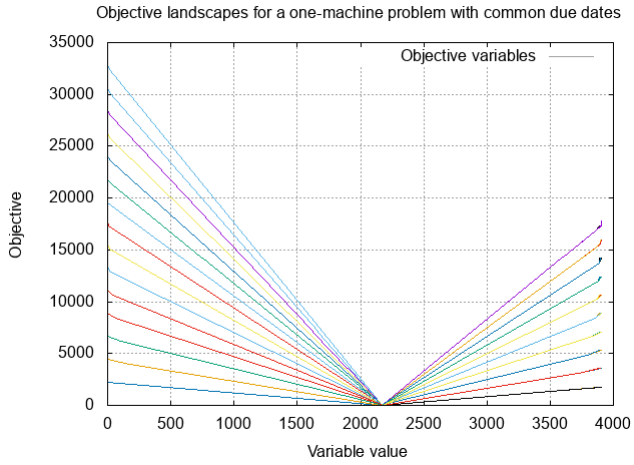

Fig. 5. Objective landscapes for a problem with common due dates [2]

5 Objective Landscape Exploitation

Once available, the objective landscape functions could be used in several ways during the search. In this article we focus on their use in the context of Large Neighborhood Search (LNS) but they could be used in variable/value ordering heuristic in a classical CP search tree, and more generally in Constrained Optimization Problems. In an LNS framework as the one used in the automatic search of CP Optimizer [6], landscapes can be used:

– To prevent cost degradation for some variables in LNS moves
– To define new types of neighborhoods
– To select variables and values in LNS completion strategies

At the root node of an LNS move, if we are trying to improve an incumbent solution S of cost Z, once the selected fragment has been relaxed and given that constraint $f(x) \leq Z$ has been propagated, there are three values of interest for a given objective variable x_i:

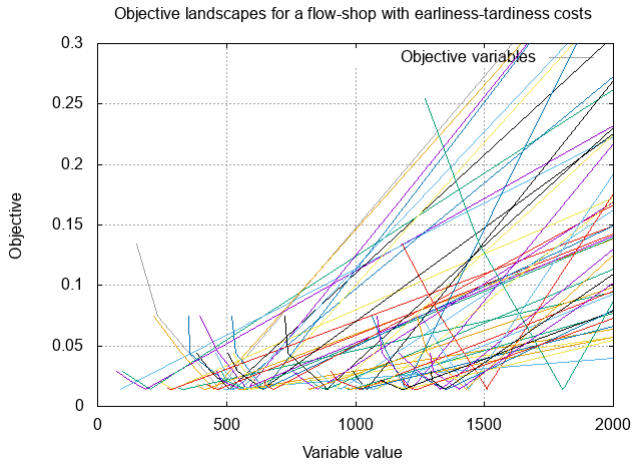

Fig. 6. Objective landscapes for a flow-shop with earliness-tardiness cost [10]

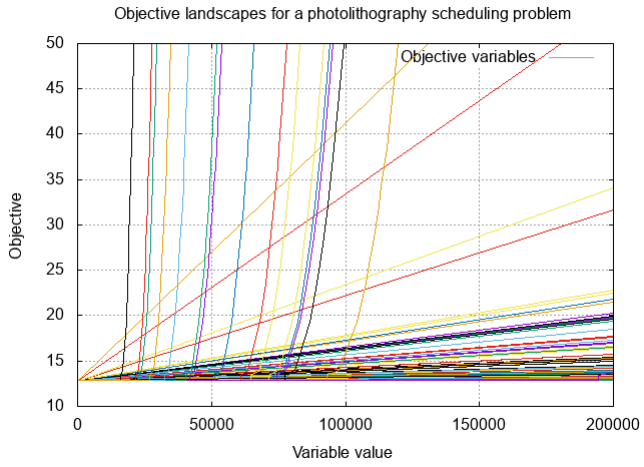

Fig. 7. Objective landscapes for a problem in semiconductor manufacturing [4]

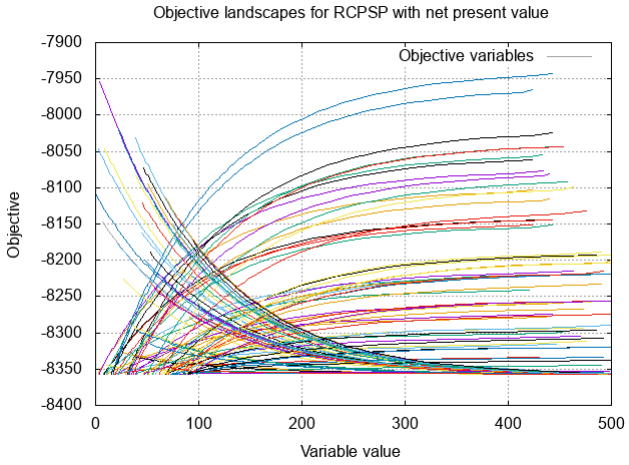

Fig. 8. Objective landscapes for an RCPSP with net present value [13]

- X_i^L the current lower bound of the variable
- X_i^U the current upper bound of the variable
- X_i^S the value of the variable in solution S

We have of course $X_i^L \leq X_i^S \leq X_i^U$ as solution S is consistent with the current bounds. Thanks to the landscape function L_i of the variable, we can compute some interesting indicators illustrated on Fig. 9:

- $l_i = L_i(X_i^S)$ is the objective landscape value of variable x_i at solution S
- As function L_i is quasiconvex, the set $\{v \in [X_i^L, X_i^U] | L_i(v) \leq l_i\}$ is a convex interval denoted $[X_i^{S-}, X_i^{S+}]$, one of its endpoint (X_i^{S-} or X_i^{S+}) being equal to X_i^S. This interval represents the values v in the current domain of x_i that do not degrade the landscape function value compared to solution S.
- $o_i = l_i - \min_{v \in [X_i^L, X_i^U]} L_i(v)$ is an optimistic bound on how much the objective landscape value of the variable could be improved by choosing the best value in the current domain.
- $p_i = \max_{v \in [X_i^L, X_i^U]} L_i(v) - l_i$ is a pessimistic bound on how much the objective landscape value of the variable could be degraded by choosing the worse value in the current domain.
- ω_i is the derivative of landscape function L_i evaluated at value X_i^S in direction of the improvement of the landscape value. It measures how much a small change from the solution value could improve the landscape function.
- π_i is the derivative of landscape function L_i evaluated at value X_i^S in direction of the degradation of the landscape value. It measures how much a small change from the solution value could degrade the landscape function.

We are far from having investigated all the above indicators and, more generally, all the potentialities of landscapes during the search. In our initial implementation in CP Optimizer 12.7.1, landscapes are exploited as follows.

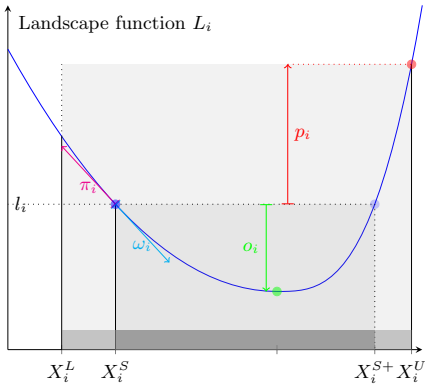

Fig. 9. Some objective landscape indicators

- At a given LNS move, the landscape strategy described below is used with a certain probability that is learned online as described in [5]
- Landscape strategy:
  - A certain number of objective variables are randomly selected. The ratio of selected variables is learned online from a set of possible ratio values in $[0, 1]$.
  - For each selected objective variable x_i, a constraint $v \in [X_i^{S-}, X_i^{S+}]$ is added to ensure that the landscape value of these variables is not degraded during the LNS iteration.

This type of strategy focuses the search on the improvement of a subset of the objective terms, it is particularly useful for objective expressions that are not well exploited by constraint propagation, typically like *sum* expressions which are extremely common in scheduling problems.

In the flow-shop scheduling example, when the landscape strategy is used at an LNS move, it will randomly select a subset of jobs and for each selected job i, post the additional constraint on the job end time that it must not degrade its earliness-tardiness cost ET_i compared to what it was in the incumbent solution.

6 Results

In this section, we compare the performance of CP Optimizer V12.7.1 (containing the implementation of landscapes as described in sections 4 and 5) with the same version that do not compute and use landscapes[4]. The comparison was performed on the IBM scheduling benchmark which, as of today, consists of 135 different families of scheduling problems collected from different sources (classical instances for famous scheduling problems like job-shop or RCPSP, benchmarks proposed in academic papers, industrial scheduling problems modeled by

[4] A specific parameter can be used to switch off objective landscapes.

our consultants, partners or end-users, problems submitted on our Optimization forums, *etc.*). The performance comparison between the two versions of the automatic search is measured in terms of resolution time *speed-up*. As the engine is not able to solve all instances to optimality in reasonable time, the time speed-up estimates how many times faster the default version of the automatic search (using landscapes) is able to reach similar quality solutions to the version without landscapes. Results are summarized on Fig. 10. The 135 different families are sorted on the x-axis by increasing speed-up factor. Each point is the average speed-up over the different instances of the family. The geometrical mean of the speed-up over the 135 families is around 1.5. On Table 1, the families are classified by objective types, depending on the aggregation function (usually, a *sum* or a *max*) and the nature of the aggregated terms. These objective types are also shown with different marks on Fig. 10. As expected, the objective landscapes have a strong added value in the case of sums of terms involving expressions with large domains (like start/end of intervals with an average speed-up of 2.62, interval lengths with a speed-up larger than 3 or resource capacities (height)). As an illustration, for the flow-shop scheduling example with earliness-tardiness cost described in the introduction, the average speed-up factor is larger than 18. Note that for some families of scheduling problems the speed-up factor is close to two orders of magnitude. The speed-up is much more modest for objectives like makespan (max of ends) or total resource allocation costs (sum of presence of intervals).

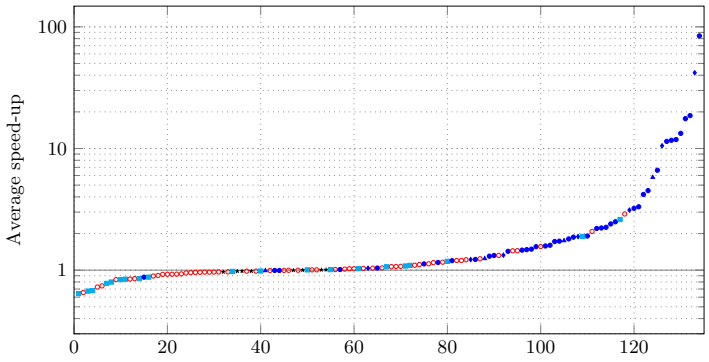

Fig. 10. Average speed-up when using objective landscapes measured on 135 different families of scheduling problems.

7 Conclusion and Discussion

This paper introduces the notion of *objective landscapes* in CP. Objective landscapes are light-weight structures that can be fast computed before the search by exploiting constraint propagation on the objective function expression. We

Table 1. Average speed-up by objective type

Aggregation type	Main variables type	Number of families	Average speed-up	Mark on Fig. 10
Max	Start/End	55	1.05	○
Sum	Start/End	40	2.62	●
Sum	Presence	29	0.98	■
Sum	Length	7	3.31	♦
Sum	Height	4	1.88	▲

show some formal properties of landscapes and give some results about their implementation in the automatic search of CP Optimizer: an average speed-up of 50% on a large panel of scheduling problems, with up to almost 2 orders of magnitude for some applications.

We have only explored a small part of the potentials of objective landscapes and there are still many interesting open questions:

- Landscapes are currently defined by only considering the objective expression $f(x)$ and the objective variables x. The definition of objective variables clearly depend on how the problem is formulated. There may be many different subsets of decision variables x such that the objective functionally depends on x. Some may be more interesting than others. There is here an evident link with the notion of *functional dependency* heavily studied in relational databases but much less in CP [14].
- In fact functional dependency is not strictly required. We could also compute some landscapes on non-objective variables by considering the constraints $c(x, y)$ of the problem. But this messes up the theoretical framework about exact landscape computations mostly because achieving bound-consistency is in general impossible on the whole problem. It also results in a phenomenon that tends to hide the lowest part of landscape functions L_i as, because of constraint propagation, the model including constraints $c(x, y)$ will start to fail for larger values of z resulting in a larger value for Z^L.
- Objective landscapes can be extended without too much difficulty to handle holes in the domains. The landscape structure would be slightly more complex, looking like a tree describing how values or intervals of values are removed from the domain of a variable x_i when the objective z is decreasing. This type of landscape could be useful for variables with a discrete semantics.
- As mentioned in Section 5, we only explored the tip of the iceberg as about how to exploit objective landscapes to guide the search.

Appendix: Proof of Proposition 3

Let $v \in D_i$, we want to prove that $L_i(v) = f_i^*(v)$. We distinguish 3 cases depending on the position of v with respect to $X_i^L(Z^L)$ and $X_i^U(Z^L)$.

Case 1: $v \in [X_i^L(Z^L), X_i^U(Z^L)]$

By definition of the landscape function we have $L_i(v) = Z^L$.

We first note that by definition of Z^L, for every objective value $z < Z^L$ as the propagation of $f(x) \leq z$ fails, it means that Z^L is a lower bound on $Z^* = \min_{x \in D} f(x)$.

As by definition of function f_i^* we have $\forall v \in D_i, Z^* \leq f_i^*(v)$. We can deduce: $L_i(v)(= Z^L) \leq f_i^*(v)$.

We will now use the bound-consistency property of propagation to show that $f_i^*(X_i^L(Z^L)) \leq Z^L$ and $f_i^*(X_i^U(Z^L)) \leq Z^L$. By definition $X_i^L(Z^L)$ is the minimal value in the domain of x_i after propagation of $f(x) \leq Z^L$. Because propagation is bound-consistency on x_i, it means that there exist some x with $x_i = X_i^L(Z^L)$ such that $f(x) \leq Z^L$ (otherwise, this value would have been filtered from the domain). Furthermore, by definition of function f_i^*, we have $f_i^*(X_i^L(Z^L)) \leq f(x)$, thus $f_i^*(X_i^L(Z^L)) \leq Z^L$. The proof that $f_i^*(X_i^U(Z^L)) \leq Z^L$ is symmetrical. Finally, the quasiconvexity of f_i^* implies that for all $v \in [X_i^L(Z^L), X_i^U(Z^L)]$, $f_i^*(v) \leq Z^L(= L_i(v))$.

Case 2: $v < X_i^L(Z^L)$

By definition of the landscape we have $L_i(v) = \min\{z | X_i^L(z) \leq v\}$.

We first prove that $f_i^*(v) \geq L_i(v)$. By contradiction, suppose that $f_i^*(v) < L_i(v)$ and let $z' = f_i^*(v)$. This would mean that there exist x with $x_i = v$ such that $f(x) = z'$. For such a z' we would have $X_i^L(z') \leq v$ because x supports objective value z' and thus value v cannot be removed from the domain of x_i. Such a value $z' < L_i(v)$ such that $X_i^L(z') \leq v$ contradict the fact $L_i(v)$ is the smallest z satisfying $X_i^L(z) \leq v$. This proves $f_i^*(v) \geq L_i(v)$.

For proving that $f_i^*(v) \leq L_i(v)$, we distinguish 2 sub-cases depending on whether or not there exist an objective value z such that $X_i^L(z) = v$.

Case 2.1: There exist a value z such that $X_i^L(z) = v$ so that $L_i(v) = z$.

By definition of X_i^L, this means that v is the minimal value in the domain of x_i after propagating $f(x) \leq z$. Because propagation is bound-consistent on x_i, it means that there exist some x with $x_i = v$ such that $f(x) \leq z$ (otherwise, this value would have been filtered from the domain). Furthermore, by definition of function f_i^*, we have $f_i^*(v) \leq f(x)$, thus $f_i^*(v) \leq z(= L_i(v))$.

Case 2.2: There does not exist any value z such that $X_i^L(z) = v$.

This is the case of a discontinuity of function $X_i^L(z)$. Let $v^+ = X_i^L(L_i(v))$ denote the value of X_i^L at the discontinuity. By definition of L_i, we have $L_i(v) = L_i(v^+)$. Furthermore, as v^+ satisfies case 2.1 above, we also know that $f_i^*(v^+) \leq L_i(v^+)$. We have seen in case 1 that $f_i^*(X_i^L(Z^L)) \leq Z^L$ and, of course, $Z^L \leq L_i(v^+)$. To summarize: $v \in [X_i^L(Z^L), v^+]$, $f_i^*(X_i^L(Z^L)) \leq L_i(v^+)$ and $f(v^+) \leq L_i(v^+)$. Thus, by quasiconvexity of f_i^*: $f(v) \leq L_i(v^+)(= L_i(v))$.

Case 3: $v > X_i^U(Z^L)$

This case is the symmetrical of Case 2. □

References

1. Beck, J.C., Refalo, P.: A hybrid approach to scheduling with earliness and tardiness costs. Annals of Operations Research 118(1-4), 49–71 (2003)
2. Biskup, D., Feldmann, M.: Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates. Computers and Operations Research 28(8), 787–801 (2001)
3. Greenberg, H., Pierskalla, W.: A Review of Quasi-Convex Functions. Operations Research 19(7), 1553–1570 (1971)
4. Knopp, S., Dauzère-Pérès, S., Yugma, C.: Modeling Maximum Time Lags in Complex Job-Shops with Batching in Semiconductor Manufacturing. In: Proc. 15th International Conference on Project Management and Scheduling (PMS 2016). pp. 227–229 (2016)
5. Laborie, P., Godard, D.: Self-adapting large neighborhood search: Application to single-mode scheduling problems. In: Baptiste, P., Kendall, G., Munier-Kordon, A., Sourd, F. (eds.) Proceedings of the 3rd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA). pp. 276–284. Paris, France (28-31 August 2007) (28-31 August 2007 2007)
6. Laborie, P., Rogerie, J., Shaw, P., Vilím, P.: IBM ILOG CP Optimizer for Scheduling. Constraints Journal (2018)
7. Laborie, P., Rogerie, J.: Temporal Linear Relaxation in IBM ILOG CP Optimizer. Journal of Scheduling 19(4), 391–400 (2016)
8. Michel, L., Van Hentenryck, P.: Activity-Based Search for Black-Box Constraint Programming Solvers. In: Proc. 9th International Conference on Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems (CPAIOR 2012). pp. 228–243 (2012)
9. Monette, J., Deville, Y., Hentenryck, P.V.: Just-in-time scheduling with constraint programming. In: Proc. 19th International Conference on Automated Planning and Scheduling (ICAPS 2009) (2009)
10. Morton, T., Pentico, D.: Heuristic Scheduling Systems. Wiley (1993)
11. Pesant, G.: Counting-Based Search for Constraint Optimization Problems. In: Proc. 13th AAAI Conference on Artificial Intelligence (AAAI 2016). pp. 3441–3447 (2016)
12. Refalo, P.: Impact-based search strategies for constraint programming. In: Proc. 10th International Conference on Principles and Practice of Constraint Programming (CP 2004). pp. 557–571 (2004)
13. Vanhoucke, M.: A scatter search heuristic for maximising the net present value of a resource-constrained project with fixed activity cash flows. International Journal of Production Research 48, 1983 – 2001 (2010)
14. Vardi, M.Y.: Fundamentals of Dependency Theory. Tech. Rep. RJ4858, IBM Research (1985)