JMTS: Jurnal Mitra Teknik Sipil
Vol. 6, No. 1, Februari 2023: hlm 133-144

EISSN 2622-545X

OPTIMASI PROYEK REPETITIF DENGAN METODE *DYNAMIC PROGRAMMING*

Erica Hosanna[1] dan Oei Fuk Jin[2]

[1]Program Studi Magister Teknik Sipil, Universitas Tarumanagara, Jl. Letjen S. Parman No. 1, Jakarta, Indonesia
erica.327202003@stu.untar.ac.id

[2]Program Studi Magister Teknik Sipil, Universitas Tarumanagara, Jl. Letjen S. Parman No. 1, Jakarta, Indonesia
fukjin.untar@gmail.com

Masuk: 21-06-2022, revisi: 02-02-2023, diterima untuk diterbitkan: 03-02-2023

ABSTRACT

Repetitive Scheduling Method is a scheduling method tailored specifically for projects with repetitive activities. Repetitive Scheduling Method ensures that repetitive activities are not interrupted, however, the resulting schedule is not optimized. Project optimization is done to optimize the cost and duration to be as optimized as possible, but applications of project optimization in repetitive projects are inefficient and leaves a lot of room for error. One possible solution is by using dynamic programming method to optimize the project, in which the optimization will be done through a program. This research is done on a housing settlement project in Bogor, West Java. Research analyses uses dynamic programming with the program Python and Google OR-Tools. This research starts by designing a set of dynamic programming script for project optimization, testing it on a project, and comparing it to manual application. Project optimization analyses using dynamic programming will result on comparisons between cost and duration in a project. Research results proves that the dynamic programming method are faster and more efficient than manual application.

Keywords: dynamic programming; repetitive scheduling method; project scheduling; proyek optimization; Python

ABSTRAK

Repetitive Scheduling Method adalah suatu metode penjadwalan yang dirancang khusus untuk proyek dengan pekerjaan berulang. *Repetitive Scheduling Method* memastikan kelangsungan aktivitas berulang tidak terinterupsi, tetapi jadwal yang dihasilkan biasanya memiliki biaya yang tidak optimal. Percepatan proyek bisa dilakukan untuk mempercepat durasi dan mengecilkan biaya, hanya, penerapan percepatan proyek pada proyek repetitif tidak efisien dan memiliki banyak ruang untuk kesalahan. Salah satu solusi yang bisa digunakan untuk optimasi proyek adalah *dynamic programming*, dimana data akan diolah oleh program komputer. Penelitian akan dilakukan pada sebuah proyek perumahan di Bogor, Jawa Barat. Analisis penelitian akan menggunakan metode *dynamic programming* dengan program Python dan Google OR-Tools. Penelitian ini dimulai dengan mengembangkan serangkaian *script dynamic programming* untuk optimasi proyek, mengujinya terhadap proyek, dan membandingkan dengan proses optimasi secara manual. Hasil analisis menunjukkan bahwa penggunaan *dynamic programming* lebih efisien dan lebih cepat dibanding dengan manual.

Kata kunci: pemrograman dinamis; penjadwalan proyek repetitif; penjadwalan proyek; optimasi proyek; Python

1. PENDAHULUAN

Project crashing adalah proses iterasi untuk menemukan durasi dengan biaya minimal pada suatu proyek. Setiap aktivitas pada jalur kritis dipelajari dan dipercepat sesuai dengan batasan yang tersedia, dengan harapan bahwa biaya percepatan proyek menurunkan biaya tidak langsung, dan secara keseluruhan, dapat menurunkan total biaya proyek.

Pada umumnya, untuk mengetahui durasi dan biaya optimal suatu proyek dimulai dengan menggunakan metode jaringan kerja. Dalam metode ini, jalur kritis dalam proyek terdeteksi, dan total durasinya dapat ditemukan dengan menambahkan durasi dari seluruh aktivitas pada jalur tersebut. *Project crashing* dilakukan pada aktivitas dalam jalur kritis dengan tampilan *Arrow Diagram Network* (ADN), dan prosesnya dicatat pada setiap iterasi (Pico, 2013).

Proyek repetitif adalah proyek dengan aktivitas berulang. Jaringan kerja untuk proyek repetitif umumnya disajikan dalam bentuk *repetitive scheduling method* (RSM). Metode RSM dapat menunjukan jadwal setiap aktivitas pada unit atau lokasi tertentu, tetapi RSM tidak dapat digunakan untuk mendapatkan durasi dan biaya percepatan aktivitas. Sehingga, untuk *project crashing* sebuah proyek repetitif kembali menggunakan tampilan *Arrow Diagram Method* (ADM) (Zhang, 2015).

Sementara itu, masalah terbesar dalam menggunakan tampilan ADN untuk mempercepat sebuah proyek repetitif adalah jumlah aktivitas, dimana jumlah aktivitas yang perlu diperhitungkan dalam sebuah proyek repetitif jauh melebihi jumlah aktivitas dalam proyek pada umumnya (Hegazy et al., 2020). Hal ini menyebabkan proses *crashing* yang panjang dan lama, serta, jika proses ini dilakukan secara manual, maka akan terdapat banyak ruang untuk kesalahan. Dalam proses optimasi ini *dynamic programming* dapat digunakan untuk memperhitungkan proses *crashing* (Vanhoucke, 2013).

Tujuan utama dalam penelitian ini adalah untuk melakukan optimasi penjadwalan proyek repetitif dengan menggunakan metode *dynamic programming* serta mengembangkan *script dynamic programming* yang bisa digunakan untuk crashing durasi proyek. Tujuan lain dari penelitian ini juga untuk melihat keunggulan apa saja yang diperoleh dari pengembangan *script* dengan metode *dynamic programming* ini.

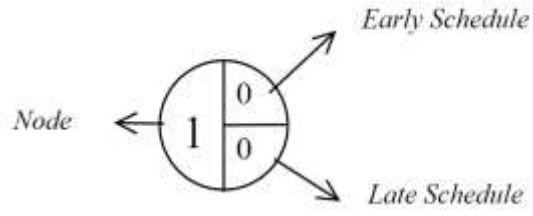
Dynamic programming

Dynamic Programming (DP) adalah sebuah teknik algoritma untuk menyelesaikan sebuah masalah. Prinsip utama dalam *dynamic programming* adalah pemecahan permasalahan yang rumit menjadi sub-masalah sederhana, dan setiap sub-masalah ini hanya memerlukan satu solusi optimal (Vanhoucke, 2013). Solusi optimal untuk sub-masalah digabung untuk menemukan solusi optimal untuk permasalahan utama.

Solusi untuk permasalahan DP dinamakan *script*. Sebuah *script* adalah gabungan perintah dan pernyataan dalam program yang dapat menyelesaikan sebuah masalah. Solusi untuk permasalah DP terdiri dari 4 bagian utama yaitu tahap (*stage*), pernyataan dan pernyataan variabel (*state and state variables*), pernyataan transisi (*state transition*) dan pilihan optimal. Ketika sebuah permasalahan DP dibagi menjadi sub-masalah sederhana, maka setiap sub-masalah adalah sebuah tahap dari solusi utama. Solusi akhir yang optimal didapatkan melalui pilihan terbaik pada setiap tahap. Dalam menyelesaikan sebuah sub-masalah dapat menggunakan beberapa pernyataan atau perintah. Pernyataan ini didefinisikan dengan sebuah parameter yaitu pernyataan variabel. Setelah solusi optimal untuk setiap tahap ditemukan, dibutuhkan sebuah pernyataan yang menghubungkan antara sub-masalah. Sebuah pernyataan transisi adalah bagaimana sebuah sub-masalah dihubungkan dengan sub-masalah lain. Pada setiap tahap akan menggunakan pilihan optimal untuk menghasilkan solusi akhir yang optimal.

Arrow Diagram Network (ADN)

Arrow Diagram Network (ADN) adalah sistem jaringan kerja dimana aktifitas dilambangkan dengan anak panah (*activity on the arc*) (Dimyati, 2014). Gambar 1 digunakan untuk memperhitungkan *Early Schedule* (ES) dan *Late Schedule* (LS). ES adalah waktu awal kejadian tersebut dapat dijadwalkan, sementara LS adalah waktu paling terakhir kejadian tersebut dapat dijadwalkan.



Gambar 1. Contoh simpul ADN

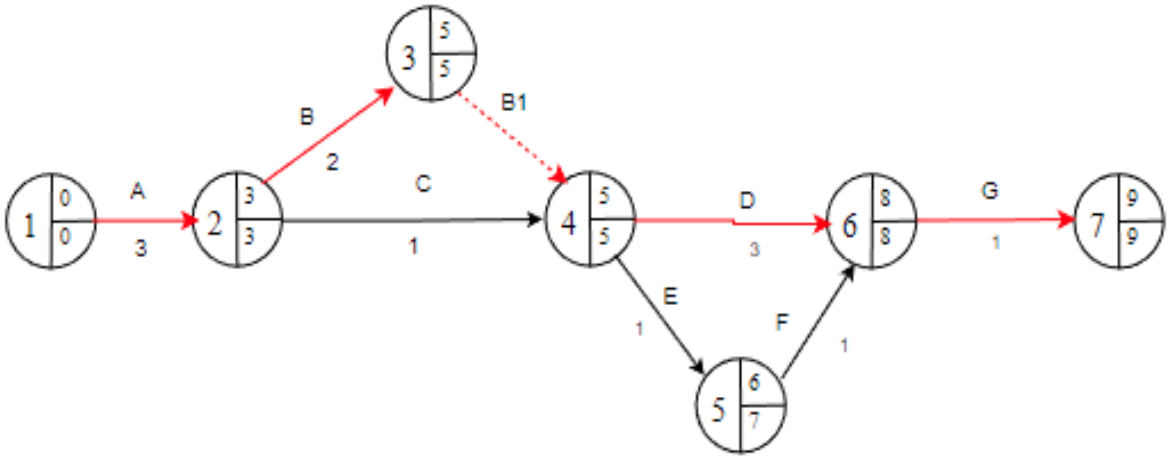
Simbol anak panah digunakan untuk merepresentasikan sebuah aktivitas. Anak panah dimulai dari satu simpul (*nodes*) ke simpul yang lain. Kepala anak panah menunjukkan arah kejadian (simpul) berikutnya. Sebuah simpul melambangkan suatu kejadian. Nomor simpul, nama aktivitas dan kode aktivasi merupakan penamaan unik dan tidak boleh diulang dalam proyek. Aktivitas dummy dilambangkan dengan anak panah dengan garis yang terputus-putus. Aktivitas dummy umumnya tidak memiliki durasi, tetapi digambarkan untuk menyatakan ketergantungan antara aktivitas (Gambar 2).

Jalur kritis pada sebuah proyek dapat terdeteksi jika nilai *total float* (TF) adalah 0. *Total float* adalah jumlah waktu yang tersedia untuk keterlambatan atau perlambatan pelaksanaan kegiatan tanpa mempengaruhi penyelesaian proyek secara keseluruhan. Nilai *total float* dapat dirumuskan sebagai berikut:

$$TF = LS(A) - ES(A) \quad (1)$$

$$FF = ES(B) - ES(A) \quad (2)$$

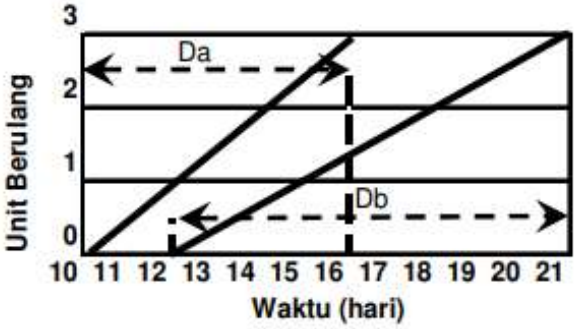
 EISSN 2622-545X



Gambar 2. Contoh ADN

Repetitive Scheduling Method (RSM)

Proyek dengan kegiatan yang sama dan berulang-ulang adalah sebuah proyek repetitif (Harris & Ioannou, 1998). RSM adalah metode penjadwalan menggunakan sumbu x dan y. Sumbu x menunjukkan waktu kerja dan sumbu y menunjukkan jumlah unit pekerjaan atau lokasi kegiatan yang dilaksanakan. Pada Gambar 3, Da adalah total durasi aktivitas A untuk 3 unit rumah. Aktivitas A dilakukan dari hari ke-10 hingga hari ke-16, sepanjang 6 hari. Aktivitas A untuk unit rumah pertama dilakukan pada hari ke-10 hingga hari ke-12. Db adalah durasi aktivitas B untuk 3 unit rumah.



Gambar 3. Contoh *repetitive scheduling method* (Harris & Ioannou, 1998)

Dengan Da = Durasi kegiatan A untuk jumlah total unit rumah dan Db = Durasi kegiatan B untuk jumlah total unit rumah.

Project crashing

Project Crashing atau percepatan proyek adalah metode untuk mengurangi durasi pengerjaan proyek yang dilakukan secara sistematis dan analitis (Hansen, 2015). Percepatan proyek biasanya dilakukan pada jalur kritis. Aktivitas yang dipilih untuk dipercepat adalah aktivitas yang memiliki *Cost Slope* (CS) paling rendah. *Cost Slope* (CS) adalah biaya yang ditambahkan jika durasi dipercepat selama 1 hari. Persamaan 1 adalah rumus (CS).

$$CS = \frac{CC - NC}{CD - ND} \quad (1)$$

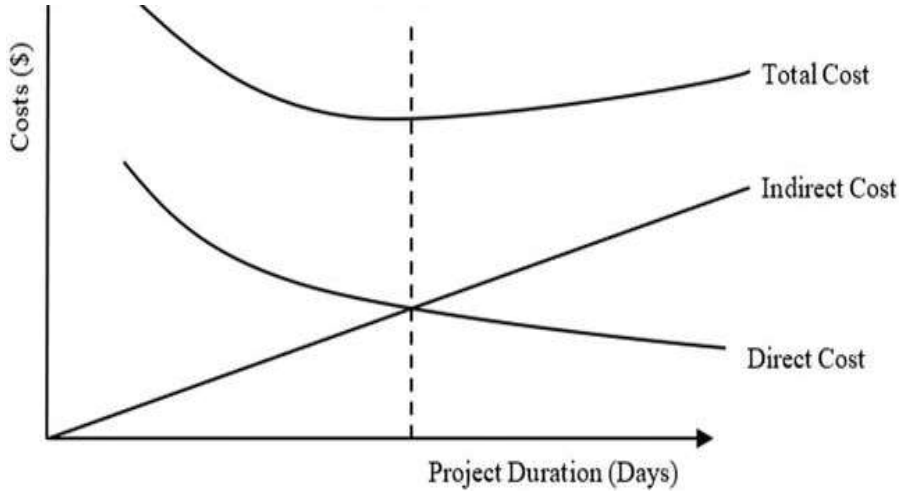
dengan ND = sebagai durasi normal dari suatu aktivitas (*normal duration)*, CD = sebagai durasi paling cepat dari suatu aktivitas (*crash duration)*, NC = sebagai biaya normal dari suatu aktivitas (*normal cost*) dan CC = sebagai biaya untuk waktu pengerjaan yang sudah dipercepat (*crash cost*).

Percepatan proyek berisiko menambahkan biaya proyek karena penambahan sumber daya maupun durasi kerja. Penambahan biaya dapat dikendalikan dengan metode *time-cost trade off* (Robinson, 1975). *Discrete time-cost trade off* (D-TCTO) adalah metode optimasi yang dikembangkan untuk menyelesaikan masalah penjadwalan dan biaya.

time-cost trade off sering digunakan di saat bersamaan dengan *project crashing* untuk mengendalikan perubahan biaya yang terjadi pada setiap percepatan.

D-TCTO memperhitungkan 3 biaya pada setiap iterasi, yaitu biaya langsung, biaya tidak langsung, dan biaya total. Biaya tidak langsung adalah biaya harian atau bulanan yang digunakan untuk keperluan di luar lapangan seperti administrasi dan manajemen. Biaya langsung adalah biaya yang digunakan untuk menjalankan aktivitas. Biaya total adalah biaya tidak langsung ditambah biaya langsung.

D-TCTO dapat digambarkan dalam bentuk grafik biaya-waktu seperti pada Gambar 4. Sumbu x merupakan durasi waktu berjalannya proyek, sementara sumbu y merupakan biaya. Tujuan umum menggunakan D-TCTO adalah untuk menentukan durasi proyek dengan biaya minimum.



Gambar 4. Contoh grafik biaya-waktu

2. METODE PENELITIAN

Pada penelitian ini digunakan pokok permasalahannya yaitu menyusun penjadwalan konstruksi untuk proyek berulang dengan *dynamic programming*. Dengan menggunakan metode ini, memungkinkan penjadwalan yang dapat dipercepat sesuai dengan batasan yang diinginkan oleh pihak pemilik. Penelitian ini, menggunakan studi kasus pembangunan proyek perumahan yang terletak pada daerah Bogor, Jawa Barat.

Penelitian dimulai dengan pengumpulan data melalui dokumen-dokumen terkait dengan proyek. Data tersebut lalu dirapikan sesuai dengan kebutuhan proyek. Data yang sudah diolah kemudian masuk ke tahap analisis. Analisis menggunakan data 3 unit rumah dan menggunakan 2 metode, yaitu secara manual dan secara *dynamic programming*. Sebuah *script* untuk *project crashing* dikembangkan untuk analisis secara *dynamic programming*. Jika proses iterasi pada kedua metode sama, maka *script* yang dikembangkan dinyatakan sukses. Proses penelitian dirangkum pada Gambar 5.

3. HASIL DAN PEMBAHASAN

Studi kasus

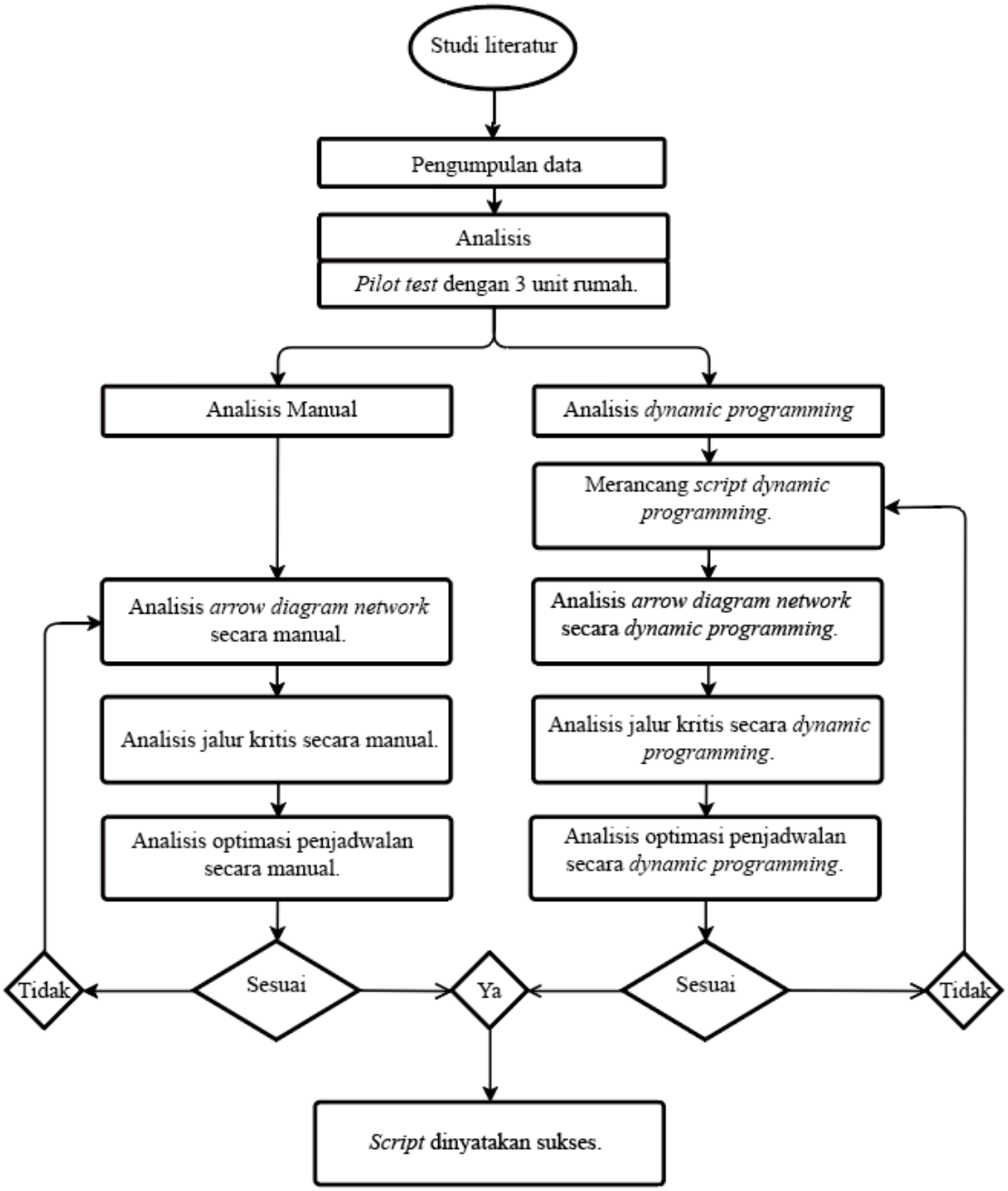
Pertama-tama, biaya pembangunan untuk 1 unit rumah harus dihitung terlebih dahulu dan dapat dilihat pada Tabel 1. Tabel 1 berisi aktivitas, durasi dan biaya normal, serta durasi dan biaya percepatan. Biaya tidak langsung yang dibutuhkan proyek adalah Rp. 850.000,00/ hari.

Tabel 1. Data proyek untuk 1 unit rumah

act	*Normal Duration (ND)* (hari)	*Normal Cost (NC)* (Rp)	*Crash Duration (CD)* (hari)	*Crash Cost (CC)*(Rp)	*Cost Slope (CS)*(Rp)
A	2	2.069.250,00	2	2.069.250,00	-
B	30	8.853.315,00	20	13.853.315,00	500.000,00
C	28	3.595.500,00	18	8.595.500,00	500.000,00

Tabel 1 (lanjutan). Data proyek untuk 1 unit rumah

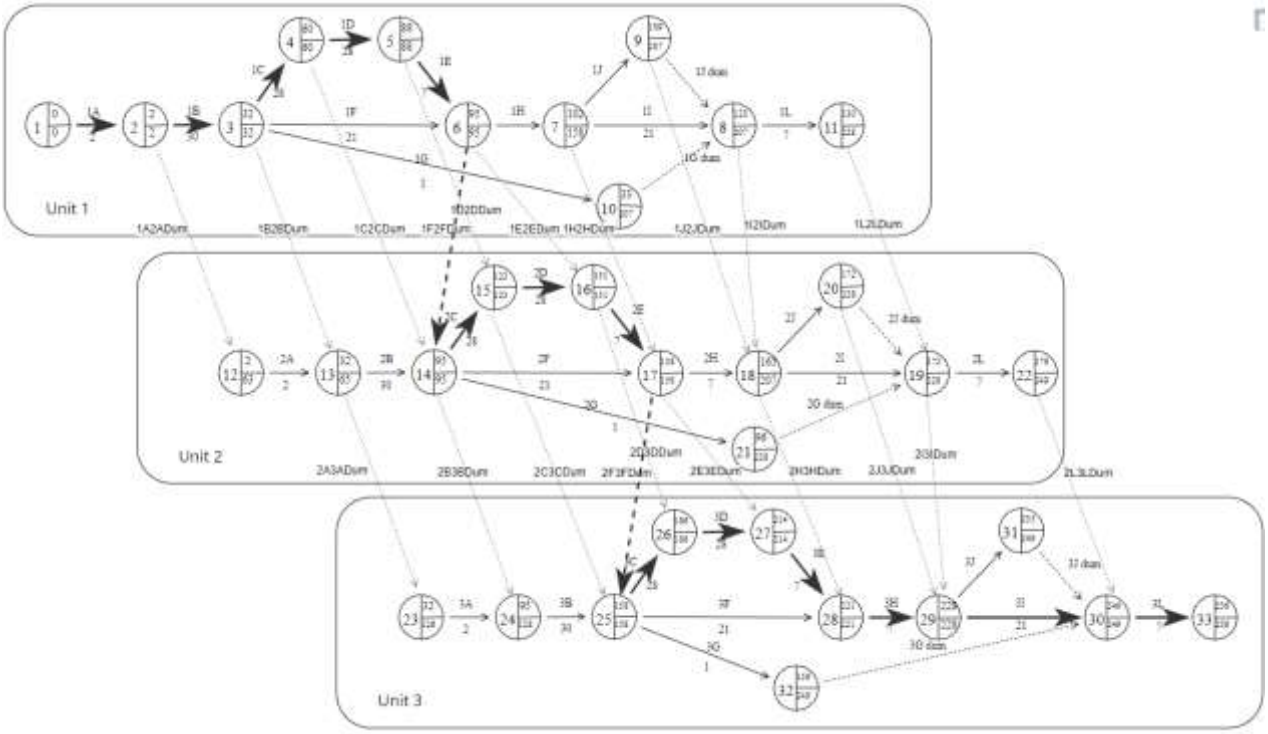
act	*Normal Duration (ND)* (hari)	*Normal Cost (NC)* (Rp)	*Crash Duration (CD)* (hari)	*Crash Cost (CC)*(Rp)	*Cost Slope (CS)*(Rp)
D	28	6.650.500,00	18	13.650.500,00	500.000,00
E	7	10.246.850,00	5	12.246,850,00	400.000,00
F	21	23.239.950,00	14	24.639.950,00	200.000,00
H	7	4.165.000,00	5	4.465.000,00	150.000,00
I	21	9.894.360,00	14	16.894.360,00	1.000.000,00
J	7	4.560.000,00	7	4.560.000,00	-
G	1	12.000.000,00	1	12.000.000,00	-
L	7	2.090.000,00	5	2.790.000,00	350.000,00
Total		87.404.725,00		136.704.725,00	



Gambar 5. Diagram alur penelitian

Diagram ADN untuk pekerjaan 3 unit rumah ditampilkan pada Gambar 6 yang menunjukkan hubungan aktivitas dalam 3 unit rumah tersebut. Aktivitas dimulai dari simpul ke-1 dan berakhir pada simpul ke-33. Unit rumah pertama dikerjakan pada simpul ke-1 hingga simpul ke-11, unit rumah ke-2 dikerjakan pada simpul ke-12 hingga simpul ke-22, unit rumah ke-3 dikerjakan pada simpul ke-23 hingga simpul ke 33. Anak panah merupakan alur aktivitas yang perlu dilakukan. Aktivitas A dimulai pada simpul ke-1 dan berakhir pada simpul ke-2. Aktivitas A diikuti dengan aktivtias B dan aktivitas A pada unit rumah berikutnya. Aktivitas B dimulai pada simpul ke-2 dan berakhir pada simpul ke-3, sementara aktivitas 1A2Adum adalah sebuah jalur dummy yang menyambung antara aktivitas 1A dan 2A.

Jalur kritis dengan nilai *total float* 0 tedeteksi berada pada simpul 1 - 2 - 3 - 4 - 5 - 6 - 14 - 15 - 16 - 17 - 25 - 26 - 27 - 28 - 29 - 30 - 33. Aktivitas yang terdapat pada jalur kritis antara lain: 1A - 1B - 1C - 1D - 1E - 1F2F dum - 2C - 2D - 2E - 2F3F dum - 3C - 3D - 3E - 3H - 3I - 3L. Seperti yang dapat terlihat pada Gambar 6, durasi yang dibutuhkan untuk membangun 3 rumah adalah 256 hari.



Gambar 6. ADN 3 unit rumah

Analisis secara manual

Untuk memulai *project crashing* secara manual, diketahui pada tahap 0, biaya langsung untuk pembangunan 3 rumah adalah Rp. 262.214.175,00. Durasi proyek adalah 256 hari sehingga biaya tidak langsung adalah Rp. 850.000,00*256 = Rp. 217.600.000,00. Dengan demikian, biaya total proyek adalah sebesar Rp. 479.814.175,00. Gambar 7 merupakan diagram ADN pada iterasi pertama. Aktivitas pada jalur kritis dengan *cost slope* termurah adalah aktivitas 3H dengan biaya Rp. 150.000,00 untuk percepatan 1 hari. Dengan mempercepat aktivitas 3H (Aktivitas pada simpul 28 - 29) sebanyak 2 hari, maka biaya langsung proyek bertambah Rp. 300.000,00 (Rp. 150.000,00 *2) dan biaya tidak langsung berkurang Rp. 1.700.000,00.

Durasi proyek menjadi 254 hari, sehingga biaya tidak langsung pada iterasi ini adalah Rp. 215.900.000,00 (Rp 850.000,00*254 hari), dan biaya langsung menjadi Rp. 262.514.175,00. Total biaya adalah Rp.478.414.175,00. Proses iterasi dilanjutkan hingga proyek tidak bisa dipercepat lagi. Hasil pada setiap iterasi dicatat dan dapat dilihat pada Tabel 2.

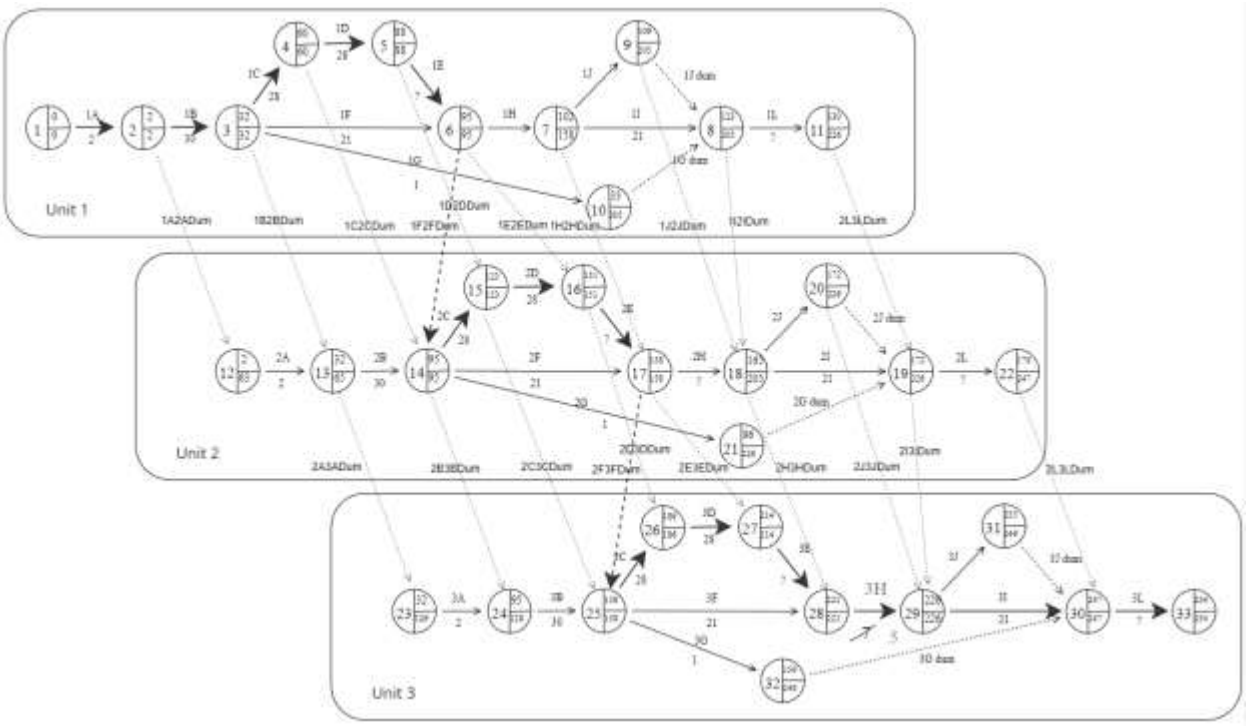
Tabel 2. Biaya-waktu secara manual

No. Iterasi	Aktifitas	*Crash* (hari)	Durasi	Biaya Tidak Langsung	Biaya Langsung	Total Biaya
0	-	-	256	Rp. 217.600.000,00	Rp 262.214.175,00	Rp. 479.814.175,00
1	3H	2	254	Rp. 215.900.000,00	Rp. 262.514.175,00	Rp. 478.414.175,00

 EISSN 2622-545X

Tabel 2 (lanjutan). Biaya-waktu secara manual

No. Iterasi	Aktifitas	*Crash* (hari)	Durasi	Biaya Tidak Langsung	Biaya Langsung	Total Biaya
2	3L	2	252	Rp. 214.200.000,00	Rp. 263.214.175,00	Rp. 477.414.175,00
3	1E	2	250	Rp. 212.500.000,00	Rp. 264.014.175,00	Rp. 476.514.175,00
4	2E	2	248	Rp. 210.800.000,00	Rp. 264.814.175,00	Rp. 475.614.175,00
5	3E	2	246	Rp. 209.100.000,00	Rp. 265.614.175,00	Rp. 474.714.175,00
6	1B	10	236	Rp. 200.600.000,00	Rp. 270.614.175,00	Rp. 471.214.175,00
7	1C	10	226	Rp. 192.100.000,00	Rp. 275.614.175,00	Rp. 467.714.175,00
8	2C	10	216	Rp. 183.600.000,00	Rp. 280.614.175,00	Rp. 464.214.175,00
9	3C	10	206	Rp. 175.100.000,00	Rp. 285.614.175,00	Rp. 460.714.175,00
10	1D	10	196	Rp. 166.600.000,00	Rp. 290.614.175,00	Rp. 457.214.175,00
11	2D	10	186	Rp. 158.100.000,00	Rp. 295.614.175,00	Rp. 453.714.175,00
12	3D	10	176	Rp. 149.600.000,00	Rp. 300.614.175,00	Rp. 450.214.175,00
13	3I	7	169	Rp. 143.650.000,00	Rp. 307.614.175,00	Rp. 451.264.175,00

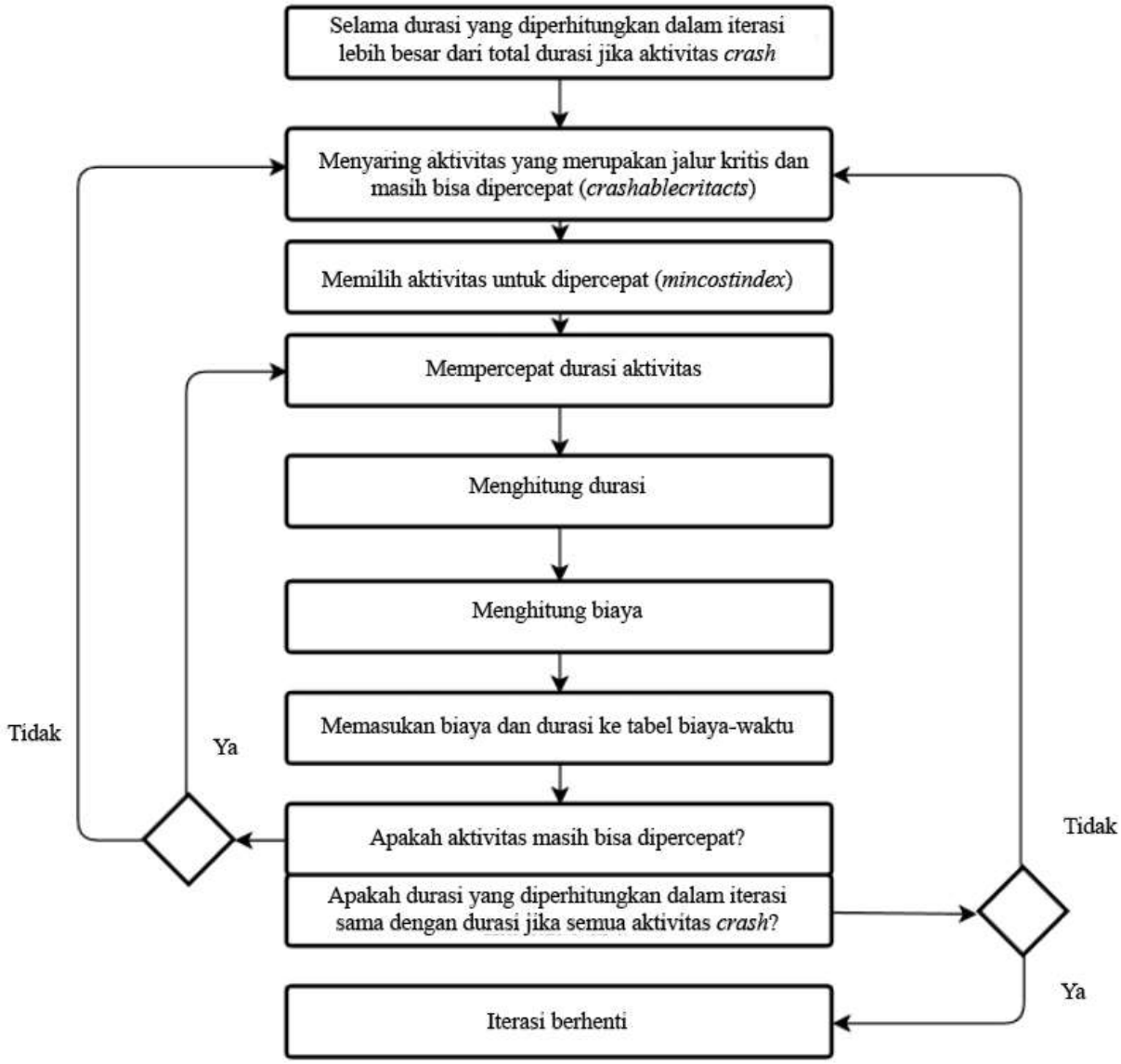


Gambar 7. ADN iterasi ke-1 manual

dengan aktivitas = aktivitas yang dipercepat, *crash* = jumlah hari aktivitas dipercepat, durasi = hasil durasi total setelah aktivitas dipercepat, biaya tidak langsung = Rp. 850.000,00*durasi, biaya langsung = total biaya aktivitas normal dan *crash*, dan biaya total = biaya tidak langsung + biaya langsung.

Analisis secara *dynamic programming*

Analisis secara *dynamic programming* dijalankan dengan menggunakan program Python dan OR-Tools. Proses *script dynamic programming* dapat dilihat pada Gambar 8, aktivitas disaring agar hanya aktivitas dalam jalur kritis dan yang masih bisa dipercepat dimasukkan ke tabel '*crashablecritacts'*. Dari tabel '*crashablecritacts'* dipilih aktivitas dengan penambahan *cost slope* paling kecil yaitu '*mincostindex'*. Aktivitas tersebut dipercepat, durasi baru dan biayanya akan dihitung. Durasi, biaya, dan aktivitas tersebut akan dimasukkan ke dalam tabel baru yaitu Tabel Biaya Waktu. Proses ini akan terus berulang hingga durasi yang dihitung sama dengan total durasi jika proyek dikerjakan dengan *crash*.



Gambar 8. Proses *script dynamic programming*

Script yang dihasilkan kemudian dijalankan dengan menggunakan data pada Tabel 1 dan Gambar 6. *Script* menghitung total durasi proyek dengan pengerjaan normal (256 hari) dan juga total durasi dengan *crash* (169 hari). Total biaya hasil perhitungan pun dapat dilihat pada Tabel 3. Sesuai dengan gambar 6, pada Tabel 3, iterasi berhenti jika durasi yang dihitung sama dengan total durasi dengan *crash* (169 hari).

Tabel 3. Biaya-waktu secara *dynamic programming*

No. Iterasi	Aktifitas	*Crash* (hari)	Durasi	Biaya Tidak Langsung	Biaya Langsung	Total Biaya
0	-	-	256	Rp. 217.600.000,00	Rp 262.214.175,00	Rp. 479.814.175,00
1	3H	2	254	Rp. 215.900.000,00	Rp. 262.514.175,00	Rp. 478.414.175,00
2	3L	2	252	Rp. 214.200.000,00	Rp. 263.214.175,00	Rp. 477.414.175,00
3	1E	2	250	Rp. 212.500.000,00	Rp. 264.014.175,00	Rp. 476.514.175,00
4	2E	2	248	Rp. 210.800.000,00	Rp. 264.814.175,00	Rp. 475.614.175,00
5	3E	2	246	Rp. 209.100.000,00	Rp. 265.614.175,00	Rp. 474.714.175,00
6	1B	10	236	Rp. 200.600.000,00	Rp. 270.614.175,00	Rp. 471.214.175,00
7	1C	10	226	Rp. 192.100.000,00	Rp. 275.614.175,00	Rp. 467.714.175,00
8	1D	10	216	Rp. 183.600.000,00	Rp. 280.614.175,00	Rp. 464.214.175,00
9	2C	10	206	Rp. 175.100.000,00	Rp. 285.614.175,00	Rp. 460.714.175,00

Tabel 3 (lanjutan). Biaya-waktu secara *dynamic programming*

No. Iterasi	Aktifitas	*Crash* (hari)	Durasi	Biaya Tidak Langsung	Biaya Langsung	Total Biaya
10	2D	10	196	Rp. 166.600.000,00	Rp. 290.614.175,00	Rp. 457.214.175,00
11	3C	10	186	Rp. 158.100.000,00	Rp. 295.614.175,00	Rp. 453.714.175,00
12	3D	10	176	Rp. 149.600.000,00	Rp. 300.614.175,00	Rp. 450.214.175,00
13	3I	7	169	Rp. 143.650.000,00	Rp. 307.614.175,00	Rp. 451.264.175,00

Perbandingan analisis manual dan *dynamic programming*

Hasil iterasi pada analisis secara manual dan analisis secara *dynamic programming* dibandingkan. Tabel 2 merupakan hasil iterasi secara manual dan Tabel 3 merupakan hasil iterasi secara *dynamic programming*. Kedua metode memiliki hasil pengurangan total biaya yang sama, tetapi dapat dilihat bahwa aktivitas yang digunakan terdapat sedikit perbedaan.

Pada total durasi 226 hari hingga 176 hari (Tabel 3), dapat dilihat bahwa aktivitas yang dipercepat adalah aktivitas C dan D. Aktivitas C dan D memiliki *cost slope* (CS) yang sama, yaitu Rp. 500.000,00. Analisis secara manual memilih aktivitas untuk dipercepat dengan urutan berikut : 1C, 2C, 3C, 1D, 2D dan 3D. Analisis secara *dynamic programming* memilih aktivitas untuk dipercepat dengan urutan berikut : 1C, 1D, 2C, 2D, 3C, dan 3D. Karena biaya CS yang sama, *dynamic programming* menggunakan urutan input aktivitas. Perbedaan pada aktivitas ini tidak terlalu berdampak pada hasil akhir percepatan karena kedua aktivitas memiliki penambahan biaya yang sama.

Tabel 2 dan Tabel 3 menghasilkan tabel biaya-waktu seperti diperlihatkan pada Gambar 9. Biaya tidak langsung berkurang pada setiap iterasi, sementara biaya langsung bertambah pada setiap iterasi. Proses project crashing berhenti pada iterasi ke-13 dengan total durasi 169 hari, setelah aktivitas 3I dipercepat. Aktivitas 3I merupakan aktivitas terakhir yang dapat dipercepat, dengan penambahan biaya Rp. 1.000.000,00. Dapat dilihat bahwa total biaya setelah percepatan aktivitas 3I meningkat dari Rp. 450.214.175,00 ke Rp. 451.264.175,00. Dengan adanya peningkatan total biaya tersebut, maka biaya optimum terdapat pada iterasi ke-12, setelah percepatan aktivitas 3D. Iterasi ke-12 juga ditandai dengan total biaya termurah diantara semua iterasi, yaitu sebesar Rp. 450.214.175,00 dengan durasi proyek 176 hari yang merupakan durasi optimum dalam proyek ini.

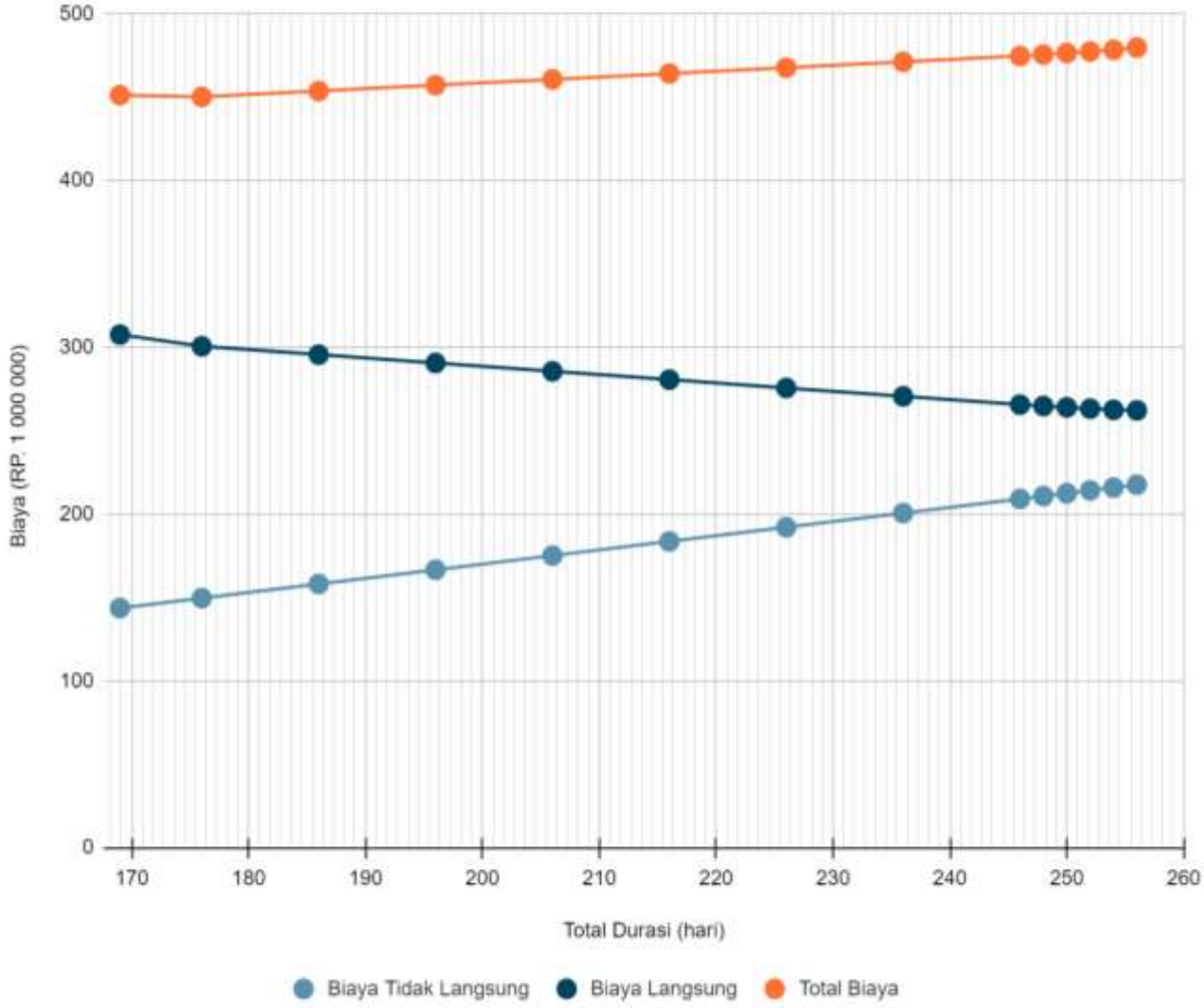
Jadwal Proyek

Biaya optimum yang terdapat pada iterasi ke-12, dengan durasi 176 hari dan biaya total sebesar Rp. 450.214.175,00, menghasilkan jadwal aktivitas kritis seperti yang diperlihatkan pada Tabel 4. Kemudian Tabel 4 menghasilkan jadwal optimal proyek yang terdapat pada Gambar 10.

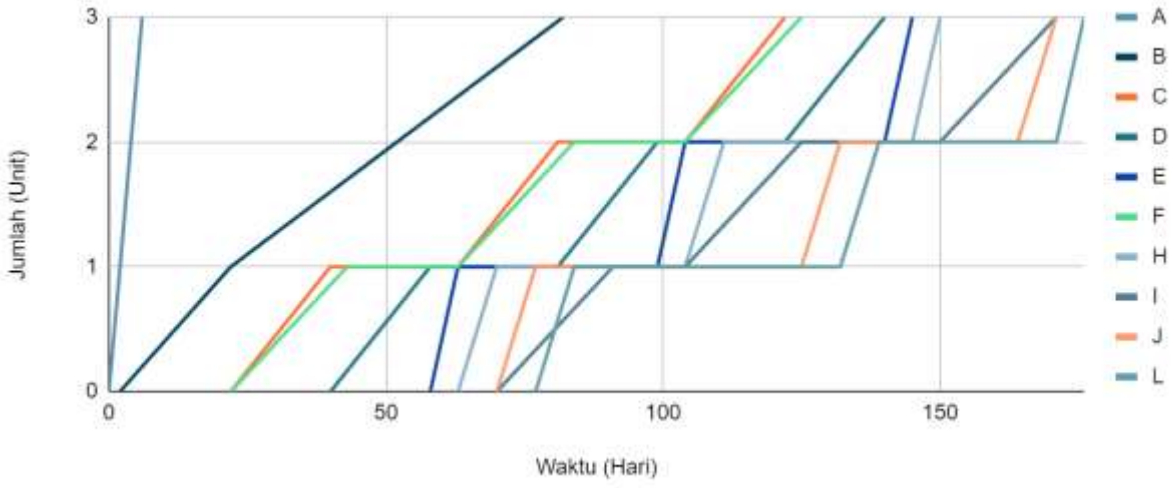
Tabel 4. Jadwal aktivitas kritis

No. Iterasi	Aktifitas	Durasi	Start Time (ST)	Finish Time (FT)
0	1A	2	0	2
1	1B	20	2	22
2	1C	18	22	40
3	1D	18	40	58
4	1E	5	58	63
5	1F2FDum	0	63	63
6	2C	18	63	81
7	2D	18	81	99
8	2E	5	99	104
9	2F3FDum	0	104	104
10	3C	18	104	122
11	3D	18	122	140
12	3E	5	140	145
13	3H	5	145	150
14	3I	21	150	171
15	3L	5	171	176

dengan aktivitas = aktivitas kritis, durasi = durasi yang digunakan setiap aktivitas setelah optimasi, dan kolom *'ST'* dan *'FT'* yang merupakan waktu mulai dan selesai untuk aktivitas tersebut.



Gambar 9. Grafik kurva biaya-waktu



Gambar 10. Jadwal optimal proyek repetitif

4. KESIMPULAN

Optimasi penjadwalan proyek repetitif dapat dilakukan dengan menggunakan metode *dynamic programming*. *Dynamic programming* dapat menghasilkan Tabel Biaya-Waktu dan dapat memilih aktivitas yang perlu dipercepat dalam proyek dengan benar. *Dynamic programming* juga dapat menghasilkan jadwal mulai dan berakhirnya suatu aktivitas dalam jalur kritis. *Dynamic programming* tidak dapat menghasilkan jadwal mulai dan berakhirnya suatu aktivitas yang tidak merupakan bagian dari jalur kritis.

Pengembangan *script dynamic programming* yang dihasilkan dapat digunakan untuk proses *crashing* durasi proyek. *Script* yang dikembangkan dapat berjalan dengan baik dan benar. Tabel Biaya-Waktu yang dihasilkan *dynamic programming* sesuai dengan Tabel Biaya-Waktu yang dihasilkan secara manual, dengan sedikit perbedaan yang tidak berdampak dalam pemilihan aktivitas karena memiliki *costslope* yang sama.

Setelah merancang script untuk proses *project crashing, project crashing* dengan *dynamic programming* dapat diulang dengan mudah, sehingga jika terjadi kesalahan pada input data, dapat diubah pada *dataframe* utama pada program Excel. *Dynamic programming* dapat menghitung sendiri dari awal hingga akhir proses dan mengeluarkan output baru yang sesuai dengan perubahan input. Flexibilitas pada tahap input data ini memberi keunggulan lain. *Script dynamic programming* untuk *project crashing* yang dikembangkan pada proyek ini dapat digunakan untuk proyek lain yang serupa dengan perubahan pada data proyek tersebut. Dengan demikian, penggunaan DP dalam optimasi proyek repetitif jauh lebih efisien, cepat, serta dapat meminimalisir kesalahan yang sering terjadi pada perhitungan manual.

DAFTAR PUSTAKA

Dimyati, H. (2014). *Model Kepemimpinan dan System Pengambilan Keputusan*. Pustaka Setia.

Harris, R. B., & Ioannou, P. G. (1998). Scheduling Projects with Repeating Activities. *Journal of Construction Engineering and Management*, *124*(4), 269–278. https://doi.org/10.1061/(asce)0733-9364(1998)124:4(269)

Hegazy, T., Saad, D. A., & Mostafa, K. (2020). Enhanced Repetitive-Scheduling Computation and Visualization. *Journal of Construction Engineering and Management*, *146*(10). https://doi.org/10.1061/(asce)co.1943-7862.0001911

Hansen, S. (2015). *Manajemen Kontrak Konstruksi*. Gramedia Pustaka.

Pico, W. D. J. (2013). *Project Control: Integrating Cost and Schedule in Construction* (edisi pertama). RSMeans.

Robinson, D. R. (1975). A Dynamic Programming Solution to Cost-Time Tradeoff for CPM. *Management Science*, *22*(2), 158–166. https://doi.org/10.1287/mnsc.22.2.158

Vanhoucke, M. (2013). *Project Management with Dynamic Scheduling: Baseline Scheduling, Risk Analysis and Project Control* (edisi kedua). Springer.

Zhang, L. (2015). *Repetitive Project Scheduling: Theory and Methods* (edisi pertama). Elsevier.