VolCE v3.0

Cunjing Ge, Feifei Ma and Jian Zhang
Institute of Software
Chinese Academy of Sciences
Email: {gecj,maff,zj}@ios.ac.cn

February 27, 2017

1 Introduction

1.1 What is VolCE?

VolCE is designed for computing or estimating the size of the solution space of an SMT formula where the theory T is restricted to the linear arithmetic theory. (SMT stands for Satisfiability Modulo Theories.) The prototype tool presented in [3] computes the exact volume of the solution space. However, exact volume computation in general is an extremely difficult problem. It has been proved to be #P-hard, even for explicitly described polytopes. On the other hand, it suffices to have an approximate value of the volume in many cases. Later we implemented a tool to estimate the volume of polytopes [1] and integrated it into the framework of [3]. The new tool is called VolCE. It can efficiently handle instances of dozens of dimensions with high accuracy. In addition, VolCE also accepts constraints involving independent Boolean variables.

1.2 What can VolCE do?

VolCE uses the following three packages:

- PolyVest [1], which can be used to estimate the volume of polytopes.
- Vinci [5], a software package that implements several algorithms for (exact) volume computation.
- LattE (Lattice point Enumeration) [2], a software package dedicated to the problems of counting lattice points and integration inside convex polytopes.

2 Installation

- Step 1: Make sure that g++ (version 4.8 or higher version) is installed on your machine (you can type "`g++ -v`" to check this).
- Step 2: The functionality of `VolCE` is dependent on some other libraries: `boost`, `glpk`, and `Armadillo`. On Ubuntu or Debian, you can use "apt-cache search" and "apt-get install" to find and install all of these libraries. Or you can download these libraries from:

Library	URL
`boost`	http://www.boost.org/
`glpk`	http://www.gnu.org/software/glpk/
`Armadillo`	http://arma.sourceforge.net/

  Note: `Armadillo` also requires `gfortran`, `lapack` and `blas`.
- Step 3: Open a shell (command line), change into the directory that was created by unpacking the `VolCE` archive, and type:

```
sh build.sh
```

- Step 4: Build and install `LattE` [2]. Then move the executable files (`count` and `scdd_gmp`) into directory "`bin/`".

  Note: Move or copy 'volce3' with directory 'bin/' together since `VolCE3` requires the tools in 'bin/'.

This release of `VolCE` has been successfully built on the following operating systems:

- Ubuntu 14.04 on 64-bit with g++ 4.8.4
- Ubuntu 12.04 on 32-bit with g++ 4.8.1

3 The `SMT-LIBv2` Language Inputs

The input of `VolCE` is an SMT formula where the theory T is restricted to the linear arithmetic theory. It involves variables of various types (including integers, reals and Booleans). We usually use b_i to denote Boolean variables, x_j to denote numeric variables. In the input formula, there can be logical operators (like AND, OR, NOT), arithmetic operators (like addition, subtraction, scalar multiplication) and comparison operators (like $<$, $\leq$, $>$, $\geq$, $=$, $\neq$). `VolCE` employs the `SMT-LIBv2` language. For details of this language, visit the website:

`http://www.smt-lib.org/`

Table 1: Supported SMT-LIBv2 Components

Commands	`declare-fun declare-const define-fun assert`
Variable Types	`Int Real Bool`

Identifiers	`let ite`
	`and or not => xor`
	`+ - * /`
	`= distinct > >= < <=`

`VolCE` recognizes `SMT-LIBv2` format from the file name extension "`.smt2`". Table 1 lists the essential commands, variable types and identifiers of `SMT-LIBv2` language for describing an SMT(BV) formula. `VolCE` ignores some basic commands like `set-logic`, `set-info`, `check-sat`, `exit`. It directly checks all of the assertions.

An example of an SMT(BV) formula is of the following:

$$\begin{aligned}&(x \leq 1) \text{ AND } (y \leq 1) \text{ AND } (x \geq 0) \text{ AND } (y \geq 0) \text{ AND}\\&((x + y < 1) \text{ OR } (\text{NOT } x < y)) \text{ AND}\\&((x + y < 1) \text{ OR } x < y \text{ OR } b) \text{ AND}\\&((\text{NOT } x + y < 1) \text{ OR } x < y).\end{aligned} \tag{1}$$

In the `SMT-LIBv2` language, the above Formula 1 would be written like this:

```
(set-logic QF_LRA)
(set-info :f1.smt2)
(set-info :smt-lib-version 2.0)
(set-info :status sat)
(declare-fun x () Real)
(declare-fun y () Real)
(declare-fun b () Bool)
(assert (and (<= x 1) (<= y 1) (>= x 0) (>= y 0)))
(assert (let ((v1 (< (+ x y) 1)) (v2 (< x y)))
(and (or v1 (not v2)) (or v1 v2 b) (or (not v1) v2))))
(check-sat)
(exit)
```

See the file `examples/f1_lra.smt2` and `examples/f1_lia.smt2`.

4 Running `VolCE`

To run `VolCE`, you should switch your working directory to the absolute path of `VolCE`.

`VolCE` has a help menu. To view it, simply type the command "`./volce --help`".

The general usage of `VolCE` is

```
% ./volce3 [OPTION]... <INPUT-FILE> [OUTPUT-FILE]
```

The meanings of the options are given in the following table.

Table 2: Command-line Options of `VolCE`

Option	Meaning
`-P`, `-p`	Enable `PolyVest` for volume approximation. The input of linear inequalities are reals. By default, `VolCE` calls `PolyVest`. And it requires that all the numeric variables are reals (QF_LRA logic).
`-V`, `-v`	Enables `Vinci` for volume computation. The input variables in the linear inequalities are reals (QF_LRA logic).
`-L`, `-l`	Enables `LattE` to count the number of integer solutions. The input variables in the linear inequalities should be integers. This option is usually enabled in the case of integer variables (QF_LIA logic).
`-w={0, 1, ...}`	Specify the word length of numeric variables in bit-wise. Then each variable is automatically bounded by the range $[-2^{w-1}, 2^{w-1}-1]$. Set word length to 0 will disable this feature and the default value is 0.
`-epsilon=real`	VolCE with PolyVest can approximate the volume with (ϵ, δ)-bound, i.e., the result lies in the interval $[(1+\epsilon)^{-1}\#F, (1+\epsilon)\#F]$ with probability at least $1-\delta$. So this parameter designates the value of ϵ. The default value is 0.5.
`-delta=real`	This parameter designates the value of δ. It should be a real in the domain $(0, 1)$. The default value is 0.1.
`-frw=real`	This option sets the weight of first round of estimation, and it only works while `PolyVest` is enabled. Generally, the larger weight, the more accurate, however, the slower. This value should be a real in the domain $(0, 1]$. The default value is 0.01.
`-fact={0, 1}`	Enable (1) or disable (0) the bunch strategy. By default, this strategy is enabled.
`-bunch={0, 1}`	Enable (1) or disable (0) the factorization strategy. It can be very efficient for problems whose variables are less connective. By default, this strategy is enabled.
`-verb={0, 1}`	The verbosity of output. Positive value will enable pretty print. Otherwise, only print the final result. The default value is 1.

5 Examples

`VolCE` has extensive benchmarks which can be found in zip file "benchmarks.zip". In this section, we introduce several examples to describe the usage and the functions of `VolCE`.

Example 1 For the above Formula 1, there are two versions: real variables (`f1_lra.smt2`) and integer variables (`f1_lia.smt2`).

Execute the command for volume estimation and computation:

```
% ./volce3 -P -V benchmarks/lra/f1_lra.smt2
```

And we obtain the result:

```
====================================
============ Parameters ============   Index Volume
====================================   1 0.5
                                       2 0.25
-P Enable PolyVest.
-maxc=1 Set max coefficient to 1.      ====================================
-minc=0.01 Set min coefficient to 0.01.============ PolyVest ==============
-V Enable Vinci.                       ====================================
-w=0 Disabled default bounds.
-bunch=1 Bunch strategy turned on.     FIRST ROUND
-fact=1 Constraints factorization      Index Volume
turned on.                             1 0.202534
-verb=1 Pretty print turned on.        2 0.12575

Input File: "examples/f1_lra.smt2"     SEC & LAST ROUND
                                       Index Coef Volume
VolCE Directory: ...                   1 1 0.544088
Working Directory: ...   ... OK       2 1 0.256463

====================================   ====================================
========== Problem Scale ===========   =========== Statistics =============
====================================   ====================================

Number of Boolean variables: 1         The number of bunches: 2
Number of inequalities: 6              The number of bunches (factorized):0
Number of numeric variables: 2         The number of calls (vol): 6
Number of assertions: 2                The number of vol reuses: 0
                                       The average dims for each call (vol): 2
====================================
=========== SMT Solving ============   ====================================
====================================
                                       The total volume (Vinci): 0.75
#Bunches: 2                            The total volume (PolyVest): 0.800552

====================================   ====================================
============== Vinci ===============
====================================
```

By default, the estimation guarantee is set to $\epsilon = 0.5$ and $\delta = 0.1$. To change this guarantee, simply use `-epsilon` and `-delta` parameters, like:

```
% ./volce3 -epsilon=0.1 -delta=0.01 benchmarks/lra/f1_lra.smt2
```

There is an integer version of this problem. To test it, execute the command:

```
% ./volce3 -L benchmarks/lia/f1_lia.smt2
```

The format of output is similar with options `-P` and `-V`. We can find that the number of solutions is 2. In the following, we present some analysis of these results.

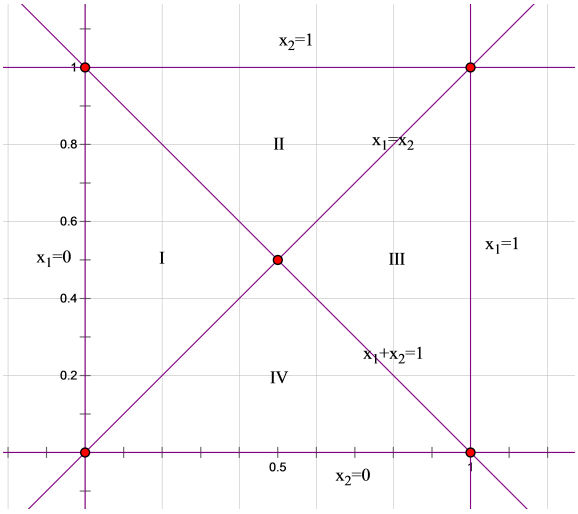


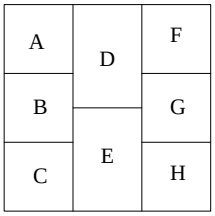
Figure 1: Solution Space of Formula 1

Figure 1 shows the linear constraints in Formula 1. The plane is splitted into 4 areas, since b_4, b_5, b_6, b_7 are always True. The pair $\{b_1, b_3\}$ determines the counted areas.

- Area I: $\{b_1 = \text{True}, b_3 = \text{True}\}$. It has no lattice points.
- Area II: $\{b_1 = \text{True}, b_3 = \text{False}\}$. It has 1 lattice point: $\{0, 1\}$.
- Area III: $\{b_1 = \text{False}, b_3 = \text{False}\}$. It has 2 lattice points: $\{1, 0\}$ and $\{1, 1\}$.
- Area IV: $\{b_1 = \text{False}, b_3 = \text{True}\}$. It has 1 lattice point: $\{0, 0\}$.

There are 3 Boolean solutions for Formula 1: $\{b_1 = \text{True}, b_2 = \text{True}, b_3 = \text{True}\}$, $\{b_1 = \text{True}, b_2 = \text{False}, b_3 = \text{True}\}$, $\{b_1 = \text{False}, b_2 = \text{True}, b_3 = \text{False}\}$. Thus the volume of solution space is $2 \times vol(Area_I) + vol(Area_{III}) = 0.75$. And there are $2 \times 0 + 2 = 2$ integer solutions (lattice points).

Example 2 Here is an exercise for young pupils: In a square, there are 8 sub-areas. See below. Color them so that the neighboring sub-areas use different colors. How many different coloring schemes are there?



Obviously, this problem can be regarded as counting the solutions of SMT(LAC) formulas. The input consists of the following inequalities:

$$xA \neq xB, \quad xA \neq xD, \quad xB \neq xC, \quad xB \neq xD, \quad xB \neq xE,$$

$$xC \neq xE, \quad xD \neq xE, \quad xD \neq xF, \quad xD \neq xG,$$

$$xE \neq xG, \quad xE \neq xH, \quad xF \neq xG, \quad xG \neq xH.$$

Executing the following command:

```
% ./volce3 -L -w=2 benchmarks/lia/coloring.smt2
```

We can find that there are 768 solutions. The option `-w=2` here claims that the size of each variable's domain is 4.

Example 3 In [3], we described a program called *getop()* and analyze the execution frequency of its paths. For *Path1*, its path condition[1] is:

```
(NOT ((c = 32) OR (c = 9) OR (c = 10))) AND
((c != 46) AND ((c < 48) OR (c > 57)))
```

Here `c` is a variable of type `char`; it can be regarded as an integer variable within the domain [-128..127].

For the above path condition, we can compute the number of solutions by executing the command:

```
% ./volce3 -L -w=8 benchmarks/lia/program_analysis/getopPath1.smt2
```

[1]The path condition is a set of constraints such that any input data satisfying these constraints will make the program execute along that path.

We find that the path condition has 242 solutions. (We do not need to use the option `-w=8`, because the default word length is 8.)

Given that the size of the whole search space is 256, we conclude that the frequency of executing *Path1* is about 0.945 (i.e., 242/256). This means, if the input string has only one character, most probably, the program will follow this path.

Another path, *Path2*, has the following path condition:

```
((c0 = 32) OR (c0 = 9) OR (c0 = 10)) AND
(NOT ((c1 = 32) OR (c1 = 9) OR (c1 = 10))) AND
(NOT ((c1 != 46) AND ((c1 < 48) OR (c1 > 57)))) AND
(NOT ((c2 >= 48) AND (c2 <= 57))) AND
(NOT (c2 = 46))
```

Given this set of constraints, and using `LattE`, our tool tells us that the number of solutions is 8085. The executed command is:

```
% ./volce3 -L -w=8 benchmarks/lia/program_analysis/getopPath2.smt2
```

So the path execution frequency is $8085/(256 * 256 * 256)$ which is roughly 0.00048.

Example 4 Hoare's program FIND takes an array A[N] and an integer as input, and partitions the array into two parts.

Assume that $N = 8$. We may extract two execution paths from the program, and generate the path conditions. The first path condition is the following:

```
(A[0] < A[3]);  !(A[1] < A[3]);  (A[3] < A[7]);
!(A[3] < A[6]);  !(A[2] < A[3]);  !(A[3] < A[5]);
!(A[3] < A[4]);  (A[0] < A[4]);  (A[6] < A[4]);  (A[5] < A[4]).
```

Setting the word length to 4, we can find that the number of solutions is 4075920. The executed command is:

```
% ./volce3 -L -w=4 benchmarks/lia/program_analysis/FINDpath1.smt2
```

The second path condition is a bit more complicated:

```
!(A[0] < A[3]);  (A[3] < A[7]);  (A[3] < A[6]);
(A[3] < A[5]);  (A[3] < A[4]);  !(A[1] < A[3]);
(A[3] < A[2]);  (A[3] < A[1]);  (A[1] < A[0]);
(A[2] < A[0]);  !(A[0] < A[7]);  !(A[4] < A[0]);
(A[0] < A[6]);  !(A[0] < A[5]);  (A[1] < A[7]);
(A[2] < A[7]);  !(A[7] < A[5]);  !(A[1] < A[5]);
!(A[2] < A[5]);  (A[5] < A[2]);  (A[2] < A[1]).
```

Executing the command:

```
% ./volce3 -L -w=4 benchmarks/lia/program_analysis/FINDpath2.smt2
```

we find that the number of solutions is 87516. So, the first path is executed much more frequently than the second one. (We assume that the input space is evenly distributed.)

References

[1] C. Ge, F. Ma and J. Zhang. A fast and practical method to estimate volumes of convex polytopes. Dec. 2013. http://arxiv.org/abs/1401.0120v1

[2] LattE, available at https://www.math.ucdavis.edu/~latte/

[3] F. Ma, S. Liu and J. Zhang. Volume computation for Boolean combination of linear arithmetic constraints. In: *Proc. CADE-22*, LNCS 5663, pp.453–468, 2009.

[4] SMT-LIB: The Satisfiability Modulo Theories Library. http://www.smt-lib.org/

[5] Vinci, available at
http://www.math.u-bordeaux1.fr/~aenge/?category=software&page=vinci