Optimal Strategy Schedules for Everyone

Hans-Jörg Schurr[1]

[1]University of Lorraine, CNRS, Inria, and LORIA

August 11, 2022
PAAR 2022 — Haifa, Israel

- Multiple tools to work with *static* strategy schedules
- Can generate schedules
- Focus on simplicity and stability
- Implemented in Python
  - with few extra dependencies
- Available at `https://gitlab.uliege.be/verit/schedgen`

Motivation

Welcome to the veriT team!
Now the noble task of submitting veriT to the SMT competition falls to you.

Motivation

Great!
How do I update the strategy schedule?

Motivation

I use my profound knowledge of veriT and careful study of the strategies to build schedules by hand.

Motivation

Motivation

I wonder if I could use
integer programming
to generate optimal schedules.

Motivation

My three day hack works well,
now I will spend weeks to add
features and to polish everything.

What is a strategy?

- A *strategy* is a full parameterization of the system
- For an SMT solver:
  - select preprocessing methods
  - select instantiation procedures
  - set limits for instantiation procedures
  - ...

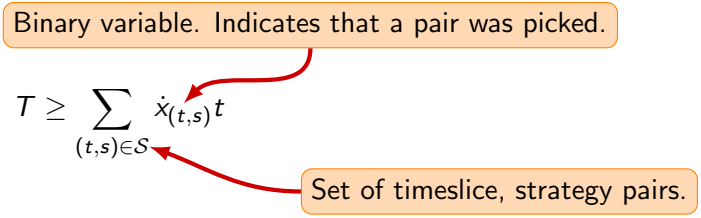
What is a strategy schedule?

- A finite list $[(t_1, s_1), \ldots, (t_n, s_n)]$
- t_i are time limits
- $s_i \in S$ are strategies
- $\sum_i t_i \leq T$ is the total timeout
- We require that the t_i are from finite set TS of allowed time slices
- In the following $\mathcal{S} = \mathrm{TS} \times S$
- Furthermore, we have training benchmarks (denoted b)

What is a strategy schedule?

- A finite list $[(t_1, s_1), \ldots, (t_n, s_n)]$
- t_i are time limits
- $s_i \in S$ are strategies
- $\sum_i t_i \leq T$ is the total timeout
- We require that the t_i are from finite set TS of allowed time slices
- In the following $\mathcal{S} = \mathrm{TS} \times S$
- Furthermore, we have training benchmarks (denoted b)

Encoding


Binary variable. Indicates that a pair was picked.
$T \geq \sum_{(t,s)\in\mathcal{S}} \dot{x}_{(t,s)} t$
Set of timeslice, strategy pairs.

Encoding

$$T \geq \sum_{(t,s)\in\mathcal{S}} \dot{x}_{(t,s)}\, t$$

$$\dot{x}_s = \sum_{1\leq i\leq n} \dot{x}_{(t_i,s)}$$

Pick each strategy only once.

Encoding

$$T \geq \sum_{(t,s)\in\mathcal{S}} \dot{x}_{(t,s)}\, t$$

$$\dot{x}_s = \sum_{1\leq i\leq n} \dot{x}_{(t_i,s)}$$

$$x_b = \sum_{\dot{x}\in X_b} \dot{x} \text{ with } X_b := \left\{\dot{x}_{(t,s)} \mid (t,s) \in \mathcal{S} \text{ and } s \text{ solves } b \text{ in time } \leq t\right\}$$

Set of pairs that solve the benchmark b.

How often is the benchmark b solved?

Encoding

$$T \geq \sum_{(t,s)\in\mathcal{S}} \dot{x}_{(t,s)}\, t$$

$$\dot{x}_s = \sum_{1\leq i\leq n} \dot{x}_{(t_i,s)}$$

$$x_b = \sum_{\dot{x}\in X_b} \dot{x} \text{ with } X_b := \left\{\dot{x}_{(t,s)} \mid (t,s) \in \mathcal{S} \text{ and } s \text{ solves } b \text{ in time} \leq t\right\}$$

$$\dot{x}_b|X_b| \geq x_b$$ Force $\dot{x}_b$ to 1 if $x_b > 1$.

$$\dot{x}_b \leq x_b + 0.5$$ Force $\dot{x}_b$ to 0 if $x_b = 0$.

Encoding

$$T \geq \sum_{(t,s)\in\mathcal{S}} \dot{x}_{(t,s)}\, t$$

$$\dot{x}_s = \sum_{1\leq i\leq n} \dot{x}_{(t_i,s)}$$

$$x_b = \sum_{\dot{x}\in X_b} \dot{x} \text{ with } X_b := \left\{\dot{x}_{(t,s)} \mid (t,s) \in \mathcal{S} \text{ and } s \text{ solves } b \text{ in time} \leq t\right\}$$

$$\dot{x}_b|X_b| \geq x_b$$

$$\dot{x}_b \leq x_b + 0.5$$

$$\text{maximize} \sum_{b\in B} \dot{x}_b$$

Count the solved benchmarks.

What's in the box?

- `schedgen-optimize` – generate schedules
- `schedgen-finalize` – generate scripts from a schedule and a template
- `schedgen-simulate` – calculate the benchmarks solved by a schedule
- `schedgen-query` – list unsolved benchmarks, compare schedules
- `schedgen-visualize` – inspect a schedule visually

What's in the box?

- `schedgen-optimize` – generate schedules
- `schedgen-finalize` – generate scripts from a schedule and a template
- `schedgen-simulate` – calculate the benchmarks solved by a schedule
- `schedgen-query` – list unsolved benchmarks, compare schedules
- `schedgen-visualize` – inspect a schedule visually

What's in the box?

- `schedgen-optimize` – generate schedules
- `schedgen-finalize` – generate scripts from a schedule and a template
- `schedgen-simulate` – calculate the benchmarks solved by a schedule
- `schedgen-query` – list unsolved benchmarks, compare schedules
- `schedgen-visualize` – inspect a schedule visually

What's in the box?

- `schedgen-optimize` – generate schedules
- `schedgen-finalize` – generate scripts from a schedule and a template
- `schedgen-simulate` – calculate the benchmarks solved by a schedule
- `schedgen-query` – list unsolved benchmarks, compare schedules
- `schedgen-visualize` – inspect a schedule visually

What's in the box?

- `schedgen-optimize` – generate schedules
- `schedgen-finalize` – generate scripts from a schedule and a template
- `schedgen-simulate` – calculate the benchmarks solved by a schedule
- `schedgen-query` – list unsolved benchmarks, compare schedules
- `schedgen-visualize` – inspect a schedule visually

Walkthrough: Input Data

```
benchmark    ; logic ; strategy       ; solved ; time
base01.smt2 ; UF    ; base-strategy ; yes    ; 0.5189
base02.smt2 ; UF    ; base-strategy ; yes    ; 0.2164
base03.smt2 ; UF    ; base-strategy ; yes    ; 0.1754
...
```

This is artificial example data. All exampes are included in the source code repository.

Walkthrough: schedgen-optimize

```
$ schedgen-optimize.py
    -l UF --epsilon 0.1 -t 6 \
    -s 0.5 1.0 2 3 4 5 6 \
    -c -d contrib/example_data.csv \
        contrib/example_schedule.csv
```

```
$ schedgen-optimize.py
    -l UF --epsilon 0.1 -t 6 \
    -s 0.5 1.0 2 3 4 5 6 \
      --pre-schedule one_second_schedule.csv \
      --pre-schedule-time 1 \
    -c -d contrib/example_data.csv \
        contrib/example_schedule.csv
```

Walkthrough: Generated Schedule

```
time  ; strategy
1.100 ; base-strategy
1.000 ; extra01
0.900 ; extra02
...
```

Walkthrough: schedgen-finalize

```
$ schedgen-finalize.py
    -l UF -t 6 \
    -s contrib/example_schedule.csv \
    --executable ./veriT \
    contrib/scheduler_template schedule.sh
```

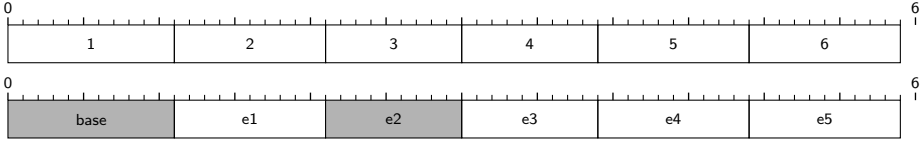
```
...
case "$logic" in
  {% for logic in logics %}
  {{ logic }})
  {% for time,strategy in schedules[logic] -%}
    {%- if loop.last -%}
      finishwith {{ strategy }}
      ;;
    {% else %}
      trywith {{ (time*1000)|int }} {{ strategy }}
    {% endif -%}
  {%- endfor -%}{%- endfor %}
esac
...
```

Walkthrough: schedgen-finalize

```
case "$logic" in
  UF)
      trywith 1100 base-strategy
      trywith 1000 extra01
      trywith 900 extra02
      ...
      finishwith extra5
      ;;
esac
```

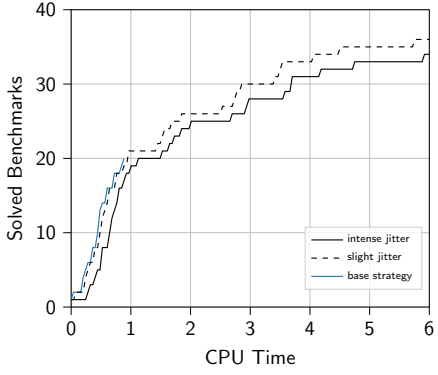
Walkthrough: Visualize

```
$ schedgen-visualize.py -t 6 -p out.pgf \
    -a contrib/example_shorthand.csv \
       contrib/example_schedule.csv
```



Walkthrough: Simulate

```
$ schedgen-simulate.py -l UF -t 6 \
    -c -d contrib/example_data.csv \
    --mu 0.05 --sigma 0.01 --seed 1 \
    contrib/example_schedule.csv simulation_1.txt
```



Walkthrough: Query

```
$ schedgen-query.py -c -d contrib/example_data.csv \
      -q unsolved contrib/example_schedule.csv
    special01.smt2
    unsolved.smt2
```

- `compare` Solved by virtual best solver, but not the schedule
- `best` Virtual best solver (score and solved benchmarks)
- `schedule` Schedule performance (score and solved benchmarks)

Does it work?

- SMT-COMP 2020, 2021, 2022
- Isabelle/HOL `smt` tactic: best strategy, three complementary strategies
  - Best: only timeslice is 3 s, generate 3 s schedule
  - Complementary: same, but 9 s schedule,
- Evaluate new features: generate schedules with and without

Does it work?

Solved	Split 1	Split 2	Split 3	Split 4	Split 5	Arith. Mean	(σ)
virtual best	1355	1318	1328	1293	1338	1326	(23.1)
generated	**1349**	**1306**	**1317**	**1283**	**1326**	**1316**	**(24.4)**
greedy	1340	1303	1314	1275	1326	1312	(24.7)
best strategy	1311	1267	1280	1243	1299	1280	(26.7)
PAR-2 score						Arith. Mean	(σ)
virtual best	160 501	174 213	170 347	182 938	167 371	171 074	(8 316)
generated	**164 388**	**179 811**	**175 453**	**187 851**	**172 102**	**175 921**	**(8 736)**
greedy	169 183	183 040	178 817	192 482	173 655	179 435	(8 974)
best strategy	176 844	192 438	187 772	201 248	180 966	187 854	(9 606)

9000 benchmarks. Five splits of 7200 training benchmarks and 1800 evaluation benchmarks.

Future Work

- `mach'ma`
  - Parallelizing schedule runner.
  - Idea: use `cgroups` to handle memory contention.
- Tool to find promising strategies.
  - Well researched field.
  - I don't want to reinvent the wheel, but would fit well into the toolbox.
- Out of scope: strategy selection based on benchmark features.

Thank you for Your Attention!

`https://gitlab.uliege.be/verit/schedgen`

I am happy about feedback and bug reports.

Order

- The order of the strategies in the schedule is nondeterministic
- Best effort order: pick pair with lowest cost
- Lowest cost: sum of solving time plus solving time by virtual best solver for unsolved benchmarks