TUGAS AKHIR – KS184822

OPTIMASI BIAYA DAN WAKTU PROYEK PEMBANGUNAN JEMBATAN MENGGUNAKAN *CONSTRAINT PROGRAMMING* DAN *GENETIC ALGORITHM*

MOCH. ABDILLAH NAFIS
NRP 062116 4000 0062

Dosen Pembimbing
Dr. Dra. Kartika Fithriasari, M.Si.

PROGRAM STUDI SARJANA
DEPARTEMEN STATISTIKA
FAKULTAS SAINS DAN ANALITIKA DATA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2020

TUGAS AKHIR – KS184822

OPTIMASI BIAYA DAN WAKTU PROYEK PEMBANGUNAN JEMBATAN MENGGUNAKAN *CONSTRAINT PROGRAMMING* DAN *GENETIC ALGORITHM*

MOCH. ABDILLAH NAFIS
NRP 062116 4000 0062

Dosen Pembimbing
Dr. Dra. Kartika Fithriasari, M.Si.

PROGRAM STUDI SARJANA
DEPARTEMEN STATISTIKA
FAKULTAS SAINS DAN ANALITIKA DATA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2020

FINAL PROJECT – KS184822

OPTIMIZATION OF COST AND PERFORMANCE IMPROVEMENT OF BRIDGE CONSTRUCTION USING CONSTRAINT PROGRAMMING DAN GENETIC ALGORITHM

MOCH. ABDILLAH NAFIS
SN 062116 4000 0062

Supervisor
Dr. Dra. Kartika Fithriasari, M.Si.

UNDERGRADUATE PROGRAMME
DEPARTMENT OF STATISTICS
FACULTY OF SCIENCE AND DATA ANALYTICS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2020

LEMBAR PENGESAHAN

OPTIMASI BIAYA DAN WAKTU PROYEK PEMBANGUNAN JEMBATAN MENGGUNAKAN *CONSTRAINT PROGRAMMING* DAN *GENETIC ALGORITHM*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Statistika
pada
Program Studi Sarjana Departemen Statistika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember

Oleh :
Moch. Abdillah Nafis
NRP. 062116 4000 0062

Disetujui oleh Pembimbing:
Dr. Dra. Kartika Fithriasari, M.Si. ( )
NIP. 19691212 199303 2 002

Mengetahui,
Kepala Departemen

Dr. Dra. Kartika Fithriasari, M.Si.
NIP. 19691212 199303 2 002

SURABAYA, 14 JUNI 2020

(*Halaman ini sengaja dikosongkan*)

OPTIMASI BIAYA DAN WAKTU PROYEK PEMBANGUNAN JEMBATAN MENGGUNAKAN *CONSTRAINT PROGRAMMING* DAN *GENETIC ALGORITHM*

Nama Mahasiswa : Moch. Abdillah Nafis
NRP : 062116 4000 0062
Departemen : Statistika
Dosen Pembimbing : Dr. Dra. Kartika Fithriasari, M.Si.

Abstrak

Suatu proyek konstruksi memiliki rencana dan jadwal pelaksanaan untuk membatasi waktu penyelesaian sebuah pekerjaan. Namun tidak jarang rencana dan jadwal pelaksanaan yang telah dibuat tidak sesuai dengan kenyataan dilapangan, sehingga mengakibatkan keterlambatan dalam penyelesaian proyek. Ada dua constraints yang signifikan terhadap pelaksanaan proyek yaitu waktu dan biaya. Pada penelitian ini akan dilakukan upaya melakukan percepatan proyek yang belum dilaksanakan, dengan mempertimbangkan waktu dan biaya (crashing), dimana dalam penelitian ini adalah penambahan jumlah pekerja. Pemborong mengharapkan rencana proyek dapat dipercepat minimal 30 hari kerja. Metode yang digunakan adalah Constraint Programming dan Genetic Algorithm, dimana metode Constraint Programming akan diberi batasan jumlah iterasi, mengingat sangat banyaknya kombinasi persentase penambahan yang ada. Pada penelitian ini, digunakan dua evaluasi yaitu nilai cost slope, yaitu kerugian per hari yang timbul akibat adanya percepatan, dan lama waktu yang dibutuhkan untuk mendapatkan kombinasi persentase terbaik. Dihasilkan bahwa dalam kasus ini, metode Genetic Algorithm lebih baik daripada Constraint Programming, ditinjau dari nilai cost slope dan lama waktu untuk mendapatkan kombinasi persentase terbaik.

Kata Kunci: Biaya, Constraint Programming, Cost slope, Durasi, Genetic Algorithm, Jumlah Pekerja

(*Halaman ini sengaja dikosongkan*)

OPTIMIZATION OF TIME AND COST IN BRIDGE CONSTRUCTION USING CONSTRAINT PROGRAMMING AND GENETIC ALGORITHM

Name : **Moch. Abdillah Nafis**
Student Number : **062116 4000 0062**
Department : **Statistics**
Supervisor : **Dr. Dra. Kartika Fithriasari, M.Si.**

Abstract

A construction project has a plan and implementation schedule to limit the completion time of a job. But sometimes plans and implementation schedules that have been made were not match in the field, resulting in delays in project completion. There are two significant constraints for project implementation, time and cost. In this research, efforts will be made to accelerate projects that have not been implemented yet, taking into account the time and cost (crashing), in this research will be the addition of the number of workers. Contractor expects the project plan can be accelerated at least 30 working days. The methods used are Constraint Programming and Genetic Algorithm, where the Constraint Programming method will be limited by the number of iterations, since the number of combinations is very large. In this study, goodness of method will be evaluated by the value of the cost slope, which is the loss per day due to acceleration, and the length of time required to get the best percentage combination. It was found that in this case, the method of Genetic Algorithm is better than Constraint Programming, in terms of cost slope value and the length of time to get the best percentage combination.

Keywords: ***Constraint Programming, Cost, Cost slope, Duration, Genetic Algorithm, Number of Workers***

(*Halaman ini sengaja dikosongkan*)

KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa, atas segala berkat dan karunia-Nya, sehingga penulis dapat menyelesaikan laporan Tugas Akhir yang berjudul **"Optimasi Biaya dan Waktu Proyek Pembangunan Jembatan Menggunakan *Constraint Programming* dan *Genetic Algorithm*"** dengan lancar.

Penulis menyadari bahwa Tugas Akhir ini dapat terselesaikan tidak terlepas dari bantuan dan dukungan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada :

1. Dr. Dra. Kartika Fithriasari, M.Si. selaku Kepala Departemen Statistika dan dosen pembimbing Tugas Akhir yang senantiasa memberikan perhatian, bimbingan, dan pengarahan selama menyelesaikan Tugas Akhir ini.
2. Dr. Santi Wulan Purnami, S.Si., M.Si. selaku Sekretaris Departemen Statistika Bidang Akademik dan Kemahasiswaan, dan Dr. Vita Ratnasari, S Si., M.Si. selaku Sekretaris Departemen Statistika Bidang Sumber Daya Keuangan dan Manusia yang telah memberikan fasilitas, sarana, dan prasarana.
3. Dr. Suhartono selaku dosen wali selama masa studi yang telah banyak memberikan saran dan arahan dalam proses belajar di Departemen Statistika.
4. Dra. Wiwiek Setya Winahju, M.S. dan Dr. Irhamah, S.Si., M.Si. selaku dosen penguji yang telah banyak memberi masukan kepada penulis dalam penyelesaian Tugas Akhir.
5. Zony Zulfadli, S.T., M.T. yang telah memberikan kesempatan dan kepercayaan kepada penulis untuk mengolah dan menganalisis data Rencana Anggaran Biaya proyek pembangunan jembatan di CV. Cremona Teknik Consultant.
6. Orang tua dan keluarga yang selalu memberikan motivasi dan doa sehingga penulis dapat menyelesaikan Tugas Akhir ini.
7. Seluruh dosen dan *staff* pengajar Program Studi Sarjana Departemen Statistika FSAD ITS yang telah membantu penulis selama proses perkuliahan.
8. Teman-teman Statistika ITS angkatan 2016, yang selalu memberikan dukungan kepada penulis selama ini.

9. Semua pihak yang telah membantu dalam penyelesaian Tugas Akhir ini yang tidak bisa penulis sebutkan satu persatu.

Terlepas dari itu, penulis menyadari bahwa masih banyak kekurangan pada laporan ini baik dari segi penulisan maupun isi. Oleh karena itu, dengan terbuka kami menerima segala kritik dan saran dari pembaca sebagai bahan perbaikan bagi penulisan selanjutnya. Besar harapan penulis untuk mendapatkan kritik dan saran yang membangun sehingga Tugas Akhir ini dapat memberikan manfaat bagi semua.

Surabaya, Juni 2020

Penulis

DAFTAR ISI

Halaman

HALAMAN JUDUL i
COVER PAGE ii
LEMBAR PENGESAHAN iii
ABSTRAK..... v
ABSTRACT vii
KATA PENGANTAR ix
DAFTAR ISI xi
DAFTAR GAMBAR xiii
DAFTAR TABEL..... xv
DAFTAR LAMPIRAN xvii
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 4
1.3 Tujuan 5
1.4 Manfaat 5
1.5 Batasan Masalah 5
BAB II TINJAUAN PUSTAKA 7
2.1 Optimasi..... 7
2.2 *Constraint Programming* 10
2.3 *Genetic Algorithm*..... 11
2.3.1 Seleksi..... 16
2.3.2 *Crossover* 17
2.3.3 Mutasi 19
2.4 Proyek Konstruksi..... 20
2.5 *Activity Crashing* 21
BAB III METODOLOGI PENELITIAN..... 23
3.1 Sumber Data 23
3.2 Variabel Penelitian dan Struktur..... 24
3.3 Langkah Analisis 26
BAB IV ANALISIS DAN PEMBAHASAN 31
4.1 Rancangan Awal Jadwal Pelaksanaan Pembangunan Jembatan..... 31
4.2 Optimasi dengan *Constraint Programming*. 38

4.3 Optimasi dengan *Genetic Algorithm* 41
BAB V KESIMPULAN DAN SARAN 47
5.1 Kesimpulan 47
5.2 Saran 48
DAFTAR PUSTAKA 49
LAMPIRAN 53
BIODATA PENULIS 71

DAFTAR GAMBAR

Halaman

Gambar 2.1 Kerangka Berfikir Dari Algoritma Genetika.....14
Gambar 3.1 Hubungan Antara Variabel yang Digunakan Sebagai Perhitungan Optimasi 24
Gambar 3.2 Diagram Alir Penelitian 29
Gambar 4.1 Ilustrasi Rancangan Awal Jadwal Pelaksanaan Pembangunan Jembatan 31

(*Halaman ini sengaja dikosongkan*)

DAFTAR TABEL

Halaman

Tabel 2.1 Contoh Crossover 1 – Titik 18
Tabel 2.2 Contoh Crossover 2 – Titik 18
Tabel 2.3 Contoh Crossover Seragam 18
Tabel 2.4 Contoh Mutasi Pengkodean Biner 19
Tabel 2.5 Contoh Mutasi Pengkodean Permutasi 20
Tabel 3.1 Aktivitas Pembangunan Proyek Pembangunan Jembatan 23
Tabel 3.2 Variabel Penelitian 25
Tabel 3.3 Struktur Data Penelitian 25
Tabel 3.4 Ilustrasi Kombinasi Persentasi Penambahan Pekerja 26
Tabel 3.5 Ilustrasi Kromosom pada Populasi 27
Tabel 4.1 BPH Pembangunan Jembatan Setiap Jenis Pekerja ... 32
Tabel 4.2 Koefisien OH Pembangunan Jembatan 33
Tabel 4.3 Volume Pekerjaan dan Durasi Setiap Item Pekerjaan 35
Tabel 4.4 Jumlah Pekerja Sebelum Dilakukan Optimasi 37
Tabel 4.5 Kombinasi Terbaik dengan *Constraint Programming* 39
Tabel 4.6 Jumlah Pekerja Setelah Dilakukan Optimasi menggunakan *Constraint Programming* 39
Tabel 4.7 Ilustrasi Struktur Populasi 41
Tabel 4.8 Kombinasi Terbaik dengan *Genetic Algorithm* 42
Tabel 4.9 Jumlah Pekerja Setelah Dilakukan Optimasi menggunakan *Genetic Algorithm* 43
Tabel 4.10 Perbandingan Kedua Metode 45

(*Halaman ini sengaja dikosongkan*)

DAFTAR LAMPIRAN

Halaman

Lampiran 1. Perhitungan Nilai *Cost Slope* 53
Lampiran 2. Program *Python* untuk Menyusun Struktur Data .54
Lampiran 3. Program *Python* untuk *Constraint Programming* 57
Lampiran 4. Program *Python* untuk *Genetic Algorithm*.....60
Lampiran 5. Grafik *Fitness* Terbaik setiap Generasi pada 10 kali Percobaan Metode *Genetic Algorithm* 68
Lampiran 6. Catatan Lama Waktu *Running* pada 10 kali Percobaan Metode *Genetic Algorithm* 69
Lampiran 7. Surat Keterangan Pengambilan Data 70

(*Halaman ini sengaja dikosongkan*)

BAB I

PENDAHULUAN

Pendahuluan terdiri dari latar belakang, tujuan penelitian, manfaat penelitian, dan batasan masalah yang digunakan dalam penelitian ini.

1.1 Latar Belakang

Suatu proyek konstruksi memiliki rencana dan jadwal pelaksanaan untuk membatasi waktu penyelesaian sebuah pekerjaan. Namun tidak jarang rencana dan jadwal pelaksanaan yang telah dibuat tidak sesuai dengan kenyataan dilapangan, sehingga mengakibatkan keterlambatan dalam penyelesaian proyek. Ada dua *constraints* yang signifikan terhadap pelaksanaan proyek yaitu waktu dan biaya (Mohanty dkk, 2011).

Menurut (Assaf, A. dkk, 1995), penyebab keterlambatan antara lain dapat dilihat dari aspek bahan, pekerja, peralatan, keuangan, situasi, perubahan, hubungan dengan pemerintah, kontrak, waktu dan kontrol. Sedangkan menurut (Proboyo, 1999), secara umum keterlambatan proyek sering terjadi karena adanya perubahan perencanaan selama proses pelaksanaan, manajerial yang buruk dengan organisasi kontraktor, rencana kerja yang tidak terurus dengan baik dan terpadu, gambar dan spesifikasi yang tidak lengkap, ataupun kegagalan kontraktor dalam melaksanakan pekerjaan.

Permasalahan terkait rencana dan jadwal pelaksanaan sebuah proyek konstruksi pernah dibahas dalam penelitian oleh (Feng dkk, 1997) yang berjudul "*Using Genetic Algorithms to Solve Construction Time-Cost Trade-Off Problems*", yang menggunakan metode *Critical Path Method* (CPM) yang menggunakan pendekatan Algoritma Genetika, karena pendekatan eksak tidak cukup efisien untuk menangani jaringan CPM yang kompleks. Penelitian ini membahas prinsip Algoritma Genetika untuk menyelesaikan permasalahan *Time-Cost Trade-Off* dimana pada jalur kritis akan dilakukan pengurangan biaya seoptimal mungkin tanpa memberikan dampak yang signifikan pada waktu penyelesaian proyek.

Terdapat banyak jenis metode untuk mengalokasikan sumber daya pada sebuah proyek, salah tiganya yaitu metode konvensional, *heuristic* dan *metaheuristic*. Metode konvensional adalah algoritma yang menggunakan perhitungan matematis bia-sa. Ada beberapa metode konvensional yang biasa digunakan un-tuk melakukan pencarian jalur terpendek, diantaranya algoritma *Djikstraa*, algoritma *Floyd-Warshall*, dan algoritma *Bellman-Ford*. Metode *heuristic* adalah sub bidang dari kecerdasan buatan yang digunakan untuk melakukan pencarian dan penentuan jalur terpendek (Mutahkiroh dkk, 2007). Metode *metaheuristik* merupakan metode yang memiliki skema pencarian solusi yang terinspirasi dari prinsip-prinsip alamiah yang dikembangkan oleh makhluk hidup, antara lain: evolusi dan seleksi alam yang diadopsi oleh Algoritma Genetika, interaksi sosial dari sekawanan ikan maupun burung yang ditiru oleh *Particle Swarm Optimization* (PSO), maupun interaksi koloni semut untuk menemukan sumber makanan yang menginspirasi *Ant Colony Optimization* (ACO) (Prayogo dkk, 2017).

Salah satu metode matematis yang dapat digunakan untuk menangani kasus optimasi sumber daya pada proyek pembangunan jembatan ini adalah *Constraint Programming*, dimana *constraints* merupakan beberapa macam batasan yang ditentukan dan menjadi kendala yang mana harus dikurangi pelanggaran terhadap batasan tersebut. Pada penelitian yang dilakukan oleh (Firdaus, 2017) yang berjudul "Implementasi Penjadwalan Kuliah *Job Shop* Dengan Perancangan Jadwal Kuliah Menggunakan *Constraints Programming*", dilakukan pembuatan jadwal kuliah menggunakan metode *constraints programming* yang memenuhi *hard constraints*. Langkah awal yang dilakukan adalah mempresentasikan masalah penjadwalan sebagai sebuah *constraints satisfaction problem* (CSP), kemudian proses pencarian solusi CSP menggunakan *blind search algorithm* yaitu menguji seluruh ruang solusi yang ada. Hasil penelitian adalah suatu slot jadwal kuliah yang memenuhi *constraints* sehingga tidak terjadi bentrok dalam menyusun jadwal perkuliahan.

Algoritma Genetika adalah algoritma pencarian yang bekerja berdasarkan mekanisme seleksi alam dan genetika. Pada genetika, kromosom tersusun dari gen-gen. Tiap gen mempunyai sifat tertentu (*allele*), dan posisi tertentu (*locus*). Satu atau lebih kromosom bergabung membentuk paket genetik yang disebut *genotif*. Interaksi *genotif* dengan lingkungannya disebut *fenotif*. Pada algoritma genetik, kromosom berpadanan dengan string dan gen dengan karakter. Setiap karakter mempunyai posisi (*locus*) dan arti tertentu (*allele*). Satu atau lebih string bergabung membentuk struktur (*genotif*), dan bila struktur tersebut di-*decode*-kan akan diperoleh salah satu alternatif solusi (*fenotif*) (Saputro & Yento, 2020).

Pada penelitian yang dilakukan oleh (Hooshyar dkk, 2008) dengan judul "*A Genetic Algorithm to Time-Cost Trade off in Project Scheduling*", disajikan pendekatan Algoritma Genetika untuk menyelesaikan *Time-Cost Trade off Problem* (TCTP). Dalam algoritma ini diperkenalkan operator mutasi yang berdasarkan properti dari setiap proyek, dan terdapat dua variabel kontrol yang digunakan. Pada penelitian ini, dilakukan perbandingan efisiensi antara algoritma tersebut dengan algoritma Siemens. Hasil penelitian menunjukkan bahwa meskipun algoritma Siemens mampu secara optimal menyelesaikan contoh berukuran kecil, metode heuristik seperti Algoritma Genetika tetap dibutuhkan untuk memecahkan masalah yang lebih besar dan lebih kompleks.

Berdasarkan uraian diatas, pada penelitian ini akan dilakukan upaya untuk melakukan percepatan dengan mempertimbangkan waktu dan biaya (*crashing*), salah satunya yaitu penambahan jumlah pekerja menggunakan pendekatan eksak *Constraint Programming*, pendekatan *Genetic Algorithm*. Penelitian ini dilakukan pada proyek pembangunan jembatan yang belum dilaksanakan, dan pemborong mengharapkan penelitian ini akan menghasilkan kemungkinan-kemungkinan yang ada untuk mempercepat durasi proyek tanpa menambah biaya secara signifikan, untuk kemudian dari hal tersebut akan diambil keputusan untuk kegiatan ini. Pada penelitian ini, hal penting yang perlu dicatat yang pertama adalah membayar pekerja harus dilakukan secara harian (7 jam

kerja). Jika ada hari dimana pekerja tidak bekerja selama 7 jam karena item pekerjaan terkait sudah selesai di hari tersebut, maka hal tersebut adalah *waste* yang tentu saja harus diminimalisir. Yang kedua adalah jika dilakukan pengurangan durasi proyek terlalu besar, maka akan mengakibatkan perbandingan selisih waktu dan biaya sebelum dan sesudah optimasi, yang disebut *cost slope*, akan semakin besar. Artinya, usaha pengurangan waktu menjadi tidak berarti dibandingkan dengan dampak membengkaknya biaya proyek. Dua hal tersebut tentu saja akan dihindari oleh pemborong. Maka pada penelitian ini akan ditentukan batasan pada percepatan waktu proyek, yaitu minimal 30 hari kerja, maksimal 40% penambahan pekerja per item pekerjaan, dan akan ditinjau selisih antara biaya hasil optimasi dan biaya optimal sebenarnya pada setiap hari pengurangannya.

1.2 Rumusan Masalah

Akan dilakukan upaya untuk melakukan percepatan dengan mempertimbangkan waktu dan biaya (*crashing*) pada proyek pembangunan jembatan yang belum dilaksanakan dengan menambah jumlah pekerja, menggunakan pendekatan eksak *Constraint Programming* dan pendekatan *Genetic Algorithm*. Pemborong mengharapkan penelitian ini akan menghasilkan kombinasi terbaik dalam rangka mempercepat durasi proyek tanpa menambah biaya secara signifikan, untuk kemudian dari hal tersebut akan diambil keputusan untuk kegiatan ini. Pemborong mengharapkan rencana proyek dapat dipercepat minimal 30 hari kerja, sedangkan batas maksimal penambahan pekerja setiap item pekerjaan adalah 40%. Metode yang digunakan adalah *Constraint Programming* dan *Genetic Algorithm*, dimana metode *Constraint Programming* akan diberi batasan jumlah iterasi, mengingat sangat banyaknya kombinasi persentase penambahan yang ada. Pada penelitian ini, digunakan dua evaluasi yaitu nilai *cost slope*, yaitu kerugian per hari yang timbul akibat adanya percepatan, dan lama waktu yang dibutuhkan untuk mendapatkan kombinasi persentase terbaik. Kombinasi yang optimal, ditandai oleh nilai *cost slope* yang kecil, oleh sebab mempertimbangkan waktu dan biaya proyek. Sedangkan metode opti-

masi yang efisien adalah metode yang dapat menghasilkan nilai optimum dalam waktu yang singkat.

1.3 Tujuan

Tujuan yang ingin dicapai pada penelitian ini adalah sebagai berikut:

1. Mendapatkan persentase penambahan pekerja dengan nilai *cost slope* yang optimum menggunakan metode *Constraint Programming* dan *Genetic Algorithm*.
2. Membandingkan metode terbaik antara *Constraint Programming* dan *Genetic Algorithm* ditinjau dari nilai *cost slope* dan lama waktu untuk mendapatkan kombinasi persentase terbaik.

1.4 Manfaat

Hasil dari penelitian ini diharapkan dapat memberikan gambaran mengenai optimasi biaya dan waktu dari proyek bangunan, sehingga proyek sejenis di masa depan dapat direncanakan dan dikontrol dengan lebih baik. Selain itu, penelitian ini juga dapat digunakan sebagai informasi atau wawasan penggunaan metode *Constraint Programming* dan *Genetic Algorithm* terkait optimasi biaya dan waktu dari proyek bangunan.

1.5 Batasan Masalah

Batasan yang digunakan pada penelitian ini adalah sebagai berikut:

1. Pekerja yang dimaksud pada penelitian ini adalah pekerja yang memiliki *skill* tunggal, yaitu setiap pekerja yang sudah mengerjakan satu aktivitas tidak dapat mengerjakan aktivitas yang lainnya.
2. Pada penelitian ini hanya dilakukan optimasi pada jumlah pekerja.

(*Halaman ini sengaja dikosongkan*)

BAB II

TINJAUAN PUSTAKA

2.1 Optimasi

Dalam bahasa matematika, optimasi adalah model yang memiliki tujuan penyelesaian masalah dengan cara yang terbaik. Sebagai contoh dalam proyek pembangunan jembatan, tujuan tersebut dapat berupa meminimalkan beban jembatan atau memaksimalkan kekuatan jembatan (Griva dkk, 2009). Tujuan yang lebih khusus seperti fungsi tujuan yang dibahas dalam penelitian ini, yaitu meminimalkan waktu dan biaya dalam suatu proyek pembangunan jembatan.

Fungsi tujuan pada penelitian ini adalah meminimumkan *cost slope* berdasarkan persentase penambahan pekerja pada setiap item pekerjaan yang ada, karena *cost slope* adalah perbandingan antara selisih waktu dan biaya, sebelum dan sesudah proyek. Untuk lebih jelasnya, nilai *cost slope* dari sebuah aktivitas (item pekerjaan) adalah sebagai berikut (Hooshyar dkk, 2008):

$$CS = \frac{bf - ba}{da - df}, \tag{2.1}$$

dimana :

ba	:	Biaya awal
bf	:	Biaya setelah penambahan pekerja
CS	:	*Cost slope*
da	:	Durasi awal
df	:	Durasi setelah penambahan pekerja

Fungsi tujuan pada penelitian ini, yaitu meminimumkan *cost slope*, dinotasikan sebagai $f(x_1, x_2, \dots, x_N)$, dimana fungsi tersebut adalah sebagai berikut.

$$f(x_1, x_2, \dots, x_N) = CS, \tag{2.2}$$

dengan N adalah jumlah item pekerjaan, x_i adalah persentase penambahan pekerja pada item pekerjaan ke-i ($i = 1,2, \dots, N$), dan *CS* adalah nilai *Cost Slope* seperti pada persamaan (2.1), yang perhitungannya dijabarkan pada Lampiran 1. Adapun fungsi kendala yang harus dipenuhi pada penelitian ini yaitu

$$da - df \geq 30, \tag{2.3}$$

$$0 \leq x_i \leq 0.4, \tag{2.4}$$

Nilai durasi awal (*da*) setiap item pekerjaan sudah tersedia pada data. Setiap item pekerjaan memiliki nomor urutan pekerjaan, dan pada kasus ini urutan pekerjaan berbentuk kelompok urutan, dimana item pekerjaan pada kelompok urutan selanjutnya dapat dilaksanakan setelah semua item pekerjaan pada kelompok urutan sebelumnya diselesaikan. Sehingga program yang dibuat dalam penelitian ini akan disesuaikan untuk kasus serupa. Maka dari itu, untuk menghitung durasi awal total adalah berdasarkan jumlah dari maksimal durasi pada masing-masing kelompok.

Adapun biaya awal (*ba*) setiap item pekerjaan dihitung dari nilai volume pada item pekerjaan, Nilai OH (Orang Harian) awal pada setiap item pekerjaan dan jenis pekerja, serta biaya per hari untuk setiap jenis pekerja yang sudah tersedia pula pada data.

$$ba_{ij} = V_i \times OH_{ij} \times BPH_j, \tag{2.5}$$

$$ba = \sum_{i=1}^{N} \sum_{j=1}^{n} ba_{ij}, \tag{2.6}$$

dimana :

ba	:	Biaya awal
ba_{ij}	:	Biaya awal item pekerjaan ke-i pada jenis pekerja ke-j
BPH_j	:	Biaya per hari untuk jenis pekerja ke-j
OH_{ij}	:	Nilai OH (Orang Harian) awal pada item pekerjaan ke-i pada jenis pekerja ke-j

V_i : Volume item pekerjaan ke-i
n : Jumlah jenis pekerja

Jumlah pekerja pada masing-masing item pekerjaan dan jenis pekerja, didapatkan dari nilai volume dan durasi awal pada item pekerjaan, serta nilai OH awal pada setiap item pekerjaan dan jenis pekerja. Perhitungan jumlah pekerja adalah sebagai berikut.

$$JP_{ij} = \left\lceil \frac{V_i \times OH_{ij}}{da_i} \right\rceil \tag{2.7}$$

dimana :

da_i : Durasi awal item pekerjaan ke-i
JP_{ij} : Jumlah pekerja pada item pekerjaan ke-i untuk jenis pekerja ke-j
OH_{ij} : Nilai OH (Orang Harian) awal pada item pekerjaan ke-i pada jenis pekerja ke-j
V_i : Volume item pekerjaan ke-i

Perhitungan biaya setiap item pekerjaan setelah penambahan pekerja akan melibatkan perhitungan koefisien OH yang baru, dengan persamaan sebagai berikut.

$$bf_{ij} = V_i \times OHB_{ij} \times BPH_j, \tag{2.8}$$

$$bf = \sum_{i=1}^{N} \sum_{j=1}^{n} bf_{ij}, \tag{2.9}$$

$$OHB_{ij} = \frac{OH_{ij} \times [JP_{ij} \times (1 + x_i)]}{JP_{ij}}, \tag{2.10}$$

dimana :

bf : Biaya item pekerjaan setelah penambahan pekerja
bf_{ij} : Biaya item pekerjaan ke-i pada jenis pekerja ke-j setelah penambahan pekerja
BPH_j : Biaya per hari untuk jenis pekerja ke-j
JP_{ij} : Jumlah pekerja pada item pekerjaan ke-i untuk jenis pekerja ke-j

OH_{ij}	:	Nilai OH (Orang Harian) awal pada item pekerjaan ke-i pada jenis pekerja ke-j
OHB_{ij}	:	Nilai OH pada item pekerjaan ke-i pada jenis pekerja ke-j setelah penambahan pekerja
V_i	:	Volume item pekerjaan ke-i
n	:	Jumlah jenis pekerja
N	:	Jumlah item pekerjaan
x_i	:	Persentase penambahan pekerja pada item pekerjaan ke-i

Adapun untuk menghitung durasi setelah penambahan pekerja adalah sebagai berikut.

$$df_i = \frac{da_i \times \sum_{j=1}^{n} JP_{ij}}{\sum_{j=1}^{n} \lceil JP_{ij} \times (1 + x_i) \rceil}, \tag{2.11}$$

dimana :

da_i	:	Durasi awal item pekerjaan ke-i
df_i	:	Durasi item pekerjaan ke-i setelah penambahan pekerja
JP_{ij}	:	Jumlah pekerja pada item pekerjaan ke-i untuk jenis pekerja ke-j
n	:	Jumlah jenis pekerja
x_i	:	Persentase penambahan pekerja pada item pekerjaan ke-i

Kemudian perhitungan nilai *df* yang akan dimasukkan dalam perhitungan nilai *cost slope*, sama dengan cara menghitung nilai *da* yaitu jumlah dari maksimal durasi pada masing-masing kelompok urutan item pekerjaan. Sehingga setelah didapatkan nilai *ba*, *bf*, *da*, dan *df*, maka dapat dilanjutkan pada perhitungan nilai *cost slope*.

2.2 *Constraint Programming*

Menyelesaikan masalah ini dengan pendekatan eksak akan tidak efisien di dalam penelitian ini, karena fungsi tujuan meminimumkan *cost slope* pada penelitian ini dirasa cukup rumit, terutama dengan adanya peran pembulatan keatas dari hasil penambahan pekerja setiap unit pada setiap item pekerjaan. Maka akan diberi

batasan untuk penggunaan metode ini, yaitu dengan menentukan nilai-nilai persentase tertentu, kemudian kombinasi persentase terbaik akan didapatkan dari nilai-nilai tersebut.

Pada penelitian ini akan ditentukan nilai-nilai persentase tersebut, yaitu 0% yang artinya pada item pekerjaan tersebut tidak perlu dilakukan penambahan pekerja, dan 40% yang artinya pada item pekerjaan tersebut akan dilakukan penambahan pekerja maksimal. Permasalahan pada pelaksanaan proyek dalam waktu dan anggaran ini juga dapat ditangani dengan *Constraint Programming*.

Constraints programming adalah pembelajaran sistem komputasi berdasarkan *constraints*. Ide utama dari c*onstraints programming* ini adalah menyelesaikan masalah dengan menyatakan *constraints* (kondisi, sifat, kebutuhan) yang harus dipenuhi oleh solusi. Dengan kata lain, *constraints programming* adalah pendekatan alternatif himpunan kebutuhan (*constraints*) yang berurutan diselesaikan metode umum ataupun spesifik untuk domain (Firdaus, 2017).

Constraints merupakan beberapa macam batasan yang ditentukan dan menjadi kendala yang mana harus dikurangi pelanggaran terhadap batasan tersebut. Dalam hal ini terdapat dua jenis *constraint*, yaitu *hard constraint* dan *soft constraint*. *Hard constraints* merupakan suatu kondisi yang utama harus dipenuhi dan apabila dilanggar maka hasilnya tidak *valid*. *Soft constraints* merupakan suatu kondisi atau syarat yang masih bisa dilanggar, namun diharapkan kondisi tersebut terpenuhi untuk hasil penjadwalan yang optimal (Firdaus, 2017). Penelitian ini memiliki dua *hard constraint*, yaitu rencana proyek dapat dipercepat minimal 30 hari kerja, dan batas maksimal penambahan pekerja setiap item pekerjaan adalah 40%.

2.3 *Genetic Algorithm*

Mengingat bahwa pada penelitian ini metode eksak memiliki keterbatasan pada penentuan kombinasi persentase penambahan pekerja, maka dibutuhkan metode heuristik untuk memecahkan

masalah yang cukup kompleks seperti ini. Seorang manajer proyek tentu membutuhkan metode dengan hasil optimasi yang optimum dalam waktu yang singkat. Adapun pendekatan yang akan digunakan dalam penelitian ini adalah Algoritma Genetika.

Algoritma Genetika (*Genetic Algorithm*) sebagai cabang Algoritma Evolusi merupakan metode yang *adaptive* yang biasa digunakan untuk memecahkan suatu pencarian nilai dalam sebuah masalah optimasi. Algoritma ini didasarkan pada proses genetika yang ada dalam makhluk hidup, yaitu perkembangan generasi dalam suatu populasi yang alami, secara lambaut laun mengikuti prinsip proses seleksi alami yaitu "*siapa yang kuat dia yang bertahan (survive)*". Dengan meniru teori evolusi ini, Algoritma Genetika dapat digunakan untuk mencari solusi atas permasalahan dalam dunia nyata (Sivanadam, dkk, 2007).

Algoritma Genetika ini ditemukan di Universitas Michigan, Amerika Serikat oleh John Holland tahun 1975 melalui sebuah penelitian dan dipopulerkan oleh salah satu mahasiswanya, David Goldberg. Dia mendefinisikan Algoritma genetika ini sebagai metode algoritma pencarian berdasarkan pada mekanisme seleksi alam dan genetik alam. Algoritma genetika berusaha menerapkan pemahaman mengenai evolusi alamiah pada tugas-tugas pemecahan masalah (*problem solving*). Pendekatan yang diambil oleh algoritma ini adalah dengan menggabungkan secara acak berbagai pilihan solusi terbaik di dalam suatu kumpulan untuk mendapatkan generasi solusi terbaik berikutnya yaitu pada suatu kondisi yang memaksimalkan kecocokannya atau lazim disebut *fitness*. Generasi ini akan merepresentasikan perbaikan-perbaikan pada populasi awalnya. Dengan melakukan proses ini secara berulang, algoritma ini diharapkan dapat mensimulasikan proses evolusioner.

Ada 3 keuntungan utama dalam mengaplikasikan algoritma genetika pada masalah-masalah optimasi (Sam'ani, 2012):

a. Algoritma genetika tidak memerlukan kebutuhan matematis banyak mengenai masalah optimasi.

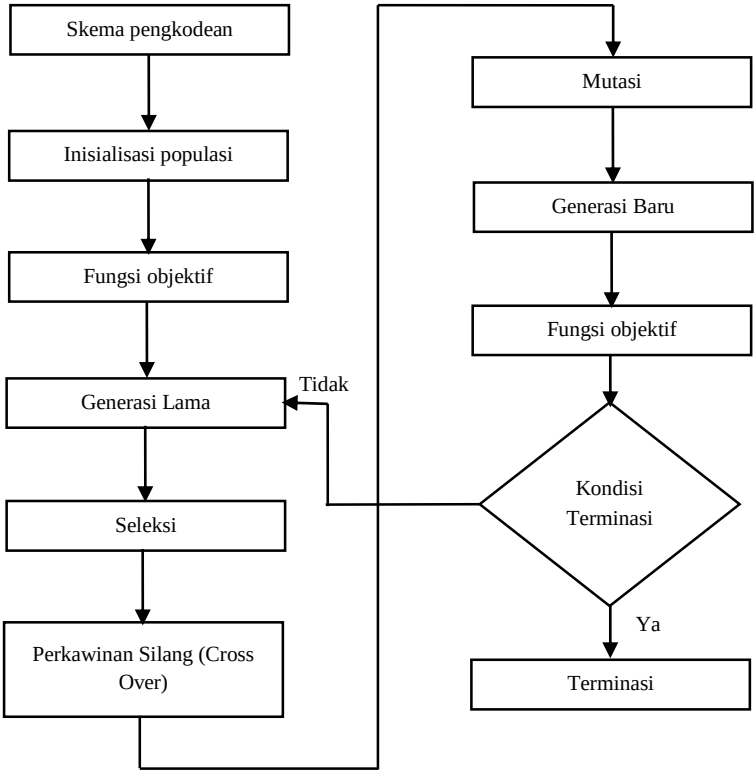
b. Kemudahan dan kenyamanan pada operator-operator evolusi membuat algoritma genetika sangat efektif dalam melakukan pencarian global.
c. Algoritma genetika menyediakan banyak fleksibelitas untuk digabungkan dengan metode heuristik yang tergantung domain, untuk membuat implementasi yang efisien pada masalah-masalah khusus.

Proses yang dilakukan dengan benar akan menghasilkan solusi-solusi yang paling tepat bagi permasalahan yang dihadapi. Untuk menggunakan algoritma genetik, solusi permasalahan direpresentasikan sebagai kromosom. Tiga aspek penting untuk penggunaan algoritma genetika yaitu:

1. Definisi fungsi *fitness*
2. Definisi dan implementasi representasi genetik
3. Definisi dan implementasi operasi genetik.

Jika ketiga aspek di atas telah didefinisikan, Algoritma genetika akan bekerja dengan baik. Tentu saja, Algoritma genetika bukanlah solusi terbaik untuk memecahkan segala masalah. Sebagai contoh, metode tradisional telah diatur untuk mencari penyelesaian dari fungsi analitis *convex* yang "berperilaku baik" yang variabelnya sedikit. Pada kasus-kasus ini, metode berbasis kalkulus lebih unggul dari Algoritma genetika karena metode ini dengan cepat menemukan solusi minimum ketika Algoritma genetika masih melakukan analisa bobot dari populasi awal. Algoritma genetika secara umum dapat diilustrasikan dalam kerangka berfikir pada Gambar 2.1. (Liu dkk, 2010)

Algoritma genetika memberikan suatu pilihan bagi penentuan nilai parameter dengan meniru cara reproduksi genetik, pembentukan kromosom baru serta seleksi alami seperti yang ter-jadi pada makhluk hidup. Metode ini juga sering disebut sebagai metode analisis heuristik. Dalam kasus pada penelitian ini, pendekatan tersebut akan digunakan untuk mendapatkan kombinasi yang terbaik dari persentase penambahan pekerja setiap item pekerjaan, yaitu kombinasi dengan nilai *cost slope* minimum.



Gambar 2.1 Kerangka Berfikir Dari Algoritma Genetika

Menurut (Goldberg, 1989), Algoritma genetika mempunyai karakteristik-karakteristik yang perlu diketahui sehingga dapat dibedakan dari prosedur pencarian atau optimasi yang lain, yaitu:

1. Algoritma genetika dengan pengkodean dari himpunan solusi permasalahan berdasarkan parameter yang telah ditetapkan dan bukan parameter itu sendiri.
2. Algoritma genetika pencarian pada sebuah solusi dari sejumlah individu individu yang merupakan solusi permasalahan bukan hanya dari sebuah individu.
3. Algoritma genetika informasi fungsi objektif (*fitness*), sebagai cara untuk mengevaluasi individu yang mempunyai solusi terbaik, bukan turunan dari suatu fungsi.

4. Algoritma genetika menggunakan aturan-aturan transisi peluang, bukan aturan-aturan deterministik.

Variabel dan parameter yang digunakan pada Algoritma ini adalah:

1. Fungsi *fitness* (fungsi tujuan) yang dimiliki oleh masing-masing individu untuk menentukan tingkat kesesuaian individu tersebut dengan kriteria yang ingin dicapai.
2. Populasi jumlah individu yang dilibatkan pada setiap generasi.
3. Probabilitas terjadinya persilangan (*crossover*) pada suatu generasi.
4. Probabilitas terjadinya mutasi pada setiap individu.
5. Jumlah generasi yang akan dibentuk yang menentukan lama *running*.

Secara umum struktur dari suatu Algoritma genetika dapat didefinisikan dengan langkah-langkah sebagai berikut:

1. Membangkitkan populasi awal

Dibangkitkan secara random sehingga didapatkan solusi awal. Populasi terdiri atas sejumlah kromosom yang merepresentasikan solusi yang diinginkan.

2. Membentuk generasi baru

Untuk membentuk generasi baru, digunakan operator reproduksi/ seleksi, *crossover* dan mutasi. Proses ini dilakukan berulang-ulang sehingga didapatkan jumlah kromosom yang cukup untuk membentuk generasi baru dimana generasi baru ini merupakan representasi dari solusi baru. Generasi baru ini dikenal denga istilah anak (*offspring*).

3. Evaluasi solusi

Pada tiap generasi, kromosom akan melalui proses evaluasi dengan menggunakan alat ukur yang dinamakan *fitness*. Nilai *fitness* suatu kromosom menggambarkan kualitas kromosom da-lam populasi tersebut. Proses ini akan mengevaluasi setiap populasi dengan menghitung nilai *fitness* setiap kromosom dan mengevaluasinya sampai terpenuhi criteria berhenti. Bila kriteria berhenti belum terpenuhi maka akan dibentuk lagi generasi baru dengan mengulangi langkah dua. Beberapa kriteria berhenti sering

digunakan antara lain: berhenti pada generasi tertentu, berhenti setelah dalam beberapa generasi berturut-turut didapatkan nilai *fitness* tertinggi tidak berubah, berhenti dalam *n* generasi tidak didapatkan nilai *fitness* yang lebih tinggi.

Karena penelitian ini, fungsi tujuan adalah meminimalkan nilai *cost slope*, dan adanya aturan bahwa individu yang memiliki nilai *fitness* tinggi lebih mampu bertahan hidup pada generasi berikutnya (Hasan, U. dkk, 2018), maka nilai *fitness* yang bisa digunakan adalah:

$$f_i = \frac{1}{1 + CS_i}, \tag{2.12}$$

dimana :

f_i : *Fitness* pada individu ke-i

CS_i : *Cost slope* pada individu ke-i

Algoritma genetika merupakan proses pencarian yang heuristik dan acak sehingga penekanan pemilihan operator yang digunakan sangat menentukan keberhasilan Algoritma genetika dalam menemukan solusi optimum dari suatu masalah yang diberikan. Hal yang harus diperhatikan adalah menghindari terjadinya konvergensi *premature*, yaitu mencapai solusi optimum yang belum waktunya, dalam arti bahwa solusi yang diperoleh adalah hasil optimum lokal. Operator genetik yang digunakan setelah proses evaluasi tahap pertama membentuk populasi baru dari generasi lama. Operator-operator tersebut adalah operator seleksi, *crossover* dan mutasi (Kulcsar dkk, 2013).

2.3.1 Seleksi

Seleksi bertujuan memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang paling *fit*. Langkah pertama dalam seleksi ini adalah mendapatkan nilai *fitness*. Masing-masing individu dalam suatu wadah seleksi akan menerima probabilitas reproduksi yang tergantung pada nilai objektif dirinya sendiri terhadap nilai objektif dari semua individu dalam wadah seleksi tersebut.

Kemampuan Algoritma genetika untuk memproduksi kromosom yang lebih baik secara progresif tergantung pada penekanan selektif (*selective pressure*) yang diterapkan ke populasi. Penekanan selektif dapat diterapkan dalam dua cara. Cara pertama adalah membuat lebih banyak kromosom anak yang dipelihara dalam populasi dan memilih hanya kromosom-kromosom terbaik bagi generasi berikut. Walaupun orang tua dipilih secara acak, metode ini akan terus menghasilkan kromosom yang lebih baik berhubungan dengan penekanan selektif yang diterapkan pada individu anak tersebut.

Cara lain menerapkan penekanan selektif adalah memilih orang tua yang lebih baik ketika membuat keturunan baru. Dengan metode ini, hanya kromosom sebanyak yang dipelihara dalam populasi yang perlu dibuat bagi generasi berikutnya. Walaupun penekanan selektif tidak diterapkan ke level keturunan, metode ini akan terus menghasilkan kromosom yang lebih baik, karena adanya penekanan selektif yang diterapkan ke orangtua. Ada beberapa metode untuk memilih kromosom yang sering digunakan antara lain seleksi roda rolet (*roulette wheel selection*), seleksi ranking (*rank selection*) dan seleksi turnamen (*tournament selection*).

Pada penelitian ini akan digunakan proses seleksi turnamen. Proses seleksi oleh turnamen ini lebih efisien daripada metode seleksi lainnya dan mengarah ke solusi optimal (Kristiadi & Hartanto, 2019). Sebelum proses seleksi berdasarkan turnamen dimulai, ia harus menentukan jumlah peserta (N). Kemudian, pemilihan turnamen secara acak memilih N kandidat dari populasi. Pemenang turnamen adalah kromosom dengan nilai *fitness* tertinggi. Proses seleksi ini diulang sampai 2 kromosom dipilih sebagai pasangan induk, yang kemudian dilanjutkan pada tahap *crossover*.

2.3.2 *Crossover*

Crossover (perkawinan silang) bertujuan menambah variasi string dalam populasi dengan penyilangan antar-string yang diperoleh dari sebelumnya. Beberapa jenis *crossover* tersebut adalah:

a. *Crossover* 1-titik

Pada *crossover* dilakukan dengan memisahkan suatu string menjadi dua bagian dan selanjutnya salah satu bagian dipertukarkan dengan salah satu bagian dari string yang lain yang telah dipisahkan dengan cara yang sama. Proses yang demikian dinamakan operator *crossover* satu titik seperti diperlihatkan pada tabel 2.1.

Tabel 2.1 Contoh *Crossover* 1 - Titik

Kromosom Orangtua 1	11001011
Kromosom Orangtua 2	11011111
Keturunan	11001111

b. *Crossover* 2-titik

Proses *crossover* ini dilakukan dengan memilih dua titik silang (*crossover*). Kromosom keturunan kemudian dibentuk dengan barisan bit dari awal kromosom sampai titik *crossover* pertama disalin dari orangtua pertama, bagian dari titik *crossover* pertama dan kedua disalin dari orangtua kedua, kemudian selebihnya disalin dari orangtua pertama lagi seperti diperlihatkan pada tabel 2.2.

Tabel 2.2 Contoh *Crossover* 2 - Titik

Kromosom Orangtua 1	11001011
Kromosom Orangtua 2	11011111
Keturunan	11011111

c. *Crossover* seragam

Crossover seragam menghasilkan kromosom keturunan, yaitu dengan menyalin bit-bit secara acak dari kedua orangtuanya seperti diperlihatkan pada tabel 2.3.

Tabel 2.3 Contoh *Crossover* Seragam

Kromosom Orangtua 1	11001011
Kromosom Orangtua 2	11011111
Keturunan	11011111

2.3.3 Mutasi

Mutasi merupakan proses mengubah nilai dari satu atau beberapa gen dalam suatu kromosom. Operasi *crossover* yang dilakukan pada kromosom dengan tujuan untuk memperoleh kromosom-kromosom baru sebagai kandidat solusi pada generasi mendatang dengan *fitness* yang lebih baik, dan lama-kelamaan menuju solusi optimum yang diinginkan.

Untuk menghindari konvergensi *premature* dan tetap menjaga perbedaan (*diversity*) kromosom-kromosom dalam populasi, selain melakukan penekanan selektif yang lebih efisien, operator mutasi juga dapat digunakan. Proses mutasi dalam sistem biologi berlangsung dengan mengubah isi *allele* gen pada suatu *locus* dengan *allele* yang lain. Proses mutasi ini bersifat acak sehingga tidak selalu menjamin bahwa setelah proses mutasi akan diperoleh kromosom dengan *fitness* yang lebih baik. Operator mutasi merupakan operasi yang menyang-kut satu kromosom tertentu. Beberapa cara operasi mutasi diterapkan dalam algoritma genetika menurut jenis pengkodean terhadap *phenotype*, antara lain (Hasan, U. dkk, 2018):

a. Mutasi dalam Pengkodean Biner

Mutasi pada pengkodean biner merupakan operasi yang sangat sederhana. Proses yang dilakukan adalah menginversi nilai bit pada posisi tertentu yang terpilih secara acak (atau menggunakan skema tertentu) pada kromosom, yang disebut *inverse* bit seperti diperlihatkan pada tabel 2.4.

Tabel 2.4 Contoh Mutasi Pengkodean Biner

Kromosom sebelum mutasi	1 0 0 1 0 1 1 1
Kromosom setelah mutasi	1 0 0 1 0 0 1 1

b. Mutasi dalam Pengkodean Permutasi

Proses mutasi yang dilakukan dalam pengkodean biner, namun dengan mengubah langsung bit-bit pada kromosom tidak dapat dilakukan pada pengkodean permutasi karena konsistensi urutan permutasi harus diperhatikan. Salah satu cara yang dapat dilakukan adalah dengan memilih dua posisi (*locus*) dari kromosom

dan kemudian nilainya saling dipertukarkan seperti diperlihatkan pada tabel 2.5.

Tabel 2.5 Contoh Mutasi Pengkodean Permutasi

Kromosom sebelum mutasi	1 2 3 4 6 5 8 7 9
Kromosom setelah mutasi	1 2 7 4 6 5 8 3 9

c. Mutasi dalam Pengkodean Nilai

Mutasi pada pengkodean nilai hampir sama dengan yang dilakukan pada pengkodean biner, tetapi yang dilakukan bukan melakukan inversi nilai bit. Penerapannya bergantung pada jenis nilai yang digunakan. Sebagai contoh untuk nilai riil, proses mutasi dapat dilakukan seperti yang dilakukan pada pengkodean permutasi, dengan saling mempertukarkan nilai dua gen pada kromosom.

d. Mutasi dalam Pengkodean Pohon

Mutasi dalam pengkodean pohon dapat dilakukan antara lain dengan cara mengubah operator (+, -, *, /) atau nilai yang terkandung pada suatu verteks pohon yang dipilih. Atau, dapat ju-ga dilakukan dengan memilih dua verteks dari pohon dan saling mempertukarkan operator atau nilainya.

Tidak setiap gen selalu dimutasi tetapi mutasi dikontrol dengan probabilitas tertentu yang disebut dengan *mutation probability* dengan notasi p_m dan *mutation rate* dengan notasi r_m, dimana p_m adalah kemungkinan mutasi terjadi pada gen-gen yang menyusun sebuah kromosom. Jika tidak terjadi mutasi maka keturunan yang dihasilkan setelah *crossover* tidak berubah. Jika terjadi mutasi bagian kromosom akan berubah. Jika probabilitas 100%, semua kromosom dimutasi. Jika probabilitasnya 0%, tidak ada. Sedangkan r_m digunakan untuk menentukan jumlah kromosom yang mengalami mutasi (Sugiarto dkk, 2015). Pada penelitian ini, setiap gen akan mempunyai peluang yang sama untuk terjadi mutasi.

2.4 Proyek Konstruksi

Pada penelitian ini digunakan metode optimasi biaya dan waktu crashing. Secara umum pengertian dari proyek adalah suatu

kegiatan pekerjaan yang saling berhubungan secara beran-tai untuk mencapai satu atau beberapa tujuan dengan kendala waktu, biaya dan hasil akhir yang diinginkan. Proyek konstruksi mempunyai kesamaan tipikal dalam beberapa hal yaitu waktu dan usaha untuk menyelesaikan dalam waktu yang telah ditentukan, penggunaan sejumlah sumber daya dan sejumlah investasi yang sangat besar. Namun, seiring dengan perkembangan kemajuan dalam pelaksanaan proyek konstruksi, terdapat beberapa *constraints* yang signifikan. Kedua *constraints* yang dimaksud adalah waktu dan biaya (Mohanty dkk, 2011). Bahkan perihal waktu dalam penyelesaian proyek konstruksi merupakan salah satu area yang sangat krusial (Gerk & Qassim, 2008).

Waktu dan biaya merupakan salah satu asset yang dimiliki oleh proyek konstruksi. Aset lain yang dimiliki oleh proyek konstruksi adalah banyak partisipan yang membentuk proyek konstruksi seperti owner, kontraktor, konsultan, *stakeholders* dan regulator. Oleh karena itu, proyek konstruksi dapat disebut sebagai industri konstruksi yang sangat komplek (Enhassi dkk, 2009).

2.5 *Activity Crashing*

Salah satu upaya yang dilakukan untuk merespon terhadap kondisi proyek yang terlambat adalah *crashing* (Kuhl & Tolentino-Peňa, 2008). C*rashing* adalah upaya untuk mengalokasikan sejumlah uang dalam jumlah yang besar dalam rangka untuk mempercepat jadwal pelaksanaan proyek dari sejumlah aktivitas yang telah dijadwalkan (Islam, 2013).

Secara tidak langsung, *crashing* merupakan cerminan terhadap pertambahan biaya. *Crashing* merupakan salah satu keputusan yang tepat yang dapat dilakukan terhadap proyek yang be-rada dalam kondisi terlambat. *Crashing* ditentukan dengan pertimbangan evaluasi terhadap besarnya pertambahan biaya akibat *crashing* dan besarnya biaya akibat penalti. Dengan kata lain, harus dipertimbangkan keuntungan dari tindakan *crashing* tersebut.

Metode *crashing* merupakan upaya untuk mengurangi waktu pada sejumlah aktivitas yang berada pada jalur kritis. Jalur kritis

merupakan salah satu penyebab keterlambatan. Hal ini terjadi karena aktivitas yang berada pada jalur kritis tidak memiliki *float* atau waktu tenggang. Fokus utama pada *crashing* adalah waktu yang dimiliki pada sejumlah aktivitas yang berada pada jalur kritis dan mengabaikan ketidakpastian pada sejumlah aktivitas yang tidak berada pada jalur kritis.

Kinerja proyek konstruksi merupakan salah satu cara untuk melakukan persetujuan proyek konstruksi kepada *owner* yang dapat diukur dari beberapa faktor. Seperti tepat biaya, tepat mutu, tepat waktu sehingga dapat memberikan kepuasan kepada *owner* dan pihak-pihak lain yang akan menjadi user dari proyek tersebut. Oleh karena itu, kinerja proyek konstruksi menentukan tingkat keberhasilan yang dapat memberikan kepada *owner*, dengan mempertimbangkan faktor-faktor yang signifikan. Dapat ditunjukkan bahwa ada bagian yang beririsan antara *crashing* dan kinerja proyek konstruksi. Pada satu sisi, *crashing* merupakan upaya untuk melakukan percepatan dengan mempertimbangkan waktu dan biaya. Sementara kinerja proyek dapat diukur dari faktor waktu dan biaya. Kedua faktor tersebut merupakan faktor yang dominan di dalam proyek konstruksi khususnya dalam pelaksanaan optimasi proyek konstruksi, seperti *crashing* (Tatar & S. Patil., 2010). Beberapa aktivitas *crashing*:

1. Mengadakan shift pekerjaan
2. Memperpanjang waktu kerja (lembur)
3. Menambah jumlah pekerja
4. Menggunakan alat bantu yang lebih produktif

Pada penelitian ini akan dilakukan salah satu dari aktivitas diatas, yaitu dengan penambahan jumlah pekerja. Sehingga didapatkan *cost slope*, yaitu perbandingan antara selisih biaya dan waktu sebelum dilakukan aktivitas crashing dan sesudahnya, dengan selisih waktu sebagai penyebutnya. Semakin kecil nilai *cost slope* menunjukkan indikasi semakin bagusnya aktivitas crashing tersebut.

BAB III

METODOLOGI PENELITIAN

Metodologi penelitian ini membahas mengenai sumber data penelitian yang digunakan, variabel dan struktur data penelitian, serta langkah analisis penelitian ini.

3.1 Sumber Data

Data yang digunakan pada penelitian ini adalah data sekunder yaitu ringkasan Rencana Anggaran Biaya (RAB) proyek pembangunan sebuah jembatan yang diperoleh dari perusahaan CV. Cremona Teknik Consultant yang bergerak di bidang konsultan proyek bangunan. Jadwal dan Rencana Anggaran Biaya didapat dari perusahaan CV. Cremona Teknik Consultant pada :

Hari : 14 Januari 2020
Pukul : 14.00

Data yang digunakan adalah Jadwal dan Rencana Anggaran Biaya yang pada akhirnya akan diringkas menjadi tiga variabel, yaitu durasi, biaya dan persentase penambahan pekerja. Data terdiri dari 15 aktivitas seperti ditunjukkan pada tabel 3.1.

Tabel 3.1 Aktivitas Pembangunan Proyek Pembangunan Jembatan

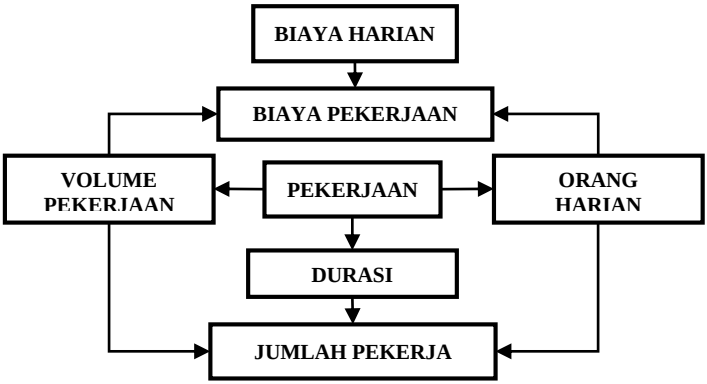
Kode	Kegiatan	Urutan
A	Penyediaan Baja Struktur Grade 345	1
B	Penyediaan Tiang Pancang Baja 508x12 mm	1
C	Penyediaan Tiang Pancang Baja 408x10 mm	1
D	Penyediaan Tiang Pancang Beton Pratekan Pracetak ukuran 200x200 mm	1
E	Penyediaan Struktur Jembatan Rangka Baja Standar 60 m	2
F	Pengangkutan Bahan Jembatan yang disediakan Pengguna Jasa	2
G	Pemancangan Tiang Pancang Beton Pratekan Pracetak 200x200 mm	3

Tabel 3.1 Aktivitas Pembangunan Proyek Pembangunan Jembatan (Lanjutan)

Kode	Kegiatan	Urutan
H	Pembuatan Beton, fc'15 Mpa	4
I	Pembuatan Beton, fc'10 Mpa	4
J	Pembuatan Beton struktur, fc'20 MPa	5
K	Pembuatan Beton struktur, fc'20 MPa yang dilaksanakan di air	5
L	Pembuatan Beton struktur, fc'30 MPa	6
M	Pembuatan Beton struktur, fc'25 Mpa	6
N	Pemasangan Jembatan Rangka Baja Standar Panjang 60 M	7
O	Pemasangan Baja Struktur	8

3.2 Variabel Penelitian dan Struktur

Berdasarkan hasil studi literatur mengenai faktor-faktor yang memiliki peran dalam optimasi, yaitu variabel yang digunakan dalam perhitungan *cost slope*. Gambar 3.1 menunjukkan hubungan antara variabel yang digunakan dalam perhitungan *cost slope*. Aktivitas *crashing* dilakukan dengan menambah jumlah pekerja secara optimum, dimana kombinasi persentase item pekerjaan yang terbaik tersebut didapatkan dari hasil optimasi.



Gambar 3. 1 Hubungan Antara Variabel yang Digunakan Sebagai Perhitungan Optimasi

Untuk lebih jelasnya, variabel yang akan digunakan dalam penelitian ini ditampilkan pada Tabel 3.2.

Tabel 3.2 Variabel Penelitian

Variabel	Nama Variabel	Skala Data
BPH	Biaya per hari untuk setiap jenis pekerja	Rasio
V	Volume item pekerjaan	Rasio
da	Durasi awal setiap item pekerjaan	Rasio
R	Urutan pekerjaan	Ordinal
OH	Nilai OH (Orang Harian) awal setiap item pekerjaan, pada setiap jenis pekerja	Rasio
X	Persentase penambahan pekerja setiap item pekerjaan	Rasio
CS	*Cost slope*	Rasio

Biaya per hari untuk setiap jenis pekerja, volume item pekerjaan, durasi awal setiap item pekerjaan, urutan pekerjaan dan nilai orang harian sudah ada pada data. Penelitian ini akan menghasilkan kombinasi persentase pekerjaan yang optimum, yang dievaluasi menggunakan nilai *cost slope*. Struktur data secara umum yang digunakan dalam penelitian ini terdapat pada Tabel 3.3.

Tabel 3.3 Struktur Data Penelitian

Item Pekerjaan ke-	V	*da*	R	OH	*x*	*CS*
1	V_1	da_1	R_1	OH_1	x_1	CS_1
2	V_2	da_2	R_2	OH_2	x_2	CS_2
3	V_3	da_3	R_3	OH_3	x_3	CS_3
⋮	⋮	⋮	⋮	⋮	⋮	⋮
15	V_{15}	da_{15}	R_{15}	OH_{15}	x_{15}	CS_{15}

Dengan biaya pekerja perhari sudah ditentukan masing-masing untuk setiap jenis pekerja.

3.3 Langkah Analisis

Langkah-langkah yang digunakan dalam penelitian ini adalah sebagai berikut.

1. Melakukan studi literatur dan referensi terkait topik dan metode yang dilakukan dalam penelitian.
2. Menghitung nilai durasi dan biaya pekerjaan. Adapun beberapa tahapan dalam langkah ini yakni sebagai berikut.
   a. Nilai durasi pekerjaan sudah ditentukan terlebih dahulu.
   b. Biaya pekerjaan didapatkan dari perkalian antara volume, orang harian dan biaya pekerja setiap harinya.
3. Menentukan batasan-batasan sebagai *constraint* yaitu pada nilai durasi dan biaya pekerjaan seperti pada Persamaan (2.3) dan Persamaan (2.4)
4. Melakukan aktivitas *crashing* (pada penelitian ini adalah penambahan pekerja) dengan *Constraint Programming* dengan *software* Python. Adapun beberapa tahapan dalam *Constraint Programming* ini yakni sebagai berikut.
   a. Mendefinisikan persentase penambahan pekerja setiap item (x_i) adalah 0% dan 40% supaya memenuhi batasan pada Persamaan (2.3). Sehingga akan dihasilkan berbagai kombinasi persentase penambahan pekerja seperti pada Tabel 3.4.

Tabel 3.4 Ilustrasi Kombinasi Persentasi Penambahan Pekerja

Item 1	Item 2	…	Item 14	Item 15
0	0	…	0	0
0	0	…	0	40
0	0	…	40	0
0	0	…	40	40
…	…	…	…	…
40	40	…	40	40

   b. Dari kombinasi yang didapatkan tersebut masing-masing akan dilakukan perhitungan nilai *cost slope* seperti rumus pada Lampiran 1.

c. Kombinasi persentase yang menghasilkan nilai penyebut dari perhitungan pada tahap 4b kurang dari 30, akan dibuang dan tidak diikutsertakan pada tahap berikutnya, supaya hasil optimasi memenuhi batasan pada Persamaan (2.3) yaitu penambahan durasi harus minimal sebesar 30 hari.
d. Kemudian dari kombinasi yang ada, kombinasi yang menghasilkan nilai *cost slope* paling kecil diambil sebagai solusi yang optimum.

5. Melakukan aktivitas *crashing* dengan pendekatan *Genetic Algorithm* dengan *software* Python. Adapun beberapa tahapan dalam *Genetic Algorithm* ini yakni sebagai berikut.
   a. Membangkitkan populasi awal dengan jumlah kromosom yang akan ditentukan yaitu sebayak 20, dengan x_i sebagai gen dan kombinasi persentase penambahan pekerja sebagai kromosom. Bangkitan setiap populasi dicontohkan pada Tabel 3.5.

Tabel 3.5 Ilustrasi Bangkitan Kromosom pada Populasi

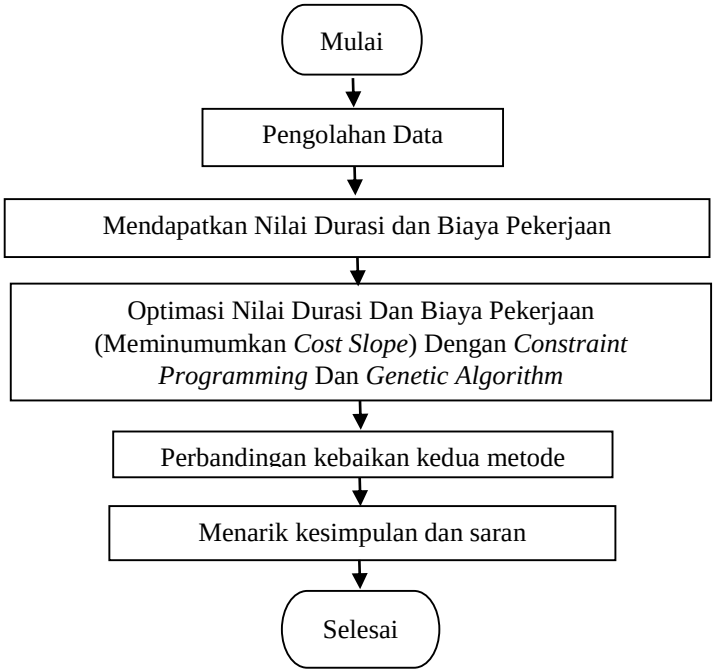
Kromosom	Item 1	Item 2	…	Item 14	Item 15
1	14	5	…	7	21
2	0	11	…	29	9
…	…	…	…	…	…
20	13	15	…	37	6

   Setiap kromosom yang dibangkitkan harus memenuhi batasan seperti pada Persamaan (2.4).
   b. Menghitung nilai *cost slope* dari setiap kromosom sehingga dihasilkan suatu individu, seperti rumus pada Lampiran 1.
   c. Menghitung nilai *fitness* pada setiap individu seperti pada Persamaan (2.12). Akan ditambahkan aturan bahwa individu yang tidak memenuhi batasan pada Persamaan (2.3), yaitu kombinasi persentase yang menghasilkan nilai penyebut dari perhitungan pada tahap 5b kurang dari 30, akan memiliki nilai *fitness* sebesar nol, supaya hasil optimasi memenuhi ba-

tasan yaitu penambahan durasi harus minimal sebesar 30 hari.

d. Dilakukan prosedur seleksi turnamen, dimana seluruh individu pada populasi dipasang-pasangkan, dan *fitness* anggota pasangan akan dibandingkan. Individu dengan *fitness* tertinggi dalam pasangan akan dilanjutkan pada tahapan *crossover*.
e. Dilakukan prosedur *crossover*, dengan teknik yang digunakan adalah *crossover* satu titik, dimana dua orang tua akan menghasilkan dua anak, sehingga jumlah populasi kembali seperti pada tahap 5a.
f. Dilakukan prosedur mutasi, dimana teknik yang digunakan adalah pengkodean nilai yaitu menambah atau mengurangi nilai gen secara acak dengan nilai acak pula, dengan mencoba berbagai nilai p_m dan r_m. Penambahan atau pengurangan nilai pada gen dibatasi sedemikian rupa sehingga nilai gen tetap berada pada 0% dan 40%.
g. Menghitung nilai *cost slope* setiap individu baru, dengan langkah seperti pada tahap 4b.
h. Menghitung nilai *fitness* pada setiap individu baru seperti pada tahap 5c.
i. Dengan populasi yang baru, prosedur diulang dari tahap 5b hingga sejumlah generasi yang ditentukan, yaitu 100 generasi. Individu dengan nilai *fitness* tertinggi akan diambil sebagai solusi yang optimum.

6. Dilakukan perbandingan antara optimasi menggunakan *Constraint Programming* dan *Genetic Algorithm* yang menghasilkan *cost slope* minimum, dan juga mempertimbangkan kecepatan *running* setiap metode. Langkah-langkah di atas dapat digambarkan dengan diagram alir yang disajikan pada Gambar 3.2.



Gambar 3.2 Diagram Alir Penelitian

(*Halaman ini sengaja dikosongkan*)

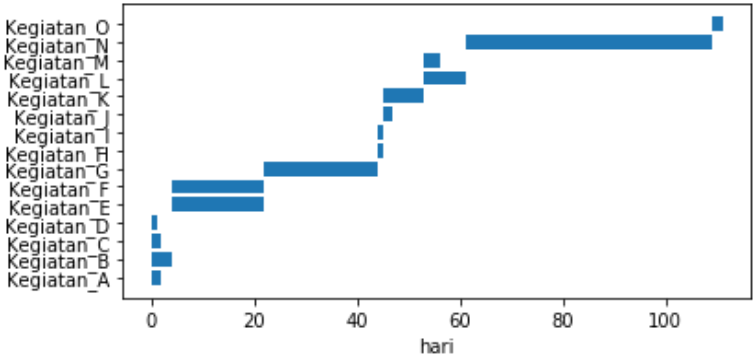
BAB IV

ANALISIS DAN PEMBAHASAN

Pada bab ini akan dibahas tentang upaya untuk menambah jumlah pekerja dengan tujuan mempercepat waktu proyek dengan mempertimbangkan waktu serta biaya, menggunakan pendekatan eksak *Constraint Programming* dan Algoritma Genetika. Kemudian akan dilakukan perbandingan dari kedua metode tersebut menggunakan nilai *cost slope*.

4.1 Rancangan Awal Jadwal Pelaksanaan Pembangunan Jembatan

Supaya perencanaan, pelaksanaan, dan pengawasan pekerjaan konstruksi betul-betul dapat dikerjakan dengan efektif dan efisien, maka diperlukan untuk merencanakan berbagai hal dengan matang. Beberapa hal tersebut adalah setiap jenis pekerjaan, urutan pelaksanaan pekerjaan, menghitung waktu pelaksanaan setiap jenis pekerjaan, menghitung kebutuhan alat, bahan, tenaga dan waktu pengadaan. Ilustrasi rancangan awal jadwal pelaksanaan ditunjukkan pada Gambar 4.1.



Gambar 4.1 Ilustrasi Rancangan Awal Jadwal Pelaksanaan Pembangunan Jembatan

Pada Gambar 4.1 ditunjukkan bahwa ada beberapa pekerjaan dapat dilakukan bersamaan, dan ada yang mengharuskan untuk beberapa pekerjaan sebelumnya, selesai. Namun semua sudah berbentuk kelompok urutan pekerjaan, sehingga optimasi yang dilakukan, khususnya untuk perhitungan percepatan durasi sebagai perbandingan untuk menghitung nilai *cost slope*, akan berdasarkan jumlah dari maksimal durasi pada masing-masing kelompok. Kemudian ditunjukkan data Biaya Per Hari (BPH) yang dikeluarkan untuk setiap jenis pekerja pada Tabel 4.1.

Tabel 4.1 BPH Pembangunan Jembatan Setiap Jenis Pekerja

Jenis Pekerja	BPH
Pekerja	Rp121.682,00
Tukang	Rp189.953,00
Mandor	Rp206.075,00

Tabel 4.1 menunjukkan bahwa upah tertinggi berturut-turut adalah mandor, tukang dan pekerja. Hal tersebut juga menjadi perhatian dalam meminimumkan nilai *cost slope* dengan menambah persentase pekerja pada setiap item pekerjaan. Selain itu, perlu dipertimbangkan juga koefisien nilai Orang Harian (OH), yaitu upah pekerja standar per hari kerja. Koefisien OH pada pembangunan jembatan yang sudah terlampir pada data RAB ditunjukkan pada tabel 4.2.

Tabel 4.2 menunjukkan bahwa koefisien OH untuk mandor selalu lebih sedikit daripada koefisien OH untuk pekerja dan tukang. Diketahui pula koefisien OH untuk pekerja lebih sedikit daripada koefisien OH untuk tukang pada kegiatan ke-8 hingga kegiatan ke-13, yaitu tahap pembuatan beton, yang menunjukkan bahwa pekerjaan ini membutuhkan banyak tukang, yaitu pekerja dengan keahlian tertentu, daripada pekerja tanpa keahlian tertentu. Sedangkan untuk kegiatan selain itu, komposisi pekerja tanpa keahlian khusus lebih akan lebih banyak daripada komposisi tukang.

Tabel 4.2 Koefisien OH Pembangunan Jembatan

Item Pekerjaan	Pekerja	Tukang	Mandor
Penyediaan Baja Struktur Grade 345	0.008367	0.001673	0.000837
Penyediaan Tiang Pancang Baja 508x12 mm	0.234772	0.078257	0.019564
Penyediaan Tiang Pancang Baja 408x10 mm	0.201234	0.067078	0.016769
Penyediaan Tiang Pancang Beton Pratekan Pracetak ukuran 200x200 mm	0.207603	0.069201	0.0173
Penyediaan Struktur Jembatan Rangka Baja Standar 60 m	0.008367	0.001673	0.000837
Pengangkutan Bahan Jembatan yang disediakan Pengguna Jasa	0.009036	0.001506	0.000753
Pemancangan Tiang Pancang Beton Pratekan Pracetak 200x200 mm	3.162651	3.162651	1.581325
Pembuatan Beton, fc’15 Mpa	0.590361	0.885542	0.098394
Pembuatan Beton, fc’10 Mpa	0.393574	0.787149	0.098394

Tabel 4.2 Koefisien OH Pembangunan Jembatan (Lanjutan)

Item Pekerjaan	Pekerja	Tukang	Mandor
Pembuatan Beton struktur, fc'20 MPa	0.590361	0.885542	0.098394
Pembuatan Beton struktur, fc'20 MPa yang dilaksanakan di air	1.718876	2.578313	0.214859
Pembuatan Beton struktur, fc'30 MPa	1.180723	2.951807	0.196787
Pembuatan Beton struktur, fc'25 Mpa	0.787149	1.180723	0.098394
Pemasangan Jembatan Rangka Baja Standar Panjang 60 M	0.043507	0.013052	0.002175
Pemasangan Baja Struktur	0.02008	0.006024	0.001004

Kemudian durasi setiap item pekerjaan, urutan dan volume pekerjaan juga akan ditunjukkan pada Tabel 4.3. Dari nilai durasi dan urutan pekerjaan pada Tabel 4.3 dapat dihitung durasi awal yang direncanakan pada RAB, dengan cara perhitungan berkelompok.

Pada urutan ke-1, durasi kegiatan terlama adalah pada kegiatan B yaitu 4 hari, yang menunjukkan bahwa urutan ke-1 dilakukan selama 4 hari. Pada urutan ke-2, kegiatan E dan F memiliki durasi yang sama, maka kegiatan pada urutan ke-2 dilakukan selama 18 hari. Durasi proyek awal (*da*) adalah hasil dari penjumlahan durasi setiap urutan kegiatan, yaitu 111 hari.

Tabel 4.3 Volume Pekerjaan dan Durasi Setiap Item Pekerjaan

Item Pekerjaan	Durasi (Hari)	Urutan	Volume	Satuan
Penyediaan Baja Struktur Grade 345	2	1	8366.4	kg
Penyediaan Tiang Pancang Baja 508x12 mm	4	1	357.7933001	m
Penyediaan Tiang Pancang Baja 408x10 mm	2	1	417.4255167	m
Penyediaan Tiang Pancang Beton Pratekan Pracetak ukuran 200x200 mm	1	1	416.7571606	m
Penyediaan Struktur Jembatan Rangka Baja Standar 60 m	18	2	8366.4	kg
Pengangkutan Bahan Jembatan yang disediakan Pengguna Jasa	18	2	9296	kg
Pemancangan Tiang Pancang Beton Pratekan Pracetak 200x200 mm	22	3	17.70666667	m
Pembuatan Beton, fc’15 Mpa	1	4	71.14285714	m^3

Tabel 4.3 Volume Pekerjaan dan Durasi Setiap Item Pekerjaan (Lanjutan)

Item Pekerjaan	Durasi (Hari)	Urutan	Volume	Satuan
Pembuatan Beton, fc'10 Mpa	1	4	71.14285714	m^3
Pembuatan Beton struktur, fc'20 MPa	2	5	71.14285714	m^3
Pembuatan Beton struktur, fc'20 MPa yang dilaksanakan di air	8	5	32.57943925	m^3
Pembuatan Beton struktur, fc'30 MPa	8	6	71.14285714	m^3
Pembuatan Beton struktur, fc'25 Mpa	3	6	71.14285714	m^3
Pemasangan Jembatan Rangka Baja Standar Panjang 60 M	48	7	3217.846154	kg
Pemasangan Baja Struktur	2	8	6972	kg

Akan diberikan contoh perhitungan jumlah pekerja awal untuk kegiatan A dan jenis pekerja "Pekerja".

$$JP_{A,Pekerja} = \left\lceil \frac{V_A \times OH_{A,Pekerja}}{da_A} \right\rceil = \left\lceil \frac{8366.4 \times 0.008367}{2} \right\rceil$$

$$= \lceil 35.0000019432 \rceil = 36$$

Maka jumlah pekerja awal untuk kegiatan A dan jenis pekerja "Pekerja" adalah sebanyak 36 orang. Rincian jumlah pekerja awal pada setiap kegiatan dan jenis pekerja sebelum dilakukan optimasi ditunjukkan pada Tabel 4.4.

Pada optimasi yang akan dilakukan, akan ditentukan persentase penambahan jumlah pekerja pada setiap item pekerjaan. Hal tersebut akan mengakibatkan berubahnya tingkat produktivitas, dan lebih lanjut menghasilkan nilai durasi setiap item pekerjaan dan koefisien OH baru. Dari hal tersebut, percepatan dan kerugian total proyek akibat penambahan akan dapat dihitung, sehingga nilai *cost slope* juga dapat dihitung. Program *Python* untuk melakukan hal tersebut terlampir pada Lampiran 2.

Tabel 4.4 Jumlah Pekerja Sebelum Dilakukan Optimasi

Item Pekerjaan	Pekerja	Tukang	Mandor
Penyediaan Baja Struktur Grade 345	36	7	4
Penyediaan Tiang Pancang Baja 508x12 mm	21	7	2
Penyediaan Tiang Pancang Baja 408x10 mm	42	14	4
Penyediaan Tiang Pancang Beton Pratekan Pracetak ukuran 200x200 mm	87	29	8
Penyediaan Struktur Jembatan Rangka Baja Standar 60 m	4	1	1
Pengangkutan Bahan Jembatan yang disediakan Pengguna Jasa	5	1	1

Tabel 4.4 Jumlah Pekerja Sebelum Dilakukan Optimasi (Lanjutan)

Item Pekerjaan	Pekerja	Tukang	Mandor
Pemancangan Tiang Pancang Beton Pratekan Pracetak 200x200 mm	3	3	2
Pembuatan Beton, fc'15 Mpa	43	64	7
Pembuatan Beton, fc'10 Mpa	28	56	7
Pembuatan Beton struktur, fc'20 MPa	22	32	4
Pembuatan Beton struktur, fc'20 MPa yang dilaksanakan di air	7	11	1
Pembuatan Beton struktur, fc'30 MPa	11	27	2
Pembuatan Beton struktur, fc'25 Mpa	19	29	3
Pemasangan Jembatan Rangka Baja Standar Panjang 60 M	3	1	1
Pemasangan Baja Struktur	70	21	4

4.2 Optimasi dengan *Constraint Programming*

Penyelesaian kasus ini dengan *Constraint Programming* menghasilkan kombinasi terbaik untuk persentase penambahan kerja pada setiap item pekerjaan. Kombinasi terbaik untuk persentase penambahan kerja dengan metode ini ditunjukkan pada Tabel 4.5. Program *Python* untuk menghasilkan persentase penambahan

pekerja yang optimal dengan *Constraint Programming*, terlampir pada Lampiran 3.

Tabel 4.5 Kombinasi Terbaik dengan *Constraint Programming*

Item Pekerjaan	Persentase Penambahan Pekerja (%)
Penyediaan Tiang Pancang Baja 508x12 mm	40
Pemancangan Tiang Pancang Beton Pratekan Pracetak 200x200 mm	40
Pemasangan Jembatan Rangka Baja Standar Panjang 60 M	40

Pada penelitian ini, optimasi menggunakan metode ini menunjukkan hasil bahwa penambahan sebesar 40% hanya dilakukan pada beberapa item pekerjaan, yaitu pada kegiatan penyediaan tiang pancang baja dengan diameter 508 mm dan tebal 12 mm, pemancangan tiang pancang beton pratekan pracetak ukuran 200 mm x 200 mm, dan pemasangan jembatan rangka baja standar dengan panjang 60 meter. Rincian jumlah pekerja dari hasil optimasi *Constraint Programming* ditunjukkan pada Tabel 4.6.

Tabel 4.6 Jumlah Pekerja Setelah Dilakukan Optimasi menggunakan *Constraint Programming*

Item Pekerjaan	Pekerja	Tukang	Mandor
Penyediaan Baja Struktur Grade 345	36	7	4
Penyediaan Tiang Pancang Baja 508x12 mm	30	10	3
Penyediaan Tiang Pancang Baja 408x10 mm	42	14	4

Tabel 4.6 Jumlah Pekerja Setelah Dilakukan Optimasi menggunakan *Constraint Programming* (Lanjutan)

Item Pekerjaan	Pekerja	Tukang	Mandor
Penyediaan Tiang Pancang Beton Pratekan Pracetak ukuran 200x200 mm	87	29	8
Penyediaan Struktur Jembatan Rangka Baja Standar 60 m	4	1	1
Pengangkutan Bahan Jembatan yang disediakan Pengguna Jasa	5	1	1
Pemancangan Tiang Pancang Beton Pratekan Pracetak 200x200 mm	5	5	3
Pembuatan Beton, fc'15 Mpa	43	64	7
Pembuatan Beton, fc'10 Mpa	28	56	7
Pembuatan Beton struktur, fc'20 MPa	22	32	4
Pembuatan Beton struktur, fc'20 MPa yang dilaksanakan di air	7	11	1
Pembuatan Beton struktur, fc'30 MPa	11	27	2
Pembuatan Beton struktur, fc'25 Mpa	19	29	3
Pemasangan Jembatan Rangka Baja Standar Panjang 60 M	5	2	2
Pemasangan Baja Struktur	70	21	4

Selanjutnya, total percepatan dari hasil optimasi adalah sebesar 30 hari, yang menunjukkan bahwa optimasi ini sudah memenuhi batasan pada Persamaan (2.4). Kerugian total yang timbul akibat penambahan pekerja ini adalah sebesar Rp42.678.182,00. Selain itu, nilai *cost slope* dari metode optimasi ini adalah sebesar Rp1.422.606,07. Artinya, kerugian per hari percepatan pada pembangunan jembatan dengan metode *Constraint Programming* adalah sebesar Rp1.422.606,07. Selain itu, metode *Constraint Programming* membutuhkan waktu selama 427,45 detik untuk mendapatkan kombinasi terbaik persentase penambahan kerja pada setiap item pekerjaan.

4.3 Optimasi dengan *Genetic Algorithm*

Genetic Algorithm dapat digunakan untuk melakukan pendekatan dalam mendapatkan kombinasi terbaik persentase penambahan kerja pada setiap item pekerjaan. Jumlah kromosom pada setiap populasi yang diterapkan pada *Genetic Algorithm* di penelitian ini adalah sebanyak 20 kromosom. Struktur populasi akan berbentuk seperti ilustrasi pada Tabel 4.7 yang menunjukkan struktur pada generasi pertama.

Tabel 4.7 Ilustrasi Struktur Populasi

Kromosom	**Item 1**	**Item 2**	**…**	**Item 14**	**Item 15**	***Fitness* (x10^{-7})**
1	14	5	…	7	21	3,77
2	0	11	…	29	9	3,53
…	…	…	…	…	…	
20	13	15	…	37	6	3,52

Kemudian dari adanya proses seleksi, *crossover* dan mutasi pada setiap generasi, akan dilakukan upaya untuk mendapatkan nilai *fitness* setinggi-tingginya. *Mutation probability* (p_m) dan *mutation rate* (r_m) bernilai antara nol dan satu. Untuk hasil terbaik, akan dilakukan beberapa skenario p_m dan r_m yang berbeda, dan pada penelitian ini hasil terbaik didapat dengan p_m dan r_m bernilai

0,3. Jumlah generasi dibatasi yaitu sebanyak 100. Kombinasi terbaik untuk persentase penambahan pekerja dengan metode *Genetic Algorithm* ditunjukkan pada Tabel 4.8. Adapun program *Python* untuk menghasilkan persentase penambahan pekerja yang optimal dengan *Genetic Algorithm*, terlampir pada Lampiran 4.

Optimasi menggunakan *Genetic Algorithm* menunjukkan hasil bahwa penambahan pekerja hanya dilakukan pada kegiatan penyediaan struktur jembatan rangka baja standar 60 meter, pengangkutan bahan jembatan yang disediakan pengguna jasa, pemancangan tiang pancang beton pratekan pracetak ukuran 200 mm x 200 mm, dan pemasangan jembatan rangka baja standar dengan panjang 60 m.

Tabel 4.8 Kombinasi Terbaik dengan *Genetic Algorithm*

Item Pekerjaan	Persentase Penambahan Pekerja (%)
Penyediaan Struktur Jembatan Rangka Baja Standar 60 m	9
Pengangkutan Bahan Jembatan yang disediakan Pengguna Jasa	40
Pemancangan Tiang Pancang Beton Pratekan Pracetak 200x200 mm	4
Pemasangan Jembatan Rangka Baja Standar Panjang 60 M	2

Kombinasi pada Tabel 4.8 merupakan kombinasi terbaik dari berbagai kemungkinan kombinasi yang dihasilkan dari bangkitan populasi pada proses *Genetic Algorithm*. Pada Lampiran 5, ditunjukkan nilai *fitness* terbaik pada setiap generasi, pada 10 kali percobaan. Dihasilkan bahwa *fitness* terbaik dari generasi ke generasi, hampir selalu meningkat hingga sekitar generasi ke-30 dan ke-40, kemudian tetap memiliki nilai yang sama hingga generasi ke-

100. Artinya, nilai *fitness* sudah konvergen pada kesepuluh percobaan. Selain itu, dari kesepuluh percobaan tersebut, hanya ada dua nilai *cost slope* yang berbeda, yaitu Rp1.274.097,96 sebanyak 8 kali dan Rp1.334.840,37 sebanyak 2 kali, yaitu pada percobaan ke-6 dan ke-8. Adapun rincian jumlah pekerja dari hasil optimasi menggunakan *Genetic Algorithm* ditunjukkan pada Tabel 4.9.

Seperti pada metode *Constraint Programming*, total percepatan dari hasil optimasi adalah sebesar 30 hari, yang menunjukkan bahwa optimasi ini sudah memenuhi batasan pada Persamaan (2.4). Kerugian total yang timbul akibat penambahan pekerja ini adalah sebesar Rp38.222.939,00. Selain itu, nilai *cost slope* dari metode optimasi ini adalah sebesar Rp1.274.097,96, diambil dari nilai terbaik dari kesepuluh percobaan. Artinya, kerugian per hari percepatan pada pembangunan jembatan dengan metode *Genetic Algorithm* adalah sebesar Rp1.274.097,96.

Tabel 4.9 Jumlah Pekerja Setelah Dilakukan Optimasi menggunakan *Genetic Algorithm*

Item Pekerjaan	Pekerja	Tukang	Mandor
Penyediaan Baja Struktur Grade 345	36	7	4
Penyediaan Tiang Pancang Baja 508x12 mm	21	7	2
Penyediaan Tiang Pancang Baja 408x10 mm	42	14	4
Penyediaan Tiang Pancang Beton Pratekan Pracetak ukuran 200x200 mm	87	29	8
Penyediaan Struktur Jembatan Rangka Baja Standar 60 m	5	2	2

Tabel 4.9 Jumlah Pekerja Setelah Dilakukan Optimasi menggunakan *Genetic Algorithm* (Lanjutan)

Item Pekerjaan	Pekerja	Tukang	Mandor
Pengangkutan Bahan Jembatan yang disediakan Pengguna Jasa	7	2	2
Pemancangan Tiang Pancang Beton Pratekan Pracetak 200x200 mm	4	4	3
Pembuatan Beton, fc'15 Mpa	43	64	7
Pembuatan Beton, fc'10 Mpa	28	56	7
Pembuatan Beton struktur, fc'20 MPa	22	32	4
Pembuatan Beton struktur, fc'20 MPa yang dilaksanakan di air	7	11	1
Pembuatan Beton struktur, fc'30 MPa	11	27	2
Pembuatan Beton struktur, fc'25 Mpa	19	29	3
Pemasangan Jembatan Rangka Baja Standar Panjang 60 M	4	2	2
Pemasangan Baja Struktur	70	21	4

Selain itu, dari kesepuluh percobaan yang dilakukan, metode *Genetic Algorithm* rata-rata membutuhkan waktu selama 229,15 detik untuk mendapatkan kombinasi terbaik persentase penambahan kerja pada setiap item pekerjaan, dimana catatan lama waktu pada masing-masing percobaan telah dilampirkan pada Lampiran

6. Akan dilakukan perbandingan dari kedua metode yang digunakan pada penelitian ini, ditinjau dari nilai *cost slope* dan lama waktu untuk mendapatkan kombinasi persentase terbaik.

Tabel 4.10 Perbandingan Kedua Metode

Metode	***Cost Slope***	**Lama Waktu Mendapatkan Solusi (detik)**
Constraint Programming	Rp1.422.606,07	427,45
Genetic Algorithm	Rp1.274.097,96	229,15

Tabel 4.10 menunjukkan bahwa nilai *cost slope* yang dihasilkan oleh metode *Genetic Algorithm* lebih kecil daripada nilai yang dihasilkan oleh metode *Constraint Programming*. Selain itu, *Genetic Algorithm* membutuhkan waktu lebih singkat untuk mendapatkan kombinasi terbaik persentase pekerjaan setiap item pekerjaan. Maka dalam kasus ini, metode *Genetic Algorithm* lebih baik daripada *Constraint Programming*, ditinjau dari nilai *cost slope* dan lama waktu untuk mendapatkan kombinasi persentase terbaik.

(*Halaman ini sengaja dikosongkan*)

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan analisis yang telah dilakukan, diperoleh beberapa kesimpulan sebagai berikut.

1. Optimasi menggunakan *Constraint Programming* menunjukkan hasil bahwa persentase penambahan jumlah pekerja sebesar 40% dilakukan pada kegiatan penyediaan tiang pancang baja dengan diameter 508 mm dan tebal 12 mm, pemancangan tiang pancang beton pratekan pracetak ukuran 200 mm x 200 mm, dan pemasangan jembatan rangka baja standar dengan panjang 60 meter. Sedangkan Optimasi menggunakan *Genetic Algorithm* menunjukkan hasil bahwa persentase penambahan jumlah pekerja dilakukan pada kegiatan penyediaan struktur jembatan rangka baja standar 60 meter yaitu sebesar 9%, pengangkutan bahan jembatan yang disediakan pengguna jasa sebesar 40%, pemancangan tiang pancang beton pratekan pracetak ukuran 200 mm x 200 mm sebesar 4%, dan pemasangan jembatan rangka baja standar dengan panjang 60 m sebesar 2%.
2. Nilai *cost slope* dengan metode *Constraint Programming* adalah sebesar Rp1.422.606,07. Metode *Constraint Programming* membutuhkan waktu selama 427,45 detik untuk mendapatkan kombinasi terbaik persentase penambahan kerja pada setiap item pekerjaan. Sedangkan nilai *cost slope* dengan metode *Genetic Algorithm* adalah sebesar Rp1.274.097,96. Selain itu, metode *Genetic Algorithm* rata-rata membutuhkan waktu selama 229,15 detik untuk mendapatkan kombinasi terbaik persentase penambahan kerja pada setiap item pekerjaan. Maka dalam kasus ini, metode *Genetic Algorithm* lebih baik daripada *Constraint Programming,* ditinjau dari nilai *cost slope* dan lama waktu untuk mendapatkan kombinasi persentase terbaik.

5.2 Saran

Berdasarkan pertimbangan hasil penelitian ini, maka terdapat beberapa saran yang direkomendasikan sebagai berikut:

1. Penelitian ini dapat dikembangkan tidak hanya optimasi yang dilakukan terhadap jumlah pekerja. Jumlah *shift* maupun sumberdaya lainnya dapat dijadikan sebagai variabel yang akan meminimumkan nilai *cost slope*.
2. Pada penelitian ini urutan pekerjaan berbentuk kelompok, sehingga program yang dibuat dalam penelitian ini hanya cocok untuk kasus serupa. Pada penelitian selanjutnya, dapat dikembangkan supaya lebih fleksibel untuk optimasi biaya dan waktu pada sebuah proyek.
3. Pada penelitian selanjutnya dapat dikembangkan untuk metode yang mudah digunakan untuk melakukan optimasi terkait kasus serupa, menemukan solusi yang memuaskan dan efisien dari segi kecepatan. Beberapa metode matematis sederhana maupun pendekatan dengan *Simulated Annealing* dapat menjadi pertimbangan.

DAFTAR PUSTAKA

Assaf, A., S., Hazzmi, M. A., & Khalil, M. A. (1995). Causes of Delay in Large Building Construction Projects. *Article in Journal of Management in Engineering 11(2)*, 45-50.

Enhassi, A., Mohamed, S., & Abushaban, S. (2009). Factors Affecting the Performance of Construction Projects In the Gaza Strip. *Journal of Civil Engineering And Management 15 (3)*, 269 - 280.

Feng, C.-W., Liu, L., & Burns, S. A. (1997). USING GENETIC ALGORITHMS TO SOLVE CONSTRUCTION TIME-COST TRADE-OFF PROBLEMS. *J. Comput. Civ. Eng., 11(3)*, 184-189.

Firdaus. (2017). Implementasi Penjadwalan Kuliah Job Shop Dengan Perancangan Jadwal Kuliah Menggunakan Constraints Programming. *SinkrOn Jurnal & Penelitian Teknik Informatika, Volume1 Nomor 2*.

Gerk, J., & Qassim, R. (2008). Project acceleration via activity crashing, overlapping, and substitution. *IEEE Transactions on Engineering Management 55 (4)*, 590-601.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* New York: Addison-Wesley Publishing.

Griva, I., G. Nash, S., & Sofer, A. (2009). Chapter 1 Optimization Models. Dalam I. Griva, S. G. Nash, & A. Sofer, *Linear and Nonlinear Optimization (Second Edition)* (hal. 3). Fairfax, Virginia: The Society for Industrial and Applied Mathematics.

Hasan, U., Hermanto, T., & Muttaqin, M. R. (2018). PENJADWALAN MATA KULIAH MENGGUNAKAN ALGORITMA GENETIKA DI STT WASTUKANCANA PURWAKARTA. *Jurnal Ilmiah Komputer dan Informatika (KOMPUTA)*.

Hooshyar, B., Rahmani, A., & Shenasa, M. (2008). A Genetic Algorithm to Time- Cost Trade off in Project Scheduling. *IEEE Congress on Evolutionary Computation*.

Islam, N. (2013). Complex Project Crashing Algorithm. *IOSR Journal of Business and Management (IOSR-JBM)*.

Kristiadi, D., & Hartanto, R. (2019). Genetic Algorithm for Lecturing Schedule Optimization (Case Study: University of Boyolali). *Indonesian Journal of Computing and Cybernetics Systems*.

Kuhl, M., & Tolentino-Peña, R. (2008). A Dynamic Crashing Method For Project Management Using Simulation-Based Optimization. *Proceedings of the 2008 Winter Simulation Conference*.

KULCSAR, R. M., BOROZAN, I. S., ARGESANU, V., & MADARAS, L. (2013). Car Seats Ergonomic Evaluation. *Tome XI. Fascicule (ISSN 1584-2665)*.

Liu, Q., Yang, X., Zhang, L., & Majetich, G. (2010). Optimization of Ultrasonic-Assisted Extraction Of Chlorogenic Acid From Folium Eucommiae and Evaluation of Its Antioxidant Activity. *Journal of Medicinal Plants Research*.

Mohanty, A., Mishra, J., & Satpathy, B. (2011). Activity modes selection for project crashing through deterministic simulation. *Journal of Industrial Engineering and Management. 4 (4)*, 610-623.

Mutahkiroh, I., Saptono, F., Hasanah, N., & Wiryadinata, R. (2007). Pemanfaatan Metode Heuristic dalam Pencarian Jalur Terpendek dengan Algoritma Semut dan Algoritma Genetika. *Seminar Nasional Aplikasi Teknologi Informasi 2007*.

Prayogo, D., Gosno, R. A., Evander, R., & Limanto, S. (2017). Implementasi Metode Metaheuristic Symbiotic Organisms Search dalam Penentuan Tata Letak Fasilitas Proyek Konstruksi Berdasarkan Jarak Tempuh Pekerja. *Jurnal Teknik Industri, Vol. 19, No. 2*, 103-114.

Proboyo, B. (1999). Keterlambatan Waktu Pelaksanaan Proyek : Klasifikasi Dan Peringkat Dari Penyebab-Penyebabnya. *Dimensi Teknik Sipil, 1 (2).*

Sam'ani. (2012). Rancang Bangun Sistem Penjadwalan Perkuliahan dan Ujian Akhir Semester dengan Pendekatan Algoritma Genetika. *Tesis Pascasarjana, Universitas Diponegoro*.

Saputro, N., & Yento. (2020, January 24). *PEMAKAIAN ALGORITMA GENETIK UNTUK PENJADWALAN JOB SHOP DINAMIS.* Diambil kembali dari Jurusan Teknik Industri, Fakultas Teknologi Industri, Universitas Kristen Petra: http://puslit.petra.ac.id/journals/industrial

Sivanadam, S., Sumathi, S., & Deepa, S. (2007). *Sivanadam, S.N., Sumathi, S. & Deepa, S.N.* Heidelberg.

Sugiarto, E., Winarno, S., & Fahmi, A. (2015). PENJADWALAN PERKULIAHAN OTOMATIS BERBASIS FUZZY LOGIC DAN GENETIC ALGORITHM PADA UNIVERSITAS DIAN NUSWANTORO. *Techno.COM, Vol. 14, No. 4*, 315-328.

Tatar, B. V., & S. Patil., R. (2010). Time and Cost Optimization Techniques in Construction Project Management. *International Journal of Latest Trends in Engineering and Technology 7 (3)*, 230-237.

LAMPIRAN

Lampiran 1. Perhitungan Nilai *Cost Slope*

$$CS = \frac{\sum_{i=1}^{N}\sum_{j=1}^{n} V_i \times OH_{ij}\left(\frac{\left\lceil\left\lceil\frac{V_i \times OH_{ij}}{da_i}\right\rceil \times (1+x_i)\right\rceil}{\left\lceil\frac{V_i \times OH_{ij}}{da_i}\right\rceil} - 1\right) \times BPH_j}{\sum_{i=1}^{N}\max\left[da_{i1}, da_{i2}, \dots, da_{i(n_i)}\right] - \sum_{i=1}^{N}\max\left[\left(\frac{da_i \times \sum_{j=1}^{n}\left\lceil\frac{V_i \times OH_{ij}}{da_i}\right\rceil}{\sum_{j=1}^{n}\left\lceil\left\lceil\frac{V_i \times OH_{ij}}{da_i}\right\rceil \times (1+x_i)\right\rceil}\right)_1, \left(\frac{da_i \times \sum_{j=1}^{n}\left\lceil\frac{V_i \times OH_{ij}}{da_i}\right\rceil}{\sum_{j=1}^{n}\left\lceil\left\lceil\frac{V_i \times OH_{ij}}{da_i}\right\rceil \times (1+x_i)\right\rceil}\right)_2, \dots, \left(\frac{da_i \times \sum_{j=1}^{n}\left\lceil\frac{V_i \times OH_{ij}}{da_i}\right\rceil}{\sum_{j=1}^{n}\left\lceil\left\lceil\frac{V_i \times OH_{ij}}{da_i}\right\rceil \times (1+x_i)\right\rceil}\right)_{(n_i)}\right]},$$

dimana :

BPH_j	:	Biaya per hari setiap jenis pekerja
CS	:	*Cost slope*
da_i	:	Durasi awal setiap item pekerjaan
n	:	Jumlah jenis pekerja
N	:	Jumlah item pekerjaan
OH_{ij}	:	Koefisien orang harian setiap item pekerjaan dan jenis pekerja
V_i	:	Volume setiap item pekerjaan setiap item pekerjaan
x_i	:	Persentase penambahan pekerja setiap item pekerjaan

Lampiran 2. Program *Python* untuk Menyusun Struktur Data

```
#import libraries
import numpy as np
import pandas as pd
import math
import random
import matplotlib.pyplot as plt
import statistics
import itertools
import time

#Import Data
data=pd.read_csv(r'C:/Users/Nafis/Desktop/TA_KERJA/TA_NAFIS.
csv', sep=';')

#Definisi BPH, Volume, Durasi Awal, Orang Harian dan Biaya
Awal
bph = data.BPH.T.dropna(axis=0)
vol = data.VOL.dropna(axis=0)
da = data.DURASI_AWAL.dropna(axis=0)
da_total = sum(da)
urutan = data.Urutan.dropna(axis=0)
jumlah_urutan = max(urutan)
oh = data.drop(data.columns[0:5], axis=1).dropna(axis=0)
oh = np.array(oh)
jumlah_item = pd.DataFrame(da).shape[0]
jenis_pekerja = pd.DataFrame(bph).shape[0]
a = pd.concat([da,urutan], axis=1)
b = a.groupby(urutan)
c = np.zeros(jumlah_urutan+1)
for i in range(1,jumlah_urutan+1):
    c[i] = c[i-1] + max(b.get_group(i-1).DURASI_AWAL)
total_durasi_awal = c
start = np.zeros(len(data))
finish = np.zeros(len(data))
for i in range(len(data)):
    start[i] = total_durasi_awal[urutan[i]]
```

Lampiran 2. Program *Python* untuk Menyusun Struktur Data (Lanjutan)

```
    finish[i] = da[i] + total_durasi_awal[urutan[i]]
start = pd.DataFrame(start)
finish = pd.DataFrame(finish)
start.rename(columns={0:'start'}, inplace=True)
finish.rename(columns={0:'finish'}, inplace=True)
data_jadwal = pd.concat([data["kegiatan"],start,finish,urutan,da],
axis=1)

#Membuat Ilustrasi Rancangan Awal Jadwal Pelaksanaan
Pembangunan Jembatan
fig,ax=plt.subplots(figsize=(6,3))
labels=[]
for i, kegiatan in enumerate(data_jadwal.groupby("kegiatan")):
    labels.append(kegiatan[0])
    for r in kegiatan[1].groupby("Urutan"):
        data = r[1][["start", "DURASI_AWAL"]]
        ax.broken_barh(data.values, (i-0.4,0.8) )
ax.set_yticks(range(len(labels)))
ax.set_yticklabels(labels)
ax.set_xlabel("hari")
plt.tight_layout()
plt.show()
```

Lampiran 2. Program *Python* untuk Menyusun Struktur Data (Lanjutan)

```
#Lanjutan Definisi Data
pekerja_awal=np.zeros((jumlah_item,jenis_pekerja))
pekerja_awal_peritem=np.zeros(jumlah_item)
ba=np.zeros((jumlah_item,jenis_pekerja))
biaya_awal=np.zeros(jumlah_item)
for i in range(jumlah_item):
    for j in range(jenis_pekerja):
        # Change the required number of zeros to ones
        pekerja_awal[i,j] = math.ceil(oh[i,j]*vol[i]/da[i])
        pekerja_awal_peritem[i] = sum(pekerja_awal[i,:])
        ba[i,j] = bph[j]*vol[i]*oh[i,j]
        biaya_awal[i] = sum(ba[i,:])
a = pd.concat([da,urutan], axis=1)
b = a.groupby(urutan)
c = 0
for i in range(jumlah_urutan+1):
    c = c + max(b.get_group(i).DURASI_AWAL)
total_durasi_awal = c
```

Lampiran 3. Program *Python* untuk *Constraint Programming*

```
start_time = time.time()

#-------------------------------------------------------

kombinasi = 1
stuff = np.zeros(kombinasi+1)

for i in range(kombinasi+1):
    stuff[i] = (.4/kombinasi*i)

stuffs = np.zeros((jumlah_item,kombinasi+1))

for i in range(jumlah_item):
    stuffs[i] = stuff

stuffed = np.zeros((0,kombinasi+1))

for element in itertools.product(*stuffs):
    stuffed = np.append(stuffed, element)

stuffed = np.split(stuffed,len(stuffed)/jumlah_item)
stuffed = np.array(stuffed)

#-------------------------------------------------------

# Set up an initial array of all zeros
population = stuffed
individuals = len(stuffed)
penambahan_pekerja =
np.zeros((individuals,jumlah_item,jenis_pekerja))
penambahan_pekerja_peritem = np.zeros((individuals,jumlah_item))
pekerja_akhir = np.zeros((individuals,jumlah_item,jenis_pekerja))
ohb = np.zeros((individuals,jumlah_item,jenis_pekerja))
bf = np.zeros((individuals,jumlah_item,jenis_pekerja))
biaya_akhir = np.zeros((individuals,jumlah_item))
df = np.zeros((individuals,jumlah_item))
cs = np.zeros(individuals)
```

Lampiran 3. Program *Python* untuk *Constraint Programming* (Lanjutan)

```
# Loop through each row (individual)
for i in range(individuals):
    for j in range(jumlah_item):
        for k in range(jenis_pekerja):
            #Definisi Jumlah Penambahan Pekerja
            penambahan_pekerja[i,j,k]=math.ceil
(population[i,j]*pekerja_awal[j,k])
            pekerja_akhir[i,j,k]=
penambahan_pekerja[i,j,k]+pekerja_awal[j,k]
            ohb[i,j,k]=oh[j,k]*pekerja_akhir[i,j,k]/pekerja_awal[j,k]
            bf[i,j,k]=bph[k]*vol[j]*ohb[i,j,k]
            biaya_akhir[i,j]=sum(bf[i,j,:])
            penambahan_pekerja_peritem[i,j]=
sum(penambahan_pekerja[i,j,:])
            df[i,j]=
math.ceil(da[j]*pekerja_awal_peritem[j]/(pekerja_awal_peritem[j]+p
enambahan_pekerja_peritem[i,j]))
    a = pd.DataFrame([df[i]]).transpose()
    b = a.groupby(urutan)
    c = 0
    for l in range(jumlah_urutan+1):
        c = c + max(b.get_group(l)[0])
    total_durasi_akhir = c
    cs[i] = sum(biaya_akhir[i]-biaya_awal)/(total_durasi_awal-
total_durasi_akhir)
    if total_durasi_awal-total_durasi_akhir < 30: cs[i] = np.inf

best_score = np.min(cs.astype(float))
index=[i for i, j in enumerate(cs) if j == best_score]
populasi=population[index]
populasi
print("--- %s seconds ---" % (time.time() - start_time))
```

Lampiran 3. Program *Python* untuk *Constraint Programming* (Lanjutan)

```
#Menghitung Cost Slope
def place_value(number):
    return ("{:,}".format(number))
place_value(best_score)

#Menampilkan Persentase Penambahan Pekerja
population[index[0]]

#Menampilkan Tabel Penambahan Pekerja
a = pd.DataFrame([df[index[0]]]).transpose()
a.rename(columns={0:'DURASI_AKHIR'}, inplace=True)
a

#Menampilkan Jumlah Hari Setelah Dilakukan Optimasi
a.rename(columns={0:'DURASI_AKHIR'}, inplace=True)
b = a.groupby(urutan)
c = 0
for l in range(jumlah_urutan+1):
    c = c + max(b.get_group(l).DURASI_AKHIR)
total_durasi_akhir = c
total_durasi_akhir
```

Lampiran 4. Program *Python* untuk *Genetic Algorithm*

```
data=pd.read_csv(r'C:/Users/Nafis/Desktop/TA_KERJA/TA_NAFIS.
csv', sep=';')

#Definisi BPH, Volume, Durasi Awal, Orang Harian dan Biaya
Awal
bph = data.BPH.T.dropna(axis=0)
vol = data.VOL.dropna(axis=0)
da = data.DURASI_AWAL.dropna(axis=0)
da_total = sum(da)
urutan = data.Urutan.dropna(axis=0)
jumlah_urutan = max(urutan)
oh = data.drop(data.columns[0:5], axis=1).dropna(axis=0)
oh = np.array(oh)
jumlah_item = pd.DataFrame(da).shape[0]
jenis_pekerja = pd.DataFrame(bph).shape[0]
pekerja_awal=np.zeros((jumlah_item,jenis_pekerja))
pekerja_awal_peritem=np.zeros(jumlah_item)
ba=np.zeros((jumlah_item,jenis_pekerja))
biaya_awal=np.zeros(jumlah_item)
for i in range(jumlah_item):
    for j in range(jenis_pekerja):
        # Change the required number of zeros to ones
        pekerja_awal[i,j] = math.ceil(oh[i,j]*vol[i]/da[i])
        pekerja_awal_peritem[i] = sum(pekerja_awal[i,:])
        ba[i,j] = bph[j]*vol[i]*oh[i,j]
        biaya_awal[i] = sum(ba[i,:])

#Bangkitan Populasi Awal
def create_starting_population(individuals):
    # Set up an initial array of all zeros
    population = np.zeros((individuals, jumlah_item))
    penambahan_pekerja =
np.zeros((individuals,jumlah_item,jenis_pekerja))
    penambahan_pekerja_peritem =
np.zeros((individuals,jumlah_item))
    pekerja_akhir = np.zeros((individuals,jumlah_item,jenis_pekerja))
    ohb = np.zeros((individuals,jumlah_item,jenis_pekerja))
```

Lampiran 4. Program *Python* untuk *Genetic Algorithm* (Lanjutan)

```
    bf = np.zeros((individuals,jumlah_item,jenis_pekerja))
    biaya_akhir = np.zeros((individuals,jumlah_item))
    df = np.zeros((individuals,jumlah_item))
    cs = np.zeros(individuals)

    # Loop through each row (individual)
    for i in range(individuals):
        df[i] = da_total
        while da_total - sum(df[i]) < 30:
            for j in range(jumlah_item):
                    # Change the required number of zeros to ones
                population[i,j] = round(random.uniform(0,40/100),2)
                            for k in range(jenis_pekerja):
                    #Definisi Jumlah Penambahan Pekerja
                    penambahan_pekerja[i,j,k] =
math.ceil(population[i,j]*pekerja_awal[j,k])
                    pekerja_akhir[i,j,k] =
penambahan_pekerja[i,j,k]+pekerja_awal[j,k]
                    ohb[i,j,k]=oh[j,k]*pekerja_akhir[i,j,k]/pekerja_awal[j,k]
                    bf[i,j,k]=bph[k]*vol[j]*ohb[i,j,k]
                    biaya_akhir[i,j]=sum(bf[i,j,:])
                    penambahan_pekerja_peritem[i,j] =
sum(penambahan_pekerja[i,j,:])
                    df[i,j] =
math.ceil(da[j]*pekerja_awal_peritem[j]/(pekerja_awal_peritem[j]+p
enambahan_pekerja_peritem[i,j]))
        a = pd.DataFrame([df[i]]).transpose()
        b = a.groupby(urutan)
        c = 0
        for l in range(jumlah_urutan+1):
            c = c + max(b.get_group(l)[0])
        total_durasi_akhir = c
        cs[i] = sum(biaya_akhir[i]-biaya_awal)/(total_durasi_awal-
total_durasi_akhir)

    return population.astype(float)
```

Lampiran 4. Program *Python* untuk *Genetic Algorithm* (Lanjutan)

```
#Definisi Menghitung Fitness
def calculate_fitness(population):
    individuals = len(population)
    fitness_scores = np.zeros(individuals)

    # Set up an initial array of all zeros
    penambahan_pekerja =
np.zeros((individuals,jumlah_item,jenis_pekerja))
    penambahan_pekerja_peritem =
np.zeros((individuals,jumlah_item))
    pekerja_akhir = np.zeros((individuals,jumlah_item,jenis_pekerja))
    ohb = np.zeros((individuals,jumlah_item,jenis_pekerja))
    bf = np.zeros((individuals,jumlah_item,jenis_pekerja))
    biaya_akhir = np.zeros((individuals,jumlah_item))
    df = np.zeros((individuals,jumlah_item))
    cs = np.zeros(individuals)

    # Loop through each row (individual)
    for i in range(individuals):
        for j in range(jumlah_item):
            for k in range(jenis_pekerja):

                #Definisi Jumlah Penambahan Pekerja
penambahan_pekerja[i,j,k]=math.ceil(population[i,j]*pekerja_awal[j,
k])

pekerja_akhir[i,j,k]=penambahan_pekerja[i,j,k]+pekerja_awal[j,k]
                ohb[i,j,k]=oh[j,k]*pekerja_akhir[i,j,k]/pekerja_awal[j,k]
                bf[i,j,k]=bph[k]*vol[j]*ohb[i,j,k]
                biaya_akhir[i,j]=sum(bf[i,j,:])

penambahan_pekerja_peritem[i,j]=sum(penambahan_pekerja[i,j,:])

df[i,j]=math.ceil(da[j]*pekerja_awal_peritem[j]/(pekerja_awal_perite
m[j]+penambahan_pekerja_peritem[i,j]))
            a = pd.DataFrame([df[i]]).transpose()
```

Lampiran 4. Program *Python* untuk *Genetic Algorithm* (Lanjutan)

```
            a.rename(columns={0:'DURASI_AKHIR'}, inplace=True)
            b = a.groupby(urutan)
            c = 0
            for l in range(jumlah_urutan+1):
                c = c + max(b.get_group(l).DURASI_AKHIR)
            total_durasi_akhir = c
            cs[i] = sum(biaya_akhir[i]-biaya_awal)/(total_durasi_awal-
total_durasi_akhir)
            if total_durasi_awal-total_durasi_akhir < 30: cs[i] = np.inf

        # Sum number of genes that are identical to the reference
            if population[i,j] < 0 or population[i,j] > 0.4:
                fitness_scores[i] = 0
            else:
                fitness_scores[i] = 1/(1+cs[i])

    return fitness_scores.astype(float)

#Definisi Seleksi
def select_individual_by_tournament(population, scores):
    # Get population size
    population_size = len(scores)

    # Pick individuals for tournament
    fighter_1 = 0
    fighter_2 = 0
    while fighter_1 == fighter_2:
        fighter_1 = random.randint(0, population_size-1)
        fighter_2 = random.randint(0, population_size-1)

    # Get fitness score for each
    fighter_1_fitness = scores[fighter_1]
    fighter_2_fitness = scores[fighter_2]

    # Identify undividual with highest fitness
    # Fighter 1 will win if score are equal
```

Lampiran 4. Program *Python* untuk *Genetic Algorithm* (Lanjutan)

```
    if fighter_1_fitness >= fighter_2_fitness:
        winner = fighter_1
    else:
        winner = fighter_2

    # Return the chromsome of the winner
    return population[winner, :]

#Definisi Crossover
def breed_by_crossover(parent_1, parent_2):
    # Get length of chromosome
    chromosome_length = len(parent_1)

    # Pick crossover point, avoding ends of chromsome
    crossover_point = random.randint(1,chromosome_length-1)

    # Create children. np.hstack joins two arrays
    child_1 = np.hstack((parent_1[0:crossover_point],
                        parent_2[crossover_point:]))
    child_2 = np.hstack((parent_2[0:crossover_point],
                        parent_1[crossover_point:]))
    # Return children
    return child_1, child_2

#Definisi Mutasi
def randomly_mutate_population(individu, mutation_probability,
mutation_rate) :
    # Apply random mutation
    random_mutation_array = random.random()
    random_mutation_boolean = \
        random_mutation_array <= mutation_probability
```

Lampiran 4. Program *Python* untuk *Genetic Algorithm* (Lanjutan)

```
    if random_mutation_boolean :
      gen_change_total = int(len(individu) * mutation_rate)
      for i in range(gen_change_total) :
         gen_index_selected = random.randint(0,len(individu) - 1 )
         nilai_geser = round(random.uniform(0,40/100),2)
         flag_loop = True
         while(flag_loop) :
            nilai_geser = round(random.uniform(0,40/100),2)
            if individu[gen_index_selected] + nilai_geser <= 0.4 and
individu[gen_index_selected] - nilai_geser >= 0 :
               flag_loop = False
               flag_probabiity = bool(random.getrandbits(1))
               if flag_probabiity :
                  individu[gen_index_selected] += nilai_geser
               else :
                  individu[gen_index_selected] -= nilai_geser

   # Return mutation population
   return individu

#Eksekusi pada Kasus
start_time = time.time()
# Set general parameters
chromosome_length = jumlah_item
population_size = 20
maximum_generation = 100
best_score_progress = [] # Tracks progress

# (this is used just to illustrate GAs)
# Create starting population
population = create_starting_population(population_size)

# Display best score in starting population
scores = calculate_fitness(population)
best_score = np.max(scores.astype(float))
```

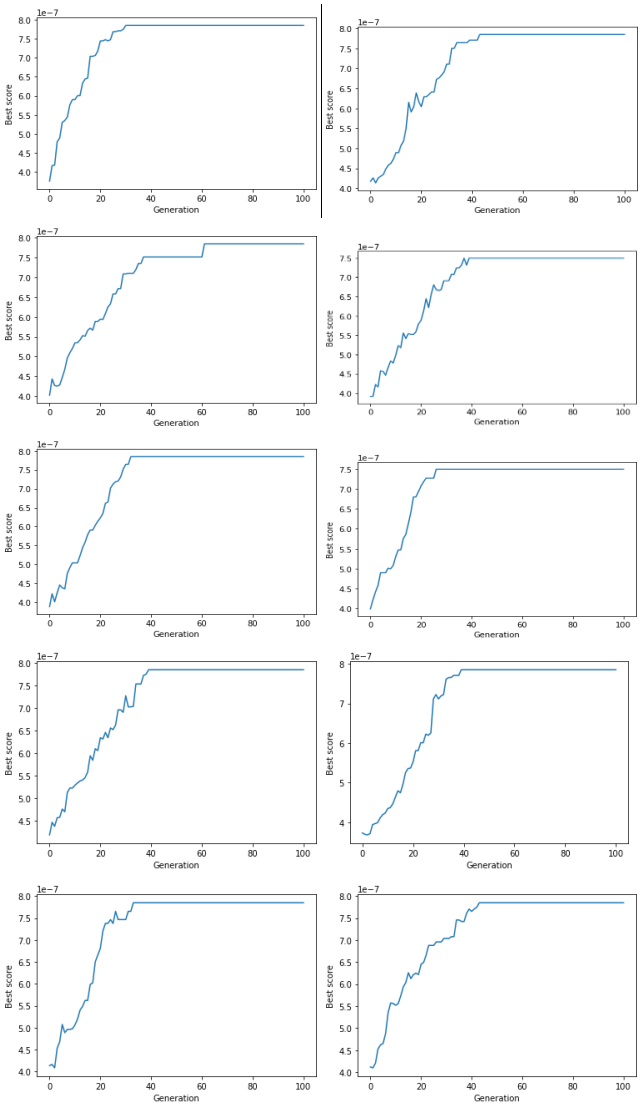
Lampiran 4. Program *Python* untuk *Genetic Algorithm* (Lanjutan)

```
# Add starting best score to progress tracker
best_score_progress.append(best_score)

# Now we'll go through the generations of genetic algorithm
for generation in range(maximum_generation):
    # Create an empty list for new population
    new_population = []

    # Create new popualtion generating two children at a time
    for i in range(int(population_size/2)):
        parent_1 = select_individual_by_tournament(population, scores)
        parent_2 = select_individual_by_tournament(population, scores)
        child_1, child_2 = breed_by_crossover(parent_1, parent_2)
        new_population.append(child_1)
        new_population.append(child_2)

    # Replace the old population with the new one
    population = np.array(new_population)

    # Apply mutation
    mutation_probability = 0.3
    mutation_rate = 0.3
    index = 0
    for individu in population:
        individu = randomly_mutate_population(individu,
mutation_probability, mutation_rate)
        population[index] = np.round(individu,2)
        index += 1

    # Score best solution, and add to tracker
    scores = calculate_fitness(population)
    best_score = np.max(scores.astype(float))
    best_score_progress.append(best_score.astype(float))

# Plot progress
%matplotlib inline
```

Lampiran 4. Program *Python* untuk *Genetic Algorithm* (Lanjutan)

```
plt.plot(best_score_progress)
plt.xlabel('Generation')
plt.ylabel('Best score')
plt.show()
print("--- %s seconds ---" % (time.time() - start_time))

#Menghitung Cost Slope
index=[i for i, j in enumerate(scores) if j == best_score]
populasi=population[index[0]]
penambahan_pekerja = np.zeros((jumlah_item,jenis_pekerja))
pekerja_akhir = np.zeros((jumlah_item,jenis_pekerja))

for j in range(jumlah_item):
    for k in range(jenis_pekerja):
        #Definisi Jumlah Penambahan Pekerja

penambahan_pekerja[j,k]=math.ceil(populasi[j]*pekerja_awal[j,k])
        pekerja_akhir[j,k]=penambahan_pekerja[j,k]+pekerja_awal[j,k]

def place_value(number):
    return ("{:,}".format(number))
cost_slope=place_value(1/best_score-1)
cost_slope

#Menampilkan Tabel Penambahan Pekerja
penambahan_pekerja=pd.DataFrame(penambahan_pekerja.astype(int
))
for i in range(jumlah_item):
    penambahan_pekerja.rename(index={i: data.kegiatan[i]})
penambahan_pekerja

#Menampilkan Persentase Penambahan Pekerja
persentase_penambahan=pd.DataFrame(populasi*100)
persentase_penambahan
```

Lampiran 5. Grafik *Fitness* Terbaik setiap Generasi pada 10 kali Percobaan Metode *Genetic Algorithm*



Lampiran 6. Catatan Lama Waktu *Running* pada 10 kali Percobaan Metode *Genetic Algorithm*

Percobaan Ke-	Lama Waktu (detik)
1	197.8563895225525
2	231.0607988834381
3	215.53475499153137
4	198.79236221313477
5	211.6499252319336
6	275.1132779121399
7	230.9415898323059
8	237.74091935157776
9	244.48363614082336
10	248.34005689620972

Lampiran 7. Surat Keterangan Pengambilan Data

CV. CREMONA TEKNIK CONSULTANT
PLANNING AND SUPERVISI
Alamat : Jl. Gamelan No.5 RT.34 Telp. 0541-7761682 / 081229003535 Fax. 0541-738495 Samarinda – Kalimantan Timur

Nomor :2/CTC-01/V/2020
Perihal : Izin memperoleh data untuk Tugas Akhir

Samarinda, 5 Mei 2020

Yth,
Kepala Departemen Statistika
Institut Teknologi Sepuluh Nopember Fakultas Sains Dan Analitika Data
Dr. Dra. Kartika Fithriasari,M.Si.

Dengan Hormat,

Yang bertanda tangan dibawah ini :
Nama : ZONY YULFADLI,S.T.,M.T
Jabatan : Direktur

Menerangkan bahwa,

Nama	: MOCH . ABDILLAH NAFIS
NRP	: 06211640000062
Program Studi	: Sarjana (S1)
Judul Tugas Akhir	: Perbandingan Hasil Optimalisasi Peningkatan Kinerja Biaya dan Waktu Proyek Bangunan Setelah Dilakukan Crashing Project Menggunakan Program Linier , Genetic Algoritma dan Modified Genetic Algoritma

Telah kami setujui untuk melaksanakan proses pengambilan data Rencana Anggaran Biaya.

Demikian surat ini kami sampaikan untuk dapat dipergunakan seperlunya.

DIREKTUR
CV. CREMONA TEKNIK CONSULTANT

CV. CREMONA TEKNIK CONSULTANT
PLANNING AND SUPERVISI CONSULTANT
ZONY YULFADLI,S.T.,M.T.

Scanned by TapScanner

BIODATA PENULIS

Penulis dengan nama lengkap Moch. Abdillah Nafis merupakan anak pertama dari pasangan Moch. Adi Mulyono dan Siti Ulfah. Penulis lahir di Surabaya pada tanggal 18 Agustus 1999. Pendidikan formal penulis ditempuh di SDN Perak Utara 1/59, SD Negeri Pacarkeling 8, SMP Negeri 16 dan SMA Negeri 18 Surabaya. Selanjutnya, penulis melanjutkan pendidikan formal pada Program Studi Sarjana Departemen Statistika ITS. Penulis diterima di ITS pada tahun 2016 melalui jalur SBMPTN. Selama perkulihan, penulis aktif dalam organisasi dan kompetisi. Penulis aktif di *Statistics Computer Course* (SCC) Himpunan Mahasiswa Departemen Statistika ITS (HIMASTA-ITS) sebagai Staf PR pada periode 2017-2018 dan Manajer PR pada periode 2018-2019. Penulis juga aktif di Forum Studi Islam Statistika ITS (FORSIS-ITS) sebagai Staf Media pada periode 1438-1439 H dan sebagai ketua umum pada periode 1439-1440 H. Penulis juga pernah terlibat sebagai Sie Keamanan dan Perizinan Pekan Raya Statistika (PRS) pada tahun 2017 dan 2018. Selain aktif dalam organisasi, penulis juga mengikuti berbagai perlombaan di bidang statistika dan *data mining* skala nasional. Capaian prestasi penulis yaitu tercatat beberapa kali sebagai semifinalis, finalis dan grand finalis pada kompetisi-kompetisi tersebut. Penulis juga pernah mendapatkan kesempatan beberapa kali untuk menjadi surveyor dan analis data di beberapa instansi. Segala kritik dan saran akan diterima oleh penulis untuk perbaikan kedepannya. Jika ingin berdiskusi mengenai Tugas Akhir ini, maka dapat mengirimkan email ke manafis49@gmail.com atau melalui nomor 0853-3127-5749.