VERIFYING MILP CERTIFICATES WITH SMT SOLVERS

Optimizing Learning Activities in EdTech

Executive Summary

Objective and Innovation

The paper presents an SMT-based tool for verifying VIPR certificates, ensuring the correctness of solutions from mixed-integer linear programming (MILP) solvers by utilizing the relationship between VIPR certificate correctness and specific satisfiability formulas.

Significance

MILP is essential for solving complex optimization problems, making the verification of its results critical, particularly in high precision and correctness-dependent fields.

Technical Approach

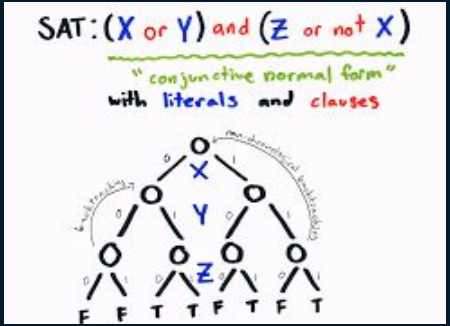
Describes how MILP problems and VIPR certificates are transformed into quantifier-free linear real/integer arithmetic problems for verification using SMT solvers, through translation into SMT-LIB format.

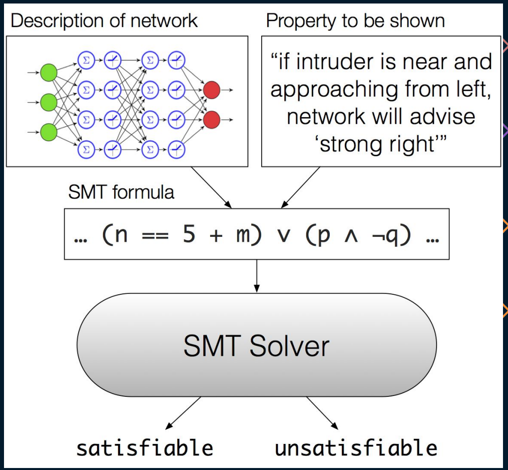
Evaluation and Results

Testing on benchmark instances showed that the VIPR certificate checker is effective and can offer performance advantages over traditional methods in some scenarios.

Background

SAT Solver returns satisfiability (T/F) on boolean formula input.
SMT Solver extends it by handling num, inequalities and logics.
SMT Checker uses SMT Solver to check if solution follows rules.
It works on logical operators between formula unlike MC states
eg check if partially filled Sudoku can be completed using rules





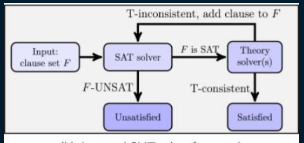
Mixed Integer Linear Programming (MILP) Solver provides optimal solutions for mixed (whole objects only) integer linear (min / max) problems involving both discrete and continuous variables
eg the best way to spend $10 to buy sugar, lemons and water to make best lemonade

Let us assume we have **3** games on this machine
For every ***x*** coin, you get to play ***2x*** minutes of **G1**
For every ***y*** coin, you get to play ***3y*** minutes of **G2**
For every ***z*** coin, you get to play ***5z*** minutes of **G3**

To "***optimize***" your gaming time, you need to decide coins for which games to use more i.e., you need to find the optimal values of **x**, **y** and **z** and see how much "***change***" that brings to your "***overall***" gaming time/ experience

MILP Solvers are used in experimental maths to find counter-egs CFT and math reasoning

Neural-based (partial derivatives) models like LLMs despite displaying emergent abilities of reasoning and decision-making need pre-training on big data yet hallucinate & reversal curse



(b) A general SMT solver framework

$$\nabla_\theta(f) = \begin{bmatrix} \frac{\partial f(\theta)}{\partial x} \\ \frac{\partial f(\theta)}{\partial y} \\ \frac{\partial f(\theta)}{\partial z} \end{bmatrix}$$

However, there wasn't a reliable way to verify the correctness of MILP answers.
VIPR certificate format acts as proof for MILP's solution but can't be understood by SMT Solver

The hypothesis of the paper is encoding QF_LIRA translation of MILP constraints into series of numbers, inequalities & logics, SMT Solver can understand, to check if there exists solution (variable assignment) that follows all the rules and constraints encoded in the statements to validate the VIPR

Experiments and Results

[Fig 1] MILP-MPS files encode information in four sections:

1. ROWS specifies type of each constraint (>=, <=, or =)
2. RHS specifies the right hand side value of each constraint
3. COLUMNS specifies for variable its coefficient in each constraint
4. BOUNDS specifies upper and lower bounds of each variables

[Fig 1] MILP to SMT Transformation Steps:

1. Variable specified in COLUMNS will be encoded as declare-fun
2. ROWS, RHS, COLUMNS encode information on each constraint
3. Each such constraint is transformed into an assert statement
4. COLUMNS, BOUNDS encode lower/upper bounds for variables
5. Each such bound is transformed into assert statement

[Fig 2] VIPR certificate contains two main parts:

1. describes original MILP problem (constraints, variable types etc)
2. describes reasoning (assumptions and inferences constraints)

[Fig 2] VIPR to SMT Transformation Steps [see Methodology slide]

Design of derivations in VIPR is based on branch-and-cut and split-cuts principles in MILP domain. VIPR "flattens" branch-and-bound tree and renders it into series of statements that can be verified sequentially

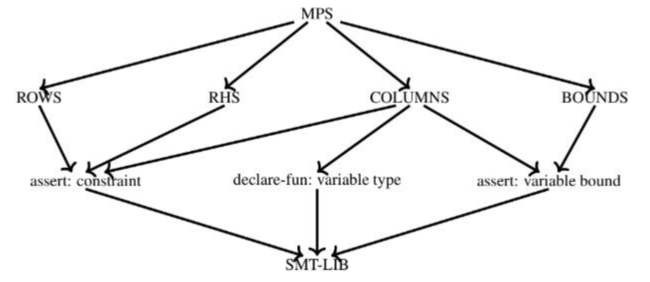


Figure 1: Transformation from MILP instance (MPS format) to SMT instance (SMT-LIB format)

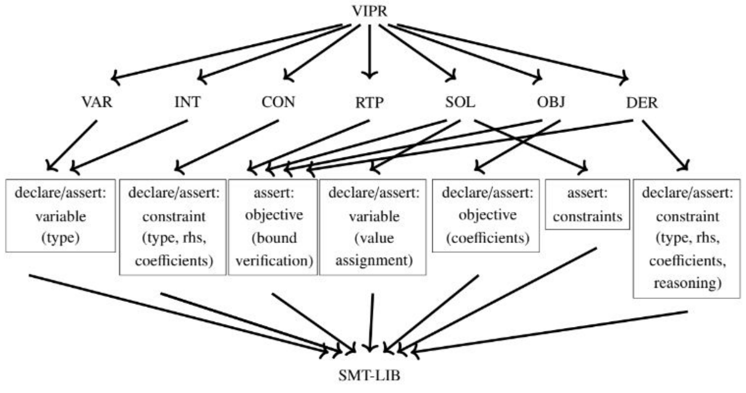


Figure 2: Transformation from VIPR format to SMT-LIB format

Experiments and Results Summary

Common File Formats for MILP:-

1. LP: human-readable and resembles linear constraints algebraically
2. MPS: machine-friendly with higher precision for constraint coefficients

Paper’s translation tool accepts MILP instances in MPS and produces equivalent SMT instances in SMT-LIB

MPS Sections:-

1. ROWS: Constraint types (≤, ≥, =)
2. RHS: Right-hand side values of constraints (assumed constant)
3. COLUMNS: Coefficients of each variable in each constraint
4. BOUNDS: Upper and lower bounds for each variable

Transforming MILP into VIPER:-

1. Variables: Each variable is declared using declare-fun statements.
2. Constraints: Each is translated into assert statement capturing specified inequality or equality

Serves as a starting point for evaluating effectiveness for verification instead of directly solving the instance

Experiments and Results Summary

Components of VIPER:-

1. System Part: describes the original MILP problem
2. Inference Part: explains reasoning behind proof

Transforming VIPER into SMT:

1. Fixed Variables:
   SMT solvers used for symbolic computations based on reasoning within certificate
   To achieve this, the checker assigns a fixed variable in SMT-LIB format to each constant (coefficient)
2. System Part Transformation:
   Asserts not used to directly check if each constraint holds. Coefficients represented as fixed variables
3. Inference Part Transformation:
   Coefficients in each inferred constraint are also encoded
   Reasoning steps are translated into assert statements verifying inequalities between constants

Methodology

Detailed Transformation of VIPER into SMT-Lib:

1. VAR & INT sections:
   Define variable types (real or integer)
   Encoded using declare-fun and assert statements (example provided)
2. CON section:
   Specifies constraints
   Each coefficient is declared as a real variable and assigned a value using declare-fun and assert
3. SOL section:
   Represents variable assignments provided by the MILP solver
   Verified using assert statements to ensure they satisfy constraints from the CON section
4. OBJ section:
   Defines the objective function coefficients
   Similar to CON translation, coefficients are encoded as real variables with declare-fun and assert
5. DER section:
   Contains information about inferred constraints and their reasoning
   Similar to the CON translation, with additional assert checking the validity of inferred constraints

SOL is valid if it contains at least one solution that satisfies all constraints in the CON and bound in RTP, or contains no solutions. RTP section reports infeasibility. DER is valid if every listed constraint in is derived from previously listed constraints in section and last constraint either implies bound in RTP or implies that RTP section reports infeasibility

Key Findings

Proposition 1: Validity of the result stated in RTP of VIPR relies on both SOL and DER sections being valid. This means that to ensure correctness, we must verify both the solution provided by the MILP solver (SOL section) and the inferred reasoning (DER section)

To show that a VIPR certificate is equivalent to its transformation into an SMT instance, we only need to demonstrate the equivalence of the transformed SOL and DER sections:

Lemma 1: SOL is valid if transformed statements in SMT-LIB don't result in contradictions
Lemma 2: DER is valid if transformed statements in SMT-LIB don't result in contradictions

Theorem 1 (consolidates these findings):
Result stated in RTP of VIPR is valid if statements in SMT-LIB don't lead to contradictions

	Base		VIPR2SMT-cvc5			VIPR2SMT-Z3			MPS2SMT-cvc5		MPS2SMT-Z3	
	Ver.	Time	Ver.	Trans.	Time	Ver.	Trans.	Time	Ver.	Time	Ver.	Time
Easy	46	155.6	31	30.8	890.2	26	45.9	1886.9	14	2002.5	6	2243.9
Hard	9	34.6	7	59.1	1235.1	5	12.1	47.0	4	65.5	3	62.9

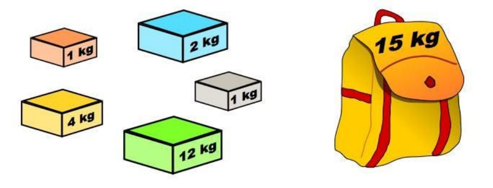
Table 1: Aggregated computational results for three ways to verify MILP results: check VIPR by original C++ checker (Base), transform VIPR to SMT and solve with cvc5/Z3 (VIPR2SMT), and transform MPS file to SMT and solve with cvc5/Z3 (MPS2SMT). Ver. column represents the number of instances solved within time and memory limit (4 hours and 256G RAM). Trans. column represents the average time on VIPR2SMT transformation for the solved instances. Time column represents the average time (transformation time plus solver time) for the solved instances.

Key Findings Analogy

Mystery: Goal is to buy candy to maximize sugar rush with allowance
Clues: Information about candy prices, budget, and buying restrictions
Code: The clues are written in a secret mathematical language called MILP

Introducing the Super Decoder: SMT solver understands the code and translates it for person
Simplifying it for SMT Solver: People created translator that turns code (VIPR) into simpler one
Special Rules: SMT solver follows rules (QF_LIRA) to understand numbers, math and restrictions
Solving the Mystery: Once decoded, SMT solver helps find best candy combo within the clues

Proposition: Think of the proposition as a crucial clue. It states that the solution stated in the VIPR certificate is correct only if both the solution (SOL) and reasoning (DER) sections are valid
Lemma 1 : check if shopping list (SOL) adds up correctly without any mistakes
Lemma 2: ensure reasons for picking certain candies (DER) make sense and don't contradict
Theorem: double-checking candy choices with another person
"If both agree shopping list and reasons for picking candies make sense, then choices are correct



Key Discussion Points of Verifying MILP Certs with SMT Solvers

- Transformation aims to minimize workload on SMT solvers by focusing on symbolic reasoning within cert
- Most verification involves checking formulas with fixed variables, reducing computational burden
- VIPR2SMT approach successfully verified the validity of most
- VIPR2SMT outperforms MPS2SMT in terms of the number of solved instances
- An instance solved by MPS2SMT but not VIPR2SMT suggests potential complementarity
- Good success rate, exceeding a direct translation of the original MILP problem (MPS2SMT) to SMT-LIB

Limitations

1. **Computational Complexity:** especially for large-scale problems in real world
2. **Limited Scope of Verification:** Doesn't guarantee solution is globally optimal
3. **Error Sensitivity:** Errors in translation can lead to misleading verification result
4. **Transfer Learning:** Optimization problems may require significant modification
5. **Data Requirements:** Lack of sufficient data could hinder the verification

Open-Questions

1. How can they augment Planning (PRIMA) and Decision-Making (LDM^2) agents?
2. Can there be a feedback mechanism implemented to improve them like model-based RL?