1 Performances

In order to evaluate the *execution time* performances of our implementation, we made some benchmark tests with different numbers of stages and miners involved in the mining game. Subsequently, we compared the execution time performances of our implementation with the execution time performances of the computation of the mining probability using the Wolfram Mathematica library for the hypoexponential distribution[1]. The latter computes the mining probability through the expression we proposed in the paper.

The tests were executed on Wolfram Mathematica 11.3.0.0. The machine where the tests have been performed was a Notebook HP Pavilion Laptop 15-cs2023nl, with a quad-core CPU Intel Core™ i7-8565U CPU running at 1.80GHz, 8MB cache, and 16GB (2x8) SO-DIMM SDRAM DDR4 running at 2400MHz. The notebook was running Windows 10 Home version 10.0.17763 build 17763.

1.1 Time performance results

M = 2 miners. Table 1 lists the execution time performances with $M = 2$ competing miners and a variable number of stages, starting from 1 up to 7. In table 1, the cell [k, Miner$_z$] contains $\{\lambda_{z,0}, \ldots, \lambda_{z,k-1}\}$ – the positive real λ-parameters relative to the hypoexponential distribution of miner z – $\forall z \in \{1, \ldots, 2\}$ and $\forall k \in \{1, \ldots, 7\}$. Columns $\mathrm{T_{Lib}}$ and $\mathrm{T_{Imp}}$ store the values of the execution times (measured in seconds) of our implementation and the execution times of the Mathematica library, respectively. Column $Norm^{-1}$ is defined as the inverse of $\mathrm{T_{Imp}}$ normalized to $\mathrm{T_{Lib}}$: $Norm^{-1} \stackrel{\text{def}}{=} \mathrm{T_{Lib}}/\mathrm{T_{Imp}}$. This value has been used as a metric to quantify the relative improvements our implementation brings. In doing this, we followed some suggestions pointed out in [1]. Some of the floating point values in the Results columns are truncated.

The table indicates that time the system library takes to compute the mining probability grows significantly when some of the λ-parameters are exponentiations of *transcendental* numbers. This is presumably caused by the integration of an expression composed of these exponentiations that the Mathematica library performs to compute the mining probability. In contrast, our implementation execution time is not affected by λ-parameters of this type.
Finally, table 1 shows that, in the examined examples, our implementation is at least one order of magnitude faster than the Mathematica library.

M = 3, M = 4, and M = 5 miners. Tables 2, 3, and 4 list the execution time performances with a variable number of stages (from 1 up to 7) and, respectively, $M = 3$, $M = 4$, and $M = 5$ miners.

In these tables we focused on positive integer λ-parameters. The tables indicate that, with as the λ-parameters become larger, $\mathrm{T_{Lib}}$ grows more than $\mathrm{T_{Imp}}$. However, with a larger number of miners, the value of $Norm^{-1}$ generally decreases. At the same time, by considering only λ-parameters of value lower than 100, with a larger number of stages, the value of $Norm^{-1}$ slightly decreases.

M = 10 miners. Table 5 lists the execution time performances with $M = 10$ competing miners and a variable number of stages, starting from 1 up to 4. It is noticeable that, with $k \geq 3$, our implementation takes more time than the Mathematica library.

[1] https://reference.wolfram.com/language/ref/HypoexponentialDistribution.html.

Table 1: Execution time with $M = 2$.

k	Miners		Results		
	Miner_1	Miner_2	T_{Imp}	T_{Lib}	Norm^{-1}
1	$\{4\}$	$\{\Pi e^{-3}\}$	0.0043	0.2274	52.1639
2	$\{5, 2/3\}$	$\{e^{2/3}, e^{2/3}\}$	0.0014	0.7934	533.832
3	$\{4^{8/9}, 4^{9/10}, 4^{10/11}\}$	$\{3^{11/12}, 3^{12/13}, 3^{13/14}\}$	0.0114	56.466	4934.66
4	$\{1000, e^{e^{2/3}}, 3 \cdot \Pi^{-4}, 5^2\}$	$\{\Pi/e^{\Pi}, 500, \Pi^{4/5}, 3 \cdot \Pi^{-4}\}$	0.0043	31.448	7284.97
5	$\{10^4, 5 \cdot 10^4, 4 \cdot 10^4, 56666, 3 \cdot 10^6\}$	$\{3 \cdot 10^4, 7 \cdot 10^4, 2 \cdot 10^4, 45555, 4 \cdot 10^6\}$	0.0048	5.6608	1155.37
6	$\{7, 14, 21, 28, 35, 42\}$	$\{8, 16, 24, 32, 40, 48\}$	0.0067	0.3433	51.1336
7	$\{10 \cdot \Pi, 20/3 \cdot \Pi, 5 \cdot \Pi, 3 \cdot \Pi, 2 \cdot \Pi, 67 \cdot \Pi, 6 \cdot \Pi\}$	$\{3 \cdot \Pi, 5 \cdot \Pi, 10 \cdot \Pi, 2 \cdot \Pi, 67 \cdot \Pi, 20/3 \cdot \Pi, 6 \cdot \Pi\}$	0.0119	1.51854	126.74

Table 2: Execution time with $M = 3$.

k	Miners			Results		
	Miner_1	Miner_2	Miner_3	T_{Imp}	T_{Lib}	Norm^{-1}
1	$\{4\}$	$\{4\}$	$\{7\}$	0.0006	0.0374	56.0669
2	$\{5, 40\}$	$\{45, 3\}$	$\{4, 14\}$	0.0022	0.1651	74.8944
3	$\{67, 125, 62\}$	$\{61, 53, 12\}$	$\{2, 100, 70\}$	0.0062	0.3545	56.9656
4	$\{35, 39, 50, 14\}$	$\{35, 39, 50, 14\}$	$\{35, 39, 50, 14\}$	0.0135	0.1938	14.3085
5	$\{146, 753, 134, 224, 566\}$	$\{411, 145, 225, 334, 56\}$	$\{566, 111, 220, 100, 50\}$	0.0238	14.6317	613.528
6	$\{47, 82, 79, 19, 45, 89\}$	$\{62, 37, 62, 58, 56, 81\}$	$\{55, 55, 55, 55, 55, 55\}$	0.0320	5.8112	181.424
7	$\{78, 24, 167, 456, 321, 245, 21\}$	$\{134, 135, 322, 42, 123, 245, 532\}$	$\{135, 146, 220, 532, 214, 256, 155\}$	0.0598	6.7030	111.981

Table 3: Execution time with $M = 4$.

k	Miners				Results		
	Miner_1	Miner_2	Miner_3	Miner_4	T_{Imp}	T_{Lib}	Norm^{-1}
1	{4}	{4}	{7}	{2}	0.0007	0.0357	47.2139
2	{5, 40}	{45, 3}	{4, 14}	{1, 4}	0.0047	0.218	45.9609
3	{67, 125, 62}	{61, 53, 12}	{2, 100, 70}	{56, 90, 70}	0.0179	0.6572	36.5787
4	{35, 39, 50, 14}	{35, 39, 50, 14}	{35, 39, 50, 14}	{25, 57, 45, 1}	0.0484	0.5381	11.0969
5	{146, 753, 134, 224, 566}	{411, 145, 225, 334, 56}	{566, 111, 220, 100, 50}	{32, 56, 164, 156, 94}	0.1098	17.2613	157.152
6	{47, 82, 79, 19, 45, 89}	{62, 37, 62, 58, 56, 81}	{55, 55, 55, 55, 55, 55}	{34, 56, 9, 35, 67, 21}	0.207	23.7886	114.904
7	{78, 24, 167, 456, 321, 245, 21}	{134, 135, 322, 42, 123, 245, 532}	{135, 146, 220, 532, 214, 256, 155}	{63, 142, 135, 324, 256, 256, 237}	0.6023	19.4365	32.27

Table 4: Execution time with $M = 5$.

k	Miners					Results		
	Miner_1	Miner_2	Miner_3	Miner_4	Miner_5	T_{Imp}	T_{Lib}	Norm^{-1}
1	{4}	{4}	{7}	{2}	{6}	0.001	0.036	35.7017
2	{5, 40}	{45, 3}	{4, 14}	{1, 4}	{2, 7}	0.0093	0.311	33.155
3	{67, 125, 62}	{61, 53, 12}	{2, 100, 70}	{56, 90, 70}	{43, 47, 51}	0.0504	1.1728	23.2355
4	{35, 39, 50, 14}	{35, 39, 50, 14}	{35, 39, 50, 14}	{25, 57, 45, 1}	{53, 12, 42, 17}	0.1727	0.8965	5.1885
5	{146, 753, 134, 224, 566}	{411, 145, 225, 334, 56}	{566, 111, 220, 100, 50}	{32, 56, 164, 156, 94}	{266, 136, 356, 245, 134}	0.5164	21.6938	42.0036
6	{47, 82, 79, 19, 45, 89}	{62, 37, 62, 58, 56, 81}	{55, 55, 55, 55, 55, 55}	{34, 56, 9, 35, 67, 21}	{64, 44, 56, 63, 45, 65}	1.3486	42.6996	31.6607
7	{78, 24, 167, 456, 321, 245, 21}	{134, 135, 322, 42, 123, 245, 532}	{135, 146, 220, 532, 214, 256, 155}	{63, 142, 135, 324, 256, 256, 237}	{123, 245, 143, 156, 163, 167, 167}	4.3255	56.1133	12.9727

Table 5: Execution time with $M = 10$.

	Miners					Results		
k	Miner_6	Miner_7	Miner_8	Miner_9	Miner_{10}	T_{Imp}	T_{Lib}	Norm^{-1}
1	{6}	{7}	{8}	{9}	{10}	0.0018	0.036	19.3783
2	{6, 7}	{7, 13}	{8, 14}	{9, 14}	{10, 11}	0.2642	0.8693	3.29
3	{ 6, 7, 54 }	{ 7, 13, 23 }	{ 8, 14, 23 }	{ 9, 14, 31 }	{ 10, 11, 12 }	11.722	2.479	0.211
4	{ 6, 7, 54, 34 }	{ 7, 13, 23, 13 }	{ 8, 14, 23, 34 }	{ 9, 14, 31, 14 }	{ 10, 11, 12, 12 }	250.641	32.874	0.131

2 Conclusion

In conclusion, we argue that with low values of k and M our implementation is more time-efficient than the Mathematica library, while the opposite happens with large values for k and M.
The realization of some implementation techniques to let our package scale better with large values for k and M, together with the benchmarking for other features (such as CPU or memory usage), and the implementations of the mining probability expression for other programming languages are left for future work.

References

[1] P. J. Fleming and J. J. Wallace. "How Not to Lie with Statistics: The Correct Way to Summarize Benchmark Results". In: *Commun. ACM* 29.3 (Mar. 1986), pp. 218–221.