Optimization Modulo the Theories of Signed Bit-Vectors and Floating-Point Numbers *

Patrick Trentin and Roberto Sebastiani

DISI, University of Trento, Italy

Abstract. Optimization Modulo Theories (OMT) is an important extension of SMT which allows for finding models that optimize given objective functions, typically consisting in linear-arithmetic or pseudo-Boolean terms. However, many SMT and OMT applications, in particular from SW and HW verification, require handling bit-precise representations of numbers, which in SMT are handled by means of the theory of Bit-Vectors (BV) for the integers and that of Floating-Point Numbers (FP) for the reals respectively. Whereas an approach for OMT with (unsigned) BV has been proposed by Nadel & Ryvchin, unfortunately we are not aware of any existing approach for OMT with FP.

In this paper we fill this gap. We present a novel OMT approach, based on the novel concept of attractor and dynamic attractor, which extends the work of Nadel & Ryvchin to signed BV and, most importantly, to FP. We have implemented some OMT(BV) and OMT(FP) procedures on top of OptiMATHSAT and tested the latter ones on modified problems from the SMT-LIB repository.

The empirical results support the validity and feasibility of the novel approach.

1 Introduction

Optimization Modulo Theories (OMT) [34, 19, 35, 37, 21, 20, 30, 29, 8, 28, 38–40, 7, 31, 41, 5, 42, 22, 27, 6] is an important extension to Satisfiability Modulo Theories which allows for finding models that optimize one or more objectives, which typically consist in some linear-arithmetic or Pseudo-Boolean function application.

However, many SMT and OMT applications, in particular from SW and HW verification, require handling bit-precise representations of numbers, which in SMT are handled by means of the theory of Bit-Vectors (BV) for the integers and that of Floating-Point Numbers (FP) for the reals respectively. (For instance, during the verification process of a piece of software, one may look for the minimum/maximum value of some int [resp. float] parameter causing an SMT(BV) [resp. SMT(FP)] call to return SAT—which typically corresponds to the presence of some bug—so that to guarantee a safe range for such parameter.)

OMT for the theory of (unsigned) bit-vectors (OMT(BV)) was proposed by Nadel and Ryvchin [31], although a reduction to the problem to MaxSAT was already implemented in the SMT/OMT solver Z3 [9]. The work in [31] was based on the observation that OMT on unsigned BV can be seen as lexicographic optimization over the bits in

*We would like to thank the anonymous reviewers for their insightful comments and suggestions, and we thank Alberto Griggio for support with MATHSAT code.
the bitwise representation of the objective, ordered from the most-significant bit (MSB) to the least-significant bit (LSB).

In this paper we address—for the first time to the best of our knowledge—OMT for the theory of signed Bit-Vectors and, most importantly, for the theory of Floating-Point Arithmetic (OMT(FP)), by exploiting some properties of the two’s complement encoding for signed BV and of the IEEE 754-2008 encoding for FP respectively.

We start from introducing the notion of attractor, which represent (the bitwise encoding of) the target value for the objective which the optimization process aims at. This allows us for easily leverage the procedure of [31] to work with both signed and unsigned Bit-Vectors, by minimizing lexicographically the bitwise distance between the objective and the attractor, that is, by minimizing lexicographically the bitwise-xor between the objective and the attractor.

Unfortunately there is no such notion of (fixed) attractor for FP numbers, because the target value moves as long as the bits of the objective are updated from the MSB to the LSB, and the optimization process may have to change dynamically its aim, even at the opposite direction. (For instance, as soon as the minimization process realizes there is no solution with a negative value for the objective and thus sets its MSB to 0, the target value is switched from $-\infty$ to 0+, and the search switches direction, from the maximization of the exponent and the significand to their minimization.)

To cope with this fact, we introduce the notions of dynamic attractor and attractor trajectory, representing the dynamics of the moving target value, which are progressively updated as soon as the bits of the objective are updated from the MSB to the LSB. Based on these ideas, we present novel OMT(FP) procedures, which require at most $n + 2$, incremental calls to an SMT(FP) solver, $n$ being the number of bits in the representation of the objective. Notice that these procedures do not depend on the underlying SMT(FP) procedure used, provided the latter allows for accessing and setting the single bits of the objective.

We have implemented these OMT(BV) and OMT(FP) procedures on top of the OPTIMATHSAT OMT solver [42]. We have run an experimental evaluation of the OMT(FP) procedures on modified SMT(FP) problems from the SMT-LIB library. The empirical results support the validity and feasibility of the novel approach.

The rest of the paper is organized as follows. In §2 we provide the necessary background on BV and FP theories and reasoning. In §3 we provide the novel theoretical definitions and results. In §4 we describe our novel OMT(FP) procedures. In §5 we present the empirical evaluation. In §6 we conclude, hinting some future directions.

2 Background

We assume some basic knowledge on SAT and SMT and briefly introduce the reader to the Bit-Vector and Floating-Point theories.

**Bit-Vectors.** A bit is a Boolean variable that can be interpreted as 0 or 1. A Bit-Vector (BV) variable $v^{[n]}$ is a vector of $n$ bits, where $v[0]$ is the Most Significant Bit (MSB)
and \(v[n-1]\) is the Least Significant Bit (LSB).\(^1\) A BV constant of width \(n\) is an interpreted vector of \(n\) values in \(\{0,1\}\). We overline a bit value or a BV value to denote its complement (e.g., \([11010010]\) is \([00101101]\)). A BV variable/constant of width \(n\) can be unsigned, in which case its domain is \([0,2^n-1]\), or signed, which we assume to comply with the Two’s complement representation, so that its domain is \([-2^{(n-1)},2^{(n-1)}-1]\). Therefore, the vector \([11111111]\) can be interpreted either as the unsigned BV constant \(255\)\(^{[8]}\) or as the signed BV constant \(-1\)\(^{[8]}\). Following the SMT-LIB v2 standard \([3]\), we may also represent a BV constant in binary (e.g. \#b00110100\)) or in hexadecimal (e.g. \#x1C\)) form. A BV term is built from BV constants, variables and interpreted BV functions which represent standard RTL operators: word concatenation (e.g. \(3^{[8]} \circ x^{[8]}\)), sub-word selection (e.g. \(3^{[8]}[6:3]\))\(^{[4]}\)), modulo-n sum and multiplication (e.g. \(x^{[8]} +_8 y^{[8]}\) and \(x^{[8]} \cdot_8 y^{[8]}\)), bit-wise operators (like, e.g., and\(_n\), or\(_n\), xor\(_n\), nxor\(_n\), not\(_n\))\(^{[8]}\), left and right shift \(<_<n,>>_n\). A BV atom can be built by combining BV terms with interpreted predicates like \(\geq_n\), \(<_n\) (e.g. \(0^{[8]} \geq_8 x^{[8]}\)) and equality. We refer the reader to \([3, 24]\) for further details on the syntax and semantics of Bit-Vector theory.

There are two main techniques for BV satisfiability, the “eager” and the “lazy” approach, which are substantially complementary to one another \([25]\). In the eager approach, BV terms and constraints are encoded into SAT via bit-blasting \([23, 17, 16, 24, 33, 32]\). In the lazy approach, BV terms are not immediately expanded –so to avoid any scalability issue– and the BV solver is comprised by a layered set of techniques, each of which deals with a sub-portion of the BV theory \([15, 10, 18, 24]\).

**Floating-Point.** The theory of Floating-Point Numbers (FP), \([3, 36, 13]\), is based on the IEEE standard 754-2008 \([4]\) for floating-point arithmetic, restricted to the binary case. A FP sort is an indexed nullary sort identifier of the form \(_\text{FP} <\text{ebits} > <\text{sbits}>\)\(^{[6]}\) s.t. both ebits and sbits are positive integers greater than one, ebits defines the number of bits in the exponent and sbits defines the number of bits in the significand, including the hidden bit. A FP variable \(v[n]\) with sort \(_\text{FP} <\text{ebits} > <\text{sbits}>\) can be indifferently viewed as a vector of \(n = \text{ebits} + \text{sbits}\) bits, where \(v[0]\) is the Most Significant Bit (MSB) and \(v[n-1]\) is the Least Significant Bit (LSB), or as a triplet of Bit-Vectors \((\text{sign}, \text{exp}, \text{sig})\)\(^{[8]}\) s.t. sign is a BV of size 1, exp is a BV of size ebits and sig is a BV of size sbits – 1. A FP constant is a triplet of BV constants. Given a fixed floating-point sort, i.e. a pair \(\langle\text{ebits}, \text{sbits}\rangle\), the following FP constants are implicitly defined:

| value         | Symbol | BV Repr.                                      |
|---------------|--------|-----------------------------------------------|
| plus infinity | \(_ +\infty <\text{ebits} > <\text{sbits}>\) | \(\text{fp } \#b0 \#b1...1 \#b0...0\) |
| minus infinity| \(_ -\infty <\text{ebits} > <\text{sbits}>\) | \(\text{fp } \#b1 \#b1...1 \#b0...0\) |
| plus zero     | \(_ +0 <\text{ebits} > <\text{sbits}>\)     | \(\text{fp } \#b0 \#b0...0 \#b0...0\) |
| minus zero    | \(_ -0 <\text{ebits} > <\text{sbits}>\)     | \(\text{fp } \#b1 \#b0...0 \#b0...0\) |
| not-a-number  | \(_ \text{NaN} <\text{ebits} > <\text{sbits}>\) | \(\text{fp } \#t \#b1...1 \#s\) |

\(^{1}\) Although most often in the literature the indexes \(i\in [0,...,n-1]\) use to grow from the LSB to the MSB, in this paper we use the opposite notation because we always reason from the MSB down to the LSB, so that to much simplify the explanation.
where $\tau$ is either 0 or 1 and $s$ is a $BV$ which contains at least a 1.

Setting aside special $FP$ constants, the remaining $FP$ values can be classified to be either normal or subnormal (a.k.a. denormal) [4]. A $FP$ number is said to be subnormal when every bit in its exponent is equal to zero, and normal otherwise. The significand of a normal $FP$ number is always interpreted as if the leading binary digit is equal to 1, while for denormalized $FP$ values the leading binary digit is always 0. This allows for the representation of numbers that are closer to zero, although with reduced precision.

Example 1. Let $x$ be the normal $FP$ constant ($\_ FP \#b0 \#b1100 \#b0101000$), and $y$ be the subnormal $FP$ constant ($\_ FP \#b0 \#b0000 \#b0101000$), so that their corresponding sort is ($\_ FP <4> <8>$). Then, according to the semantics defined in the IEEE standard 754-2008 [4], the floating-point value of $x$ and $y$ in decimal notation is given by:

$$
\begin{align*}
x &= (-1)^0 \cdot 2^{12-7} \cdot \left(1 + \sum_{i=1}^{7} (x[4+i] \cdot 2^{-i})\right) = 1 \cdot 2^5 \cdot \left(1 + \frac{1}{22} + \frac{1}{24}\right) = 42 \\
y &= (-1)^0 \cdot 2^{10-7+1} \cdot \left(0 + \sum_{i=1}^{7} (y[4+i] \cdot 2^{-i})\right) = 1 \cdot 2^{-6} \cdot \left(\frac{1}{22} + \frac{1}{24}\right) = \frac{5}{2^{10}}.
\end{align*}
$$

The theory of $FP$ provides a variety of built-in floating-point operations as defined in the IEEE standard 754-2008. This includes binary arithmetic operations (e.g. $+,-,\cdot,\div$), basic unary operations (e.g. $abs, -$), binary comparison operations (e.g. $\leq, <, \neq, =, >, \geq$), the remainder operation, the square root operation and more. Importantly, arithmetic operations are performed as if with infinite precision, but the result is then rounded to the “nearest” representable $FP$ number according to the specified rounding mode. Five rounding modes are made available, as in [4].

The most common approach for $FP$-satisfiability is to encode $FP$ expressions into $BV$ formulas based on the circuits used to implement floating-point operations, using appropriate under- and over-approximation schemes—or a mixture of both— to improve performance [14, 44, 45, 43]. Then, the $BV$-Solver is used to deal with the $FP$ formula, using either the eager or the lazy $BV$ approach. An alternative approach, based on abstract interpretation, is presented in [11, 12, 26]. With this technique, called Abstract CDCL (ACDCL), the set of feasible solutions is over-approximated with floating-point intervals, so that intervals-based conflict analysis is performed to decide $FP$-satisfiability.

3 Theoretical Framework

We present our generalization of [31] to the case of signed/unsigned Bit-Vector Optimization, and then move on to deal with Floating-Point Optimization.

3.1 Bit-Vector Optimization

Without any loss of generality, we assume that every objective function $f(\ldots)$ is replaced by a variable $obj$ of the same type by conjoining “$obj = f(\ldots)$” to the input
formula. We use the symbol $n$ to denote the bit-width of obj, and $\text{obj}[i]$ to denote the $i$-th bit of obj, where $\text{obj}[0]$ and $\text{obj}[n-1]$ are the Most Significant Bit (MSB) and the Least Significant Bit (LSB) of obj respectively. We define the Bit-Vector Optimization problem as follows.

**Definition 1.** (OMT($\text{BV}$)). Let $\varphi$ be a SMT($\text{BV}$) formula and $\text{obj}$ be a signed or unsigned $\text{BV}$ variable occurring in $\varphi$. We call an Optimization Modulo $\text{BV}$ problem, OMT($\text{BV}$), the problem of finding a model $\mathcal{M}$ for $\varphi$ (if any) whose value of $\text{obj}$, denoted with $\min_{\text{obj}}(\varphi)$, is minimum wrt. the total order relation $\leq_n$ for signed $\text{BV}$s if $\text{obj}$ is signed, and the one for unsigned $\text{BV}$s otherwise. (The dual definition where we look for the maximum follows straightforwardly)

Hereafter, we generalize the unsigned $\text{BV}$ maximization procedures described in [31] to the case of signed and unsigned $\text{BV}$ optimization. To this extent, we introduce the novel notion of $\text{BV}$ attractor.

**Definition 2.** (Attractor, attractor equalities). When minimizing [resp. maximizing], we call attractor for $\text{obj}$ the smallest [resp. greatest] $\text{BV}$-value $\text{attr}$ of the sort of $\text{obj}$. We call vector of attractor equalities the vector $A$ s.t. $A[i] = (\text{obj}[i] = \text{attr}[i])$, $k \in [0..n-1]$.

**Example 2.** If $\text{obj}[8]$ is an unsigned $\text{BV}$ objective of width 8, then its corresponding attractor $\text{attr}$ is $0[8]$, i.e. $[00000000]$, when $\text{obj}[8]$ is minimized and it is $255[8]$, i.e. $[11111111]$, when $\text{obj}[8]$ is maximized. When $\text{obj}[8]$ is instead a signed $\text{BV}$ objective, following the two’s complement encoding, the corresponding $\text{attr}$ is $-128[8]$, i.e. $[10000000]$, for minimization and $127[8]$, i.e. $[01111111]$, for maximization. □

In essence, the attractor can be seen as the target value of the optimization search and therefore it can be used to determine the desired improvement direction and to guide the decisions taken by the optimization search. By construction, if a model $\mathcal{M}$ satisfies all equalities $A[i]$, then $\mathcal{M}(\text{obj}) = \text{attr}$.

More in general, if $\mathcal{M}$ is a model of $\varphi$, then the value of obj in $\mathcal{M}$, denoted with $\mathcal{M}(\text{obj})$, is given by

$$\tau(\text{obj}) = \sum_{i=0}^{i=n-1} (2^{n-1-i} \cdot \text{ITE}(\mathcal{M}(A[i]), \text{attr}[i], \overline{\text{attr}[i]}))$$

(1)

when obj is an unsigned $\text{BV}$ objective, and by

$$\tau(\text{obj}) = \sum_{i=1}^{i=n-1} (2^{n-1-i} \cdot \text{ITE}(\mathcal{M}(A[i]), \text{attr}[i], \overline{\text{attr}[i]}))$$

$$- (2^{n-1}) \cdot \text{ITE}(\mathcal{M}(A[0]), \text{attr}[0], \overline{\text{attr}[0]})$$

(2)

when obj is a signed $\text{BV}$ objective, using the two’s complement representation. The function $\text{ITE}$, appearing in both previous equations, returns $\text{attr}[i]$ if the attractor equality $A[i]$ is true in $\mathcal{M}$ and $\overline{\text{attr}[i]}$ otherwise.
We use the symbol $\mu_k$ to denote a generic (possibly partial) assignment which assigns at least the $k$ most-significant bits of obj. We use the symbol $\tau_k$ to denote an assignment to all and only the $k$ most-significant bits of obj. Given $i < k$, we denote by $\mu_k[i]$ [resp. $\tau_k[i]$] the value in $\{0,1\}$ assigned to $\text{obj}[i]$ by $\mu_k$ [resp. $\tau_k$]. Moreover, we use the expression $[\mu_k]$, where $i \leq k$ to denote the restriction of $\mu_k$ to all and only the $i$ most-significant bits of obj, $\text{obj}[0], \ldots, \text{obj}[i - 1]$. Given a model $M$ of $\varphi$ and a variable $v$, we denote by $M(v)$ the evaluation of $v$ in $M$. With a little abuse of notation, and when this does not cause ambiguities, we sometimes use an attractor equality $A[i] = \text{attr}[i]$ to denote the single-bit assignment $\text{obj}[i] := \text{attr}[i]$ and its negation $\neg A[i]$ to denote the assignment to the complement value $\text{obj}[i] := \text{attr}[i]$.

**Definition 3.** (lexicographic maximization) Consider an OMT instance $\langle \varphi, \text{obj} \rangle$ and the vector of attractor equalities $A$. We say that an assignment $\tau_n$ to obj lexicographically maximizes $A$ wrt. $\varphi$ iff, for every $k \in [0..n - 1]$,

\begin{itemize}
  \item $\tau_n[k] = \text{attr}[k]$ if $\varphi \land [\tau_n]_k \land A[k]$ is unsatisfiable,
  \item $\tau_n[k] = \text{attr}[k]$ otherwise.
\end{itemize}

where $A[k]$ is the attractor equality $(\text{obj}[k] = \text{attr}[k])$. (The dual definition of “lexicographically minimizes” comes by switching $\text{attr}[k]$ with $\neg \text{attr}[k]$.) Given a model $M$ for $\varphi$, we say that $M$ lexicographically maximizes $A$ wrt. $\varphi$ iff its restriction to obj lexicographically maximizes $A$ wrt. $\varphi$.

Starting from the MSB to the LSB, $\tau_n$ [resp. $M$] in Definition 3 assigns to each $\text{obj}[k]$ the value $\text{attr}[k]$ unless it is inconsistent wrt. $\varphi$ and the assignments to the previous $\text{obj}[i], i \in [0..k - 1]$. Notice that this corresponds to minimize [resp. maximize] the value $\sum_{k=0}^{n-1} 2^{n-1-k} \cdot (\text{obj}[k] \text{xor}_1 \text{attr}[k])$ [resp. $\sum_{k=0}^{n-1} 2^{n-1-k} \cdot (\text{obj}[k] \lnot \text{xor}_1 \text{attr}[k])$],

where $\text{xor}$ is the bitwise-xor operator and $\lnot \text{xor}$ is its complement— because $2^{n-1-i} > \sum_{k=i+1}^{n-1} 2^{n-1-k}$.

The following fact derives from the above definitions and the properties of two’s complement representation adopted by the SMT-LIB2 standard\(^2\) for signed $\mathcal{B}$\(^\mathcal{V}\).

**Theorem 1.** An optimal solution of an OMT($\mathcal{B}$\(^\mathcal{V}\)) problem $\langle \varphi, \text{obj} \rangle$ is any model $M$ of $\varphi$ which lexicographically maximizes the vector of attractor equalities $A$.

**Proof.** (We investigate the minimization case, since the maximization case is dual.)

In the case of minimization with unsigned $\mathcal{B}$\(^\mathcal{V}\), $\text{attr}$ is $[00\ldots00]$, so that the lexicographic optimization corresponds to minimize $\sum_{k=0}^{n-1} 2^{n-1-k} \cdot \text{obj}[k]$ which is the standard minimization for unsigned $\mathcal{B}$\(^\mathcal{V}\).

In the case of minimization with signed $\mathcal{B}$\(^\mathcal{V}\), $\text{attr}$ is $[10\ldots00]$, so that the lexicographic optimization corresponds to minimize $2^{n-1} \cdot \overline{\text{obj}[0]} + \sum_{k=1}^{n-1} 2^{n-1-k} \cdot \text{obj}[k]$ which —by means of subtracting the constant value $2^{n-1}$— is equivalent to minimize $-2^{n-1} \cdot \overline{\text{obj}[0]} + \sum_{k=1}^{n-1} 2^{n-1-k} \cdot \text{obj}[k]$, which is the standard minimization for two’s complement $\mathcal{B}$\(^\mathcal{V}\).

\(^2\)If the standard adopted were the sign-and-magnitude binary encoding, then Theorem 1 would not hold. Nevertheless, in such a case we could adopt a simplified version of the technique for $F$\(^\mathcal{P}\) optimization described in §3.2.
Definitions 2 and 3 with Theorem 1 suggest thus a direct extension to the minimization/maximization of signed $BV$ of the algorithm for unsigned $BV$ in [31]: apply the unsigned-$BV$ maximization [resp. minimization] algorithm of [31] to the objective $\text{obj'} \overset{\text{def}}{=} (\text{obj } \text{nxor}_n \text{ attr})$ [resp. $\text{obj'} \overset{\text{def}}{=} (\text{obj } \text{xor}_n \text{ attr})$] instead than simply to $\text{obj}$ [resp. $\overline{\text{obj}}$].

**Example 3.** Let $\text{obj}[^3]$ be a signed $BV$ goal of 3 bits to be minimized and $\text{attr} \overset{\text{def}}{=} [100]$ be its attractor, so that the corresponding vector of attractor equalities $A$ is equal to $[\text{obj}[0] = 1, \text{obj}[1] = 0, \text{obj}[2] = 0]$.

An assignment $\tau_3 \overset{\text{def}}{=} \{A[0], \neg A[1], \neg A[2]\}$ (for which $\text{obj}[^3] = \overline{1[^3]}$) is lexicographically better than $\tau'_3 \overset{\text{def}}{=} \{\neg A[0], A[1], A[2]\}$ (for which $\text{obj}[^3] = 0[^3]$), because the former satisfies the *attractor equality* corresponding to the MSB while the latter does not. Moreover, the assignment $\tau''_3 \overset{\text{def}}{=} \{A[0], \neg A[1], A[2]\}$ (for which $\text{obj}[^3] = 2[^3]$), because —all the rest being equal— the latter assignment makes the *attractor equality* ($\text{obj}[2] = 0$) true.

\section*{3.2 Floating-Point Optimization}

We define the *Floating-Point Optimization problem* as follows.

**Definition 4.** (OMT($FP$)). Let $\varphi$ be a SMT($FP$) formula and $\text{obj}$ be a $FP$ variable occurring in $\varphi$. We call an **Optimization Modulo $FP$ problem**, the problem of finding a model $M$ for $\varphi$ (if any) whose value of $\text{obj}$, denoted with $\min_{\text{obj}}(\varphi)$, is either

- minimum wrt. the usual total order relation $\leq$ for $FP$ numbers, if $\varphi$ is satisfied by at least one model $M'$ s.t. $M'(\text{obj})$ is not $\text{NAN}$,
- some binary representation of $\text{NAN}$, otherwise.

(The dual definition where we look for the maximum follows straightforwardly.)

Definition 4 is made necessarily convoluted by the fact that $\text{obj}$ can be $\text{NAN}$. In fact, in the SMT-LIBv2 standard the comparisons $\{\leq, <, \geq, >\}$ between $\text{NAN}$ and any other $FP$ value are always evaluated false because $\text{NAN}$ has multiple representations at the binary level (see Table 1). Also, requiring the optimal solution to be always different from $\text{NAN}$ makes the resulting OMT($FP$) problem $\langle \varphi \land \neg \text{IsNAN}(\text{obj}), \text{obj} \rangle$ unsatisfiable when $\varphi$ is satisfied only by models $M$ s.t. $M(\text{obj})$ is $\text{NAN}$. For these reasons, we admit $\text{NAN}$ as the optimal solution value for $\text{obj}$ if and only if $\varphi$ is satisfied only by models $M$ s.t. $M(\text{obj})$ is $\text{NAN}$.

In the rest of this section we assume that we have already checked, in sequence, that

1. the input formula $\varphi$ is satisfiable —by invoking an SMT($FP$) solver on $\varphi$. If the solver returns $\text{UNSAT}$, then there is no need to proceed;
2. $\varphi$ is satisfied by at least one model $M'$ s.t. $M'(\text{obj})$ is not $\text{NAN}$ —by invoking an SMT($FP$) solver on $\varphi \land \neg \text{IsNAN}(\text{obj})$ if the model $M$ returned by the previous SMT call is s.t. $M(\text{obj})$ is $\text{NAN}$. If the solver returns $\text{UNSAT}$, then we conclude that the minimum is $\text{NAN}$.
After that, we can safely focus our investigation on the restricted OMT($\mathcal{FP}$) problem $\langle \varphi_{\text{noNaN}}, \text{obj} \rangle$, where $\varphi_{\text{noNaN}} \overset{\text{def}}{=} \varphi \land \neg \text{isNaN}(\text{obj})$, knowing it is satisfiable.

In Section §3.1, we have introduced the concept of a $\mathcal{BV}$ objective attractor, and we have shown how this value can be used to drive the optimization search towards the optimum value, when minimizing or maximizing a signed or unsigned $\mathcal{BV}$ goal. However, in the case of floating-point optimization, it is not possible to statically determine the attractor value in advance, before the search is even started. This is due to the more complex representation of $\mathcal{FP}$ variables, which uses three separate Bit-Vectors (i.e. sign, exponent and significand), and the presence of various classes of special values (i.e. zeros, infinity, NaN), which make definition 2 ambiguous for $\mathcal{FP}$ optimization. We illustrate this problem with the following example.

Example 4. Let $\langle \varphi_{\text{noNaN}}, \text{obj} \rangle$ be an OMT($\mathcal{FP}$) problem where obj is a $\mathcal{FP}$ objective, of sort $(_\mathcal{FP} \ 3 \ 5)$, to be minimized. To make our explanation easier to follow, we show in Table 1 a short list of sample values for an $\mathcal{FP}$ variable of the same sort as obj. Each $\mathcal{FP}$ value is represented as a triplet of Bit-Vectors $\langle \text{sign}, \text{exp}, \text{sig} \rangle$—following the SMT-LIB v2 conventions described in Section §2—and also in decimal notation.

| sign | exp | sig | value           |
|------|-----|-----|-----------------|
| #b0 | #b111 #b1111 | NAN |                |
|      | ... | ... | NAN             |
| #b0 | #b111 #b0000 | $+\infty$ | $\frac{31}{2}$ |
|      | ... | ... | ...             |
| #b0 | #b000 #b0001 | $\frac{1}{\pi}$ | 0              |
| #b0 | #b000 #b0000 | 0               |                |
| #b1 | #b000 #b0000 | $-\frac{1}{\pi}$ |                |
|      | ... | ... | ...             |
| #b1 | #b111 #b1111 | $-\frac{31}{2}$ |                |
| #b1 | #b111 #b0000 | $-\infty$ |                |
|      | ... | ... | NAN             |
| #b1 | #b111 #b1111 | NAN |                |

Table 1. Sample values for a $\mathcal{FP}$ variable with sort $(_\mathcal{FP} \ 3 \ 5)$.

From Table 1, we immediately notice that the binary representation of both the exponent and the significant of a Floating-Point number grows in opposite directions in the positive and in the negative domains. In addition, by sorting the values according to their binary representation, we observe that $-\infty$ [resp. $+\infty$] is not the smallest [resp. greatest] representable $\mathcal{FP}$ value in the negative [resp. positive] domain. In fact, both extreme ends of the table are occupied by NAN, which has multiple binary representations.

In what follows, we temporarily disregard the effects of unit-propagation, which might assign some (or all) bits of obj as a result of some constraints in $\varphi_{\text{noNaN}}$, and pick some values as candidate attractors for an $\mathcal{FP}$ goal to be minimized.
Suppose that the attractor is chosen to be equal to the value $-\infty$ listed at row 9 in Table 1, which is the smallest $FP$ value wrt. total order relation $\leq$ for $FP$ numbers. Assume that the optimal value of the $FP$ goal is the sub-normal $FP$ value ($fp\ #b1\ #b000\ #b1111$) (i.e. $-\frac{15}{64}$). Then, it can be seen that after both the sign and the exponent bits have been decided to be equal $\#b1$ and $\#b000$ respectively, the remaining bits of the attractor pull the search in the wrong direction, that is, towards $-0$.

Selecting a different $FP$ value as candidate attractor does not really solve the problem, or rather, it results in a different set of issues.

For instance, an attractor equal to the NAN value listed at row 10 in Table 1, which is the smallest representable $FP$ value according to the binary ordering, would solve the problem for the previous case in which the optimum $FP$ value is ($fp\ #b1\ #b000\ #b1111$). However, this attractor would remain an unsuitable choice for an OMT($FP$) instance where the $FP$ goal is forced to be positive, because after the sign bit of the objective function has been decided to be equal $\#b0$ the remaining bits of the attractor drive the search in the wrong direction, that is, towards $+\infty$. ⊥

Since there is no statically-determined $FP$ value that can be used as an attractor when dealing with floating-point optimization, we introduce the new concept of dynamic attractor.

**Definition 5.** *(Dynamic Attractor.)* Let $\langle \varphi_{noNaN}, \text{obj} \rangle$ be a restricted OMT($FP$) problem, where $\varphi_{noNaN} \defeq \varphi \land \neg\text{IsNaN(obj)}$ is a satisfiable SMT($FP$) formula and $\text{obj}$ is a $FP$ objective to be minimized [resp. maximized]. Let $k \in [0..n]$ and $\tau_k$ be an assignment to the $k$ most-significant bits of $\text{obj}$.

Then, we say that an $FP$-value $\text{attr}_{\tau_k}$ for $\text{obj}$ is a **dynamic attractor for** $\text{obj}$ wrt. $\tau_k$ iff it is the smallest [resp. largest] $FP$ value different from NAN s.t. the $k$ most-significant bits of $\text{attr}_{\tau_k}$ have the same value of the $k$ most-significant bits of $\text{obj}$ in $\tau_k$. We call vector of attractor equalities the vector $A_{\tau_k}$ s.t. $A_{\tau_k}[i] \defeq (\text{obj}[i] = \text{attr}_{\tau_k}[i])$, $i \in [0..n-1]$.

The following fact derives from the above definitions and the properties of IEEE 754-2008 standard representation adopted by SMT-LIBv2 standard for $FP$.

**Lemma 1.** Let $\langle \varphi_{noNaN}, \text{obj} \rangle$ be a restricted minimization [resp. maximization] OMT($FP$) problem, let $\tau_k$ be an assignment to $\text{obj}[0]...\text{obj}[k-1]$ and $\text{attr}_{\tau_k}$ be its corresponding dynamic attractor, for some $k \in [0..n-1]$. Let $\tau_{k+1} \defeq \tau_k \cup \{\text{obj}[k] := \text{attr}_{\tau_k}[k]\}$ and $\tau'_{k+1} \defeq \tau_k \cup \{\text{obj}[k] := \text{attr}_{\tau_k}[k]\}$, and let $\mathcal{M}$, $\mathcal{M}'$ two models for $\varphi_{noNaN}$ which extend $\tau_{k+1}$ and $\tau'_{k+1}$ respectively.

Then $\mathcal{M}(\text{obj}) \leq \mathcal{M}'(\text{obj})$ [resp. $\mathcal{M}(\text{obj}) \geq \mathcal{M}'(\text{obj})$].

**Proof.** (We prove the case of minimization, since that of maximization is dual wrt. the value of the sign bit.) We distinguish three cases based on the value of $k$.

**Case** $k = 0$ (**sign bit**). Then $	ext{attr}_{\tau_0}[0] = 1$, $\tau_1 = \{\text{obj}[0] = 1\}$ and $\tau'_1 = \{\text{obj}[0] = 0\}$, where $\text{obj}[0]$ is the MSB of $\text{obj}$ and represents the sign of the floating-point value. Then $\text{obj}$ is smaller or equal zero in every model $\mathcal{M}$ and larger or equal zero in every model $\mathcal{M}'$ of $\varphi_{noNaN}$, so that $\mathcal{M}(\text{obj}) \leq \mathcal{M}'(\text{obj})$ is verified.
Case \( k \in [1..ebits] \) (exponent bits), where \( ebits \) is the number of bits in the exponent of \( \text{obj} \). Then, \( \text{attr}_{\tau_k}[k] \) is 1 if \( \tau_k[0] = 1 \) and 0 otherwise.

In the first case, \( \text{obj} \) can only be negative-valued in both \( \mathcal{M} \) and \( \mathcal{M}' \). More precisely, \( \mathcal{M}(\text{obj}) \) can be either \(-\infty\) or a normal negative value, whereas \( \mathcal{M}'(\text{obj}) \) can be either a normal or a sub-normal negative value. Hereafter, we consider only the case in which both have a normal negative value, because the case in which \( \mathcal{M}(\text{obj}) = -\infty \) or \( \mathcal{M}'(\text{obj}) \) is sub-normal are both trivial, given that the absolute value of any sub-normal \( \mathbb{FP} \) number is smaller than the absolute value of any normal \( \mathbb{FP} \) number. Furthermore, we disregard the significand bits in \( \mathcal{M} \) and \( \mathcal{M}' \) because their contribution to the value of \( \text{obj} \) is always less significant than that of the bits in the exponent. Given these premises, the exponent value of \( \text{obj} \) in every possible \( \mathcal{M} \) is larger than the exponent of \( \text{obj} \) in every possible \( \mathcal{M}' \) by a value equal to \( 2^{ebits-k} \) and therefore, given that both \( \mathcal{M}(\text{obj}) \) and \( \mathcal{M}'(\text{obj}) \) are negative-valued, \( \mathcal{M}(\text{obj}) \leq \mathcal{M}'(\text{obj}) \).

The case in which \( \tau_k[0] = 0 \), that is when \( \text{obj} \) can only be positive-valued in both \( \mathcal{M} \) and \( \mathcal{M}' \), is dual.

Case \( k > ebits \) (significand bits). Then there are three sub-cases.

If for every \( i \in [1..ebits] \) the value of \( \tau_k[i] \) is equal 1, then the only possible value of \( \mathcal{M}(\text{obj}) \) for every possible \( \mathcal{M} \) is +\( \infty \), and therefore \( \text{attr}_{\tau_k}[k] = 0 \). On the other hand, there exists no possible model \( \mathcal{M}' \) of \( \varphi_{\text{noNaN}} \), because the assignment \( \text{obj}[k] = 1 \) would imply \( \text{obj} \) being equal to \( \text{NAN} \), so the statement \( \mathcal{M}(\text{obj}) \leq \mathcal{M}'(\text{obj}) \) is vacuously true.

If instead there is some \( i \in [1..ebits] \) s.t. \( \tau_k[i] = 0 \), then \( \text{attr}_{\tau_k}[k] = 1 \) if \( \tau_k[0] = 1 \) (i.e. \( \text{obj} \) is negative-valued) and 0 otherwise (i.e. \( \text{obj} \) is positive-valued). In both cases, we can disregard the exponent bits in \( \mathcal{M} \) and \( \mathcal{M}' \) because their contribution to the value of \( \text{obj} \) is the same in either model. For the same reasons, since \( \mathcal{M}(\text{obj}) \) and \( \mathcal{M}'(\text{obj}) \) can only be either both normal or both sub-normal, we can ignore the contribution of the leading hidden bit and focus on the bits of the significand.

When \( \tau_k[0] = 1 \) and \( \text{obj} \) must be negative-valued, the decimal value of the significand in \( \mathcal{M} \) is larger than the decimal value of every possible significand in \( \mathcal{M}' \) by exactly \( 2^{-k-ebits} \). Given that both \( \mathcal{M}(\text{obj}) \) and \( \mathcal{M}'(\text{obj}) \) are negative-valued, we have that \( \mathcal{M}(\text{obj}) \leq \mathcal{M}'(\text{obj}) \).

The case in which \( \tau_k[0] = 0 \), that is when \( \text{obj} \) can only be positive-valued in both \( \mathcal{M} \) and \( \mathcal{M}' \), is dual.

\( \Box \)

Lemma 1 states that, given the current assignment \( \tau_k \) to the \( k \) most-significant-bits of \( \text{obj}, \text{obj}[k] = \text{attr}_{\tau_k}[k] \) is always the best extension of \( \tau_k \) to the next bit (when consistent). A dynamic attractor \( \text{attr}_{\tau_k} \) can thus be used by the optimization search to guide the assignment of the \( k+1 \)-th bit of \( \text{obj} \) towards the direction of maximum gain which is allowed by \( \tau_k \), so that to obtain the “best” extension \( \tau_{k+1} \) of \( \tau_k \). Once the (new) assignment \( \tau_{k+1} \) is found, the OMT solver can compute the dynamic attractor \( \text{attr}_{\tau_{k+1}} \) for \( \text{obj} \) wrt. \( \tau_{k+1} \) and then use it to assign the \( k+2 \)-th bit of \( \text{obj} \), and so on.

Let \( \langle \varphi_{\text{noNaN}}, \text{obj} \rangle \) be an OMT(\( \mathbb{FP} \)) instance, s.t. \( \text{obj} \) is a \( \mathbb{FP} \) variable of \( n \) bits, and \( \tau_0 \) be an initially empty assignment. If at each step of the optimization search the assignment of the \( k \)-th bit of \( \text{obj} \) is guided by the dynamic attractor for \( \text{obj} \) wrt. \( \tau_k \), then the corresponding sequence of \( n \) dynamic attractors (of increasing order \( k \)) is unique and depends exclusively on \( \varphi_{\text{noNaN}} \). Intuitively, this is the case because the (current)
dynamic attractor always points in the direction of maximum gain. We illustrate this in the following example.

**Example 5.** Let \(<\varphi_{\text{noNaN}}, \text{obj}>\) be an OMT(\(\mathcal{FP}\)) problem where \text{obj} is a \(\mathcal{FP}\) objective, of sort \(_{\text{FP}}\mathcal{FP}\ 3\ 5\), to be minimized, as in Example 4. At the beginning of the search, nothing is known about the structure of the solution. Therefore, \(\tau_0 = \emptyset\) and, since \text{obj} is being minimized, the dynamic attractor for \text{obj} wrt. \(\tau_0\) (i.e. \(\text{attr}_{\tau_0}\)) is equal to \((\mathcal{FP} \#b1 \#b11 \#b0000)\) (i.e. \(-\infty\)), which gives a preference to any feasible value of \text{obj} in the negative domain.

If at some point of the optimization search we discover that the domain of the objective function can only be positive, so that the first bit of \text{obj} is permanently set to 0 in \(\tau_1\), then the new dynamic attractor for \text{obj} wrt. \(\tau_1\) (i.e. \(\text{attr}_{\tau_1}\)) is equal to \((\mathcal{FP} \#b0 \#b0000 \#b0000)\) (i.e. \(+0\)).

Furthermore, if later on we also find out that at least one bit in the exponent of \text{obj} can be assigned to 0 in a feasible solution of the problem that extends \(\tau_i\), for some \(i\), then we can remove \(+\infty\) from the optimization search interval.

\[\Diamond\]

**Definition 6.** (Attractor Trajectory \(\mathcal{A}_{\varphi}\)). Consider the restricted OMT(\(\mathcal{FP}\)) problem \(<\varphi_{\text{noNaN}}, \text{obj}>\) s.t. \(\varphi_{\text{noNaN}} \equiv \varphi \wedge \neg\text{lsNaN}(\text{obj})\) as in Definition 5, a triplet of inductively-defined sequences \(\langle \{\tau_0, \tau_1, ... , \tau_n\}, \{\text{attr}_{\tau_0}, \text{attr}_{\tau_1}, ... , \text{attr}_{\tau_n}\}, \{A_{\tau_0}, A_{\tau_1}, ..., A_{\tau_n}\} \rangle\) — where each \(\tau_k\) is an assignment to the first \(k\) most-significant bits of \text{obj} s.t. \(\tau_k \subseteq \tau_{k+1}\), \(\text{attr}_{\tau_k}\) is its corresponding dynamic attractor and \(A_{\tau_k}\) is its corresponding vector of attractor equalities — so that, for every \(k \in [0...n-1]\):

1. \(\tau_{k+1}[k] = \text{attr}_{\tau_k}[k]\) if \(\varphi_{\text{noNaN}} \wedge \tau_k \wedge A_{\tau_k}[k]\) is unsatisfiable,
2. \(\tau_{k+1}[k] = \text{attr}_{\tau_k}[k]\) otherwise.

Then we define the attractor trajectory \(\mathcal{A}_{\varphi}\) as the vector \([A_{\tau_0}[0], ..., A_{\tau_{n-1}}[n-1]]\).

The attractor trajectory \(\mathcal{A}_{\varphi}\) contains those attractor equalities \(\langle \text{obj}[k] = \text{attr}_{\tau_k}[k]\rangle\) which are of critical importance for the decisions taken by the optimization search. Intuitively, this is the case because the value of the \(k\)-th bit of \text{obj} (i.e. \text{obj}[k]) is still undecided in \(\tau_k\).

**Example 6.** Let \(<\varphi_{\text{noNaN}}, \text{obj}>\) be a restricted OMT(\(\mathcal{FP}\)) problem where \text{obj} is a \(\mathcal{FP}\) objective, of sort \(_{\text{FP}}\mathcal{FP}\ 3\ 5\), to be minimized, as in Example 4. We consider the case in which the input formula \(\varphi_{\text{noNaN}}\) requires \text{obj} to be larger or equal 2\(9/2\) and it does not impose any other constraint on the value of \text{obj}. Given the sequence of (partial) assignments \(\tau_0, ..., \tau_k\) in Figure 1, the corresponding list of dynamic attractors and the corresponding vectors of attractor equalities, then the attractor trajectory \(\mathcal{A}_{\varphi}\) is equal to the vector \([\text{obj}[0] = 1, \text{obj}[1] = 0, \text{obj}[2] = 0, \text{obj}[3] = 0, \text{obj}[4] = 0, \text{obj}[5] = 0, \text{obj}[6] = 0, \text{obj}[7] = 0]\).

\[\Diamond\]

**Lemma 2.** Consider \(<\varphi_{\text{noNaN}}, \text{obj}>\), \(\tau_0, ..., \tau_n, \text{attr}_{\tau_0}, ..., \text{attr}_{\tau_n}, A_{\tau_0}, ..., A_{\tau_n}\), and \(\mathcal{A}_{\varphi}\) as in definition 6. Then \(\tau_n\) lexicographically maximizes \(A_{\varphi}\) wrt. \(\varphi_{\text{noNaN}}\).

**Proof.** By Definition 6, we have that, for each \(k \in [0...n-1]\),
Fig. 1. An example of $\mathcal{FP}$ optimization using the dynamic attractor. ("$\Rightarrow$ SAT/UNSAT" denotes the satisfiability of $\varphi_{\text{noNaN}} \land \mathbf{\tau}_k \land A_{\mathbf{\tau}_k}[k]$, the symbols "$\uparrow$" and "$\downarrow$" stand for "the same as above". For ease of illustration, we have underlined the critical bit $\text{attr}_{\mathbf{\tau}_k}[k]$ in the attractors and each attractor equality of the attractor trajectory $A_{\mathbf{\tau}_k}$ inside the vectors of attractor equalities.)

\begin{align*}
\tau_0 &= \emptyset \\
\text{attr}_{\mathbf{\tau}_0} &= (fp \ #b1 \ #b11 \ #b0000) = [1.111.1111] \quad \text{[i.e. } -\infty \text{] } \Rightarrow \text{UNSAT} \\
\tau_1 &= \tau_0 \cup \{\text{obj}[0] = 0\} \\
\text{attr}_{\mathbf{\tau}_1} &= (fp \ #b0 \ #b000 \ #b0000) = [0.000.0000] \quad \text{[i.e. } +0 \text{] } \Rightarrow \text{UNSAT} \\
\tau_2 &= \tau_1 \cup \{\text{obj}[1] = 1\} \\
\text{attr}_{\mathbf{\tau}_2} &= (fp \ #b0 \ #b100 \ #b0000) = [0.100.0000] \quad \text{[i.e. } +2 \text{] } \Rightarrow \text{UNSAT} \\
\tau_3 &= \tau_2 \cup \{\text{obj}[2] = 1\} \\
\text{attr}_{\mathbf{\tau}_3} &= (fp \ #b0 \ #b110 \ #b0000) = [0.110.0000] \quad \text{[i.e. } +8 \text{] } \Rightarrow \text{SAT} \\
\tau_4 &= \tau_3 \cup \{\text{obj}[3] = 0\} \\
\text{attr}_{\mathbf{\tau}_4} &= (fp \ #b0 \ #b110 \ #b0000) = [0.110.0000] \quad \text{["" ] } \Rightarrow \text{UNSAT} \\
\tau_5 &= \tau_4 \cup \{\text{obj}[4] = 1\} \\
\text{attr}_{\mathbf{\tau}_5} &= (fp \ #b0 \ #b110 \ #b1000) = [0.110.1000] \quad \text{[i.e. } +12 \text{] } \Rightarrow \text{UNSAT} \\
\tau_6 &= \tau_5 \cup \{\text{obj}[5] = 1\} \\
\text{attr}_{\mathbf{\tau}_6} &= (fp \ #b0 \ #b110 \ #b1100) = [0.110.1100] \quad \text{[i.e. } +14 \text{] } \Rightarrow \text{SAT} \\
\tau_7 &= \tau_6 \cup \{\text{obj}[6] = 0\} \\
\text{attr}_{\mathbf{\tau}_7} &= (fp \ #b0 \ #b110 \ #b1100) = [0.110.1100] \\
\tau_8 &= \tau_7 \cup \{\text{obj}[7] = 1\} \\
\text{attr}_{\mathbf{\tau}_8} &= (fp \ #b0 \ #b110 \ #b1101) = [0.110.1101] \quad \text{[i.e. } +29/2 \text{]}
\end{align*}

(i) $\tau_{k+1}[k] = \text{attr}_{\mathbf{\tau}_k}[k]$ if $\varphi_{\text{noNaN}} \land \mathbf{\tau}_k \land A_{\mathbf{\tau}_k}[k]$ is unsatisfiable,

(ii) $\tau_{k+1}[k] = \text{attr}_{\mathbf{\tau}_k}[k]$ otherwise.

By construction, $\tau_{k} = [\tau_{n}]_k$. Therefore, we can replace $\tau_{k}$ with $[\tau_{n}]_k$ so that

(i) $[\tau_{n}]_{k+1}[k] = \text{attr}_{[\tau_{n}]_k}[k]$ if $\varphi_{\text{noNaN}} \land [\tau_{n}]_k \land A_{[\tau_{n}]_k}[k]$ is unsatisfiable,

(ii) $[\tau_{n}]_{k+1}[k] = \text{attr}_{[\tau_{n}]_k}[k]$ otherwise.

We notice the following facts. For each $k \in [0..n-1]$, $[\tau_{n}]_k \subset \tau_{n}$. Furthermore, for each $k \in [0..n-1]$, $A_{\varphi}[k] = A_{[\tau_{n}]_k}[k]$ because $A_{\varphi}[k] = A_{\mathbf{\tau}_k}[k]$ by the definition of attractor trajectory, and $A_{\mathbf{\tau}_k}[k] = A_{[\tau_{n}]_k}[k]$ by the equality $\tau_{k} = [\tau_{n}]_k$. Thus, we can replace $[\tau_{n}]_{k+1}$ with $\tau_{n}$ and $A_{[\tau_{n}]_k}[k]$ with $A_{\varphi}[k]$, as follows. For each $k \in [0..n-1]$,

(i) $\tau_{n}[k] = \text{attr}_{\mathbf{\tau}_k}[k]$ if $\varphi_{\text{noNaN}} \land [\tau_{n}]_k \land A_{\varphi}[k]$ is unsatisfiable,

(ii) $\tau_{n}[k] = \text{attr}_{\mathbf{\tau}_k}[k]$ otherwise.

Hence, $\tau_{n}$ lexicographically maximizes $A_{\varphi}$ wrt. $\varphi_{\text{noNaN}}$.

Finally, we make the following two observations. The first is that the sequence $\tau_{0}, \tau_{1}, ..., \tau_{n}$ in definition 6 can be iteratively constructed using its list of requirements, for instance, by means of a sequence of incremental calls to an SMT solver. The second, more important, observation is that $\tau_{n}$ corresponds to the assignment of values which makes obj optimal in $\varphi_{\text{noNaN}}$.

Using the above definitions, we show that the following fact holds.
Theorem 2. Let \( \langle \varphi_{\text{noNaN}}, \text{obj} \rangle \), \( \tau_0, \ldots, \tau_n \), \( \text{attr}_\tau, \ldots, \text{attr}_{\tau_n}, A_{\tau_0}, \ldots, A_{\tau_n} \), and \( \mathcal{A}_\varphi \) be as in definition 6. Then, any model \( \mathcal{M} \) of \( \varphi_{\text{noNaN}} \) which lexicographically maximizes the attractor trajectory \( \mathcal{A}_\varphi \) is an optimal solution for the OMT(\( \mathcal{F}\mathcal{P} \)) problem \( \langle \varphi_{\text{noNaN}}, \text{obj} \rangle \).

Proof. (We prove the case of minimization, since that of maximizations is dual.) By Lemma 2 we have that \( \tau_n \) lexicographically maximize \( \mathcal{A}_\varphi \). Let \( \mathcal{M} \) be a model of \( \varphi_{\text{noNaN}} \) which lexicographically maximizes \( \mathcal{A}_\varphi \), and let \( \mu \) be its restriction to \( \text{obj} \). Since both \( \tau_n \) and \( \mathcal{M} \) lexicographically maximize \( \mathcal{A}_\varphi \), for the uniqueness of \( \tau_n \), we immediately notice that \( \mu = \tau_n \), so that \( \tau_k = [\mu]_k \) for each \( k \in [0..n] \) and \( \mu \) lexicographically maximize \( \mathcal{A}_\varphi \).

By definition, \( \mathcal{M} \) is an optimal solution for \( \langle \varphi_{\text{noNaN}}, \text{obj} \rangle \) iff there exists no other model \( \mathcal{M}' \) for it s.t. \( \mathcal{M}'(\text{obj}) < \mathcal{M}(\text{obj}) \). Hence, we show by contradiction that no such \( \mathcal{M}' \) can exist.

Assume (for the sake of contradiction), that there exists a model \( \mathcal{M}' \) for \( \varphi_{\text{noNaN}} \), s.t. \( \mathcal{M}'(\text{obj}) < \mathcal{M}(\text{obj}) \), and let \( \mu' \) be the restriction of \( \mathcal{M}' \) to \( \text{obj} \). Then there must be at least one index \( i \) for which \( \mu[i] \neq \mu'[i] \). Let \( m \) be the smallest such index. Recalling that \( \tau_m = [\mu]_m \) and \( \tau_{m+1} = [\mu]_{m+1} \), we set \( \tau'_m = \mu'_{|m+1} \). Then, \( \tau_m \subseteq \tau'_{m+1}, \tau_m \subset \tau'_{m+1}, \tau_{m+1} \neq \tau'_{m+1} \). In particular, \( \tau_{m+1}[m] = \tau'_{m+1}[m] \) and therefore \( \tau_{m+1}[m] = \text{attr}_\tau[m] \) if \( \tau'_{m+1}[m] = \text{attr}_\tau'[m] \), and vice versa.

Then, we distinguish two cases.

In the first case, \( \tau_{m+1}[m] = \text{attr}_\tau[m] \) and \( \tau'_{m+1}[m] = \text{attr}_\tau'[m] \). From \( \tau_{m+1}[m] = \text{attr}_\tau[m] \) and the fact that \( \mu \) lexicographically maximizes \( \mathcal{A}_\varphi \), we derive that \( \varphi_{\text{noNaN}} \land \tau_m \land \mathcal{A}_\varphi [m] \) is unsatisfiable, where \( \mathcal{A}_\varphi [m] \overset{\text{def}}{=} (\text{obj}[m] = \text{attr}_\tau[m]) \). Since \( \tau_m \subseteq \tau'_{m+1} \), \( \tau'_m \land \mathcal{A}_\varphi [m] \), we conclude that \( \varphi_{\text{noNaN}} \land \mu' = \bot \), so that \( \mathcal{M}' \) cannot be a model of \( \varphi_{\text{noNaN}} \), contradicting the initial assumption.

In the second case, \( \tau_{m+1}[m] = \text{attr}_\tau[m] \) and \( \tau'_{m+1}[m] = \text{attr}_\tau'[m] \). Therefore, by Lemma 1, for every pair of models \( \mathcal{M}_1, \mathcal{M}_2 \) for \( \varphi_{\text{noNaN}} \) which extend respectively \( \tau_{m+1} \) and \( \tau'_{m+1} \) we have that \( \mathcal{M}_1(\text{obj}) \leq \mathcal{M}_2(\text{obj}) \). Since \( \tau_{m+1} = [\mu]_{m+1} \) and \( \tau'_{m+1} = [\mu']_{m+1} \), it follows that \( \mathcal{M}'(\text{obj}) \not\leq \mathcal{M}(\text{obj}) \), contradicting the initial assumption. \( \square \)

4 OMT(\( \mathcal{F}\mathcal{P} \)) Procedures

In this paper, we consider two approaches for dealing with OMT(\( \mathcal{F}\mathcal{P} \)): a basic linear/binary search, based on the inline OMT schema for OMT(\( \mathcal{LRA} \cup \mathcal{T} \)) presented in [38], and Floating-Point Optimization with Binary Search (OFP-BS), a brand-new engine inspired by the OBV-BS algorithm for unsigned Bit-Vectors in [31] and by Theorem 2 and relative definitions in §3.2.

4.1 OMT-based Approach

The OMT-based approach for OMT(\( \mathcal{F}\mathcal{P} \)) adapts the linear- and binary-search schemata for OMT(\( \mathcal{LRA} \cup \mathcal{T} \)) presented in [38] to deal with \( \mathcal{F}\mathcal{P} \) objectives.
In the basic linear-search schema, the optimization search is advanced by means of a sequence of linear cuts, each of which forces the OMT solver to look for a new model $M'$ which improves the value of $\text{obj}$ wrt. the most recent model $M$. In the binary-search schema, instead, the OMT solver learns an incremental sequence of cuts which bisect the current domain of the objective function. For clarity, we recap here the essential elements of the binary-search schema presented in [37, 38]. At the beginning of the optimization search and following each update of the lower- ($lb$) and upper- ($ub$) bounds of $\text{obj}$, the OMT solver computes a pivoting value $\text{pivot} \equiv \lfloor \rho \cdot ub + (1 - \rho) \cdot lb \rfloor$, for some value of $\rho$ (e.g. $\frac{1}{2}$). If $\text{pivot}$ lies inside the range $[lb, ub]$, a cut of the form $\{ \text{obj} < \text{pivot} \}$ is learned. Otherwise, if –due to rounding side-effects of $\mathcal{FP}$ operations– $\text{pivot}$ lies outside the range $[lb, ub]$, a cut of the form $\{ \text{obj} < ub \}$ is learned instead. If the cut is satisfiable, the upper-bound of $\text{obj}$ is updated with a new model value of $\text{obj}$. Otherwise, the lower-bound is made equal to $\text{pivot}$ [resp. $ub$]. The algorithm terminates when the search interval $[lb, ub]$ becomes empty. In general, it is reasonable to expect the binary-search schema to converge towards the optimal solution faster than the linear-search schema, because the feasible domain of a $\mathcal{FP}$ goal can be comprised by an exponentially large number of values (wrt. the bit-width of the cost function).

In either schema, whenever the optimization engine encounters for the first time a solution s.t. $\text{obj} = \text{NAN}$, the OMT solver learns a unit-clause of the form $\neg(\text{isNAN}(\text{obj}))$ so as to look for an optimal solution different from NAN (if any).

When dealing with $\mathcal{FP}$ objectives, differently from the case of $\mathcal{LRA}$ in [38], it is not necessary to implement a specialized optimization procedure within the $\mathcal{FP}$-Solver in order to guarantee the termination of the optimization search. Indeed, such procedure is not available when Floating-Point terms are bit-blasted into Bit-Vectors eagerly, or when the $\text{ACDCL} \mathcal{FP}$-Solver is used, because by the time the optimization procedure is called the domain interval of any $\mathcal{FP}$ term contains a singleton value. Conversely, such a minimization procedure could be envisaged when the OMT solver uses a lazy $\mathcal{FP}$-Solver as back-end, so as to speed-up the convergence towards the optimal solution$^3$.

4.2 Floating-Point Optimization with Binary Search

The Floating-Point Optimization with Binary Search algorithm is a new engine for OMT($\mathcal{FP}$) which is inspired by the OBV-BS algorithm for OMT($\mathcal{BV}$) [31] and is a direct implementation of Definition 6 and Theorem 2.

The optimization search tries to lexicographically maximize an implicit attractor trajectory vector $\mathcal{A}_\phi$, which is incrementally derived from the current value of the dynamic attractor. The raw value of the dynamic attractor’s bits drive the optimization search towards the direction of maximum gain at any given point in time, without disrupting any decision that has been already made. The dynamic attractor is incrementally updated along the search, based on the outcome of the previous rounds of the optimization search. At each round, one bit of the objective function is assigned its final value.

$^3$Currently, there is no such specialized optimization procedure embedded within the lazy $\mathcal{FP}$-Solver of OPTIMATHSAT, so we won’t describe this approach any further.
function OFP-BS (ϕ, obj)
1: ⟨res, M⟩ := SMT.CHECK_UNDER_ASSUMEPTIONS(ϕ, ∅) 
2: if (res == UNSAT) then 
3: return ⟨res, ∅⟩ // ϕ is unsatisfiable 
4: if (M(obj) == NAN) then 
5: ⟨res, M′⟩ := SMT.CHECK_UNDER_ASSUMEPTIONS(ϕ ∧ ¬IsNaN(obj), ∅) 
6: if (res == UNSAT) then 
7: return ⟨SAT, M⟩ // obj can only be NAN 
8: else 
9: M := M′ 
10: ϕ := ϕ ∧ ¬IsNaN(obj) 
11: τ := ∅ // from now on, obj cannot be equal NAN 
12: attr_v := UPDATE_DYNAMICATTRACTOR(τ) 
13: SMT.SET_BRANCHING_PREFERENCE(obj) 
14: SMT.UPDATE_BITS_POLARITY_TO(obj, attr_v) 
15: for i := 0 up to n − 1 do 
16: eq := (obj[i] = attr_v[i]) // attractor equality A_v[i] 
17: if (M |= eq) then 
18: τ := τ ∪ {eq} 
19: else 
20: SMT.SET_BRANCHING_PREFERENCE(obj) 
21: SMT.UPDATE_BITS_POLARITY_TO(obj, attr_v) 
22: ⟨res, M′⟩ := SMT.CHECK_UNDER_ASSUMEPTIONS(ϕ, τ ∪ {eq}) 
23: if (res == SAT) then 
24: τ := τ ∪ {eq} 
25: M := M′ 
26: else 
27: τ := τ ∪ {¬eq} 
28: attr_v := UPDATE_DYNAMICATTRACTOR(τ) 
29: return ⟨SAT, M⟩ 

Fig. 2. OFP-BS Algorithm for Floating-Point optimization.

The first round decides the sign, the next batch of rounds decides the exponent and the remaining rounds decide the fine-grained details of the significand.

The pseudo-code of OFP-BS is shown in Figure 2. The arguments of the algorithm are the input formula ϕ and the FP objective obj, where obj is a FP variable with ebits bits in the exponent, sbits − 1 in the significand and n = ebits + sbits bits overall.

The procedure starts by checking whether the input formula ϕ is satisfiable and immediately terminates if that is not the case (lines 1-3). If obj = NAN in M then the procedure checks whether there exists a model M′ for ϕ ∧ ¬IsNaN(obj) (lines 4-5). If this is not the case, the procedure terminates immediately and returns the pair ⟨SAT, M⟩ (line 7). Otherwise, the model M is updated with the new model M′, and ϕ is permanently extended with the constraint ¬IsNaN(obj) (lines 9-10).

At this point, the procedure initializes the value of the dynamic attractor by invoking an external function UPDATE_DYNAMICATTRACTOR() with the empty assignment τ.
as parameter, so that the returned value is equal to $-\infty$ when minimizing and $+\infty$ when maximizing (lines 11-12). Then, the execution moves to the section of code implementing the core part of the OFP-BS algorithm (lines 15-28), which consists of a loop over the bits of \texttt{obj}, starting from the MSB \texttt{obj}[0] down to the LSB \texttt{obj}[n - 1].

Inside this loop, OFP-BS first checks whether the value of \texttt{obj}[i] in \texttt{M} matches the \textit{i}-th bit of the (current) dynamic attractor \texttt{attr}_\tau. If this is the case, then the \textit{i}-th bit is already set to its “best” value in \texttt{M}. Thus, the assignment \(\tau\) is extended so as to permanently set \texttt{obj}[i] = \texttt{attr}_\tau[i] (line 16), and the optimization search moves to the next iteration of the loop. If instead \texttt{obj}[i] \neq \texttt{attr}_\tau[i] in \texttt{M}, we need to verify whether the value of the objective function in \texttt{M} can be improved by forcing the \textit{i}-th bit of \texttt{obj} equal to the \textit{i}-th bit of the dynamic attractor. To do so, we incrementally invoke the underlying SMT solver, this time checking the satisfiability of \(\varphi\) under the list of assumptions \(\tau \cup \{\texttt{obj}[i] = \texttt{attr}_\tau[i]\}\) (line 22). If the SMT solver returns \texttt{SAT}, then the value of the objective function has been successfully improved. Hence, \(\tau\) is extended with an assignment setting \texttt{obj}[i] equal to \texttt{attr}_\tau[i], and \texttt{M} is replaced with the new model \texttt{M}' (lines 23-25). Otherwise, it is not possible to improve the objective function by toggling the value of \texttt{obj}[i], and \(\tau\) is extended so as to permanently set \texttt{obj}[i] \neq \texttt{attr}_\tau[i] (line 27). At this point, there is a mismatch between the value of the first \(i + 1\) bits of \texttt{obj} in \texttt{M}, corresponding to the assignment \(\tau\), and those of the current dynamic attractor. This mismatch is resolved by calling the function \textsc{update}\_\textsc{dynamic}\_\textsc{attractor}() with the updated assignment \(\tau\) as parameter (line 28). In either case, the execution moves to the next iteration of loop.

After exactly \(n\) iterations of the loop, the optimization search terminates with the pair \(\texttt{SAT, M}\), where \texttt{M} is the optimum model of the given OMT\((FP \cup T)\) instance. The OFP-BS algorithm requires at most \(n + 2\) incremental calls to an underlying SMT\((FP)\) solver. The test in rows 17-18 allows for saving lots of such SMT calls when the current model already assigns \texttt{obj}[i] to its corresponding value in the attractor.

The function \textsc{update}\_\textsc{dynamic}\_\textsc{attractor}() takes as input \(\tau\), a (partial) assignment over the \(k\) most-significant bits of \texttt{obj} and, when \texttt{obj} is minimized \(^4\), and it

\(^4\)The implementation of \textsc{update}\_\textsc{dynamic}\_\textsc{attractor}() is dual when \texttt{obj} is maximized.
essentially works as follows. If \( \tau = \emptyset \), then nothing is known about the solution of the problem, so \(-\infty\) is returned. Otherwise, the procedure must compute the smallest \( \mathcal{FP} \) value different from \( \text{NAN} \) (if any) which extends \( \tau \). Since \( \tau \neq \emptyset \) then we know that the sign of the objective function has been permanently decided in \( \tau \). If \( \text{obj}[0] = 0 \) in \( \tau \), i.e. \( \text{obj} \) must be positive, the procedure must return the smallest positive \( \mathcal{FP} \) value admitted by \( \tau \). Hence, we extend \( \tau \) with \( \bigcup_{i=|\tau|}^{n-1} \text{obj}[i] = 0 \) and return the corresponding \( \mathcal{FP} \) value. If \( \text{obj}[0] = 1 \) in \( \tau \), i.e. \( \text{obj} \) can be negative values, the procedure must return the largest negative \( \mathcal{FP} \) value admitted by \( \tau \). We first check whether there exists a bit in the exponent of \( \text{obj} \) which is assigned to 0 in \( \tau \). If that is the case, we extend \( \tau \) with \( \bigcup_{i=|\tau|}^{n-1} \text{obj}[i] = 1 \) and return the corresponding \( \mathcal{FP} \) value. Otherwise, the procedure returns the value \(-\infty\), which is still a viable extension of \( \tau \).

### 4.3 Search Enhancements

Given a \( \mathcal{FP} \) value \( \text{attr} \) and a \( \mathcal{FP} \) goal \( \text{obj} \), (a combination of) the following techniques can be used to adjust the behavior of the optimization search, similarly what has been proposed for the case of OMT\((BV)\) by Nadel et al. in [31].

- **branching preference**: the bits of the \( \mathcal{FP} \) objective \( \text{obj} \) are marked, inside the OMT solver, as preferred variables for branching starting from the MSB down to the LSB. This ensures that conflicts involving the value of the objective function are handled as early as possible, possibly reducing the amount of work that needs to be redone after each back-jump.

- **polarity initialization**: the phase-saving value of each \( \text{obj}[i] \) is initialized with the value of \( \text{attr}[i] \). This encourages the OMT solver to assign the bits of \( \text{obj} \) so as to reassemble the bits of \( \text{attr} \), thus possibly speeding-up the convergence towards the optimal value.

In the case of the basic OMT schema described in Section §4.1, the effectiveness of either technique depends on the initial choice for \( \text{attr} \). In the lucky case, the value of \( \text{attr} \) pulls the optimization search in the right direction and speeds up the search. In the unlucky case, when \( \text{attr} \) pulls in the wrong direction, there is no visible effect or an overall slow down. For instance, in the case of the linear-search optimization schema, enabling both options with an unlucky choice of \( \text{attr} \) can cause the OMT solver to start the search from the furthest possible point from the optional solution, and thus enumerate an exponential number of intermediate solutions. Naturally, the OMT-based optimization search algorithm is still guaranteed to terminate even in the worst-case scenario, but the unpredictable performance makes using either technique a generally unsuitable option in practice.

In the case of the OFP-BS algorithm described in Section §4.2, we use the latest value of the dynamic attractor \( \text{attr}_\tau \) for both the branching preference (lines 11 and 18 of Figure 2) and the polarity initialization (rows 12 and 19 of Figure 2) techniques. We observe that the value of every bit in the dynamic attractor can change after the sign of the objective function has been decided. Furthermore, the value of all the significand’s bits in the dynamic attractor can also change during the process of determining the optimal exponent value of the objective function (see, e.g., Example 4). As a consequence,
if the OMT solver applies either enhancement before the correct improving direction is known, this may cause the underlying OMT engine to advance the search starting from a sub-optimal set of initial decisions. Enabling both enhancements at the same time could make things even worse. In order to mitigate this issue, we have designed a variant of our optimization-search approach which does not apply either enhancement on those bits of the objective function for which the best improving direction is not yet known. We have called this variant safe bits restriction.

5 Experimental Evaluation

We assess the performance of OptiMathSAT (v. 1.6.2) on a set of OMT($FP$) formulas that have been automatically generated using the SMT($FP$) benchmark-set of [3]. The formulas, the results and the scripts necessary to reproduce these results are made publicly available and can be downloaded from [1].

Experiment Setup. This experiment has been performed on an i7-6500U 2.50GHz Intel Quad-Core machine with 16GB of ram and running Ubuntu Linux 17.10. For each formula being tested we used a timeout of 600 seconds. The OMT($FP$) instances used in this experiment have been automatically generated starting from the satisfiable formulas included in the SMT($FP$) benchmark-set of [3]. We did not consider any of the unsatisfiable instances that are present in the remote repository.

For each of the original SMT($FP$) formulas we applied the following transformations. First, we either relaxed or removed some of the constraints in the original problem, so as to broaden the set of feasible solutions. This step is necessary because the majority of the original SMT($FP$) formulas admits only one solution. However, this is not necessarily the ideal situation when comparing different optimization approaches. Second, for each $FP$ variable $v$ appearing inside a SMT($FP$) problem we generated a pair of OMT($FP$) instances, one for the minimization and another for the maximization of $v$. At the end of this step, we obtained 39536 OMT($FP$) formulas. Third, we randomly selected up to 300 OMT($FP$) instances from each of the five groups of problems in the OMT($FP$) benchmark-set. This filtering step yielded a total of 1120 SMT-LIBv2 formulas.

We consider two OMT-based baseline configurations, OptiMathSAT(OMT+LIN) and OptiMathSAT(OMT+BIN), that run the linear- and the binary-search respectively. These configurations have been tested using both the eager and the lazy $FP$ approaches. The third baseline approach, named OptiMathSAT(EAGER+OBV-BS), is based on a reduction of the OMT($FP$) problem to OMT($BV$) and it uses OptiMathSAT’s implementation of the OBV-BS engine5 presented by Nadel et al. in [31]. For this test, we have generated an OMT($BV$) benchmark-set using a $BV$ encoding that mimics the essential aspects of the OFP-BS algorithm described Section §4.2.

We compared these baseline approaches with a configuration using the OFP-BS algorithm and the eager $FP$ approach, namely OptiMathSAT(EAGER+OFP-BS).

5The binaries of the original OMT($BV$) tools presented in [31] are not publicly available.
We have separately tested the effect of enabling the branching preference (BP), the polarity initialization (PI) and the safe bits restriction (SO) enhancements described in Section §3.2, whenever these options were supported by the given configuration.

Last, in order to assess the significance of the optimization problems used in this experiment, we have collected the run-time statistics of OPTIMATHSAT on the SMT formulas obtained by stripping the objective function from each OMT instance. We named this configuration OPTIMATHSAT(EAGER+SMT).

We have not included other tools in our experiment because we are not aware of any other OMT(PP) solver. For all problem instances, we verified the correctness of the optimal solution found by each configuration with an SMT solver (MATHSAT5). When terminating, all tools returned the same optimum value. In order to perform this cross-check as efficiently as possible, we enabled model generation on every configuration so that the optimum model could be extracted and verified.

Experiment Results. The results of this experiment are listed in Table 2. Figure 4 depicts the loc-scale cactus plot of the same data, for a visual comparison among the different configurations. In addition, Figures 5, 6 and 7 show a selection of relevant
Fig. 4. Comparison among various Optimathsat configurations on a subset of 1120 Omt(FP) formulas generated from the Smt(FP) formulas of [3].

pairwise comparisons among various Optimathsat configurations. Figure 5 focuses on variants of the Omt-based linear-search approach. Figure 6 depicts variants of the OMT-based binary-search approach, whereas Figure 7 focuses on the OFP-BS engine.

For what concerns OMT-based linear-search optimization, we observe that Optimathsat performs the best when no enhancement is enabled. In particular, the empirical evidence suggests that enabling branching preference significantly increases the number of timeouts, generally deteriorating the performance (plot 1A in Fig. 5). Enabling only polarity initialization does not result in an appreciable change on the running time of the solver (plot 1B in Fig. 5). In contrast, enabling both enhancements at the same time has a small chance to result in a small improvement of the search time (plot 2A in Fig. 5), but it generally worsens the performance and results in a drastic increase in the number of timeouts (Table 2). We justify these results as follows. First, when only polarity initialization is used, the phase-saving value that is being set by Optimathsat does not really matter because the optimization search is dominated by the structure of the formula itself rather than by the bits of the FP objective. Second, when polarity initialization is used on top of branching preference, there is an even more drastic decrease in performance due to the fact that the initial phase-saving value that is statically assigned by the OMT solver to the bits of the FP objective cannot
Fig. 5. Pairwise comparisons on OMT($\mathcal{FP}$) formulas using OMT-based linear-search and other configurations. (Blue points denote satisfiable benchmarks, green denotes a timeout.)
Fig. 6. Pairwise comparisons on OMT($\mathcal{FP}$) formulas using OMT-based binary-search and other configurations. (Blue points denote satisfiable benchmarks, green denotes a timeout.)
Fig. 7. Pairwise comparisons on OMT(\(\mathcal{FP}\)) formulas using the OFP-BS engine and other configurations. (Blue points denote satisfiable benchmarks, green denotes a timeout.)
be expected to be “good enough” for any situation. In fact, as illustrated in example 4, the initial phase-saving can be misleading and force the OMT solver—when running in linear-search—to explore an exponential number of intermediate satisfiable solutions.

In the case of the OMT-based binary-search optimization approach, we observe that it solves more formulas than linear-search and it generally appears to be faster (plot 3B in Fig. 5). Overall, polarity initialization does not seem to be beneficial, whereas enabling branching preference increases the number of formulas solved within the timeout. This behavior is different from the linear-search approach, and we conjecture that it is due to the fact that, with the OMT-based binary-search approach, branching over the bits of the objective function can reveal in advance any (partial) assignment to the bits of the objective function that it is inconsistent wrt. the pivoting cuts learned by the optimization engine.

Using the lazy $FP$ engine results in fewer formulas being solved, although a significant number of these benchmarks is solved faster than with any other configuration (over 90 instances, for both configurations).

The OPTIMATHSAT(EAGER+OBV-BV) configuration is able to solve 1013 formulas within the timeout, showing that OMT($FP$) can be reduced to OMT($BV$) effectively, and that—on the given benchmark-set—the performance of this approach are comparable with the best OMT($FP$) configurations being tested.

Overall, the best performance is obtained by using the OFP-BS engine, with up to 1019 benchmark-set instances being solved in correspondence to the OPTIMATHSAT(EAGER+OFP-BS+PI) configuration. In plot 2B of Figures 5 and 6, we show the pairwise comparison of the best OFP-BS configuration with the best OMT-based run.

Similarly to the case of OMT-based optimization with linear-search, we observe that enabling branching preference generally makes the performance worse (plot 1A in Fig. 7). Instead, when polarity initialization is used we observe a general performance improvement that does not only result in an increase in the number of formulas being solved within the timeout, but also a noticeable reduction of the solving time as a whole. This is in contrast with the case of OMT-based optimization, and it can be explained by the fact that OFP-BS uses an internal heuristic function to dynamically determine and update the most appropriate phase-saving value for the bits of the objective function. An equally important role is played by the safe bits restriction, that limits the effects of branching preference and polarity initialization to only certain bits of the dynamic attractor. As illustrated by the plots in the second and third rows of Figure 7 and by the data in Table 2, this feature is particularly effective when used in combination with branching preference.

The results of OPTIMATHSAT over the SMT-only version of the benchmark-set are reported in Table 2 and in the scatter-plot 3B in Fig. 6, and show that for a large number of instances the OMT problem is considerably harder than its SMT-only version. There are a few exceptions to this rule, that we ascribe to the fact that the removal of the objective function alters the internal stack of formulas, and this can have unpredictable consequences on the behavior of various internal heuristics that depend on it. A solution can be found in a shorter amount of time when the sequence of (heuristic) choices is compatible with its assignment and it requires little back-tracking effort.
6 Conclusions and Future Work

We have presented for the first time OMT procedures (for signed Bit-Vectors and) Floating-Point numbers, based on the novel notions of attractor, dynamic attractor and attractor trajectory, which we have implemented in OPTIMATHSAT and tested on modified problems from SMT-LIB.

Ongoing research involves implementing our OFP-BS procedure on top of the ACDCL SMT($\mathcal{FP}$) procedure —which is not immediate to do efficiently because the latter approach does not allow directly accessing and setting the single bits of the objective (since $\mathcal{BV}$ and $\mathcal{FP}$ are not signature-disjoint). Future research involves experimenting the new OMT procedure directly on problems coming from bit-precise SW and HW verification, produced, e.g., by the NuXmv model checker [2].

References

1. http://disi.unitn.it/trentin/resources/floatingpoint_test.tar.gz.
2. NuXmv. https://nuxmv.fbk.eu.
3. SmtLib2. www.smtlib.cs.uiowa.edu/.
4. IEEE standard 754, 2008. http://grouper.ieee.org/groups/754/.
5. H. F. Albuquerque, R. F. Araujo, I. V. de Bessa, L. C. Cordeiro, and E. B. de Lima Filho. OptCE: A Counterexample-Guided Inductive Optimization Solver. In SBMF, volume 10623 of Lecture Notes in Computer Science, pages 125–141. Springer, 2017.
6. R. F. Araujo, H. F. Albuquerque, I. V. de Bessa, L. C. Cordeiro, and J. E. C. Filho. Counterexample guided inductive optimization based on satisfiability modulo theories. Sci. Comput. Program., 165:3–23, 2018.
7. R. Araújo, I. Bessa, L. C. Cordeiro, and J. E. C. Filho. SMT-based Verification Applied to Non-convex Optimization Problems. In 2016 VI Brazilian Symposium on Computing Systems Engineering (SBESC), pages 1–8, Nov 2016.
8. N. Bjorner and A.-D. Phan. $\nu Z$ - Maximal Satisfaction with Z3. In Proc International Symposium on Symbolic Computation in Software Science, Gammart, Tunisia, December 2014. EasyChair Proceedings in Computing (EPiC).
9. N. Bjorner, A.-D. Phan, and L. Fleckenstein. $\nu Z$ - An Optimizing SMT Solver. In Proc. TACAS, volume 9035 of LNCS. Springer, 2015.
10. M. Bozzano, R. Bruttomesso, A. Cimatti, A. Francêz, Z. Hanna, Z. Khasidashvili, A. Palti, and R. Sebastiani. Encoding RTL Constructs for MathSAT: a Preliminary Report. In Proc. 3rd Workshop of Pragmatics on Decision Procedure in Automated Reasoning, PDPAR’05, ENTCS. Elsevier, 2005.
11. M. Brain, V. D’Silva, A. Griggio, L. Haller, and D. Kroening. Interpolation-Based Verification of Floating-Point Programs with Abstract CDCL. In SAS, pages 412–432, 2013.
12. M. Brain, V. D’Silva, A. Griggio, L. Haller, and D. Kroening. Deciding floating-point logic with abstract conflict driven clause learning. Formal Methods in System Design, 45(2):213–245, 2014.
13. M. Brain, C. Tinelli, P. Rümmer, and T. Wahl. An Automatable Formal Semantics for IEEE-754 Floating-Point Arithmetic. In ARITH, pages 160–167. IEEE, 2015.
14. A. Brilliout, D. Kroening, and T. Wahl. Mixed abstractions for floating-point arithmetic. In 2009 Formal Methods in Computer-Aided Design, pages 69–76, Nov 2009.
15. R. Brinkmann and R. Drechsler. RTL-datapath verification using integer linear programming. In Proc. ASP-DAC 2002, pages 741–746. IEEE, 2002.
16. R. Brummayer. Efficient SMT Solving for Bit-Vectors and the Extensional Theory of Arrays. PhD thesis, Informatik, Johannes Kepler University Linz, 2009.
17. R. Brummayer and A. Biere. Boolector: An efficient smt solver for bit-vectors and arrays. In TACAS, pages 174–177, Berlin, Heidelberg, 2009. Springer-Verlag.
18. R. Bruttomesso, A. Cimatti, A. Franzén, A. Griggio, Z. Hanna, A. Nadel, A. Palti, and R. Sebastiani. A Lazy and Layered SMT(BV) Solver for Hard Industrial Verification Problems. In CAV, volume 4590 of LNCS, pages 547–560. Springer, 2007.
19. A. Cimatti, A. Franzén, A. Griggio, R. Sebastiani, and C. Stenico. Satisfiability modulo the theory of costs: Foundations and applications. In TACAS, volume 6015 of LNCS, pages 99–113. Springer, 2010.
20. A. Cimatti, A. Griggio, B. J. Schaafsma, and R. Sebastiani. A Modular Approach to MaxSAT Modulo Theories. In International Conference on Theory and Applications of Satisfiability Testing, SAT, volume 7962 of LNCS, July 2013.
21. I. Dillig, T. Dillig, K. L. McMillan, and A. Aiken. Minimum Satisfying Assignments for SMT. In CAV, pages 394–409, 2012.
22. K. Fazekas, F. Bacchus, and A. Biere. Implicit Hitting Set Algorithms for Maximum Satisfiability Modulo Theories. In IJCAR, volume 10900 of Lecture Notes in Computer Science, pages 134–151. Springer, 2018.
23. V. Ganesh and D. L. Dill. A Decision Procedure for Bit-Vectors and Arrays. In IJCAR, volume 10900 of Lecture Notes in Computer Science, pages 134–151. Springer, 2018.
24. L. Hadarean. An Efficient and Trustworthy Theory Solver for Bit-vectors in Satisfiability Modulo Theories. PhD thesis, New York University, 2015.
25. L. Hadarean, K. Bansal, D. Jovanovic, C. Barrett, and C. Tinelli. A Tale of Two Solvers: Eager and Lazy Approaches to Bit-Vectors. In CAV, volume 8559 of Lecture Notes in Computer Science, pages 680–695. Springer, 2014.
26. L. Haller, A. Griggio, M. Brain, and D. Kroening. Deciding Floating-Point Logic with Systematic Abstraction. In Proc. of FMCAD, 2012. To Appear.
27. G. Kovásznai, C. Biró, and B. Erdélyi. Puli - a problem-specific omt solver. EasyChair Preprint no. 371, EasyChair, 2018.
28. D. Larraz, A. Oliveras, E. Rodríguez-Carbonell, and A. Rubio. Minimal-Model-Guided Approaches to Solving Polynomial Constraints and Extensions. In SAT, 2014.
29. Y. Li, A. Albarghouthi, Z. Kincad, A. Gurfinkel, and M. Chechik. Symbolic Optimization with SMT Solvers. In POPL, 2014.
30. P. Manolios and V. Papavasileiou. Ilp modulo theories. In CAV, pages 662–677, 2013.
31. A. Nadel and V. Ryvchin. Bit-Vector Optimization. In Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2016, volume 9636 of LNCS. Springer, 2016.
32. A. Niemetz. Bit-Precise Reasoning Beyond Bit-Blasting. PhD thesis, Informatik, Johannes Kepler University Linz, 2017.
33. A. Niemetz, M. Preiner, A. Fröhlich, and A. Biere. Improving Local Search For Bit-Vector Logics in SMT with Path Propagation. In Proc. 4th Intl. Work. on Design and Implementation of Formal Tools and Systems (DFTS’15), page 10 pages, 2015.
34. R. Nieuwenhuis and A. Oliveras. On SAT Modulo Theories and Optimization Problems. In Proc. Theory and Applications of Satisfiability Testing - SAT 2006, volume 4121 of LNCS. Springer, 2006.
35. O. Roc. Optimization Modulo Theories. Master’s thesis, Polytechnic University of Catalonia, 2011. http://hdl.handle.net/2099.1/14204.
36. P. Ruemmer and T. Wahl. An SMT-LIB Theory of Binary Floating-Point Arithmetic. SMT 2010 Workshop, July 2010. Available at http://www.philipp.ruemmer.org/publications/smt-fpa.pdf.
37. R. Sebastiani and S. Tomasi. Optimization in SMT with L(A(Q) Cost Functions. In IJCAR, volume 7364 of LNAI, pages 484–498. Springer, July 2012.
38. R. Sebastiani and S. Tomasi. Optimization Modulo Theories with Linear Rational Costs. 
ACM Transactions on Computational Logics, 16(2), March 2015.

39. R. Sebastiani and P. Trentin. OptiMathSAT: A Tool for Optimization Modulo Theories. In 
Proc. International Conference on Computer-Aided Verification, CAV 2015, volume 9206 of 
LNCS. Springer, 2015.

40. R. Sebastiani and P. Trentin. Pushing the Envelope of Optimization Modulo Theories with 
Linear-Arithmetic Cost Functions. In Proc. Int. Conference on Tools and Algorithms for the 
Construction and Analysis of Systems, TACAS’15, volume 9035 of LNCS. Springer, 2015.

41. R. Sebastiani and P. Trentin. On Optimization Modulo Theories, MaxSMT and Sorting Net-
works. In Proc. Int. Conference on Tools and Algorithms for the Construction and Analysis 
of Systems, TACAS’17, volume 10205 of LNCS. Springer, 2017.

42. R. Sebastiani and P. Trentin. OptiMathSAT: A Tool for Optimization Modulo Theories. 
Journal of Automated Reasoning, Dec 2018.

43. A. Zeljić, P. Backeman, C. M. Wintersteiger, and P. Rümmer. Exploring approximations for 
floating-point arithmetic using uppsat. In D. Galmiche, S. Schulz, and R. Sebastiani, editors, 
Automated Reasoning, pages 246–262, Cham, 2018. Springer International Publishing.

44. A. Zeljić, C. M. Wintersteiger, and P. Rümmer. Approximations for model construction. 
In S. Demri, D. Kapur, and C. Weidenbach, editors, Automated Reasoning, pages 344–359, 
Cham, 2014. Springer International Publishing.

45. A. Zeljić, C. M. Wintersteiger, and P. Rümmer. An approximation framework for solvers 
and decision procedures. Journal of Automated Reasoning, 58(1):127–147, Jan 2017.