# Automatic Planning Method of Pipe-Line Systems by Petri Nets


**Jiliang Luo, Zexuan Lin, Xuhang Li, Wei Liu, and Chunrong Pan**

**Abstract** This paper presents an automatic planning method for pipe-line systems under complex logic control rules based on Petri nets. Petri nets are usually modeled manually, which can lead to incomplete models of complex logic control systems. To address this issue, we extend the planning domain definition language (PDDL), which can be automatically translated into Petri nets. For complex logic control pipe-line systems, given a set of tasks, it is difficult to quickly formulate the process execution sequence through manual calculation. This paper designs an automatic pipe-line system planning method based on Petri nets, which can accurately generate Petri nets model and process execution sequences. The beer filtration system is taken as an example to illustrate the method.

**Keywords** Pipe-line systems · Petri nets · PDDL · Modeling · Automatic planning

J. Luo (✉) · C. Pan
School of Mechanical and Electrical Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China
e-mail: jlluo@hqu.edu.cn

J. Luo · Z. Lin · X. Li · W. Liu
College of Information Science and Engineering, Huaqiao University, Xiamen 361021, China

Fujian Engineering Research Center of Motor Control and System Optimal Schedule, Xiamen 361021, China

Z. Lin
e-mail: linzexuan@stu.hqu.edu.cn



S. Yadav et al. (eds.), *Energy Power and Automation Engineering*, Lecture Notes in Electrical Engineering 1118, https://doi.org/10.1007/978-981-99-8878-5_53

## 1 Introduction

In the chemical industry, pipe-line systems are essential [1]. However, it is challenging to model and control these systems with numerous valves in the factory. Additionally, control requirements must be modified dynamically and quickly when transportation tasks change or equipment is damaged. Traditional methods rely on trial-and-error experiments to derive systems models and control diagrams, which can be error-prone and time-consuming. Therefore, researching a reliable method for automatically generating valves models and planning valves controls is crucial for the efficient operation of pipe-line transportation system. Petri nets are mathematical modeling tools for discrete event systems. They can graphically represent concurrency, asynchrony, distribution, parallelism, uncertainty, or randomness. Petri nets are a mathematical tool used to establish state equations representing dynamic behavior [2, 3]. They deduce unknown state information and display dynamic behavior in graphical form. Petri nets are a modeling tool primarily used in computer science, automation, and mechanical engineering [4–7].

The pipe-line system is driven by events such as valves openings or closings, making them amenable to modeling with Petri nets. However, creating a Petri nets model of a pipe-line systems is still challenging. Current modeling approaches often involve modular modeling [8, 9], which is still a manual process. Researchers divide the pipe-line system into multiple operations controlled by valves and create a Petri nets model for each operation. They analyze the relationships between the operations and then connect each operation to the Petri nets through shared places or transitions. However, this approach heavily relies on the researchers' understanding of Petri nets and requires abstraction of the actual system into places and transitions in the Petri nets. Therefore, it is necessary to find a suitable way to ensure the integrity of the Petri nets.

The planning domain definition language (PDDL) is a kind of used to describe the computer automation planning problem of formal languages [10]. PDDL is often used to describe the domain of a planning problem (such as robot navigation, resource scheduling, and so on), including information about the actions, states, and goals available in the domain.

This paper presents an automatic planning method for pipe-line system based on Petri nets, taking a beer filter pipe-line systems as an example. Firstly, PDDL describes the system, and the content of the description is analyzed to design an automatic planning modeling algorithm to convert it into a Petri nets. Then, the operation of the control valves is complex, and the Petri nets model is used to design the automatic planning algorithm of the valves operation. When the given task, the valves operation sequence of the system can be quickly calculated by using the algorithm. Finally, we make various simulation experiments, and the results show that the automatic planning method can ensure the integrity of the model and the feasibility of the systems.

The structure of this paper is as follows: Section II here is not presents basic concepts about ordinary Petri nets and PDDL and modern-day the syntax of PDDL, Section III describes the problem of pipe-line transportation systems. Section IV, automatic planning by PDDL and Petri nets. Section V concludes this work.

## 2 Problem Statement

The pipe-line system is controlled through a set of valves that transfer fluid from one tank to another tank. In order to make the article more readable, such as the implementation example 1, beer filtration equipment is used to illustrate the proposed concepts and methods.

*Example* 1. The beer filter device completes filling, filtering, bottling, cleaning, and other tasks, as shown in Fig. 1. It consists of a feedstock tank *Ts*, three filters *MMS*1, *MMS*2, and *MMS*3, two buffer tanks $T1$ and $T2$, a washer *CIP* supply and collection system, and 20 double disk plug valves $v_1 \ldots v_{20}$ [11]. Notice that each valve can be switched to either OPEN or CLOSE position. When the valve is opened, the fluid in the vertical and horizontal pipe-line under its control mixes and then flows out through all outlet pipes, while the fluid in the vertical and horizontal pipes flows separately when the valve is CLOSE. There are 12 operations $o_1, o_2, o_3, o_4, o_5 \ldots o_{12}$ in this plant, which are summarized in Table 1. $o_1 \ldots o_7$ is the operation that controls the valve beer filtration, buffering, and filling, and $o_8 \ldots o_{12}$ is the operation that controls the valve for cleaning the filter and buffer tank. Filters *MMS*1, *MMS*2, and *MMS*3 need to be cleaned after being used once, twice, and three times, respectively; Tank $T1$ and $T2$ need to be cleaned after being used twice, and the cleaning time is 2 *min* each time. In addition, since beer and cleaners are not allowed to mix in this system, the barrier between the two materials should always remain intact.

Relationships between tasks are complex because they can share and compete for resources. This prompted us to construct a PN model, which described the control instructions of these valves through PDDL and converted them into Petri nets, so as to ensure the integrity of the model. The operation of the valve is also complicated, and it is difficult to calculate the operation instructions manually. Therefore, an algorithm needs to be built to automatically calculate the operation instructions.

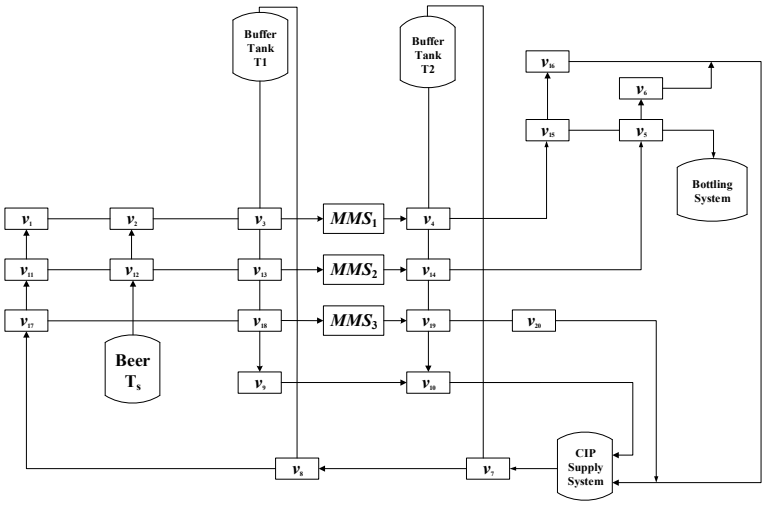


**Fig. 1** Process flow diagram of a beer filtration plant

**Table 1** Production operations of the beer filtration plant

| Operations | Open valves | Closed valves | Tanks | Time |
|---|---|---|---|---|
| $o_1$ | $v_2, v_3$ | $\overline{v}_8, \overline{v}_9, \overline{v}_{12}, \overline{v}_1$ | $T_1$ | 4 s |
| $o_2$ | $v_{12}, v_3$ | $\overline{v}_8, \overline{v}_9, \overline{v}_3, \overline{v}_{11}$ | $T_1$ | 4 s |
| $o_3$ | $v_3, v_4$ | $\overline{v}_8, \overline{v}_9, \overline{v}_7, \overline{v}_{10}, \overline{v}_1, \overline{v}_2$ | $T_1, T_2, M_1$ | 4 s |
| $o_4$ | $v_{13}, v_{14}$ | $\overline{v}_8, \overline{v}_9, \overline{v}_7, \overline{v}_{10}, \overline{v}_{11}, \overline{v}_{12}, \overline{v}_3, \overline{v}_4$ | $T_1, T_2, M_2$ | 4 s |
| $o_5$ | $v_{18}, v_{19}$ | $\overline{v}_8, \overline{v}_9, \overline{v}_7, \overline{v}_{10}, \overline{v}_3, \overline{v}_4, \overline{v}_{13}, \overline{v}_{14}, \overline{v}_{17}$ | $T_1, T_2, M_3$ | 4 s |
| $o_6$ | $v_4, v_5$ | $\overline{v}_7, \overline{v}_{10}, \overline{v}_1, \overline{v}_6$ | $T_2$ | 4 s |
| $o_7$ | $v_{14}, v_{15}$ | $\overline{v}_7, \overline{v}_{10}, \overline{v}_{11}, \overline{v}_{16}, \overline{v}_5, \overline{v}_4$ | $T_2$ | 4 s |
| $o_8$ | $v_8, v_9$ | $\overline{v}_3, \overline{v}_{13}, \overline{v}_{10}, \overline{v}_7$ | $T_1$ | 2 min |
| $o_9$ | $v_7, v_{10}$ | $\overline{v}_4, \overline{v}_{14}$ | $T_2$ | 2 min |
| $o_{10}$ | $v_1, v_6$ | $\overline{v}_2, \overline{v}_3, \overline{v}_4, \overline{v}_5, \overline{v}_7, \overline{v}_8, \overline{v}_{11}, \overline{v}_{17}$ | $M_1$ | 2 min |
| $o_{11}$ | $v_{11}, v_{16}$ | $\overline{v}_{12}, \overline{v}_{13}, \overline{v}_{14}, \overline{v}_{15}, \overline{v}_7, \overline{v}_8, \overline{v}_{17}$ | $M_2$ | 2 min |
| $o_{12}$ | $v_{17}, v_{20}$ | $\overline{v}_{18}, \overline{v}_{19}, \overline{v}_7, \overline{v}_8$ | $M_3$ | 2 min |

## 3 Automatic Planning by PDDL and Petri Nets

Currently PDDL is sufficient to describe the system to be modeled, but it generates a large amount of redundant data when modeling a pipe-line system. For example, when n tasks or device usage is encountered, *n* objects are generated during modeling. The extension of PDDL to *n* tasks is accomplished by designing a singleton type, and

*g(x)* can be converted to places [12]. So this paper extends the device usage times of PDDL, we define the count type $C = (T, L)$ to be a binary type, and the predicate is not used to describe the state. $T$ represents the number of times the object position is used and is a positive integer. $L$ represents the position of the object and is a positive integer. $T$ is converted to a Petri net position token, representing the number of times an object of this type has been used, and L is also converted to a Petri net position, representing the position of the object type. The conversion function is denoted as *f(x)*. Li [12] designed the function *h(x)* that converts predicates to places and its action to places and the transition function *k(x, y, z)*. Algorithm 1 is further improved on these bases. By placing the PDDL description file of the system into Algorithm 1, Petri net model and valve operation sequence can be automatically generated. Firstly, initialize $C^+$, $C^-$, $m_0$, $m_g$, $T$, $F$, and $O$ to be empty; secondly, we traverse singleton types, degree types, and predicates, respectively, to generate places; thirdly, go through each action and create places, transitions, and two arcs. And then assembled into Petri nets; fourthly, insert the weight of the directed arc ($F$) of $P$ pointing to $T$ into $C^+$, and the weight of the directed arc ($F$) of pointing to $P$ into $C^-$; finally, the idea of depth-first search is used to find the valve operation and output the valves operation sequences.

**Algorithm 1** Automatic planning by PDDL and Petri nets

**Input:** Entity actions set $A$, initial states set $I$, goal states set $G$, singleton set $S$, count set $C$, predicates set $P$, and object set $O$;
**Output:** Sequence of operation set $O$;

Initialize $C^+ \leftarrow \{\}$, $C^- \leftarrow \{\}$, $m_0 \leftarrow \{\}$, $m_g \leftarrow \{\}$, $T \leftarrow \{\}$, $F \leftarrow \{\}$, $O \leftarrow \{\}$, P $\leftarrow \{\}$;
**for all** c in C **do**
    Create a place p, $Mc$(p) $\leftarrow f(c)$, and P $\leftarrow$ P $\cup$ { p};
**end for**
**for all** s in S **do**
    Create a place p, $Mc$(p) $\leftarrow g(s)$, and P $\leftarrow$ P $\cup$ { p};
**end for**
**for all** *obj* in $O$ **do**
  **for all** $p$ in $P$ **do**
    Create a place p, $Mc$(p) $\leftarrow h(p)$, and P $\leftarrow$ P $\cup$ { p};
  **end for**
**end for**
**for all** $a$ in $A$ **do**
    Create a place p, create two transition $t_i$, *to*, and create two directed arc $f_i, f_o$;
    $f_i$ is from $t_i$ point to p, $f_o$ is from p point to $t_o$;
    Calculate p, $t_i$, $t_o$, $f_i$, $f_o$ into $k(x,y,z)$;
    P $\leftarrow$ P $\cup$ { p}, T $\leftarrow$ T $\cup$ { $t_i$, $t_o$}, and F $\leftarrow$ F $\cup$ { $f_i, f_o$};
**end for**
P, T, F $\in$ PN

(continued)

(continued)

20: Insert the weight of the directed arc ($F$) of $P$ pointing to $T$ into $C^+$, and the weight of the directed arc ($F$) of T pointing to $P$ into $C^-$;
21: **for all** $i$ in $I$ **do**
22: Insert $i$ into $m_o$;
23: **end for**
24: **for all** $g$ in $G$ **do**
25: Insert $g$ into $m_g$;
26: **end for**
27: $list \leftarrow m_0$;
28: **if** $m_k == m_g$ **then**
29: Insert $m_k$ in $list$;
30: Execution order of places is determined by the post-set places from transitions, which are placed into $P$;
31: **for all** o in seek the place of mark a in $P$ **do**
32: $o$ insert into $O$;
33: **end for**
34: Output $O$;
35: **else**
36: Calculate the sum of transition sets of enabled state under $m_k$ state: $E_k = \{ t \in T \mid m_k \geq C - (. , t);$
37: **if** $E_k = \emptyset$ **then**
38: Exit with failure;
39: **end if**
40: **for all** each transition $t$ in Petri nets **do**
41: **if** $t$ in $E_k$ **then**
42: Fire $t$ to generate a new state $m_{k+1}$,and return to step 33;
43: **end if**
44: **end for**
45: **end if**

## 4 Conclusion

Describe each operation in the beer filtration system using PDDL and improved PDDL, and convert them into Petri nets for display, as shown in Fig. 2. When the processing plant has a task, it can automatically calculate the operation sequence of the control valves through the automatic planning method. When there are three tasks, the order of operation can be calculated automatically through algorithm calculation, and Gantt chart can be listed through operation sequence, as shown in Fig. 3.

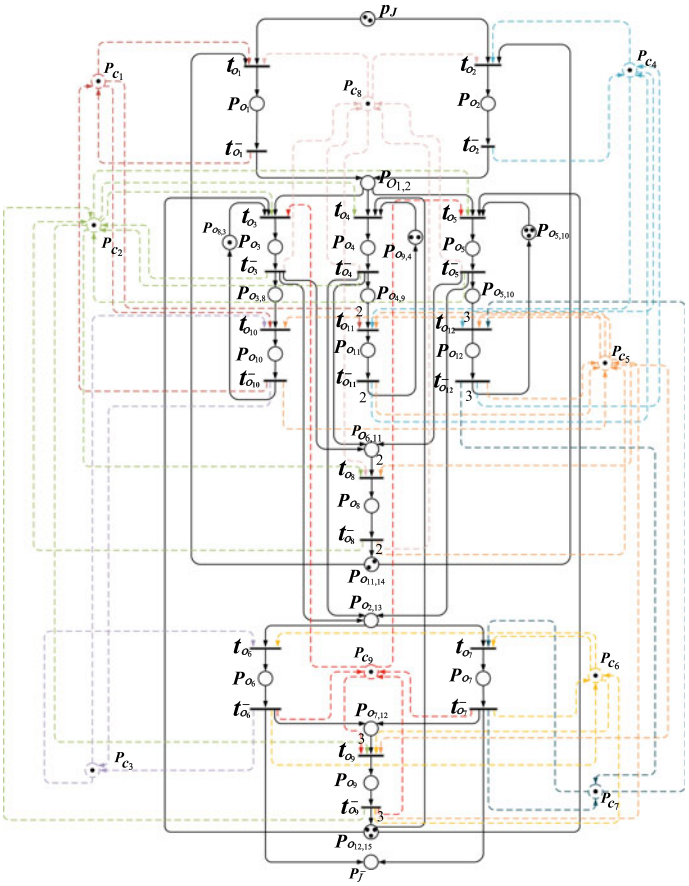


**Fig. 2** The process flow diagram of a beer filtration plant

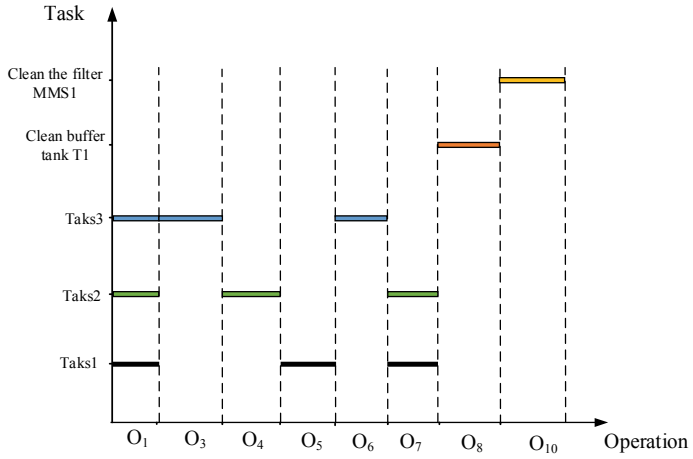


**Fig. 3** Beer filtration Gantt chart

**Acknowledgements** This work was supported in part by National Natural Science Foundation of China under Grants 61973130.

## References

1. Fu Y, Luo J, Lin W, Huang Y, Ye J (2018) Petri-net controller for pipe-line transportation system. In: 2018 5th international conference on control, decision and information technologies (CoDIT), pp 809–814
2. Luo J, Huang Y-S, Weng Y-S (2020) Design of variable traffic light control systems for preventing two-way grid network traffic jams using timed petri nets. IEEE Trans Intell Transp Syst 21(7):3117–3127
3. Wang S, Luo J (2018) Synthesis of mobile robot control system on embedded chips via petri nets. J Chin Inst Eng 41(6):442–451
4. Luo J, Tan K, Luo H, Zhou M (2021) Inference approach based on petri nets. Inf Sci 547:1008–1024
5. Boyer M, Diaz M (1999) Non equivalence between time Petri nets and time stream Petri nets. In: Proceedings 8th international workshop on Petri nets and performance models (Cat. No.PR00331), Zaragoza, Spain, pp 198–207. https://doi.org/10.1109/PNPM.1999.796566
6. Zhai Z (2010) Further study on liveness of time Petri net. In: 2010 international conference on mechanic automation and control engineering, Wuhan, China, pp 5508–5511. https://doi.org/10.1109/MACE.2010.5535580
7. Gong M, Song H, Tan J, Xie Y, Song J (2017) Fault diagnosis of motor based on mutative scale back propagation net evolving fuzzy Petri nets. In: 2017 Chinese Automation Congress (CAC), Jinan, China, pp 3826–3829. https://doi.org/10.1109/CAC.2017.8243447
8. Luo J, Ni H, Zhou M (2015) Control program design for automated guided vehicle systems via Petri nets. IEEE Trans Syst Man Cybern Syst 45(1):44–55. https://doi.org/10.1109/TSMC.2014.2342199
9. Davidrajuh R (2019) A new modular petri net for modeling large discrete-event systems: a proposal based on the literature study. Computers 8(4):83

10. McDermott D (1998) PDDL-the planning domain definition language. In: Yale center for computational vision and control. New Haven, CI, USA
11. Yeh M-L, Chang C-T (2012) An automata based method for online synthesis of emergency response procedures in batch processes. Comput Chem Eng 38:151–170
12. Li X, Luo J, Li J, Yi S, Pan C (2022) Parallel Petri nets modeling method of manufacturing system based on the improved PDDL. In: 2022 IEEE international conference on networking, sensing and control (ICNSC), Shanghai, China, pp 1–6. https://doi.org/10.1109/ICNSC55942.2022.10004131