International Journal of Application on Sciences, Technology and Engineering
(IJASTE)
Volume 1, Issue 2,2023.ISSN:2987-2499

Repetitive Project Optimization with Dynamic Programming Method

Erica Hosanna[1, a)] and Oei Fuk Jin[1, b)]

Author Affiliations
[1]Universitas Tarumanagara. Jl. Letjen S.Parman no. 1, West Jakarta, Indonesia, 11440
Author Emails:
a) *erica.hosanna@gmail.com*
b) *fukjin.untar@gmail.com*

Submitted: March 2023, Revised: April 23 2023, Accepted: May 22, 2023

ABSTRACT

Repetitive Scheduling Method is a scheduling method tailored specifically for projects with repetitive activities. Project optimization is done to optimize the cost and duration to be as small and fast as they can be, but applications of project optimization in repetitive projects are inefficient and leaves a lot of room for error. Research analyses uses dynamic programming with the program Python and Google OR-Tools. This research starts with designing a set of dynamic programing script for project optimization, testing it on a project, and comparing it to results based on manual application. Project optimization analyses using dynamic programming will result on comparisons between cost and duration in a project. The script is then slightly adjusted and implemented on multiple critical path project. It is concluded that dynamic programming method is more efficient and faster in project optimization.

INTRODUCTION

Project optimization is an iteration proses to ascertain the optimal duration in a project. Each activity in the critical path is studied and crashed according to the constraints available. This is done to ensure that the cost of project crashing will reduce the indirect cost, and thus, will decrease the total cost of the project.

A repetitive project is a project with repetitive activities. Optimizing a repetitive project proves to be harder. The best method to optimize a repetitive project is by using arrow diagram network (ADN) method (Zhang, 2015), and yet, the largest problem of using ADN to optimize the project is the number of activities that needs to be processed (Bhoyari, 2014), which far exceeds a normal project. This causes project optimization for repetitive project to be long, arduous, with a lot of room for error.

Dynamic Programming (DP) is a programming method that can be used for project optimization (Vanhoucke, 2013). Dynamic programming specializes in solving a problem with similar sub-problems, in which each sub- problems require 1 optimal solution. The solution comes in the form of a script, which consists of a combination of commands and statements that will solve the sub-problem. This, thus, causes a sharp decline of human error in the optimizing process, limiting the errors only to the script designed, and to the data input. Project optimization can be separated into several simpler stages, where a script can be designed to solve the problems of each stage.

The main objective of this research is to optimize a repetitive project schedule using dynamic programming method, and to develop a script that can solve project optimization

problems. The last objective is to know the advantages and disadvantages of using dynamic programming method in repetitive projects.

Dynamic Programming

Dynamic Programming (DP) is and algorithm technique to solve a problem. The main principle in dynamic programming is to solve a complex problem by separating it into several simpler problems, in which these simpler problems have one optimal solution (Vanhoucke, 2013). These solutions are combined to form the optimal solution to the main problem.

A script is a collection of commands and statements in a program that can solve a problem. DP consists of stages, state and state variables, state transition, and the optimal choice. Each sub-problem is considered one stage of the main solution. A combination of commands and statements are used to procure the optimal solution for each stage. These commands and statements are considered as state and state variables. After the optimal solution for each stage is found, a statement is needed to connect the solutions. A transition statement is the statement that defines how one sub-problem is related to other sub-problems. A combination of these items produces the optimal solution to the main problem.

Repetitive Scheduling Method (RSM)

A project with similar and repetitive activity is called a repetitive project (Harris dan Ioannou, 1998). Repetitive scheduling method (RSM) is a method of scheduling for repetitive projects. RSM uses the x-axis as the duration and the y axis as the location or number of units. In Fig. 1, Da is the total duration for activity A of 3 housing units. Activity A starts on the 10^{th} day and ends on the 16^{th} day. Activity A for the first unit starts on the 10^{th} day and ends on the 12^{th} day. Db is the duration of activity B for all units

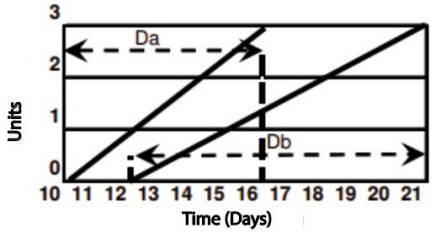

FIGURE 1. Repetitive Scheduling Method (Harris dan Ioannou 1998)

Project Optimization

Project Crashing is a method to reduce the duration of a project that is done systematically and analytically (Hansen, 2015). Project crashing is done on activities in the critical path. The chosen activity has the smallest cost slope. Cost slope (CS) is the cost added for every 1 day on which the activity is crashed. The formula for CS is as follows:

$$CS = \frac{-N}{-N}$$

Where ND = Normal Duration, CD = Crash Duration, NC= Normal Cost dan CC = Crash Cost.

(1)

Project crashing risks adding costs to the project due to the additional resources. These additional costs can be controlled using Time-Cost Trade Off (Robinson, 1975). Discrete Time-Cost Trade Off (D-TCTO) is a method of optimalization developed to solve time-cost problems. D-TCTO calculates 3 costs in each iteration: indirect cost, direct cost, and total cost. Indirect cost is the daily, weekly, or monthly costs needed for off-field needs such as administration and management. Direct cost is the cost needed to proceed the project, and total cost is the sum of direct and indirect cost. D-TCTO is drawn in Fig.2, where the x-axis is the duration of the project, while y-axis is the cost. The main objective is D-TCTO is to calculate the duration with the least amount of cost.

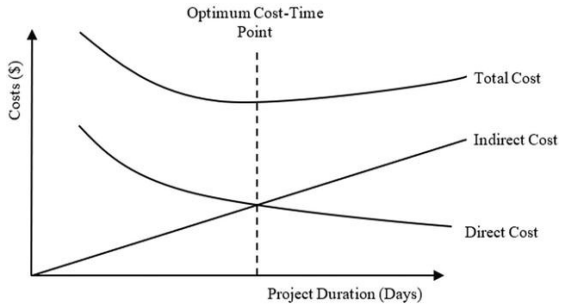

FIGURE 2. Discrete Time-Cost Diagram

METHOD

The research begins with collecting project data in the form of documents, which is then processed for the analyses stage. The first stage is the Pilot Test, where the project optimization is done both by manual and dynamic programming method. A script is designed and tested during this stage, and the script is considered a success if the results between manual and dynamic programming method are similar. The next stage is to test the script against any other possible projects and it is done to detect any limitations to the script designed. Should there be any limitations, additional script should be developed then.

RESEARCH RESULTS

Pilot Test

Using 3 housing units as the research object, analyses is done through manual method and dynamic programming method. Table 1 shows the data needed to build 1 housing unit, which consists of activities, normal cost, normal duration, crash cost, crash duration, and cost slope. The indirect cost for the project is Rp. 850.000,00/ day.

ADN diagram for 3 housing unit is displayed in Fig. 3 which shows the relationship of the activities for those units. The project begins on the 1st node and ends on the 33rd node and the arrows represent the activities. From Fig.3, it shows that the critical path on this project resides on node: 1 - 2 - 3 - 4 - 5 - 6 - 14 - 15 - 16 - 17 - 25 - 26 - 27 - 28 - 29 - 30 - 33. Activities on the aforementioned nodes are: 1A - 1B - 1C - 1D - 1E - 1F2F dum - 2C - 2D - 2E - 2F3F dum - 3C - 3D - 3E - 3H - 3I - 3L and 256 days are required to fully built 3 housing units.

TABLE 1. Project Data for 1 Housing Unit

act	*Normal Duration (ND)* (days)	*Normal Cost*	*Crash Duration*	*Crash Cost*	*Cost Slope*
A	2	2.069.250,0	2	2.069.250,00	-
B	30	8.853.315,0	20	13.853.315,00	500.000,0
C	28	3,595,500.0	18	8,595,500.00	500.000,0
D	28	6,650,500.0	18	13,650,500.00	500.000,0
E	7	10,246,850.0	5	12,246,850.00	400.000,0
F	21	23,239,950.0	14	24,639,950.00	200.000,0
H	7	4,165,000.0	5	4,465,000.00	150.000,0
I	21	9,894,360.0	14	16,894,360.00	1.000.000,0
J	7	4,560,000.0	7	4,560,000.00	-
G	1	12,000,000.0	1	12,000,000.00	-
L	7	2,090,000.0	5	2,790,000.00	350.000,0
Tota		87.404.725,0		136.704.725,00	

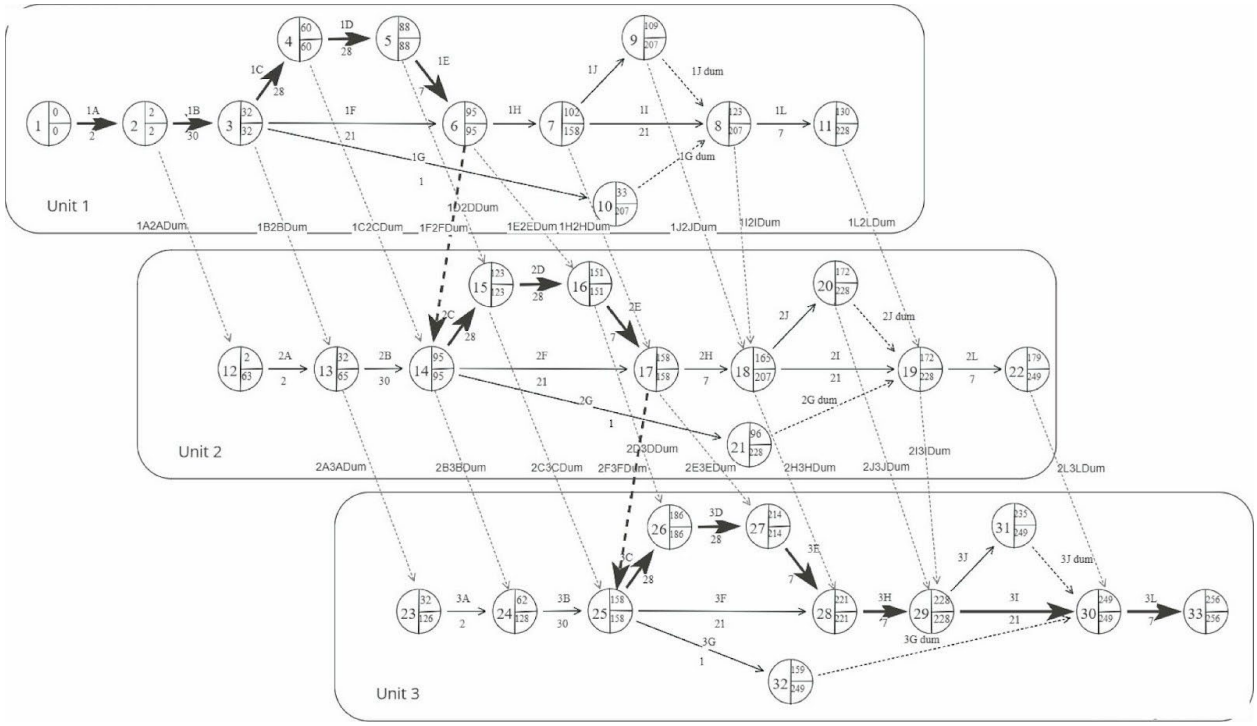

FIGURE 3. Arrow Diagram Network (3 Housing Units)

Optimization with Manual Method (Pilot Test)

From the data provided in Fig.3 and Table 1, the project optimization is done manually. In Stage-0, the direct cost needed is Rp. 262.214.175,00, and the indirect cost for 256 days is Rp. 850.000,00*256 = Rp. 217.600.000,00. Then, the total cost of the project is Rp. 479.814.175,00. An activity in the critical path with the lowest cost slope is chosen to be optimized. In this case, activity 3H (between Node 28 – 29) with the cost slope Rp. 150.000,00/ day is chosen. Activity 3H is optimized for 2 days, causing the direct cost to increase as much as Rp. 300.000,00 (Rp. 150.000,00

*2) and indirect cost to decrease as much as Rp. 1.700.000,00. This causes the project duration to

be 254 days, thus, the total indirect cost becomes Rp. 215.900.000,00 (Rp 850.000,00*254 days), and direct cost becomes Rp. 262.514.175,00. The total cost for the new schedule is Rp.478.414.175,00. This proses is repeated until the project can no longer be crashed. The iteration results can be seen on Table 2.

TABLE 2. Time-Cost Tabel (Manual)

No. Iteration	Activity	*Crash*	Duration	Indirect Cost	Direct Cost	Total Cost
0	-	-	256	Rp.	Rp	Rp. 479.814.175,00
1	3H	2	254	Rp.	Rp.	Rp. 478.414.175,00
2	3L	2	252	Rp.	Rp.	Rp. 477.414.175,00
3	1E	2	250	Rp.	Rp.	Rp. 476.514.175,00
4	2E	2	248	Rp.	Rp.	Rp. 475.614.175,00
5	3E	2	246	Rp.	Rp.	Rp. 474.714.175,00
6	1B	10	236	Rp.	Rp.	Rp. 471.214.175,00
7	1C	10	226	Rp.	Rp.	Rp. 467.714.175,00
8	2C	10	216	Rp.	Rp.	Rp. 464.214.175,00
9	3C	10	206	Rp.	Rp.	Rp. 460.714.175,00
10	1D	10	196	Rp.	Rp.	Rp. 457.214.175,00
11	2D	10	186	Rp.	Rp.	Rp. 453.714.175,00
12	3D	10	176	Rp.	Rp.	Rp. 450.214.175,00
13	3I	7	169	Rp.	Rp.	Rp. 451.264.175,00

Optimization with Dynamic Programming Method (Pilot Test)

Project optimization with dynamic programming method is done through the Python program and OR-Tools module. A script is designed to optimize a project, following the iteration process found in Fig.4. Two new variables are used during the iteration process, which are Crashablecritacts Table and Mincostindex. Crashablecritacts Table is a temporary table that will change on each iteration, which consists of activities that can be crashed during each iteration. Meanwhile, Mincostindex is the activity chosen from the Crashablecritacts Table for the crashing process in each iteration. The duration of the Mincostindex activity is then reduced by 1 day. The script then recalculates the new duration cost. This process is repeated until the resulting duration is equal to the duration produced with crash duration (CD) in Table 5.

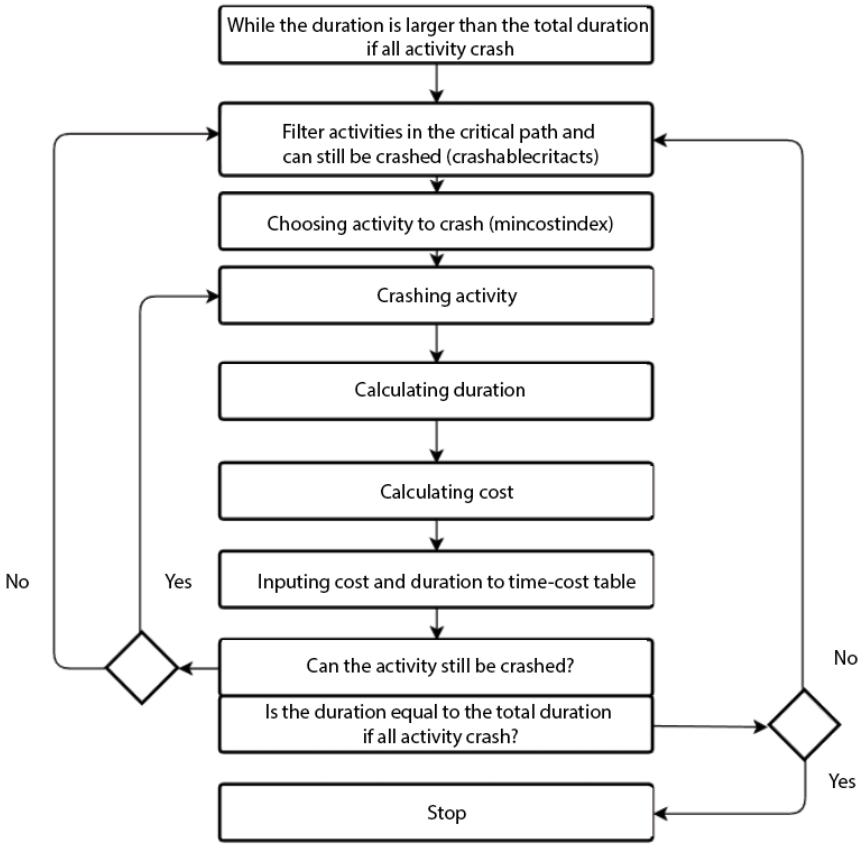

FIGURE 4. Arrow Diagram Network (Pilot Test)

The script is used to process the data in Table 1 and Fig.3. The calculation results from the script shows that the project duration before optimization is 256 days, while a fully optimized duration is 169 days. A Time-Cost Table produced by Python in this stage can be found in Fig.5.

	duration	indirectcost	directcost	totalcost	activity
0	254.0	215900000	262514175	478414175	3H
1	252.0	214200000	263214175	477414175	3L
2	250.0	212500000	264014175	476514175	1E
3	248.0	210800000	264814175	475614175	2E
4	246.0	209100000	265614175	474714175	3E
5	236.0	200600000	270614175	471214175	1B
6	226.0	192100000	275614175	467714175	1C
7	216.0	183600000	280614175	464214175	1D
8	206.0	175100000	285614175	460714175	2C
9	196.0	166600000	290614175	457214175	2D
10	186.0	158100000	295614175	453714175	3C
11	176.0	149600000	300614175	450214175	3D
12	169.0	143650000	307614175	451264175	3I

FIGURE 5. Time-Cost Table for Pilot Test (Dynamic Programming)

Difference between Manual and Dynamic Programming Method

The results from both method, manual (Table 2) and dynamic programming (Fig. 5) are then analyzed. It shows that cost and duration between Table 2 and Fig. 5 are the same, however, there

are several differences in the order of activity chosen to be crashed. On 226 days to 176 days, the activity chosen to be crashed is activity C and D. The activities chosen to be crashed with manual method in order is: 1C, 2C, 3C, 1D, 2D and 3D, while the activities chosen to be crashed with dynamic programming method in order is 1C, 1D, 2C, 2D, 3C, and 3D. The cost slope for activity C and D are Rp. 500.000,00, thus, the difference is inconsequential because the costs of both activities are similar.

Figure 6 shows the critical activities schedule produced using the script. The column 'act' is the activity, 'CP' is the critical path, in which 1 is part of the critical path while 0 means it is not part of the critical path. D is the duration of the activity. ST is the start time of activity, while ET is the finish time of the activity.

	act	CP	D	ST	ET
0	1A	1	2	0	2
1	1B	1	20	2	22
2	1C	1	18	22	40
3	1D	1	18	40	58
4	1E	1	5	58	63
5	1F2FDum	1	0	63	63
6	2C	1	18	63	81
7	2D	1	18	81	99
8	2E	1	5	99	104
9	2F3FDum	1	0	104	104
10	3C	1	18	104	122
11	3D	1	18	122	140
12	3E	1	5	140	145
13	3H	1	5	145	150
14	3I	1	21	150	171
15	3L	1	5	171	176

FIGURE 6. Critical Activities Schedule (Pilot Test)

Time-Cost Diagram produced from the script is shown in Fig. 7. The minimal cost for the project is Rp. 450.214.175,00, which means the optimum duration for this project can be found in the 12^{th} iteration with a total duration of 179 days.

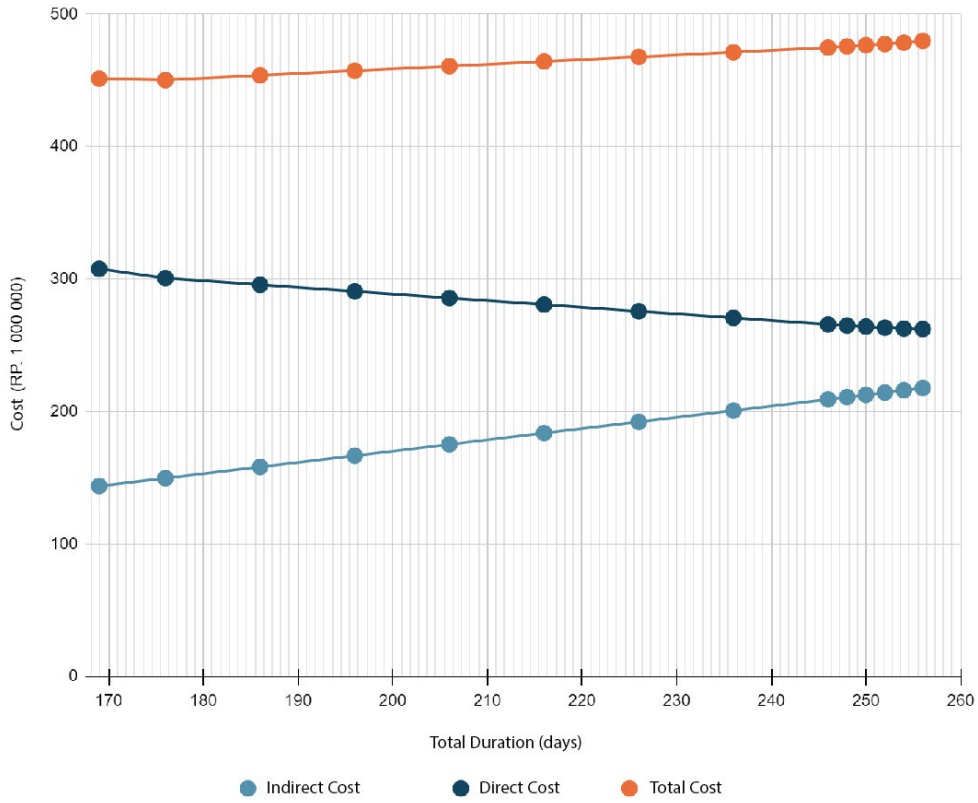

FIGURE 7. Time-Cost Diagram (Pilot Test)

Optimization for Project with Multiple Critical Paths

In this stage, the script designed in Pilot Test is tested against another project. The project data can be found in Fig. 8 and Table 3, with a daily indirect cost of Rp. 85.000,00/day. In this set of data, there are more than 1 critical path that needs to be handled by Python.

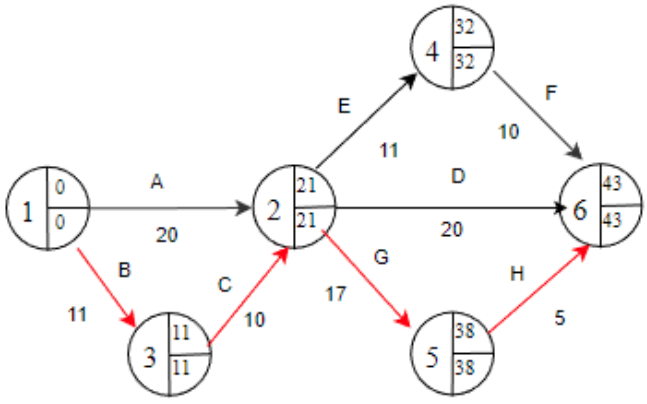

FIGURE 8. Arrow Diagram Network (Multiple Critical Paths)

TABLE 3. Project Data (Multiple Critical Paths)

act	ND (Rp. 1.000,00)	NC (Rp. 1.000,00)	CD (days)	CS (days)
A	20	500	9	50
B	11	200	5	5
C	10	100	5	25
D	20	300	9	20
E	11	200	5	20
F	10	100	5	30
G	17	100	10	15
H	5	150	3	25
Tot		1650		

Results (Multiple Critical Paths)

Project optimization is done by manual and dynamic programming method, and the results can be found in Table 4 and Table 5.

TABLE 4. Time-Cost Table for Multiple Critical Paths (Manual)

No.	Activity	CS (Rp.	Crash	D (days	Total Cost (Rp.
0	-			43	5305
1	B	5	1	42	5225
2	G	15	1	41	5155
3	GE	35	1	40	5105
4	GDE	55	5	35	5075
5	AB	55	5	30	4805
6	HDF	75	2	28	4785
7	AC	75	5	23	4735

TABLE 5. Time-Cost Table for Multiple Critical Paths (Dynamic Programming)

No.	Activity	CS (Rp. 1.000,00)	Crash	D (day	Total Cost (Rp.
0	-			43	5305
1	B	5	2	42	5230
2	G	15	1	41	5160
3	GE	35	1	40	5110
4	GDE	55	5	35	4980
5	HDF	75	5	33	4940
6	DF	50	1	33	4990
7	AB	55	5	28	4835
8	AC	75	5	23	4785

For the script to pass this stage, the Time-Cost Table in Table 5 must be similar to the one done manually (Table 4), however, there are significant differences in results.

1. Activity B is crashed for 2 days in Table 5 instead of 1 day. The root of the problem can be seen in the way the script detects the critical paths. In the 1st iteration, there are 2 critical paths, the first path consists of activities A – G – H and the second path consists of B – C

– G – H, and yet, the script can only detect one path, which is the one that consists of B – C – G – H. Activity B is chosen as Mincostindex as it has the cheapest cost slope, and yet the script is unable to take into account that to reduce the total duration of the project, activity A needs to be crashed at the same time as activity B, causing the total cost slope for crashing activity A and B to be Rp. 55.000,00. The script crashes activity B by another day, however, the total duration of the project does not decrease, this only causes A – G – H to be the only critical path for the following iteration, thus, causing most of the differences found in the next iterations.

2. Activity AB, with a total cost slope of be 55.000,00, which are supposed to be crashed after activity GDE, and yet is only crashed after activity HDF due to the first point (activity B).

These differences shows that there are 2 main problems for project optimization in the current script. First, the script's incapability to detect more than 1 critical path, and second, is the script's incapability to take into account the total cost slope of multiple activities. This shows that the script designed is incapable of processing most projects and thus, needs improvements.

Improvements to the Script (Multiple Critical Paths)

This stage uses the 2nd iteration as an example. The status of all the activities can be found in Fig. 9. As it is the second iteration, activity B is already crashed by one day.

FIGURE 9. Activity Statuses in 2nd Iteration

The improved script is developed to solve those two problems above. The first problem can be solved by adding additional constraints that limits the crashing options. In this case, the constraints are:

- If the duration of activity B equals to the duration of activity A + B, then activity A, B and C is part of the critical path, thus, the CP for each activity is 1.
- If the duration of activity G + H equals to the duration of activity E + F, then activity G, H, E and F is part of the critical path, thus, the CP for each activity is 1.
- If the duration of activity G + H equals to the duration of activity E + F and duration of activity D, then activity G, H, E, F and D is part of the critical path, thus, the CP for each activity is 1.
- If activity H can no longer be crashed, then activity H, D, E and F cannot be chosen as mincostindex.

These constraints are project specific and only apply for this project. The improved script produces a new result table (namely Crashablecritact Table) as shown in Fig.10.

	act	ND	NC	CD	CS	D	CP	maxC	C
0	B	11	200	5	5	10	1	6	1
1	G	17	100	10	15	17	1	7	0
2	C	10	100	5	25	10	1	5	0
3	H	5	150	3	25	5	1	2	0
4	A	20	500	9	50	20	1	11	0

FIGURE 10. Crashablecritact Table in 2nd Iteration (After Constraints)

The next problem is solved by fixing the process of choosing Mincostindex. A new temporary table, labeled jk_opt table, is used to process the choice. Besides activities that are required to be crashed at the same time at one point of the iteration process, Jk_opt also consists of the total cost slope for the activity set, and whether all activities in the column that are considered as critical path or not. In Fig. 10, activity A, B and C are included, therefore, in Jk_opt Table, activity AB and AC are part of the critical path. Activity A, B and C are thus, removed from the current Crashablecritact Table to form a new Crashablecritact Table in Fig. 10 (b).

	duration	act	CSt	CP
0	41	[G, E]	35	0
1	40	[G, D, E]	55	0
2	32	[A, B]	55	1
3	35	[G, D, F]	65	0
4	34	[H, D, F]	75	0
5	28	[A, C]	75	1

(a) Jk_opt Table

	act	ND	NC	CD	CS	D	CP	maxC	C
1	G	17	100	10	15	17	1	7	0
3	H	5	150	3	25	5	1	2	0

(b) Crashablecritact Table

FIGURE 11. Crashablecritact Table and Jk_opt Table in 2nd Iteration

Jk_opt table is used in tandem with the crashablecritact table to decide which activity needs to be crashed.

Additional constrains are added to choose between the two tables.

1. If the CP of all activities in Jk_opt Table is equals to 0, then the activity chosen will be from the Crashablecritact Table.
2. If there are no activities in Crashablecritact Table, then the activity chosen will be from the Jk_opt Table.
3. If the CP of some activities in Jk_opt Table and Crashablecritact Table is equals to 1, then the activity chosen will be the one with the lowest total cost slope between the two table.

In this case, the options available are activity AB and AC from Jk_opt Table, and activity G and H from Crashablecritact Table. The chosen activity is thus, activity G with 15 cost slope, which is the lowest cost slope (CS) between the 4 options.

New Results using the Improved Script

The improved script produces a time-cost table that is similar to Table 4. This shows that with additional commands and constraints, the script is capable of handling a project with multiple critical paths.

The newly improved script is tested against the data project on Pilot Test to ensure that the script is still capable of handling project with one critical path. The results produced is similar the time-cost table on Fig.5.

The newly improved script is shown to be capable of handling projects with several critical

paths and numerous of activities. With this, the script is declared completed.

CONCLUSION

Optimizing repetitive project schedule with dynamic programming method is possible to be done. With dynamic programming, it is possible to produce a Cost-Time Table and to choose the correct activity to be optimized. Dynamic programming is capable of producing start time and finish time of a critical activity, however, it cannot produce the start time and the finish time of a non-critical activity.

The script designed can be used for optimizing the duration of a repetitive project. The script designed works as intended, and is capable of making a Time-Cost Table that is similar to the one done manually.

After designing the script, project crashing using dynamic programming method can be repeated easily. Any errors made in the commands or the data inputs can be revised, and the Python program recalculates the inputs and gives out the new results accordingly. This flexibility also provides another advantage. The dynamic programming script for project crashing developed in this research can be used to solve similar projects by changing the data and slightly adjusting the script.

With this, it is concluded that the usage of dynamic programming is more efficient and faster in project optimization.

The script designed in this research can be found in:

1. Pilot Test Script:
   https://github.com/ericah-untar/ericah-untar/blob/main/.projectcrashing1jalurkritis
2. Multiple Critical Paths Script:
   https://github.com/ericah-untar/ericah-untar/blob/main/.projectcrashinglebihdari1jalurkritis

REFERENCES

1. Bhoyar, S., & Parbat, D. (2014). Repetitive project scheduling: developing CPM-like analytical capabilities. International Journal of Civil Engineering, 3(5), 37-46.
   Journal of Civil Engineering, vol. 3, no. 5, 2014, pp. 37 - 46.
2. Del Pico, W. J. (2023). Project control: Integrating cost and schedule in construction. John Wiley & Sons.
3. Hamdan, D. (2014). Model Kepemimpinan & system Pengambilan Keputusan. Bandung: Pustaka Setia.
4. Handoko, T. H. (2000). Manajemen Personalia dan Sumber Daya Manusia BPFE.
5. Hansen, S. (2015). Manajemen kontrak konstruksi. Gramedia Pustaka Utama.
6. Harris, R. B., & Ioannou, P. G. (1998). CENTER FOR CONSTRUCTION ENGINEERING AND MANAGEMENT.
7. Hegazy, T., Saad, D. A., & Mostafa, K. (2020). Enhanced repetitive-scheduling computation and visualization. Journal of Construction Engineering and Management, 146(10), 04020118.
8. Idris, I. (2014). Learning NumPy Array. Packt Publishing Ltd.

9. Ioannou, P. G., & Yang, I. T. (2016). Repetitive scheduling method: Requirements, modeling, and implementation. Journal of Construction Engineering and Management, 142(5), 04016002.
10. McKinney, W. (2012). Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. " O'Reilly Media, Inc.".
11. Molin, S., & Jee, K. (2021). Hands-On Data Analysis with Pandas: A Python data science handbook for data collection, wrangling, analysis, and visualization. Packt Publishing Ltd.
12. Mueller, J. P. (2023). Beginning programming with Python for dummies. John Wiley & Sons.
13. O'brien, J. J. (1993). CPM in construction management.
14. Didier, F., Perron, L., Mohajeri, S., Gay, S. A., Cuvelier, T., & Furnon, V. (2023). OR-Tools' Vehicle Routing Solver: a Generic Constraint-Programming Solver with Heuristic Search for Routing Problems.
15. Vanhoucke, M. (2012). Project management with dynamic scheduling (pp. 1-14). Springer Berlin Heidelberg.
16. Zhang, L. H. (2015). Repetitive project scheduling: Theory and methods. Elsevier.