Generalized Linear Integer Numeric Planning

Xiaoyou Lin[1], Qingliang Chen [1], Liangda Fang[1,3,4*], Quanlong Guan[1,2*], Weiqi Luo[1], Kaile Su[5]
[1] Jinan University, Guangzhou, China [2] Institute of Smart Education of Guangdong, Guangzhou, China [3] Pazhou Lab, Guangzhou, China
[4] Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin, China [5] Griffith University, Brisbane, Australia

Motivation

- The representative framework of classical planning is based on propositional logic. Due to the weak expressiveness of propositional logic, many applications of interest cannot be formalized as classical planning problems. Some extensions to classical planning are proposed. One extension is numeric planning. Another extension is generalized planning (GP).
- Qualitative Numeric Planning (QNP) is a decidable class of numeric and generalized extensions and serves as a numeric abstraction of GP. However, QNP is still far from being perfect and needs further improvement.

Main Results

- Introduce another generalized version of numeric planning, namely Generalized Linear Integer Numeric planning (GLINP), which is a more expressive abstract framework of GP than QNP.
- Develope an approach to synthesizing planning programs inductively from a set of state-plan pairs.
- Evaluate above approach **RegexSkeleton** on several benchmarks originated from generalized planning and qualitative numeric planning. The experimental results justify the feasibility and effectiveness of the proposed approach.

Generalized Linear Integer Numeric Planning (GLINP)

Definition 1. A LINP domain D is a tuple $\langle P, V, A\rangle$ where

- P: a finite set of propositional variables;
- V: a finite set of numeric variables;
- A: a finite set of actions defined by a pair $\langle pre, eff\rangle$ where $pre \in Form$ denotes the precondition and eff is an finite set of propositional and numeric effects.

A LINP problem is defined as a tuple $\langle D, s_I, g\rangle$ where D is an LINP domain, s_I is an initial state, and $g \in Form$ denotes a set of goal states.

Definition 2. A generalized LINP (GLINP) problem Σ is a tuple $\langle D, I, G\rangle$, where

- D: a LINP domain
- $I \in Form$: a formula denoting a set of initial states;
- $G \in Form$: a formula denoting a set of goal states.

Each LINP problem $\langle D, s_I, G\rangle$ is an instance of GLINP problem $\langle D, I, G\rangle$ where s_I is an initial state, that is, $s_I \vDash I$.

Planning Program

Definition 3. The set of planning programs (Prog) for an LINP domain D = ⟨P , V , A⟩ is recursively defined by

$$\delta \in Prog ::\varepsilon \mid a \mid \delta;\delta \mid \textbf{if } \phi \textbf{ then } \delta \textbf{ else } \delta \textbf{ fi} \mid \textbf{while } \phi \textbf{ do } \delta \textbf{ od}$$

where $a \in A$, $\phi \in Form$.

Given a GLINP problem, we are interested in synthesizing a program δ satisfying the following three properties.

Definition 7. Let $D = \langle P, V, A\rangle$ be an LINP domain, δ a program for D and s a state. The planning program δ is

- terminating in s, iff $\Theta(s,\delta)$ is finite;
- executable in s, iff $\Theta(s,\delta)$ is executable in s;
- ϕ-reaching in s, iff δ is terminating and executable in s only if $\tau(s,\delta) \vDash \phi$

where $\Theta(s,\delta)$ denotes the action sequence of executing δ in s, $\tau(s,\delta)$ *denotes* the successor state of applying an action a over s.

The Main Framework

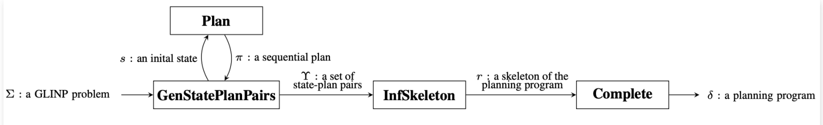
Plan
s : an inital state
π : a sequential plan
Σ : a GLINP problem
GenStatePlanPairs
Υ : a set of state-plan pairs
InfSkeleton
r : a skeleton of the planning program
Complete
δ : a planning program

Figure 1: The Main Framework

The main framework consists of four essential components: *GenStatePlanPairs*, *InfSkeleton*, *Complete* and *Plan*.

- ***GenStatePlanPairs***: generate a set Υ of state-plan pairs, including generating initial states s by imposing some restrictions on propositional and numeric variables and computing the corresponding solutions π by the planner;
- ***InfSkeleton***: infer a skeleton of the planning program r expressed as a regex according to the plans of Υ ;
- ***Complete***: obtain a complete planning program δ by filling the missing conditions in r according to Υ ;
- ***Plan*** : implemented by directly invoking existing numeric planners , e.g., Metric-FF.

Experimental Evaluation

Domain	RegexSkeleton			QNP2FOND	
	$\#(\delta)$	$\lvert\delta\rvert$	t	$\lvert p\rvert$	t
Arith	1	13	0.097	-	-
Chop	1	5	0.028	5	0.039
ClearBlock	1	7	0.027	15	0.088
Corner-A	1	11	0.048	29	0.226
Corner-R	1	13	0.064	-	-
Delivery	2	31	5.161	229	65.31
D-Return	1	41	0.215	-	-
D-Return-R	2	52	325.2	-	-
Gripper	2	31	5.571	201	112.9
Hall-A	1	37	0.135	-	-
Hall-R	1	45	0.166	-	-
NestVar	1	13	0.030	48	0.501
NestVar8	1	541	6.315	TO	TO
MNestVar	2	11	0.034	-	-
MNestVar8	8	287	311.8	-	-
PlaceBlock	1	23	0.045	71	3.234
Rewards	2	11	0.046	28	0.232
Snow	1	21	0.037	115	3.822
Spanner	1	21	0.094	-	-
TestOn	1	13	0.062	41	0.433
VisitAll	2	37	0.129	-	-
VisitAll-R	2	51	0.232	-	-

Table 2: Experimental results. "#δ", "$|\delta|$" and "$|p|$" are the depth and the length of the planning program, and the length of the policy, respectively; and "t" is the running time of synthesizing the planning programs and the policy. "TO" denotes timeout; "-" indicates that the problem cannot be formalized in QNP.

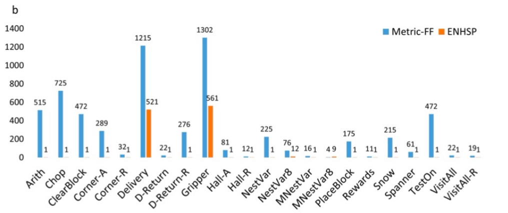

Figure 2: The threshold of bound b for each domain when RegexSkeleton outperforms Metric-FF and ENHSP in efficiency, respectively.

Results:

1. RgexSkeleton outperforms QNP2FOND in both the efficiency of solving problems and the number of solving problems.
2. RegexSkeleton outperforms Metric-FF and ENHSP in efficiency when the bound b exceeds a certain threshold.

Future Works

- Extend above approach to handle more complex GLINP problems rather than a class of GLINP problems where the goal is to decrease the values of some numeric variables.
- Identify the decidable fragment for the existence of solutions to GLINP in future work.