CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Generalized Resolution Search

Marius Posta
Jacques A. Ferland
Philippe Michelon

April 2009

CIRRELT-2009-16

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince, local 2642
Québec (Québec)
Canada G1K 7P4
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Generalized Resolution Search

Marius Posta[1,*], Jacques Ferland[1], Philippe Michelon[2]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Computer Science and Operations Research, Université de Montréal, P.O. Box 6128, Station Centre-ville, Montréal, Canada H3C 3J7

[2] Université d'Avignon et des Pays du Vaucluse, Laboratoire d'Informatique d'Avignon, F-84911 Avignon, Cedex 9, France

Abstract. Difficult discrete optimization problems are often solved using a Branch and Bound approach. Resolution search is an alternate approach proposed by Chvátal for 0-1 problems, allowing more flexibility in the search process. In this paper, we generalize the Resolution Search to any discrete problem.

Keywords. Resolution Search, optimization problems, discrete problems.

Acknowledgements. This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

---

* Corresponding author: Marius.Posta@cirrelt.ca

This document is also published as Publication #1338 by the Department of Computer Science and Operations Research of the Université de Montréal.

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2009

1. Introduction and general concepts

Consider the following problem

$$\min\{f(x) : x \in \mathcal{X}\} \tag{P}$$

where $\mathcal{X}$ is discrete and where the objective function f is defined as follows:

$$\begin{array}{rccl} f: & \mathcal{X} & \longrightarrow & \mathbb{R} \cup \{+\infty\} \\ & x & \longmapsto & \left\{ \begin{array}{l} +\infty \text{ if solution } x \text{ is not feasible} \\ f(x) \text{ otherwise} \end{array} \right. \end{array}$$

Solving (P) consists in identifying a solution $x^* \in \mathcal{X}$ which minimizes the objective function f.

Chvátal was the first author to introduce in [4] a Resolution Search approach to deal with binary variable problems, i.e. those where $\mathcal{X} = \{0,1\}^n$. This approach is an alternative to the Branch and Bound approach. Then Hanafi and Glover [5] were the first to generalize the Resolution Search approach for mixed integer programs. Furthermore, they provided interesting parallels with earlier approaches like Dynamic Branch and Bound.

A few straightforward implementation attempts were then published, where a Resolution Search approach was used to deal with problems with varying degrees of success. Demassey et al. [1] were the first to implement a Resolution Search approach to deal with the RCPSP. Then Palpant et al. [6], and more recently Boussier et al. [3], applied the approach to deal with the n^2-queens and 0-1 multidimensional knapsack problems, respectively. These authors focused on finding an application for which a Resolution Search approach would be competitive compared to the more traditional Branch and Bound approach, and they did not develop the theoretical aspects of Resolution Search much further. Demassey et al. [1], and Palpant et al. [6] report encouraging results despite not being competitive with the state of the art for their respective problems. Furthermore, Boussier et al. [3] were able to exactly solve previously unsolved instances of the 0-1 multidimensional knapsack problem.

In this paper, we further generalize the Resolution Search approach to any discrete optimization problem (P). For this purpose, we have to redefine some concepts introduced in [4], and to define new ones. We lean heavily on the ideas presented in [4] in the context of 0-1 problems to prove convergence of the search procedure to solve any discrete optimization problem.

1.1. Example

We now introduce a toy problem and an instance which will be used as an example throughout this paper. Consider a decreasing curve Γ in the plane, and suppose that its mathematical formulation is not known explicitely. However, we know where Γ intersects the coordinate axes, and for any point (l, h) an oracle is available to indicate if it lies below or above Γ. The objective is to determine the rectangle with the largest area while satisfying the following constraints:

- its extreme vertices must be $(0,0)$, $(0,h)$, $(l,0)$ and (l,h),
- l and h must be non-negative integers,
- and (l,h) must lie below Γ.

Assume that Γ intersects the coordinate axes at $(l_\cap, 0)$ and $(0, h_\cap)$. If we denote $l_{\max} = \lfloor l_\cap \rfloor$ and $h_{\max} = \lfloor h_\cap \rfloor$, then the search space for this problem is:

$$\mathcal{X} = \left\{(l,h) \in \mathbb{Z}^2 : 0 \leq l \leq l_{\max}, 0 \leq h \leq h_{max}\right\}.$$

Furthermore, denote the objective function f to be minimized over $\mathcal{X}$ as follows:

$$f(l,h) = \begin{cases} +\infty \text{ if } (l,h) \text{ lies above } \Gamma \\ -lh \text{ otherwise.} \end{cases}$$

An initial upper bound on the optimal value is $\bar{z} = 0$. Denote by $r^* = (l^*, h^*)$ the best known solution so far.

In figure 1, the curve Γ intersects the l axis between 5 and 6 and the h axis between 4 and 5. Accordingly, given that Γ is decreasing it follows that the optimal upper right corner of the rectangle must belong to the following set:

$$\mathcal{X} = \{(l,h) \in \mathbb{Z}^2 : 0 \leq l \leq 5, 0 \leq h \leq 4\}.$$

In the figure, the point $(\tilde{l}, \tilde{h}) = (3,2)$ lies below the curve so that $f(3,2) = -6$.

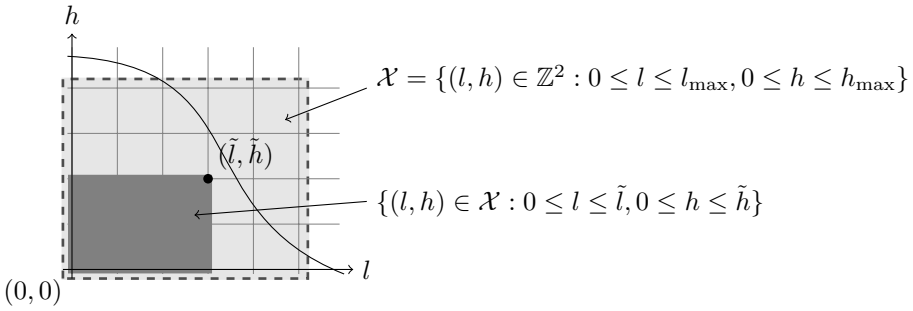


Figure 1: A problem instance, and a solution $(\tilde{l}, \tilde{h}) = (3,2)$.

1.2. Definitions

In general, a problem of type (P) is too difficult to be solved directly. For this reason we often use the popular divide-and-conquer strategy: the solution set $\mathcal{X}$ is partitioned into subsets $\mathcal{X}_1 \cup \mathcal{X}_2 \cup ... \cup \mathcal{X}_m = \mathcal{X}$ such that the objective function f is easily minimized on each subset. Then a minimum value z^* of f can be inferred on $\mathcal{X}$. Note that if we consider the parts of $\mathcal{X}$ successively one after the other, as we usually do, it is not always necessary to find a minimum

on each of them. Indeed, assume that $\bar{z}$ is the value of the best known solution when the subset $\mathcal{X}_i$ is to be examined. If we can verify that the mimimum value of f on $\mathcal{X}_i$ is greater than or equal to $\bar{z}$, then we can infer that no better solution of the problem (P) can be found in $\mathcal{X}_i$.

First we introduce some preliminary definitions.

Definition 1. A subset $\mathcal{X}_i$ of $\mathcal{X}$ is **explored** if one of the following alternatives holds:

- There is no better solution in $\mathcal{X}_i$; i.e. given an upper bound $\bar{z}$ on the optimum value of (P), $f(x) \geq \bar{z}$ for all $x \in \mathcal{X}_i$.
- $x' = \arg\min\{f(x) : x \in \mathcal{X}_i\}$ verifies $f(x') < \bar{z}$.[1]

The problem (P) is solved once $\mathcal{X}$ has been entirely explored.

Definition 2. A **predicate** γ on $\mathcal{X}$ is a function on $\mathcal{X}$ such that each element in $\mathcal{X}$ is either verified by γ or is not.

Definition 3. A set C of predicates on $\mathcal{X}$ is called a **clause**. It defines a subset $X(C)$ of $\mathcal{X}$. $X(C)$ is called its **clause cover**.

$$X(C) = \{x \in \mathcal{X} : x \text{ verifies all predicates } \gamma \in C\}$$

Any solution or any set of solutions is said to be **covered** by a clause C if it is included in the clause cover $X(C)$. For the sake of consistency, $X(\emptyset) = \mathcal{X}$. Furthermore, we assume that any predicate γ included in any clause has a **complement predicate** $\bar{\gamma}$ in the sense that any $x \in \mathcal{X}$ verifies either γ or $\bar{\gamma}$.

Here we merely extend the concept of clauses, as presented by Chvátal [4] for 0-1 problems, and subsequently generalized to integer programming by Hanafi and Glover [5].[2]

Example 1. As an example, consider the set $\mathcal{X} = \{(l,h) \in \mathbb{Z}^2 : 0 \leq l \leq 5, 0 \leq h \leq 4\}$ of our toy problem. Let us define a predicate γ, verified for any $(l,h) \in \mathcal{X}$ if and only if $0 \leq l \leq 3$. The predicate $\bar{\gamma}$ verified for any $(l,h) \in \mathcal{X}$ such that $4 \leq l \leq 5$ is complementary to γ. □

The notation $X(C)$ to designate the set of solutions verifying a clause C is borrowed from Hanafi and Glover [5].

Remark 1. Let A and B be two clauses. Then $X(A \cup B) = X(A) \cap X(B)$, and $A \subseteq B$ implies $X(B) \subseteq X(A)$.

Definition 4. A clause C is declared **nogood** if the corresponding cover $X(C)$ has been explored.[3]

[1] In this case x' becomes the best known solution and $\bar{z}$ is updated accordingly.

[2] Chvátal [4] defines his clauses as sets of literals, literals can be seen as a particular kind of predicate.

[3] Note that the word 'nogood' is borrowed from the constraint programming terminology. Chvátal [4] would say that a specific clause is '$\bar{z}$-forcing' instead.

2. Outline of the Resolution Search approach

In the Branch and Bound approach, the search space $\mathcal{X}$ is partitioned recursively. Let $\mathcal{L}$ be the list containing the subsets of $\mathcal{X}$ left to be explored (initially, $\mathcal{L} = [\mathcal{X}]$). An upper bound $\bar{z}$ on the optimal value of (P) corresponding to the best known solution is available and updated during the procedure. At a specific iteration i, a subset $\mathcal{X}_i \subseteq \mathcal{X}$ is removed from the list $\mathcal{L}$. A bounding procedure is applied on $\mathcal{X}_i$ to generate a lower bound $\underline{z}_i$ on the values in $f(\mathcal{X}_i)$. In doing so the bounding procedure also updates an upper bound $\bar{z}$ on the optimal value of (P). If $\underline{z}_i < \bar{z}$, then the subset $\mathcal{X}_i$ is split into two subsets to be added to $\mathcal{L}$. The procedure then loops to the next iteration $i+1$, unless $\mathcal{L}$ is empty, in which case $\bar{z}$ is the optimal value of (P).

The Resolution Search approach does not rely on such a bounding procedure. Instead, at each iteration of the Resolution Search, a clause U specifies a nonempty subset $X(U) \subseteq \mathcal{X}$ which is to be at least partly explored using a procedure ***obstacle***[4]. This procedure generates a nogood clause S which covers some elements of $X(U)$ (i.e. $X(S) \cap X(U) \neq \emptyset$), and also updates the upper bound $\bar{z}$ on the optimal value of (P).

Note that in order to implement efficiently the Resolution Search approach for solving an optimization problem, the definition of the *obstacle* procedure is a key point, in the same way as computing a lower bound is a matter of prime importance in a Branch and Bound scheme.

Example 2. For our toy problem, a procedure *obstacle* can be specified as follows. Consider the clause U, and select $(\tilde{l}, \tilde{h})$ randomly among the elements in $X(U)$:

- If $(\tilde{l}, \tilde{h})$ lies above Γ, then any solution (l, h) verifying $\tilde{l} \leq l \leq l_{\max}$ and $\tilde{h} \leq h \leq h_{\max}$ has a value $f(l, h) = +\infty$.
- If $(\tilde{l}, \tilde{h})$ lies below Γ, then any solution (l, h) verifying $0 \leq l \leq \tilde{l}$ and $0 \leq h \leq \tilde{h}$ is feasible and the area of the corresponding rectangle is smaller than or equal to the one corresponding to $(\tilde{l}, \tilde{h})$.

Accordingly, the nogood clause is specified as $S = \{\tilde{l} \leq l \leq l_{\max}, \tilde{h} \leq h \leq h_{\max}\}$ or $S = \{0 \leq l \leq \tilde{l}, 0 \leq h \leq \tilde{h}\}$. Since $(\tilde{l}, \tilde{h}) \in X(U)$, it follows that $X(S) \cap X(U) \neq \emptyset$. Finally, this procedure updates $\bar{z}$ and $r^* = (l^*, h^*)$ when required. □

The nogood clauses generated by *obstacle* during the search are kept in a stack of clauses $\mathcal{F}$ called a **family**. Before outlining how Resolution Search explores $\mathcal{X}$ with such a procedure *obstacle*, we extend the notion of cover to families of clauses.

[4] This name was chosen in order to stay consistent with Chvátal's terminology in [4].

Definition 5. The **reach**[5] $\mathcal{R}(\mathcal{F})$ of a family $\mathcal{F} = [C_1, C_2, ..., C_m]$ is the set of all solutions in $\mathcal{X}$ covered by at least one clause in $\mathcal{F}$:

$$\mathcal{R}(\mathcal{F}) = \bigcup_{i=1}^{m} X(C_i).$$

Procedure *ResolutionSearch*:

Step 0.

Let $\mathcal{F} = \emptyset$, the clause $U_{\mathcal{F}} = \emptyset$ and the integer $m = 0$.

Step 1.

Let S be the nogood clause generated by $obstacle(U_{\mathcal{F}})$.

Step 2.

Generate $\mathcal{F}'$ using $\mathcal{F}$ and the clause S.
If $\mathcal{R}(\mathcal{F}') = \mathcal{X}$ then $\mathcal{X}$ has been completely explored.
The search is then completed, the optimal value of f on $\mathcal{X}$ is $\bar{z}$.
Otherwise, generate a clause $U_{\mathcal{F}'}$ such that its cover $X(U_{\mathcal{F}'})$ is nonempty and does not share any solutions with $\mathcal{R}(\mathcal{F}')$, i.e. $X(U_{\mathcal{F}'}) \cap \mathcal{R}(\mathcal{F}') = \emptyset$.

Step 3.

Increment m, replace $\mathcal{F}$ by $\mathcal{F}'$ and $U_{\mathcal{F}}$ by $U_{\mathcal{F}'}$.
Return to Step 1.

The main feature of the Resolution Search is included in Step 2.

A naive way of implementing Step 2 would be as follows: generate $\mathcal{F}'$ by adding the clause S to the existing family $\mathcal{F}$. Assuming that we could generate a suitable clause $U_{\mathcal{F}'}$, then the reach of the family would grow strictly at each iteration (i.e. $\mathcal{R}(\mathcal{F}) \subsetneq \mathcal{R}(\mathcal{F}')$). Indeed, the nogood clause S has a cover $X(S)$ sharing some elements with $X(U_{\mathcal{F}})$, but $X(U_{\mathcal{F}})$ does not share any element with $\mathcal{R}(\mathcal{F})$. If follows that $\mathcal{X}$ would be explored in at most $|\mathcal{X}|$ iterations. However, since the family would grow as the search progresses, the generation of a suitable $U_{\mathcal{F}'}$ would become a problem in itself.

For this reason we impose an additional property on *obstacle*. It will allow the family $\mathcal{F}$ to maintain a special recursive structure, to generate more easily $U_{\mathcal{F}'}$ at each iteration. This structure on the family is defined in the following section.

3. Path-like structure

We first introduce the following definitions.

Definition 6. A predicate γ is said to be **markable** for a clause $U_{\mathcal{F}}$ if:

[5] The reach is equivalent to the τ function of Chvátal [4], in that $|\mathcal{R}(\mathcal{F})| = \tau(\mathcal{F})$ for 0-1 problems.

- γ partitions the search space (i.e. $X(\{\gamma\}) \neq \emptyset$ and $X(\{\bar{\gamma}\}) \neq \emptyset$), and
- γ is not in the clause $U_{\mathcal{F}}$ (i.e. $\gamma \notin U_{\mathcal{F}}$).

Example 3. Consider the toy problem presented earlier. In this context, the clauses are defined with predicates specified in terms of lower or upper bounds on the integer decision variables l and h. For the sake of clarity, the predicates (which are functions) are specified by the corresponding bounds on the variables.

Suppose that $U_{\mathcal{F}} = \{3 \leq l \leq 5, 0 \leq h \leq 3, 0 \leq l \leq 4\}$. The predicates $(1 \leq h \leq 4)$ and $(2 \leq l \leq 5)$ are markable for this $U_{\mathcal{F}}$, whereas $(0 \leq h \leq 3)$ is not because it is in $U_{\mathcal{F}}$, nor is $(0 \leq l)$ because it does not partition the search space (the predicate is verified by all $(l, h) \in \mathcal{X}$). □

Definition 7. A clause C is said to **maintain the path-like structure** for a clause $U_{\mathcal{F}}$ if for all markable predicates $\gamma \in C$, the intersection of $X(U_{\mathcal{F}})$ and $X(\bar{C}_\gamma)$ is nonempty (i.e. $X(U_{\mathcal{F}}) \cap X(\bar{C}_\gamma) \neq \emptyset$), where $\bar{C}_\gamma = (C \setminus \{\gamma\}) \cup \{\bar{\gamma}\}$.

We now define a specific recursive structure for the clause families.

Definition 8. The family $\mathcal{F}' = [C_1, C_2, ..., C_m, C_{m+1}]$ is **path-like** if the family $\mathcal{F} = [C_1, C_2, ..., C_m]$ is path-like, and if there is a clause $U_{\mathcal{F}}$ such that:

- Its cover is nonempty, i.e. $X(U_{\mathcal{F}}) \neq \emptyset$.
- Its cover has no intersection with the reach of $\mathcal{F}$; i.e. $X(U_{\mathcal{F}}) \cap \mathcal{R}(\mathcal{F}) = \emptyset$.
- The clause C_{m+1} contains at least one markable predicate for $U_{\mathcal{F}}$.
- The clause C_{m+1} maintains the path-like structure for $U_{\mathcal{F}}$.

In order to use the recursivity of this definition, we consider an empty family $\mathcal{F}_0$ to be path-like. We assume that $\mathcal{R}(\mathcal{F}_0) = \emptyset$, and that $U_{\mathcal{F}_0} = \emptyset$. The following proposition introduces a way to generate a clause $U_{\mathcal{F}'}$ when the family $\mathcal{F}'$ is path-like.

Proposition 1. *Given a path-like family $\mathcal{F}'$, consider a clause $U_{\mathcal{F}'}$ built as follows. Choose any markable predicate μ_{m+1} in C_{m+1} for $U_{\mathcal{F}}$. Let $\bar{C}_{m+1} = (C_{m+1} \setminus \{\mu_{m+1}\}) \cup \{\bar{\mu}_{m+1}\}$, and $U_{\mathcal{F}'} = U_{\mathcal{F}} \cup \bar{C}_{m+1}$. Such a clause $U_{\mathcal{F}'}$ always exists, its cover $X(U_{\mathcal{F}'})$ is nonempty and has no intersection with the reach of $\mathcal{F}'$ (i.e. $X(U_{\mathcal{F}'}) \cap \mathcal{R}(\mathcal{F}') = \emptyset$).*

Proof. The family $\mathcal{F}'$ being path-like, C_{m+1} maintains the path-like structure for $U_{\mathcal{F}}$, and we know there is at least one markable predicate $\mu_{m+1} \in C_{m+1}$ for $U_{\mathcal{F}}$. Therefore $X(U_{\mathcal{F}}) \cap X(\bar{C}_{m+1})$ is nonempty for $\bar{C}_{m+1}$ generated with any such μ_{m+1}.

Since $X(U_{\mathcal{F}}) \cap X(\bar{C}_{m+1}) = X(U_{\mathcal{F}} \cup \bar{C}_{m+1})$ and since $U_{\mathcal{F}'} = U_{\mathcal{F}} \cup \bar{C}_{m+1}$, we thus have that $X(U_{\mathcal{F}'}) = X(U_{\mathcal{F}}) \cap X(\bar{C}_{m+1})$. It follows that $X(U_{\mathcal{F}'})$ is nonempty.

We now prove by induction that $\mathcal{R}(\mathcal{F}') \cap X(U_{\mathcal{F}'}) = \emptyset$. Since we consider the reach of an empty family to be empty, the property is true for the initial family $\mathcal{F}_0 = \emptyset$.

Now suppose that $\mathcal{R}(\mathcal{F})$ and $X(U_{\mathcal{F}})$ have no intersection; i.e. $\mathcal{R}(\mathcal{F}) \cap X(U_{\mathcal{F}}) = \emptyset$. Recall that by construction, $U_{\mathcal{F}'} = U_{\mathcal{F}} \cup \bar{C}_{m+1}$.

On the one hand, since $\bar{C}_{m+1} \subseteq U_{\mathcal{F}'}$, it follows that $X(U_{\mathcal{F}'}) \subseteq X(\bar{C}_{m+1})$. Because $\mu_{m+1} \in C_{m+1}$ and $\bar{\mu}_{m+1} \in \bar{C}_{m+1}$, we know that the covers $X(\bar{C}_{m+1})$ and $X(C_{m+1})$ have no intersection. Therefore, it follows that the intersection $X(C_{m+1}) \cap X(U_{\mathcal{F}'})$ is empty.

On the other hand, since $U_{\mathcal{F}} \subsetneq U_{\mathcal{F}'}$, it follows that $X(U_{\mathcal{F}'}) \subsetneq X(U_{\mathcal{F}})$. According to the induction hypothesis, $\mathcal{R}(\mathcal{F}) \cap X(U_{\mathcal{F}}) = \emptyset$. Therefore the intersection $\mathcal{R}(\mathcal{F}) \cap X(U_{\mathcal{F}'})$ is also empty.

Finally, since $\mathcal{R}(\mathcal{F}') = \mathcal{R}(\mathcal{F}) \cup X(C_{m+1})$, it follows that $\mathcal{R}(\mathcal{F}') \cap X(U_{\mathcal{F}'}) = \emptyset$. □

Definition 9. Let us associate a unique **marked predicate** μ_i with each nogood clause C_i in a path-like family. Let $\mathcal{M} = [\mu_1, \mu_2, ..., \mu_m]$ be the set of marked predicates associated with $\mathcal{F} = [C_1, C_2, ..., C_m]$.

In proposition 1, we define $U_{\mathcal{F}'} = U_{\mathcal{F}} \cup \bar{C}_{m+1}$, and recursively

$$U_{\mathcal{F}'} = \bigcup_{i=1}^{m+1} \bar{C}_i$$

where $\bar{C}_i = (C_i \setminus \{\mu_i\}) \cup \{\bar{\mu}_i\}$. It follows that if the family is path-like, then we can easily build a clause that covers a nonempty subset of $\mathcal{X}$ having no intersection with the reach of this family. We may then call *obstacle* with this clause in order to generate a new nogood clause S.

Example 4. Returning to our toy problem, using the *obstacle* procedure defined in example 2, we shall see that given any $U_{\mathcal{F}}$, the nogood clause $S = \mathit{obstacle}(U_{\mathcal{F}})$ always maintains the path-like structure for $U_{\mathcal{F}}$.

In this problem, the predicates in the clauses are bounds on l and h for $(l, h) \in \mathcal{X}$. Therefore, the cover of a clause is defined by the largest lower bound and the lowest upper bound on l and h: $\underline{l}$, $\bar{l}$, $\underline{h}$, $\bar{h}$ respectively. This is true in particular for the clause $U_{\mathcal{F}}$, and its cover is therefore always of the following type:

$$X(U_{\mathcal{F}}) = \left\{(l, h) \in \mathcal{X} : \underline{l} \leq l \leq \bar{l}, \underline{h} \leq h \leq \bar{h}\right\}.$$

Next, recall that the procedure *obstacle* is specified by randomly selecting a vertex $(\tilde{l}, \tilde{h}) \in X(U_{\mathcal{F}})$. The procedure then infers a nogood clause S of the form $\{0 \leq l \leq \tilde{l}, 0 \leq h \leq \tilde{h}\}$ or $\{\tilde{l} \leq l \leq l_{\max}, \tilde{h} \leq h \leq h_{\max}\}$ when $(\tilde{l}, \tilde{h})$ is below or above Γ, respectively.

Finally, for each markable predicate $\gamma \in S$ for $U_{\mathcal{F}}$, it follows that for $\bar{S}_\gamma = (S \setminus \{\gamma\}) \cup \{\bar{\gamma}\}$ there is at least one vertex in both $X(\bar{S}_\gamma)$ and $X(U_{\mathcal{F}})$. Indeed, consider the following situations:

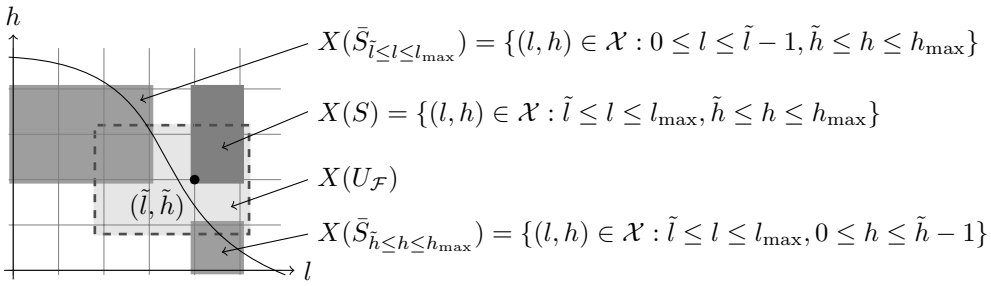


Figure 2: $S = \mathit{obstacle}(U_{\mathcal{F}})$ maintains the path-like structure for $U_{\mathcal{F}}$.

- If $\gamma = (\tilde{l} \leq l \leq l_{\max})$ then $(\tilde{l}-1, \tilde{h}) \in X(\bar{S}_{\gamma})$. Indeed, $X(\{\bar{\gamma}\}) \neq \emptyset$, hence $\tilde{l} \geq 1$. Also, since γ is not in $U_{\mathcal{F}}$, it follows that the largest lower bound on l in $U_{\mathcal{F}}$ is at most $\tilde{l}-1$, thus $(\tilde{l}-1, \tilde{h}) \in X(U_{\mathcal{F}})$. This case is illustrated in figure 2.
- If $\gamma = (0 \leq l \leq \tilde{l})$, $\gamma = (\tilde{h} \leq h \leq h_{\max})$, or $\gamma = (0 \leq h \leq \tilde{h})$, then using a similar argument, it follows that $(\tilde{l}+1, \tilde{h})$, $(\tilde{l}, \tilde{h}-1)$ and $(\tilde{l}, \tilde{h}+1)$ are in $X(\bar{S}_{\gamma}) \cap X(U_{\mathcal{F}})$, respectively. □

4. Updating the family

So far, the following assumptions on *obstacle* are required to hold at each iteration. Given any clause $U_{\mathcal{F}}$, denote by S the clause generated by $\mathit{obstacle}(U_{\mathcal{F}})$:

- S is a nogood clause,
- S verifies $X(S) \cap X(U_{\mathcal{F}}) \neq \emptyset$.

In order to update the family $\mathcal{F}$ using the nogood clause S generated by $\mathit{obstacle}(U_{\mathcal{F}})$, we require the following additional assumptions to hold at each iteration:

- $\mathcal{F}$ is a path-like family,
- S maintains the path-like structure for $U_{\mathcal{F}}$.

Henceforth, we require no further assumptions on the procedure *obstacle*. Note that this is in contrast to previous work in [4] and [5] where the behavior of the *obstacle* procedure is much more precisely specified. They assume that there is a 1-to-1 mapping from clauses to partial solutions, i.e. vectors whose components are not all instanciated. They also assume that the procedure *obstacle* is seperable in two specific phases: a waxing phase and a waning phase, that can be described as follows, using our terminology. First, in the waxing phase, a clause U^+ is constructed from $U_{\mathcal{F}}$, and additional predicates are added to U^+ until it becomes nogood. Then, in the waning phase, a clause S is

constructed from U^+, and predicates are removed from S one by one while maintaining the nogood property. The *obstacle* procedures satisfying such a specification are included in the set of *obstacle* procedures that we allow.

However, the path-like families and their update, as presented in this paper, are direct generalizations of the notions introduced by Chvátal [4]. There are two cases to consider: either S contains at least one markable predicate for $U_{\mathcal{F}}$, or it does not.

4.1. S contains at least one markable predicate for $U_{\mathcal{F}}$

In this case we can easily get a new path-like family $\mathcal{F}'$ from $\mathcal{F}$ and S by adding to $\mathcal{F}$ the clause S in position $m+1$; i.e. $\mathcal{F}' = [C_1, C_2, ..., C_m, C_{m+1}]$ where $C_{m+1} = S$.

Proposition 2. *If S contains at least one markable predicate for $U_{\mathcal{F}}$ and if $\mathcal{F} = [C_1, C_2, ..., C_m]$ is path-like, then the family $\mathcal{F}' = [C_1, C_2, ..., C_m, S]$ is path-like.*

Proof. Trivially, by defining $C_{m+1} = S$, it follows that C_{m+1} contains at least one markable predicate for $U_{\mathcal{F}}$. Since S was generated by $obstacle(U_{\mathcal{F}})$, it follows that C_{m+1} maintains the path-like structure for $U_{\mathcal{F}}$. Hence, $\mathcal{F}'$ is path-like by definition 8. □

In order to generate the clause $U_{\mathcal{F}'}$ to be used with *obstacle* at the next iteration, we proceed as in Proposition 1.

Example 5. Referring to our toy problem, this case can be illustrated as follows. Suppose we have

$$\mathcal{F} = \begin{bmatrix} \{\underline{0 \leq l \leq 2}, 0 \leq h \leq 3\} \\ \{0 \leq l \leq 4, \underline{0 \leq h \leq 1}\} \end{bmatrix}$$

where the underlined relations correspond to the marked predicates in $\mathcal{M}$. Accordingly, $U_{\mathcal{F}} = \{3 \leq l \leq 5, 0 \leq h \leq 3, 0 \leq l \leq 4, 2 \leq h \leq 4\}$.

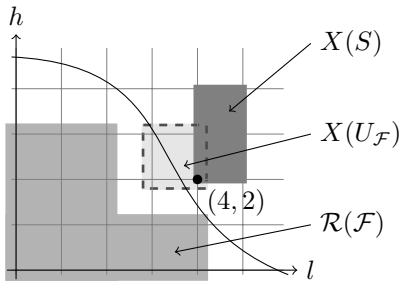


Figure 3: Search state at this iteration.

To generate S, suppose that the point $(4, 2)$ is selected randomly by *obstacle* in $X(U_{\mathcal{F}}) = \{(l, h) \in \mathbb{Z}^2 : 3 \leq l \leq 4, 2 \leq h \leq 3\}$. Note that $(4, 2)$ lies above Γ.

Thus $S = \{4 \leq l \leq 5, 2 \leq h \leq 4\}$, as illustrated in figure 3.

Since the predicate $(4 \leq l \leq 5)$ in S is markable for $U_{\mathcal{F}}$, the family $\mathcal{F}'$ is generated by adding to $\mathcal{F}$ the clause S in position 3. The predicate $\mu_S = (4 \leq l \leq 5)$ is selected as the marked predicate for the last clause, and it follows that

$$\mathcal{F}' = \begin{bmatrix} \{\underline{0 \leq l \leq 2}, 0 \leq h \leq 3\} \\ \{0 \leq l \leq 4, \underline{0 \leq h \leq 1}\} \\ \{\underline{4 \leq l \leq 5}, 2 \leq h \leq 4\} \end{bmatrix}$$

and $U_{\mathcal{F}'} = \{3 \leq l \leq 5, 0 \leq h \leq 3, 0 \leq l \leq 4, 2 \leq h \leq 4, 0 \leq l \leq 3, 2 \leq h \leq 4\}$, as illustrated in figure 4. □

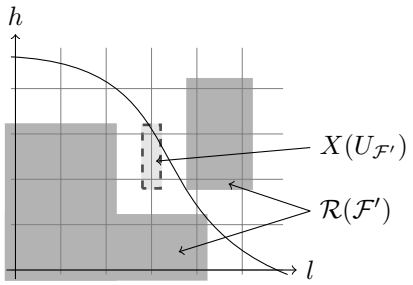


Figure 4: Search state after this iteration.

4.2. S contains no markable predicate for $U_{\mathcal{F}}$

In this case, adding the clause S at the end of $\mathcal{F}$ does not infer a path-like family. However, it is possible to deduce a new path-like family of nogood clauses using S and $\mathcal{F}$. But first, we have to introduce the notion of **clause resolution**.

Definition 10. Let A and B be two clauses such that there is one and only predicate γ such that $\gamma \in A$ and $\bar{\gamma} \in B$. The **resolvent** of A and B is defined as the clause:

$$A \nabla B = (A \setminus \{\gamma\}) \cup (B \setminus \{\bar{\gamma}\}).$$

The following result shows that the resolution operation to define the resolvent of two nogood clauses A and B preserves the nogood property of the resolvent $A \nabla B$.

Proposition 3. *If A and B are two nogood clauses having a resolvent $A \nabla B$, then $A \nabla B$ is also a nogood clause.*

Proof. Denote $A = A' \cup \{\gamma\}$ and $B = B' \cup \{\bar{\gamma}\}$. Hence $A \nabla B = A' \cup B'$.

Since γ and $\bar{\gamma}$ are complement of each other, it follows by definition that any solution in $\mathcal{X}$ verifies either γ or $\bar{\gamma}$. In particular, this is true for any solution $x \in X(A' \cup B')$.

Therefore, if x verifies γ, then since it also verifies A', it also verifies A. Otherwise, x verifies both $\bar{\gamma}$ and B', and hence B. It follows that any solution $x \in X(A' \cup B')$ verifies A or B. The assumption that A and B are nogood clauses implies that $A \nabla B = A' \cup B'$ is also nogood. □

This result is a generalization of the clause resolution mechanism for integer programming problems presented in [5], which itself is a generalization of clause resolution as presented in [4] in the context of 0-1 problems. It allows to introduce the following procedure generating recursively a new nogood clause R using the nogood clause S generated by *obstacle*$(U_{\mathcal{F}})$ and some nogood clauses in $\mathcal{F}$.

Procedure *ResolventGeneration*$(S, \mathcal{F}, \mathcal{M})$:
Step 0.
Let $R = S$ and $i = m$.
Step 1.
$\mu_i \in \mathcal{M}$ is the marked predicate associated with the nogood $C_i \in \mathcal{F}$.
If its complement $\bar{\mu}_i$ is in R, replace R by $R \nabla C_i$.
Step 2.
Decrement i.
If $i = 0$, then return R, else go to Step 1.

Proposition 4. *Let S be the nogood clause generated by obstacle$(U_{\mathcal{F}})$. If S contains no markable predicate for $U_{\mathcal{F}}$, then the clause R generated by ResolventGeneration$(S, \mathcal{F}, \mathcal{M})$ is nogood, and R contains no markable predicate for $U_{\mathcal{F}}$ either.*

Proof. We prove this proposition by induction. At the beginning of the *ResolventGeneration* procedure when $i = m$, $R = S$. S is a nogood clause and contains no markable predicate for $U_{\mathcal{F}}$, hence the proposition is true for $i = m$.

Suppose that at the beginning of iteration i of *ResolventGeneration*, R is a nogood clause and R contains no markable predicate for $U_{\mathcal{F}}$. If $\bar{\mu}_i$ is not in R, then R is not modified during this iteration. Now suppose that $\bar{\mu}_i \in R$. We first show that $\mu_i \in C_i$ is the unique element of C_i allowing R and C_i to have a resolvent.

For contradiction, suppose that there is another predicate $\gamma \neq \mu_i$ such that $\gamma \in C_i$ and $\bar{\gamma} \in R$. On the one hand, since R contains no markable predicate for $U_{\mathcal{F}}$, all predicates in R are either in $U_{\mathcal{F}}$ or are verified for all elements in $\mathcal{X}$. We know $X(C_i) \neq \emptyset$ and we have $\gamma \in C_i$, it follows that $\bar{\gamma} \in U_{\mathcal{F}}$. On the other hand, we have $\bar{C}_i = (C_i \setminus \{\mu_i\}) \cup \{\bar{\mu}_i\}$. Thus since $\gamma \in C_i$ and $\gamma \neq \mu_i$, we have $\gamma \in \bar{C}_i$. Therefore it follows that $\gamma \in U_{\mathcal{F}} = \cup_{j=1}^{m} \bar{C}_j$. But then $\gamma \in U_{\mathcal{F}}$

and $\bar{\gamma} \in U_{\mathcal{F}}$, a contradiction because $X(U_{\mathcal{F}}) \neq \emptyset$. Therefore the resolvent $R \nabla C_i = (R \setminus \{\bar{\mu}_i\}) \cup (C_i \setminus \{\mu_i\})$ exists and is a nogood clause by proposition 3.

By the induction hypothesis, $(R \setminus \{\bar{\mu}_i\})$ contains no markable predicate for $U_{\mathcal{F}}$. Also, because $U_{\mathcal{F}} = \cup_{j=1}^{m} \bar{C}_j$ where $\bar{C}_j = (C_j \setminus \{\mu_j\}) \cup \{\bar{\mu}_j\}$, we have that $(C_i \setminus \{\mu_i\}) \subsetneq \bar{C}_i \subseteq U_{\mathcal{F}}$. Therefore it follows that $R \nabla C_i$ contains no markable predicate for $U_{\mathcal{F}}$. □

We now show that if R does not cover the entire search space, there exists an index k, $1 \leq k \leq m$, such that a new path-like family can be generated by removing all the clauses C_i, $k \leq i \leq m$, from $\mathcal{F}$, and by adding the nogood clause R.

To ease the presentation, $\mathcal{F}_i = [C_1, C_2, ..., C_i]$ denotes the sub-families for $i = 1, ..., m$, and $\mathcal{F}_0 = \emptyset$. Because $\mathcal{F}$ is path-like, $\mathcal{F}_i$ is also necesseraly path-like.

Proposition 5. *Let R be the nogood clause generated by $ResolventGeneration(S, \mathcal{F}, \mathcal{M})$. Let k be the smallest index such that R contains no markable predicate for $U_{\mathcal{F}_k}$. If $k = 0$, then the search is completed, otherwise the family $\mathcal{F}' = [C_1, C_2, ..., C_{k-1}, R]$ is path-like.*

Proof. If $k = 0$, R contains no markable predicate for $U_{\mathcal{F}_0} = \emptyset$, this implies $X(R) = \mathcal{X}$ and since R is a nogood clause, it follows that the search space is completely explored.

Otherwise, we have $1 \leq k$, as well as $k \leq m$ since we know that R contains no markable predicate for $U_{\mathcal{F}} = U_{\mathcal{F}_m}$. Furthermore, since k is the smallest index such that R contains no markable predicate for $U_{\mathcal{F}_k}$, it also follows that R contains at least one markable predicate for $U_{\mathcal{F}_{k-1}}$. To complete the proof, we have to show that R maintains the path-like structure for $U_{\mathcal{F}_{k-1}}$, since the family $\mathcal{F}_{k-1}$ is path-like.

Since the clause R contains no markable predicate for $U_{\mathcal{F}_k} = U_{\mathcal{F}_{k-1}} \cup \bar{C}_k$, then it follows that all markable predicates in R for $U_{\mathcal{F}_{k-1}}$ (there is at least one) are in $\bar{C}_k \setminus U_{\mathcal{F}_{k-1}}$. Recall that $\bar{\mu}_k \notin U_{\mathcal{F}_{k-1}}$, and by construction $\bar{\mu}_k \notin R$, thus all markable predicates in R for $U_{\mathcal{F}_{k-1}}$ are also in $(\bar{C}_k \setminus \{\bar{\mu}_k\}) \setminus U_{\mathcal{F}_{k-1}}$, and thus also in $C_k \setminus U_{\mathcal{F}_{k-1}}$. Since the family $\mathcal{F}_k$ is path-like, C_k is a clause which maintains the path-like structure for $U_{\mathcal{F}_{k-1}}$, hence R also maintains the path-like structure for $U_{\mathcal{F}_{k-1}}$. □

Example 6. To illustrate this case using our toy problem, suppose that we are at the beginning of an iteration where $\bar{z} = -6$ and

$$\mathcal{F} = \begin{bmatrix} \{\underline{0 \leq l \leq 2}, 0 \leq h \leq 3\} \\ \{0 \leq l \leq 4, \underline{0 \leq h \leq 1}\} \\ \{\underline{4 \leq l \leq 5}, 2 \leq h \leq 4\} \end{bmatrix} \begin{matrix} : C_1 \\ : C_2 \\ : C_3, \end{matrix}$$

the underlined relations corresponding to the marked predicates in $\mathcal{M}$. Accordingly, $U_{\mathcal{F}} = \{3 \leq l \leq 5, 0 \leq h \leq 3, 0 \leq l \leq 4, 2 \leq h \leq 4, 0 \leq l \leq 3, 2 \leq h \leq 4\}$.

To generate S, suppose that the point $(3, 3)$ is selected randomly by *obstacle* in $X(U_{\mathcal{F}}) = \{(l, h) \in \mathbb{Z}^2 : 3 \leq l \leq 3, 2 \leq h \leq 3\}$. Note that $(3, 3)$ lies below Γ.

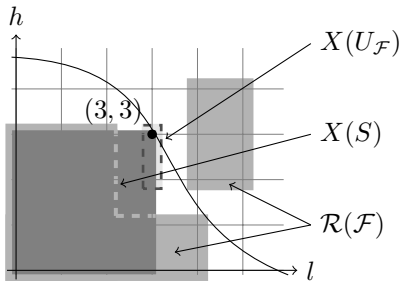


Figure 5: Search state at this iteration.

Thus $S = \{0 \leq l \leq 3, 0 \leq h \leq 3\}$ as illustrated in figure 5. Furthermore, since $f(3,3) < -6$, the *obstacle* procedure updates $\bar{z} = -9$ and $r^* = (3,3)$.

Since $S \subset U_{\mathcal{F}}$, it follows that S contains no markable predicates for $U_{\mathcal{F}}$. Hence we first generate R from the resolvents of S and the clauses in $\mathcal{F} = [C_1, C_2, C_3]$. Initiate the process with $R = S$ and $i = 3$.

$i = 3$**:** $\mu_3 = (4 \leq l \leq 5)$ implies that $\bar{\mu}_3 = (0 \leq l \leq 3)$, and this predicate is in R. Replace R with $R \nabla C_3 = (R \backslash \{\bar{\mu}_3\}) \cup (C_3 \backslash \{\mu_3\}) = \{0 \leq h \leq 3, 2 \leq h \leq 4\}$.

$i = 2$**:** $\mu_2 = (0 \leq h \leq 1)$ implies that $\bar{\mu}_2 = (2 \leq h \leq 4)$, and this predicate is in R. Replace R with $R \nabla C_2 = \{0 \leq h \leq 3, 0 \leq l \leq 4\}$.

$i = 1$**:** $\mu_1 = (0 \leq l \leq 2)$ implies that $\bar{\mu}_1 = (3 \leq l \leq 5)$, and this predicate is not in R.

The resulting nogood clause is $R = \{0 \leq h \leq 3, 0 \leq l \leq 4\}$.

Now determine the rank k such that R contains no markable predicates for $U_{\mathcal{F}_k}$. Initiate the process with $k = 0$ and $U_{\mathcal{F}_0} = \emptyset$.

$k = 0$**:** $(0 \leq h \leq 3) \in R$ is a markable predicate for $U_{\mathcal{F}_0} = \emptyset$.

$k = 1$**:** $(0 \leq l \leq 4) \in R$ is a markable predicate for $U_{\mathcal{F}_1} = U_{\mathcal{F}_0} \cup \bar{C}_1 = \{3 \leq l \leq 5, 0 \leq h \leq 3\}$.

$k = 2$**:** R contains no markable predicates for $U_{\mathcal{F}_2} = U_{\mathcal{F}_1} \cup \bar{C}_2 = \{3 \leq l \leq 5, 0 \leq h \leq 3, 0 \leq l \leq 4, 2 \leq h \leq 4\}$.

Thus $\mathcal{F}' = [C_1, R]$, and $\mu_R = (0 \leq l \leq 4)$ is selected as the marked predicate for the clause R. It follows that

$$\mathcal{F}' = \begin{bmatrix} \{\underline{0 \leq l \leq 2}, 0 \leq h \leq 3\} \\ \{\underline{0 \leq l \leq 4}, 0 \leq h \leq 3\} \end{bmatrix} \begin{matrix} : C_1 \\ : R \end{matrix},$$

and $U_{\mathcal{F}'} = \{3 \leq l \leq 5, 0 \leq h \leq 3, 5 \leq l \leq 5, 0 \leq h \leq 3\}$, as illustrated in figure 6. □

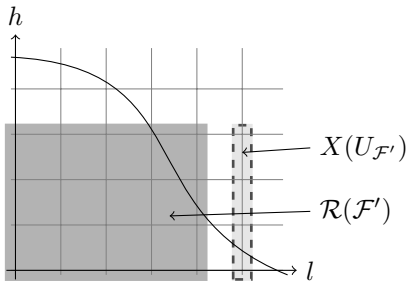


Figure 6: Search state after this iteration.

5. Complete algorithm and convergence

The Resolution Search approach can now be summarized in the following procedure:

Procedure *ResolutionSearch*:
Step 0 - Initialization.
Let $\mathcal{F} = \emptyset$, $\mathcal{M} = \emptyset$, $U_{\mathcal{F}} = \emptyset$ and $m = 0$.
Step 1 - Exploration.
Let S be the clause generated by $obstacle(U_{\mathcal{F}})$.
Step 2 - Construction.
If $S \subseteq U_{\mathcal{F}}$, go to Step 2.2.
Step 2.1 - Case S contains at least one markable predicate for $U_{\mathcal{F}}$.
Select $\mu_S \in S$, a markable predicate for $U_{\mathcal{F}}$.
Let $m' = m + 1$, $\mathcal{F}' = [C_1, ..., C_m, S]$ and $\mathcal{M}' = [\mu_1, ..., \mu_m, \mu_S]$.
Let $\bar{S} = (S \setminus \{\mu_S\}) \cup \{\bar{\mu}_S\}$ and $U_{\mathcal{F}'} = U_{\mathcal{F}} \cup \bar{S}$.
Go to Step 3.
Step 2.2 - Case S contains no markable predicate for $U_{\mathcal{F}}$.
Step 2.2.1 - Generate R.
Let R be the nogood clause generated by $ResolventGeneration(S, \mathcal{F}, \mathcal{M})$.
Let k be the smallest index such that R contains no markable predicate for $U_{\mathcal{F}_k}$.
Step 2.2.2 - Generate $\mathcal{F}'$.
If $k = 0$, return the best known solution, the search is completed.
Select $\mu_R \in R$, a markable predicate for $U_{\mathcal{F}_{k-1}}$.
Let $m' = k$, $\mathcal{F}' = [C_1, ..., C_{k-1}, R]$ and $\mathcal{M}' = [\mu_1, ..., \mu_{k-1}, \mu_R]$.
Let $\bar{R} = (R \setminus \{\mu_R\}) \cup \{\bar{\mu}_R\}$ and $U_{\mathcal{F}'} = U_{\mathcal{F}_{k-1}} \cup \bar{R}$.
Step 3 - Update.
Replace $\mathcal{F}$ by $\mathcal{F}'$, $\mathcal{M}$ by $\mathcal{M}'$, $U_{\mathcal{F}}$ by $U_{\mathcal{F}'}$, and m by m'.
Return to Step 1.

To prove the convergence of the Resolution Search procedure, we cannot rely on the strict increase of the reach $\mathcal{R}(\mathcal{F})$ at each iteration. Note that Chvátal also

observes this for a 0-1 programming problem in [4]. However, the convergence proof of the Resolution Search procedure relies on a subset of the reach $\mathcal{R}(\mathcal{F})$ that is strictly increasing at each iteration.

Definition 11. Given $\mathcal{F} = [C_1, C_2, ..., C_m]$ a path-like family and the associated set of marked predicates $\mathcal{M} = [\mu_1, \mu_2, ..., \mu_m]$, the **restricted reach** [6] $\check{\mathcal{R}}(\mathcal{F})$ of the family $\mathcal{F}$ is defined as follows:

$$\check{\mathcal{R}}(\mathcal{F}) = \bigcup_{i=1}^{m} X(\cup_{j=1}^{i-1} \bar{C}_j \cup C_i) = \bigcup_{i=1}^{m} X(U_{\mathcal{F}_{i-1}} \cup C_i).$$

Clearly, the restricted reach of any family is a subset of its reach. We consider two different lemmas to show that the restricted reach strictly increases at each iteration according to the way of generating the new family $\mathcal{F}'$.

Lemma 1. *Let S be the nogood clause generated by obstacle$(U_{\mathcal{F}})$. If S contains at least one markable predicate for $U_{\mathcal{F}}$, the family $\mathcal{F}'$ generated in Step 2.1 of the Resolution Search procedure verifies $\check{\mathcal{R}}(\mathcal{F}) \subsetneq \check{\mathcal{R}}(\mathcal{F}')$.*

Proof. By definition of the restricted reach and of the clause $U_{\mathcal{F}}$:

$$\check{\mathcal{R}}(\mathcal{F}') = \check{\mathcal{R}}(\mathcal{F}) \cup X(\cup_{j=1}^{m} \bar{C}_j \cup S) = \check{\mathcal{R}}(\mathcal{F}) \cup X(U_{\mathcal{F}} \cup S).$$

The proof is completed if we can show that $X(U_{\mathcal{F}} \cup S)$ is nonempty and not in $\check{\mathcal{R}}(\mathcal{F})$. By definitions of S and *obstacle*, $X(U_{\mathcal{F}} \cup S) = X(U_{\mathcal{F}}) \cap X(S) \neq \emptyset$. Also, since $X(U_{\mathcal{F}} \cup S) \subset X(U_{\mathcal{F}})$ and since $X(U_{\mathcal{F}}) \cap \mathcal{R}(\mathcal{F}) = \emptyset$, it follows that $X(U_{\mathcal{F}} \cup S) \cap \mathcal{R}(\mathcal{F}) = \emptyset$. Hence, since $\check{\mathcal{R}}(\mathcal{F}) \subseteq \mathcal{R}(\mathcal{F})$, it follows that $X(U_{\mathcal{F}} \cup S) \cap \check{\mathcal{R}}(\mathcal{F}) = \emptyset$. □

Lemma 2. *Let S be the nogood clause generated by obstacle$(U_{\mathcal{F}})$. If S contains no markable predicate for $U_{\mathcal{F}}$, the family $\mathcal{F}'$ generated in Step 2.2 of the Resolution Search procedure verifies $\check{\mathcal{R}}(\mathcal{F}) \subsetneq \check{\mathcal{R}}(\mathcal{F}')$.*

Proof. In this case $\mathcal{F}' = [C_1, C_2, ..., C_{k-1}, R]$ is built by adding R to $\mathcal{F}_{k-1}$. By definitions of the restricted reach and of the clause $U_{\mathcal{F}_{k-1}}$:

$$\check{\mathcal{R}}(\mathcal{F}') = \check{\mathcal{R}}(\mathcal{F}_{k-1}) \cup X(\cup_{j=1}^{k-1} \bar{C}_j \cup R) = \check{\mathcal{R}}(\mathcal{F}_{k-1}) \cup X(U_{\mathcal{F}_{k-1}} \cup R).$$

To show that $\check{\mathcal{R}}(\mathcal{F}) \subset \check{\mathcal{R}}(\mathcal{F}')$, consider any solution $x \in \check{\mathcal{R}}(\mathcal{F})$. Then $x \in X(U_{\mathcal{F}_{i-1}} \cup C_i)$ for at least one index i, $1 \leq i \leq m$.

- If $i < k$, then $x \in \check{\mathcal{R}}(\mathcal{F}_{k-1})$, and thus $x \in \check{\mathcal{R}}(\mathcal{F}')$.

[6] The restricted reach is equivalent to the σ strength function of Chvátal [4], in that $|\check{\mathcal{R}}(\mathcal{F})| = \sigma(\mathcal{F})$ for 0-1 problems.

- If $i = k$, then $x \in X(U_{\mathcal{F}_{k-1}} \cup C_k) \subsetneq X(U_{\mathcal{F}_{k-1}} \cup (C_k \setminus \{\mu_k\}))$. However, R contains no markable predicate for $U_{\mathcal{F}_k}$, and hence $X(U_{\mathcal{F}_k}) \subseteq X(R)$. Since neither μ_k nor $\bar{\mu}_k$ can belong to R, and since $U_{\mathcal{F}_k} = U_{\mathcal{F}_{k-1}} \cup \bar{C}_k$, it follows that $X(U_{\mathcal{F}_{k-1}} \cup (\bar{C}_k \setminus \{\bar{\mu}_k\})) \subseteq X(R)$. Therefore $x \in X(U_{\mathcal{F}_{k-1}} \cup R)$, and thus $x \in \check{\mathcal{R}}(\mathcal{F}')$.
- If $i > k$, then $x \in X(U_{\mathcal{F}_{i-1}} \cup C_i)$. Since $i > k$, we have $X(U_{\mathcal{F}_{i-1}} \cup C_i) \subsetneq X(U_{\mathcal{F}_k})$. As seen when $i = k$, we also have $X(U_{\mathcal{F}_k}) \subsetneq X(U_{\mathcal{F}_{k-1}} \cup R)$. Therefore $x \in X(U_{\mathcal{F}_{k-1}} \cup R)$, and thus $x \in \check{\mathcal{R}}(\mathcal{F}')$.

Next we show the inclusion to be strict.

The clause R contains no markable predicate for $U_{\mathcal{F}_k}$, and since $U_{\mathcal{F}_{k-1}} \subsetneq U_{\mathcal{F}_k} \subseteq U_{\mathcal{F}}$, R contains no markable predicate for $U_{\mathcal{F}}$ either, hence $X(U_{\mathcal{F}})$ is a subset of both $X(R)$ and $X(U_{\mathcal{F}_{k-1}})$. Therefore $X(U_{\mathcal{F}}) \subseteq X(U_{\mathcal{F}_{k-1}}) \cap X(R)$, or equivalently $X(U_{\mathcal{F}}) \subseteq X(U_{\mathcal{F}_{k-1}} \cup R)$. Then $X(U_{\mathcal{F}}) \subseteq \check{\mathcal{R}}(\mathcal{F}')$. However, $\check{\mathcal{R}}(\mathcal{F}) \subseteq \mathcal{R}(\mathcal{F})$ and $X(U_{\mathcal{F}}) \cap \mathcal{R}(\mathcal{F}) = \emptyset$. Thus since $X(U_{\mathcal{F}})$ is nonempty, it follows that there exists at least one solution in the restricted reach of $\mathcal{F}'$ that does not belong to the restricted reach of $\mathcal{F}$. □

Theorem 1. *The Resolution Search procedure completely explores $\mathcal{X}$ in at most $|\mathcal{X}|$ iterations.*

PROOF. According to the previous lemmas, $\check{\mathcal{R}}(\mathcal{F}) \subsetneq \check{\mathcal{R}}(\mathcal{F}')$ at every iteration. Since the restricted reach of a family is included in its reach, it follows that at each iteration, a subset of the reach of $\mathcal{F}$ increases strictly. In the worst case, it increases by exactly one solution at each iteration, and thus at most $|\mathcal{X}|$ iterations are needed for the search to complete. □

Having shown that the Resolution Search procedure converges, let us now concern ourselves with the nogood clauses which are discarded from the path-like family during the search.

6. Recycling nogood clauses

When the new family $\mathcal{F}'$ is generated via Step 2.2 of the Resolution Search procedure, the nogood clauses $C_k, ..., C_m$, and S are discarded. However, some of these clauses may have a cover intersecting $X(U_{\mathcal{F}'})$; i.e. $X(U_{\mathcal{F}'}) \cap X(C_i) \neq \emptyset$ for some $i = k, ..., m$, or $X(U_{\mathcal{F}'}) \cap X(S) \neq \emptyset$. If one of these nogood clauses also maintains the path-like structure for $U_{\mathcal{F}'}$, then it could be used in place of the nogood clause to be generated by $obstacle(U_{\mathcal{F}})$ at the next iteration. The computational effort is then reduced at the next iteration, and this may diminish the overall effort required to solve the problem.

Such a strategy is beneficial if we have an easy way to identify the discarded nogood clauses that could potentially be used at the next iteration of the procedure. Now we introduce a partial criterion for this.

Recall that if $\mathcal{F}'$ is generated via Step 2.2 of the Resolution Search procedure, then there exists an index $k \leq m$ such that

$$\mathcal{F}' = [C_1, C_2, ..., C_{k-1}, R].$$

Then a markable predicate $\mu_R \in (R \setminus U_{\mathcal{F}_{k-1}})$ is selected to generate $\bar{R} = (R \setminus \{\mu_R\}) \cup \{\bar{\mu}_R\}$ and $U_{\mathcal{F}'} = U_{\mathcal{F}_{k-1}} \cup \bar{R}$. It follows that for any $i = k, ..., m$

$$X(U_{\mathcal{F}'}) \cap X(C_i) \neq \emptyset \Longrightarrow \mu_R \notin C_i$$

and

$$X(U_{\mathcal{F}'}) \cap X(S) \neq \emptyset \Longrightarrow \mu_R \notin S.$$

Therefore $\mu_R \notin C_i$ ($\mu_R \notin S$) is a necessary but not sufficient condition for C_i (S) to be potentially used at the next iteration. For this purpose, the corresponding nogood clauses can be included in a **short-term memory** $\mathcal{C}$ to be used in a modified version of the Resolution Search procedure.

This modified version includes the additional Step 2.2.3 after Step 2.2.2:

Step 2.2.3.
Store the clauses $C_i \in \mathcal{F}$, $i = k, ..., m$, and the clause S in the short term memory $\mathcal{C}$, if they do not contain μ_R.

Furthermore, Step 1 is then modified to take advantage of the short-term memory $\mathcal{C}$. One possible way to do so is as follows:

Step 1.
Step 1.1 - Recycle.
If $\mathcal{C}$ is empty, go to Step 1.3.
Otherwise, Let S be the first clause in $\mathcal{C}$, and remove S from $\mathcal{C}$.
Step 1.2 - Suitability.
If S verifies $X(U_{\mathcal{F}}) \cap X(S) \neq \emptyset$ and maintains the path-like structure for $U_{\mathcal{F}}$, go to Step 2, otherwise go back to step 1.1.
Step 1.3 - Exploration.
Let S be the clause generated by $obstacle(U_{\mathcal{F}})$.

Naturally, nothing prevents the implementation of a more **long-term memory**, which for example would store nogood clauses discarded from the short-term memory. Such long-term memories are successfully used in most competitive exact strategies for the k-SAT satisfiability problems [2]. Chvátal [4] already exposed the potential of reusing discarded nogood clauses in this manner, and also of maintaining a long-term memory. Here we merely extend those ideas from the context of 0-1 programming to a more general framework.

References

[1] C. Artigues, S. Demassey, P. Michelon, *An application of resolution search to the RCPSP*, ECCO XVII, Beyrouth, 24-26 June 2004.

[2] P. Beame, H. Kautz, A. Sabharwal, *Towards Understanding and Harnessing the Potential of Clause Learning*, Journal of Artificial Intelligence Research 22, 2004.

[3] S. Boussier, M. Vasquez, Y. Vimont, S. Hanafi, P. Michelon, *Solving the 01 Multidimensional Knapsack Problem with Resolution Search*, VI ALIO/EURO Worshop on Applied Combinatorial Optimization, Buenos Aires, Argentina, 2008.

[4] V. Chvátal, *Resolution Search*, Discrete Applied Mathematics 73, 1997.

[5] S. Hanafi, F. Glover, *Resolution Search and Dynamic Branch-and-Bound*, Journal of Combinatorial Optimization 6, 2002.

[6] M. Palpant, C. Artigues, C. Oliva, *MARS: an hybrid scheme based on Resolution Search and Constraint Programming for Constraint Satisfaction Problems*, Internal Report, LIA Avignon, 2004.

Appendix

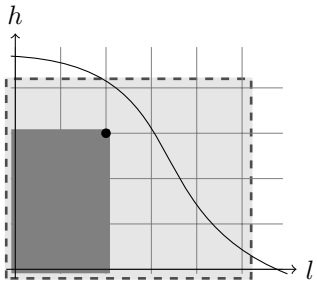
In this appendix we apply the Resolution Search procedure on our toy problem, using the *obstacle* procedure introduced before. The choice of marked predicates is arbitrary, and nogood clauses are not recycled.

The notational conventions, the symbols and and the colors are used as previously. For instance, in the figures, the cover of $U_{\mathcal{F}}$ is displayed in light gray, the reach of $\mathcal{F}$ is in medium gray, and the cover of S is in dark gray. When displaying $\mathcal{F}$, the clauses are stacked from top to bottom, and the marked predicates in each clause (i.e. the elements in $\mathcal{M}$) are underlined.

The search is initialized with the empty family $\mathcal{F} = \emptyset$, hence $m = 0$ and $U_{\mathcal{F}} = \emptyset$. Also, the upper bound $\bar{z}$ is set to $+\infty$.

Iteration 1.

To generate S, suppose that the point $(2, 3)$ is selected randomly by *obstacle* in $X(U_{\mathcal{F}}) = X(\emptyset) = \mathcal{X} = \{(l, h) \in \mathbb{Z}^2 : 0 \leq l \leq 5, 0 \leq h \leq 4\}$. This point lies below Γ, and thus *obstacle* generates $S = \{0 \leq l \leq 2, 0 \leq h \leq 3\}$. Also, since $f(2, 3) < +\infty$, we update $\bar{z} = -6$, and $r^* = (3, 2)$.



Search state at iteration 1.

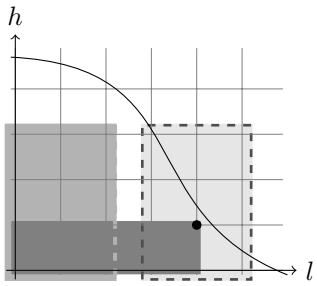
Since S contains markable predicates for $U_{\mathcal{F}}$, the family $\mathcal{F}'$ is generated by adding S to $\mathcal{F}$. The predicate $\mu_S = (0 \leq l \leq 2)$ is selected as the marked predicate for that clause. It follows that

$$\mathcal{F}' = \left[\ \{\underline{0 \leq l \leq 2}, 0 \leq h \leq 3\} \ \right]$$

and $U_{\mathcal{F}'} = \{3 \leq l \leq 5, 0 \leq h \leq 3\}$.

Iteration 2.

To generate S, suppose that the point $(4, 1)$ is selected randomly by *obstacle* in $X(U_{\mathcal{F}}) = \{(l, h) \in \mathbb{Z}^2 : 3 \leq l \leq 5, 0 \leq h \leq 3\}$. This point lies below Γ, and thus *obstacle* generates $S = \{0 \leq l \leq 4, 0 \leq h \leq 1\}$.



Search state at iteration 2.

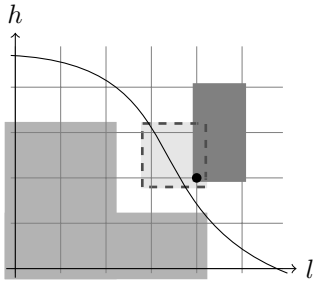
Since S contains markable predicates for $U_{\mathcal{F}}$, the family $\mathcal{F}'$ is generated by adding S to $\mathcal{F}$. The predicate $\mu_S = (0 \leq h \leq 1)$ is selected as the marked predicate for that clause. It follows that

$$\mathcal{F}' = \left[\begin{array}{l} \{\underline{0 \leq l \leq 2}, 0 \leq h \leq 3\} \\ \{0 \leq l \leq 4, \underline{0 \leq h \leq 1}\} \end{array} \right]$$

and $U_{\mathcal{F}'} = \{3 \leq l \leq 5, 0 \leq h \leq 3, 0 \leq l \leq 4, 2 \leq h \leq 4\}$.

Iteration 3.

To generate S, the procedure *obstacle* randomly selects $(4, 2)$ in $X(U_{\mathcal{F}}) = \{(l, h) \in \mathbb{Z}^2 : 3 \leq l \leq 4, 2 \leq h \leq 3\}$. This point lies above Γ, and thus *obstacle* generates $S = \{4 \leq l \leq 5, 2 \leq h \leq 4\}$.



Search state at iteration 3.

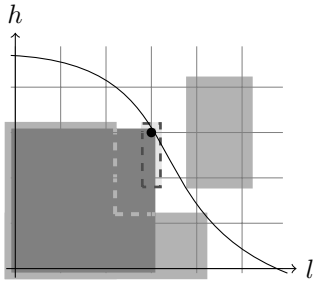
Since S contains a markable predicate for $U_{\mathcal{F}}$, the family $\mathcal{F}'$ is generated by adding S to $\mathcal{F}$. The predicate $\mu_S = (4 \leq l \leq 5)$ is selected as the marked predicate for that clause. It follows that

$$\mathcal{F}' = \left[\begin{array}{l} \{\underline{0 \leq l \leq 2}, 0 \leq h \leq 3\} \\ \{0 \leq l \leq 4, \underline{0 \leq h \leq 1}\} \\ \{\underline{4 \leq l \leq 5}, 2 \leq h \leq 4\} \end{array} \right]$$

and $U_{\mathcal{F}'} = \{3 \leq l \leq 5, 0 \leq h \leq 3, 0 \leq l \leq 4, 2 \leq h \leq 4, 0 \leq l \leq 3, 2 \leq h \leq 4\}$.

Iteration 4.

To generate S, the point $(3, 3)$ is selected randomly by *obstacle* in $X(U_{\mathcal{F}}) = \{(l, h) \in \mathbb{Z}^2 : 3 \leq l \leq 3, 2 \leq h \leq 3\}$. This point lies below Γ, and thus *obstacle* generates $S = \{0 \leq l \leq 3, 0 \leq h \leq 3\}$. Also, since $f(3, 3) < -6$, we update $\bar{z} = -9$, and $r^* = (3, 3)$.



Search state at iteration 4.

Since S contains no markable predicate for $U_{\mathcal{F}}$, we first generate R from the resolvents of S and the clauses in $\mathcal{F} = [C_1, C_2, C_3]$. Initiate the process with $R = S$ and $i = 3$.

$i = 3$**:** $\mu_3 = (4 \leq l \leq 5)$ implies that $\bar{\mu}_3 = (0 \leq l \leq 3)$, and this predicate is in R. Replace R with $R \nabla C_3 = (R \backslash \{\bar{\mu}_3\}) \cup (C_3 \backslash \{\mu_3\}) = \{0 \leq h \leq 3, 2 \leq h \leq 4\}$.

$i = 2$**:** $\mu_2 = (0 \leq h \leq 1)$ implies that $\bar{\mu}_2 = (2 \leq h \leq 4)$, and this predicate is in R. Replace R with $R \nabla C_2 = \{0 \leq h \leq 3, 0 \leq l \leq 4\}$.

$i = 1$**:** $\mu_1 = (0 \leq l \leq 2)$ implies that $\bar{\mu}_1 = (3 \leq l \leq 5)$, and this predicate is not in R.

The resulting nogood clause is $R = \{0 \leq h \leq 3, 0 \leq l \leq 4\}$.

Now determine the rank k such that R contains no markable predicate for $U_{\mathcal{F}_k}$. Initiate the process with $k = 0$ and $U_{\mathcal{F}_0} = \emptyset$.

$k = 0$**:** $(0 \leq h \leq 3) \in R$ is a markable predicate for $U_{\mathcal{F}_0} = \emptyset$.

$k = 1$**:** $(0 \leq l \leq 4) \in R$ is a markable predicate for $U_{\mathcal{F}_1} = U_{\mathcal{F}_0} \cup \bar{C}_1 = \{3 \leq l \leq 5, 0 \leq h \leq 3\}$.

$k = 2$**:** R contains no markable predicates for $U_{\mathcal{F}_2} = U_{\mathcal{F}_1} \cup \bar{C}_2 = \{3 \leq l \leq 5, 0 \leq h \leq 3, 0 \leq l \leq 4, 2 \leq h \leq 4\}$.

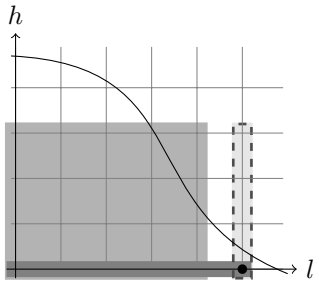
Thus $\mathcal{F}' = [C_1, R]$, and $\mu_R = (0 \leq l \leq 4)$ is selected as the marked predicate for the clause R. It follows that

$$\mathcal{F}' = \begin{bmatrix} \{\underline{0 \leq l \leq 2}, 0 \leq h \leq 3\} \\ \{\underline{0 \leq l \leq 4}, 0 \leq h \leq 3\} \end{bmatrix}$$

and $U_{\mathcal{F}'} = \{3 \leq l \leq 5, 0 \leq h \leq 3, 5 \leq l \leq 5, 0 \leq h \leq 3\}$.

Iteration 5.

To generate S, suppose that the point $(5, 0)$ is selected randomly by *obstacle* in $X(U_{\mathcal{F}}) = \{(l, h) \in \mathbb{Z}^2 : 5 \leq l \leq 5, 0 \leq h \leq 3\}$. This point lies below Γ, and thus *obstacle* generates $S = \{0 \leq l \leq 5, 0 \leq h \leq 0\}$.



Search state at iteration 5.

Since S contains a markable predicate for $U_{\mathcal{F}}$, the family $\mathcal{F}'$ is generated by adding S to $\mathcal{F}$. The predicate $\mu_S = (0 \leq h \leq 0)$ is selected as the marked predicate for that clause. It follows that

$$\mathcal{F}' = \begin{bmatrix} \{\underline{0 \leq l \leq 2}, 0 \leq h \leq 3\} \\ \{\underline{0 \leq l \leq 4}, 0 \leq h \leq 3\} \\ \{0 \leq l \leq 5, \underline{0 \leq h \leq 0}\} \end{bmatrix}$$

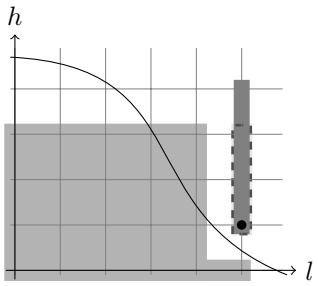
and $U_{\mathcal{F}'} = \{3 \leq l \leq 5, 0 \leq h \leq 3, 5 \leq l \leq 5, 0 \leq h \leq 3, 0 \leq l \leq 5, 1 \leq h \leq 4\}$.

Iteration 6.

To generate S, suppose that the point $(5, 1)$ is selected randomly by *obstacle* in $X(U_{\mathcal{F}}) = \{(l, h) \in \mathbb{Z}^2 : 5 \leq l \leq 5, 1 \leq h \leq 3\}$. This point lies above Γ, and thus *obstacle* generates $S = \{5 \leq l \leq 5, 1 \leq h \leq 4\}$.

Since S contains no markable predicate for $U_{\mathcal{F}}$, we first generate R from the resolvents of S and the clauses in $\mathcal{F} = [C_1, C_2, C_3]$. Initiate the process with $R = S$ and $i = 3$.

$i = 3$**:** $\mu_3 = (0 \leq h \leq 0)$ implies that $\bar{\mu}_3 = (1 \leq h \leq 4)$, and this predicate is in R. Replace R with $R \nabla C_3 = (R \setminus \{\bar{\mu}_3\}) \cup (C_3 \setminus \{\mu_3\}) = \{5 \leq l \leq 5, 0 \leq l \leq 5\}$.



Search state at iteration 6.

$i = 2$**:** $\mu_2 = (0 \le l \le 4)$ implies that $\bar{\mu}_1 = (5 \le l \le 5)$, and this predicate is in R. Replace R with $R \nabla C_2 = (R \setminus \{\bar{\mu}_2\}) \cup (C_2 \setminus \{\mu_2\}) = \{0 \le l \le 5, 0 \le h \le 3\}$.

$i = 1$**:** $\mu_1 = (0 \le l \le 2)$ implies that $\bar{\mu}_1 = (3 \le l \le 5)$, and this predicate is not in R.

The resulting nogood clause is $R = \{0 \le l \le 5, 0 \le h \le 3\}$.

Now determine the rank k such that R contains no markable predicate for $U_{\mathcal{F}_k}$. Initiate the process with $k = 0$ and $U_{\mathcal{F}_0} = \emptyset$.

$k = 0$**:** $(0 \le h \le 3) \in R$ is a markable predicate for $U_{\mathcal{F}_0} = \emptyset$.

$k = 1$**:** R contains no markable predicates for $U_{\mathcal{F}_1} = U_{\mathcal{F}_0} \cup \bar{C}_1 = \{3 \le l \le 5, 0 \le h \le 3\}$.

Thus $\mathcal{F}' = [R]$, and $\mu_R = (0 \le h \le 3)$ is selected as the marked predicate for the clause R. It follows that

$$\mathcal{F}' = \left[\ \{0 \le l \le 5, \underline{0 \le h \le 3}\} \ \right]$$

and $U_{\mathcal{F}'} = \{0 \le l \le 5, 4 \le h \le 4\}$.

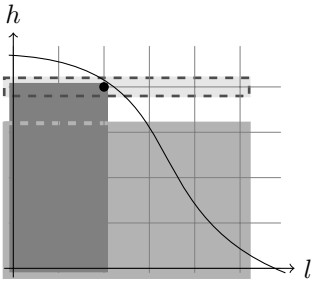
Iteration 7.

To generate S, suppose the point $(2, 4)$ is selected randomly by *obstacle* in $X(U_{\mathcal{F}}) = \{(l, h) \in \mathbb{Z}^2 : 0 \le l \le 5, 4 \le h \le 4\}$. This point lies below Γ, and thus *obstacle* generates $S = \{0 \le l \le 2, 0 \le h \le 4\}$.

Since S contains a markable predicate for $U_{\mathcal{F}}$, the family $\mathcal{F}'$ is generated by adding S to $\mathcal{F}$. The predicate $\mu_S = (0 \le l \le 2)$ is selected as the marked predicate for that clause. It follows that

$$\mathcal{F}' = \left[\begin{array}{l} \{0 \le l \le 5, \underline{0 \le h \le 3}\} \\ \{\underline{0 \le l \le 2}, 0 \le h \le 4\} \end{array} \right]$$

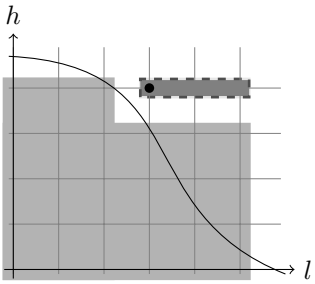
and $U_{\mathcal{F}'} = \{0 \le l \le 5, 4 \le h \le 4, 3 \le l \le 5, 0 \le h \le 4\}$.



Search state at iteration 7.

Iteration 8.

To generate S, suppose the point $(3, 4)$ is selected randomly by *obstacle* in $X(U_{\mathcal{F}}) = \{(l, h) \in \mathbb{Z}^2 : 3 \leq l \leq 5, 4 \leq h \leq 4\}$. This point lies above Γ, and thus *obstacle* generates $S = \{3 \leq l \leq 5, 4 \leq h \leq 4\}$.



Search state at iteration 8.

Since S contains no markable predicate for $U_{\mathcal{F}}$, we first generate R from the resolvents of S and the clauses in $\mathcal{F} = [C_1, C_2]$. Initiate the process with $R = S$ and $i = 2$.

$i = 2$**:** $\mu_2 = (0 \leq l \leq 2)$ implies that $\bar{\mu}_2 = (3 \leq l \leq 5)$, and this predicate is in R. Replace R with $R \nabla C_2 = (R \backslash \{\bar{\mu}_2\}) \cup (C_2 \backslash \{\mu_2\}) = \{4 \leq h \leq 4, 0 \leq h \leq 4\}$.

$i = 1$**:** $\mu_1 = (0 \leq h \leq 3)$ implies that $\bar{\mu}_1 = (4 \leq h \leq 4)$, and this predicate is in R. Replace R with $R \nabla C_1 = (R \setminus \{\bar{\mu}_1\}) \cup (C_1 \setminus \{\mu_1\}) = \{0 \leq h \leq 4, 0 \leq l \leq 5\}$.

The resulting nogood clause is $R = \{0 \leq h \leq 4, 0 \leq l \leq 5\}$.

Now determine the rank k such that R contains no markable predicate for $U_{\mathcal{F}_k}$. Initiate the process with $k = 0$ and $U_{\mathcal{F}_0} = \emptyset$.

$k = 0$**:** R contains no markable predicates for $U_{\mathcal{F}_0}$.

Thus $X(R) = \mathcal{X}$ and the search is completed. The optimum is $r^* = (3, 3)$ with value $\bar{z} = -9$.