Latex Showcase (Work In Progress)

Donghwan Shin

November 2023

1 Figures

Figure 1 shows an example figure using centering.

2 Tables

Table 1 is an example table using an external csv file and multirow. Table 2 is an example table using both an external data, multi-rows, coloured rows, and multi-columns.

3 Listings

Figure 2 shows an example listing using the Python language semantic.

4 Algorithms

It is important to use the type information for each variable (e.g., "Candidate Solution" in front of x_c in line 3) since it significantly improves the readability of the algorithm.

5 Definitions and Theorems

Table 1: Subject Systems and Logs [1]

Source	System	# Cmps	# Logs	# Tpls	# Entries	Conf
LogHub	Hadoop	19	68	41	3575	1.00
	HDFS	8	1000	16	18 741	0.95
	Linux	31	42	115	11 259	0.78
	Spark	11	217	21	67 725	0.98
	Zookeeper	18	36	40	25 298	0.91
PC	CoreSync	54	1418	204	30 223	0.94
	NGLClient	27	42	70	892	0.89
	Oobelib	12	250	147	56 557	0.89
	PDApp	10	787	75	47 394	0.87

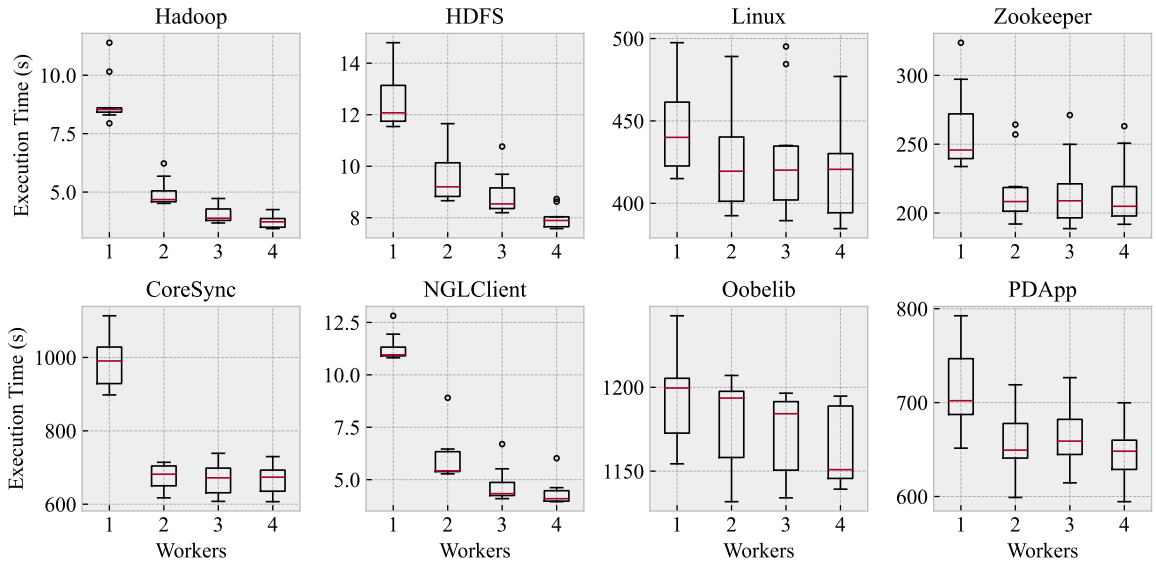

Figure 1: Relationship between the maximum number of parallel workers and the execution time of PRINS [1]

Table 2: Comparison between MINT (M) and PRINS (P) in terms of Accuracy. Differences between recall (Δ_R), specificity (Δ_S), and balanced accuracy (Δ_B) values are expressed in percentage points (pp); $\mathcal{LDS}$ is the log-component diversity score [1].

	Recall			Specificity			Balanced Accuracy			
System	M	P	Δ_R	M	P	Δ_S	M	P	Δ_B	$\mathcal{LDS}$
Hadoop	1.00	1.00	0.0	0.91	0.88	−3.1	0.96	0.94	−1.5	0.015
HDFS	0.98	0.98	−0.1	0.37	0.72	34.9	0.68	0.85	17.4	0.007
Linux	0.36	0.12	−23.5	0.89	0.98	9.5	0.62	0.55	−7.0	0.561
Zookeeper	0.22	0.10	−11.7	0.93	1.00	7.5	0.57	0.55	−2.1	0.571
CoreSync	0.95	0.93	−1.6	0.85	0.89	4.2	0.90	0.91	1.3	0.048
NGLClient	0.86	0.86	0.0	1.00	0.98	−2.5	0.93	0.92	−1.3	0.195
Oobelib	0.98	0.98	0.0	1.00	1.00	0.0	0.99	0.99	0.0	0.016
PDApp	0.97	0.95	−2.3	0.98	0.98	−0.1	0.97	0.96	−1.2	0.014
Average	0.79	0.74	−4.9	0.87	0.93	6.3	0.83	0.83	0.7	0.178

```
(2) db = DB.init(mode="default")
(4) item = getItem(db)
(6) if check(item) is "error":
(7)     logger.error("error in item: %s" % item)
```

Figure 2: Program slice S_r of the program P_{ex} when s_7 and its variable `item` are used as the slicing criterion [2]

Algorithm 1: Simple Random Search (for maximisation)

Input : Input Domain X,
Fitness Function $f : X \to Y$

Output: Best Solution $x_{best} \in X$ such that $x_{best} \approx \text{argmax}_{x \in X} f(x)$

Best Solution $x_{best} \leftarrow$ a candidate solution randomly sampled from X
while x_{best} *is the ideal solution* ***or*** *we have run out of time* **do**
    Candidate Solution $x_c \leftarrow$ a candidate solution randomly sampled from X
    if $f(x_c) > f(x_{best})$ **then**
        $x_{best} \leftarrow x_c$
return x_{best}

Definition 1 (Log Parsing as an Abstraction Function)**.** Given a set of log messages M and a generic set of parsing results A, a log parsing approach can be represented as an abstraction function $\tau \colon M \to A$.

See [3] for more examples (and how to write a theoretical paper).

References

[1] D. Shin, D. Bianculli, and L. Briand, "PRINS: scalable model inference for component-based system logs," *Empirical Software Engineering*, vol. 27, no. 4, p. 87, 2022.

[2] J. H. Dawes, D. Shin, and D. Bianculli, "Towards log slicing," in *International Conference on Fundamental Approaches to Software Engineering*. Springer Nature Switzerland Cham, 2023, pp. 249–259.

[3] D. Shin, Z. A. Khan, D. Bianculli, and L. Briand, "A theoretical framework for understanding the relationship between log parsing and anomaly detection," in *Runtime Verification*, L. Feng and D. Fisman, Eds. Cham: Springer International Publishing, 2021, pp. 277–287.