Benchmarks of Dijkstra Search in Basis Marking Space

Ziyue Ma, Minqiang Zou, Jiafeng Zhang, Zhiwu Li

September 10, 2020

In this note we report three benchmarks of Algorithm 1 reported in paper *Determining Optimal Control Sequences for Reconfiguration in Petri Nets* [1] (called "Algorithm 1" for short in the sequel) that is currently *under review*. Results are carried out using our program (in Python 3) on a laptop computer with Intel Core i7-10750H 2.60/2.59GHz CPU and 16G RAM.

1 Benchmark 1

The net in Figure 1 is the net from Figure 2 in paper [1] which models a manufacturing cell that consists of two assembly stations (composed by $\{p_2 - p_{10}, t_1 - t_9\}$ and $\{p_{11} - p_{19}, t_{10} - t_{18}\}$, respectively), one idle place p_1, and one stock $\{t_{19}, p_{20}, t_{20}, p_{21}, t_{21}, p_{22}, t_{22}\}$.

The cost to fire each transition is depicted in the parenthesis next to the transition.

1.1 Parameterized Source Markings

The source marking shown in Figure 1 is parameterized as: $M_0 = \beta \cdot p_1 + \alpha \cdot (p_2 + p_5 + p_{12} + p_{13})$. Suppose that the assembly station on the left ($\{p_2 - p_{10}, t_1 - t_9\}$) has some fatal error. Hence, a reconfiguration request with target marking set

$$\mathcal{M}_{target} = \{M \mid M(p_{19}) + M(p_{21}) \geq 2\alpha + \beta\}$$

is initiated, i.e., all materials in the system must be transferred to either the stock (place p_{21}) or the output buffer of the station on the right (place p_{19}). On the other hand, there is a control specification

$$\mathcal{M}_F = \{M \mid M(p_{15}) \geq 2\}$$

that requires that during the reconfiguration process place p_{15} must hold at most one token.

1.1.1 Comparison of Algorithm 1 with Transition-wise Dijkstra Search

According to Conditions 1 and 2 in [1], we choose a basis partition $\pi = (T_E, T_I)$ with

$$T_E = \{t_9, t_{15}, t_{17}, t_{20}\}, \quad T_I = T \setminus T_E.$$

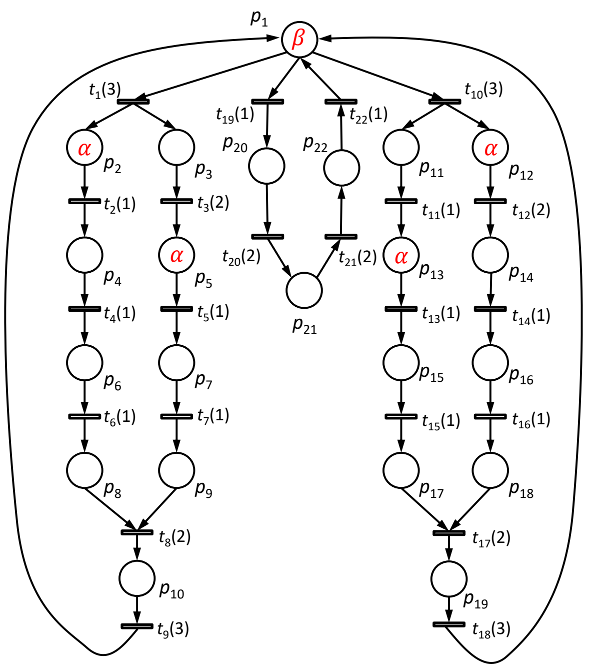

Figure 1: The Petri net for Benchmark 1.

The simulation results for the Dijkstra search (i.e., Algorithm 1) in [1] are summarized in Table 1. Our Dijkstra search in [1] terminates and returns an optimal control sequence in a reasonable time even for nets whose state space is very large. For instance, for the case $\alpha = 6$, a D-BRG with size 6236 is constructed in about 10 seconds, and an optimal control sequence that consists of 102 transitions is obtained.

On the other hand, column 3 of Table 1 summarizes the time of a conventional transition-wise Dijkstra search. Such transition-wise search runs out of time[1] when $\alpha \geq 2$ (column 3).

$\alpha = \beta$	$\|RG\|$	Time(s)	Dijkstra Search Space	Time(s)	$\|\sigma_{min}\|$	c_{min}
1	7464	8.995	24	0.015	17	24
2	2092528	o.t.	153	0.086	34	48
3	o.s.	o.t.	547	0.320	51	72
4	o.s.	o.t.	1450	1.099	68	96
5	o.s.	o.t.	3198	3.262	85	120
6	o.s.	o.t.	6236	9.988	102	144
7	o.s.	o.t.	11116	26.87	119	168
8	o.s.	o.t.	18524	69.83	136	192

Table 1: Simulation result of Benchmark 1.1. "o.s." and "o.t." denotes "oversize" and "overtime", respectively.

[1]In this note, "overtime" refers to the case in which our program does not terminate in 20 minutes.

1.1.2 Comparison of Algorithm 1 with ILP-based Dijkstra Search

Now we compare Algorithm 1 with an ILP-based Dijkstra method we implement which is briefly described in the following (this program is also available on our Github). Given a basis partition that satisfies Condition 1 but not Condition 2 in [1], a Dijkstra search can also be carried out in basis marking space: in such a method, for each basis markings explored, an ILP (based on the state equation of the Petri net) needs to be solved to check if the target set $\mathcal{M}_{target}$ is reachable by firing implicit transitions. The ILP solver we use is *Gurobi Optimizer 9.0* (academic licence).

In this simulation, we arbitrary choose three basis partitions that satisfies Condition 1 but not Condition 2 in [1]. The results are summarized in Table 2. Since a basis marking space depends on the choice of set T_E, the size of the basis marking space and hence may vary when T_E changes. In all entries, the ILP-based Dijkstra returns the same optimal control sequence as that of Algorithm 1. However, in all entries, all three ILP-based Dijkstra search are much slower than Algorithm 1.

Even in the third case where $T_{E,3} = \{t_1, t_{15}, t_{18}, t_{20}\}$ leads to a search space whose size is similar to that of Algorithm 1, the performance of Algorithm 1 is still much better than the ILP-based one. This is because that in ILP-based algorithm a large amount of time (about 80%) is spent on solving the ILPs. For instance, for $\alpha = \beta = 6$, Algorithm 1 explore 6236 nodes to reach $\mathcal{M}_{target}$ in about 10s, while ILP 3 explores 8369 nodes in 86s among which 61s are due to the solvation of ILPs.

$\alpha = \beta$	Dijkstra Search Space	Time(s)	ILP 1	Time(s)	ILP 2	Time(s)	ILP 3	Time(s)
1	24	0.015	25	0.285	38	0.315	21	0.263
2	153	0.086	192	1.610	303	2.567	159	1.332
3	547	0.320	724	5.980	1214	10.03	605	4.859
4	1450	1.099	2089	17.45	3480	30.84	1750	14.29
5	3198	3.262	4878	45.46	8165	85.95	4045	38.69
6	6236	9.988	10145	115.1	16734	235.2	8369	86.116
7	11116	26.87	18963	294.0	31152	680.6	15516	203.12
8	18524	69.83	33191	782.4	53881	1882.2	26960	561.44

Table 2: Simulation result of Benchmark 1.1. Column "ILP 1, 2, 3" denote the number of nodes (i.e., the state space) explored. Selection of T_E's using ILP-based Dijkstra: ILP 1: $T_{E,1} = \{t_9, t_{10}, t_{15}, t_{20}\}$, ILP 2: $T_{E,2} = \{t_9, t_{14}, t_{15}, t_{19}\}$, ILP 3: $T_{E,3} = \{t_1, t_{15}, t_{18}, t_{20}\}$. "o.s." and "o.t." denotes "oversize" and "overtime", respectively.

1.2 Randomized Source Markings and Forbidden Sets

We now consider randomized source marking M_s that satisfies the following condition:

$$\begin{cases} M_s(p_1) + M_s(p_{20}) + M_s(p_{21}) + M_s(p_{22}) = \beta \\ M_s(p_6) + M_s(p_4) + M_s(p_8) = M_s(p_3) + M_s(p_7) + M_s(p_9) = \alpha \\ M_s(p_{11}) + M_s(p_{13}) + M_s(p_{15}) + M_s(p_{17}) = M_s(p_{12}) + M_s(p_{14}) + M_s(p_{16}) + M_s(p_{18}) = \alpha \\ M_s(p_2) = M_s(p_5) = M_s(p_{10}) = M_s(p_{19}) = 0 \\ \alpha, \beta \in [2, 6] \end{cases}$$

The set of target markings $\mathcal{M}_{target}$ is defined as:

$$\mathcal{M}_{target} = \{M \mid M(p_{18}) + M(p_{22}) \geq 2\alpha + \beta\}$$

The set of forbidden markings $\mathcal{M}_F$ is randomized as:

$$\mathcal{M}_F = \{M \mid M(p_2) + M(p_5) \geq \gamma\}, \quad \gamma = 1, 2, 3$$

Note that for $\gamma = 1$ transitions t_1 and t_3 are not allowed to fire: in such a case if $M_s(p_3) \neq 0$ the target set cannot be reached, i.e., there does not exist a control sequence.

According to Conditions 1 and 2 in [1], we choose a basis partition $\pi = (T_E, T_I)$ with

$$T_E = \{t_2, t_5, t_{16}, t_{18}, t_{21}\}, \quad T_I = T \setminus T_E.$$

This randomized experiment is carried out for 60 times (using program `Basis.py`). The average running time of all entries is 18.37s. All results can be found in file `Benchmark.xls` (in sheet "Benchmark 1.2"). Among all randomized entries, the *slowest* ten entries are summarized in Tables 3 and 4.

Entry	Source Marking	α	β	γ
1	$[2, 0, 1, 5, 0, 0, 4, 0, 0, 0, 5, 2, 0, 1, 0, 2, 0, 0, 0, 3, 0, 0]$	5	5	3
2	$[0, 0, 6, 1, 0, 5, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 6]$	6	6	3
3	$[0, 0, 4, 3, 0, 0, 0, 1, 0, 0, 1, 4, 3, 0, 0, 0, 0, 0, 0, 1, 1, 2]$	4	4	2
4	$[0, 0, 5, 6, 0, 0, 1, 0, 0, 0, 6, 0, 0, 1, 0, 4, 0, 1, 0, 0, 2, 1]$	6	3	2
5	$[6, 0, 0, 2, 0, 0, 2, 3, 3, 0, 5, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$	5	6	2
6	$[0, 0, 2, 5, 0, 0, 1, 0, 2, 0, 3, 2, 0, 1, 2, 2, 0, 0, 0, 0, 2, 0]$	5	2	3
7	$[0, 0, 5, 6, 0, 0, 1, 0, 0, 0, 5, 0, 0, 3, 0, 1, 1, 2, 0, 4, 0, 0]$	6	4	2
8	$[3, 0, 3, 0, 0, 0, 2, 6, 1, 0, 0, 0, 0, 0, 0, 0, 6, 6, 0, 0, 0, 0]$	6	3	3
9	$[0, 0, 2, 2, 0, 3, 0, 0, 3, 0, 5, 1, 0, 4, 0, 0, 0, 0, 0, 3, 0, 2]$	5	5	3
10	$[0, 0, 3, 6, 0, 0, 3, 0, 0, 0, 2, 2, 2, 1, 0, 3, 2, 0, 0, 2, 1, 2]$	6	5	2

Table 3: The *slowest* ten randomized source markings of Benchmark 1.2.

2 Benchmark 2

The net in Figure 2 is taken from [2 ,3] which models a hospital emergency service system (HESS). The detailed interpretation of the places and transitions of this system can be found in [2,3]. In brief,

Entry	Time-RG(s)	Dijkstra Search Space	Time(s)	$\|\sigma_{min}\|$	c_{min}
1	o.t.	9767	24.16	64	102
2	o.t.	10674	24.92	55	91
3	o.t.	9708	25.05	53	84
4	o.t.	10900	30.33	66	103
5	o.t.	11046	30.56	64	106
6	o.t.	9899	30.76	54	85
7	o.t.	17482	99.45	73	116
8	o.t.	19713	106.9	50	89
9	o.t.	27210	200.8	55	89
10	o.t.	29860	366.9	70	110

Table 4: Simulation result of Benchmark 1.2. "o.s." and "o.t." denotes "oversize" and "overtime", respectively. Column "Time-RG" is the corresponding time of using a transition-wise Dijkstra search.

place p_{22} represents the count of current patients in the Emergency Department and its capacity is 10, the largest number of patients that can be held by the department.

The initial marking of this net is $M_0 = 10 \cdot p_{22} + 7 \cdot p_{17} + 9 \cdot p_{18} + 4 \cdot p_{19} + 2 \cdot p_{20} + 2 \cdot p_{21}$, i.e., the department is empty. Now suppose that the department has run for sometime and an reconfiguration request is initiated such that the hospital should be emptied as soon as possible. The target set is

$$\mathcal{M}_{target} = \{M \mid M(p_{22}) \geq 10\}.$$

The forbidden set during the reconfiguration is

$$\mathcal{M}_F = \{M \mid M(p_8) + M(p_9) + M(p_{11}) + M(p_{12}) + M(p_{13}) \geq 4\}$$

due to the capacity of beds. We assume that the costs of all transitions in this system are unitary.

This randomized experiment is carried out for 30 entries (using program `Basis.py`). The average running time of all entries is 22.14s. All results can be found in file `Benchmark.xls` (in sheet "Benchmark 2"). Among all randomized entries, the *slowest* eight entries are summarized in Tables 5 and 6.

Entry	Source Marking
1	$[4, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 2, 0, 0, 5, 9, 4, 2, 2, 0]$
2	$[0, 3, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 2, 0, 7, 8, 3, 2, 2, 0]$
3	$[2, 1, 0, 1, 2, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 5, 7, 4, 2, 1, 1]$
4	$[0, 2, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 7, 9, 3, 1, 1, 2]$
5	$[0, 2, 2, 0, 1, 2, 0, 2, 0, 0, 1, 0, 0, 0, 0, 0, 6, 8, 4, 0, 2, 0]$
6	$[1, 2, 0, 1, 0, 1, 1, 2, 0, 1, 1, 0, 0, 0, 0, 0, 5, 9, 4, 0, 2, 0]$
7	$[0, 7, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 6, 9, 4, 1, 2, 1]$
8	$[0, 2, 0, 0, 0, 1, 2, 1, 0, 0, 1, 1, 0, 1, 1, 0, 6, 8, 4, 1, 2, 0]$

Table 5: The *slowest* eight randomized source markings of Benchmark 2.

Entry	Time-RG(s)	Dijkstra Search Space	Time(s)	$\lvert\sigma_{min}\rvert$
1	o.t.	8411	30.13	81
2	o.t.	8493	32.65	65
3	o.t.	8470	33.18	73
4	o.t.	8780	34.54	54
5	o.t.	10101	41.26	76
6	o.t.	10669	53.91	74
7	o.t.	13269	75.95	75
8	o.t.	13969	89.98	60

Table 6: Simulation result of Benchmark 2. "o.s." and "o.t." denotes "oversize" and "overtime", respectively. Column "Time-RG" is the corresponding time of using a transition-wise Dijkstra search.

References

[1] Z. Ma, M. Zou, J. Zhang, Z. Li, "Determining Optimal Control Sequences for Reconfiguration in Petri Nets," IEEE Transactions on Automatic Control, *under review*.

[2] J. Li, M. Zhouc, T. Guo, Y. Gan, X. Dai, "Robust control reconfiguration of resource allocation systems with Petri nets and integer programming," Automatica, 50 (2014), 915–923.

[3] R. Sampath, H. Darabi, U. Buy, L. Jing, "Control reconfiguration of discrete event systems with dynamic control specifications," IEEE Transactions on Automation Science and Engineering, 2008(5), 84–100.

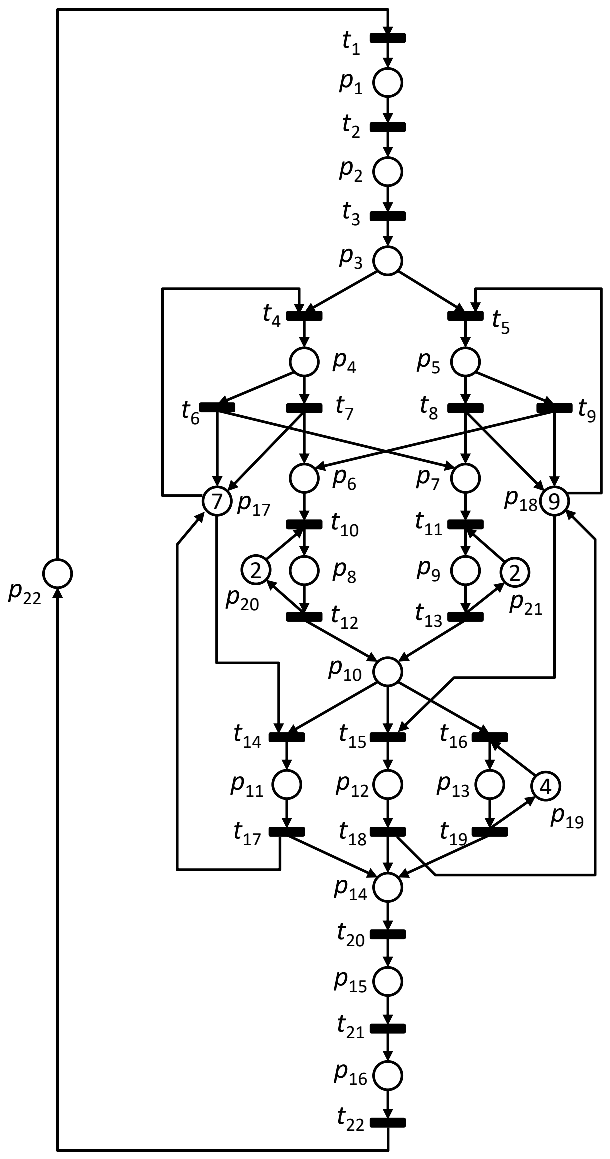

Figure 2: Benchmark 2.