A Note on Grid-Type Directed Acyclic Graph for Important Property of Resource Allocation Problem

Mochamad Nizar Palefi Ma'ady(✉) and Tabina Shafa Nabila Syahda

Department of Information System, Institut Teknologi Telkom Surabaya, Surabaya, Indonesia
nizar@ittelkom-sby.ac.id,
tabina.shafa.21@student.is.ittelkom-sby.ac.id

Abstract. We consider the following resource allocation problem on a directed acyclic graph (the precedence graph). Suppose we have a given X units of a resource that must be distributed among N activities with some of given return functions $r_i(x)$. The problem is to allocate all of the X units of resource to the activities so as to maximize the total return. Specific constraint is added that the total allocation to the designated M machines ($M < N$) that must be at least Y but at most Z units ($Y \leq \mathrm{Z} \leq X$). We present and illustrate the dynamic programming procedure in a grid-type directed acyclic graph by conducting a numerical solution. As the result, we point out the optimal policy is in independent of the order of machines as important property of resource allocation problem. We represent the posed constrained problem in finding a best obstacle-avoiding path.

Keywords: grid-type directed acyclic graph · resource allocation · dynamic programming · machine order · constrained problem

1 Introduction

Consider the following resource allocation problem. We are given X units of a resource that must be distributed among N activities. We impose an additional constraint to a certain designated M machine (M < N) that is to allocate the X units to be at least Y but at most Z units, where Y ≤ Z ≤ X. The resource allocation problem is how to optimally allocate the units among the activities in order to maximize the total return $r_i(x)$ [1–3].

In many literatures [4–8], quantitative indicators are proposed to describe the characteristics of a network, including complexity index [9], network complexity, resource factor, and resource use. Investigation the application of dynamic programming to the problem of resource allocation in a project with the objective of minimizing the project completion time did by Elmaghraby [10].

We express the constrained problem into a grid-type directed acyclic graph [11–14]. Many graph and network problems that are either intractable or have complicated solutions in the general case are easy in the special case of series-parallel networks [5, 15]. Bein, Brucker, and Tamir [16] show that the minimum cost flow problem is solved by the greedy algorithm if and only if the graph is series-parallel. Other examples include sequencing problems, location problems [9], as well as many combinatorial problems [7, 8, 10, 17].

We exploit grid-type dag with the spirit of dynamic programming. For any machine k > M, we have to evaluate $V_k(X)$ for X = 0, 1, …, X-Y. Besides, $V_k(X)$ X = 0, 1, …, X is used for machine k < M. Our goal is to find maximum return value to go from machine k to end that the allocation must be carried out to satisfy the constraint of machine M. We represent this constraint as an obstacle in dag which is denoted as illegal path (-∞) (see Fig. 1). Then we denote the policy function as Π_k. In this paper, we conduct the four steps of common dynamic programming that is definition of optimal value function, recurrence relation, boundary condition and answer for the machine k = M. This approach carries out *the principle of optimality,* which observes the best path among previously known nodes. Therefore, we can leverage the idea of maximization $S = \max\{0, X - (X - Y)\}$ into the recurrence relation.

2 Methodology

The common resource allocation problem is given by X units of a resource and told that these resources must be distributed among N activities wih N data tables $r_{N+1}(X)$ (for i = 1, 2, …, N and x = 0, 1, …, X) representing return realized from an allocation of x units resource to activity i. In this context, we add a constraint of machine M, unit Y and Z that must be satisfied in finding a best path as shown in Fig. 1.

Initially, we introduce the definition of optimal value function for resource allocation via dynamic programming in grid-type dag below:

$V_m(X)$ = max return value to go, starting machine k through machine N when X units of resource still remain to be used.

We keep the value function as the common resource allocation problem without enlarging it. We propose the recurrence relation for machine *M:*

$$V_m(X) = \max_{X_m = S, \ldots, X} \left[r_m(X_{\ m}) + V_{m+1}(X - X_m) \right] \tag{1}$$

where

$$S = \max\{0, X - (X - Y)\}$$

We emphasize the argument in the boundary condition for solving the constraint to reach (*X*, *M* + *1*) by going down.

$$V_N(X) = r_N(X). \tag{2}$$

Since at least one unit must remain from stage (*M* + *1*) on:

$$X - Z = 1 \le \sum\nolimits_{i=M+1}^{N} X_i \le \mathrm{X} - \mathrm{Y} \tag{3}$$

from

$$Y \le \sum\nolimits_{i=1}^{M} X_i \le \mathrm{Z} \le \mathrm{X}$$

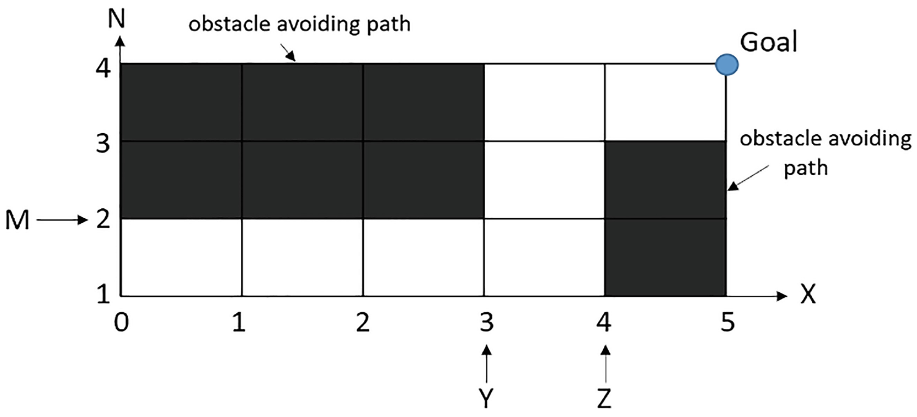

Fig. 1. Grid-type directed acyclic graph with obstacle avoidin path.

$V_m(0) = -\infty$ (illegal) and $V_m(1) = V_{m+1}(1)$ for at least one unit to the remaining (N-M) machines. Let $t = X - 1$, then $X_m(0) = S, \ldots, t$. That is the range of decision variable X_m should be between S and t. We concise variables' definition that X, Y, Z, N, and M are units of a resource, a designated constraint of minimum unit, a designated constraint of maximum unit, number of machines, and a designated constraint of certain machine, respectively. After computing the recurrence relation thoroughly, the answer is given by:

$$V_1(X) = \underset{X_m = S,\ldots,X}{\text{Max}} \left[r_1(X_m) + V_2(X - X_m) \right] \tag{4}$$

3 Numerical Solutiion

We provide a numerical example for the aforementioned method and demonstrate the solution in backward-pass stage-wise procedure. For example, let the given data where the first two machines ($M = 2$) must totally receive at least 3 units (Y = 3) but at most 4 units ($Z = X - 1 = 4$) in detail on Table 1.

Numerical computation starting from (3) in state of all resources have been stored as stage 4 followed by (1) in the stage 3 up to 2, whereas resources remain to be used in the stage 1 with the use of formula (4) regarding backward-pass procedure.

Stage 4 (boundary condition)

$$V_4(0) = r_4(0) = 0;$$

$$V_4(1) = r_4(1) = 1;$$

$$V_4(2) = r_4(2) = 3.$$

Stage 3 (recurrence relation)

$$V_3(0) = -\infty;$$

Table 1. Example for N = 4 M = 2; X = 5; Y = 3; Z = 4

x	$r_1(x)$	$r_2(x)$	$r_3(x)$	$r_4(x)$
0	0	0	0	0
1	3	1	2	1
2	4	2	4	3
3	6	4	6	6
4	9	8	8	9
5	14	13	10	12

$$V_3(1) = \max\begin{cases} x = 0 : r_3(0) + V_4(1) = 0 + 1 = 1 \\ x = 1 : r_3(1) + V_4(0) = 2 + 0 = 2 \end{cases}$$

$$= 2,\ \Pi_3(1) = 1;$$

$$V_3(2) = 4,\ \Pi_3(2) = 2.$$

Stage 2 (recurrence relation)

$$V_2(0) = -\infty;$$

$$V_2(1) = \max\begin{cases} x = 0 : r_2(0) + V_3(1) = 0 + 2 = 2 \\ x = 1 : r_2(1) + V_3(0) = -\infty \end{cases}$$

$$= 2,\ \Pi_2(1) = 0$$

$$V_2(2) = 4,\ \Pi_2(2) = 0;$$

$$V_2(3) = 5,\ \Pi_2(3) = 1;$$

$$V_2(4) = 6,\ \Pi_2(4) = 2 \text{ or } 3;$$

$$V_2(5) = 10,\ \Pi_2(5) = 4.$$

We omit detail computation above, therein the same computation shown in $V_1(5)$ below.

Stage 1 (answer is given by)

$$V_1(5) = \max\begin{cases} x = 0 : r_1(0) + V_2(5) = 0 + 10 = 10 \\ x = 1 : r_1(1) + V_2(4) = 3 + 6 = 9 \\ x = 2 : r_1(2) + V_2(3) = 4 + 5 = 9 \\ x = 3 : r_1(3) + V_2(2) = 6 + 4 = 10 \\ x = 4 : r_1(4) + V_2(1) = 9 + 2 = 11 \\ x = 5 : r_1(5) + V_2(0) = 14 + -\infty = -\infty \end{cases}$$

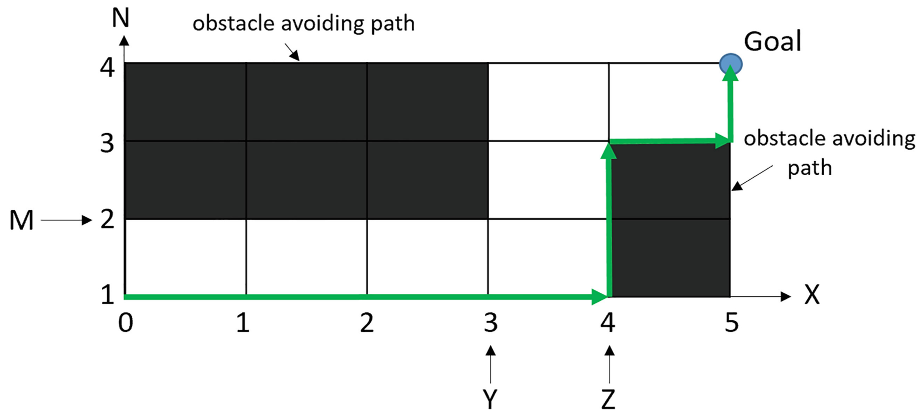

Fig. 2. The optimal allocation that satisfy the constraint is indicated by green arrow.

$$= 11, \Pi_1(5) = 4$$

Finally, we trace back the policy function $\Pi_1(5) = 4$ from stage 1 to 4 to obtain the optimal path. We can simply follow the policy function in searching the optimal path:

$V_1(5) = \Pi_1(5) = 4$ *units* $\rightarrow V_2(1) = \Pi_2(1) = 0$ *unit* $\rightarrow V_3(1) = \Pi_3(1) = 1$ *unit* $\rightarrow V_4(0) = \mathrm{r}_4(0) = 0$ *unit*

Then, we are able to draw the optimal path into the grid-type dag as Fig. 2. As the illustration, the optimal allocation is to yield the first machine as many 4 unit of resource, then the last unit will be allocated at machine 3 with the maximum value return 11 and satisfies the constraint of M machines, X and Y units.

Hence, we are tracing back the policy function $\Pi_1(5) = 4$ from stage 1 to 4 to get the optimal path. The optimal path is indicated by green arrow, where machine 1 starts with X = 5 units. The first machine consumes 4 units up to reach the machine 2, followed by 0 unit of machine 2. The remaining 1 unit is consumed of machine 3. The optimal solution path is able to avoid the obstacle such diagram in Fig. 2. In this manner, the order of the machine does not matter. Machine 1 up to machine *N* may be independent due to the construction of the proposed solution.

4 Discussion

Zhu et al. [18] investigate constraint resource allocation problem under directed (cyclic) graph, whereas we investigate the constraint under directed acyclic graph. In their proposes, dynamic programming takes into account in the solution without modeling graph to grid-type. In the resource allocation context, this paper shows a plausible dynamic programming approach by modeling a constraint problem into grid-type directed acyclic graph. It may be an alternative approach for scholars to solve resource allocation problem in a new manner with appropriate matter or we can convey a certain constrained problem may result a new approach of dynamic programming uses.

As we mentioned before, the important findings of this paper in resource allocation problem with the respected constraint of machines and resources is that the order of machine does not matter, which uses dynamic programming in grid-type directed acyclic graph. Therein, in nowadays dynamic programming applications like Xue et al. [19] and Zhou et al. [20] works tends to include different constraints with different dynamic programming approach that, as DRAIM of Zhou et al. [20] using dynamic programming winner selection method for delay-constraint and system identification technique based on neural networks of Xue et al. [19] using adaptive dynamic programming for constrained event-triggered H_∞.

5 Conclusion

The advantage of the dynamic programming approach is its generality, since it accommodates any form of the resource allocation function. We can solve the constraint problem of resource allocation that is common encountered in real problem using grid-type dag of dynamic programming. In the process of solving problem, the order of machines; 1, 2, 3, 4, does not matter. The proposed formula solution is able to solve with any order of machine resulting the same optimal allocation. That is the important property of this solving resource allocation problem in the certain constraint. Further, with the numerical solution of dynamic programming formulation, we can also develop to an automatic software with interesting visualization of grid-type dag.

Acknowledgment. This research project was supported by a grant from the research institute and community service (LPPM), Institut Teknologi Telkom Surabaya, Indonesia.

References

1. C. L. Monma, “Two-Machine Maximum Flow Time Problem With Series-Parallel Precedence Constraints: an Algorithm and Extensions.,” *Oper. Res.*, vol. 27, no. 4, pp. 792–798, 1979.
2. E. Barbierato, M. Gribaudo, and M. Iacono, “Map-reduce process algebra: a formalism to describe directed acyclic graph task-based jobs in parallel environments,” in *International Conference on Analytical and Stochastic Modeling Techniques and Applications*, 2019, pp. 85–99.
3. Q. Shang, “A dynamic resource allocation algorithm in cloud computing based on workflow and resource clustering,” *J. Internet Technol.*, vol. 22, no. 2, pp. 403–411, 2021.
4. R. Rajak, S. Kumar, S. Prakash, N. Rajak, and P. Dixit, “A novel technique to optimize quality of service for directed acyclic graph (DAG) scheduling in cloud computing environment using heuristic approach,” *J. Supercomput.*, pp. 1–24, 2022.
5. J. Chen, Y. Yang, C. Wang, H. Zhang, C. Qiu, and X. Wang, “Multitask Offloading Strategy Optimization Based on Directed Acyclic Graphs for Edge Computing,” *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9367–9378, 2022.
6. C. Lu, W. Chen, K. Ye, and C. Z. Xu, “Understanding the Workload Characteristics in Alibaba: A View from Directed Acyclic Graph Analysis,” *2020 Int. Conf. High Perform. Big Data Intell. Syst. HPBD IS 2020*, 2020.

7. L. Yang, C. Zhong, Q. Yang, W. Zou, and A. Fathalla, "Task offloading for directed acyclic graph applications based on edge computing in Industrial Internet," *Inf. Sci. (Ny).*, vol. 540, pp. 51–68, 2020.
8. F. Arrestier, K. Desnos, E. Juarez, and D. Menard, "Numerical representation of directed acyclic graphs for efficient dataflow embedded resource allocation," *ACM Trans. Embed. Comput. Syst.*, vol. 18, no. 5s, 2019.
9. W. W. Bein, P. Brucker, and A. Tamir, "Minimum cost flow algorithms for series-parallel networks," *Discret. Appl. Math.*, vol. 10, no. 2, pp. 117–124, 1985.
10. S. E. Elmaghraby, "Resource allocation via dynamic programming in activity networks," *Eur. J. Oper. Res.*, vol. 64, no. 2, pp. 199–215, 1993.
11. M. Asad Arfeen, K. Pawlikowski, and A. Willig, "A framework for resource allocation strategies in cloud computing environment," *Proc. - Int. Comput. Softw. Appl. Conf.*, no. August, pp. 261–266, 2011.
12. D. Kliazovich, J. E. Pecero, A. Tchernykh, P. Bouvry, S. U. Khan, and A. Y. Zomaya, "CA-DAG: Communication-aware directed acyclic graphs for modeling cloud computing applications," *IEEE Int. Conf. Cloud Comput. CLOUD*, pp. 277–284, 2013.
13. K. Bhasyam and K. Bazargan, "HW/SW codesign incorporating edge delays using dynamic programming," *Proc. - Euromicro Symp. Digit. Syst. Des. DSD 2003*, pp. 264–271, 2003.
14. S. Puthoor *et al.*, "Implementing directed acyclic graphs with the heterogeneous system architecture," *9th Work. Gen. Purp. Process. using GPUs, GPGPU 2016 - Proc.*, pp. 53–62, 2016.
15. O. Prila, "The algorithm of job scheduling in Grid environment based on the dynamic programming method," *Вісник Чернігівського державного технологічного університету. Серія: Технічні науки*, no. 4, pp. 153–162, 2013.
16. R. Hassin and A. Tamir, "Efficient Algorithms for Optimization and Selection on Series-Parallel Graphs," *SIAM J. Algebr. Discret. Methods*, vol. 7, no. 3, pp. 379–389, 1986.
17. W. W. Bein, J. Kamburowski, and M. F. M. Stallmann, "Optimal Reduction of Two-Terminal Directed Acyclic Graphs," *SIAM J. Comput.*, vol. 21, no. 6, pp. 1112–1129, 1992.
18. Y. Zhu, S. Member, W. Ren, W. Yu, and S. Member, "Graphs via Continuous-Time Algorithms," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. PP, no. 2, pp. 1–10, 2019.
19. S. Xue, B. Luo, D. Liu, and Y. Yang, "Constrained Event-Triggered H∞Control Based on Adaptive Dynamic Programming with Concurrent Learning," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 52, no. 1, pp. 357–369, 2022.
20. H. Zhou, X. Chen, S. He, J. Chen, and J. Wu, "DRAIM: A Novel Delay-Constraint and Reverse Auction-Based Incentive Mechanism for WiFi Offloading," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 4, pp. 711–722, 2020.