Effective Planning with More Expressive Languages

Guillem Francès
Universitat Pompeu Fabra
Barcelona, Spain
guillem.frances@upf.edu

Hector Geffner
ICREA & Universitat Pompeu Fabra
Barcelona, Spain
hector.geffner@upf.edu

Abstract

Most of the key computational ideas in classical planning assume a simple planning language where action preconditions and goals are conjunctions of *propositional* atoms. This is to facilitate the definition and computation of heuristics for guiding the search for plans. In this work, however, we show that this modeling choice hides important structural information, resulting in poorer heuristics and weaker planning performance. To address this, we show how relaxed plan heuristics can be lifted to a variable-free first-order planning language, Functional STRIPS, where atomic formulas can involve arbitrary terms. The key idea is to regard the set of atoms that are reachable in a propositional layer of the relaxed planning graph as encoding a set of *logical first-order interpretations*. A precondition or goal *formula* is then regarded as reachable in a propositional layer, potentially adding new atoms to the next layer, when the set of atoms in the layer makes the formula *satisfiable* according to the rules of first-order logic. While this satisfiability test and the resulting heuristics turn out to be intractable, we show how a meaningful polynomial approximation can be obtained by formulating the satisfiability problem as a CSP and applying constraint propagation techniques. Experiments illustrating the computational value of planning with more expressive languages are also reported.

1 Introduction

Most heuristic search planners assume a propositional representation of the problem with conjunctive goals and action preconditions, from which heuristics can be derived effectively to guide the search for plans. One of such heuristics is based on the relaxed planning graph (RPG), where the propositional layer P_i contains those atoms that can be reached after i steps in the delete relaxation [Hoffmann and Nebel, 2001]. The first P_0 layer contains all atoms true in the seed state, while layer P_{i+1} contains all atoms from the previous P_i layer plus those that can be added by actions whose preconditions are in P_i. The index of the first layer that contains all the goals captures the $h_{\max}$ heuristic [Bonet and Geffner, 2001], and the number of actions in a relaxed plan obtained backwards from the goal defines the h_{FF} heuristic.

In this work, we show how RPG heuristics can be lifted to a variable-free first-order language like Functional STRIPS [Geffner, 2000], where preconditions and goals can feature arbitrary atoms. The key idea is very simple: the sets of atoms reachable in a propositional layer of the RPG are regarded as encoding a set of *logical first-order interpretations*. A precondition or goal *formula* is then regarded as *reachable* in a propositional layer, potentially triggering the addition of new atoms to the next layer, when some interpretation implicit in the layer makes the formula *satisfiable* according to the rules of first-order logic. While this satisfiability test and the resulting heuristics are worst-case intractable, in practice they are not always so, while polynomial approximations can be obtained by formulating the problem as a CSP and applying constraint propagation techniques. The *value-accumulating semantics* generalization of RPG heuristics to first-order languages [Gregory *et al.*, 2012; Domshlak and Nazarenko, 2013] is also polynomial, but by completely ignoring such constraints.

The derivation of heuristics from first-order languages is relevant because important structural information becomes hidden when propositional encodings are used, resulting in poorer heuristics and overall planning performance. To illustrate this, consider a simple problem involving integer variables $X_1, \ldots, X_n$, actions that increase or decrease the value of any variable by one within the $[0, n]$ interval, an initial situation where all variables equal 0, and a goal where the inequalities $X_i < X_{i+1}$ need to be achieved for $1 \leq i < n$. For such a problem, a single action suffices to achieve *any* of the goals $X_i < X_{i+1}$; this means that once the problem is encoded propositionally, the *joint goal* will be reachable in one step of the RPG. In the new heuristics, however, the joint goal will be reachable only after $n - 1$ steps, when $\bigwedge_{i=1}^{n-1} X_i < X_{i+1}$ becomes logically satisfiable by the interpretation where $X_i = i - 1$, for $i = 1..n$.

In the rest of the paper, we review the Functional STRIPS language and semantics, show how the RPG heuristics can be lifted, discuss modeling issues and report empirical results. Further details can be found in [Francès and Geffner, 2015].

2 The Functional STRIPS Language

Functional STRIPS (FSTRIPS, [Geffner, 2000]) is a classical planning modeling language based on the quantifier-free fragment of first-order-logic involving *constant*, *function* and *relational or predicate symbols*, but no variable symbols. For the sake of succinctness, in what follows we treat predicate symbols as function symbols of a special Boolean type.

Syntax. FSTRIPS makes a distinction between *fluent* symbols, whose denotation may change as a result of the actions, and *fixed* symbols, whose denotation does not change. Among the latter, there is usually a finite set of object names, plus a number of *built-in* constant, function, and predicate symbols such as '3', '+' and '=', with the standard interpretation. Terms, atoms, and formulas are defined from constant, function, and predicate symbols in the standard way, except that in order to obtain compact state representations, symbols, and hence terms, are typed. A *type* is given by a finite set of fixed constant symbols. The terms $f(t)$ where f is a fluent symbol and t is a tuple of fixed constant symbols are called *state variables*, and clearly there is a finite set of them.

An action a is described by its *precondition* and its set of *effects*. The precondition $Pre(a)$ is a formula, and the effects are *updates* of the form $f(t) := w$, where $f(t)$ and w are terms of the same type, f is a fluent symbol, and t is a tuple of terms. Updates express how fluent f changes when the action is taken. Conditional effects $C \rightarrow f(t) := w$, where C is a formula (possibly $\top$), can be defined in a similar manner.

As an example, the action of moving a block b onto another block b' can be expressed by an action $move(b, b')$ with precondition $clear(b) = true \wedge clear(b') = true$, and effects $loc(b) := b'$ and $clear(loc(b)) := true$. In this case, b is a constant symbol, and the terms $clear(b)$ and $loc(b)$ are state variables. The term $clear(loc(b))$ is not a state variable, as $loc(b)$ is not a fixed constant symbol. A FSTRIPS planning problem is a tuple $\langle F, I, O, G\rangle$, where I is a set of literals defining the initial situation, G is the goal formula, O is a set of actions, and F describes the symbols and their types.

Semantics. The dynamic part of a state is fully determined by the values of the state variables; a state represents a logical interpretation over the FSTRIPS language. The denotation of a symbol or term t in state s is written as t^s. The denotation r^s of fixed symbols r does not depend on the state and it is written r^*. The denotation of standard fixed symbols like '3', '+', '=' is assumed to be given by the underlying programming language, while object names c are assumed to denote themselves so that $c^* = c$. The denotation of fixed (typed) function and relational symbols can be provided extensionally, by enumeration in the initial situation, or intensionally, by attaching actual functions to them [Dornhege *et al.*, 2009].

From the fixed denotation r^* of fixed symbols r, and the changing denotation of fluent symbols f captured by the values $[f(t)]^s$ of the state variables $f(t)$ associated with f, the denotation of arbitrary terms, atoms, and formulas follows in the standard way. The denotation t^s of any term not involving functional fluents, expressed also as t^*, is c^* if t is a constant symbol or, recursively, $g^*(t_1^*)$ if t is the compound term $g(t_1)$ where t_1 is a tuple of terms. Similarly, the denotation of a term $f(t_1)$ where f is a fluent functional symbol is defined recursively as the value $[f(c)]^s$ of the *state variable* $f(c)$ in s where c is the tuple of constant symbols that name the tuple of objects t_1^s; i.e., $c^* = t_1^s$. In the same way, the denotation $[p(t)]^s$ of an atom $p(t)$ is $true/false$ iff the result of applying the Boolean function p^* to the tuple of objects t^s yields $true/false$. The truth value B^s of the formulas B made up of such atoms in the state s follows then the usual rules.

An action a is applicable in a state s if $[Pre(a)]^s = true$. The state s_a that results from applying a in s satisfies the equation $f^{s_a}(t^s) = w^s$ for all the updates $f(t) := w$ that the action a triggers in s, and is otherwise equal to s. This means that the update changes the value of the *state variable* $f(c)$ to w^s iff the action triggers an update $f(t) := w$ in the state s for which $c^* = t^s$. For example, if $X = 2$ is true in s, then the update $X := X + 1$ increases the value of X to 3 without affecting other state variables. Similarly, if $loc(b) = b'$ is true in s, the update $clear(loc(b)) := true$ in s is equivalent to the update $clear(b') := true$.

A plan for a problem $\langle F, I, O, G\rangle$ is a sequence of applicable actions from O that maps the unique initial state where I is true into one of the states where G is true.

The problem described in the introduction, COUNTERS, can be directly encoded in FSTRIPS with a fluent function symbol val such that $val(i)$ denotes the value of X_i. Thus, $val(1), \ldots, val(n)$ are the only state variables of the problem, and actions $increment(i)$ and $decrement(i)$ update their values. The goal is the conjunction $\bigwedge_{i=0}^{n-1} val(i) < val(i+1)$. The encoding of the GROUPING domain, where blocks are split into groups and must be placed in the same cell iff they are in the same group, is shown in Figure 1.

```
types:  block, cell, direction

functions:
  loc(b: block):  cell
  next(x: cell, d: direction): cell

action move(b: block, d: direction)
  prec in-grid(next(loc(b), d))
  eff  loc(b) := next(loc(b), d)

goal:
  loc(b1) ≠ loc(b2) ∧ loc(b1) ≠ loc(b3) ∧
  loc(b2) = loc(b3)
```

Figure 1: GROUPING domain: blocks must be moved until they occupy the same cell iff they belong to the same group.

3 Functional STRIPS Heuristics

In the standard propositional relaxed planning graph, the first propositional layer P_0 contains the atoms that are true in the seed state, and layer P_{i+1} contains all those atoms in P_i plus those atoms p for which all preconditions of some action that adds p are in P_i. The goal is deemed reachable in a layer if each goal atom is in the layer. The key difference in our first-order account of the RPG is how action preconditions and goal formulas are evaluated.

A layer P_k in the lifted RPG contains one domain X^k for each state variable X in the problem. The set of all domains X^k associated with layer P_k is taken to represent a set V^k of *possible logical interpretations over the language*; namely, the set of interpretations v that result from selecting for each state variable X *one of the values* $x \in X^k$. An interpretation v assigns a truth value to any formula in the language, and a formula is considered to be *satisfiable* or *reachable* in layer P_k iff the formula is satisfied by some interpretation v in V^k. For example, if X_1, X_2 and X_3 represent the three state variables of a problem and their domains in layer P_k are $X_i^k = \{1, 2, 3\}$, then there will be 9 interpretations in V^k, but just one that satisfies the formula $(X_1 < X_2) \wedge (X_2 < X_3)$.

Finally, if the goal formula is not satisfiable in layer P_k, a value c' is added to the domain X^{k+1} of state variable X in next layer P_{k+1} if $X = f(c)$, c and c' are fixed constant symbols, and there is an action a with effect $C \rightarrow f(t_1) := t_2$ and an interpretation v in V^k such that that v satisfies the formula $Pre(a) \wedge C \wedge t_1 = c \wedge t_2 = c'$.

The heuristics resulting from such a lifted RPG that are analogous to the $h_{\max}$ and h_{FF} heuristics are called $h^*_{\max}$ and h^*_{FF}. The two pairs of heuristics are very different. For example, in the COUNTERS problem involving n variables, the values of the former heuristics for the propositional problem encoding are 1 and $n - 1$, while the values of the latter are $n - 1$ and $n * (n - 1)/2$, which is actually the optimal heuristic value. The satisfiability tests make the lifted heuristics intractable in general. Yet under the standard restriction that action preconditions, conditions, and goals are *conjunctions* of atoms, the satisfiability tests can be easily mapped into a *constraint satisfaction problem* (CSP) [Dechter, 2003; Rossi *et al.*, 2006]. While solving these CSPs is hard in general (NP-complete), on average this is not so. In addition, general *constraint propagation* methods such as *arc-consistency* [Mackworth, 1977] and specialized methods arising from the use of *global constraints* [Rossi *et al.*, 2006; Van Hoeve and Katriel, 2006] can be used to approximate these satisfiability tests in polynomial time [Francès and Geffner, 2015]. The resulting polynomial heuristics that approximate $h^*_{\max}$ and h^*_{FF} are called $h^c_{\max}$ and h^c_{FF} respectively.

When all condition, precondition, and goal formulas are conjunctions of atoms where no state variable appears more than once and where there are no nested fluents, the $h^*_{\max}$ and h^*_{FF} heuristics, and their approximations $h^c_{\max}$ and h^c_{FF}, become equivalent to the $h_{\max}$ and h_{FF} heuristics according to the value-accumulating semantics [Ivankovic and Haslum, 2015]. Otherwise, the former capture non-unary constraints on state variables that are missed by the latter.

4 A Functional STRIPS Planner

The FS planner accepts any problem represented in Functional STRIPS, with the sole restriction that preconditions and goal formulas are conjunctions of literals. Conditional effects are not yet supported. FS can use any of the four lifted RPG heuristics, $h^*_{\max}$, h^*_{FF}, $h^c_{\max}$, and h^c_{FF}, to drive a plain greedy best first search. The construction of the RPG and computation of the heuristics is mapped into a number of CSPs which are handled by the standard Gecode CP solver [Gecode Team, 2006], except in simple cases where they are solved by a hand-coded procedure to avoid the overhead of interacting with Gecode. The integration with a CP solver further yields the possibility of using any of the large catalog of global constraints available in Gecode as externally-defined fixed predicate or function symbols. This helps representationally and computationally, as the computation of the lifted RPG benefits from efficient ad-hoc propagators. The goal of stacking all blocks in a single tower in BLOCKSWORLD, for instance, can be compactly encoded with the single goal atom $\mathit{alldiff}(loc(b_1), \ldots, loc(b_n))$. The encodings below use the standard $\mathit{alldiff}$ and sum constraints.

5 Experimental Results

We test the FS planner on some FSTRIPS domains using the h^c_{FF} heuristic and compare it to standard planners on equivalent PDDL models. To compare the h_{FF} and h^c_{FF} heuristics, we run FF [Hoffmann and Nebel, 2001] and Metric-FF [Hoffmann, 2003], the latter on encodings with numeric fluents if suitable, using the same greedy best-first search strategy (i.e. $f(n) = h(n)$ and EHC disabled). We also run the state-of-the-art Fast-Downward planner (LAMA-2011 configuration), that uses different search algorithms and exploits additional heuristic information from helpful actions and landmarks [Helmert, 2006; Richter and Westphal, 2010]. All planners run a maximum of 30 min. on a cluster with AMD Opteron 6300@2.4Ghz nodes, and are allowed up to 8GB of memory. FS's source code and all problem encodings are available on gfrances.github.io/pubs.

Domains. We consider four families of simple domains that illustrate the modeling and performance advantages of using Functional STRIPS with the FS planner. The GROUPING and COUNTERS domains, already introduced, show how the h^c_{FF} heuristic, although more expensive, pays off due to its increased accuracy. We test instances with increasing number of variables of three COUNTERS variations, labeled **COUNTERS-0**, **COUNTERS-RND** and **COUNTERS-INV**, where variables are initially set to 0, to random values, and to decreasing values, respectively. In **GROUPING** (Fig. 1), we test random instances with increasing grid sizes and numbers of blocks and groups. The type of goals in COUNTERS and GROUPING can be modeled in PDDL with existential variables, but the way standard planners compile these away produces an exponential blowup during preprocessing. We thus use alternative propositional PDDL encodings requiring additional actions and conditional effects.

SIMPLE-SOKOBAN is the classical Sokoban without obstacles. In this type of domain, delete-free heuristics produce relaxed plans that push all stones to the closer goal cell, even if several stones end up on the same cell. FSTRIPS, in contrast, allows us to express the (implicit) fact that stones must be placed in different cells with an extra goal atom $\mathit{alldiff}(loc(s_1), \ldots, loc(s_k))$. Other heuristic planners might also be able to model such a constraint indirectly, but not to use it to improve the heuristic as FS. We test a variation where all stones concentrate near a single goal cell (**SIMPLE-SOKOBAN**) plus another variation with random initial posi-

Domain	N	Coverage		Plan length			Node expansions			Time (s.)		
		FF	FS	FF	FS	R	FF	FS	R	FF	FS	R
COUNT-0	13	**13**	11	770.09	**270.09**	2.51	770.09	**270.09**	2.51	**33.98**	318.42	0.13
COUNT-I	13	8	**9**	946.14	**204.43**	4.99	946.14	**204.43**	4.99	**34.16**	272.11	0.65
COUNT-R	39	17	**28**	499.00	**87.35**	4.10	499.00	**88.76**	4.04	154.50	**25.68**	3.39
GROUP.	72	48	**55**	424.24	**43.86**	9.61	681.24	**104.79**	12.28	354.45	**83.12**	41.57
GARD.	51	20	**33**	366.85	**86.55**	4.10	2635.95	**456.45**	14.28	205.89	**7.84**	39.68
SOK.	17	5	**8**	65.40	**34.20**	1.43	404.60	**64.80**	3.18	**0.07**	3.98	0.01
SOK-R	81	**53**	34	121.29	**59.38**	1.67	2964.88	**624.82**	7.23	**3.38**	214.33	0.03

Table 1: **Summary of results** for FF and FS using a greedy best-first search with heuristics h_{FF} and h^c_{FF} (FF's EHC disabled). N is number of instances; length, node expansion and time figures are averaged over instances solved by both planners; R (for *ratio*) is the average of the per-instance FF / FS ratios. LAMA and Metric-FF results are discussed in the text.

tions (**SIMPLE-SOKOBAN-RND**). In **GARDENING**, an agent needs to water some plants with water loaded from a tap and poured into plants unit by unit. Standard delete-free heuristics are misleading in this context [Coles *et al.*, 2008], since in the delete relaxation one unit of water is enough to water all plants. When using FS, however, we can easily enforce a *flow constraint* so that the total amount of water obtained from the tap equals the total amount of water poured into the plants; for this we only need an extra goal atom $poured(p_1) + \cdots + poured(p_n) = total$ that maps into a sum global constraint. We test random instances with one water tap and increasing grid sizes and number of plants.

Results. Table 1 shows an overview of the results of FS and FF. Metric-FF and LAMA results are discussed later. As expected, h^c_{FF} *is more expensive than* h_{FF}: node expansion in FS is an order of magnitude slower in COUNTERS, 20% slower in GROUPING, and 1 to 3 orders of magnitude slower in SIMPLE-SOKOBAN. In GARDENING, remarkably, FS expands nodes twice as fast, presumably because of the sum global constraint propagator. At the same time, however, h^c_{FF} *is much more informed than* h_{FF}, as witnessed by the significantly smaller number of nodes expanded by FS in all domains. In GROUPING and GARDENING, particularly, FS expands on average an order of magnitude fewer nodes than FF. In COUNTERS, for instance, FF expands a higher number of nodes because, as predicted, each of the $X_i < X_{i+1}$ inequalities is conceived as *independent*, which guides the search towards heuristic plateaus. In GROUPING, h^c_{FF} is more informed, as it understands the constraints between goal atoms such as $loc(b_1) = loc(b_2)$ and $loc(b_2) \neq loc(b_3)$.

Overall, does the increased accuracy of h^c_{FF} compensate its higher cost? In terms of quality, *plan length is consistently shorter for FS*, showing that the heuristic is able to guide the search towards better solutions. In GROUPING, for instance, FS plans require on average an order of magnitude fewer block moves. More significantly, *FS coverage is higher in* 5 *of the* 7 *domains*, the exceptions being COUNTERS-0 and SIMPLE-SOKOBAN-RND, where plan length is however still significantly better. In SIMPLE-SOKOBAN-RND, the random distribution of stones makes it less likely that h_{FF} produces the previously-identified misleading relaxed plans. In SIMPLE-SOKOBAN, though, where stones are placed close together, FS is indeed able to heuristically exploit the *alldiff* constraint and scale up better than FF, with larger coverage and shorter plans. In three of the domains, the increased accuracy even results in a better total search time: in GARDENING, for instance, search is on average almost 40 times faster.

A full discussion of LAMA and Metric-FF results is omitted for space reasons, but they confirm the previous conclusions. In brief, Metric-FF does not scale up to large problems, nor takes advantage in the heuristic of constraints such as the GARDENING "flow constraint". LAMA offers a faster node expansion rate than FF, but plan length and total node expansions are still consistently larger than FS, resulting in a better coverage for FS in 5 of the 7 domains.

6 Conclusions

We have shown how to lift relaxed plan heuristics to a more expressive variable-free first-order planning language, Functional STRIPS, by regarding the atoms in a RPG layer as succinctly encoding a set of first-order logical interpretations, and then using the standard notion of satisfiability to determine when a formula is reachable in a layer. This *lifted RPG* is able to capture certain constraints that become otherwise lost, resulting in more informed heuristics when precondition or goal formulas feature different atoms referring to the same state variable. Although these heuristics are worst-case intractable, we have shown that they can be cost-effective in practice, and that polynomial approximations can be computed by using constraint propagation techniques.

We have implemented these ideas in FS, the first FSTRIPS planner, which additionally allows to define fixed functions and predicates through global constraints. The fact that the performance of FS is improved by adding redundant constraints is like in CSP and SAT, where solver performance can be improved by explicating implicit constraints. This distinguishes FS from the existing heuristic search planners that either make no room for explicit constraints or cannot use them in the computation of the heuristic. Recently, we have also extended FSTRIPS and FS to handle existential quantification in preconditions and goals without having to compile them away [Francès and Geffner, 2016], by associating such variables with CSP variables in the CSP model used for computing the FS heuristics. Furthermore, these heuristics can be computed without having to fully ground the action schemas.

Acknowledgments

We thank Miquel Ramírez for useful feedback. This work is partially funded by grants TIN2015-67959 and CSD2010-00034 from MEC, Spain.

References

[Bonet and Geffner, 2001] B. Bonet and H. Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1–2):5–33, 2001.

[Coles *et al.*, 2008] A. Coles, M. Fox, D. Long, and A. Smith. A hybrid relaxed planning graph-LP heuristic for numeric planning domains. In *Proc. 18th Int. Conf. on Automated Planning and Scheduling*, pages 52–59, 2008.

[Dechter, 2003] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[Domshlak and Nazarenko, 2013] C. Domshlak and A. Nazarenko. The complexity of optimal monotonic planning: the bad, the good, and the causal graph. *Journal of Artificial Intelligence Research*, pages 783–812, 2013.

[Dornhege *et al.*, 2009] C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel. Semantic attachments for domain-independent planning systems. In *Proc. 19th Int. Conf. on Automated Planning and Scheduling*, pages 114–121, 2009.

[Francès and Geffner, 2015] G. Francès and H. Geffner. Modeling and computation in planning: Better heuristics from more expressive languages. In *Proc. 25th Int. Conf. on Automated Planning and Scheduling*, pages 70–78, 2015.

[Francès and Geffner, 2016] G. Francès and H. Geffner. E-STRIPS: Existential quantification in planning and constraint satisfaction. In *Proc. 25th Int. Joint Conf. on Artificial Intelligence*, 2016.

[Gecode Team, 2006] Gecode Team. Gecode: Generic constraint development environment, 2006. Available from `http://www.gecode.org`.

[Geffner, 2000] H. Geffner. Functional STRIPS: A more flexible language for planning and problem solving. In J. Minker, editor, *Logic-Based Artificial Intelligence*, pages 187–205. Kluwer, 2000.

[Gregory *et al.*, 2012] P. Gregory, D. Long, M. Fox, and C. Beck. Planning modulo theories: Extending the planning paradigm. In *Proc. 22nd Int. Conf. on Automated Planning and Scheduling*, 2012.

[Helmert, 2006] M. Helmert. The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[Hoffmann and Nebel, 2001] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[Hoffmann, 2003] J. Hoffmann. The metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research*, 20:291–341, 2003.

[Ivankovic and Haslum, 2015] F. Ivankovic and P. Haslum. Optimal planning with axioms. In *Proc. 24th Int. Joint Conf. on Artificial Intelligence*, pages 1580–1586, 2015.

[Mackworth, 1977] A.K. Mackworth. Consistency in networks of relations. *Artificial intelligence*, 8(1):99–118, 1977.

[Richter and Westphal, 2010] S. Richter and M. Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39(1):127–177, 2010.

[Rossi *et al.*, 2006] F. Rossi, P. Van Beek, and T. Walsh. *Handbook of constraint programming*. Elsevier, 2006.

[Van Hoeve and Katriel, 2006] W. Van Hoeve and I. Katriel. Global constraints. *Handbook of constraint programming*, pages 169–208, 2006.